


# SAS® Forecast Server Procedures 13.2

## User's Guide



The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2014. *SAS® Forecast Server Procedures 13.2: User's Guide*. Cary, NC: SAS Institute Inc.

**SAS® Forecast Server Procedures 13.2: User's Guide**

Copyright © 2014, SAS Institute Inc., Cary, NC, USA

All rights reserved. Produced in the United States of America.

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

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

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

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

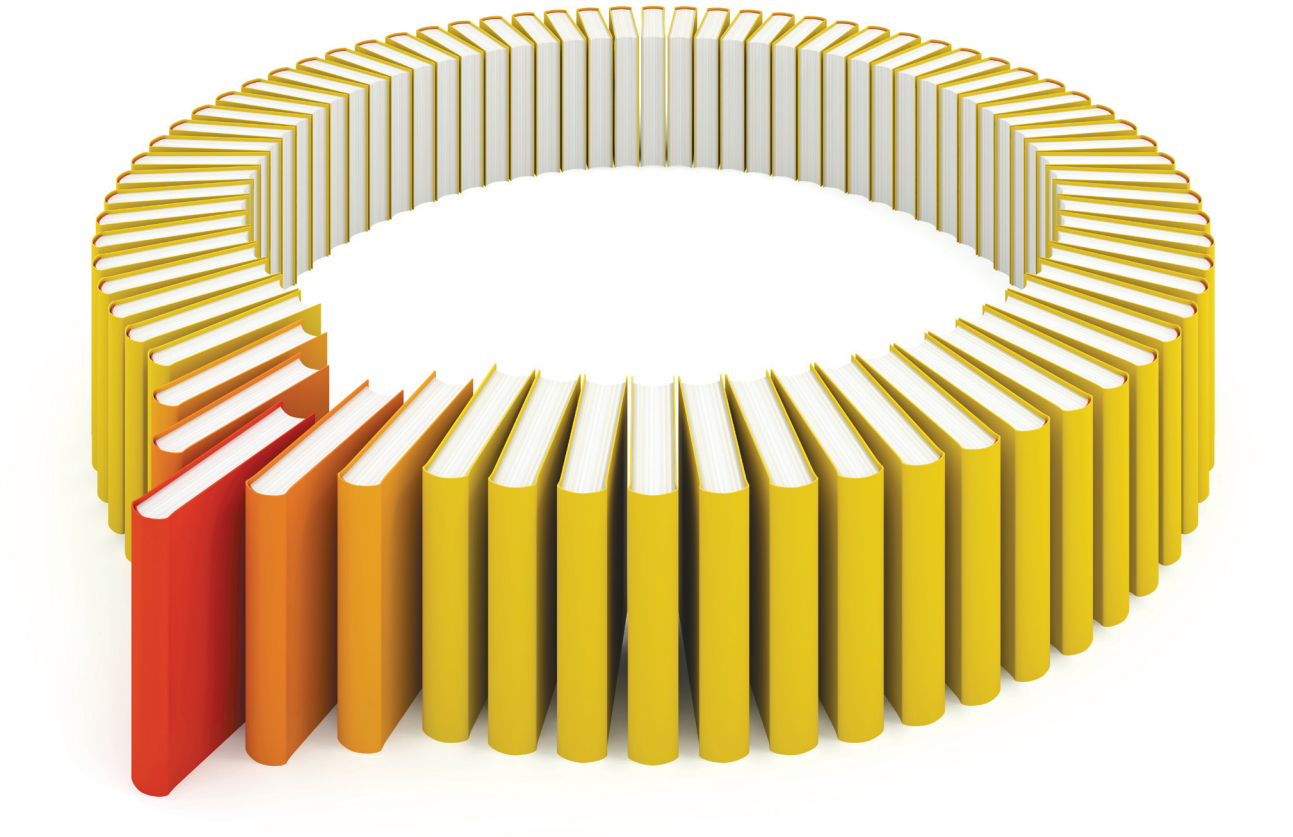
SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

August 2014

SAS provides a complete selection of books and electronic products to help customers use SAS® software to its fullest potential. For more information about our offerings, visit **support.sas.com/bookstore** or call 1-800-727-3228.

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

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



# Gain Greater Insight into Your SAS® Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.

support.sas.com/bookstore
*for additional books and resources.*

sas
THE POWER TO KNOW®

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies. © 2013 SAS Institute Inc. All rights reserved. S107969US.0613

# Contents

**I General Information** **1**
Chapter 1. What's New in SAS Forecast Server Procedures 13.2 . . . . . 3
Chapter 2. Introduction . . . . . 5

**II Procedure Reference** **23**
Chapter 3. The HPF Procedure . . . . . 25
Chapter 4. The HPFARIMASPEC Procedure . . . . . 75
Chapter 5. The HPFDIAGNOSE Procedure . . . . . 89
Chapter 6. The HPFDIAGSPEC Procedure . . . . . 157
Chapter 7. The HPFENGINE Procedure . . . . . 181
Chapter 8. The HPFESMSPEC Procedure . . . . . 261
Chapter 9. The HPFEVENTS Procedure . . . . . 273
Chapter 10. The HPFEXMSPEC Procedure . . . . . 317
Chapter 11. The HPFIDMSPEC Procedure . . . . . 323
Chapter 12. The HPFORECAST Procedure . . . . . 335
Chapter 13. The HPFRECONCILE Procedure . . . . . 417
Chapter 14. The HPFREPOSITORY Procedure (Experimental) . . . . . 447
Chapter 15. The HPFSELECT Procedure . . . . . 459
Chapter 16. The HPFTEMPRECON Procedure . . . . . 489
Chapter 17. The HPFUCMSPEC Procedure . . . . . 521

**III Forecasting Details** **541**
Chapter 18. Forecasting Process Summary . . . . . 543
Chapter 19. Forecasting Process Details . . . . . 573
Chapter 20. Forecast Combination Computational Details . . . . . 607
Chapter 21. Forecast Model Selection Graph Details . . . . . 623
Chapter 22. Using Forecasting Model Score Files and DATA Step Functions . . . . . 639
Chapter 23. Using User-Defined Models . . . . . 647
Chapter 24. Using External Forecasts . . . . . 657
Chapter 25. Using Auxiliary Data Sets . . . . . 661

**Subject Index** **675**

**Syntax Index** **680**

# Acknowledgments

## Credits

### Documentation

| | |
|---|---|
| Editing | Anne Baxter, Ed Huddleston |
| Technical Review | Jennifer Beeman, Ed Blair, Ming-Chun Chang, Bruce Elsheimer, Wilma S. Jackson, Michael J. Leonard, Kevin Meyer, Rajesh Selukar, Donna E. Woodward |
| Documentation Production | Tim Arnold |

### Software

The procedures and functions in the SAS Forecast Server Procedures software were implemented by members of the Advanced Analytics Division. Program development includes design, programming, debugging, support, documentation, and technical review. In the following list, the names of the developers who currently support the procedure or function are presented first.

| | |
|---|---|
| HPF Procedure | Michael J. Leonard |
| HPFARIMASPEC | Rajesh Selukar, Ed Blair |
| HPFDIAGNOSE | Ed Blair, Youngjin Park, Michael J. Leonard, Rajesh Selukar |
| HPFENGINE | Ed Blair, Keith Crowe, Michael J. Leonard, Rajesh Selukar |
| HPFESMSPEC | Michael J. Leonard, Ed Blair, Keith Crowe |
| HPFEVENTS | Wilma S. Jackson, Michael J. Leonard |
| HPFEXMSPEC | Michael J. Leonard, Ed Blair, Keith Crowe |
| HPFIDMSPEC | Michael J. Leonard, Ed Blair, Keith Crowe |

| | |
|---|---|
| HPFORECAST | Ed Blair |
| HPFRECONCILE | Michele Trovero, Mahesh Joshi |
| HPFREPOSITORY | Ed Blair, Keith Crowe |
| HPFSELECT | Ed Blair, Keith Crowe, Michael J. Leonard |
| HPFTEMPRECON | Michele Trovero, Ed Blair |
| HPFUCMSPEC | Rajesh Selukar, Ed Blair |
| HPFSCSIG function | Ed Blair, Keith Crowe |
| HPFSCSUB function | Ed Blair, Keith Crowe, Rajesh Selukar |
| Testing | Jennifer Beeman, Ming-Chun Chang, Bruce Elsheimer, Michael J. Leonard |

## Technical Support

| | |
|---|---|
| Members | Gina Marie Mondello, Donna E. Woodward |

# Part I

# General Information

# Chapter 1
# What's New in SAS Forecast Server Procedures 13.2

## Overview

The SAS High-Performance Forecasting product has been renamed the SAS Forecast Server Procedures product. This represents a change in name only and does not affect the use of the software in any way. The SAS High-Performance Forecasting product was renamed the SAS Forecast Server Procedures product in order to distinguish its functionality from that of other SAS High-Performance Analytics products.

The information in this book applies to SAS Forecast Server Procedures 13.2. Each release provides new features while maintaining all the capabilities of previous releases.

## Enhancements

The forecast scoring facility (HPFSCSUB) enables you to specify options in the form of name-value pairs at the end the argument list. For more information, see Chapter 22, "Using Forecasting Model Score Files and DATA Step Functions."

# Chapter 2
# Introduction

**Contents**


Overview of SAS Forecast Server Procedures Software . . . . . **5**
Uses of SAS Forecast Server Procedures Software . . . . . 7
Contents of SAS Forecast Server Procedures Software . . . . . 7
About This Book . . . . . **10**
Chapter Organization . . . . . 10
Typographical Conventions . . . . . 11
Options Used in Examples . . . . . 11
Where to Turn for More Information . . . . . **13**
Accessing the SAS Forecast Server Procedures Sample Library . . . . . 13
Online Help System . . . . . 13
SAS Technical Support Services . . . . . 14
Related SAS Software . . . . . **14**
Base SAS Software . . . . . 14
SAS/GRAPH Software . . . . . 16
SAS/STAT Software . . . . . 17
SAS/IML Software . . . . . 18
SAS/INSIGHT Software . . . . . 18
SAS/OR Software . . . . . 19
SAS/QC Software . . . . . 20
Other Statistical Tools . . . . . 20
References . . . . . **21**


## Overview of SAS Forecast Server Procedures Software

SAS Forecast Server Procedures software provides a large-scale automatic forecasting system. The software provides for the automatic selection of time series models for use in forecasting time-stamped data.

Given a time-stamped data set, the software provides the following automatic forecasting process:

1. accumulates the time-stamped data to form a fixed-interval time series
2. diagnoses the time series by using time series analysis techniques
3. creates a list of candidate model specifications that are based on the diagnostics
4. fits each candidate model specification to the time series

5. generates forecasts for each candidate fitted model
6. uses a model selection criterion to select the most appropriate model specification based on either in-sample or holdout-sample forecast performance (fit statistic comparison)
7. refits the selected model specification to the entire range of the time series
8. creates a forecast score from the selected fitted model
9. generate forecasts from the forecast score
10. evaluates the forecast by using in-sample analysis

The software also provides for out-of-sample forecast performance analysis.

For time series data that do not have causal inputs (input variables or calendar events), the HPF procedure provides a single, relatively easy-to-use batch interface that supports the preceding automatic forecasting process. The HPF procedure uses exponential smoothing models and intermittent demand models in an automated way to extrapolate the time series. The HPF procedure is relatively simple to use and requires only one procedure call.

For time series data that either have or do not have causal inputs (input variables or calendar events or both), the software provides several procedures that provide a batch interface that supports the preceding automatic forecasting process with more complicated models. These procedures must be used in the proper sequence in order to get the desired results. Forecasting time series of this nature normally requires more than one procedure call.

Input variables are recorded in the time-stamped data set. These input variables might or might not be incorporated in time series models that are used to generate forecasts.

Calendar events are specified by the HPFEVENTS procedure. These event definitions are used to generate discrete-valued indicator variables or dummy variables. These event definitions are stored in a SAS data set. These indicator variables might or might not be incorporated in time series models that are used to generate forecasts.

Given the specified calendar events and input variables, the HPFDIAGNOSE procedure diagnoses the time series and decides which, if any, of the calendar events or input variables are determined to be useful in forecasting the time series. The HPFDIAGNOSE procedure automatically generates candidate model specifications and a model selection list by using time series analysis techniques. These model specifications and model selection lists can then be used to automatically generate forecasts.

You can specify model specifications by using one of the following model specification procedures:

- The HPFARIMASPEC procedure enables you to specify one of the family of autoregressive integrated moving average with exogenous inputs models.
- The HPFESMSPEC procedure enables you to specify one of the family of exponential smoothing models.
- The HPFEXMSPEC procedure enables the forecast to be generated by an external source.
- The HPFIDMSPEC procedure enables you to specify one of the family of intermittent demand models.

- The HPFSELECT procedure enables you to specify a model selection list. The model selection list refers to one or more candidate model specifications and specifies how to choose the appropriate model for a given time series.
- The HPFUCMSPEC procedure enables you to specify one of the family of unobserved component models.

Regardless of whether the model specifications or model selection lists are specified or automatically generated, the HPFENGINE procedure uses these files to automatically select an appropriate forecasting model, estimate the model parameters, and forecast the time series.

Most of the computational effort that is associated with automatic forecasting consists of time series analysis, diagnostics, model selection, and parameter estimation. Forecast scoring files summarize the time series model's parameter estimates and the final states (historical time series information). These files can be used to quickly generate the forecasts that are required for the iterative nature of scenario analysis, stochastic optimization, and goal seeking computations. The HPFSCSUB function can be used to score time series information.

## Uses of SAS Forecast Server Procedures Software

SAS Forecast Server Procedures software provides tools for a wide variety of applications in business, government, and academia. Major uses of SAS Forecast Server Procedures include forecasting, forecast scoring, market response modeling, and time series data mining.

## Contents of SAS Forecast Server Procedures Software

### Procedures

SAS Forecast Server Procedures software includes the following procedures:

HPFARIMASPEC — The HPFARIMASPEC procedure creates an autoregressive integrated moving average (ARIMA) model specification file. The output of the procedure is an XML (extensible markup language) file that stores the intended ARIMA model specification. This XML specification file can be used to populate the model repository that the HPFENGINE procedure uses. Likewise, the XML files that are generated by the other model specification procedures in this section can also be used to populate the model repository that PROC HPFENGINE uses. For more information, see Chapter 4, "The HPFARIMASPEC Procedure."

HPFDIAGNOSE — The HPFDIAGNOSE procedure is an automatic modeling procedure that finds the best model among ARIMA models, exponential smoothing models, and unobserved component models.

The HPFDIAGNOSE procedure has the following functionality:

- intermittency test
- functional transformation test

- simple differencing and seasonal differencing test
- tentative simple ARMA order identification
- tentative seasonal ARMA order identification
- outlier detection
- significance test of events
- transfer functions identification
- intermittent demand model
- exponential smoothing model
- unobserved component model

The HPFDIAGNOSE procedure produces output that is compatible with HPFENGINE. As a result, the task of candidate model specification, model selection, and forecasting can be entirely automated by HPFDIAGNOSE and HPFENGINE in combination. For more information, see Chapter 5, "The HPFDIAGNOSE Procedure."

HPFDIAGSPEC

The HPFDIAGSPEC procedure creates a diagnostic preferences file to be used by PROC HPFORECAST. The output of the procedure is an XML file that stores information that is used to control the automatic time series model identification process for PROC HPFORECAST. For more information, see Chapter 6, "The HPFDIAGSPEC Procedure," and Chapter 12, "The HPFORECAST Procedure."

HPFENGINE

The HPFENGINE procedure provides large-scale automatic forecasting of transactional or time series data. The HPFENGINE procedure extends the foundation that is built by the HPF procedure, enabling you to determine the list of models over which automatic selection is performed.

The HPFENGINE procedure supports the following time series model families and forecasting methods:

- ARIMA
- unobserved component models
- exponential smoothing models
- intermittent demand models
- external models
- forecast combinations

Furthermore, you can completely customize the operation by defining your own code to generate forecasts.

For models that have inputs, the STOCHASTIC statement is especially helpful for automatically forecasting inputs that have no future values.

Also supported is the generation of a portable forecast score. The output of the SCORE statement is a file or catalog entry which, when used with the new function HPFSCSUB, can efficiently generate forecasts outside of the HPFENGINE procedure. For more information, see Chapter 7, "The HPFENGINE Procedure."

HPFESMSPEC — The HPFESMSPEC procedure creates an exponential smoothing models (ESM) specification file. The output of the procedure is an XML file that stores the intended ESM specification. For more information, see Chapter 8, "The HPFESMSPEC Procedure."

HPFEVENTS — The HPFEVENTS procedure provides a way to create and manage events that are associated with time series. The procedure can create events, read events from an events data set, write events to an events data set, and create dummy variables that are based on those events, if date information is provided.

A SAS event is used to model any incident that disrupts the normal flow of the process that generated the time series. Examples of commonly used events include natural disasters, retail promotions, strikes, advertising campaigns, policy changes, and data recording errors.

An event has a reference name, a date or dates that are associated with the event, and a set of qualifiers. The event exists separately from any time series; however, the event can be applied to one or more time series. When the event is applied to a time series, a dummy variable is generated that can be used to analyze the impact of the event on the time series.

For more information, see Chapter 9, "The HPFEVENTS Procedure."

HPFEXMSPEC — The HPFEXMSPEC procedure creates an external models (EXM) specification file. The output of the procedure is an XML file that stores the intended EXM specification. For more information, see Chapter 10, "The HPFEXMSPEC Procedure."

HPFORECAST — The HPFORECAST procedure is a high-performance procedure that forecasts time series data. It incorporates much of the automatic time series forecasting functionality that is provided in the HPFDIAGNOSE and HPFENGINE procedures. For more information, see Chapter 12, "The HPFORECAST Procedure."

HPFIDMSPEC — The HPFIDMSPEC procedure creates an intermittent demand models (IDM) specification file. The output of the procedure is an XML file that stores the intended IDM specification. For more information, see Chapter 11, "The HPFIDMSPEC Procedure."

HPFRECONCILE — The HPFRECONCILE procedure reconciles forecasts of time series data at two different levels of aggregation. Optionally, the HPFRECONCILE procedure can disaggregate forecasts from upper-level forecasts or aggregate forecasts from lower level-forecasts. Additionally, the procedure enables you to specify the direction and the method of reconciliation and equality constraints and bounds on the reconciled values at each point in time. For more information, see Chapter 13, "The HPFRECONCILE Procedure."

HPFSCSIG — The HPFSCSIG function generates a sample signature for subsequent use by the HPFSCSUB function. For more information, see Chapter 22, "Using Forecasting Model Score Files and DATA Step Functions," and Chapter 7, "The HPFENGINE Procedure."

HPFSCSUB — The HPFSCSUB function uses score files to produce forecasts outside of the HPFENGINE procedure. Because it is a function, it is particularly well suited for use within other SAS programming contexts (such as the DATA step) or

| | |
|---|---|
| | procedures that permit the specification of functions (such as the NLP procedure). The only input required is a reference to the score function, the horizon, and future values of any inputs. For more information, see Chapter 22, "Using Forecasting Model Score Files and DATA Step Functions," and Chapter 7, "The HPFENGINE Procedure." |
| HPFSELECT | The HPFSELECT procedure creates model lists. A model list contains references to candidate model specifications that are stored in the model repository. A model selection list selects the best model from its set of candidates based on the performance of each model's forecast. A model combination list combines the forecasts from its candidates to generate a combined forecast. The output of the procedure is an XML file that stores the intended model selection list. For more information, see Chapter 15, "The HPFSELECT Procedure." |
| HPFTEMPRECON | The HPFTEMPRECON procedure reconciles forecasts of time series data that are performed at two different frequencies. A low-frequency forecast serves as a benchmark that is used to perform adjustments to a high-frequency forecast. Constraints are imposed so that the adjusted high-frequency forecasts sum to the corresponding low-frequency forecasts over encompassing time periods. For more information, see Chapter 16, "The HPFTEMPRECON Procedure." |
| HPFUCMSPEC | The HPFUCMSPEC procedure creates an unobserved component models (UCM) specification file. The output of the procedure is an XML file that stores the intended UCM specification. For more information, see Chapter 17, "The HPFUCMSPEC Procedure." |

# About This Book

This book is a user's guide to SAS Forecast Server Procedures software. Because this software is a part of the SAS System, this book assumes that you are familiar with Base SAS software and that you have the books *SAS Language: Reference* and *SAS Procedures Guide* available for reference. It also assumes that you are familiar with SAS data sets, the SAS DATA step, and basic SAS procedures such as the PRINT procedure and the SORT procedure.

## Chapter Organization

The new features added to the software since the previous publication of this user's guide are summarized in Chapter 1, "What's New in SAS Forecast Server Procedures 13.2." If you have used the SAS Forecast Server Procedures software in the past, you might want to skim this chapter to see what is new.

Following the brief "What's New" chapter, this book is divided into three major parts:

- Part One contains general information to aid you in working with SAS Forecast Server Procedures software.

  The current chapter provides an overview of this software and summarizes related SAS publications, products, and services.

- Part Two, the "Procedure Reference," contains the chapters that explain the procedures that make up SAS Forecast Server Procedures software. The chapters that document each of the procedures appear in alphabetical order by procedure name and are organized as follows:

  1. Each chapter begins with an "Overview" section, which briefly describes the procedure.
  2. The "Getting Started" section provides a brief tutorial introduction about how to use the procedure.
  3. The "Syntax" section is a reference to the SAS statements and options that control the procedure.
  4. The "Details" section discusses various technical details.
  5. The "Examples" section contains additional examples of the use of the procedure.
  6. The "References" section contains technical references on methodology.

- Part Three provides a summary of and computational details about the SAS Forecast Server Procedures software.

## Typographical Conventions

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

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

## Options Used in Examples

### Output of Examples

For each example, the procedure output is numbered consecutively starting with 1, and each output is given a title. Each page of output produced by a procedure is enclosed in a box.

Most of the output shown in this book is produced with the following SAS System options:

```
options linesize=80 pagesize=200 nonumber nodate;
```

The template STATDOC.TPL is used to create the HTML output that appears in the online (CD) version. A style template controls stylistic HTML elements such as colors, fonts, and presentation attributes. The style template is specified in the ODS HTML statement as follows:

```
ODS HTML style=statdoc;
```

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

## Graphics Options

The examples that contain graphical output are created with a specific set of options and symbol statements. The statements you see in the examples creates the color graphics that appear in the online (CD) version of this book. A slightly different set of options and statements is used to create the black-and-white graphics that appear in the printed version of the book.

If you run the examples, you might get slightly different results. This may occur because not all graphic options for color devices translate directly to black-and-white output formats. For complete information about SAS/GRAPH software and graphics options, see *SAS/GRAPH Software: Reference*.

The following GOPTIONS statement is used to create the online (color) version of the graphic output.

```
filename GSASFILE '<file-specification>';

goptions reset=all
         gaccess=GSASFILE    gsfmode=replace
         fileonly
         transparency        dev = gif
         ftext = swiss       lfactor = 1
         htext = 4.0pct      htitle = 4.5pct
         hsize = 5.5in       vsize = 3.5in
         noborder            cback = white
         horigin = 0in       vorigin = 0in ;
```

The following GOPTIONS statement is used to create the black-and-white version of the graphic output, which appears in the printed version of the manual.

```
filename GSASFILE '<file-specification>';

goptions reset=all
         gaccess=GSASFILE    gsfmode=replace
         fileonly
         dev = pslepsf
         ftext = swiss       lfactor = 1
         htext = 3.0pct      htitle = 3.5pct
         hsize = 5.5in       vsize = 3.5in
         border              cback = white
         horigin = 0in       vorigin = 0in;
```

In most of the online examples, the plot symbols are specified as follows:

```
symbol1 value=dot color=white height=3.5pct;
```

The SYMBOL*n* statements used in online examples order the symbol colors as follows: white, yellow, cyan, green, orange, blue, and black.

In the examples that appear in the printed manual, symbol statements specify COLOR=BLACK and order the plot symbols as follows: dot, square, triangle, circle, plus, x, diamond, and star.

# Where to Turn for More Information

This section describes other sources of information about the SAS Forecast Server Procedures software.

## Accessing the SAS Forecast Server Procedures Sample Library

The sample library includes many examples that illustrate the use of this software, and it includes the examples that are used in this documentation. To access these sample programs, select **Help** from the menu and select **SAS Help and Documentation**. From the Contents list, choose **Learning to Use SAS** and then **Sample SAS Programs**.

## Online Help System

You can access online help information about SAS Forecast Server Procedures software in two ways, depending on whether you are using the SAS windowing environment in the command line mode or the menu mode.

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

If you are using the SAS windowing environment menus, you can make the following selections from the **Help** menu:

- **SAS Help and Documentation**
- **SAS Products** (on the **Contents** tab)
- **SAS Forecast Server Procedures**

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

## SAS Technical Support Services

As for all SAS products, the SAS Technical Support staff is available to respond to problems and answer technical questions regarding the use of SAS Forecast Server Procedures software.

# Related SAS Software

Many features not found in the SAS Forecast Server Procedures software are available in other parts of the SAS System. If you do not find something you need in this software, you might find it in one of the following SAS software products.

## Base SAS Software

The features provided by the SAS Forecast Server Procedures software are extensions to the features provided by Base SAS software. Many data management and reporting capabilities you will need are part of Base SAS software. Refer to *SAS Language: Reference* and the *SAS Procedures Guide* for documentation of Base SAS software.

The following sections summarize Base SAS software features of interest to users of SAS Forecast Server Procedures software. See Chapter 3, "Working with Time Series Data" (*SAS/ETS User's Guide*), for further discussion of some of these topics as they relate to time series data and SAS Forecast Server Procedures software.

### SAS DATA Step

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

### Base SAS Procedures

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

| | |
|---|---|
| CATALOG | for managing SAS catalogs |
| CHART | for printing charts and histograms |
| COMPARE | for comparing SAS data sets |
| CONTENTS | for displaying the contents of SAS data sets |
| COPY | for copying SAS data sets |
| CORR | for computing correlations |
| CPORT | for moving SAS data libraries between computer systems |
| DATASETS | for deleting or renaming SAS data sets |

| | |
|---|---|
| FREQ | for computing frequency crosstabulations |
| MEANS | for computing descriptive statistics and summarizing or collapsing data over cross sections |
| PLOT | for printing scatter plots |
| PRINT | for printing SAS data sets |
| RANK | for computing rankings or order statistics |
| SORT | for sorting SAS data sets |
| SQL | for processing SAS data sets with structured query language |
| STANDARD | for standardizing variables to a fixed mean and variance |
| TABULATE | for printing descriptive statistics in tabular format |
| TIMEPLOT | for plotting variables over time |
| TRANSPOSE | for transposing SAS data sets |
| UNIVARIATE | for computing descriptive statistics |

## Global Statements

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

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

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

The following Base SAS statements are useful with SAS Forecast Server Procedures procedures:

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

### SAS Functions

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

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

### Formats, Informats, and Time Intervals

Base SAS software provides formats to control the printing of data values, informats to read data values, and time intervals to define the frequency of time series.

## SAS/GRAPH Software

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

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

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

## SAS/STAT Software

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

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

- logistic regression
- censored regression
- principal component analysis
- structural equation models using covariance structure analysis
- factor analysis
- survival analysis
- discriminant analysis
- cluster analysis
- categorical data analysis, including log-linear and conditional logistic models
- general linear models
- mixed linear and nonlinear models
- generalized linear models
- response surface analysis
- kernel density estimation
- LOESS regression
- spline regression
- two-dimensional kriging
- multiple imputation for missing values

## SAS/IML Software

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

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

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

### Kalman Filtering and Time Series Analysis in SAS/IML

SAS/IML software includes a library for Kalman filtering and time series analysis. The library provides the following functions:

- generating univariate, multivariate, and fractional time series
- computing likelihood function of ARMA, VARMA, and ARFIMA models
- computing an autocovariance function of ARMA, VARMA, and ARFIMA models
- checking the stationarity of ARMA and VARMA models
- filtering and smoothing of time series models with the Kalman method
- fitting AR, periodic AR, time-varying coefficient AR, VAR, and ARFIMA models
- handling Bayesian seasonal adjustment model

## SAS/INSIGHT Software

SAS/INSIGHT software is a highly interactive tool for data analysis. You can explore data through a variety of interactive graphs that include bar charts, scatter plots, box plots, and three-dimensional rotating plots. You can examine distributions and perform parametric and nonparametric regression, analyze general linear models and generalized linear models, examine correlation matrixes, and perform principal component analyses. Any changes you make to your data show immediately in all graphs and analyses. You can also configure SAS/INSIGHT software to produce graphs and analyses tailored to the way you work.

SAS/INSIGHT software is an integral part of the SAS System. You can use it to examine output from a SAS procedure, and you can use any SAS procedure to analyze results from SAS/INSIGHT software.

SAS/INSIGHT software includes features for both displaying and analyzing data interactively. A data window displays a SAS data set as a table in which the columns of the table display variables and the rows display observations. Data windows provide data management features for editing, transforming, subsetting, and sorting data. A graph window displays different types of graphs: bar charts, scatter plots, box plots, and rotating plots. Graph windows provide interactive exploratory techniques such as data brushing and highlighting. Analysis windows display statistical analyses in the form of graphs and tables. Analysis windows include the following features:

- univariate statistics
- robust estimates
- density estimates
- cumulative distribution functions
- theoretical quantile-quantile plots
- multiple regression analysis with numerous diagnostic capabilities
- general linear models
- generalized linear models
- smoothing spline estimates
- kernel density estimates
- correlations
- principal components

SAS/INSIGHT software might be of interest to users of SAS Forecast Server Procedures software for interactive graphical viewing of data, editing data, exploratory data analysis, and checking distributional assumptions.

## SAS/OR Software

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

- solving transportation problems
- linear, integer, and mixed-integer programming
- nonlinear programming and optimization
- scheduling projects

- plotting Gantt charts
- drawing network diagrams
- solving optimal assignment problems
- network flow programming

SAS/OR software might be of interest to users of SAS Forecast Server Procedures software for its mathematical programming features. In particular, the NLP procedure in SAS/OR software solves nonlinear programming problems and can be used for constrained and unconstrained maximization of user-defined likelihood functions.

## SAS/QC Software

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

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

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

## Other Statistical Tools

Many other statistical tools are available in Base SAS, SAS/STAT, SAS/OR, SAS/QC, SAS/INSIGHT, and SAS/IML software. If you do not find something you need in SAS Forecast Server Procedures software, you might find it in SAS/STAT software or in Base SAS software. If you still do not find it, look in other SAS software products or contact the SAS Technical Support staff.

# References

Leonard, M. J. (2000), "Promotional Analysis and Forecasting for Demand Planning: A Practical Time Series Approach," .

Leonard, M. J. (2002), "Large-Scale Automatic Forecasting: Millions of Forecasts," .

Leonard, M. J. (2003), "Mining Transactional and Time Series Data," Paper presented at the International Symposium of Forecasting.

Leonard, M. J. (2004), "Large-Scale Automatic Forecasting with Inputs and Calendar Events," Paper presented at the International Symposium on Forecasting, Sydney.

# Part II

# Procedure Reference

# Chapter 3
# The HPF Procedure

## Contents


Overview: HPF Procedure . . . . . **26**
Getting Started: HPF Procedure . . . . . **27**
Syntax: HPF Procedure . . . . . **30**
Functional Summary . . . . . 30
PROC HPF Statement . . . . . 32
BY Statement . . . . . 35
FORECAST Statement . . . . . 36
ID Statement . . . . . 39
IDM Statement . . . . . 42
Smoothing Model Specification Options for IDM Statement . . . . . 44
Details: HPF Procedure . . . . . **47**
Accumulation . . . . . 47
Missing Value Interpretation . . . . . 49
Diagnostic Tests . . . . . 49
Model Selection . . . . . 49
Transformations . . . . . 50
Parameter Estimation . . . . . 50
Smoothing Model Parameter Specification Options . . . . . 50
Missing Value Modeling Issues . . . . . 50
Forecasting . . . . . 50
IDM Smoothing Model Forecast Bounds Options . . . . . 51
Inverse Transformations . . . . . 51
Statistics of Fit . . . . . 51
Forecast Summation . . . . . 51
Comparison to the Time Series Forecasting System . . . . . 51
Data Set Output . . . . . 52
OUT= Data Set . . . . . 52
OUTEST= Data Set . . . . . 52
OUTFOR= Data Set . . . . . 53
OUTPROCINFO= Data Set . . . . . 53
OUTSTAT= Data Set . . . . . 53
OUTSUM= Data Set . . . . . 55
OUTSEASON= Data Set . . . . . 56
OUTTREND= Data Set . . . . . 57
Printed Output . . . . . 58
ODS Table Names . . . . . 59

ODS Graphics . . . . . 60
Examples: HPF Procedure . . . . . **61**
Example 3.1: Automatic Forecasting of Time Series Data . . . . . 61
Example 3.2: Automatic Forecasting of Transactional Data . . . . . 63
Example 3.3: Specifying the Forecasting Model . . . . . 65
Example 3.4: Extending the Independent Variables for Multivariate Forecasts . . . . 66
Example 3.5: Forecasting Intermittent Time Series Data . . . . . 68
Example 3.6: Illustration of ODS Graphics . . . . . 70
References . . . . . **74**

# Overview: HPF Procedure

The HPF procedure provides a quick and automatic way to generate forecasts for many time series or transactional data in one step. The procedure can forecast millions of series at a time, with the series organized into separate variables or across BY groups.

- For typical time series, you can use the following smoothing models:
  - simple
  - double
  - linear
  - damped trend
  - seasonal (additive and multiplicative)
  - Winters method (additive and multiplicative)
- Additionally, transformed versions of these models are provided:
  - log
  - square root
  - logistic
  - Box-Cox
- For intermittent time series (series where a large number of values are zero-valued), you can use an intermittent demand model such as Croston's method and the average demand model.

All parameters associated with the forecast model are optimized based on the data. Optionally, the HPF procedure can select the appropriate smoothing model for you by using holdout sample analysis based on one of several model selection criteria.

The HPF procedure writes the following information to output data sets:

- time series extrapolated by the forecasts
- series summary statistics
- forecasts and confidence limits
- parameter estimates
- fit statistics

The HPF procedure optionally produces printed output for these results by using the Output Delivery System (ODS).

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

Additionally, the Time Series Forecasting System of SAS/ETS software can be used to interactively develop forecasting models, estimate the model parameters, evaluate the models' ability to forecast, and display these results graphically. See Chapter 45, "Overview of the Time Series Forecasting System" (*SAS/ETS User's Guide*), for details.

Also, the EXPAND procedure can be used for the frequency conversion and transformations of time series. See Chapter 15, "The EXPAND Procedure" (*SAS/ETS User's Guide*), for details.

## Getting Started: HPF Procedure

The HPF procedure is simple to use for someone who is new to forecasting, and yet at the same time it is powerful for the experienced professional forecaster who needs to generate a large number of forecasts automatically. It can provide results in output data sets or in other output formats by using the Output Delivery System (ODS). The following examples are more fully illustrated in the section "Examples: HPF Procedure" on page 61.

Given an input data set that contains numerous time series variables recorded at a specific frequency, the HPF procedure can automatically forecast the series as follows:

```
PROC HPF DATA=<input-data-set> OUT=<output-data-set>;
   ID <time-ID-variable> INTERVAL=<frequency>;
   FORECAST <time-series-variables>;
   RUN;
```

For example, suppose that the input data set SALES contains numerous sales data recorded monthly, the variable that represents time is DATE, and the forecasts are to be recorded in the output data set NEXTYEAR. The HPF procedure could be used as follows:

```
proc hpf data=sales out=nextyear;
   id date interval=month;
   forecast _ALL_;
run;
```

The preceding statements automatically select the best fitting model, generate forecasts for every numeric variable in the input data set (SALES) for the next twelve months, and store these forecasts in the output data set (NEXTYEAR). Other output data sets can be specified to store the parameter estimates, forecasts, statistics of fit, and summary data.

If you want to print the forecasts by using the Output Delivery System (ODS), then you need to add PRINT=FORECASTS:

```
proc hpf data=sales out=nextyear print=forecasts;
   id date interval=month;
   forecast _ALL_;
run;
```

Other results can be specified to output the parameter estimates, forecasts, statistics of fit, and summary data by using ODS.

The HPF procedure can forecast time series data, whose observations are equally spaced by a specific time interval (for example, monthly, weekly), and also transactional data, whose observations are not spaced with respect to any particular time interval.

Given an input data set that contains transactional variables not recorded at any specific frequency, the HPF procedure accumulates the data to a specific time interval and forecasts the accumulated series as follows:

```
PROC HPF DATA=<input-data-set> OUT=<output-data-set>;
   ID <time-ID-variable> INTERVAL=<frequency>
      ACCUMULATE=<accumulation>;
   FORECAST <time-series-variables>;
RUN;
```

For example, suppose that the input data set WEBSITES contains three variables (BOATS, CARS, PLANES) that are Internet data recorded on no particular time interval, and the variable that represents time is TIME, which records the time of the Web hit. The forecasts for the total daily values are to be recorded in the output data set NEXTWEEK. The HPF procedure could be used as follows:

```
proc hpf data=websites out=nextweek lead=7;
   id time interval=dtday accumulate=total;
   forecast boats cars planes;
run;
```

The preceding statements accumulate the data into a daily time series, automatically generate forecasts for the BOATS, CARS, and PLANES variables in the input data set (WEBSITES) for the next seven days, and store the forecasts in the output data set (NEXTWEEK).

The HPF procedure can specify a particular forecast model or select from several candidate models based on a selection criterion. The HPF procedure also supports transformed models and holdout sample analysis.

Using the previous WEBSITES example, suppose that you want to forecast the BOATS variable by using the best seasonal forecasting model that minimizes the mean absolute percent error (MAPE), forecast the CARS variable by using the best nonseasonal forecasting model that minimizes the mean square error (MSE)

by using holdout sample analysis on the last five days, and forecast the PLANES variable by using the log Winters method (additive). The HPF procedure could be used as follows:

```
proc hpf data=websites out=nextweek lead=7;
   id time interval=dtday accumulate=total;
   forecast boats   / model=bests criterion=mape;
   forecast cars    / model=bestn criterion=mse holdout=5;
   forecast planes  / model=addwinters transform=log;
run;
```

The preceding statements demonstrate how each variable in the input data set can be modeled differently and how several candidate models can be specified and selected based on holdout sample analysis or the entire range of data.

The HPF procedure is also useful in extending independent variables in regression or autoregression models where future values of the independent variable are needed to predict the dependent variable.

Using the WEBSITES example, suppose that you want to forecast the ENGINES variable by using the BOATS, CARS, and PLANES variable as regressor variables. Since future values of the BOATS, CARS, and PLANES variables are needed, the HPF procedure can be used to extend these variables in the future:

```
proc hpf data=websites out=nextweek lead=7;
   id time interval=dtday accumulate=total;
   forecast engines / model=none;
   forecast boats   / model=bests criterion=mape;
   forecast cars    / model=bestn crierion=mse holdout=5;
   forecast planes  / model=addwinters transform=log;
run;

proc autoreg data= nextweek;
   model engines = boats cars planes;
   output out=enginehits p=predicted;
run;
```

The preceding HPF procedure statements generate forecasts for BOATS, CARS, and PLANES in the input data set (WEBSITES) for the next seven days and extend the variable ENGINES with missing values. The output data set (NEXTWEEK) of the PROC HPF statement is used as an input data set for the PROC AUTOREG statement. The output data set of PROC AUTOREG contains the forecast of the variable ENGINE based on the regression model with the variables BOATS, CARS, and PLANES as regressors. See Chapter 8, "The AUTOREG Procedure" (*SAS/ETS User's Guide*), for details about autoregression.

The HPF procedure can also forecast intermittent time series (series where a large number of values are zero-valued). Typical time series forecasting techniques are less effective in forecasting intermittent time series.

For example, suppose that the input data set INVENTORY contains three variables (TIRES, HUBCAPS, LUGBOLTS) that are demand data recorded on no particular time interval, the variable that represents time is DATE, and the forecasts for the total weekly values are to be recorded in the output data set NEXTMONTH. The models requested are intermittent demand models, which can be specified as MODEL=IDM. Two intermittent demand models are compared, the Croston model and the average demand model. The HPF procedure could be used as follows:

```
proc hpf data=inventory out=nextmonth lead=4 print=forecasts;
   id date interval=week accumulate=total;
   forecast tires hubcaps lugbolts / model=idm;
run;
```

In the preceding example, the total demand for inventory items is accumulated on a weekly basis, and forecasts are generated that recommend future stocking levels.

# Syntax: HPF Procedure

The following statements are used with the HPF procedure.

**PROC HPF** *options* **;**
**BY** *variables* **;**
**FORECAST** *variable-list / options* **;**
**ID** *variable INTERVAL= interval options* **;**
**IDM** *options* **;**

## Functional Summary

Table 3.1 summarizes the statements and options that control the HPF procedure.

**Table 3.1** HPF Functional Summary

| Description | Statement | Option |
|---|---|---|
| **Statements** | | |
| Specifies BY-group processing | BY | |
| Specifies variables to forecast | FORECAST | |
| Specifies the time ID variable | ID | |
| Specifies intermittent demand model | IDM | |
| **Data Set Options** | | |
| Specifies the input data set | PROC HPF | DATA= |
| Specifies to output forecasts only | PROC HPF | NOOUTALL |
| Specifies the output data set | PROC HPF | OUT= |
| Specifies the parameter output data set | PROC HPF | OUTEST= |
| Specifies the forecast output data set | PROC HPF | OUTFOR= |
| Specifies the forecast procedure information output data set | PROC HPF | OUTPROCINFO= |
| Specifies the output data set for seasonal statistics | PROC HPF | OUTSEASON= |
| Specifies the statistics output data set | PROC HPF | OUTSTAT= |
| Specifies the summary output data set | PROC HPF | OUTSUM= |
| Specifies trend statistics output data set | PROC HPF | OUTTREND= |
| Replaces actual values held back | FORECAST | REPLACEBACK |

| Description | Statement | Option |
|---|---|---|
| Replaces missing values | FORECAST | REPLACEMISSING |
| Specifies forecast value type to append for OUT= | FORECAST | USE= |
| **Accumulation and Seasonality Options** | | |
| Specifies the accumulation frequency | ID | INTERVAL= |
| Specifies the length of seasonal cycle | PROC HPF | SEASONALITY= |
| Specifies the interval alignment | ID | ALIGN= |
| Specifies that the time ID variable values are not sorted | ID | NOTSORTED |
| Specifies the starting time ID value | ID | START= |
| Specifies the ending time ID value | ID | END= |
| Specifies the accumulation statistic | ID, FORECAST | ACCUMULATE= |
| Specifies missing value interpretation | ID, FORECAST | SETMISSING= |
| Specifies zero value interpretation | ID, FORECAST | ZEROMISS= |
| **Forecasting Horizon, Holdout, Holdback Options** | | |
| Specifies data to hold back | PROC HPF | BACK= |
| Specifies the forecast holdout sample size | FORECAST | HOLDOUT= |
| Specifies the forecast holdout sample percent | FORECAST | HOLDOUTPCT= |
| Specifies the forecast horizon or lead | PROC HPF | LEAD= |
| Specifies the horizon to start summation | PROC HPF | STARTSUM= |
| **Forecasting Model and Selection Options** | | |
| Specifies the confidence limit width | FORECAST | ALPHA= |
| Specifies the model selection criterion | FORECAST | CRITERION= |
| Specifies intermittency threshold | FORECAST | INTERMITTENT= |
| Specifies the forecast model | FORECAST | MODEL= |
| Specifies to use median forecats | FORECAST | MEDIAN |
| Specifies backcast initialization | FORECAST | NBACKCAST= |
| Specifies the seasonality test threshold | FORECAST | SEASONTEST= |
| Specifies the model transformation | FORECAST | TRANSFORM= |
| **Intermittent Demand Model Options** | | |
| Specifies the model for average demand | IDM | AVERAGE= |
| Specifies the base value | IDM | BASE= |
| Specifies the model for demand intervals | IDM | INTERVAL= |
| Specifies the model for demand sizes | IDM | SIZE= |
| **Printing and Plotting Control Options** | | |
| Specifies the time ID format | ID | FORMAT= |
| Specifies types of graphical output | PROC HPF | PLOT= |
| Specifies types of printed output | PROC HPF | PRINT= |
| Specifies that detailed printed output is desired | PROC HPF | PRINTDETAILS |

| Description | Statement | Option |
|---|---|---|
| **Miscellaneous Options** | | |
| Specifies that analysis variables are processed in sorted order | PROC HPF | SORTNAMES |
| Limits error and warning messages | PROC HPF | MAXERROR= |

## PROC HPF Statement

**PROC HPF** *options* **;**

The following options can be used in the PROC HPF statement.

**BACK=***n*

specifies the number of observations before the end of the data where the multistep forecasts are to begin. The default is BACK=0.

**DATA=***SAS-data-set*

names the SAS data set that contain the input data for the procedure to forecast. If the DATA= option is not specified, the most recently created SAS data set is used.

**LEAD=***n*

specifies the number of periods ahead to forecast (forecast lead or horizon). The default is LEAD=12.

The LEAD= value is relative to the last observation in the input data set and not to the last nonmissing observation of a particular series. Thus if a series has missing values at the end, the actual number of forecasts computed for that series will be greater than the LEAD= value.

**MAXERROR=***number*

limits the number of warning and error messages produced during the execution of the procedure to the specified value. The default is MAXERROR=50. This option is particularly useful in BY-group processing where it can be used to suppress the recurring messages.

**NOOUTALL**

specifies that only forecasts are written to the OUT= and OUTFOR= data sets. The NOOUTALL option includes only the final forecast observations in the output data sets, not the one-step forecasts for the data before the forecast period.

The OUT= and OUTFOR= data set will contain only the forecast results that start at the next period that follows the last observation and go to the forecast horizon specified by the LEAD= option.

**OUT=***SAS-data-set*

names the output data set to contain the forecasts of the variables that are specified in the subsequent FORECAST statements. If an ID variable is specified, it will also be included in the OUT= data set. The values are accumulated based on the ACCUMULATE= option, and forecasts are appended to these values based on the FORECAST statement USE= option. The OUT= data set is particularly useful in extending the independent variables when forecasting dependent series associated with regression and

autoregression models. If the OUT= option is not specified, a default output data set DATA*n* is created. If you do not want the OUT= data set created, then use OUT=_NULL_.

**OUTEST=***SAS-data-set*
names the output data set to contain the model parameter estimates and the associated test statistics and probability values. The OUTEST= data set is particularly useful for evaluating the significance of the model parameters and understanding the model dynamics.

**OUTFOR=***SAS-data-set*
names the output data set to contain the forecast time series components (actual, predicted, lower confidence limit, upper confidence limit, prediction error, and prediction standard error). The OUTFOR= data set is particularly useful for displaying the forecasts in tabular or graphical form.

**OUTPROCINFO=***SAS-data-set*
names the output data set to contain information in the SAS log, specifically the number of notes, errors, and warnings and the number of series processed, forecasts requested, and forecasts failed.

**OUTSEASON=***SAS-data-set*
names the output data set to contain the seasonal statistics. The statistics are computed for each season as specified by the ID statement INTERVAL= option or the SEASONALITY= option. The OUTSEASON= data set is particularly useful when analyzing transactional data for seasonal variations.

**OUTSTAT=***SAS-data-set*
names the output data set to contain the statistics of fit (or goodness-of-fit statistics). The OUTSTAT= data set is particularly useful for evaluating how well the model fits the series. The statistics of fit are based on the entire range of the time series regardless of whether the HOLDOUT= option is specified.

**OUTSUM=***SAS-data-set*
names the output data set to contain the summary statistics and the forecast summation. The summary statistics are based on the accumulated time series when the ACCUMULATE= or SETMISSING= options are specified. The forecast summations are based on the LEAD=, STARTSUM=, and USE= options. The OUTSUM= data set is particularly useful when forecasting large numbers of series and a summary of the results is needed.

**OUTTREND=***SAS-data-set*
names the output data set to contain the trend statistics. The statistics are computed for each time period as specified by the ID statement INTERVAL= option. The OUTTREND= data set is particularly useful when analyzing transactional data for trends.

**PLOT=***option* | **(***options* **)**
specifies the graphical output desired. By default, the HPF procedure produces no graphical output. The following printing options are available:

| | |
|---|---|
| ACF | plots prediction error autocorrelation function graphics. |
| ALL | is the same as specifying all of the PLOT= options. |
| BASIC | equivalent to specifying PLOT=(CORR ERRORS MODELFORECASTS). In the context of IDM models, the StockingLevelPlot is generated in place of the prediction error correlation panel plot. |

CORR plots the prediction error series graphics panel containing the ACF, IACF, PACF, and white noise probability plots. Note in the context of IDM models PLOT=CORR produces no additional output.

ERRORS plots prediction error time series graphics.

FORECASTS plots forecast graphics.

FORECASTSONLY plots the forecast in the forecast horzion only.

IACF plots prediction error inverse autocorrelation function graphics.

LEVELS plots smoothed level component graphics.

MODELFORECASTS plots the one-step ahead model forecast and its confidence bands in the historical period; the forecast and its confidence bands over the forecast horizon.

MODELS plots model graphics.

PACF plots prediction error partial autocorrelation function graphics.

PERIODOGRAM plots prediction error periodogram.

SEASONS plots smoothed seasonal component graphics.

SPECTRUM plots periodogram and smoothed periodogram of the prediction error series in a single graph. The SPECTRUM plot admits the specification of options to control some aspects of the generation of the smoothed periodogram. No options are required to use PLOT=SPECTRUM. To specify options use the following syntax:

**SPECTRUM=(***options***)**

Valid options for the SPECTUM plot include:

ALPHA=value specifies the significance level for upper and lower confidence limits about the smoothed periodogram estimates of spectral density. The default is ALPHA=0.4.

CENTER=YES|NO specifies whether mean adjustment is desired for the error series before computation of the smoothed periodogram estimates of spectral density. The default is NO.

TRENDS plots smoothed trend (slope) component graphics.

WN plots white noise graphics.

For example, PLOT=FORECASTS plots the forecasts for each series.

**PRINT=***option* | **(***options* **)**

specifies the printed output desired. By default, the HPF procedure produces no printed output. The following printing options are available:

ESTIMATES prints the results of parameter estimation (OUTEST= data set).

FORECASTS prints the forecasts (OUTFOR= data set).

PERFORMANCE prints the performance statistics for each forecast.

PERFORMANCESUMMARY prints the performance summary for each BY group.

PERFORMANCEOVERALL prints the performance summary for all of the BY groups.

| | |
|---|---|
| SEASONS | prints the seasonal statistics (OUTSEASON= data set). |
| STATISTICS | prints the statistics of fit (OUTSTAT= data set). |
| STATES | prints the backcast, initial, and final states. |
| SUMMARY | prints the summary statistics for the accumulated time series (OUTSUM= data set). |
| TRENDS | prints the trend statistics (OUTTREND= data set). |
| ALL | is the same as PRINT=(ESTIMATES FORECASTS STATISTICS SUMMARY). PRINT=(ALL TRENDS SEASONS) prints all of the options in the preceding list. |

For example, PRINT=FORECASTS prints the forecasts, PRINT=(ESTIMATES FORECASTS) prints the parameter estimates and the forecasts, and PRINT=ALL prints all of the preceding output.

The PRINT= option produces printed output for these results with the Output Delivery System (ODS). The PRINT= option produces results similar to the data sets listed in parenthesis for some of the options in the preceding list.

**PRINTDETAILS**
specifies that output requested with the PRINT= option be printed in greater detail.

**SEASONALITY=***number*
specifies the length of the seasonal cycle. For example, SEASONALITY=3 means that every group of three observations forms a seasonal cycle. The SEASONALITY= option is applicable only for seasonal forecasting models. By default, the length of the seasonal cycle is one (no seasonality) or the length implied by the INTERVAL= option specified in the ID statement. For example, INTERVAL=MONTH implies that the length of the seasonal cycle is 12.

**SORTNAMES**
specifies that the variables specified in the FORECAST statements be processed in sorted order.

**STARTSUM=***n*
specifies the starting forecast lead (or horizon) for which to begin summation of the forecasts specified by the LEAD= option. The STARTSUM= value must be less than the LEAD= value. The default is STARTSUM=1—that is, the sum from the one-step-ahead forecast to the multistep forecast specified by the LEAD= option.

The prediction standard errors of the summation of forecasts take into account the correlation between the multistep forecasts. The section "Details: HPF Procedure" on page 47 describes the STARTSUM= option in more detail.

## BY Statement

**BY** *variables* **;**

A BY statement can be used with PROC HPF to obtain separate analyses for groups of observations defined by the BY variables.

When a BY statement appears, the procedure expects the input data set to be sorted in order of the BY variables.

If your input data set is not sorted in ascending order, use one of the following alternatives:

- Sort the data by using the SORT procedure with a similar BY statement.
- Specify the NOTSORTED or DESCENDING option in the BY statement for the HPF procedure. The NOTSORTED option does not mean that the data are unsorted but rather that the data are arranged in groups (according to values of the BY variables) and that these groups are not necessarily in alphabetical or increasing numeric order.
- Create an index on the BY variables by using the DATASETS procedure.

For more information about the BY statement, see *SAS Language Reference: Concepts*. For more information about the DATASETS procedure, see the discussion in the *Base SAS Procedures Guide.*

## FORECAST Statement

**FORECAST** *variable-list / options* **;**

The FORECAST statement lists the numeric variables in the DATA= data set whose accumulated values represent time series to be modeled and forecast. The options specify which forecast model is to be used or how the forecast model is selected from several possible candidate models.

A data set variable can be specified in only one FORECAST statement. Any number of FORECAST statements can be used. The following options can be used with the FORECAST statement.

**ACCUMULATE=***option*

specifies how the data set observations are to be accumulated within each time period for the variables listed in the FORECAST statement. If the ACCUMULATE= option is not specified in the FORECAST statement, accumulation is determined by the ACCUMULATE= option of the ID statement. See the ID statement ACCUMULATE= option for more details.

**ALPHA=***number*

specifies the significance level to use in computing the confidence limits of the forecast. The ALPHA= value must be between 0 and 1. The default is ALPHA=0.05, which produces 95% confidence intervals.

**CRITERION=***option*

**SELECT=***option*

specifies the model selection criterion (statistic of fit) to be used to select from several candidate models. This option is often used in conjunction with the HOLDOUT= option. The CRITERION= option can also be specified as SELECT=. The default is CRITERION=RMSE.

**HOLDOUT=***n*

specifies the size of the holdout sample to be used for model selection. The holdout sample is a subset of actual time series that end at the last nonmissing observation. If the ACCUMULATE= option is specified, the holdout sample is based on the accumulated series. If the holdout sample is not specified, the full range of the actual time series is used for model selection.

For each candidate model specified, the holdout sample is excluded from the initial model fit and forecasts are made within the holdout sample time range. Then, for each candidate model specified, the statistic of fit specified by the CRITERION= option is computed by using only the observations in the holdout sample. Finally, the candidate model, which performs best in the holdout sample, based on this statistic, is selected to forecast the actual time series.

The HOLDOUT= option is used only to select the best forecasting model from a list of candidate models. After the best model is selected, the full range of the actual time series is used for subsequent model fitting and forecasting. It is possible that one model will outperform another model in the holdout sample but perform less well when the entire range of the actual series is used.

If the MODEL=BESTALL and HOLDOUT= options are used together, the last one hundred observations are used to determine whether the series is intermittent. If the series is determined not to be intermittent, holdout sample analysis is used to select the smoothing model.

**HOLDOUTPCT=***number*

specifies the size of the holdout sample as a percentage of the length of the time series. If HOLDOUT=5 and HOLDOUTPCT=10, the size of the holdout sample is $min(5, 0.1T)$ where $T$ is the length of the time series with the beginning and ending missing values removed. The default is 100 (100%).

**INTERMITTENT=***number*

specifies a number greater than one which is used to determine whether or not a time series is intermittent. If the average demand interval is greater than this number, then the series is assumed to be intermittent. This option is used with the MODEL=BESTALL option. The default is INTERMITTENT=1.25.

**MEDIAN**

specifies that the median forecast values are to be estimated. Forecasts can be based on the mean or median. By default the mean value is provided. If no transformation is applied to the actual series by using the TRANSFORM= option, the mean and median forecast values are identical.

**MODEL=***model-name*

specifies the forecasting model to be used to forecast the actual time series. A single model can be specified or a group of candidate models can be specified. If a group of models is specified, the model used to forecast the accumulated time series is selected based on the CRITERION= option and the HOLDOUT= option. The default is MODEL=BEST. The following forecasting models are provided:

| | |
|---|---|
| NONE | no forecast. The accumulated time series is appended with missing values in the OUT= data set. This option is particularly useful when the results stored in the OUT= data set are subsequently used in regression or autoregression analysis where forecasts of the independent variables are needed to forecast the dependent variable. |
| SIMPLE | simple (single) exponential smoothing |
| DOUBLE | double (Brown) exponential smoothing |
| LINEAR | linear (Holt) exponential smoothing |
| DAMPTREND | damped trend exponential smoothing |
| ADDSEASONAL\|SEASONAL | additive seasonal exponential smoothing |
| MULTSEASONAL | multiplicative seasonal exponential smoothing |
| WINTERS | Winters multiplicative Method |

| | |
|---|---|
| ADDWINTERS | Winters additive Method |
| BEST | best candidate smoothing model (SIMPLE, DOUBLE, LINEAR, DAMPTREND, ADDSEASONAL, WINTERS, ADDWINTERS) |
| BESTN | best candidate nonseasonal smoothing model (SIMPLE, DOUBLE, LINEAR, DAMPTREND) |
| BESTS | best candidate seasonal smoothing model (ADDSEASONAL, WINTERS, ADDWINTERS) |
| IDM\|CROSTON | intermittent demand model such as Croston's method or average demand model. An intermittent time series is one whose values are mostly zero. |
| BESTALL | best candidate model (IDM, BEST) |

The BEST, BESTN, and BESTS options specify a group of models by which the HOLDOUT= option and CRITERION= option are used to select the model used to forecast the accumulated time series based on holdout sample analysis. Transformed versions of the preceding smoothing models can be specified using the TRANSFORM= option.

The BESTALL option specifies that if the series is intermittent, an intermittent demand model such as Croston's method or average demand model (MODEL=IDM) is selected; otherwise, the best smoothing model is selected (MODEL=BEST). Intermittency is determined by the INTERMITTENT= option.

Chapter 19, "Forecasting Process Details," describes the preceding smoothing models and intermittent models in greater detail.

**NBACKCAST=***n*

specifies the number of observations used to initialize the backcast states. The default is the entire series.

**REPLACEBACK**

specifies that actual values excluded by the BACK= option be replaced with one-step-ahead forecasts in the OUT= data set.

**REPLACEMISSING**

specifies that embedded missing actual values be replaced with one-step-ahead forecasts in the OUT= data set.

**SEASONTEST=***option*

specifies the options related to the seasonality test. This option is used with the MODEL=BEST and MODEL=BESTALL options.

The following options are provided:

SEASONTEST=NONE no test

SEASONTEST=(SIGLEVEL=number) significance probability value

Series with strong seasonality have small test probabilities. SEASONTEST=(SIGLEVEL=0) always implies seasonality. SEASONTEST=(SIGLEVEL=1) always implies no seasonality. The default is SEASONTEST=(SIGLEVEL=0.01).

**SETMISSING=***option* | *number*
specifies how missing values (either actual or accumulated) are to be assigned in the accumulated time series for variables listed in the FORECAST statement. If the SETMISSING= option is not specified in the FORECAST statement, missing values are set based on the SETMISSING= option of the ID statement. See the ID statement SETMISSING= option for more details.

**TRANSFORM=***option*
specifies the time series transformation to be applied to the actual time series. The following transformations are provided:

| | |
|---|---|
| NONE | no transformation is applied. This option is the default. |
| LOG | logarithmic transformation |
| SQRT | square-root transformation |
| LOGISTIC | logistic transformation |
| BOXCOX(*n* ) | Box-Cox transformation with parameter number where number is between –5 and 5 |
| AUTO | automatically choose between NONE and LOG based on model selection criteria |

When the TRANSFORM= option is specified, the time series must be strictly positive. After the time series is transformed, the model parameters are estimated by using the transformed series. The forecasts of the transformed series are then computed, and finally the transformed series forecasts are inverse transformed. The inverse transform produces either mean or median forecasts depending on whether the MEDIAN option is specified.

The TRANSFORM= option is not applicable when MODEL=IDM is specified.

**USE=***option*
specifies which forecast values are appended to the actual values in the OUT= and OUTSUM= data sets. The following USE= options are provided:

| | |
|---|---|
| PREDICT | The predicted values are appended to the actual values. This option is the default. |
| LOWER | The lower confidence limit values are appended to the actual values. |
| UPPER | The upper confidence limit values are appended to the actual values. |

Thus, the USE= option enables the OUT= and OUTSUM= data sets to be used for worst-, best-, average-, and median-case decisions.

**ZEROMISS=***option*
specifies how beginning and/or ending zero values (either actual or accumulated) are interpreted in the accumulated time series for variables listed in the FORECAST statement. If the ZEROMISS= option is not specified in the FORECAST statement, missing values are set based on the ZEROMISS= option of the ID statement. See the ID statement ZEROMISS= option for more details.

## ID Statement

**ID** *variable INTERVAL= interval* < *options* > **;**

The ID statement names a numeric variable that identifies observations in the input and output data sets. The ID variable's values are assumed to be SAS date, time, or datetime values. In addition, the ID statement specifies the (desired) frequency associated with the actual time series. The ID statement options also specify how the observations are accumulated and how the time ID values are aligned to form the actual time series. The information specified affects all variables specified in subsequent FORECAST statements. If the ID statement is specified, the INTERVAL= option must also be specified. If an ID statement is not specified, the observation number, with respect to the BY group, is used as the time ID.

The following options can be used with the ID statement.

**ACCUMULATE=***option*

specifies how the data set observations are to be accumulated within each time period. The frequency (width of each time interval) is specified by the INTERVAL= option. The ID variable contains the time ID values. Each time ID variable value corresponds to a specific time period. The accumulated values form the actual time series, which is used in subsequent model fitting and forecasting.

The ACCUMULATE= option is particularly useful when there are zero or more than one input observations that coincide with a particular time period (for example, transactional data). The EXPAND procedure offers additional frequency conversions and transformations that can also be useful in creating a time series.

The following options determine how the observations are to be accumulated within each time period based on the ID variable and the frequency specified by the INTERVAL= option:

| | |
|---|---|
| NONE | No accumulation occurs; the ID variable values must be equally spaced with respect to the frequency. This is the default option. |
| TOTAL | Observations are accumulated based on the total sum of their values. |
| AVERAGE \| AVG | Observations are accumulated based on the average of their values. |
| MINIMUM \| MIN | Observations are accumulated based on the minimum of their values. |
| MEDIAN \| MED | Observations are accumulated based on the median of their values. |
| MAXIMUM \| MAX | Observations are accumulated based on the maximum of their values. |
| N | Observations are accumulated based on the number of nonmissing observations. |
| NMISS | Observations are accumulated based on the number of missing observations. |
| NOBS | Observations are accumulated based on the number of observations. |
| FIRST | Observations are accumulated based on the first of their values. |
| LAST | Observations are accumulated based on the last of their values. |
| STDDEV \| STD | Observations are accumulated based on the standard deviation of their values. |
| CSS | Observations are accumulated based on the corrected sum of squares of their values. |
| USS | Observations are accumulated based on the uncorrected sum of squares of their values. |

If the ACCUMULATE= option is specified, the SETMISSING= option is useful for specifying how accumulated missing values are treated. If missing values should be interpreted as zero, then SETMISSING=0 should be used. The section "Details: HPF Procedure" on page 47 describes accumulation in greater detail.

**ALIGN=***option*

controls the alignment of SAS dates used to identify output observations. The ALIGN= option accepts the following values: BEGINNING | BEG | B, MIDDLE | MID | M, and ENDING | END | E. BEGINNING is the default.

**END=***option*

specifies a SAS date, datetime, or time value that represents the end of the data. If the last time ID variable value is less than the END= value, the series is extended with missing values. If the last time ID variable value is greater than the END= value, the series is truncated. For example, END="&sysdate"D uses the automatic macro variable SYSDATE to extend or truncate the series to the current date. This option and the START= option can be used to ensure that data that are associated with each BY group contain the same number of observations.

**FORMAT=***format*

specifies the SAS format for the time ID values. If the FORMAT= option is not specified, the default format is implied from the INTERVAL= option.

**INTERVAL=***interval*

specifies the frequency of the input time series. For example, if the input data set consists of quarterly observations, then INTERVAL=QTR should be used. If the SEASONALITY= option is not specified, the length of the seasonal cycle is implied from the INTERVAL= option. For example, INTERVAL=QTR implies a seasonal cycle of length 4. If the ACCUMULATE= option is also specified, the INTERVAL= option determines the time periods for the accumulation of observations.

The basic intervals are YEAR, SEMIYEAR, QTR, MONTH, SEMIMONTH, TENDAY, WEEK, WEEKDAY, DAY, HOUR, MINUTE, SECOND. See Chapter 4, "Date Intervals, Formats, and Functions" (*SAS/ETS User's Guide*), for the intervals that can be specified.

**NOTSORTED**

specifies that the time ID values not be in sorted order. The HPF procedure sorts the data with respect to the time ID prior to analysis.

**SETMISSING=***option* | *number*

specifies how missing values (either actual or accumulated) are to be assigned in the accumulated time series. If a number is specified, missing values are set to number. If a missing value indicates an unknown value, this option should not be used. If a missing value indicates no value, a SETMISSING=0 should be used. You would typically use SETMISSING=0 for transactional data because no recorded data usually implies no activity. The following options can also be used to determine how missing values are assigned:

| | |
|---|---|
| MISSING | Missing values are set to missing. This is the default option. |
| AVERAGE \| AVG | Missing values are set to the accumulated average value. |
| MINIMUM \| MIN | Missing values are set to the accumulated minimum value. |
| MEDIAN \| MED | Missing values are set to the accumulated median value. |

| | |
|---|---|
| MAXIMUM \| MAX | Missing values are set to the accumulated maximum value. |
| FIRST | Missing values are set to the accumulated first nonmissing value. |
| LAST | Missing values are set to the accumulated last nonmissing value. |
| PREVIOUS \| PREV | Missing values are set to the previous accumulated nonmissing value. Missing values at the beginning of the accumulated series remain missing. |
| NEXT | Missing values are set to the next accumulated nonmissing value. Missing values at the end of the accumulated series remain missing. |

If SETMISSING=MISSING is specified and the MODEL= option specifies a smoothing model, the missing observations are smoothed over. If MODEL=IDM is specified, missing values are assumed to be periods of no demand—that is, SETMISSING=MISSING is equivalent to SETMISSING=0.

**START=***option*

specifies a SAS date, datetime, or time value that represents the beginning of the data. If the first time ID variable value is greater than the START= value, missing values are added at the beginning of the series. If the first time ID variable value is less than the START= value, the series is truncated. This option and the END= option can be used to ensure that data that are associated with each By group contain the same number of observations.

**ZEROMISS=***option*

specifies how beginning and/or ending zero values (either actual or accumulated) are interpreted in the accumulated time series. The following options can also be used to determine how beginning and/or ending zero values are assigned:

| | |
|---|---|
| NONE | Beginning and/or ending zeros unchanged. This is the default. |
| LEFT | Beginning zeros are set to missing. |
| RIGHT | Ending zeros are set to missing. |
| BOTH | Both beginning and ending zeros are set to missing. |

If the accumulated series is all missing and/or zero, the series is not changed.

## IDM Statement

**IDM** *options* **;**

The IDM statement is used to specify an intermittent demand model. An intermittent demand series can be analyzed in two ways: individually modeling both demand interval and size component or jointly modeling these components by using the average demand component (demand size divided by demand interval). The IDM statement specifies the two smoothing models to be used to forecast the demand interval component (INTERVAL= option) and the demand size component (SIZE= option) or specifies the single smoothing model to be used to forecast the average demand component (AVERAGE= option). Therefore, two smoothing models (INTERVAL= and SIZE= options) must be specified or one smoothing model (AVERAGE= option) must be specified. Only one statement can be specified.

The following examples illustrate typical uses of the IDM statement:

```
/* default specification */
idm;

/* demand interval model only specification */
idm interval=(transform=log);

/* demand size model only specification */
idm size=(method=linear);

/* Croston's Method */
idm interval=(method=simple)
    size    =(method=simple);

/* Log Croston's Method */
idm interval=(method=simple transform=log)
    size    =(method=simple transform=log);

/* average demand model specification */
idm average=(method=bestn);
```

The default specification uses both the INTERVAL= option and SIZE= option defaults for the decomposed (Croston's) demand model and the AVERAGE= option defaults for the average demand model.

The following example illustrates how to automatically choose the decomposed demand model by using MAPE as the model selection criterion:

```
idm interval=( method=simple transform=auto criterion=mape )
    size    =( method=simple transform=auto criterion=mape );
forecast sales / model=idm criterion=mape;
```

The preceding statements fit two forecast models (simple and log simple exponential smoothing) to both the demand interval and size components. The forecast model that results in the lowest in-sample MAPE for each component is used to forecast the component.

The following example illustrates how to automatically choose the average demand model by using MAPE as the model selection criterion:

```
idm average=(method=simple transform=auto criterion=mape);
forecast sales / model=idm;
```

The preceding statements fit two forecast models (simple and log simple exponential smoothing) to the average demand component. The forecast model that results in the lowest in-sample MAPE is used to forecast the component.

Combining the two preceding examples, the following example illustrates how to automatically choose between the decomposed demand model and the average demand model by using MAPE as the model selection criterion:

```
idm interval=(method=simple transform=auto criterion=mape)
    size    =(method=simple transform=auto criterion=mape)
    average =(method=simple transform=auto criterion=mape);
forecast sales / model=idm criterion=mape;
```

The preceding statements automatically select between the decomposed demand model and the average demand model as described previously. The forecast model that results in the lowest in-sample MAPE is used to forecast the series.

The following options can be specified in the IDM statement:

**AVERAGE=(***smoothing-model-options* **)**
specifies the smoothing model used to forecast the demand average component. See the section "Smoothing Model Specification Options for IDM Statement" on page 44.

**BASE=AUTO** | *number*
specifies the base value of the time series used to determine the demand series components. The demand series components are determined based on the departures from this base value. If a base value is specified, this value is used to determine the demand series components. If BASE=AUTO is specified, the time series properties are used to automatically adjust the time series. For the common definition of Croston's method use BASE=0 which defines departures based on zero. The default is BASE=0.

Given a time series $y_t$ and base value $b$, the time series is adjusted by the base value to create the base adjusted time series, $x_t = y_t - b$. Demands are assumed to occur when the base adjusted series is nonzero (or when the time series $y_t$ departs from the base value $b$).

When BASE=AUTO, the base value is automatically determined by the time series median, minimum and maximum values, and the INTERMITTENT= option value of the FORECAST statement.

**INTERVAL=(***smoothing-model-options* **)**
specifies the smoothing model used to forecast the demand interval component. See the section "Smoothing Model Specification Options for IDM Statement" on page 44.

**SIZE=(***smoothing-model-options* **)**
specifies the smoothing model used to forecast the demand size component. See the section "Smoothing Model Specification Options for IDM Statement" on page 44.

## Smoothing Model Specification Options for IDM Statement

The smoothing model options describe how to forecast the demand interval, size, and average demand components (INTERVAL= option, SIZE= option, and AVERAGE= option).

If the smoothing model options are not specified, the following are the defaults for the demand interval, size, and average components:

```
interval=(transform=auto method=bestn
          levelrest=(0.0001 0.9999)
          trendrest=(0.0001 0.9999)
          damprest =(0.0001 0.9999) criterion=rmse bounds=(1,.));

size    =(transform=auto method=bestn
          levelrest=(0.0001 0.9999)
          trendrest=(0.0001 0.9999)
          damprest =(0.0001 0.9999) criterion=rmse);
```

```
average  =(transform=auto method=bestn
           levelrest=(0.0001 0.9999)
           trendrest=(0.0001 0.9999)
           damprest =(0.0001 0.9999) criterion=rmse);
```

The preceding smoothing model options provide the typical automation in intermittent demand model selection.

The following smoothing model options can be specified:

**BOUNDS=(** *number*, *number* **)**
specifies the component forecast bound. See the section "IDM Smoothing Model Forecast Bounds Options" on page 51.

**CRITERION=***option*

**SELECT=***option*
specifies the model selection criterion (statistic of fit) to be used to select from several candidate models. This option is often used in conjunction with the HOLDOUT= option specified in the FORECAST statement. The CRITERION= option can also be specified as SELECT=. The default is CRITERION=RMSE. The statistics of fit provided are the same as those provided in the FORECAST statement.

**DAMPPARM=***number*
specifies the damping weight parameter initial value. See the section "IDM Smoothing Model Forecast Bounds Options" on page 51.

**DAMPREST=(***number*, *number* **)**
specifies the damping weight parameter restrictions. See the section "IDM Smoothing Model Forecast Bounds Options" on page 51.

**LEVELPARM=***number*
specifies the level weight parameter initial value. See the section "IDM Smoothing Model Forecast Bounds Options" on page 51.

**LEVELREST=(***number*, *number* **)**
specifies the level weight parameter restrictions. See the section "IDM Smoothing Model Forecast Bounds Options" on page 51.

**MEDIAN**
specifies that the median forecast values are to be estimated. Forecasts can be based on the mean or median. By default the mean value is provided. If no transformation is applied to the actual series by using the TRANSFORM= option, the mean and median component forecast values are identical.

**METHOD=***method-name*
specifies the forecasting model to be used to forecast the demand component. A single model can be specified, or a group of candidate models can be specified. If a group of models is specified, the model used to forecast the accumulated time series is selected based on the CRITERION= option of the IDM statement and the HOLDOUT= option of the FORECAST statement. The default is METHOD=BESTN. The following forecasting models are provided:

| | |
|---|---|
| SIMPLE | simple (single) exponential smoothing |
| DOUBLE | double (Brown) exponential smoothing |
| LINEAR | linear (Holt) exponential smoothing |
| DAMPTREND | damped trend exponential smoothing |
| BESTN | best candidate nonseasonal smoothing model (SIMPLE, DOUBLE, LINEAR, DAMPTREND) |

**NOEST**
specifies that the smoothing model parameters are fixed values. To use this option, all of the smoothing model parameters must be explicitly specified. By default, the smoothing model parameters are optimized.

**NOSTABLE**
specifies that the smoothing model parameters are not restricted to the additive invertible region of the parameter space. By default, the smoothing model parameters are restricted to be inside the additive invertible region of the parameter space.

**TRANSFORM=***option*
specifies the time series transformation to be applied to the demand component. The following transformations are provided:

| | |
|---|---|
| NONE | no transformation |
| LOG | logarithmic transformation |
| SQRT | square-root transformation |
| LOGISTIC | logistic transformation |
| BOXCOX(*n* ) | Box-Cox transformation with parameter number where number is between –5 and 5 |
| AUTO | automatically choose between NONE and LOG based on model selection criteria. This option is the default. |

When the TRANSFORM= option is specified, the demand component must be strictly positive. After the demand component is transformed, the model parameters are estimated by using the transformed component. The forecasts of the transformed component are then computed, and finally the transformed component forecasts are inverse transformed. The inverse transform produces either mean or median forecasts depending on whether the MEDIAN option is specified.

**TRENDPARM=***number*
specifies the trend weight parameter initial value. See the section "Smoothing Model Parameter Specification Options" on page 50.

**TRENDREST=(***number***,** *number* **)**
specifies the trend weight parameter restrictions. See the section "Smoothing Model Parameter Specification Options" on page 50.

# Details: HPF Procedure

The HPF procedure can be used to perform trend and seasonal analysis on transactional data. For trend analysis, various sample statistics are computed for each time period defined by the time ID variable and INTERVAL= option. For seasonal analysis, various sample statistics are computed for each season defined by the INTERVAL= or the SEASONALITY= option. For example, suppose the transactional data ranges from June 1990 to January 2000, then the trend statistics are computed for every month: June 1990, July 1990, . . . , January 2000. The seasonal statistics are computed for each season: January, February, . . . , December.

The HPF procedure can be used to forecast time series data as well as transactional data. If the data are transactional, then the procedure must first accumulate the data into a time series before the data can be forecast. The procedure uses the following sequential steps to produce forecasts, with the options that control the step listed to the right:

**Table 3.2** PROC HPF Processing Steps and Control Options

| Step | Operation | Option | Statement |
|---|---|---|---|
| 1 | Accumulation | ACCUMULATE= | ID, FORECAST |
| 2 | Missing Value Interpretation | SETMISSING= | ID, FORECAST |
| 3 | Diagnostic Tests | INTERMITTENT= | FORECAST |
| | | SEASONTEST= | FORECAST |
| 4 | Model Selection | MODEL= | FORECAST |
| | | HOLDOUT=, | FORECAST |
| | | HOLDOUTPCT= | FORECAST |
| | | CRITERION= | FORECAST |
| 5 | Transformations | TRANSFORM= | FORECAST |
| 6 | Parameter Estimation | MODEL= | FORECAST |
| 7 | Forecasting | MODEL= | FORECAST |
| | | LEAD= | PROC HPF |
| 8 | Inverse Transformation | TRANSFORM= | FORECAST |
| | | MEDIAN | FORECAST |
| 9 | Statistics of Fit | OUTSTAT= | PROC HPF |
| 10 | Summation of Forecasts | LEAD=, | PROC HPF |
| | | STARTSUM= | PROC HPF |

Each of the steps shown in Table 3.2 is described in the following sections.

## Accumulation

If the ACCUMULATE= option is specified, data set observations are accumulated within each time period. The frequency (width of each time interval) is specified by the INTERVAL= option. The ID variable contains the time ID values. Each time ID value corresponds to a specific time period. Accumulation is particularly useful when the input data set contains transactional data, whose observations are not spaced with respect to any particular time interval. The accumulated values form the actual time series, which is used in subsequent analyses.

For example, suppose a data set contains the following observations:

```
19MAR1999    10
19MAR1999    30
11MAY1999    50
12MAY1999    20
23MAY1999    20
```

If the INTERVAL=MONTH is specified, all of the preceding observations fall within three time periods of March 1999, April 1999, and May 1999. The observations are to be accumulated within each time period as follows.

If the ACCUMULATE=NONE option is specified, an error is generated because the ID variable values are not equally spaced with respect to the specified frequency (MONTH).

If the ACCUMULATE=TOTAL option is specified:

```
O1MAR1999    40
O1APR1999    .
O1MAY1999    90
```

If the ACCUMULATE=AVERAGE option is specified:

```
O1MAR1999    20
O1APR1999    .
O1MAY1999    30
```

If the ACCUMULATE=MINIMUM option is specified:

```
O1MAR1999    10
O1APR1999    .
O1MAY1999    20
```

If the ACCUMULATE=MEDIAN option is specified:

```
O1MAR1999    20
01APR1999    .
O1MAY1999    20
```

If the ACCUMULATE=MAXIMUM option is specified:

```
O1MAR1999    30
O1APR1999    .
O1MAY1999    50
```

If the ACCUMULATE=FIRST option is specified:

```
O1MAR1999    10
O1APR1999    .
O1MAY1999    50
```

If the ACCUMULATE=LAST option is specified:

```
O1MAR1999    30
O1APR1999    .
O1MAY1999    20
```

If the ACCUMULATE=STDDEV option is specified:

```
O1MAR1999    14.14
O1APR1999    .
O1MAY1999    17.32
```

As can be seen from the preceding examples, even though the data set observations contained no missing values, the accumulated time series might have missing values.

## Missing Value Interpretation

Sometimes missing values should be interpreted as unknown values. The forecasting models used by the HPF procedure can effectively handle missing values (see the section "Missing Value Modeling Issues" on page 50). But sometimes missing values are known (such as when missing values are created from accumulation), and no observations should be interpreted as no (zero) value. In the former case, the SETMISSING= option can be used to interpret how missing values are treated. The SETMISSING=0 option should be used when missing observations are to be treated as no (zero) values. In other cases, missing values should be interpreted as global values, such as minimum or maximum values of the accumulated series. The accumulated and interpreted time series is used in subsequent analyses.

## Diagnostic Tests

The INTERMITTENT= option set the thresholds for categorizing a series as intermittent or non-intermittent. The SEASONTEST= option set the thresholds for categorizing a series as seasonal or nonseasonal.

## Model Selection

When more than one candidate model is specified, forecasts for each candidate model are compared by using the model selection criterion specified by the CRITERION= option. The selection criterion is computed by using the multistep forecasts in the holdout sample range if the HOLDOUT= or HOLDOUTPCT= options are specified, or by the one-step-ahead forecasts for the full range of the time series if the HOLDOUT= and HOLDOUTPCT= options are not specified. The candidate model with the best selection criterion is selected to forecast the time series.

## Transformations

If the TRANSFORM= option is specified, the time series is transformed prior to model parameter estimation and forecasting. Only strictly positive series can be transformed. An error is generated when the TRANSFORM= option is used with a nonpositive series.

## Parameter Estimation

All parameters associated with the model are optimized based on the data with the default parameter restrictions imposed. If the TRANSFORM= option is specified, the transformed time series data are used to estimate the model parameters.

## Smoothing Model Parameter Specification Options

The parameter options are used to specify smoothing model parameters. If the parameter restrictions are not specified, the default is (0.0001 0.9999), which implies that the parameters are restricted between 0.0001 and 0.9999. Parameters and their restrictions are required to be greater than or equal to –1 and less than or equal to 2. Missing values indicate no lower and/or upper restriction. If the parameter initial values are not specified, the optimizer uses a grid search to find an appropriate initial value.

## Missing Value Modeling Issues

The treatment of missing values varies with the forecasting model. For the smoothing models, missing values after the start of the series are replaced with one-step-ahead predicted values, and the predicted values are applied to the smoothing equations. See Chapter 19, "Forecasting Process Details," for greater detail about how missing values are treated in the smoothing models. For MODEL=IDM, specified missing values are assumed to be periods of no demand.

The treatment of missing values can also be specified by the user with the SETMISSING= option, which changes the missing values prior to modeling.

Even though all of the observed data are nonmissing, using the ACCUMULATE= option can create missing values in the accumulated series.

## Forecasting

After the model parameters are estimated, one-step-ahead forecasts are generated for the full range of the actual (optionally transformed) time series data, and multistep forecasts are generated from the end of the observed time series to the future time period after the last observation specified by the LEAD= option. If there are missing values at the end of the time series, the forecast horizon will be greater than that specified by the LEAD= option.

## IDM Smoothing Model Forecast Bounds Options

These options specify the demand component forecast bounds. The forecast bounds restrict the component forecasts. The default for demand interval forecasts is BOUNDS=1. The lower bound for the demand interval forecast must be greater than one. The default for demand size forecasts is BOUNDS=(.,.), which is equivalent to no bounds. The demand size forecasts bounds are applied after the forecast is adjusted by the base value.

## Inverse Transformations

If the TRANSFORM= option is specified, the forecasts of the transformed time series are inverse transformed. By default, the mean (expected) forecasts are generated. If the MEDIAN option is specified, the median forecasts are generated.

## Statistics of Fit

The statistics of fit (or goodness-of-fit statistics) are computed by comparing the actual time series data and the generated forecasts. If the TRANSFORM= option is specified, the statistics of fit are based on the inverse transformed forecasts.

## Forecast Summation

The multistep forecasts generated by the preceding steps are summed from the STARTSUM= number to the LEAD= number. For example, if STARTSUM=4 and LEAD=6, the 4-step-ahead through 6-step-ahead forecasts are summed. The predictions are simply summed. However, the prediction error variance of this sum is computed by taking into account the correlation between the individual predictions. The upper and lower confidence limits for the sum of the predictions is then computed based on the prediction error variance of the sum.

The forecast summation is particularly useful when it is desirable to model in one frequency yet the forecast of interest is another frequency. For example, if a time series has a monthly frequency (INTERVAL=MONTH) and you want a forecast for the third and fourth future months, a forecast summation for the third and fourth month can be obtained by specifying STARTSUM=3 and LEAD=4.

Variance-related computations are computed only when no transformation is specified (TRANSFORM=NONE).

## Comparison to the Time Series Forecasting System

With the exception of model selection, the techniques used in the HPF procedure are identical to the Time Series Forecasting System of SAS/ETS software. For model parameter estimation, the default parameter restrictions are imposed.

## Data Set Output

The HPF procedure can create the OUT=, OUTEST=, OUTFOR=, OUTSTAT=, OUTSUM=, OUTSEASON=, and OUTTREND= data sets. In general, these data sets contain the variables listed in the BY statement. In general, if a forecasting step related to an output data step fails, the values of this step are not recorded or are set to missing in the related output data set, and appropriate error and/or warning messages are recorded in the log.

## OUT= Data Set

The OUT= data set contains the variables specified in the BY, ID, and FORECAST statements. If the ID statement is specified, the ID variable values are aligned and extended based on the ALIGN= and INTERVAL= options. The values of the variables specified in the FORECAST statements are accumulated based on the ACCUMULATE= option, and missing values are interpreted based on the SETMISSING= option. If the REPLACEMISSING option is specified, embedded missing values are replaced by the one-step-ahead forecasts.

These variable values are then extrapolated based on their forecasts, or they are extended with missing values when the MODEL=NONE option is specified. If USE=LOWER is specified, the variable is extrapolated with the lower confidence limits; if USE=UPPER is specified, the variable is extrapolated using the upper confidence limits; otherwise, the variable values are extrapolated with the predicted values. If the TRANSFORM= option is specified, the predicted values contain either mean or median forecasts depending on whether or not the MEDIAN option is specified.

If any of the forecasting steps fail for particular variable, the variable values are extended by missing values.

## OUTEST= Data Set

The OUTEST= data set contains the variables specified in the BY statement as well as the variables listed in this section. For variables listed in FORECAST statements where the option MODEL=NONE is specified, no observations are recorded for these variables. For variables listed in FORECAST statements where the option MODEL=NONE is not specified, the following variables contain observations related to the parameter estimation step:

| | |
|---|---|
| _NAME_ | variable name |
| _MODEL_ | forecasting model |
| _TRANSFORM_ | transformation |
| _PARM_ | parameter name |
| _EST_ | parameter estimate |
| _STDERR_ | standard errors |
| _TVALUE_ | *t* values |
| _PVALUE_ | probability values |

If the parameter estimation step fails for a particular variable, no observations are recorded.

## OUTFOR= Data Set

The OUTFOR= data set contains the variables specified in the BY statement as well as the variables listed in this section. For variables listed in FORECAST statements where the option MODEL=NONE is specified, no observations are recorded for these variables. For variables listed in FORECAST statements where the option MODEL=NONE is not specified, the following variables contain observations related to the forecasting step:

| | |
|---|---|
| _NAME_ | variable name |
| _TIMEID_ | time ID values |
| ACTUAL | actual values |
| PREDICT | predicted values |
| STD | prediction standard errors |
| LOWER | lower confidence limits |
| UPPER | upper confidence limits |
| ERROR | prediction errors |

If the forecasting step fails for a particular variable, no observations are recorded. If the TRANSFORM= option is specified, the values that are listed in this section are the inverse transform forecasts. If the MEDIAN option is specified, the median forecasts are stored; otherwise, the mean forecasts are stored.

## OUTPROCINFO= Data Set

The OUTPROCINFO= data set contains information about the run of the HPF procedure. The following variables are present:

| | |
|---|---|
| _SOURCE_ | name of the procedure, in this case HPF |
| _NAME_ | name of an item being reported. The item can be the number of errors, notes, or warnings, number of forecasts requested, and so on. |
| _LABEL_ | descriptive label for the item in _NAME_ |
| _STAGE_ | current stage of the procedure. For HPFENGINE this is set to ALL. |
| _VALUE_ | value of the item specified in _NAME_ |

## OUTSTAT= Data Set

The OUTSTAT= data set contains the variables specified in the BY statement as well as the variables listed in this section. For variables listed in FORECAST statements where the option MODEL=NONE is specified, no observations are recorded for these variables. For variables listed in FORECAST statements where the option MODEL=NONE is not specified, the following variables contain observations related to the statistics of fit step:

| | |
|---|---|
| _NAME_ | variable name |
| _REGION_ | the region in which the statistics are calculated. Statistics calculated in the fit region are indicated by FIT. Statistics calculated in the forecast region, which happens only if the BACK= option is greater than zero, are indicated by FORECAST. |
| DFE | degrees of freedom error |
| N | number of observations |
| NOBS | number of observations used |
| NMISSA | number of missing actuals |
| NMISSP | number of missing predicted values |
| NPARMS | number of parameters |
| TSS | total sum of squares |
| SST | corrected total sum of squares |
| SSE | sum of square error |
| MSE | mean square error |
| UMSE | unbiased mean square error |
| RMSE | root mean square error |
| URMSE | unbiased root mean square error |
| MAPE | mean absolute percent error |
| MAE | mean absolute error |
| MASE | mean absolute scaled error |
| RSQUARE | R-square |
| ADJRSQ | adjusted R-square |
| AADJRSQ | Amemiya's adjusted R-square |
| RWRSQ | random walk R-square |
| AIC | Akaike information criterion |
| AICC | finite sample corrected AIC |
| SBC | Schwarz Bayesian information criterion |
| APC | Amemiya's prediction criterion |
| MAXERR | maximum error |
| MINERR | minimum error |
| MINPE | minimum percent error |
| MAXPE | maximum percent error |
| ME | mean error |
| MPE | mean percent error |
| MDAPE | median absolute percent error |
| GMAPE | geometric mean absolute percent error |

| | |
|---|---|
| MINPPE | minimum predictive percent error |
| MAXPPE | maximum predictive percent error |
| MSPPE | mean predictive percent error |
| MAPPE | symmetric mean absolute predictive percent error |
| MDAPPE | median absolute predictive percent error |
| GMAPPE | geometric mean absolute predictive percent error |
| MINSPE | minimum symmetric percent error |
| MAXSPE | maximum symmetric percent error |
| MSPE | mean symmetric percent error |
| SMAPE | symmetric mean absolute percent error |
| MDASPE | median absolute symmetric percent error |
| GMASPE | geometric mean absolute symmetric percent error |
| MINRE | minimum relative error |
| MAXRE | maximum relative error |
| MRE | mean relative error |
| MRAE | mean relative absolute error |
| MDRAE | median relative absolute error |
| GMRAE | geometric mean relative absolute error |
| MINAPES | minimum absolute error percent of standard deviation |
| MAXAPES | maximum absolute error percent of standard deviation |
| MAPES | mean absolute error percent of standard deviation |
| MDAPES | median absolute error percent of standard deviation |
| GMAPES | geometric mean absolute error percent of standard deviation |

If the statistics-of-fit step fails for particular variable, no observations are recorded. If the TRANSFORM= option is specified, the values that are listed in this section are computed based on the inverse transform forecasts. If the MEDIAN option is specified, the median forecasts are the basis; otherwise, the mean forecasts are the basis.

## OUTSUM= Data Set

The OUTSUM= data set contains the variables specified in the BY statement as well as the variables listed in this section. The OUTSUM= data set records the summary statistics for each variable specified in a FORECAST statement. For variables listed in FORECAST statements where the option MODEL=NONE is specified, the values related to forecasts are set to missing. For variables listed in FORECAST statements where the option MODEL=NONE is not specified, the forecast values are set based on the USE= option.

Variables related to summary statistics are based on the ACCUMULATE= and SETMISSING= options:

| | |
|---|---|
| _NAME_ | variable name |
| _STATUS_ | forecasting status. Nonzero values imply that no forecast was generated for the series. |
| NOBS | number of observations |
| N | number of nonmissing observations |
| NMISS | number of missing observations |
| MIN | minimum value |
| MAX | maximum value |
| MEAN | mean value |
| STDDEV | standard deviation |

Variables related to forecast summation are based on the LEAD= and STARTSUM= options:

| | |
|---|---|
| PREDICT | forecast summation predicted values |
| STD | forecast summation prediction standard errors |
| LOWER | forecast summation lower confidence limits |
| UPPER | forecast summation upper confidence limits |

Variance-related computations are computed only when no transformation is specified (TRANSFORM=NONE).

Variables related to multistep forecast are based on the LEAD= and USE= options:

_LEAD*n*_ multistep forecast (*n* ranges from one to the value of the LEAD= option). If USE=LOWER, this variable contains the lower confidence limits; if USE=UPPER, this variable contains the upper confidence limits; otherwise, this variable contains the predicted values.

If the forecast step fails for a particular variable, the variables related to forecasting are set to missing. The OUTSUM= data set contains a summary of the (accumulated) time series and optionally its forecasts for all series.

## OUTSEASON= Data Set

The OUTSEASON= data set contains the variables specified in the BY statement as well as the variables listed in this section. The OUTSEASON= data set records the seasonal statistics for each variable specified in a FORECAST statement.

Variables related to seasonal statistics are based on the INTERVAL= or SEASONALITY= options:

| | |
|---|---|
| _NAME_ | variable name |
| _TIMEID_ | time ID values |
| _SEASON_ | seasonal index |
| NOBS | number of observations |
| N | number of nonmissing observations |

| | |
|---|---|
| NMISS | number of missing observations |
| MIN | minimum value |
| MAX | maximum value |
| RANGE | range value |
| SUM | summation value |
| MEAN | mean value |
| STDDEV | standard deviation |
| CSS | corrected sum of squares |
| USS | uncorrected sum of squares |
| MEDIAN | median value |

The preceding statistics are computed for each season.

## OUTTREND= Data Set

The OUTTREND= data set contains the variables specified in the BY statement as well as the variables listed in this section. The OUTTREND= data set records the trend statistics for each variable specified in a FORECAST statement.

Variables related to trend statistics are based on the INTERVAL= and SEASONALITY= options:

| | |
|---|---|
| _NAME_ | variable name |
| _TIMEID_ | time ID values |
| _SEASON_ | seasonal index |
| NOBS | number of observations |
| N | number of nonmissing observations |
| NMISS | number of missing observations |
| MIN | minimum value |
| MAX | maximum value |
| RANGE | range value |
| SUM | summation value |
| MEAN | mean value |
| STDDEV | standard deviation |
| CSS | corrected sum of squares |
| USS | uncorrected sum of squares |
| MEDIAN | median value |

The preceding statistics are computed for each time period.

## Printed Output

The HPF procedure optionally produces printed output for these results by using utilizing the Output Delivery System (ODS). By default, the procedure produces no printed output. All output is controlled by the PRINT= and PRINTDETAILS options associated with the PROC HPF statement. In general, if a forecasting step related to printed output fails, the values of this step are not printed and appropriate error and/or warning messages are recorded in the log. The printed output is similar to the output data set; these similarities are described as follows.

### PRINT=SUMMARY

prints the summary statistics and forecast summaries similar to the OUTSUM= data set.

### PRINT=ESTIMATES

prints the parameter estimates similar to the OUTEST= data set.

### PRINT=FORECASTS

prints the forecasts similar to the OUTFOR= data set. If the MODEL=IDM option is specified, the demand series predictions table is also printed. This table is based on the demand index (when demands occurred).

### PRINT=PERFORMANCE

prints the performance statistics.

### PRINT=PERFORMANCESUMMARY

prints the performance summary for each BY group.

### PRINT=PERFORMANCEOVERALL

prints the performance summary for all BY groups.

### PRINT=STATES

prints the backcast, initial, and final smoothed states.

### PRINT=SEASONS

prints the seasonal statistics similar to the OUTSEASON= data set.

### PRINT=STATISTICS

prints the statistics of fit similar to the OUTSTAT= data set.

### PRINT=TRENDS

prints the trend statistics similar to the OUTTREND= data set.

### PRINTDETAILS

is the opposite of the NOOUTALL option. Specifically, if PRINT=FORECASTS and the PRINTDETAILS options are specified, the one-step-ahead forecasts throughout the range of the data are printed as well as the information related to a specific forecasting model such as the smoothing states. If the PRINTDETAILS option is not specified, only the multistep forecasts are printed.

## ODS Table Names

Table 3.3 relates the PRINT= options to ODS tables:

**Table 3.3** ODS Tables Produced in PROC HPF

| ODS Table Name | Description | Printing Option |
|---|---|---|
| DescStats | Descriptive statistics | PRINT=SUMMARY |
| DemandSummary | Demand summary | PRINT=SUMMARY (MODEL=IDM option only) |
| ForecastSummary | Forecast summary | PRINT=SUMMARY |
| ForecastSummation | Forecast summation | PRINT=SUMMARY |
| ModelSelection | Model selection | PRINT=ESTIMATES |
| ParameterEstimates | Parameter estimates | PRINT=ESTIMATES |
| Forecasts | Forecast | PRINT=FORECASTS |
| Demands | Demands | PRINT=FORECASTS (MODEL=IDM option only) |
| Performance | Performance statistics | PRINT=PERFORMANCE |
| PerformanceSummary | Performance summary | PRINT=PERFORMANCESUMMARY |
| PerformanceSummary | Performance overall | PRINT=PERFORMANCEOVERALL |
| SeasonStatistics | Peasonal statistics | PRINT=SEASONS |
| SmoothedStates | Smoothed states | PRINT=STATES |
| DemandStates | Demand states | PRINT=STATES (MODEL=IDM option only) |
| FitStatistics | Evaluation (in-sample) statistics of fit | PRINT=STATISTICS |
| PerformanceStatistics | Performance (out-of-sample) statistics of fit | PRINT=STATISTICS |
| TrendStatistics | Trend statistics | PRINT=TRENDS |

The ODS table ForecastSummary is related to all time series within a BY group. The other tables are related to a single series within a BY group.

## ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*).

Before you create graphs, ODS Graphics must be enabled (for example, with the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" in that chapter.

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" in that chapter.

This section describes the use of ODS Graphics for creating graphics with the HPF procedure.

### ODS Graph Names

PROC HPF assigns a name to each graph it creates by using ODS. You can use these names to refer to the graphs when you use ODS. The names are listed in Table 3.4.

You select the desired plots via the PLOT= option in the HPF statement.

**Table 3.4** ODS Graphics Produced by PROC HPF

| ODS Graph Name | Plot Description | Statement | PLOT= Option |
|---|---|---|---|
| DemandErrorsPlot | Average demand errors | PROC HPF | PLOT=ERRORS |
| DemandForecastsPlot | Average demand forecasts | PROC HPF | PLOT=FORECASTS |
| DemandIntervalHistogram | Demand interval histogram | PROC HPF | PLOT=MODELS |
| DemandIntervalPlot | Demand interval forecast plot | PROC HPF | PLOT=MODELS |
| DemandSizeHistogram | Demand size histogram | PROC HPF | PLOT=MODELS |
| DemandSizePlot | Demand size forecast plot | PROC HPF | PLOT=MODELS |
| ErrorACFNORMPlot | Standardized autocorrelation of prediction errors | PROC HPF | PLOT=ACF |
| ErrorACFPlot | Autocorrelation of prediction errors | PROC HPF | PLOT=ACF |
| ErrorCorrelationPlots | Prediction error plot panel | PROC HPF | PLOT=CORR |
| ErrorHistogram | Prediction error histogram | PROC HPF | PLOT=ERRORS |
| ErrorIACFNORMPlot | Standardized inverse autocorrelation of prediction errors | PROC HPF | PLOT=IACF |
| ErrorIACFPlot | Inverse autocorrelation of prediction errors | PROC HPF | PLOT=IACF |
| ErrorPACFNORMPlot | Standardized partial autocorrelation of prediction errors | PROC HPF | PLOT=PACF |
| ErrorPACFPlot | Partial autocorrelation of prediction errors | PROC HPF | PLOT=PACF |
| ErrorPeriodogramPlot | Periodogram of prediction errors | PROC HPF | PLOT=PERIODOGRAM |
| ErrorPlot | Prediction errors | PROC HPF | PLOT=ERRORS |
| ErrorSpectralDensityPlot | Combined periodogram and spectral density estimate plot | PROC HPF | PLOT=SPECTRUM |

**Table 3.4** (continued)

| ODS Graph Name | Plot Description | Statement | Option |
|---|---|---|---|
| ErrorWhiteNoiseLogProbPlot | White noise log probability plot of prediction errors | PROC HPF | PLOT=WN |
| ErrorWhiteNoiseProbPlot | White noise probability plot of prediction errors | PROC HPF | PLOT=WN |
| ForecastsOnlyPlot | Forecasts only plot | PROC HPF | PLOT=FORECASTSONLY |
| ForecastsPlot | Forecasts plot | PROC HPF | PLOT=FORECAST |
| LevelStatePlot | Smoothed level state plot | PROC HPF | PLOT=LEVELS |
| ModelForecastsPlot | Model and forecasts plot | PROC HPF | PLOT=MODELFORECAST |
| ModelPlot | Model plot | PROC HPF | PLOT=MODELS |
| SeasonStatePlot | Smoothed season state plot | PROC HPF | PLOT=SEASONS |
| StockingAveragePlot | Stocking average plot | PROC HPF | PLOT=FORECASTS |
| StockingLevelPlot | Stocking level plot | PROC HPF | PLOT=FORECASTS |
| TrendStatePlot | Smoothed trend state plot | PROC HPF | PLOT=TRENDS |

# Examples: HPF Procedure

## Example 3.1: Automatic Forecasting of Time Series Data

This example illustrates how the HPF procedure can be used for the automatic forecasting of time series data. Retail sales data is used for this illustration.

The following DATA step creates a data set from data recorded monthly at numerous points of sales. The data set, SALES, contains a variable DATE that represents time and a variable for each sales item. Each value of the DATE variable is recorded in ascending order, and the values of each of the other variables represent a single time series:

```
data sales;
   format date date9.;
   input date : date9. shoes socks laces dresses coats
         shirts ties belts hats blouses;
datalines;
01JAN1994  3557   3718   6368.80    575    987   10.8200   15.0000   102.600   12410   15013

   ... more lines ...
```

The following HPF procedure statements automatically forecast each of the monthly time series:

```
proc hpf data=sales out=nextyear;
   id date interval=month;
   forecast _ALL_;
run;
```

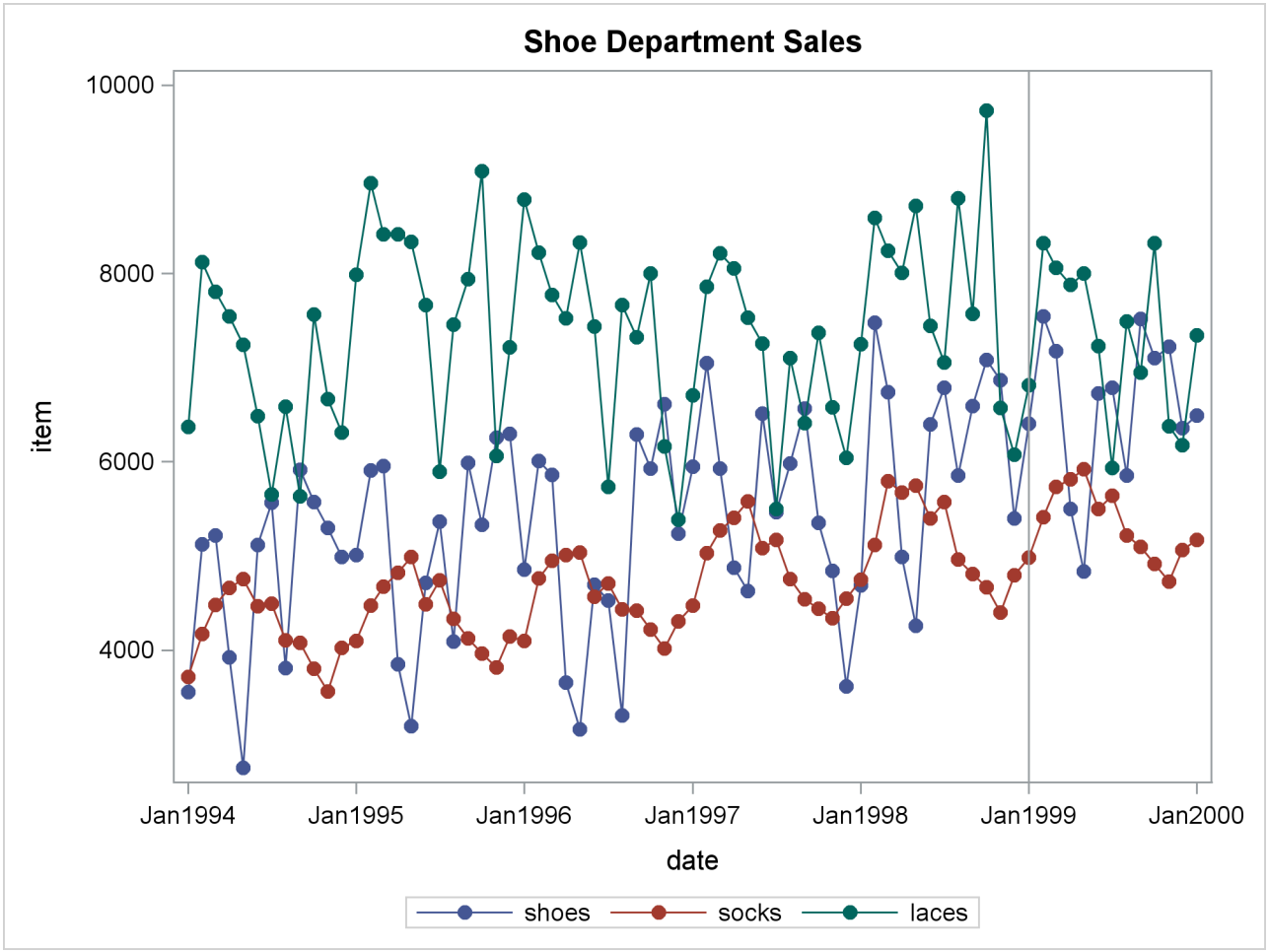
The preceding statements automatically select the best fitting model, generate forecasts for every numeric variable in the input data set (SALES) for the next twelve months, and store these forecasts in the output data set (NEXTYEAR).

The following SGPLOT procedure statements plot the forecasts related to shoe sales:

```
title1 "Shoe Department Sales";
proc sgplot data=nextyear;
   series x=date y=shoes / markers
      markerattrs=(symbol=circlefilled) lineattrs=(pattern=1);
   series x=date y=socks / markers
      markerattrs=(symbol=circlefilled) lineattrs=(pattern=1);
   series x=date y=laces / markers
      markerattrs=(symbol=circlefilled) lineattrs=(pattern=1);
   xaxis values=('01JAN1994'd to '01DEC2000'd by year);
   yaxis label='item';
   refline '01JAN1999'd / axis=x;
run;
```

The SGPLOT procedure results are shown in Output 3.1.1. The historical data is shown to the left of the vertical reference line, and the forecasts for the next twelve monthly periods is shown to the right.

**Output 3.1.1** Retail Sales Forecast Plots



The following HPF procedure statements are identical to the preceding statements except that the PRINT=FORECASTS option is specified:

```
proc hpf data=sales out=nextyear print=forecasts;
   id date interval=month;
   forecast _ALL_;
run;
```

In addition to automatically forecasting each of the monthly time series, the preceding statements print the forecasts by using the Output Delivery System (ODS), which is partially shown in Output 3.1.2. This output shows the predictions, prediction standard errors, and the upper and lower confidence limits for the next twelve monthly periods.

**Output 3.1.2** Forecast Tables

**Shoe Department Sales**

**The HPF Procedure**

Forecasts for Variable shoes

| Obs | Time | Forecasts | Standard Error | 95% Confidence Limits | |
|---|---|---|---|---|---|
| 62 | FEB1999 | 7548.0041 | 607.5238 | 6357.2792 | 8738.7289 |
| 63 | MAR1999 | 7177.1472 | 699.4400 | 5806.2701 | 8548.0244 |
| 64 | APR1999 | 5497.5595 | 780.7609 | 3967.2964 | 7027.8227 |
| 65 | MAY1999 | 4838.2001 | 854.5169 | 3163.3778 | 6513.0224 |
| 66 | JUN1999 | 6728.4521 | 922.5244 | 4920.3375 | 8536.5668 |
| 67 | JUL1999 | 6786.1094 | 985.9738 | 4853.6362 | 8718.5826 |
| 68 | AUG1999 | 5853.9650 | 1045.6953 | 3804.4399 | 7903.4900 |
| 69 | SEP1999 | 7517.0144 | 1102.2949 | 5356.5561 | 9677.4728 |
| 70 | OCT1999 | 7100.2489 | 1156.2315 | 4834.0769 | 9366.4210 |
| 71 | NOV1999 | 7224.6449 | 1207.8618 | 4857.2793 | 9592.0106 |
| 72 | DEC1999 | 6357.1556 | 1257.4701 | 3892.5594 | 8821.7518 |
| 73 | JAN2000 | 6492.2657 | 1305.2871 | 3933.9500 | 9050.5815 |

## Example 3.2: Automatic Forecasting of Transactional Data

This example illustrates how the HPF procedure can be used to automatically forecast transactional data. Internet data is used for this illustration.

The data set WEBSITES contains data recorded at several Internet Web sites. WEBSITES contains a variable TIME and the variables BOATS, CARS, and PLANES that represent Internet Web site data. Each value of the TIME variable is recorded in ascending order, and the values of each of the other variables represent a transactional data series.

The following HPF procedure statements automatically forecast each of the transactional data series:

```
proc hpf data=websites out=nextweek lead=7;
   id time interval=dtday accumulate=total;
   forecast boats cars planes;
run;
```

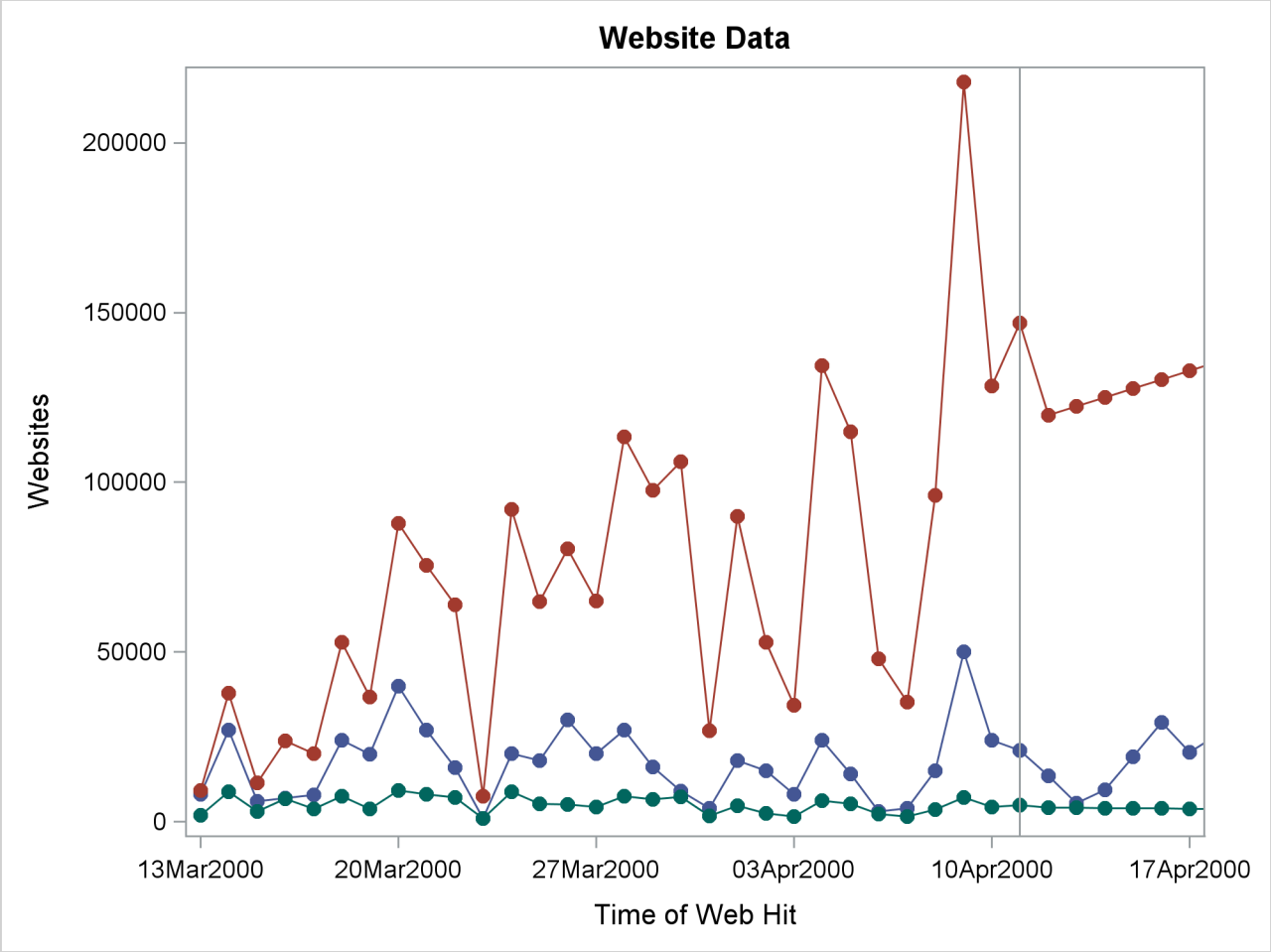
The preceding statements accumulate the data into a daily time series, automatically generate forecasts for the BOATS, CARS, and PLANES variables in the input data set (WEBSITES) for the next week, and store the forecasts in the output data set (NEXTWEEK).

The following SGPLOT procedure statements plot the forecasts related to the Internet data:

```
title1 "Website Data";
proc sgplot data=nextweek noautolegend;
   series x=time y=boats / markers
      markerattrs=(symbol=circlefilled) lineattrs=(pattern=1);
   series x=time y=cars / markers
      markerattrs=(symbol=circlefilled) lineattrs=(pattern=1);
   series x=time y=planes / markers
      markerattrs=(symbol=circlefilled) lineattrs=(pattern=1);
   xaxis values=('13MAR2000:00:00:00'dt to '18APR2000:00:00:00'dt by dtweek);
   yaxis label='Websites';
   refline '11APR2000:00:00:00'dt / axis=x;
run;
```

The SGPLOT procedure results are shown in Output 3.2.1. The historical data is shown to the left of the vertical reference line, and the forecasts for the next twelve monthly periods are shown to the right.

**Output 3.2.1** Internet Data Forecast Plots




## Example 3.3: Specifying the Forecasting Model

In the previous example, the HPF procedure was used to automatically select the appropriate forecasting model by using the root mean square error (RMSE) as the default selection criterion. This example illustrates how the HPF procedure can be used to more narrowly specify the possible candidate models. Internet data from the previous example are used for this illustration.

This example forecasts the BOATS variable by using the best seasonal forecasting model (BESTS) that minimizes the mean absolute percent error (MAPE), forecasts the CARS variable by using the best nonseasonal forecasting model (BESTN) that minimizes the mean square error (MSE) using holdout sample analysis, and forecasts the PLANES variable by using the log Winters (additive) method. The following HPF procedure statements forecast each of the transactional data series based on these requirements:

```
proc hpf data=websites out=nextweek lead=7;
   id time interval=dtday accumulate=total;
   forecast boats   / model=bests criterion=mape;
   forecast cars    / model=bestn criterion=mse holdout=5;
   forecast planes  / model=addwinters transform=log;
run;
```

## Example 3.4: Extending the Independent Variables for Multivariate Forecasts

In the previous example, the HPF procedure was used to forecast several transactional series variables by using univariate models. This example illustrates how the HPF procedure can be used to extend the independent variables associated with a multiple regression forecasting problem. Specifically, PROC HPF is used to extend the independent variables for use in forecasting a regression model.

This example accumulates and forecasts the BOATS, CARS, and PLANES variables as illustrated in the previous example. In addition, it accumulates the ENGINES variable to form a time series that is then extended with missing values within the forecast horizon with the specification of MODEL=NONE.

```
proc hpf data=websites out=nextweek lead=7;
   id time interval=dtday accumulate=total;
   forecast engines / model=none;
   forecast boats   / model=bests criterion=mape;
   forecast cars    / model=bestn criterion=mse holdout=5;
   forecast planes  / model=winters transform=log;
run;
```

The following AUTOREG procedure statements are used to forecast the ENGINES variable by regressing on the independent variables (BOATS, CARS, and PLANES).

```
proc autoreg data=nextweek;
   model engines = boats cars planes / noprint;
   output out=enginehits p=predicted;
run;
```

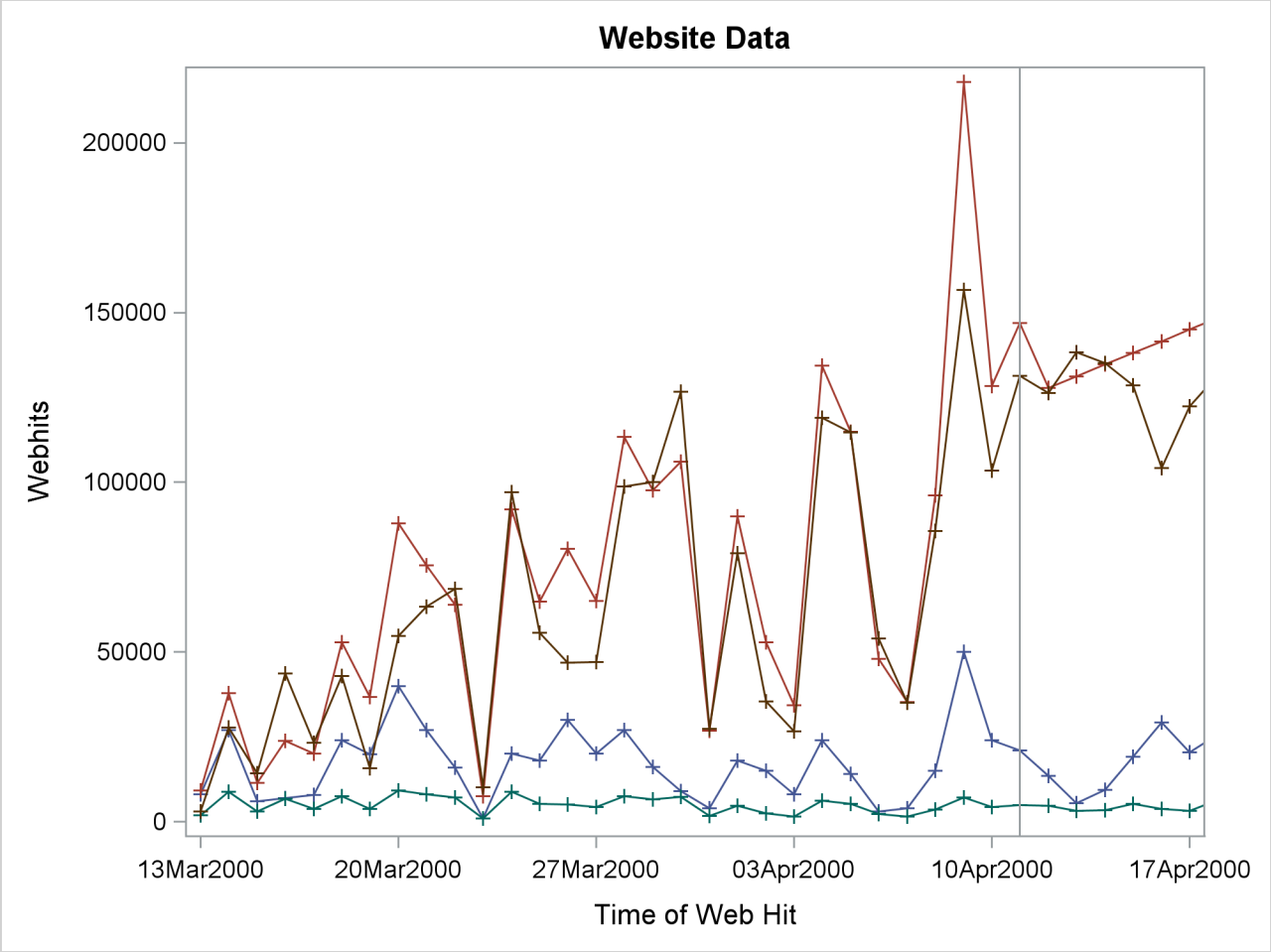
The output data set (NEXTWEEK) of the PROC HPF statement is used as an input data set for the PROC AUTOREG statement. The output data set of PROC AUTOREG contains the forecast of the variable ENGINES based on the regression model with the variables BOATS, CARS, and PLANES as regressors. See Chapter 8, "The AUTOREG Procedure" (*SAS/ETS User's Guide*) for details about autoregression models.

The following SGPLOT procedure statements plot the forecasts related to the ENGINES variable:

```
proc sgplot data=enginehits noautolegend;
   series x=time y=boats / markers
      markerattrs=(symbol=plus) lineattrs=(pattern=1);
   series x=time y=cars / markers
      markerattrs=(symbol=plus) lineattrs=(pattern=1);
   series x=time y=planes / markers
      markerattrs=(symbol=plus) lineattrs=(pattern=1);
   series x=time y=predicted / markers
      markerattrs=(symbol=plus) lineattrs=(pattern=1);
   xaxis values=('13MAR2000:00:00:00'dt to '18APR2000:00:00:00'dt by dtweek);
   yaxis label='Webhits';
   refline '11APR2000:00:00:00'dt / axis=x;
run;
```

The SGPLOT procedure results are shown in Output 3.4.1. The historical data is shown to the left of the vertical reference line, and the forecasts for the next four weekly periods is shown to the right.

**Output 3.4.1** Internet Data Forecast Plots



## Example 3.5: Forecasting Intermittent Time Series Data

This example illustrates how the HPF procedure can be used to forecast intermittent time series data. Inventory demand is used for this illustration.

The following DATA step creates a data set from inventory data recorded at no particular frequency. The data set INVENTORY contains a variable DATE that represents time and the demand variables (TIRES, HUBCAPS, and LUGBOLTS), which represent inventory items. Each value of the DATE variable is recorded in ascending order, and the values of each of the other variables represent a transactional data series.

```
data inventory;
   format date date9.;
   input date : date9. tires hubcaps lugbolts;
datalines;
01JUN1997  0  0  0
01JUL1997  0  1  5
01AUG1997  0  0  0

   ... more lines ...
```

The following HPF procedure statements forecast each of the transactional data series by using an intermittent demand model:

```
proc hpf data=inventory out=nextmonth lead=4 print=forecasts;
   id date interval=week accumulate=total;
   forecast tires hubcaps lugbolts / model=idm;
run;
```

The preceding statements accumulate the data into a weekly time series, generate forecasts for the TIRES, HUBCAPS, and LUGBOLTS variables in the input data set (INVENTORY) for the four weekly periods, and store the forecasts in the output data set (NEXTMONTH). The PRINT=FORECAST option produces results which are partially shown in Output 3.5.1. The first two tables record the demand series and predictions. The third table represents forecasts or recommended stocking levels.

**Output 3.5.1** Forecast Tables

## Website Data

### The HPF Procedure

**Demands for Variable tires**

| Demand | | Demand Intervals | | | | | Demand Size | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Index | Time | Actual | Predicted | Std | 95% Confidence Limits | | Error | Actual | Predicted | Std | 95% Confidence Limits | | Error |
| 1 | Sun, 31 Aug 1997 | 14 | 14.0000 | . | . | . | . | 6.00000 | 4.42151 | 2.18902 | 0.13110 | 8.71191 | 1.57849 |
| 2 | Sun, 26 Oct 1997 | 8 | 14.0000 | 6.6736 | 0.9200 | 27.0800 | -6.0000 | 4.00000 | 4.74216 | 2.18902 | 0.45176 | 9.03257 | -0.74216 |
| 3 | Sun, 1 Mar 1998 | 18 | 12.8920 | 6.6736 | -0.1880 | 25.9720 | 5.1080 | 2.00000 | 4.59140 | 2.18902 | 0.30100 | 8.88180 | -2.59140 |
| 4 | Sun, 26 Apr 1998 | 8 | 13.8353 | 6.6736 | 0.7553 | 26.9153 | -5.8353 | 2.00000 | 4.06498 | 2.18902 | -0.22542 | 8.35539 | -2.06498 |
| 5 | Sun, 31 May 1998 | 5 | 12.7577 | 6.6736 | -0.3223 | 25.8377 | -7.7577 | 2.00000 | 3.64550 | 2.18902 | -0.64490 | 7.93591 | -1.64550 |
| 6 | Sun, 27 Sep 1998 | 17 | 11.3251 | 6.6736 | -1.7549 | 24.4051 | 5.6749 | 6.00000 | 3.31123 | 2.18902 | -0.97917 | 7.60164 | 2.68877 |
| 7 | Sun, 3 Jan 1999 | 14 | 12.3731 | 6.6736 | -0.7069 | 25.4531 | . | . | 3.85743 | 2.18902 | -0.43297 | 8.14783 | . |

Stock = (Interval Actual)*(Predict) - (Size Actual)

**Demands for Variable tires**

| Demand | | Estimate of Mean Demand per Period | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Index | Time | Actual | Predict | Std | 95% Confidence Limits | | Error | Stock |
| 1 | Sun, 31 Aug 1997 | 0.42857 | 0.31582 | . | . | . | 0.1127495 | -1.578494 |
| 2 | Sun, 26 Oct 1997 | 0.50000 | 0.33873 | 0.22476 | -0.10180 | 0.77926 | 0.1612742 | -1.290193 |
| 3 | Sun, 1 Mar 1998 | 0.11111 | 0.35614 | 0.25064 | -0.13510 | 0.84738 | -.2450321 | 4.410578 |
| 4 | Sun, 26 Apr 1998 | 0.25000 | 0.29381 | 0.21241 | -0.12251 | 0.71013 | -.0438128 | 0.350503 |
| 5 | Sun, 31 May 1998 | 0.40000 | 0.28575 | 0.22756 | -0.16026 | 0.73176 | 0.1142508 | -0.571254 |
| 6 | Sun, 27 Sep 1998 | 0.35294 | 0.29238 | 0.25893 | -0.21512 | 0.79987 | 0.0605614 | -1.029543 |
| 7 | Sun, 3 Jan 1999 | . | 0.27553 | 0.20420 | -0.12470 | 0.67576 | . | . |

Stock = (Interval Actual)*(Predict) - (Size Actual)

**Forecasts for Variable tires**

| Obs | Time | Forecasts | Standard Error | 95% Confidence Limits | |
|---|---|---|---|---|---|
| 84 | Sun, 3 Jan 1999 | 0.27553 | 0.20420 | -0.12470 | 0.67576 |
| 85 | Sun, 10 Jan 1999 | 0.27553 | 0.20420 | -0.12470 | 0.67576 |
| 86 | Sun, 17 Jan 1999 | 0.27553 | 0.20420 | -0.12470 | 0.67576 |
| 87 | Sun, 24 Jan 1999 | 0.27553 | 0.20420 | -0.12470 | 0.67576 |

## Example 3.6: Illustration of ODS Graphics

This example illustrates the use of ODS graphics.

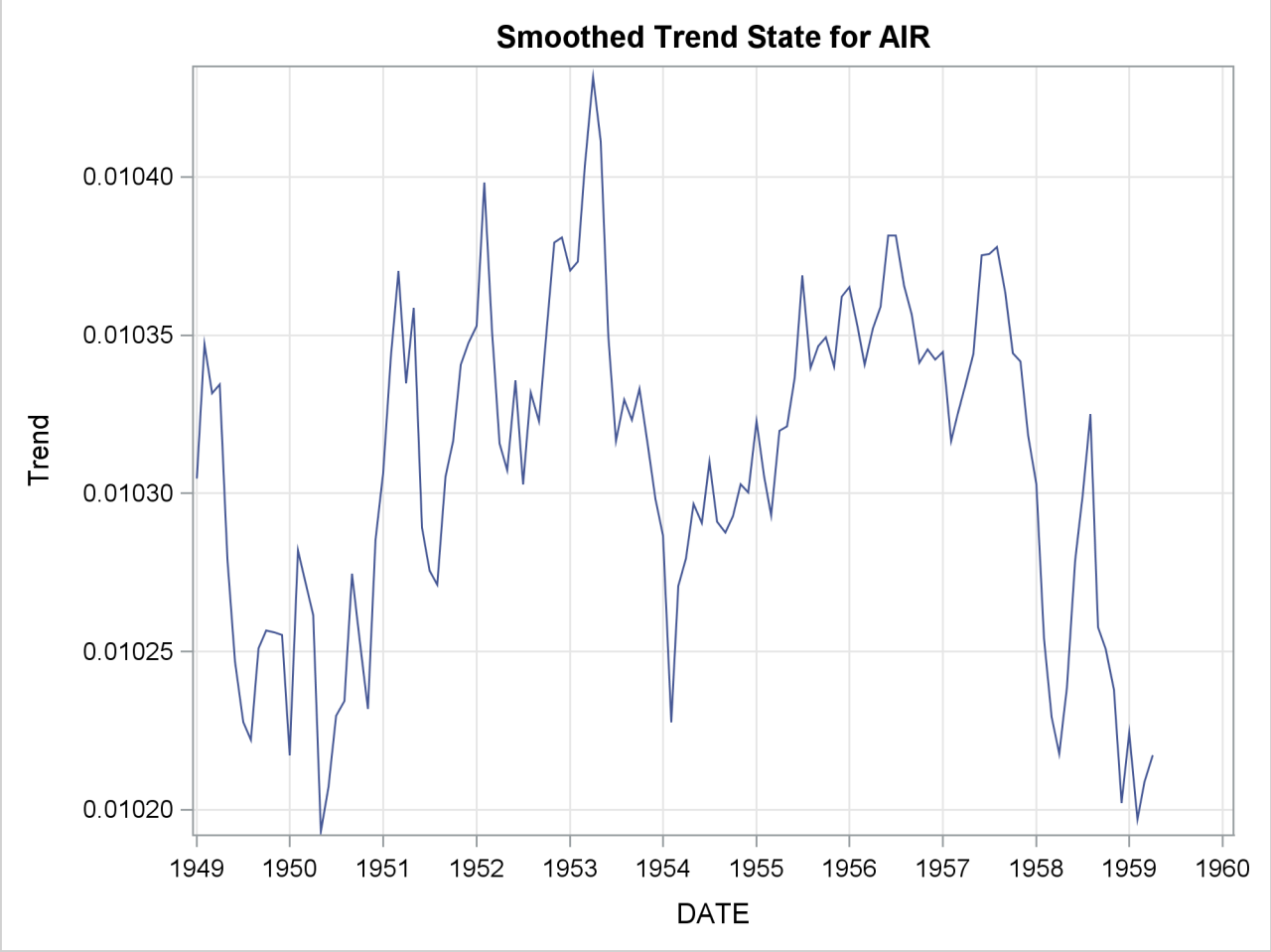
The following statements use the SASHELP.AIR data set to automatically forecast the time series of international airline travel.

The graphical displays are requested by specifying the PLOT= option in the PROC HPF statement. In this case, all plots are requested. Output 3.6.1 through Output 3.6.5 show a selection of the plots created.

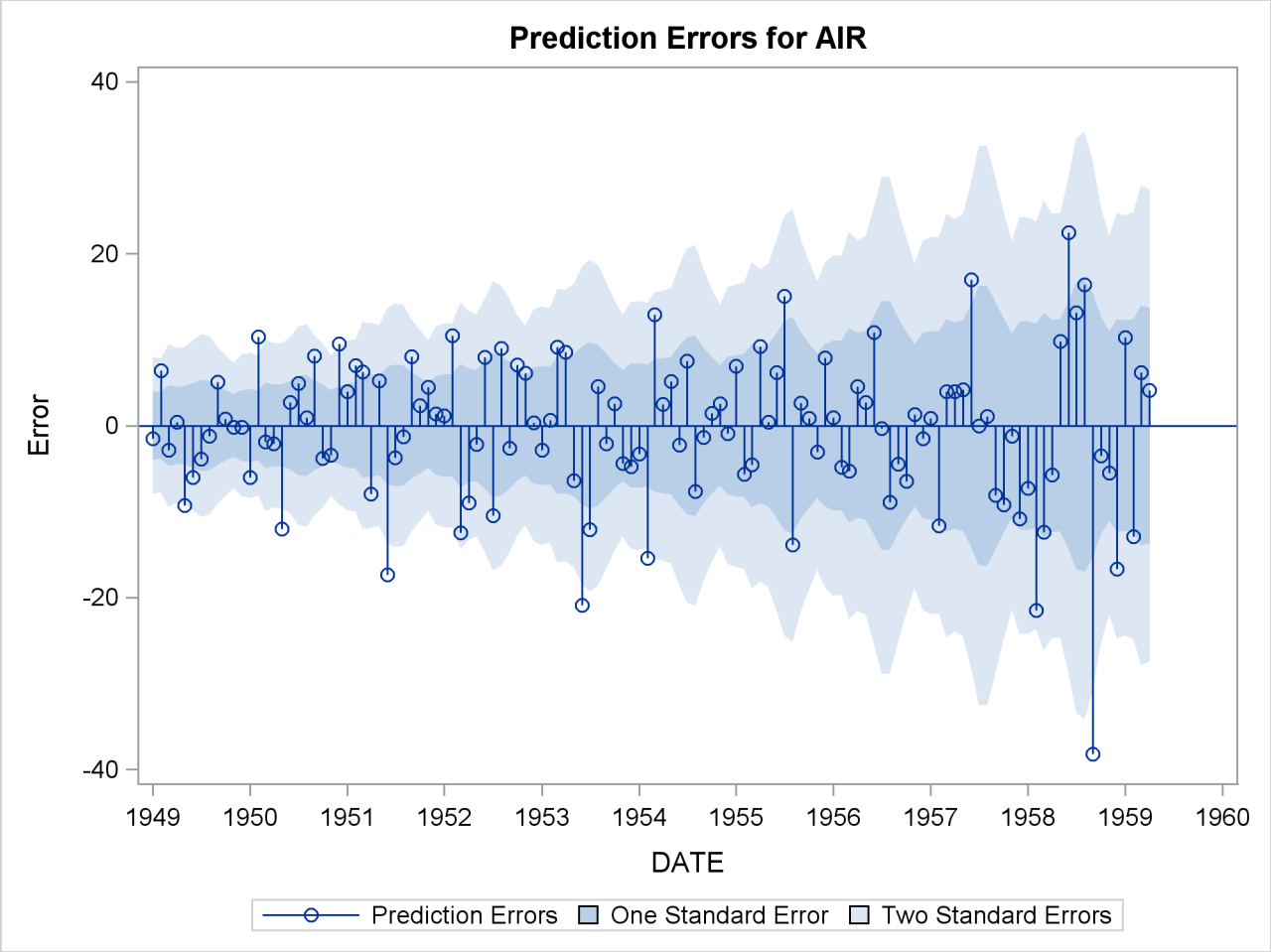
For specific information about the graphics available in the HPF procedure, see the "ODS Graphics" on page 60 section.

```
proc hpf data=sashelp.air out=_null_
         lead=20
         back=20
         print=all
         plot=all;
   id date interval=month;
   forecast air / model=best transform=auto criterion=mape;
run;
```

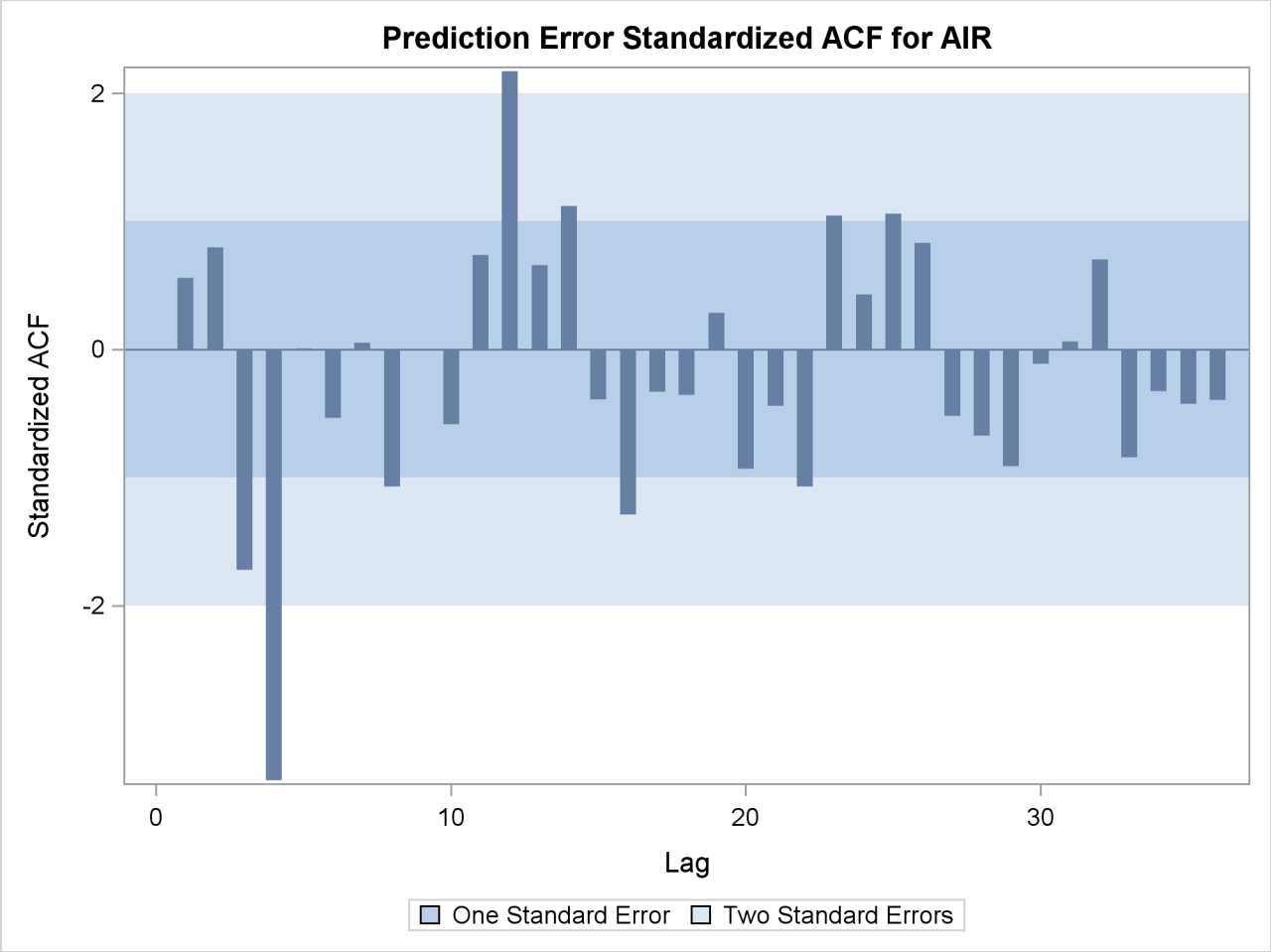
**Output 3.6.1** Smoothed Trend Plot



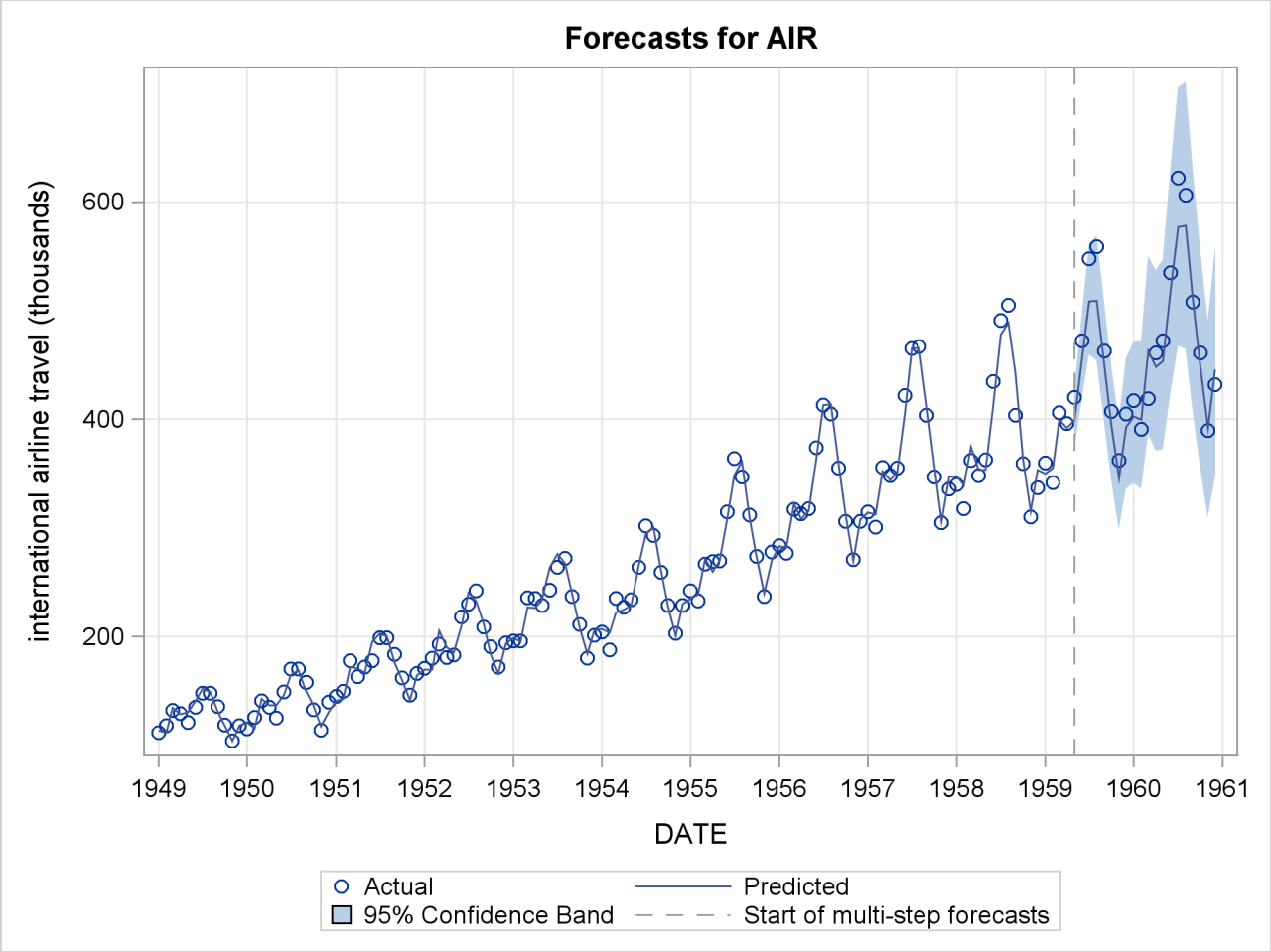
**Output 3.6.2** Prediction Error Plot



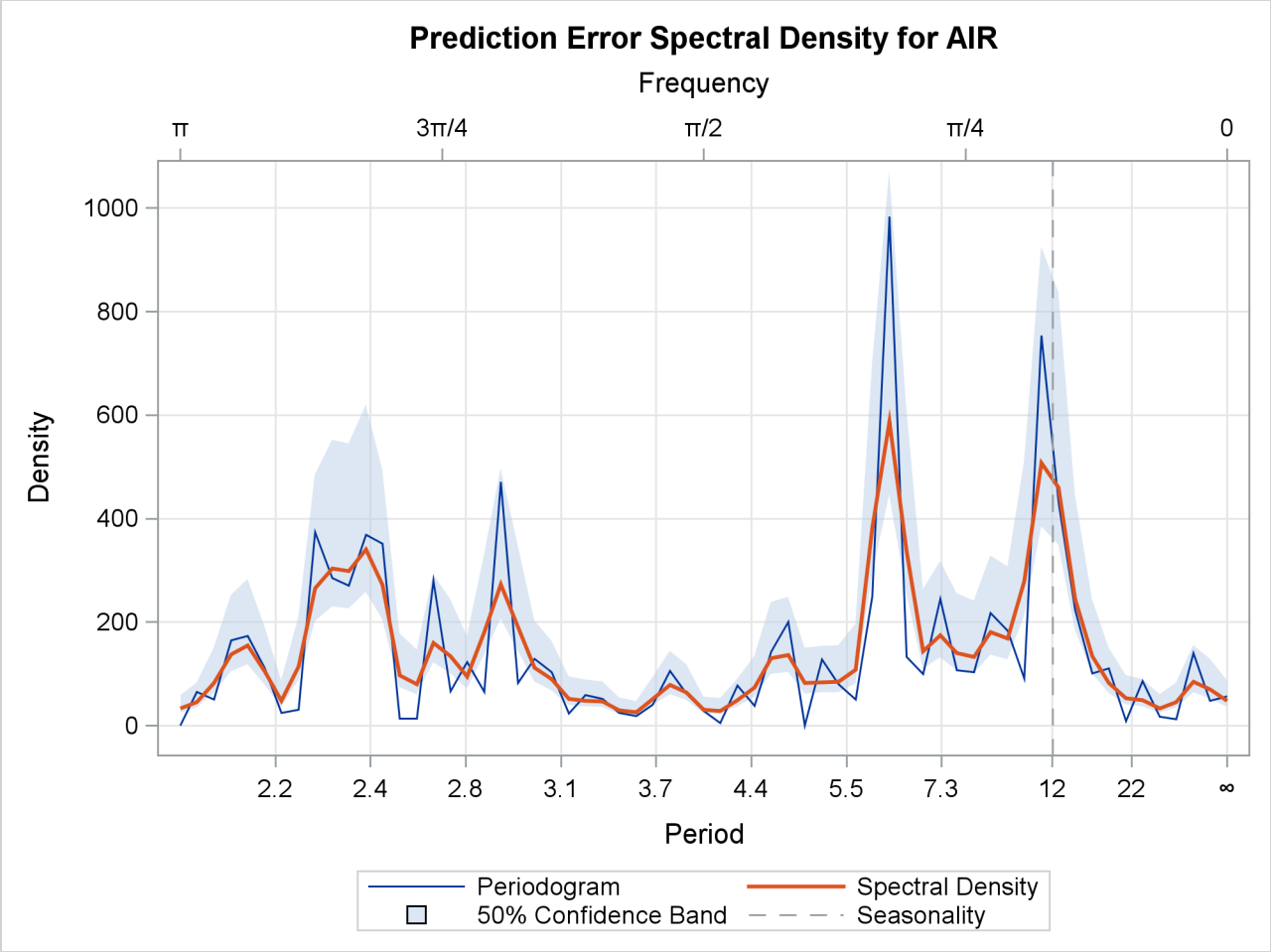
**Output 3.6.3** Prediction Error Standardized ACF Plot



**Output 3.6.4** Forecast Plot



**Output 3.6.5** Prediction Error Spectral Density




## References

Pyle, D. (1999), *Data Preparation for Data Mining*, San Francisco: Morgan Kaufmann.

# Chapter 4
# The HPFARIMASPEC Procedure

**Contents**


Overview: HPFARIMASPEC Procedure . . . . . **75**
Getting Started: HPFARIMASPEC Procedure . . . . . **75**
Syntax: HPFARIMASPEC Procedure . . . . . **77**
Functional Summary . . . . . 77
PROC HPFARIMASPEC Statement . . . . . 78
FORECAST Statement . . . . . 79
INPUT Statement . . . . . 80
ESTIMATE Statement . . . . . 82
Examples: HPFARIMASPEC Procedure . . . . . **83**
Example 4.1: Some HPFARIMASPEC Syntax Illustrations . . . . . 83
Example 4.2: How to Include ARIMA Models in a Model Selection List . . . . . 85
Example 4.3: A Generic Seasonal Model Specification Suitable for Different Season Lengths . . . . . 86
References . . . . . **88**


## Overview: HPFARIMASPEC Procedure

The HPFARIMASPEC procedure is used to create an ARIMA (autoregressive integrated moving average) model specification file. The output of this procedure is an XML file that stores the intended ARIMA model specification. This XML specification file can be used for different purposes—for example, to populate the model repository used by the HPFENGINE procedure (see Chapter 7, "The HPFENGINE Procedure"). You can specify very general ARIMA models with this procedure. In particular, any model that can be analyzed with the ARIMA procedure can be specified; see Chapter 7, "The ARIMA Procedure" (*SAS/ETS User's Guide*). Moreover, the model specification can include series transformations such as log or Box-Cox transformations.

## Getting Started: HPFARIMASPEC Procedure

The following example shows how to create an ARIMA model specification file. In this example the specification for an Airline model with one input is created.

```
proc hpfarimaspec repository=work.arima
                  name=Airline1
                  label="Airline model with one input";
   forecast symbol=Y q=(1)(12) dif=(1, 12) noint
            transform=log;
   input symbol=X dif=(1, 12);
   estimate method=ml;
run;
```

The options in the PROC HPFARIMASPEC statement are used to specify the location of the specification file that will be output. Here the REPOSITORY= option specifies that the output file be placed in the catalog WORK.ARIMA, the NAME= option specifies that the name of the file be Airline1.xml, and the LABEL= option specifies a label for this catalog member. The other statements in the procedure specify the ARIMA model and the options used to control the parameter estimation process for the model. The model specification begins with the FORECAST statement that specifies the following:

- transformation, such as log or Box-Cox, and the differencing orders associated with the variable that is to be forecast
- autoregressive and moving average polynomials
- presence or absence of the constant in the model

According to the FORECAST statement, the model contains no constant term and has a two-factor moving average polynomial of orders 1 and 12. The forecast variable is log transformed and differenced with differencing orders 1 and 12. The SYMBOL= option in the FORECAST statement can be used to provide a convenient name for the forecast variable. This name is only a placeholder, and a proper data variable is associated with this name when this model specification is used in actual data analysis.

Next, the INPUT statement provides the transfer function specification associated with the input variable in the model. In the INPUT statement you can specify the following:

- transformation, such as log or Box-Cox, and the lagging and differencing orders associated with the input variable
- numerator and denominator polynomials associated with the transfer function input

In this case the input variable is differenced with differencing orders 1 and 12, and it enters the model as a simple regressor. Here again the SYMBOL= option can be used to supply a convenient name for the input variable. If a model contains multiple input variables, then each input variable has to be specified with a separate INPUT statement.

Lastly, the ESTIMATE statement specifies that the model be estimated using the ML method of estimation.

# Syntax: HPFARIMASPEC Procedure

The HPFARIMASPEC procedure uses the following statements.

**PROC HPFARIMASPEC** *options* **;**
**FORECAST** *options* **;**
**INPUT** *options* **;**
**ESTIMATE** *options* **;**

## Functional Summary

Table 4.1 summarizes the statements and options that control the HPFARIMASPEC procedure.

**Table 4.1** HPFARIMASPEC Functional Summary

| Description | Statement | Option |
|---|---|---|
| **Model Repository Options** | | |
| Specifies the model repository | PROC HPFARIMASPEC | REPOSITORY= |
| Specifies the model specification name | PROC HPFARIMASPEC | NAME= |
| Specifies the model specification label | PROC HPFARIMASPEC | LABEL= |
| **Options for Specifying Symbolic Series Names** | | |
| Specifies a symbolic name for the response series | FORECAST | SYMBOL= |
| Specifies a symbolic name for the input series | INPUT | SYMBOL= |
| Specifies a predefined trend as the input series | INPUT | PREDEFINED= |
| **Options for Specifying the Model** | | |
| Specifies the response series transformation | FORECAST | TRANSFORM= |
| Specifies the response series transformation options | FORECAST | TRANSOPT= |
| Specifies the response series differencing orders | FORECAST | DIF= |
| Specifies the input series transformation | INPUT | TRANSFORM= |
| Specifies the input series differencing orders | INPUT | DIF= |
| Specifies the input series lagging order | INPUT | DELAY= |
| Specifies the ARIMA part of the model | FORECAST | |
| Specifies the AR polynomial | FORECAST | P= |
| Specifies autoregressive starting values | FORECAST | AR= |
| Specifies the MA polynomial | FORECAST | Q= |

| Description | Statement | Option |
|---|---|---|
| Specifies moving average starting values | FORECAST | MA= |
| Suppresses the constant term in the model | FORECAST | NOINT |
| Specifies a starting value for the mean parameter | FORECAST | MU= |
| Specifies the NOISE variance | FORECAST | NOISEVAR= |
| Specifies the transfer function part of the model | INPUT | |
| Specifies the numerator polynomial of a transfer function | INPUT | NUM= |
| Specifies the starting values for the numerator polynomial coefficients | INPUT | NC= |
| Specifies the starting value for the zero degree numerator polynomial coefficient | INPUT | NZ= |
| Specifies the denominator polynomial of a transfer function | INPUT | DEN= |
| Specifies the starting values for the denominator polynomial coefficients | INPUT | DC= |
| **Options to Control the Parameter Estimation** | | |
| Specifies the estimation method | ESTIMATE | METHOD= |
| Suppress the iterative estimation process | ESTIMATE | NOEST |
| Specifies the maximum number of iterations | ESTIMATE | MAXITER= |
| Specifies the convergence criterion | ESTIMATE | CONVERGE= |

## PROC HPFARIMASPEC Statement

**PROC HPFARIMASPEC** *options* **;**

The following options can be used in the PROC HPFARIMASPEC statement.

**LABEL=***SAS-label*

specfies a descriptive label for the model specification to be stored in the SAS catalog or external file reference. The LABEL= option can also be specified as SPECLABEL=.

**NAME=***SAS-name*

names the model specification to be stored in the SAS catalog or external file reference. The NAME= option can also be specified as SPECNAME=.

**REPOSITORY=***SAS-catalog-name*

**REPOSITORY=***SAS-file-reference*

names the SAS catalog or external file reference to contain the model specification. The REPOSITORY= option can also be specified as MODELREPOSITORY=, MODELREP=, or REP=.

## FORECAST Statement

**FORECAST** *options* **;**

The FORECAST statement specifies the operations to be performed on the response series as well as the autoregressive and moving average polynomials in the model. The presence or absence of a constant in the model is also specified here.

The following options are used in the FORECAST statement.

**SYMBOL=***variable*

**VAR=***variable*

specifies a symbolic name for the dependent series. This symbol specification is optional. If the SYMBOL= option is not specified, *Y* is used as a default symbol.

**DIF=***order*

**DIF=***( order1, order2, . . . )*

specifies the differencing orders for the dependent series. For example, DIF= (1 12) specifies that the series be differenced using the operator $(1 - B)(1 - B^{12})$. The differencing orders can be positive integers or they can be "s", which indicates a placeholder that will be substituted later with an appropriate value. The use of placeholders is explained further in Example 4.3.

**P=***order*

**P=(***lag, . . . , lag* **) . . . (***lag, . . . , lag* **)**

**P=(***lag, . . . , lag* **)<** $s_1$ **> . . . (***lag, . . . , lag* **)<** $s_k$ **>**

specifies the autoregressive part of the model. By default, no autoregressive parameters are fit.

P=($l_1, l_2, \ldots, l_k$) defines a model with autoregressive parameters at the specified lags. P= *order* is equivalent to P=(1, 2, . . . , *order* ).

A concatenation of parenthesized lists specifies a factored model. For example, P=(1,2,5)(6,12) specifies the autoregressive model

$$(1 - \phi_{1,1}B - \phi_{1,2}B^2 - \phi_{1,3}B^5)(1 - \phi_{2,1}B^6 - \phi_{2,2}B^{12})$$

Optionally, you can specify *multipliers* after the parenthesized lists. For example, P=(1)(1)12 is equivalent to P=(1)(12), and P=(1,2)4(1)12(1,2)24 is equivalent to P=(4,8)(12)(24,48). These multipliers can either be positive integers or they can be "s", which indicates a placeholder that will be substituted later with an appropriate value. The use of placeholders in the multiplier specification is explained in Example 4.3.

**Q=***order*

**Q=(***lag, . . . , lag* **) . . . (***lag, . . . , lag* **)**

**Q=(***lag, . . . , lag* **)<** $s_1$ **> . . . (***lag, . . . , lag* **)<** $s_k$ **>**

specifies the moving average part of the model. By default, no moving average parameters are fit.

The manner of specification of the moving average part is identical to the specification of the autoregressive part described in the P= option.

**AR=***value* **...**
lists starting values for the autoregressive parameters.

**MA=***value* **...**
lists starting values for the moving average parameters.

**NOCONSTANT**
**NOINT**
suppresses the fitting of a constant (or intercept) parameter in the model. (That is, the parameter $\mu$ is omitted.)

**MU=***value*
specifies the MU parameter.

**NOISEVAR=***value*
specifies the noise variance. This is useful only if you want to specify an externally published model that is fully specified.

**TRANSFORM=***option*
specifies the transformation to be applied to the time series. The following transformations are provided:

| | |
|---|---|
| NONE | no transformation applied |
| LOG | logarithmic transformation |
| SQRT | square-root transformation |
| LOGISTIC | logistic transformation |
| BOXCOX(*n* ) | Box-Cox transformation with parameter number where number is between –5 and 5 |

When the TRANSFORM= option is specified, the intended time series must be strictly positive.

**TRANSOPT=***option*
specifies that mean or median forecasts be estimated. The following options are provided:

| | |
|---|---|
| MEAN | Mean forecasts are estimated. This is the default. |
| MEDIAN | Median forecasts are estimated. |

If no transformation is applied to the actual series with the TRANSFORM= option, the mean and median time series forecast values are identical.

## INPUT Statement

**INPUT** *options* **;**

The INPUT statements specify the transfer function inputs in the model. A separate INPUT statement is needed for each of the transfer function inputs. In this statement you can specify all the features of the transfer function associated with the input variable under consideration. The following options are used in the INPUT statement.

**(SYMBOL|VAR)=***variable*

specifies a symbolic name for the transfer function input series. This symbol specification is optional. If the SYMBOL= option or the PREDEFINED= option is not specified, then *X* is used as a default symbol. If there are multiple INPUT statements, then an attempt is made to generate a unique set of input symbols.

**PREDEFINED=***option*

associates a predefined trend or a set of seasonal dummy variables with this transfer function. The SYMBOL=, and PREDEFINED= options are mutually exclusive.

In the following list of options, let $t$ represent the observation count from the start of the period of fit for the model, and let $X_t$ be the value of the time trend variable at observation $t$.

| | |
|---|---|
| LINEAR | a linear trend, with $X_t = t - c$ |
| QUADRATIC | a quadratic trend, with $X_t = (t - c)^2$ |
| CUBIC | a cubic trend, with $X_t = (t - c)^3$ |
| INVERSE | an inverse trend, with $X_t = 1/t$ |
| SEASONAL | seasonal dummies. For a seasonal cycle of length $s$, the seasonal dummy regressors include $X_{i,t} : 1 \le i \le (s-1), 1 \le t \le n$ for models that include an intercept term, and $X_{i,t} : 1 \le i \le (s), 1 \le t \le n$ for models that do not include an intercept term. |

Each element of a seasonal dummy regressor is either zero or one, based on the following rule:

$$X_{i,t} = \begin{cases} 1 & \text{when } i = t \\ 0 & \text{otherwise} \end{cases}$$

Note that if the model includes an intercept term, the number of seasonal dummy regressors is one less than $s$ to ensure that the linear system is full rank.

**DIF=***order*

**DIF=***( order1, order2, . . . )*

specifies the differencing orders for the input series. See the DIF= option of the FORECAST statement for additional information.

**DIF=SAME**

specifies that the differencing orders from the FORECAST statement be used.

**DELAY=***order*

specifies the delay (or lag) order for the input series.

**NUM=***order*

**NUM= (***lag, . . . , lag* **) . . . (***lag, . . . , lag* **)**

**NUM= (***lag, . . . , lag* **)<** $s_1$ **> . . . (***lag, . . . , lag* **)<** $s_k$ **>**

specifies the numerator polynomial of the transfer function. See the P= option of the FORECAST statement for additional information about the polynomial order specification.

**DEN=***order*

**DEN= (***lag, . . . , lag* **) . . . (***lag, . . . , lag* **)**

**DEN= (***lag, . . . , lag* **)<** $s_1$ **> . . . (***lag, . . . , lag* **)<** $s_k$ **>**

specifies the denominator polynomial of the transfer function. See the P= option of the FORECAST statement for additional information about the polynomial order specification.

**NC=***value* **. . .**

lists starting values for the numerator polynomial coefficients.

**DC=***value* **. . .**

lists starting values for the denominator polynomial coefficients.

**NZ=***value*

specifies the scale parameter—that is, the zero degree coefficient of the numerator.

**TRANSFORM=***option*

specifies the transformation to be applied to the time series. The following transformations are provided:

| | |
|---|---|
| NONE | no transformation applied |
| LOG | logarithmic transformation |
| SQRT | square-root transformation |
| LOGISTIC | logistic transformation |
| BOXCOX(*n* ) | Box-Cox transformation with parameter number where number is between –5 and 5 |

When the TRANSFORM= option is specified, the intended time series must be strictly positive.

## ESTIMATE Statement

**ESTIMATE** *options* **;**

This is an optional statement in the procedure. Here you can specify the estimation method or whether to hold the model parameters fixed to their starting values. You can also specify some parameters that control the nonlinear optimization process.

The following options are available.

**METHOD=ML**

**METHOD=ULS**

**METHOD=CLS**

specifies the estimation method to use. METHOD=ML specifies the maximum likelihood method. METHOD=ULS specifies the unconditional least squares method. METHOD=CLS specifies the conditional least squares method. METHOD=CLS is the default.

**NOEST**
uses the values specified with the AR=, MA=, and so on, as final parameter values. The estimation process is suppressed except for the estimation of the residual variance. The specified parameter values are used directly by the next FORECAST statement. Use of NOEST requires that all parameters be specified via the AR=, MA=, and so on. Partially specified models will cause an error when used by the HPFENGINE procedure. When NOEST is specified, standard errors, *t* values, and the correlations between estimates are displayed as 0 or missing. (The NOEST option is useful, for example, when you wish to generate forecasts that correspond to a published model.)

**CONVERGE=***value*
specifies the convergence criterion. Convergence is assumed when the largest change in the estimate for any parameter is less than the CONVERGE= option value. If the absolute value of the parameter estimate is greater than 0.01, the relative change is used; otherwise, the absolute change in the estimate is used. The default is CONVERGE=0.001.

**DELTA=***value*
specifies the perturbation value for computing numerical derivatives. The default is DELTA=0.001.

**MAXITER=*n***
**MAXIT=*n***
specifies the maximum number of iterations allowed. The default is MAXITER=50.

**NOLS**
begins the maximum likelihood or unconditional least squares iterations from the preliminary estimates rather than from the conditional least squares estimates that are produced after four iterations.

**NOSTABLE**
specifies that the autoregressive and moving average parameter estimates for the noise part of the model not be restricted to the stationary and invertible regions, respectively.

**SINGULAR=***value*
specifies the criterion for checking singularity. If a pivot of a sweep operation is less than the SINGULAR= value, the matrix is deemed singular. Sweep operations are performed on the Jacobian matrix during final estimation and on the covariance matrix when preliminary estimates are obtained. The default is SINGULAR=1E–7.

# Examples: HPFARIMASPEC Procedure

## Example 4.1: Some HPFARIMASPEC Syntax Illustrations

The following statements illustrate the PROC HPFARIMASPEC syntax for some of the commonly needed modeling activities. Suppose that a variety of ARIMA models are to be fit to a data set that contains a sales series as the forecast variable and several promotional events as predictor series. In all these cases the model repository is kept the same, work.arima, and the models are named as *model1, model2,* and so on, to ensure uniqueness. Note that in a given repository, the models must have unique names. The symbols for the forecast and input variables are *sales* and *promo1, promo2,* and so on, respectively.

```
/* Two transfer functions */
proc hpfarimaspec repository=work.arima
                  name=model1;
   forecast symbol=sales transform=log
            q=(1)(12) dif=(1,12) noint;
   input symbol=promo1 dif=(1, 12) den=2;
   input symbol=promo2 num=2 delay=3;
run;

/* Box-Cox transform and Estimation Method=ML */

proc hpfarimaspec repository=work.arima
                  name=model2;
   forecast symbol=sales transform=BoxCox(0.8) p=2;
   estimate method=ml;
run;

/* suppress parameter estimation: in this    */
/* case all the parameters must be specified */

proc hpfarimaspec repository=work.arima
                  name=model3;
   forecast symbol=sales transform=log
            p=2 ar=0.1 0.8 mu=3.5;
   estimate noest method=ml;
run;

/* Supply starting values for the parameters */

proc hpfarimaspec repository=work.arima
                  name=model4;
   forecast symbol=sales transform=log
            p=2 ar=0.1 0.8 mu=3.5;
   input symbol=promo1
         den=1 dc=0.1 nz=-1.5;
run;

/* Create a generic seasonal Airline model with one input
   that is applicable for different season lengths */

proc hpfarimaspec repository=work.arima
                  name=model5
                  label="Generic Airline Model with One Input";
   forecast symbol=Y q=(1)(1)s dif=(1, s) noint
            transform= log;
   input symbol=X dif=(1, s);
run;
```

## Example 4.2: How to Include ARIMA Models in a Model Selection List

One of the primary uses of the HPFARIMASPEC procedure is to add candidate ARIMA models to a model selection list that can be used by the HPFENGINE procedure (see Chapter 7, "The HPFENGINE Procedure"). The HPFARIMASPEC procedure is used to create the ARIMA model specifications, and the HPFSELECT procedure is used to add the specifications to a model selection list (see Chapter 15, "The HPFSELECT Procedure"). This example illustrates this scenario.

Here the Gas Furnace Data, "Series J" from Box and Jenkins (1976), is used. This data set contains two series, Input Gas Rate and Output CO2. The goal is to forecast the output CO2, using the input Gas Rate as a predictor if necessary.

The following DATA step statements read the data in a SAS data set.

```
data seriesj;
   input GasRate CO2 @@;
   date = intnx( 'day', '01jan1950'd, _n_-1 );
   format date DATE.;
datalines;
-0.109  53.8  0.000  53.6  0.178  53.5  0.339  53.5
 0.373  53.4  0.441  53.1  0.461  52.7  0.348  52.4
 0.127  52.2 -0.180  52.0 -0.588  52.0 -1.055  52.4

... more lines ...
```

Three candidate models are specified, *m1, m2*, and *m3*. Out of these three models, *m1* is known to be a good fit to the data. It is a transfer function model that involves the input Gas Rate. The other two models are simplified versions of *m1*. The following syntax shows how to specify these models and how to create a selection list that combines them by using the HPFSELECT procedure. In the HPFSELECT procedure note the use of the INPUTMAP option in the SPEC statement. It ties the symbolic variable names used in the HPFARIMASPEC procedure with the actual variable names in the data set. If the symbolic names were appropriate to start with, then the INPUTMAP option is not necessary.

```
* make spec1;
proc hpfarimaspec repository=work.mycat
                  name=m1;
   forecast symbol=y p=2;
   input symbol=x delay=3 num=(1,2) den=1;
   estimate method=ml;
run;

* make spec2;
proc hpfarimaspec repository=work.mycat name=m2;
   forecast symbol=y p=2;
   input symbol=x delay=3;
   estimate method=ml;
run;
```

```
* make spec3;
proc hpfarimaspec repository=work.mycat
                  name=m3;
   forecast symbol=y p=2;
   estimate method=ml;
run;

* make a selection list that includes m1, m2 and m3;
proc hpfselect repository=work.mycat
               name=myselect;

   spec m1 / inputmap(symbol=y var=co2)
             inputmap(symbol=x var=gasrate);

   spec m2 / inputmap(symbol=y var=co2)
             inputmap(symbol=x var=gasrate);

   spec m3 / inputmap(symbol=y var=co2);
run;
```

This selection list can now be used in the HPFENGINE procedure for various types of analyses. The following syntax shows how to compare these models based on the default comparison criterion, mean absolute percentage error (MAPE). As expected, model *m1* turns out to be the best of the three compared (see Figure 4.2.1).

```
proc hpfengine data=seriesj
               repository=work.mycat
               globalselection=myselect
               lead=0
               print=(select);
   forecast co2;
   input    gasrate;
run;
```

**Output 4.2.1** Model Selection Based on the MAPE Criterion

**The HPFENGINE Procedure**

| Model Selection Criterion = MAPE | | | |
|---|---|---|---|
| **Model** | **Statistic** | **Selected** | **Label** |
| **M1** | 0.31478330 | Yes | ARIMA: Y ~ P = 2 + INPUT: Lag(3) X NUM = 2 DEN = 1 |
| **M2** | 0.50671996 | No | ARIMA: Y ~ P = 2 + INPUT: Lag(3) X |
| **M3** | 0.53295590 | No | ARIMA: Y ~ P = 2 |

## Example 4.3: A Generic Seasonal Model Specification Suitable for Different Season Lengths

In the case of many seasonal model specifications, it is possible to describe a generic specification that is applicable in a variety of situations just by changing the season length specifications at appropriate places. As an example, consider the Airline model, which is very useful for modeling seasonal data. The Airline model for a monthly series can be specified using the following syntax:

```
proc hpfarimaspec repository=work.specs
                  name=MonthlyAirline
     label="Airline Model For A Series With Season Length 12";
   forecast symbol=Y q=(1)(1)12 dif=(1, 12) noint
            transform= log;
run;
```

It is easy to see that the same syntax is applicable to a quarterly series if the multiplier in the MA specification is changed from 12 to 4 and the seasonal differencing order is similarly changed from 12 to 4. A generic specification that allows for late binding of season lengths can be generated by the following syntax:

```
proc hpfarimaspec repository=work.specs
                  name=GenericAirline
                  label="Generic Airline Model";
    forecast symbol=Y q=(1)(1)s dif=(1, s) noint
             transform= log;
run;
```

In this syntax the multiplier in the MA specification is changed from 12 to "s", and similarly the seasonal differencing order 12 is changed to "s". This syntax creates a template for the Airline model that is applicable to different season lengths. When the HPFENGINE procedure, which actually uses such model specifications to estimate the model and produce the forecasts, encounters such "generic" specification it automatically creates a proper specification by replacing the placeholders for the seasonal multiplier and the seasonal differencing order with the value implied by the ID variable or its SEASONALITY= option. The following example illustrates the use of this generic spec. It shows how the same spec can be used for monthly and quarterly series. The parameter estimates for monthly and quarterly series are given in Output 4.3.1 and Output 4.3.2, respectively.

```
/* Create a selection list that contains
   the Generic Airline Model */

proc hpfselect repository=work.specs
               name=genselect;
   spec GenericAirline;
run;

/* Monthly interval */

proc hpfengine data=sashelp.air
               repository=work.specs
               globalselection=genselect
               print=(estimates);
   id date interval=month;
   forecast air;
run;
```

**Output 4.3.1** Parameter Estimates for the Monthly Series

**The HPFENGINE Procedure**

**Parameter Estimates for GENERICAIRLINE Model**

| Component | Parameter | Estimate | Standard Error | t Value | Approx Pr > \|t\| |
|---|---|---|---|---|---|
| AIR | MA1_1 | 0.37727 | 0.08196 | 4.60 | <.0001 |
| AIR | MA2_12 | 0.57236 | 0.07802 | 7.34 | <.0001 |

```
/* Create a quarterly series to illustrate
   accumulating the monthly Airline series to quarterly*/

proc timeseries data=sashelp.air out=Qair;
   id date interval=quarter;
   var air / accumulate=total;
run;

/* Quarterly interval */

proc hpfengine data=Qair
               repository= work.specs
               globalselection=genselect
               print=(estimates);
   id date interval=quarter;
   forecast air;
run;
```

**Output 4.3.2** Parameter Estimates for the Quarterly Series

**The HPFENGINE Procedure**

**Parameter Estimates for GENERICAIRLINE Model**

| Component | Parameter | Estimate | Standard Error | t Value | Approx Pr > \|t\| |
|---|---|---|---|---|---|
| AIR | MA1_1 | 0.05892 | 0.15594 | 0.38 | 0.7075 |
| AIR | MA2_4 | 0.50558 | 0.14004 | 3.61 | 0.0008 |

# References

Box, G. E. P. and Jenkins, G. M. (1976), *Time Series Analysis: Forecasting and Control*, Rev. Edition, San Francisco: Holden-Day.

# Chapter 5
# The HPFDIAGNOSE Procedure

## Contents


Overview: HPFDIAGNOSE Procedure . . . . . **90**
Getting Started: HPFDIAGNOSE Procedure . . . . . **91**
Default Settings . . . . . 94
The Role of the IDM Statement . . . . . 95
Syntax: HPFDIAGNOSE Procedure . . . . . **96**
Functional Summary . . . . . 96
PROC HPFDIAGNOSE Statement . . . . . 100
ADJUST Statement . . . . . 107
ARIMAX Statement . . . . . 108
BY Statement . . . . . 111
COMBINE Statement . . . . . 112
ESM Statement . . . . . 116
EVENT Statement . . . . . 117
FORECAST Statement . . . . . 117
ID Statement . . . . . 118
IDM Statement . . . . . 120
INPUT Statement . . . . . 121
TRANSFORM Statement . . . . . 122
TREND Statement . . . . . 122
UCM Statement . . . . . 123
Details: HPFDIAGNOSE Procedure . . . . . **125**
Adjustment Operations . . . . . 125
Data Preparation . . . . . 125
Functional Transformation . . . . . 127
Stationarity Test . . . . . 128
ARMA Order Selection . . . . . 130
Transfer Functions in an ARIMAX Model . . . . . 131
Outliers . . . . . 133
Intermittent Demand Model . . . . . 134
Exponential Smoothing Model . . . . . 135
Unobserved Components Model . . . . . 135
Values of Status . . . . . 138
Holdout Sample . . . . . 139
EVENTS . . . . . 139
Thread Usage in PROC HPFDIAGNOSE . . . . . 141
Relationship with PROC HPFENGINE . . . . . 142

Data Set Input/Output . . . . . 144
ODS Table Names . . . . . 148
Examples: HPFDIAGNOSE Procedure . . . . . **149**
Example 5.1: Selection of Input Variables . . . . . 149
Example 5.2: Selection of Events and Input Variables . . . . . 150
Example 5.3: Intermittent Demand Series . . . . . 152
Example 5.4: Exponential Smoothing Model . . . . . 152
Example 5.5: Unobserved Components Model . . . . . 153
Example 5.6: Automatic Model Combination . . . . . 154
References . . . . . **155**

# Overview: HPFDIAGNOSE Procedure

The HPFDIAGNOSE procedure provides a comprehensive set of tools for automated univariate time series model identification. Time series data can have outliers, structural changes, and calendar effects. In the past, finding a good model for time series data usually required experience and expertise in time series analysis.

The HPFDIAGNOSE procedure automatically diagnoses the statistical characteristics of time series and identifies appropriate models. The models that HPFDIAGNOSE considers for each time series include autoregressive integrated moving average with exogenous inputs (ARIMAX), exponential smoothing, and unobserved components models. Log transformation and stationarity tests are automatically performed. The ARIMAX model diagnostics find the autoregressive (AR) and moving average (MA) orders, detect outliers, and select the best input variables. The unobserved components model (UCM) diagnostics find the best components and select the best input variables.

The HPFDIAGNOSE procedure provides the following functionality:

- intermittency (or interrupted series) test
- functional transformation test
- simple differencing and seasonal differencing tests
- tentative simple ARMA order identification
- tentative seasonal ARMA order identification
- outlier detection
- significance test of events (indicator variables)

- transfer function identification
  - intermittency test
  - functional transformation for each regressor
  - simple differencing order and seasonal differencing order for each regressor
  - time delay for each regressor
  - simple numerator and denominator polynomial orders for each regressor
- intermittent demand model (automatic selection)
- exponential smoothing model (automatic selection)
- unobserved components model (automatic selection)

PROC HPFDIAGNOSE can be abbreviated as PROC HPFDIAG.

## Getting Started: HPFDIAGNOSE Procedure

This section outlines the use of the HPFDIAGNOSE procedure and shows examples of how to create ARIMA, ESM, and UCM model specifications.

The following example prints the diagnostic tests of an ARIMA model. In the HPFDIAGNOSE statement, the SEASONALITY=12 option specifies the length of the seasonal cycle of the time series, and the PRINT=SHORT option prints the chosen model specification. The FORECAST statement specifies the dependent variable (Air). The ARIMAX statement specifies that an ARIMA model is to be diagnosed.

```
proc hpfdiagnose data=sashelp.air
                 seasonality=12
                 print=short;
   forecast air;
   transform;
   arimax;
run;
```

Figure 5.1 shows the ARIMAX model specification. The log transformation test and trend test are conducted by default. The log transformation was applied to the dependent series and the seasonal ARIMA $(1,1,0)(0,1,1)_{12}$ model was selected. The default model selection criterion (RMSE) was used. The STATUS column explains warnings or errors during diagnostic tests. STATUS=OK indicates that the model was successfully diagnosed.

**Figure 5.1** ARIMAX Specification

**The HPFDIAGNOSE Procedure**

**ARIMA Model Specification**

| Variable | Functional Transform | Constant | p | d | q | P | D | Q | Seasonality | Model Criterion | Statistic | Status |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AIR | LOG | NO | 1 | 1 | 0 | 0 | 1 | 1 | 12 | RMSE | 10.8353 | OK |

The following example prints the diagnostic tests of an ESM for airline data. The ID statement INTERVAL=MONTH option specifies an implied seasonality of 12. The ESM statement specifies that an exponential smoothing model is to be diagnosed.

```
proc hpfdiag data=sashelp.air print=short;
   id date interval=month;
   forecast air;
   transform;
   esm;
run;
```

Figure 5.2 shows the ESM specification. The chosen model specification applied the log transformation and selected a multiplicative seasonal model with a trend component (WINTERS).

**Figure 5.2** ESM Specification

**The HPFDIAGNOSE Procedure**

**Exponential Smoothing Model Specification**

| Variable | Functional Transform | Selected Model | Component | Model Criterion | Statistic |
|---|---|---|---|---|---|
| AIR | LOG | WINTERS | LEVEL | RMSE | 10.6521 |
| | | | TREND | | |
| | | | SEASONAL | | |

The following example prints the diagnostic tests of a UCM for airline data. The UCM statement specifies that an unobserved component model is to be diagnosed.

```
proc hpfdiag data=sashelp.air print=short;
   id date interval=month;
   forecast air;
   transform;
   ucm;
run;
```

When the column SELECTED=YES, as shown in Figure 5.3, the component is significant. When the column SELECTED=NO, the component is insignificant.

When SELECTED=YES, the STOCHASTIC column has either YES or NO. STOCHASTIC=YES indicates that a component has a statistically significant variance, indicating the component is changing over time; STOCHASTIC=NO indicates that the variance of a component is not statistically significant, but the component itself is still significant.

Figure 5.3 shows that the irregular, level, slope, and seasonal components are selected. The irregular, level, and seasonal components have statistically significant variances. The slope component is constant over the time.

**Figure 5.3** Select Components

**The HPFDIAGNOSE Procedure**

| Unobserved Components Model(UCM) Specification | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Variable | Functional Transform | Component | Selected | Stochastic | Seasonality | Model Criterion | Statistic | Status |
| **AIR** | LOG | IRREGULAR | YES | YES | | RMSE | 10.9801 | OK |
| | | LEVEL | YES | YES | | | | |
| | | SLOPE | YES | NO | | | | |
| | | SEASON | YES | YES | 12 | | | |

The following example shows how to pass a model specification created by the HPFDIAGNOSE procedure to the HPFENGINE procedure.

An ARIMAX model specification file, a model selection list, and a model repository Sasuser.Mycat are created by the HPFDIAGNOSE procedure. The ARIMAX model specification file and the model selection list are contained in the Sasuser.Mycat repository.

The OUTEST= data set is used to transmit the diagnostic results to the HPFENGINE procedure by the INEST= option. The Work.Est_One data set contains the information about the data set variable and the model selection list.

```
proc datasets lib=sasuser mt=catalog nolist;
   delete hpfscor mycat;
run;

proc hpfdiag data=sashelp.air outest=est_one
             modelrepository=sasuser.mycat criterion=MAPE;
   id date interval=month;
   forecast air;
   transform;
   arimax;
run;

proc hpfengine data=sashelp.air print=(select)
               modelrepository=sasuser.mycat inest=est_one;
   forecast air;
   id date interval=month;
run;
```

Figure 5.4 shows the DIAG0 model specification created by the HPFDIAGNOSE procedure in the previous example. The model specification is labeled DIAG0 because the HPFDIAGNOSE procedure uses BASENAME=DIAG by default.

**Figure 5.4** Model Selection from the HPFENGINE Procedure

**The HPFENGINE Procedure**

| Model Selection Criterion = MAPE | | | |
|---|---|---|---|
| Model | Statistic | Selected | Label |
| **diag0** | 2.9422734 | Yes | ARIMA: Log( AIR ) ~ P = 1 D = (1,12) Q = (12) NOINT |

The following example shows how the HPFDIAGNOSE and HPFENGINE procedures can be used to select a single model specification from among multiple candidate model specifications.

In this example the HPFDIAGNOSE procedure creates three model specifications and adds them to the model repository SASUSER.MYCAT created in the previous example.

```
proc hpfdiag data=sashelp.air outest=est_three
             modelrepository=sasuser.mycat;
   id date interval=month;
   forecast air;
   transform;
   arimax;
   esm;
   ucm;
run;

proc hpfengine data=sashelp.air print=(select)
               modelrepository=sasuser.mycat inest=est_three;
   forecast air;
   id date interval=month;
run;
```

If new model specification files are added to a model repository that already exists, then the suffixed number of the model specification file name and the model selection list file name are sequential.

This example adds three model specification files (DIAG2, DIAG3, and DIAG4) to the model repository Sasuser.Mycat which already contains DIAG0 and DIAG1.

Figure 5.5 shows the three model specifications (DIAG2, DIAG3, DIAG4) found by the HPFDIAGNOSE procedure.

**Figure 5.5** Model Selection

**The HPFENGINE Procedure**

| Model Selection Criterion = RMSE | | | |
|---|---|---|---|
| **Model** | **Statistic** | **Selected** | **Label** |
| **diag2** | 10.835333 | No | ARIMA: Log( AIR ) ~ P = 1 D = (1,12) Q = (12) NOINT |
| **diag3** | 10.652082 | Yes | Log Winters Method (Multiplicative) |
| **diag4** | 10.980119 | No | UCM: Log( AIR ) = TREND + SEASON + ERROR |

## Default Settings

The following example shows the HPFDIAGNOSE procedure with the default settings. The data sets Aaa, Bbb, and Ccc are not specific data sets.

```
proc hpfdiag data=aaa print=all;
   id date interval=month;
   forecast y;
run;
```

The HPFDIAGNOSE procedure always performs the intermittency test first. If the HPFDIAGNOSE procedure determines that the series is intermittent, then the preceding example is equivalent to the following statements:

```
proc hpfdiag data=aaa print=all;
   id date interval=month;
   forecast y;
   idm intermittent=2 base=auto;
run;
```

However, if the HPFDIAGNOSE procedure determines that the series is not intermittent, then the default settings are equivalent to the following statements:

```
proc hpfdiag data=aaa print=all siglevel=0.05
             criterion=rmse holdout=0 holdoutpct=0 prefilter=yes
             back=0 errorcontrol=(severity=all stage=all);
   id date interval=month;
   forecast y;
   transform type=none;
   trend dif=auto sdif=auto;
   arimax method=minic p=(0:5)(0:2) q=(0:5)(0:2) perror=(5:10)
      outlier=(detect=maybe maxnum=2 maxpct=2
               siglevel=0.01 filter=full);
   esm method=best;
run;
```

## The Role of the IDM Statement

The HPFDIAGNOSE procedure always performs the intermittency test first regardless of which model statement is specified. The IDM statement controls only the intermittency test by using the INTERMITTENT= and BASE= options.

The following example specifies the IDM statement to control the intermittency test. If the HPFDIAGNOSE procedure determines that the series is intermittent, then an intermittent demand model is fitted to the data.

However, if the series is not intermittent, ARIMAX and exponential smoothing models are fitted to the data, even though the IDM statement is specified.

```
proc hpfdiag data=bbb print=all;
   id date interval=month;
   forecast x;
   idm intermittent=2.5 base=auto;
run;
```

The following example specifies the ESM statement. If the series is intermittent, an intermittent demand model is fitted to the data, even though the ESM statement is specified. Buf if the series is not intermittent, an ESM is fitted to the data. The same is true when the ARIMAX and UCM statements are specified.

```
proc hpfdiag data=ccc print=all;
   id date interval=month;
   forecast z;
   esm;
run;
```

# Syntax: HPFDIAGNOSE Procedure

The HPFDIAGNOSE procedure uses the following statements:

**PROC HPFDIAGNOSE** *options* ;
**ADJUST** *variable = ( variable-list ) / options* ;
**ARIMAX** *options* ;
**BY** *variables* ;
**COMBINE** *options* ;
**ESM** *option* ;
**EVENT** *event-names* ;
**FORECAST** *variables* ;
**ID** *variable INTERVAL=interval options* ;
**IDM** *options* ;
**INPUT** *variables* ;
**TRANSFORM** *options* ;
**TREND** *options* ;
**UCM** *options* ;

## Functional Summary

Table 5.1 summarizes the statements and options that control the HPFDIAGNOSE procedure.

**Table 5.1** HPFDIAGNOSE Functional Summary

| Description | Statement | Option |
|---|---|---|
| **Statements** | | |
| Specifies BY group processing | BY | |
| Specifies model combination options | COMBINE | |
| Specifies event definitions | EVENT | |
| Specifies variables to be forecast | FORECAST | |
| Specifies the time ID variable | ID | |
| Specifies input variables | INPUT | |
| Specifies log transform test and other functional transformation types | TRANSFORM | |

**Table 5.1** *continued*

| Description | Statement | Option |
|---|---|---|
| Specifies the differencing test | TREND | |
| Specifies ARIMAX model options | ARIMAX | |
| Specifies the exponential smoothing model | ESM | |
| Specifies the intermittent demand model options | IDM | |
| Specifies the unobserved components model | UCM | |
| Specifies that the dependent values be adjusted | ADJUST | |
| **Model Repository Options** | | |
| Specifies the model repository | HPFDIAGNOSE | REPOSITORY= |
| Respects the CHOOSE= option in the HPFSELECT procedure | HPFDIAGNOSE | RETAINCHOOSE= |
| Specifies the base name for all specification files | HPFDIAGNOSE | BASENAME= |
| Specifies the base name for combined model list files | HPFDIAGNOSE | COMBINEBASE= |
| Specifies the base name for model selection list files | HPFDIAGNOSE | SELECTBASE= |
| Specifies the base name for model specification files | HPFDIAGNOSE | SPECBASE= |
| **Data Set Options** | | |
| Specifies the auxiliary input data sets | HPFDIAGNOSE | AUXDATA= |
| Specifies the input data set | HPFDIAGNOSE | DATA= |
| Specifies the mapping output data set | HPFDIAGNOSE | OUTEST= |
| Specifies the events data set | HPFDIAGNOSE | INEVENT= |
| Specifies the events data set organized by BY groups | HPFDIAGNOSE | EVENTBY= |
| Specifies the output data set that contains the outliers | HPFDIAGNOSE | OUTOUTLIER= |
| Specifies the output data set that contains the information in the SAS log | HPFDIAGNOSE | OUTPROCINFO= |
| **Accumulation Options** | | |
| Specifies the length of the seasonal cycle | HPFDIAGNOSE | SEASONALITY= |
| Specifies the accumulation frequency | ID | INTERVAL= |
| Specifies the interval alignment | ID | ALIGN= |
| Specifies the starting time ID value | ID | START= |
| Specifies the ending time ID value | ID | END= |
| Specifies the accumulation statistic | ID, FORECAST, INPUT, ADJUST | ACCUMULATE= |

**Table 5.1** *continued*

| Description | Statement | Option |
|---|---|---|
| Specifies the missing value interpretation | ID, FORECAST, INPUT, ADJUST | SETMISSING= |
| Specifies the zero value interpretation | ID, FORECAST, INPUT, ADJUST | ZEROMISS= |
| Specifies how missing values are trimmed | ID, FORECAST, INPUT, ADJUST | TRIMMISS= |
| **Transformation Test Options** | | |
| Specifies the AR order for the log transformation test | TRANSFORM | P= |
| Specifies the type of the functional transformation | TRANSFORM | TYPE= |
| Specifies the method of the forecasts of the transformed series | TRANSFORM | TRANSOPT= |
| **Trend Test Options** | | |
| Specifies simple differencing | TREND | DIFF= |
| Specifies seasonal differencing | TREND | SDIFF= |
| Specifies the AR order for the augmented unit root test | TREND | P= |
| **ARIMAX Model Options** | | |
| Specifies the ARMA order selection criterion | ARIMAX | CRITERION= |
| Specifies the range of the AR orders for obtaining the error series used in the MINIC method | ARIMAX | PERROR= |
| Specifies the range of the AR orders | ARIMAX | P= |
| Specifies the range of the MA orders | ARIMAX | Q= |
| Specifies the range of the denominator orders of the transfer function | ARIMAX | DEN= |
| Specifies the range of the numerator orders of the transfer function | ARIMAX | NUM= |
| Specifies the tentative order identification method | ARIMAX | METHOD= |
| Specifies the outlier detection | ARIMAX | OUTLIER= |
| Specifies the identification order of the components | ARIMAX | IDENTIFYORDER= |
| Suppresses the intercept (constant) term | ARIMAX | NOINT |

**Table 5.1** *continued*

| Description | Statement | Option |
|---|---|---|
| **Unobserved Components Model Option** | | |
| Specifies the components to test for inclusion in the UCM | UCM | COMPONENT= |
| **Exponential Smoothing Model Option** | | |
| Specifies the method of the ESM | ESM | METHOD= |
| **Model Combination Options** | | |
| Specifies the combination weight method | COMBINE | METHOD= |
| Specifies the encompassing test | COMBINE | ENCOMPASS= |
| Specifies the forecast combination criterion | COMBINE | CRITERION= |
| Specifies the percentage of missing forecast values | COMBINE | MISSPERCENT= |
| Specifies the percentage of missing horizon values | COMBINE | HORMISSPERCENT= |
| Specifies the combination of missing forecast values | COMBINE | MISSMODE= |
| **Significance Level Option** | | |
| Specifies the significance level for diagnostic tests | HPFDIAGNOSE, TRANSFORM, TREND, ARIMAX, UCM | SIGLEVEL= |
| Specifies the significance level to control confidence limits in the model selection list files | HPFDIAGNOSE | ALPHA= |
| **Event Variable Control Option** | | |
| Specifies the maximum number of events to be selected | HPFDIAGNOSE | SELECTEVENT= |
| Specifies a hint for event inclusion in the generated model | EVENT | REQUIRED= |
| **Input Variable Control Options** | | |
| Specifies the maximum number of input variables to be selected | HPFDIAGNOSE | SELECTINPUT= |
| Specifies the transformation and differencing of the input variables | HPFDIAGNOSE | TESTINPUT= |
| Specifies the maximum missing percentage of the input variables | HPFDIAGNOSE | INPUTMISSINGPCT= |
| Specifies a hint for input variable inclusion in the generated model | INPUT | REQUIRED= |

**Table 5.1** *continued*

| Description | Statement | Option |
|---|---|---|
| **Model Selection Options** | | |
| Specifies the model selection criterion | HPFDIAGNOSE | CRITERION= |
| Specifies the forecast holdout sample size | HPFDIAGNOSE | HOLDOUT= |
| Specifies the forecast holdout sample percentage | HPFDIAGNOSE | HOLDOUTPCT= |
| Specifies data to hold back | HPFDIAGNOSE | BACK= |
| Specifies the minimum number of observations needed to fit a trend or seasonal model | HPFDIAGNOSE | MINOBS= |
| Specifies the threshold of forward selection of inputs and events | HPFDIAGNOSE | ENTRYPCT= |
| **Printing Options** | | |
| Specifies printed output for only the model specifications | HPFDIAGNOSE | PRINT=SHORT |
| Specifies printed output for PRINT=SHORT and summary of the transformation and trend tests | HPFDIAGNOSE | PRINT=LONG |
| Specifies detailed printed output | HPFDIAGNOSE | PRINT=ALL |
| Specifies control of message printing in the log | HPFDIAGNOSE | ERRORCONTROL= |
| **Data Prefilter Option** | | |
| Specifies how to handle missing and extreme values prior to diagnostic tests | HPFDIAGNOSE | PREFILTER= |
| **Miscellaneous Options** | | |
| Specifies control of exception handling | HPFDIAGNOSE | EXCEPTIONS= |

## PROC HPFDIAGNOSE Statement

**PROC HPFDIAGNOSE** *options* **;**

**PROC HPFDIAG** *options* **;**

The following options can be used in the PROC HPFDIAGNOSE or HPFDIAG statement.

**ALPHA=***value*

specifies the confidence level size to use in computing the confidence limits in the model selection list files. The ALPHA= value must be between (0, 1). The default is ALPHA=0.05, which produces 95% confidence intervals.

**AUXDATA=***SAS-data-set*

names a SAS data set that contains auxiliary input data for the procedure to use for supplying explanatory variables in a forecast. See section "AUXDATA= Data Set" on page 144 for more information.

**BACK=***number*

specifies the number of observations before the end of the data. If BACK=$n$ and the number of observations is $T$, then the first $T - n$ observations are used to diagnose a series. The default is BACK=0.

**BASENAME=***SAS-name*

prefixes any generated XML specification file name in the absence of other contextual base name options(i.e. COMBINEBASE=, SPECBASE=, SELECTBASE=). If the BASENAME=MYSPEC, then the generated specification files are named MYSPEC0, ..., MYSPEC9999999999. The default *SAS-name* uses the prefix DIAG and generates file names DIAG0, ..., DIAG9999999999. The generated files are stored in the model repository defined by the REPOSITORY= option.

**COMBINEBASE=***SAS-name*

prefixes the combined model list file name. If you specify COMBINEBASE=MYCOMB, the automatically generated combined model list files are named MYCOMB0, MYCOMB1, and so on. If not specified, combined model list files are named according to the rules for the BASENAME= option. Combined model list files are stored in the model repository defined by the REPOSITORY= option. You should note that automatic model combinations are only generated when you specify the COMBINE statement as part of the HPFDIAGNOSE procedure statement block, otherwise, specifying this option has no effect.

**CRITERION=***option*

specifies the model selection criterion to select the best model. This option is often used in conjunction with the HOLDOUT= and HOLDOUTPCT= options. The default is CRITERION=RMSE. The following statistics of fit are provided:

| | |
|---|---|
| SSE | sum of square error |
| MSE | mean squared error |
| RMSE | root mean squared error |
| UMSE | unbiased mean squared error |
| URMSE | unbiased root mean squared error |
| MAXPE | maximum percent error |
| MINPE | minimum percent error |
| MPE | mean percent error |
| MAPE | mean absolute percent error |
| MDAPE | median percent error |
| GMAPE | geometric mean percent error |
| MAPES | mean absolute error percent of standard deviation |
| MDAPES | median absolute error percent of standard deviation |
| GMAPES | geometric mean absolute error percent of standard deviation |

| | |
|---|---|
| MINPPE | minimum predictive percent error |
| MAXPPE | maximum predictive percent error |
| MPPE | mean predictive percent error |
| MAPPE | symmetric mean absolute predictive percent error |
| MDAPPE | median predictive percent error |
| GMAPPE | geometric mean predictive percent error |
| MINSPE | minimum symmetric percent error |
| MAXSPE | maximum symmetric percent error |
| MSPE | mean symmetric percent error |
| SMAPE | symmetric mean absolute percent error |
| MDASPE | median symmetric percent error |
| GMASPE | geometric mean symmetric percent error |
| MINRE | minimum relative error |
| MAXRE | maximum relative error |
| MRE | mean relative error |
| MRAE | mean relative absolute error |
| MDRAE | median relative absolute error |
| GMRAE | geometric mean relative absolute error |
| MAXERR | maximum error |
| MINERR | minimum error |
| ME | mean error |
| MAE | mean absolute error |
| MASE | mean absolute scaled error |
| RSQUARE | R-square |
| ADJRSQ | adjusted R-square |
| AADJRSQ | Amemiya's adjusted R-square |
| RWRSQ | random walk R-square |
| AIC | Akaike information criterion |
| AICC | Akaike information Corrected criterion |
| SBC | Schwarz Bayesian information criterion |
| APC | Amemiya's prediction criterion |

**DATA=***SAS data set*

specifies the name of the SAS data set that contains the time series. If the DATA= option is not specified, the most recently created SAS data set is used.

**DELAYEVENT=***number*

specifies the delay lag for the events. If the option is not specified, the delay lag for the events is set to zero by default.

**DELAYINPUT=***number*

specifies the delay lag for the inputs. If the option is not specified, the delay lag for the inputs is appropriately chosen by the procedure.

**ENTRYPCT=***number*

specifies a threshold to check the percentage increment of the criterion between two candidate models. The ENTRYPCT=value should be in (0,100); the default is ENTRYPCT=0.1.

**ERRORCONTROL=( SEVERITY= (** *severity-options***) STAGE= (** *stage-options***) MAXMESSAGE=***number***)**

allows finer control of message printing. The error severity level and the HPFDIAGNOSE procedure processing stages are set independently. The MAXMESSAGE=*number* option controls the number of messages printed. A logical 'and' is taken over all the specified options and any message.

Available *severity-options* are as follows:

| | |
|---|---|
| LOW | specifies low severity, minor issues |
| MEDIUM | specifies medium severity problems |
| HIGH | specifies severe errors |
| ALL | specifies all severity levels of LOW, MEDIUM, and HIGH options |
| NONE | specifies that no messages from PROC HPFDIAGNOSE are printed |

Available *stage-options* are as follows:

| | |
|---|---|
| PROCEDURELEVEL | specifies that the procedure stage is option processing and validation |
| DATAPREP | specifies the accumulation of data and the application of SETMISS= and ZEROMISS= options |
| DIAGNOSE | specifies the diagnostic process |
| ALL | specifies all PROCEDURELEVEL, DATAPREP, and DIAGNOSE options |

Examples are as follows.

The following statement prints high- and moderate-severity errors at any processing stage of PROC HPFDIAGNOSE:

```
errorcontrol=(severity=(high medium) stage=all)
```

The following statement prints high-severity errors only during the data preparation:

```
errorcontrol=(severity=high stage=dataprep)
```

The following statement turns off messages from PROC HPFDIAGNOSE:

```
errorcontrol=(severity=none stage=all)
errorcontrol=(maxmessage=0)
```

Each of the following statements specifies the default behavior:

```
errorcontrol=( severity=(high medium low)
               stage=(procedurelevel dataprep diagnose) )

errorcontrol=(severity=all stage=all)
```

**EVENTBY=***SAS data set*
specifies the name of the event data set that contains the events for specific BY groups that are created by DATA steps. The events in the EVENT statement are used in all BY groups, but the events in the EVENTBY= data set are used in the specific BY group.

**EXCEPTIONS=***except-option*
specifies the desired handling of arithmetic exceptions during the run. You can specify *except-option* as one of the following:

IGNORE specifies that PROC HPFDIAGNOSE stop on an arithmetic exception. No recovery is attempted. This is the default behavior if the EXCEPTIONS= option is not specified.

CATCH specifies that PROC HPFDIAGNOSE skip the generation of diagnostic output for the variable that produces the exception in the current BY group. PROC HPFDIAGNOSE generates a record to the OUTEST= data set with a blank select list name in the _SELECT_ column. The blank select list name reflects the handled exception on that combination of variable and BY group.

**HOLDOUT=***number*
specifies the size of the holdout sample to be used for model selection. The holdout sample is a subset of the dependent time series that ends at the last nonmissing observation. The statistics of a model selection criterion are computed using only the holdout sample. The default is HOLDOUT=0.

**HOLDOUTPCT=***value*
specifies the size of the holdout sample as a percentage of the length of the dependent time series. If HOLDOUT=5 and HOLDOUTPCT=10, the size of the holdout sample is $min(5, 0.1T)$ where $T$ is the length of the dependent time series with beginning and ending missing values removed. The default is HOLDOUTPCT=0.

**INEST=***SAS data set*
contains information that maps forecast variables to models or selection lists, and data set variables to model variables.

**INEVENT=***SAS data set*
specifies the name of the event data set that contains the event definitions created by the HPFEVENTS procedure. If the INEVENT= data set is not specified, only SAS predefined event definitions can be used in the EVENT statement.

For more information about the INEVENT= option, see Chapter 9, "The HPFEVENTS Procedure."

**INPUTMISSINGPCT=***value*
specifies the size of the missing observation as a percentage of the length of the input time series. If INPUTMISSINGPCT=50, then the input time series that has more than 50% missing data is ignored in the model. The default is INPUTMISSINGPCT=10.

**INSELECTNAME=***SAS-name*
specifies the name of a catalog entry that serves as a model selection list. This is the selection list that includes existing model specification files. A selection list created by the HPFDIAGNOSE procedure includes the existing model specification files.

**MINOBS=(SEASON=***number* **TREND=***number***)**

SEASON= specifies that no seasonal model is fitted to any series with fewer nonmissing observations than *number* × (season length). The value of *number* must be greater than or equal to 1. The default is *number* = 2.

TREND= specifies that no trend model is fitted to any series with fewer nonmissing observations than *number* . The value of *number* must be greater than or equal to 1. The default is *number* = 1.

**NODIAGNOSE**
specifies that the series is not diagnosed. If the INSELECTNAME= option and OUTEST= option are specified, the existing model specification files are written to the OUTEST data set.

**NOINESTOPTS**
specifies that the selection lists referred to by the INEST= option are not used in the diagnosed version.

**OUTEST=***SAS data set*
contains information that maps data set variables to model symbols and references model specification files and model selection list files.

**OUTOUTLIER=***SAS data set*
contains information that is associated with the detected outliers.

**OUTPROCINFO=** *SAS-data-set*
names the output data set to contain the summary information of the processing done by PROC HPFDIAGNOSE. It is particularly useful for easy programmatic assessment of the status of the procedure's execution via a data set instead of looking at or parsing the SAS log.

**PREFILTER=MISSING | YES | EXTREME | BOTH**
specifies how missing and extreme values are handled prior to diagnostic tests.

MISSING specifies that smoothed values for missing data are applied for tentative order selection and missing values are used for the final diagnostics.

YES specifies that smoothed values for missing data are applied to overall diagnoses. This option is the default.

EXTREME specifies that extreme values are set to missing for a tentative ARIMA model and extreme values are used for the final ARIMAX model diagnostics.

BOTH is equivalent to both YES and EXTREME.

If the input variables have missing values, they are always smoothed for the diagnostics.

**PRINT=NONE | SHORT | LONG | ALL**
specifies the print option.

| | |
|---|---|
| NONE | suppresses the printed output. This option is the default. |
| SHORT | prints the model specifications. This option also prints only the significant input variables, events, and outliers. |
| LONG | prints the summary of the transform, the stationarity test, and the determination of ARMA order in addition to all of the information printed by PRINT=SHORT. |
| ALL | prints the details of the stationarity test and the determination of ARMA order. This option prints the detail information about all input variables and events under consideration. |

**REPOSITORY=***catalog*
contains information about model specification files and model selection list files. The REPOSITORY= option can also be specified as MODELREPOSITORY=, MODELREP=, or REP=. The default model repository is Sasuser.Hpfdflt.

**RETAINCHOOSE=YES | NO**
**RETAINCHOOSE=TRUE | FALSE**
specifies that the CHOOSE= option in the HPFSELECT procedure is respected when re-diagnosing series. The default is RETAINCHOOSE=YES.

**SEASONALITY=***number*
specifies the length of the seasonal cycle. The *number* should be a positive integer. For example, SEASONALITY=3 means that every group of three observations forms a seasonal cycle. By default, the length of the seasonal cycle is 1 (no seasonality) or the length implied by the INTERVAL= option specified in the ID statement. For example, INTERVAL=MONTH implies that the length of the seasonal cycle is 12.

**SELECTINPUT=SELECT | ALL |** *number*
specifies the maximum number of the input variables to select.

| | |
|---|---|
| SELECT | selects the input variables that satisfy the criteria (noncollinearity, nonnegative delay, smaller AIC). This option is the default. |
| ALL | selects the input variables that satisfy the criteria (noncollinearity, nonnegative delay). |
| *number* | selects the best *number* input variables that satisfy the criteria (noncollinearity, nonnegative delay). |

**SELECTEVENT=SELECT | ALL |***number*
specifies the maximum number of events to select.

| | |
|---|---|
| SELECT | selects the events that satisfy the criteria (noncollinearity, smaller AIC). This option is the default. |
| ALL | selects the events that satisfy the criteria (noncollinearity). |
| *number* | selects the best *number* of events that satisfy the criteria (noncollinearity). |

**SIGLEVEL=***value*

specifies the cutoff value for all diagnostic tests such as log transformation, stationarity, tentative ARMA order selection, and significance of UCM components. The SIGLEVEL=value should be between (0,1) and SIGLEVEL=0.05 is the default. The SIGLEVEL options in TRANSFORM, TREND, ARIMAX, and UCM statements control testing independently.

**SELECTBASE=***SAS-name*

prefixes the model selection list file name. If the SELECTBASE=MYSELECT, then the model selection list files are named MYSELECT0, MYSELECT1, and so on. If not specified, model selection list files are named according to the rules defined for the BASENAME= option. Model selection list files are stored in the model repository defined by the REPOSITORY= option.

**SPECBASE=***SAS-name*

prefixes the model specification file name. If the SPECBASE=MYSPEC, then the model specification files are named MYSPEC0, MYSPECT1, and so on. If not specified, model specification files are named according to the rules defined for the BASENAME= option. Model specification files are stored in the model repository defined by the REPOSITORY= option.

**TESTINPUT=TRANSFORM | TREND | BOTH**

| | |
|---|---|
| TRANSFORM | specifies that the log transform testing of the input variables is applied independently of the variable to be forecast. |
| TREND | specifies that the trend testing of the input variables is applied independently of the variable to be forecast. |
| BOTH | specifies that the log transform and trend testing of the input variables are applied independently of the variable to be forecast. |

If this option is not specified, the same differencing is applied to the input variables as is used for the variable to be forecast, and no transformation is applied to the input variables.

## ADJUST Statement

**ADJUST** *variable = ( variable-list ) / options* **;**

The ADJUST statement lists the numeric variables in the DATA= data set whose accumulated values are used to adjust the dependent values. Adjustments are performed before diagnostics.

The numeric variable listed is the variable to which adjustments specified in that statement applies. This variable must appear in a FORECAST statement.

The numeric variables used as the source of the adjustments are listed following the parentheses. For more information see the section “Adjustment Operations” on page 125 section.

The following options can be used with the ADJUST statement.

**OPERATION=***option*

specifies how the adjustments are applied to the forecast variable. The option determines how the adjustment variables are applied to the dependent variable prior to diagnostics.

Computations with missing values are handled differently in the ADJUST statement than in other parts of SAS. If any of the adjustment operations result in a nonmissing dependent value being added to, subtracted from, divided by, or multiplied by a missing value, the nonmissing dependent value is left unchanged. Division by zero produces a missing value.

The following predefined adjustment operations are provided:

| | |
|---|---|
| NONE | No adjustment operation is performed. This is the default. |
| ADD | Variables listed in the adjustment statement are added to the dependent variable. |
| SUBTRACT | Variables listed in the adjustment statement are subtracted from the dependent variable. |
| MULTIPLY | Dependent variable is multiplied by variables listed in the adjustment statement. |
| DIVIDE | Dependent variable is divided by variables listed in the adjustment statement. |
| MIN | Dependent variable is set to the minimum of the dependent variable and all variables listed in the adjustment statement. |
| MAX | Dependent variable is set to the maximum of the dependent variable and all variables listed in the adjustment statement. |

**ACCUMULATE=***option*

See the ACCUMULATE= option in the section "ID Statement" on page 118 for more details.

**SETMISSING=***option* | *number*

See the SETMISSING= option in the section "ID Statement" on page 118 for more details.

**TRIMMISS=***option*

See the TRIMMISS= option in the section "ID Statement" on page 118 for more details.

**ZEROMISS=***option*

See the ZEROMISS= option in the section "ID Statement" on page 118 for more details.

## ARIMAX Statement

**ARIMAX** < *options* > **;**

An ARIMAX statement can be used to find an appropriate ARIMAX specification.

The HPFDIAGNOSE procedure performs the intermittency test first. If the series is intermittent, an intermittent demand model is fitted to the data and the ARIMAX statement is not applicable. If the series is not intermittent, an ARIMAX model is fitted to the data.

If a model statement is not specified, the HPFDIAGNOSE procedure diagnoses ARIMAX and exponential smoothing models if the series is not intermittent, but diagnoses an intermittent demand model if the series is intermittent.

The following options can be used in the ARIMAX statement.

**PERROR=(***number* **:** *number***)**
specifies the range of the AR order for obtaining the error series used in the MINIC method. The default is (maxp:maxp+maxq).

**P=(***number* **:** *number***) (***number* **:** *number***)**
specifies the range of the nonseasonal and seasonal AR orders. The default is (0:5)(0:2).

**Q=(***number* **:** *number***) (***number* **:** *number***)**
specifies the range of the nonseasonal and seasonal MA orders. The default is (0:5)(0:2).

**DEN=(***number* **:** *number***)**
specifies the range of the denominator order of the transfer function. The default is (0:2).

**NUM=(***number* **:** *number***)**
specifies the range of the numerator order of the transfer function. The default is (0:2).

**CRITERION=AIC | SBC**
specifies the criterion for the tentative ARMA order selection. The default is CRITERION=SBC.

**SIGLEVEL=***value*
specifies the significance level to use as a cutoff value to decide the AR and MA orders. The SIGLEVEL=value should be in (0,1). The SIGLEVEL= option overrides the value of SIGLEVEL= option in the HPFDIAGNOSE statement.

**ESTMETHOD=CLS | ULS | ML**
specifies the method for choosing the tentative ARMA orders (Choi 1992; Tsay and Tiao 1984).

| | |
|---|---|
| CLS | conditional least squares method. This option is the default. |
| ULS | unconditional least squares method |
| ML | maximum likelihood method |

**METHOD=ESACF | MINIC | SCAN**
specifies the method for choosing the tentative ARMA orders (Choi 1992; Tsay and Tiao 1984).

| | |
|---|---|
| ESACF | extended sample autocorrelation function |
| MINIC | minimum information criterion. This option is the default. |
| SCAN | smallest canonical correlation analysis |

**OUTLIER=(***options***)**
specifies outlier detection in an ARIMAX model (De Jong and Penzer 1998).

DETECT=YES includes outliers detected in a model if the model that includes the outliers is successfully diagnosed.

DETECT=MAYBE includes outliers detected in a model if the model that includes the outliers is successfully diagnosed and has a smaller criterion than the model without outliers. This option is the default.

DETECT=NO no outlier detection is performed.

FILTER=FULL | SUBSET chooses a model for outlier detection. If FILTER=FULL, then use a full model. If FILTER=SUBSET, then use a subset model that includes nonseasonal AR and MA filters only. If the data have no seasonality, then the outlier detection is not affected by the FILTER= option. FILTER=FULL is the default.

MAXNUM=*number* includes up to MAXNUM= value outliers in a model. MAXNUM=2 is the default.

MAXPCT=*value* includes up to MAXPCT= value outliers in a model. MAXPCT=2 is the default. If MAXNUM=5 and MAXPCT=10, the number of the outliers is $min(5, 0.1T)$ where $T$ is the length of the time series with beginning and ending missing values removed.

TYPE=*option* | (*options*) specifies the type of outliers. If the TYPE= option is not specified, then both AO and LS types are searched for outliers. The *options* are as follows.

AO specifies additive outliers.

LS specifies level shift outliers.

TLS(*value,...,value*) specifies temporary level shift outliers. The value is a duration of a temporary level shift and should be greater than or equal to 2. Examples are TYPE=TLS(2) and TYPE=TLS(3,9,15).

SIGLEVEL=*value* specifies the cutoff value for outlier detection. The SIGLEVEL=value should be in (0,1). The SIGLEVEL=0.01 is the default. The SIGLEVEL= option overrides the value of SIGLEVEL= option in the HPFDIAGNOSE statement.

ENTRYPCT=*number* specifies a threshold to check the percentage increment of the criterion between two candidate models. The ENTRYPCT=value should be in (0,100); the default is ENTRYPCT=0.1. The ENTRYPCT= option overrides the value of the ENTRYPCT= option in the HPFDIAGNOSE statement.

If the OUTLIER= option is not specified, the HPFDIAGNOSE performs the outlier detection with the OUTLIER=(DETECT=MAYBE MAXNUM=2 MAXPCT=2 SIGLEVEL=0.01) option as default.

If the PREFILTER=EXTREME option is specified in the PROC HPFDIAGNOSE statement and extreme values are found, then these values are potential outliers. With the PREFILTER=EXTREME option, outliers might be detected even if the DETECT=NO option is specified and more than *n* number of outliers can be detected even if the MAXNUM=*n* option is specified.

**IDENTIFYORDER=ARIMA | REG | BOTH**

**IDENTIFY=ARIMA | REG | BOTH**

specifies the identification order when inputs and events are specified.

ARIMA finds an ARIMA model for the error series first and then chooses significant inputs and events. This option is the default.

REG finds a regression model first and then decides the AR and MA polynomial orders.

BOTH fits models by using the two methods and determines the better model.

**REFINEPARMS=(** *options* **)**

specifies to refine insignificant parameters of the final model, identify the factors to refine, and identify the order of factors.

SIGLEVEL= specifies the cutoff value for all refining insignificant parameters. The SIGLEVEL=value should be between (0,1); SIGLEVEL=0.4 is the default.

FACTOR=ALL refines the parameters for all factors. This option is the default.

FACTOR=ARMA refines the parameters for ARMA factor.

FACTOR=EVENT refines the parameters for EVENT factor.

FACTOR=INPUT refines the parameters for INPUT factor.

Using parentheses, more than one option can be specified. For example, the option FACTOR=( ARMA EVENT ) refines the parameters for ARMA and EVENT.

The FIRST and SECOND options take one of the factors ARMA, EVENT, and INPUT.

FIRST= specifies the factor which refines first.

SECOND= specifies the factor which refines second.

The default order of refining is ARMA, EVENT, INPUT.

**NOINT**

**NOCONSTANT**

suppresses the intercept (constant) term.

## BY Statement

**BY** *variables* **;**

A BY statement can be used with PROC HPFDIAGNOSE to obtain separate dummy variable definitions for groups of observations defined by the BY variables.

When a BY statement appears, the procedure expects the input data set to be sorted in order of the BY variables.

If your input data set is not sorted in ascending order, use one of the following alternatives:

- Sort the data using the SORT procedure with a similar BY statement.

- Specify the BY statement option NOTSORTED or DESCENDING in the BY statement for the HPFDIAGNOSE procedure. The NOTSORTED option does not mean that the data are unsorted but rather that the data are arranged in groups (according to values of the BY variables) and that these groups are not necessarily in alphabetical or increasing numeric order.
- Create an index on the BY variables using the DATASETS procedure.

For more information about the BY statement, see in *SAS Language Reference: Concepts*. For more information about the DATASETS procedure, see the discussion in the *Base SAS Procedures Guide.*

## COMBINE Statement

**COMBINE** *options* **;**

The COMBINE statement serves two purposes. Foremost, it causes PROC HPFDIAGNOSE to automatically generate a combined model for the set of time series models it generates from its diagnosis of each series during the run. Additionally, through the various options in the COMBINE statement, you define the desired combination semantics for the combined model it generates.

The following example illustrates a typical use of the COMBINE statement:

```
proc hpfdiagnose
     data=sales.data
     rep=work.diagcomb1
     outest=diagest;
  by region store product;
  id date interval=month;
  forecast Nunits;
  input Unit_Price/required=yes;
  esm;
  arimax;
  combine method=average encompass=ols misspercent=25 hormisspercent=10;
run;
```

The presence of the COMBINE statement in the PROC HPFDIAGNOSE statement block causes the automatic generation of a combined model list that includes the time series models generated from the diagnosis of the Nunits series for each BY group. The custom model selection list generated for each BY group includes a reference to another XML specification that instructs PROC HPFENGINE to generate a combined forecast from the forecasts of the individual time series models. This is strictly an XML generation process in the context of PROC HPFDIAGNOSE. No additional computational overhead is incurred in the context of PROC HPFDIAGNOSE to generate this additional combined model list. Computational details for how PROC HPFENGINE interprets the combined model list to produce combined forecasts can be found in Chapter 20, "Forecast Combination Computational Details."

A combined model list is generated for each diagnosed series when all of the following are true:

- A COMBINE statement is specified in the PROC HPFDIAGNOSE statement block.

- The number of models generated from the series diagnosis is more than 1.
- None of the generated models is an IDM model.

The following options affect the evaluation of the combined forecast produced by the combined model list. The COMBINE statement syntax is identical to that of the HPFSELECT procedure.

**CRITERION=***option*
specifies the forecast combination criterion (statistic of fit) to be used when ranking forecast candidates in the context of the COMBINE statement. This option is often used in conjunction with the ENCOMPASS= and METHOD=RANKWGT options. The default is CRITERION=RMSE. See "Valid Statistic of Fit Names" on page 473 for valid values for *option*.

**ENCOMPASS=NONE**

**ENCOMPASS=***test-name***(***test-options***)**
specifies whether a forecast encompassing test be performed, and if so which type of test. The encompassing test attempts to eliminate forecasts from consideration that fail to add significant information to the final forecast. The default is ENCOMPASS=NONE, which specifies that no encompassing test be performed.

You can specify the following values for *test-name* and *test-options*:

OLS(ALPHA=*number*) uses an OLS-based regression test to estimate pairwise encompassing between candidate forecasts. Candidates are ranked from best to worst using the CRITERION= values. Iterating from best to worst, inferior candidates are tested with the best of the untested candidates for retention in the combined set. The significance level for the test is given by specifying ALPHA=*number* option. The default value is ALPHA=0.05 when the simple form ENCOMPASS=OLS is specified. The range is 0 to 1.

HLN(ALPHA=*number*) uses the Harvey-Leybourne-Newbold (HLN) test to estimate pairwise encompassing between candidate forecasts. Candidates are ranked from best to worst using the CRITERION= values. Iterating from best to worst, inferior candidates are tested with the best of the untested candidates for retention in the combined set. The significance level for the test is given by specifying ALPHA=*number* option. The default value is ALPHA=0.05 when the simple form ENCOMPASS=HLN is specified. The range is 0 to 1.

**HORMISSPERCENT=***number*
specifies a threshold for the percentage of missing forecast values in the combination horizon used to exclude a candidate forecast from consideration in the final combination. By default, no horizon missing percentage test is performed on candidate forecasts. If specified, the admissible range is 1 to 100. The forecast horizon is the region of time in which multistep forecasts are generated. This test and the MISSPERCENT test operate independent of each other. One or both can be specified.

**METHOD=***weight-method***(***method-options***)**
specifies the method for determining the combination weights used in the weighted average of the candidate forecasts in the combination list. The default method is METHOD=AVERAGE. The simple form METHOD=*weight-method* can be used when no weight-specific options are desired.

The following values for *weight-method* and *method-options* can be specified:

AICC(*AICC-opts*) computes the combination weights based on corrected AIC weights. See Chapter 20, "Forecast Combination Computational Details," for the mathematical details of this process. Frequently there is considerable disparity between the weights because of the exponential weighting scheme, so options are allowed to affect the scaling and to cull low-scoring candidates from consideration for computational efficiency. By default, all AICC scored candidate forecasts are combined.

Possible values for *AICC-opts* include:

ABSWGT=*number* omits computed weights with values less than the specified value. The range is 0 to 1 inclusive. The remaining weights are normalized to sum to 1.

BESTPCT=*number* retains the best $N$ of the candidates as a percentage of the total number weighted, where

$$N = \max\{\lfloor \frac{number \cdot M}{100} \rfloor, 1\} \tag{5.1}$$

and $M$ denotes the number of candidate models in the combination after any specified forecast exclusion tests have been performed.

The $N$ remaining weights are normalized to sum to 1.

BESTN=*N* retains the best $N$ of the candidates as a percentage of the total number weighted. The $N$ remaining weights are normalized to sum to 1.

LAMBDA=*number* specifies the scale factor used in the computation of the AICC weights. The default is LAMBDA=1.0, which results in the usual Akaike weights.

AVERAGE computes the simple average of the forecasts selected for combination. This is the default.

ERLS(*NLP-opts*) computes the combination weights based on a constrained least squares problem to minimize the $\ell_2$ norm of the combined forecast residuals subject to the constraint that the weights sum to 1.

LAD(*LAD-opts*) computes the weights based on a least absolute deviations measure of fit for the combined forecast. A linear program is formulated according to the *LAD-opts* to minimize an objective function expressed in terms of a absolute values of a loss series subject to constraints that the weights sum to 1 and be nonnegative. Options permitted in *LAD-opts* include OBJTYPE and ERRTYPE.

The form of the objective can be specified by the OBJTYPE= option as:

OBJTYPE=L1 specifies that the objective is an $\ell_1$ norm involving loss series.

OBJTYPE=LINF specifies that the objective is an $\ell_\infty$ norm involving the loss series.

The form of the loss series in the objective can be specified as:

ERRTYPE=ABS specifies loss series terms are deviations.

ERRTYPE=APE specifies loss series terms are percentage deviations.

ERRTYPE=RAE specifies loss series terms are relative error deviations.

NERLS(*NLP-opts*) computes the combination weights based on a constrained least squares problem to minimize the $\ell_2$ norm of the combined forecast residuals subject to the constraints that the weights sum to 1 and be nonnegative.

NRLS(*NLP-opts*) computes the combination weights based on a constrained least squares problem to minimize the $\ell_2$ norm of the combined forecast residuals subject to the constraints that the weights be nonnegative.

OLS computes the combination weights that result from the ordinary least squares problem to minimize the $\ell_2$ norm of the combined forecast residuals.

RANKWGT*(W1,...,Wn)* assigns weights by using the rank of the candidate forecasts at the time the combination is performed as determined by the COMBINE statement CRITERION= values. These weights must sum to 1. If not, they are normalized and a warning is issued. The number of values specified must agree with the number of specification names declared in the SPECIFICATION statements in the PROC HPFDIAGNOSE statement block. The weights are assigned by ranking the candidate forecasts from best to worst. The best uses the first weight, W1, and so on. The set of weights used is normalized to account for candidates that fail to forecast or for candidates that are omitted from the final combination because of any of the COMBINE statement exclusion tests.

RMSEWGT computes the combination weights based on the RMSE statistic of fit for the forecast contributors. The weights are normalized to sum to 1. See Chapter 20, "Forecast Combination Computational Details," for details.

USERDEF*(W1,...,Wn)* assigns weights by using the list of user-specified values. These weights must sum to 1. If not, they are normalized and a warning is issued. The number of values specified must agree with the number of specification names declared in the SPECIFICATION statements in the PROC HPFDIAGNOSE statement block. The weights correspond with the order of the names in the SPECIFICATION statements. The set of weights used is normalized to account for candidates that fail to forecast or for candidates that are omitted from the final combination due to any of the COMBINE statement exclusion tests.

**MISSMODE=***miss-method*

specifies a method for treating missing values in the forecast combination. In a given time slice across the combination ensemble, one or more combination contributors can have a missing value. This setting determines the treatment of those in the final combination for such time indices.

The following values for *miss-method* can be specified:

RESCALE rescales the combination weights for the nonmissing contributors at each time index to sum to 1.

MISSING generates a missing combined forecast at each time index with one or more missing contributors.

The default behavior is determined by the weight method selected as follows:

- MISSMODE=RESCALE is the default for simple average, user-specified weights, ranked user weights, ranked weights, and RMSE weights.
- MISSMODE=MISSING is the default for AICC weights, OLS weights, restricted least squares weights, and LAD weights. For OLS and NRLS you cannot specify MISSMODE=RESCALE since the estimated weights are not constrained to sum to one.

**MISSPERCENT=***number*

specifies a threshold for the percentage of missing forecast values in the combination estimation region that is used to exclude a candidate forecast from consideration in the final combination. By default, no missing percentage test is performed on candidate forecasts. If specified, the admissible range is 1 to 100. This test and the HORMISSPERCENT test operate independent of each other. One or both can be specified.

**STDERR=***stderr-method*(*stderr-options*)

**SEMODE=***stderr-method*(*stderr-options*)

specifies the method for computing the prediction error variance series. This series is used to compute the prediction standard error, which in turn is used to compute confidence bands on the combined forecast.

The simple form STDERR=*stderr-method* can be used when no method-specific options are desired.

The following values for *stderr-method* and *stderr-options* can be specifed:

STDERR=DIAG computes the prediction error variance by assuming the forecast errors at time $t$ are uncorrelated so that the simple diagonal form of $\Sigma_t$ is used. This is the default method for computing prediction error variance.

STDERR=ESTCORR computes the prediction error variance by using estimates of $\rho_{i,j,t}$, the sample cross-correlation between $e_{i,t}$ and $e_{j,t}$ over the time span $t = 1, \ldots, T$, where $t$ denotes the last time index of the actual series $y_t$. Of course, this option implies that the error series $e_{i,t}$ and $e_{j,t}$ are assumed to be jointly stationary.

STDERR=ESTCORR(TAU=$\tau$) is similar to STDERR=ESTCORR except that the cross-correlation estimates are localized to a time window of $\tau$ steps. The time span $t = 1, \ldots, T$ is quantized into segments of $\tau$ steps working from $t$ backwards for in-sample cross-correlation estimates. The cross-correlation estimates from the interval $[T - \tau, T]$ are used for the period of multistep forecasts that extend beyond time $t$.

## ESM Statement

**ESM** < *option* > **;**

An ESM statement can be used to find an appropriate exponential smoothing model specification based on the model selection criterion (McKenzie 1984).

The HPFDIAGNOSE procedure performs the intermittency test first. If the series is intermittent, an intermittent demand model is fitted to the data and the ESM statement is not applicable. If the series is not intermittent, an ESM is fitted to the data.

If a model statement is not specified, the HPFDIAGNOSE procedure diagnoses ARIMAX and exponential smoothing models if the series is not intermittent, but diagnoses an intermittent demand model if the series is intermittent.

**METHOD=BEST | BESTN | BESTS**

BEST fits the best candidate smoothing model (SIMPLE, DOUBLE, LINEAR, DAMPTREND, SEASONAL, WINTERS, ADDWINTERS). This is the default.

| | |
|---|---|
| BESTN | fits the best candidate nonseasonal smoothing model (SIMPLE, DOUBLE, LINEAR, DAMPTREND). |
| BESTS | fits the best candidate seasonal smoothing model (SEASONAL, WINTERS, ADDWINTERS). |

## EVENT Statement

**EVENT** *event-names* **;**

The EVENT statement names either event names or _ALL_. The event names identify the events in the INEVENT=data set or are the SAS predefined event keywords. _ALL_ is used to indicate that all simple events in the INEVENT=data set should be included in processing. If combination events exist in the INEVENT=data set and are to be included, then they must be specified in a separate EVENT statement. The HPFDIAGNOSE procedure does not currently process group events, although if the simple events associated with the group are defined in the INEVENT=data set, they can be included in processing, either by event name or by using _ALL_.

The EVENT statement requires the ID statement.

For more information about the EVENT statement, see Chapter 9, "The HPFEVENTS Procedure."

The following option can be used in the EVENT statement:

**REQUIRED=YES | MAYBE | NO**

| | |
|---|---|
| YES | specifies that the events be included in the model as long as the model does not fail to be diagnosed. |
| MAYBE | specifies that the events be included in the model as long as the parameters of events are significant. |
| NO | specifies that the events be included in the model as long as the parameters of events are significant and the increment of the value of criterion exceeds a threshold. The default is REQUIRED=NO. |

The same differencing is applied to the events as is used for the variables to be forecast. No functional transformations are applied to the events.

## FORECAST Statement

**FORECAST** *variables* **/** < **/** *options* > **;**

Any number of FORECAST statements can be used in the HPFDIAGNOSE procedure. The FORECAST statement lists the variables in the DATA= data set to be diagnosed. The variables are dependent or response variables that you want to forecast in the HPFENGINE procedure.

The following options can be used in the FORECAST statement.

**ACCUMULATE=***option*
See the ACCUMULATE= option in the section "ID Statement" on page 118 for more details.

**SETMISSING=***option* | *number*
See the SETMISSING= option in the section "ID Statement" on page 118 for more details.

**TRIMMISS=***option*
See the TRIMMISS= option in the section "ID Statement" on page 118 for more details.

**ZEROMISS=***option*
See the ZEROMISS= option in the section "ID Statement" on page 118 for more details.

## ID Statement

**ID** *variable options* **;**

The ID statement names a numeric variable that identifies observations in the input and output data sets. The ID variable's values are assumed to be SAS date, time, or datetime values. In addition, the ID statement specifies the (desired) frequency associated with the time series. The ID statement options also specify how the observations are accumulated and how the time ID values are aligned to form the time series. The information specified affects all variables specified in subsequent FORECAST statements. If the ID statement is specified, the INTERVAL= option must also be specified. If an ID statement is not specified, the observation number (with respect to the BY group) is used as the time ID.

For more information about the ID statement, see the section "ID Statement" on page 205 in the HPFENGINE procedure.

**ACCUMULATE=***option*
specifies how the data set observations are accumulated within each time period for the variables listed in the FORECAST statement. If the ACCUMULATE= option is not specified in the FORECAST statement, accumulation is determined by the ACCUMULATE= option of the ID statement. The ACCUMULATE= option accepts the following values: NONE, TOTAL, AVERAGE | AVG, MINIMUM | MIN, MEDIAN | MED, MAXIMUM | MAX, N, NMISS, NOBS, FIRST, LAST, STDDEV | STD, CSS, USS. The default is NONE.

**ALIGN=***option*
controls the alignment of SAS dates used to identify output observations. The ALIGN= option accepts the following values: BEGINNING | BEG | B, MIDDLE | MID | M, and ENDING | END | E. BEGINNING is the default.

**END=***option*
specifies a SAS date, datetime, or time value that represents the end of the data. If the last time ID variable value is less than the END= value, the series is extended with missing values. If the last time ID variable value is greater than the END= value, the series is truncated. For example, END="&sysdate" uses the automatic macro variable SYSDATE to extend or truncate the series to the current date. This option and the START= option can be used to ensure that data associated with each BY group contains the same number of observations.

**INTERVAL=***interval*

specifies the frequency of the input time series. For example, if the input data set consists of quarterly observations, then INTERVAL=QTR should be used. If the SEASONALITY= option is not specified, the length of the seasonal cycle is implied by the INTERVAL= option. For example, INTERVAL=QTR implies a seasonal cycle of length 4. If the ACCUMULATE= option is also specified, the INTERVAL= option determines the time periods for the accumulation of observations. See *SAS/ETS User's Guide* for the intervals that can be specified.

**SETMISSING=***option* | *number*

specifies how missing values (either actual or accumulated) are assigned in the accumulated time series for variables listed in the FORECAST statement. If the SETMISSING= option is not specified in the FORECAST statement, missing values are set based on the SETMISSING= option of the ID statement. The SETMISSING= option accepts the following values: MISSING, AVERAGE | AVG, MINIMUM | MIN, MEDIAN | MED, MAXIMUM | MAX, FIRST, LAST, PREVIOUS | PREV, NEXT. The default is MISSING.

**START=***option*

specifies a SAS date, datetime, or time value that represents the beginning of the data. If the first time ID variable value is greater than the START= value, the series is prefixed with missing values. If the first time ID variable value is less than the END= value, the series is truncated. This option and the END= option can be used to ensure that data associated with each BY group contains the same number of observations.

**TRIMMISS=***option*

specifies how missing values (either actual or accumulated) are trimmed from the accumulated time series for variables listed in the FORECAST statement. The following options are provided:

| | |
|---|---|
| NONE | No missing value trimming is applied. |
| LEFT | Beginning missing values are trimmed. |
| RIGHT | Ending missing values are trimmed. |
| BOTH | Both beginning and ending missing value are trimmed. This option is the default. |

If the TRIMMISS= option is not specified in the FORECAST statement, missing values are set based on the TRIMMISS= option of the ID statement.

**ZEROMISS=***option*

specifies how beginning and/or ending zero values (either actual or accumulated) are interpreted in the accumulated time series for variables listed in the FORECAST statement. If the ZEROMISS= option is not specified in the FORECAST statement, missing values are set based on the ZEROMISS= option of the ID statement. The following options are provided:

| | |
|---|---|
| NONE | Beginning and/or ending zeros unchanged. This option is the default. |
| LEFT | Beginning zeros are set to missing. |
| RIGHT | Ending zeros are set to missing. |
| BOTH | Both beginning and ending zeros are set to missing. |

## IDM Statement

**IDM** < *options* > **;**

An IDM statement is used to control the intermittency test. The HPFDIAGNOSE procedure performs the intermittency test first.

If the series is intermittent, an intermittent demand model is fitted to the data based on the model selection criterion. However, if the series is not intermittent, ARIMAX and exponential smoothing models are fitted to the data.

If a model statement is not specified, the HPFDIAGNOSE procedure diagnoses ARIMAX and exponential smoothing models if the series is not intermittent, but diagnoses an intermittent demand model if the series is intermittent.

**INTERMITTENT=***number*

specifies a number greater than one that is used to determine whether or not a time series is intermittent. If the median demand interval is equal to or greater than this number, then the series is assumed to be intermittent. The default is INTERMITTENT=2.

**BASE=AUTO** | *value*

specifies the base value of the time series used to determine the demand series components. The demand series components are determined based on the departures from this base value. If a base value is specified, this value is used to determine the demand series components. If BASE=AUTO is specified, the time series properties are used to automatically adjust the time series. For the common definition of Croston's method, use BASE=0, which defines departures based on zero. The default is BASE=AUTO.

**METHOD=CROSTON** | **AVERAGE** | **BEST**

specifies the smoothing model of the time series.

| | |
|---|---|
| CROSTON | The two smoothing models are used to fit the demand interval component and the demand size component. |
| AVERAGE | The single smoothing model is used to fit the average demand component. |
| BEST | Both CROSTON and AVERAGE methods are used to fit the intermittent series. The default is METHOD=BEST. |

**TRANSFORM=AUTO** | **LOG** | **NONE** | **SQRT** | **LOGISTIC** | **BOXCOX***(value)*

specifies the type of functional transformation. The following transformations are provided:

| | |
|---|---|
| AUTO | Automatically choose between NONE and LOG based on model selection criteria. The default is TYPE=AUTO. |
| LOG | logarithmic transformation |
| NONE | No transformation is applied. |
| SQRT | square-root transformation |
| LOGISTIC | logistic transformation |
| BOXCOX(*value*) | Box-Cox transformation with a parameter value where the value is between –5 and 5. The default is BOXCOX(1). |

## INPUT Statement

**INPUT** *variables* < / *options* > **;**

Any number of INPUT statements can be used in the HPFDIAGNOSE procedure. The INPUT statement lists the variables in the DATA= data set to be diagnosed as regressors. The variables are independent or predictor variables to be used to forecast dependent or response variables.

The following options can be used in the INPUT statement.

**REQUIRED=YES | MAYBE | NO**

| | |
|---|---|
| YES | specifies that the input variables be included in the model as long as the model does not fail to be diagnosed. |
| MAYBE | specifies that the input variables be included in the model as long as their parameters are significant. |
| NO | specifies that the input variables be included in the model as long as their parameters are significant and the increment of the value of criterion exceeds a threshold. The default is REQUIRED=NO. |

The same differencing is applied to the REQUIRED=YES variables as is used for the variables to be forecast. No functional transformations are applied to the REQUIRED=YES variables. The delay and numerator and denominator orders of the REQUIRED=YES variables are set to zero.

The functional transform and differencing of the REQUIRED=MAYBE or REQUIRED=NO variables depends on the request of the TESTINPUT option in the PROC HPFDIAGNOSE statement.

Either the POSITIVE or NEGATIVE option with parentheses can follow the REQUIRED= option. For example, specifying REQUIRED=YES(POSITIVE) drops the input variable from the model if its coefficient is negative, while specifying REQUIRED=YES(NEGATIVE) implies the opposite. The specification of POSITIVE or NEGATIVE does not mean that constraints are imposed during the estimation of the variable's coefficient in the model.

**ACCUMULATE=***option*

See the ACCUMULATE= option in the section "ID Statement" on page 118 for more details.

**SETMISSING=***option* |*number*

See the SETMISSING= option in the section "ID Statement" on page 118 for more details.

**TRIMMISS=***option*

See the TRIMMISS= option in the section "ID Statement" on page 118 for more details.

**ZEROMISS=***option*

See the ZEROMISS= option in the section "ID Statement" on page 118 for more details.

## TRANSFORM Statement

**TRANSFORM** < *options* > **;**

A TRANSFORM statement can be used to specify the functional transformation of the series.

The following options can be used in the TRANSFORM statement.

**P=***number*
specifies the autoregressive order for the log transform test. The default is P=$\min(2, [T/10])$ where *t* is the number of observations.

**SIGLEVEL=***value*
specifies the significance level to use as a cutoff value to decide whether or not the series requires a log transformation. The SIGLEVEL=value should be in (0,1). The SIGLEVEL= option overrides the value of SIGLEVEL= option in the HPFDIAGNOSE statement.

**TRANSOPT=MEAN | MEDIAN**
specifies whether mean or median forecasts are produced. If no transformation is applied to the series, then the mean and median forecasts are identical.

| | |
|---|---|
| MEAN | The inverse transform produces mean forecasts. This is the default. |
| MEDIAN | The inverse transform produces median forecasts. |

**TYPE=AUTO | LOG | NONE | SQRT | LOGISTIC | BOXCOX***(value)*
specifies the type of functional transformation. The following transformations are provided:

| | |
|---|---|
| AUTO | Automatically choose between NONE and LOG based on model selection criteria. If the TRANSFORM statement is specified but the TYPE= option is not specified, then TYPE=AUTO is the default. |
| LOG | logarithmic transformation |
| NONE | No transformation is applied. If the TRANSFORM statement is not specified, TYPE=NONE is the default. |
| SQRT | square-root transformation |
| LOGISTIC | logistic transformation |
| BOXCOX(*value*) | Box-Cox transformation with a parameter value where the value is between –5 and 5. The default is BOXCOX(1). |

## TREND Statement

**TREND** < *options* > **;**

A TREND statement can be used to test whether the dependent series requires simple or seasonal differencing, or both. The augmented Dickey-Fuller test (Dickey and Fuller 1979) is used for the simple unit root test.

If the seasonality is less than or equal to 12, the seasonal augmented Dickey-Fuller (ADF) test (Dickey, Hasza, and Fuller 1984) is used for the seasonal unit root test. Otherwise, an AR(1) seasonal dummy test is used.

The joint simple and seasonal differencing test uses the Hasza-Fuller test (Hasza and Fuller 1979, 1984) in the special seasonality. Otherwise, proceed with the ADF test and the season dummy test.

The following options can be used in the TREND statement.

**DIFF=AUTO** | **NONE** | *number* | **(0 :** *number***)**

| | |
|---|---|
| AUTO | tests for simple differencing. This option is the default. |
| NONE | specifies that no simple differencing be used. |
| *number* | specifies the simple differencing order. The option *number*=1 means $(1-B)y_t$ and *number*=2 means $(1-B)^2 y_t$. |
| (0 : *number*) | specifies the range of simple differencing order for testing. The option *number* can be 0, 1, or 2. |

**SDIFF=AUTO** | **NONE** | *number*

| | |
|---|---|
| AUTO | tests for seasonal differencing. This option is the default. |
| NONE | specifies that no seasonal differencing be used. |
| *number* | specifies the seasonal differencing order. The option *number*=1 means $(1-B^s)y_t$ and *number*=2 means $(1-B^s)^2 y_t$ where $s$ is the seasonal period. |

**P=***number*

specifies the autoregressive order for the augmented unit root tests and a seasonality test. The default is P=$\min(5, [T/10])$ where $t$ is the number of observations.

**SIGLEVEL=***value*

specifies the significance level to use as a cutoff value to decide whether or not the series needs differencing. The SIGLEVEL=value should be in (0,1). The SIGLEVEL= option overrides the value of SIGLEVEL= option in the HPFDIAGNOSE statement.

## UCM Statement

**UCM** < *options* > **;**

A UCM statement can be used to find an appropriate unobserved component model specification (Harvey 1989, 2001; Durbin and Koopman 2001).

The HPFDIAGNOSE procedure performs the intermittency test first. If the series is intermittent, an intermittent demand model is fitted to the data and the UCM statement is not applicable. If the series is not intermittent, an unobserved component model is fitted to the data.

The following options can be used in the UCM statement.

**COMPONENT=(***components***)**

| | |
|---|---|
| ALL | tests which components and/or variances are significant in the model. This option is the default. When the series has the seasonality information, the IRREGULAR, LEVEL, SLOPE, and SEASON components are included. Otherwise, the IRREGULAR, LEVEL, SLOPE, and CYCLE components are included. |
| AUTOREG | tests if an autoregreesive component is significant in the model. |
| CYCLE | tests if two cycle components are significant in the model. The two CYCLE components are included and the LEVEL component is added. When the series has the seasonality information, the CYCLE component is not tested. |
| DEPLAG | tests if a dependent lag component is significant in the model. Only the order 1 is included. |
| IRREGULAR | tests if an irregular component is significant in the model. |
| LEVEL | tests if a level component is significant in the model. |
| SEASON | tests if a season component is significant in the model. When the series has the seasonality information, the SEASON component is not tested. |
| SLOPE | tests if a slope component is significant in the model. The LEVEL component is added. |

**SIGLEVEL=***value*

specifies the significance level to use as a cutoff value to decide which component and/or variances are significant. The SIGLEVEL=value should be in (0,1). The SIGLEVEL= option overrides the value of SIGLEVEL= option in the HPFDIAGNOSE statement.

**REFINEPARMS= ( SIGLEVEL= | FACTOR=(ALL | EVENT | INPUT) | FIRST=EVENT | INPUT)**

specifies to refine insignificant parameters of the final model, identify the factors to refine, and identify the order of factors.

| | |
|---|---|
| SIGLEVEL= | specifies the cutoff value for all refining insignificant parameters. The SIGLEVEL=value should be between (0,1); SIGLEVEL=0.4 is the default. |
| FACTOR=ALL | refines the parameters for all factors. This option is the default. |
| FACTOR=EVENT | refines the parameters for EVENT factor. |
| FACTOR=INPUT | refines the parameters for INPUT factor. |

Using parentheses, more than one option can be specified. For example, the option FACTOR=( EVENT INPUT ) refines the parameters for ARMA and EVENT.

| | |
|---|---|
| FIRST= | specifies the factor which refines first. |

The default order of refining is EVENT, INPUT.

# Details: HPFDIAGNOSE Procedure

## Adjustment Operations

Preadjustment variables can be used to adjust the dependent series prior to model diagnostics.

If $y_t$ is the dependent series and $z_{i,t}$ for $i = 1, \ldots, M$ are the $M$ adjustment series, then the adjusted dependent series, $w_t$, is

$$\begin{aligned} w_t^1 &= op_i(y_t, z_{i,t}) \\ w_t^i &= op_i(w_t^{i-1}, z_{i,t}) \text{ for } 1 < i \leq M \\ w_t &= w_t^M \end{aligned}$$

where $op_i$ represents the $i$th preadjustment operator and $w_t^i$ is the $i$th adjusted dependent series. The preadjustment operators are nested and applied sequentially from $i = 1, \ldots, M$.

## Data Preparation

The HPFDIAGNOSE procedure does not use missing data at the beginning and/or end of the series.

Missing values in the middle of the series to be forecast would be handled with the PREFILTER=MISSING or PREFILTER=YES option. The PREFILTER=MISSING option uses smoothed values for missing data for tentative order selection in the ARIMAX modeling and for tentative components selection in the UCM modeling, but it uses the original values for the final diagnostics. The PREFILTER=YES option uses smoothed values for missing data and for all diagnostics.

Extreme values in the middle of the series to be forecast can be handled with the PREFILTER=EXTREME option in the ARIMA modeling. The HPFDIAGNOSE procedure replaces extreme values with missing values when determining a tentative ARIMA model, but it uses the original values for the final diagnostics. The PREFILTER=EXTREME option detects extreme values if the absolute values of residuals are greater than 3 × STDDEV from a proper smoothed model.

If there are missing values in the middle of data for the input series, the procedure uses an interpolation method based on exponential smoothing to fill in the missing values.

The following data set provides a scenario for explaining the PREFILTER=EXTREME option.

```
data air_extreme;
   set sashelp.air;
   if _n_ = 30  then air = 500;
   if _n_ = 50  then air = 500;
   if _n_ = 100 then air = 700;
run;
```

In the following SAS statements, the HPFDIAGNOSE procedure diagnoses the new data set Air_Extreme without the PREFILTER=EXTREME option.

```
proc hpfdiagnose data=air_extreme print=short;
   id date interval=month;
   forecast air;
   transform;
   arimax;
run;
```

In Figure 5.6, the ARIMA$(0,1,1)(2,0,0)_{12}$ model is diagnosed for the time series. The diagnosed model has no seasonal differencing and is quite different from the model in Figure 5.1. The three extreme values mislead the model diagnostic tests.

**Figure 5.6** ARIMAX Model Specification without Outliers

**The HPFDIAGNOSE Procedure**

**ARIMA Model Specification**

| Variable | Functional Transform | Constant | p | d | q | P | D | Q | Seasonality | Model Criterion | Statistic | Status |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AIR | LOG | YES | 0 | 1 | 1 | 2 | 0 | 0 | 12 | RMSE | 56.6436 | OK |

In the following SAS statements, the HPFDIAGNOSE procedure diagnoses the new data set Air_Extreme with the PREFILTER=EXTREME option.

```
proc hpfdiagnose data=air_extreme
                 prefilter=extreme
                 print=short;
   id date interval=month;
   forecast air;
   transform;
   arimax;
run;
```

In Figure 5.7, the ARIMA$(0,1,1)(0,1,1)_{12}$ model is diagnosed for the time series. The required seasonal differencing is detected.

**Figure 5.7** ARIMAX Model Specification without Outliers

**The HPFDIAGNOSE Procedure**

**ARIMA Model Specification**

| Variable | Functional Transform | Constant | p | d | q | P | D | Q | Seasonality | Model Criterion | Statistic | Status |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AIR | LOG | NO | 0 | 1 | 1 | 0 | 1 | 1 | 12 | RMSE | 53.9316 | OK |

Figure 5.8 shows that the three extreme values are detected as outliers.

**Figure 5.8** Outlier Information

**ARIMA Outlier Selection**

| Variable | Type | Obs | Time | Chi-Square | Approx Pr > ChiSq |
|---|---|---|---|---|---|
| **AIR** | AO | 30 | JUN1951 | 223.88 | <.0001 |
| | AO | 50 | FEB1953 | 291.81 | <.0001 |
| | AO | 100 | APR1957 | 261.52 | <.0001 |

After the three extreme values are included in the ARIMAX model, Figure 5.9 shows that the statistic of the model selection criterion drops substantially.

**Figure 5.9** ARIMAX Model Specification with Outliers

**ARIMA Model Specification After Adjusting for Outliers**

| Variable | Functional Transform | Constant | p | d | q | P | D | Q | Seasonality | Outlier | Model Criterion | Statistic | Status |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **AIR** | LOG | NO | 0 | 1 | 1 | 0 | 1 | 1 | 12 | 3 | RMSE | 10.7626 | OK |

## Functional Transformation

The log transform test compares the MSE or MAPE value after fitting an AR($p$) model to the original data and to the logged data. If the MSE or MAPE value is smaller for the AR($p$) model fitted to the logged data, then the HPFDIAGNOSE procedure performs the log transformation.

The next two sets of SAS statements specify the same log transformation test.

```
proc hpfdiag data=sashelp.air print=all;
   id date interval=month;
   forecast air;
   transform;
   arimax;
run;

proc hpfdiag data=sashelp.air print=all;
   id date interval=month;
   forecast air;
   arimax;
   transform type=auto;
run;
```

The "Functional Transformation Test" table shown in Figure 5.10 states that the airline data requires a log transformation.

**Figure 5.10** Log Transformation Test

**The HPFDIAGNOSE Procedure**

| Functional Transformation Test | |
|---|---|
| Variable | Functional Transform |
| AIR | LOG |

## Stationarity Test

The stationarity test decides whether the data requires differencing. Note that $d$ is the simple differencing order, and $D$ is the seasonal differencing order.

The next two sets of SAS statements specify the same trend test.

```
proc hpfdiag data=sashelp.air print=all;
   id date interval=month;
   forecast air;
   transform;
   arimax;
run;

proc hpfdiag data=sashelp.air print=all;
   id date interval=month;
   forecast air;
   transform;
   arimax;
   trend diff=auto sdiff=auto;
run;
```

### Simple Differencing Order

The simple augmented Dickey-Fuller test is used to determine the simple differencing order.

If there is no unit root, then the HPFDIAGNOSE procedure sets $d = 0$.

If there is a unit root, then the double unit root test is applied. If there is a double unit root, then the HPFDIAGNOSE procedure sets $d = 2$; otherwise, $d = 1$.

Figure 5.11 and Figure 5.12 show that the series needs simple differencing because the null hypothesis test probability is greater than SIGLEVEL=0.05.

**Figure 5.11** Dickey-Fuller Unit Root Test

| Dickey-Fuller Unit Root Test | | | | |
|---|---|---|---|---|
| Type | Rho | Pr < Rho | Tau | Pr < Tau |
| Zero Mean | 0.22 | 0.7335 | 1.38 | 0.9580 |
| Single Mean | -2.42 | 0.7255 | -1.11 | 0.7118 |
| Trend | 294.41 | 0.9999 | -6.42 | <.0001 |

**Figure 5.12** Summary of Dickey-Fuller Unit Root Test

| Dickey-Fuller Unit Root Test Summary | | | | |
|---|---|---|---|---|
| Variable | Seasonality | Zero Mean | Mean | Trend |
| AIR | 1 | YES | YES | NO |

## Seasonal Differencing Order

The seasonal augmented Dickey-Fuller test is used to identify the seasonal differencing order. If the seasonality is greater than 12, the season dummy regression test is used. If there is no seasonal unit root, the HPFDIAGNOSE procedure sets $D = 0$. If there is a seasonal unit root, the HPFDIAGNOSE procedure sets $D = 1$.

Figure 5.13 and Figure 5.14 show that the series needs seasonal differencing because the null hypothesis test probability is greater than SIGLEVEL=0.05.

**Figure 5.13** Seasonal Dickey-Fuller Unit Root Test

| Seasonal Dickey-Fuller Unit Root Test(Seasonality=12) | | | | |
|---|---|---|---|---|
| Type | Rho | Pr < Rho | Tau | Pr < Tau |
| Zero Mean | -0.47 | 0.5112 | -0.13 | 0.4970 |
| Single Mean | -6.51 | 0.2186 | -1.59 | 0.1101 |

**Figure 5.14** Summary of Seasonal Dickey-Fuller Unit Root Test

| Seasonal Dickey-Fuller Unit Root Test Summary | | | | |
|---|---|---|---|---|
| Variable | Seasonality | Zero Mean | Mean | Trend |
| AIR | 12 | YES | YES | |

## Joint Differencing Orders

Hasza-Fuller (Hasza and Fuller 1979, 1984) proposed the joint unit roots test. If the seasonality is less than or equal to 12, use these tests. If there is a joint unit root, then the HPFDIAGNOSE procedure sets $D = 1$ and $d = 1$.

Figure 5.15 and Figure 5.16 show that the series needs both simple and seasonal differencing because the null hypothesis test probability is greater than SIGLEVEL=0.05.

**Figure 5.15** Joint Unit Root Test

| Joint Unit Root Test | | | | | |
|---|---|---|---|---|---|
| | | Critical Values | | | |
| Type | F Value | 90% | 95% | 99% | Approx Pr > F |
| Zero Mean | 3.2684 | 2.5695 | 3.2600 | 4.8800 | 0.0466 |
| Single Mean | 3.8360 | 5.1409 | 6.3473 | 8.8400 | 0.1476 |
| Trend | 3.0426 | 7.2635 | 8.6820 | 10.7600 | 0.2896 |

**Figure 5.16** Summary of Joint Unit Root Test

**Joint Unit Root Test Summary**

| Variable | Seasonality | Zero Mean | Mean | Trend |
|---|---|---|---|---|
| **AIR** | 1, 12 | NO | YES | |

### Seasonal Dummy Test

If the seasonality is greater than 12, the seasonal dummy test is used to decide the seasonal differencing order.

The seasonal dummy test compares the criterion (AIC) of two AR(1) models and the joint significance of the seasonal dummy parameters, where one has seasonal dummy variables and the other does not have the seasonal dummy variables.

## ARMA Order Selection

### Tentative Simple Autoregressive and Moving-Average Orders

The tentative simple autoregressive and moving-average orders (AR=$p^*$ and MA=$q^*$) are found using the ESACF, MINIC, or SCAN method.

The next two sets of SAS statements result in the same diagnoses.

```
proc hpfdiag data=sashelp.air print=all;
   id date interval=month;
   forecast air;
   transform;
   arimax;
run;

proc hpfdiag data=sashelp.air print=all;
   id date interval=month;
   forecast air;
   transform;
   arimax method=minic p=(0:5) q=(0:5) criterion=sbc;
run;
```

Figure 5.17 shows the minimum information criterion among the AR and MA orders. The AR=3 and MA=0 element has the smallest value in the table.

**Figure 5.17** Minimum Information Criterion

**Minimum Information Criterion**

| Lags | MA 0 | MA 1 | MA 2 | MA 3 | MA 4 | MA 5 |
|---|---|---|---|---|---|---|
| **AR 0** | -6.20852 | -6.30537 | -6.29093 | -6.3145 | -6.28459 | -6.26408 |
| **AR 1** | -6.31395 | -6.28157 | -6.26557 | -6.28327 | -6.25263 | -6.23399 |
| **AR 2** | -6.29952 | -6.26759 | -6.24019 | -6.24605 | -6.21542 | -6.20335 |
| **AR 3** | -6.33026 | -6.29846 | -6.26559 | -6.23155 | -6.2356 | -6.22296 |
| **AR 4** | -6.31801 | -6.28102 | -6.24678 | -6.24784 | -6.21578 | -6.19315 |
| **AR 5** | -6.29745 | -6.2603 | -6.22433 | -6.2265 | -6.19536 | -6.15861 |

### Simple Autoregressive and Moving-Average Orders

The simple autoregressive and moving-average orders ($p$ and $q$) are found by minimizing the SBC/AIC values from the models among $0 \leq p \leq p^*$ and $0 \leq q \leq q^*$ where $p^*$ and $q^*$ are the tentative simple autoregressive and moving-average orders.

### Seasonal Autoregressive and Moving-Average Orders

The seasonal AR and MA orders ($p$ and $Q$) are found by minimizing the SBC/AIC values from the models among $0 \leq P \leq 2$ and $0 \leq Q \leq 2$.

### Constant

In order to determine whether the model has a constant, two models are fitted: $(p, d, q)(P, D, Q)_s$ and $C + (p, d, q)(P, D, Q)_s$. The model with the smaller SBC/AIC value is chosen.

### Estimation Method

The ARIMA model uses the conditional least squares estimates for the parameters.

Figure 5.18 shows that the simple AR and MA orders are reduced to $p = 1$ and $q = 0$ from $p^* = 3$ and $q^* = 0$. The seasonal AR and MA orders are $P = 0$ and $Q = 1$. The selected model does not have a constant term.

**Figure 5.18** ARIMAX Specification

**ARIMA Model Specification**

| Variable | Functional Transform | Constant | p | d | q | P | D | Q | Seasonality | Model Criterion | Statistic | Status |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **AIR** | LOG | NO | 1 | 1 | 0 | 0 | 1 | 1 | 12 | RMSE | 10.8353 | OK |

## Transfer Functions in an ARIMAX Model

A transfer function filter has delay, numerator, and denominator parameters. Set $(b, k, r)$ where $b$ is the delay, $k$ is the numerator order, and $r$ is the denominator order.

### Functional Transformation for Input Variables

The default of functional transformation for the inputs is no transformation. The TESTINPUT=TRANSFORM option specifies that the same functional transformation is applied to the inputs as is used for the variable to be forecast.

Using the TESTINPUT=TRANSFORM option, you can test whether the log transformation is applied to the inputs.

### Simple and Seasonal Differencing Orders for Input Variables

The default of the simple and seasonal differencing for the inputs is the same as the simple and seasonal differencing applied to the variable to be forecast.

Using the TESTINPUT=TREND option, you can test whether the differencing is applied to the inputs.

## Cross-Correlations between Forecast and Input Variables

The cross-correlations between the variable $y_t$ to be forecast and each input variable $x_{it}$ are used to identify the delay parameters. The following steps are used to prewhiten the variable to be forecast in order to identify the delay parameter $b$.

1. Find an appropriate ARIMA model for $x_{it}$ and estimate the residual of $x_{it}$ ($e_{it}^{x}$).
2. Prewhiten $y_t$ using this model and get the residual of $y_t$ ($e_{it}^{y}$).
3. Compute the cross-correlations between $e_{it}^{x}$ and $e_{it}^{y}$ and find the first significant lag that is zero or larger. If no delay lag is significant, the variable $x_{it}$ is not included in the model.

## Simple Numerator and Denominator Orders

The high-order lag regression model and the transfer function model are compared to identify the simple numerator and denominator orders.

Fit the high-order lag regression model (lag=15) and get the coefficients. Fit the transfer function $C + (b, k, r)$ where $C$ is a constant term, $b$ is the delay parameter found in the previous section, $0 \leq k \leq 2$, and $0 \leq r \leq 2$, and get the impulse weight function (lag=15) of the transfer model. Compare the pattern of the coefficients from the high-order regression model and the transfer model.

The following SAS statements shows how to select significant input variables.

```
proc hpfdiag data=sashelp.citimon(obs=141) print=all;
   forecast conb;
   input cciutc eec eegp exvus fm1 fm1d82;
   transform;
   arimax;
run;
```

The "ARIMA Input Selection" table shown in Figure 5.19 states that the Eegp input variable is selected in the model with difference $d = 1$ and delay $b = 8$. Other input variables are not selected because of either unstable or insignificant status.

**Figure 5.19** ARIMA Input Selection

**The HPFDIAGNOSE Procedure**

**ARIMA Input Selection**

| Input Variable | Selected | Functional Transform | d | Delay | Numerator | Denominator | Status |
|---|---|---|---|---|---|---|---|
| CCIUTC | NO | NONE | 1 | 2 | 0 | 0 | Not Improved |
| EEC | NO | | | | | | Not Causal |
| EEGP | YES | NONE | 1 | 8 | 0 | 0 | OK |
| EXVUS | NO | | | | | | Not Causal |
| FM1 | NO | | | | | | Not Causal |
| FM1D82 | NO | | | | | | Not Causal |

## Outliers

Outlier detection is the default in the ARIMAX modeling.

There are two types of outliers: the additive outlier (AO) and the level shift (LS). For each detected outlier, dummy regressors or indicator variables are created. The ARIMAX model and the dummy regressors are fitted to the data.

The detection of outliers follows a forward method. First find a significant outlier. If there are no other significant outliers, detecting outlier stops at this point. Otherwise, include this outlier into a model as an input and find another significant outlier.

The same functional differencing is applied to the outlier dummy regressors as is used for the variable to be forecast.

The following data comes from Hillmer, Larcker, and Schroeder (1983).

```
data hardware;
   input hardware @@;
   label hardware="Wholesale Sales of Hardware";
   date=intnx('month','01jan67'd,_n_-1);
   format date monyy.;
datalines;
 626  614  689  686  723  778  711  824  793  831  775  689

   ... more lines ...
```

The next two sets of SAS statements result in the same outlier analysis.

```
proc hpfdiag data=hardware print=short;
   id date interval=month;
   forecast hardware;
   transform;
   arimax;
run;

proc hpfdiag data=hardware print=short;
   id date interval=month;
   forecast hardware;
   transform;
   arimax outlier=(detect=maybe maxnum=2 maxpct=2 siglevel=0.01);
run;
```

Figure 5.20 shows that the two level shifts (LS) occurred at the 96th (DEC1974) and 99th (MAR1975) observations.

**Figure 5.20** Outlier Information

**The HPFDIAGNOSE Procedure**

**ARIMA Outlier Selection**

| Variable | Type | Obs | Time | Chi-Square | Approx Pr > ChiSq |
|---|---|---|---|---|---|
| hardware | LS | 99 | MAR1975 | 25.73 | <.0001 |
| | LS | 96 | DEC1974 | 29.64 | <.0001 |

Figure 5.21 shows the ARIMA model specification with two outliers included in the model.

**Figure 5.21** ARIMAX Specification

**ARIMA Model Specification After Adjusting for Outliers**

| Variable | Functional Transform | Constant | p | d | q | P | D | Q | Seasonality | Outlier | Model Criterion | Statistic | Status |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| hardware | NONE | NO | 2 | 1 | 1 | 2 | 1 | 1 | 12 | 2 | RMSE | 45.9477 | OK |

## Intermittent Demand Model

The HPFDIAGNOSE procedure selects an appropriate intermittent demand model (IDM) based on the model selection criterion.

If a series is intermittent or interrupted, a proper IDM is selected by either individually modeling both the demand interval and size component or by jointly modeling these components by using the average demand component (demand size divided by demand interval).

The following example prints the diagnostics of an intermittent demand series. The INTERMITTENT=2.5 and BASE=0 are specified.

```
data sales;
   input hubcaps @@;
datalines;
0 1 0 0 0 1 0 0 0 0 0 2 0 4 0 0 0 0 1 0
;

proc hpfdiag data=sales print=all;
   forecast hubcaps;
   transform;
   idm intermittent=2.5 base=0;
run;
```

Output 5.22 shows that the variable to be forecast is an intermittent demand series. The interval/size demand model and average demand model were diagnosed for the time series. The value of the model selection criterion of the interval/size demand model is smaller than that of the average demand model.

**Figure 5.22** Intermittent Demand Model Specification

**The HPFDIAGNOSE Procedure**

| Intermittent Demand Model Specification | | | | | | |
|---|---|---|---|---|---|---|
| Variable | Demand Model | Functional Transform | Selected Model | Component | Model Criterion | Statistic |
| **hubcaps** | INTERVAL | NONE | SIMPLE | LEVEL | RMSE | 0.8428 |
| | SIZE | LOG | SIMPLE | LEVEL | | |
| | AVERAGE | LOG | SIMPLE | LEVEL | | 0.8736 |

## Exponential Smoothing Model

The HPFDIAGNOSE procedure selects an appropriate exponential smoothing model (ESM) based on the model selection criterion.

The following statements print the ESM specification.

```
proc hpfdiag data=sashelp.gnp print=short;
   id date interval=qtr;
   forecast gnp;
   transform;
   esm;
run;
```

The ESM specification in Figure 5.23 states that the damp-trend exponential smoothing model was automatically selected.

**Figure 5.23** ESM Specification

**The HPFDIAGNOSE Procedure**

| Exponential Smoothing Model Specification | | | | | |
|---|---|---|---|---|---|
| Variable | Functional Transform | Selected Model | Component | Model Criterion | Statistic |
| **GNP** | NONE | DAMPTREND | LEVEL | RMSE | 22.0750 |
| | | | TREND | | |
| | | | DAMP | | |

## Unobserved Components Model

The UCM statement is used to find the proper components among the level, trend, seasonal, cycles, and regression effects.

### Differencing Variables in a UCM

The variable to be forecast and the events are not differenced regardless of the result of the TREND statement. Differencing of the input variables follows the result of the option TESTINPUT=TREND or TESTINPUT=BOTH.

### Transfer Function in a UCM

The functional transformation, simple and seasonal differencing, and delay parameters for the transfer function in a UCM are the same as those that are used for the transfer function in an ARIMAX model.

The series that consists of the yearly river flow readings of the Nile, recorded at Aswan (Cobb 1978), is studied. The data consists of readings from the years 1871 to 1970.

The following DATA step statements read the data in a SAS data set and create dummy inputs for the shift in 1899 and the unusual years 1877 and 1913.

```
data nile;
   input riverFlow @@;
   year = intnx( 'year', '1jan1871'd, _n_-1 );
   format year year4.;
datalines;
1120 1160  963 1210 1160 1160  813 1230 1370 1140

   ... more lines ...
```

The series is known to have had a shift in the level starting at the year 1899, and the years 1877 and 1913 are suspected to be outlying points. The following SAS statements create the Nile_Data data set with the Shift1899, Event1877, and Event1913 variables.

```
data nile_data;
   set nile;
   if year >= '1jan1899'd then
      Shift1899 = 1.0;
   else
      Shift1899 = 0;
   if year = '1jan1913'd then
      Event1913 = 1.0;
   else
      Event1913 = 0;
   if year = '1jan1877'd then
      Event1877 = 1.0;
   else
      Event1877 = 0;
run;
```

The following SAS statements print the diagnoses of the UCM specification.

```
proc hpfdiag data=nile_data print=short;
   id year interval=year;
   forecast riverFlow;
   input Shift1899 Event1913 Event1877;
   transform;
   ucm;
run;
```

Figure 5.24 shows the three significant inputs chosen.

**Figure 5.24** UCM Input Selection

**The HPFDIAGNOSE Procedure**

**UCM Input Selection**

| Input Variable | Selected | Functional Transform | d | Delay |
|---|---|---|---|---|
| **Shift1899** | YES | NONE | 0 | 0 |
| **Event1913** | YES | NONE | 0 | 0 |
| **Event1877** | YES | NONE | 0 | 2 |

Figure 5.25 shows the UCM specification for the Nile data. The data has significant irregular and level components, along with three significant inputs.

**Figure 5.25** UCM Specification

**Unobserved Components Model(UCM) Specification**

| Variable | Functional Transform | Component | Selected | Stochastic | Period | Model Criterion | Statistic | Status |
|---|---|---|---|---|---|---|---|---|
| **riverFlow** | NONE | IRREGULAR | YES | YES | | RMSE | 117.13 | OK |
| | | LEVEL | YES | NO | | | | |
| | | SLOPE | NO | | | | | |
| | | CYCLE1 | NO | | | | | |
| | | CYCLE2 | NO | | | | | |
| | | INPUT | 3 | | | | | |

The following example has the same results as Figure 5.24. The COMPONENTS= option in the UCM statement requests that level and irregular components only be considered. Figure 5.26 shows the result from the execution of this example.

```
proc hpfdiag data=nile_data print=short;
   id year interval=year;
   forecast riverFlow;
   transform;
   input Shift1899 Event1913 Event1877;
   ucm component=(level irregular);
run;
```

**Figure 5.26** UCM Specification

**The HPFDIAGNOSE Procedure**

**Unobserved Components Model(UCM) Specification**

| Variable | Functional Transform | Component | Selected | Stochastic | Model Criterion | Statistic | Status |
|---|---|---|---|---|---|---|---|
| **riverFlow** | NONE | IRREGULAR | YES | YES | RMSE | 117.13 | OK |
| | | LEVEL | YES | NO | | | |
| | | INPUT | 3 | | | | |

## Values of Status

The meaning of the different values of status in the output is summarized as follows:

| | |
|---|---|
| OK | The model fits successfully. |
| All Missing Obs | All series values are missing. |
| All Same Values | All series values are identical. |
| Collinearity | Multi-collinearity between input series. |
| Not Causal | A delay parameter is negative. |
| Insignificant | Some of the parameters are insignificant. |
| X Model Failed | Prewhitening model of input series failed. |
| Y Model Failed | The g(y)=f(x) model failed. |
| Not Improved | Input series does not improve benchmarking model. |
| Not One of Best | When the SELECTINPUT=*n* and SELECTEVENT=*n* are specified, it is not one of the best first *n* inputs and events. |
| Unstable Model | The model is unstable. |
| May Not Converge | The model might not converge. |
| Small Variance | The series values have very small variance. |
| Singularity | The model is singular. |
| Modeling Error | A model cannot be fitted to the data. |
| Extreme Value | The series values are extremely large. |
| Error Transform | The data transformation failed. |
| Lack of Memory | The memory is insufficient. |
| Not Enough Data | The number of observations is insufficient. |
| Bad Arguments | The arguments are incorrect. |
| Non Positive Obs | The series values are either zero or negative. |

The following are the messages for the intermittent model:

| | |
|---|---|
| No Demand | There is no recorded demand. |
| Not Fitted Model | The model cannot be fitted to the data. |
| Not Selected | The model cannot be selected. |
| Not Predicted | The model cannot be predicted. |
| Not Initialized | The model cannot be initialized. |
| Negative Demand | The demand size is negative. |

## Holdout Sample

A holdout sample is useful to find models that have better out-of-sample forecasts. If the HOLDOUT= or HOLDOUTPCT= option is specified, the model selection criterion is computed using only the holdout sample region.

```
proc hpfdiag data=sashelp.air print=short holdout=10;
   id date interval=month;
   forecast air;
   transform;
   arimax;
run;
```

The ARIMA model specification in Figure 5.27 shows that the log test, trend test, and selection of ARMA orders use only the first part of the series and exclude the last 10 observations that were specified as the holdout sample. The statistic of the model selection criterion is computed using only the last 10 observations that were specified as the holdout sample.

**Figure 5.27** Use HOLDOUT Option

**The HPFDIAGNOSE Procedure**

**ARIMA Model Specification**

| Variable | Functional Transform | Constant | p | d | q | P | D | Q | Seasonality | Model Criterion | HoldOut Statistic | HoldOut Sample | Status |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AIR | LOG | NO | 1 | 1 | 0 | 0 | 1 | 1 | 12 | RMSE | 16.3008 | 10 | OK |

## EVENTS

Calendar effects such as holiday and trading day are defined by the HPFEVENTS procedure or predefined event keywords.

The HPEVENTS procedure creates the OUT= data set for the event definitions, and the HPFDIAGNOSE procedure uses these event definitions by specifying the INEVENT= option in the ARIMAX or UCM model.

### Events in an ARIMAX Model

The simple and seasonal differencing for the events in an ARIMAX are the same as those that are used for the variable to be forecast.

No functional transformations are applied to the events.

### Events in a UCM

The simple and seasonal differencing for the events in a UCM are not applied to the events.

No functional transformations are applied to the events.

The following SAS statements show how the HPEVENTS procedure can be used to create the event data set, OUT=Eventdata.

```
proc hpfevents data=nile;
   id year interval=year;
   eventkey Shift1899 = LS01JAN1899D;
   eventkey Event1913 = AO01JAN1913D;
   eventkey Event1877 = AO01JAN1877D;
   eventdata out=eventdata;
run;
```

The following SAS statements show that the HPFDIAGNOSE procedure uses this event data by specifying the INEVENT=EVENTDATA option. The EVENT statement specifies the name of events defined in the INEVENT=EVENTDATA.

```
proc hpfdiag data=nile print=short inevent=eventdata;
   id year interval=year;
   forecast riverFlow;
   event Shift1899 Event1913 Event1877;
   transform;
   ucm component=(level cycle);
run;
```

Figure 5.28 shows that two significant events are chosen.

**Figure 5.28** UCM Event Selection

**The HPFDIAGNOSE Procedure**

**UCM Event Selection**

| Event Name | Selected |
|---|---|
| **SHIFT1899** | YES |
| **EVENT1913** | YES |

Figure 5.29 shows the UCM specification for the Nile data. The data has significant irregular and level components, along with two significant events.

**Figure 5.29** UCM Specification

**Unobserved Components Model(UCM) Specification**

| Variable | Functional Transform | Component | Selected | Stochastic | Period | Model Criterion | Statistic | Status |
|---|---|---|---|---|---|---|---|---|
| **riverFlow** | NONE | IRREGULAR | YES | YES | | RMSE | 117.13 | OK |
| | | LEVEL | YES | NO | | | | |
| | | CYCLE1 | NO | | | | | |
| | | CYCLE2 | NO | | | | | |
| | | EVENT | 2 | | | | | |

The following SAS code generates the same results as the previous example without specifying an INEVENT= data set. In this example, SAS predefined event keywords are specified in the EVENT statement. Figure 5.30 shows the results from this example.

```
proc hpfdiag data=nile print=short;
   id year interval=year;
   forecast riverFlow;
   event LS01JAN1899D AO01JAN1913D AO01JAN1877D;
   transform;
   ucm component=(level irregular);
run;
```

**Figure 5.30** UCM Specification

**The HPFDIAGNOSE Procedure**

**Unobserved Components Model(UCM) Specification**

| Variable | Functional Transform | Component | Selected | Stochastic | Model Criterion | Statistic | Status |
|---|---|---|---|---|---|---|---|
| **riverFlow** | NONE | IRREGULAR | YES | YES | RMSE | 117.13 | OK |
| | | LEVEL | YES | NO | | | |
| | | EVENT | 2 | | | | |

## Thread Usage in PROC HPFDIAGNOSE

The HPFDIAGNOSE procedure can make use of threads to speed up different aspects of model identification. There are no special options beyond those present in the SAS System to control thread usage in PROC HPFDIAGNOSE. You can use the OPTIONS statement to specify the following SAS options:

**THREADS | NOTHREADS** controls whether threading is allowed in PROC HPFDIAGNOSE.

**CPUCOUNT=** controls the maximum number of threads to be used by PROC HPFDIAGNOSE.

When threading is allowed, the thread budget for the PROC HPFDIAGNOSE step is determined by the CPUCOUNT option setting. Threaded execution is considered only when the CPUCOUNT setting is more than 1.

PROC HPFDIAGNOSE can use threads in certain phases of its computations within each BY group that it processes. When threading is allowed, PROC HPFDIAGNOSE can make use of threads in these computational processes:

- model identification by model family
- preliminary predictor qualification

PROC HPFDIAGNOSE can use threads to perform model identification for different model families concurrently. You control the model families to be considered in the PROC HPFDIAGNOSE step by the following statements:

**ARIMAX** allocates one or two threads depending on the value of the IDENTIFYORDER= option. The identification orders ARIMA and REG are performed concurrently when both are requested.

**ESM** allocates one thread.

**UCM** allocates one thread.

It is possible to have up to four model identification steps executing concurrently for each BY group. This is a very coarse level of threading, but it can provide some of the the most dramatic improvement in elapsed time over nonthreaded execution for the added CPU time expenditure. Obviously, the elapsed time improvement for each BY group is effectively determined by the longest time of the respective model families that are performed concurrently.

PROC HPFDIAGNOSE can use threads as part of its variable selection process within ARIMAX and UCM model identification. When you specify predictors for consideration for these models, a preliminary qualification process is performed to determine whether a particular predictor improves the AIC or SBC of the model that is being generated. Because this step considers each predictor in isolation from the others, it represents a good opportunity for exploiting parallelism. Predictors that are considered in each BY group arise from EVENT statements, from INPUT statements, or from the EVENTBY= data set that is specified in the PROC HFDIAGNOSE step. Elapsed time reduction here depends strongly on the number of predictors to be considered in each model family. For threads to be beneficial, each thread must service a mimumum number of predictors to justify the overhead for the threads (the miminum depends on whether the predictor is an EVENT or INPUT, and EVENT variables generally require less time per variable to qualify than INPUT variables). An estimate of the useful thread count for predictor qualification is used to govern the use of threads in this context.

The thread budget (CPUCOUNT) that is allowed for the PROC HPFDIAGNOSE step is dynamically shared between coarse threading of the model families and threading of predictor qualification. Each of the specified ARIMAX (ARIMA and REG) and UCM model families execute concurrently, and each of those considers all of the declared predictors for inclusion in the respective model identification process.

## Relationship with PROC HPFENGINE

The HPFDIAGNOSE procedure diagnoses, and the HPFENGINE procedure forecasts.

There are two ways to communicate between the HPFDIAGNOSE procedure and the HPFENGINE procedure. One way is that the OUTEST= data set specified in the HPFDIAGNOSE procedure is specified as the INEST= data set in the HPFENGINE procedure. The other way is that the HPFSELECT procedure is used to communicate between the HPFDIAGNOSE procedure and the HPFENGINE procedure.

The ALPHA=, CRITERION=, HOLDOUT=, and HOLDOUTPCT= options can be changed using the HPFSELECT procedure before these options are transmitted to the HPFENGINE procedure. Otherwise, the values specified in the HPFDIAGNOSE procedure are transmitted directly to the HPFENGINE procedure.

Missing values in the input series are handled differently in the HPFDIAGNOSE procedure than in the HPFENGINE procedure. The HPFDIAGNOSE procedure uses the smoothed missing values for inputs, but the HPFENGINE procedure does not include the inputs that have missing values. This difference can produce different statistical results between the two procedures.

The model specification files created by the HPFDIAGNOSE procedure can be compared with benchmark model specifications by using the HPFESMSPEC, HPFIDMSPEC, HPFARIMASPEC, and HPFUCMSPEC procedures.

The following example shows how to combine these procedures to diagnose a time series.

```
proc datasets lib=sasuser mt=catalog nolist;
   delete hpfscor mymodel;
run;
```

Create a diagnosed model specification.

```
proc hpfdiag data=sashelp.air outest=est
             modelrepository=sasuser.mymodel;
   id date interval=month;
   forecast air;
   transform;
   arimax;
run;
```

Create an ARIMA$(0,1,1)(0,1,1)_s$ model specification.

```
proc hpfarimaspec modelrepository=sasuser.mymodel
                  specname=benchModel;
   forecast var=dep1 dif=1 12 q=(1)(12) noint transform=log;
run;
```

Create a model selection list that includes a diagnosed model (DIAG0) and a specified model (BENCHMODEL).

```
proc hpfselect modelrepository=sasuser.mymodel
               selectname=arimaSpec;
   select criterion=mape;
   spec diag0 / eventmap(symbol=_none_ event=ao135obs)
                eventmap(symbol=_none_ event=ao29obs);
   spec benchModel / inputmap(symbol=dep1 data=air);
run;
```

Select a better model from the model specification list.

```
proc hpfengine data=sashelp.air print=(select)
               modelrepository=sasuser.mymodel
               globalselection=arimaSpec;
   forecast air;
   id date interval=month;
run;
```

Figure 5.31 shows the DIAG0 and BENCHModel model specifications. The DIAG0.XML is created by the HPFDIAGNOSE procedure, and the BENCHModel is created by the HPFARIMASPEC procedure.

**Figure 5.31** Model Selection from the HPFENGINE Procedure

**The HPFENGINE Procedure**

| Model Selection Criterion = MAPE | | | |
|---|---|---|---|
| **Model** | **Statistic** | **Selected** | **Label** |
| **DIAG0** | 2.7094770 | Yes | ARIMA: Log( AIR ) ~ P = 1 D = (1,12) Q = (12) NOINT |
| **BENCHMODEL** | 2.8979097 | No | ARIMA: Log( DEP1 ) ~ D = (1,12) Q = ((1)(12)) NOINT |

The following statements delete the SAS catalogs.

```
proc datasets lib=sasuser mt=catalog nolist;
   delete hpfscor mymodel;
run;
```

## Data Set Input/Output

The AUXDATA= option specifies auxiliary input data for the HPFDIAGNOSE procedure.

The EVENTBY= and INEVENT= data sets can be specified to supply event-related inputs for the HPFDIAGNOSE procedure.

The HPFDIAGNOSE procedure can create the OUTEST=, OUTOUTLIER=, and OUTPROCINFO= data sets. In general, if diagnostics for a particular time series fail, the output that corresponds to this series is not recorded or is set to missing in the relevant output data set, and appropriate error messages or warning messages or both are recorded in the log.

### AUXDATA= Data Set

PROC HPFDIAGNOSE supports the AUXDATA= option to supply variables that are used in a forecast but are not physically part of the primary data set supplied via the DATA= option. For example, you could use AUXDATA to share a data set with explanatory variables across multiple projects, or you could use AUXDATA to separate out explanatory variables that are redundant below some level in a BY-variable hierarchy or that might not need BY-variable qualification at all. Although you can specify only one DATA= option, you can specify the AUXDATA= option multiple times in the PROC statement to supply more than one auxiliary data set for PROC HPFDIAGNOSE to use during its run. For more information and examples, see Chapter 25, "Using Auxiliary Data Sets."

### EVENTBY= Data Set

The EVENTBY= data set contains information that maps predefined events to specific BY groups and variables to be forecast. Instead of mapping predefined events to all forecast series and all BY groups, you can assign the specific events to the specific BY groups and the specific forecast series. This data set is created by a SAS DATA step.

The EVENTBY= data set has the following columns:

BY variable name contains BY variables that organize the results in BY groups.

_NAME_ contains one or more variables to be forecast.

_EVENT_ contains event names.

_REQUIRED_ indicates whether the event is included in a model.

Figure 5.32 shows an example EVENTBY= data set created by a SAS DATA step. In this example, the EVENTBY= data set contains two BY variables (REGION with two groups, R1 and R2, and STORE with three groups, S1, S2, and S3), three forecast series (Y1, Y2, and Y3), and three events (EASTER, AO11JAN2004D, and AO25JAN2008D). The data set also assigns the event EASTER to REGION=R1, STORE=S1 for the forecast variable Y3, and it assigns EASTER to REGION=R2, STORE=S3 for the forecast variable Y3.

**Figure 5.32** EVENTBY= Data Set

| Obs | region | store | _NAME_ | _EVENT_ | _REQUIRED_ |
|---|---|---|---|---|---|
| **1** | R1 | S1 | Y3 | EASTER | YES |
| **2** | R1 | S1 | Y3 | AO11JAN2004D | MAYBE |
| **3** | R1 | S1 | Y3 | AO25JAN2008D | MAYBE |
| **4** | R1 | S2 | Y1 | AO11JAN2004D | MAYBE |
| **5** | R1 | S2 | Y1 | AO25JAN2008D | YES |
| **6** | R2 | S3 | Y1 | AO11JAN2004D | MAYBE |
| **7** | R2 | S3 | Y1 | AO25JAN2008D | MAYBE |
| **8** | R2 | S3 | Y3 | EASTER | MAYBE |
| **9** | R2 | S3 | Y3 | AO11JAN2004D | MAYBE |

## INEVENT= Data Set

The INEVENT= data set contains information that describes predefined events. This data set is usually created by the HPFEVENTS procedure and is required only if events are included in a model. For more information about creating this data set, see the section "EVENTDATA Statement" on page 282

## OUTEST= Data Set

The OUTEST= data set contains information that maps data set variables to model symbols and references the model specification file and model selection list files for each variable to be forecast. This information is used by the HPFENGINE procedure for further model selection, parameter estimation, and forecasts.

In addition, this information can be used by the HPFSELECT procedure to create customized model specification files.

The OUTEST= data set has the following columns:

BY variable name contains BY variables that organize the results in BY groups.

_NAME_ contains one or more variables to be forecast.

_SELECT_ contains model selection list file names.

The model selection list file contains the information of the values of CRITERION=, ALPHA=, HOLDOUT=, and HOLDPCT= options, EVENT and OUTLIER information, and model specification file names.

_MODEL_ is not applicable in the HPFDIAGNOSE procedure.

_SCORE_ is not applicable in the HPFDIAGNOSE procedure.

_MODELVAR_ specifies the model symbol.

_DSVAR_ specifies the data set variable name.

_VARTYPE_ specifies the variable type.

Here are two examples. The first example has one model specification file with a model selection list file; the second example has two model select list files and four model specification files.

The first example uses the BASENAME=AIRSPEC and the new model repository Sasuser.Mymodel.

```
proc datasets lib=sasuser mt=catalog nolist;
   delete hpfscor mymodel;
run;

proc hpfdiag data=sashelp.air outest=est_air
             modelrepository=sasuser.mymodel
             basename=airSpec;
   id date interval=month;
   forecast air;
   transform;
   arimax;
run;
proc print data=est_air;
run;
```

Figure 5.33 shows _SELECT_=AIRSPEC1 since BASENAME=AIRSPEC is specified. Because the new model repository Sasuser.Mymodel is created, the suffix number followed by AIRSPEC starts from 0. AIRSPEC0 is the model specification file, and AIRSPEC1 is the model selection list file.

**Figure 5.33** OUTEST1

| Obs | _NAME_ | _SELECT_ | _MODEL_ | _SCORE_ | _MODELVAR_ | _DSVAR_ | _VARTYPE_ | _STATUS_ |
|---|---|---|---|---|---|---|---|---|
| 1 | AIR | AIRSPEC1 | | Y | | AIR | DEPENDENT | 0 |

The following statements delete the SAS catalogs.

```
proc datasets lib=sasuser mt=catalog nolist;
   delete hpfscor mymodel;
run;
```

The next example uses the new BASENAME=GNPSPEC and the new model repository Sasuser.Mygnp. The ESM and ARIMAX statement request that two variables to be forecast.

```
proc datasets lib=sasuser mt=catalog nolist;
   delete hpfscor mygnp;
run;

proc hpfdiag data=sashelp.gnp outest=est_gnp
             modelrepository=sasuser.mygnp
             basename=gnpSpec;
   id date interval=qtr;
   forecast consump invest;
   transform;
   esm;
   arimax;
run;

proc print data=est_gnp;
run;
```

Figure 5.34 shows two observations. Because the model repository Sasuser.Mygnp is newly created, the suffix number followed by GNPSPEC starts from 0.

The model selection list GNPSPEC2 contains the two model specifications, GNPSPEC0 is the ARIMAX model specification, and GNPSPEC1 is the ESM specification for the variable to be forecast, Consump.

The model selection list GNPSPEC5 contains the two model specifications, GNPSPEC3 is the ARIMAX model specification, and GNPSPEC4 is the ESM specification for the variable to be forecast, Invest.

**Figure 5.34** OUTEST2

| Obs | _NAME_ | _SELECT_ | _MODEL_ | _SCORE_ | _MODELVAR_ | _DSVAR_ | _VARTYPE_ | _STATUS_ |
|---|---|---|---|---|---|---|---|---|
| 1 | CONSUMP | GNPSPEC2 | | | Y | CONSUMP | DEPENDENT | 0 |
| 2 | INVEST | GNPSPEC5 | | | Y | INVEST | DEPENDENT | 0 |

The following SAS statements delete the SAS catalogs.

```
proc datasets lib=sasuser mt=catalog nolist;
   delete hpfscor myGNP;
run;
```

## OUTOUTLIER= Data Set

The OUTOUTLIER= data set contains information about the outliers and has the following variables:

BY variable name contains BY variables that organize the results in BY groups.

_NAME_ contains variables to be forecast.

_TYPE_ contains a type, either AO for an additive outlier or LS for a level shift.

_DIRECTION_ contains a direction, either UP for a positive effect or DOWN for a negative effect.

_OBS_ contains the number of the observation where the outlier happens.

_SASDATE_ contains the SAS date when the outlier happens.

_EVENT_ contains the outlier's event name.

_ESTIMATE_ contains the coefficient estimate.

_CHISQ_ contains the chi-square statistic of the coefficient estimate.

_PVALUE_ contains the $p$ -value of the coefficient estimate.

_IDSTEP_ contains the step name where the outlier diagnosis occurred. Valid values include:

- ARIMA indicates that the outlier event comes from an ARIMA model with no predictors.
- ARIMA_REG indicates that the outlier event comes from a model generated by ARIMA order selection first, followed by the selection of significant inputs.
- REG_ARIMA indicates that the outlier event comes from a model generated by the selection of significant inputs first, followed by ARIMA order selection.

The following SAS statements and Figure 5.35 show the OUTOUTLIER= data set that contains information associated with the output in Figure 5.20.

```
proc hpfdiag data=hardware outoutlier=outl;
   id date interval=month;
   forecast hardware;
   transform;
   arimax;
run;

proc print data=outl;
run;
```

**Figure 5.35** OUTOUTLIER Data Set

| Obs | _NAME_ | _TYPE_ | _DIRECTION_ | _OBS_ | _SASDATE_ | _EVENT_ | _ESTIMATE_ | _CHISQ_ | _PVALUE_ | _IDSTEP_ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | hardware | LS | DOWN | 99 | 01MAR75 | LS01MAR1975D | -125.695 | 25.7273 | .000000393 | ARIMA |
| 2 | hardware | LS | DOWN | 96 | 01DEC74 | LS01DEC1974D | -115.714 | 29.6352 | .000000052 | ARIMA |

### OUTPROCINFO= Data Set

The OUTPROCINFO= data set contains the following variables:

| | |
|---|---|
| _SOURCE_ | the source procedure that produces this data set |
| _STAGE_ | stage of the procedure execution for which the summary variable is reported |
| _NAME_ | name of the summary variable |
| _LABEL_ | description of the summary variable |
| _VALUE_ | value of the summary variable |

## ODS Table Names

The HPFDIAGNOSE procedure assigns a name to each table it creates. You can use these names to refer to the table when you use the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in Table 5.2:

**Table 5.2** ODS Tables Produced Using the HPFDIAGNOSE Procedure

| ODS Table Name | Description | Statement | Option |
|---|---|---|---|
| **ODS Tables Created by the PRINT=SHORT Option** | | | |
| ARIMAEventSelect | ARIMA event selection | EVENT | |
| ARIMAInputSelect | ARIMA input selection | INPUT | |
| ARIMASpec | ARIMA model specification | ARIMAX | |
| BestmodelSpec | selected model specification | | |
| ESMSpec | exponential smoothing model specification | ESM | |
| FinalARIMASpec | final ARIMA model specification | ARIMAX | |

**Table 5.2** *continued*

| ODS Table Name | Description | Statement | Option |
|---|---|---|---|
| IDMSpec | intermittent model specification | IDM | |
| OutlierInfo | ARIMA outlier selection | ARIMAX | |
| UCMEventSelect | UCM event selection | EVENT | |
| UCMInputSelect | UCM input selection | INPUT | |
| UCMSpec | unobserved components model specification | UCM | |
| VariableInfo | forecast variable information | | |
| **Additional ODS Tables Created by the PRINT=LONG Option** | | | |
| DFTestSummary | Dickey-Fuller unit root test | TREND | DIFF |
| JointTestSummary | joint unit root test | TREND | DIFF, SDIFF |
| SeasonDFTestSummary | seasonal Dickey-Fuller unit root test | TREND | SDIFF |
| Transform | functional transformation test | TRANSFORM | TYPE=AUTO |
| **Additional ODS Tables Created by the PRINT=ALL Option** | | | |
| DFTest | Dickey-Fuller unit root test | TREND | DIFF |
| ESACF | extended sample autocorrelation function | ARIMAX | ESACF |
| ESACFPValues | *p*-values of ESACF | ARIMAX | ESACF |
| JointTest | joint unit root test | TREND | DIFF, SDIFF |
| MINIC | minimum information criterion | ARIMAX | MINIC |
| SCAN | squared canonical correlation estimates | ARIMAX | SCAN |
| SCANPValues | *p*-values of SCAN | ARIMAX | SCAN |
| SeasonDFTest | seasonal Dickey-Fuller unit root test | TREND | SDIFF |

# Examples: HPFDIAGNOSE Procedure

## Example 5.1: Selection of Input Variables

This example requests testing of the transformation and differencing of the input variables independent of the variable to be forecast.

```
proc hpfdiagnose data=sashelp.citimon(obs=141)
     testinput=both selectinput=all print=all;
   forecast conb;
   input cciutc eec eegp exvus fm1 fm1d82;
   transform;
   arimax;
run;
```

Output 5.1.1 shows that the ARIMA $(0, 1, 0)$ model is diagnosed for the variable (Conb) to be forecast.

**Output 5.1.1** ARIMAX Specification before Input Selection

**The HPFDIAGNOSE Procedure**

**ARIMA Model Specification**

| Variable | Functional Transform | Constant | p | d | q | Model Criterion | Statistic | Status |
|---|---|---|---|---|---|---|---|---|
| **CONB** | NONE | NO | 0 | 1 | 0 | RMSE | 2363.79 | OK |

Output 5.1.2 shows that one input variable (Eegp) is selected. The input variable needs a simple differencing.

**Output 5.1.2** ARIMA Input Selection

**ARIMA Input Selection**

| Input Variable | Selected | Functional Transform | d | Delay | Numerator | Denominator | Status |
|---|---|---|---|---|---|---|---|
| **CCIUTC** | NO | NONE | 1 | 2 | 0 | 0 | Not Improved |
| **EEC** | NO | | | | | | Not Causal |
| **EEGP** | YES | NONE | 1 | 8 | 0 | 0 | OK |
| **EXVUS** | NO | | | | | | Not Causal |
| **FM1** | NO | | | | | | Not Causal |
| **FM1D82** | NO | | | | | | Not Causal |

Output 5.1.3 shows that the RMSE model selection criterion with inputs is smaller than the model selection criterion without inputs and outliers.

**Output 5.1.3** ARIMAX Specification after Input Selection

**ARIMA Model Specification After Adjusting for Inputs and Outliers**

| Variable | Functional Transform | Constant | p | d | q | Outlier | Input | Model Criterion | Statistic | Status |
|---|---|---|---|---|---|---|---|---|---|---|
| **CONB** | NONE | NO | 0 | 1 | 0 | 1 | 1 | RMSE | 2171.06 | OK |

## Example 5.2: Selection of Events and Input Variables

This example demonstrates how to select events and input variables.

```
proc hpfevents data=sashelp.gnp;
   id date interval=qtr;
   eventkey shock=AO1050BS;
   eventkey shift=LS850BS;
   eventdata out=eventdata;
run;

proc hpfdiag data=sashelp.gnp print=all inevent=eventdata
             testinput=trend;
   id date interval=qtr;
   forecast gnp;
```

```
    input consump invest exports govt;
    event shock shift;
    transform;
    arimax outlier=(detect=no);
run;
```

Output 5.2.1 shows the seasonal ARIMA (1, 1, 1) model diagnosed for the variable (Gnp) to be forecast.

**Output 5.2.1** ARIMAX Specification before Event Input Selection

**The HPFDIAGNOSE Procedure**

**ARIMA Model Specification**

| Variable | Functional Transform | Constant | p | d | q | P | D | Q | Seasonality | Model Criterion | Statistic | Status |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **GNP** | NONE | NO | 1 | 1 | 1 | 0 | 0 | 0 | 4 | RMSE | 22.1611 | OK |

Output 5.2.2 shows that the SHIFT event is significant.

**Output 5.2.2** ARIMA Event Selection

**ARIMA Event Selection**

| Event Name | Selected | d | D | Status |
|---|---|---|---|---|
| **SHOCK** | NO | 1 | 0 | Not Improved |
| **SHIFT** | YES | 1 | 0 | OK |

Output 5.2.3 shows that the input variable Consump is selected in the model.

**Output 5.2.3** ARIMA Input Selection

**ARIMA Input Selection**

| Input Variable | Selected | Functional Transform | d | D | Delay | Numerator | Denominator | Status |
|---|---|---|---|---|---|---|---|---|
| **CONSUMP** | YES | NONE | 1 | 0 | 0 | 2 | 1 | OK |
| **INVEST** | NO | NONE | 1 | 0 | 0 | 0 | 0 | Unstable Model |
| **EXPORTS** | NO | | | | | | | Not Causal |
| **GOVT** | NO | NONE | 1 | 0 | 0 | 2 | 0 | Unstable Model |

Output 5.2.4 shows that the RMSE model selection criterion with the events and input is smaller than that without the events and input.

**Output 5.2.4** ARIMAX Specification after Event Input Selection

**ARIMA Model Specification After Adjusting for Events and Inputs**

| Variable | Functional Transform | Constant | p | d | q | P | D | Q | Seasonality | Input | Event | Model Criterion | Statistic | Status |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **GNP** | NONE | NO | 1 | 1 | 1 | 0 | 0 | 0 | 4 | 1 | 1 | RMSE | 15.3900 | OK |

## Example 5.3: Intermittent Demand Series

This example shows that the data is an intermittent demand series.

```
data inventory;
   input tires @@;
datalines;
0 0 0 6 0 4 0 0 0 2 0 2 2 0 0 0 6 0 0 0
;

proc hpfdiag data=inventory print=all;
   forecast tires;
   transform;
run;
```

Output 5.3.1 shows that the variable (Tires) to be forecast is an intermittent demand series. The interval/size demand model and average demand model were diagnosed to the data. The value of model selection criterion (RMSE) of the average demand model is smaller than that of the interval/size demand model.

**Output 5.3.1** Intermittent Demand Model Specification

**The HPFDIAGNOSE Procedure**

**Intermittent Demand Model Specification**

| Variable | Demand Model | Functional Transform | Selected Model | Component | Model Criterion | Statistic |
|---|---|---|---|---|---|---|
| tires | INTERVAL | NONE | SIMPLE | LEVEL | RMSE | 0.6690 |
| | SIZE | NONE | SIMPLE | LEVEL | | |
| | AVERAGE | NONE | SIMPLE | LEVEL | | 0.5919 |

## Example 5.4: Exponential Smoothing Model

This example illustrates the use of exponential smoothing models (ESM).

```
data investment;
   input inv @@;
   label inv="Gross Investment";
datalines;
33.1 45. 77.2 44.6 48.1 74.4 113. 91.9 61.3 56.8 93.6
159.9 147.2 146.3 98.3 93.5 135.2 157.3 179.5 189.6
;

proc hpfdiag data=investment print=all;
   forecast inv;
   transform;
   esm;
run;
```

Output 5.4.1 shows that the variable (lnv) to be forecast diagnosed the damped-trend exponential smoothing model.

**Output 5.4.1** Exponential Smoothing Model Specification

**The HPFDIAGNOSE Procedure**

**Exponential Smoothing Model Specification**

| Variable | Functional Transform | Selected Model | Component | Model Criterion | Statistic |
|---|---|---|---|---|---|
| **inv** | NONE | DAMPTREND | LEVEL | RMSE | 26.2492 |
| | | | TREND | | |
| | | | DAMP | | |

## Example 5.5: Unobserved Components Model

This example illustrates the use of the UCM statement in the HPFDIAGNOSE procedure.

```
data ozone;
   input ozone @@;
   label ozone  = 'Ozone Concentration'
         x1     = 'Intervention for post 1960 period'
         summer = 'Summer Months Intervention'
         winter = 'Winter Months Intervention';
   date = intnx( 'month', '31dec1954'd, _n_ );
   format date monyy.;
   month = month( date );
   year = year( date );
   x1 = year >= 1960;
   summer = ( 5 < month < 11 ) * ( year > 1965 );
   winter = ( year > 1965 ) - summer;
datalines;

   ... more lines ...


proc hpfdiag data=ozone print=all;
   id date interval=month;
   forecast ozone;
   input x1 summer winter;
   transform;
   ucm;
run;
```

Output 5.5.1 shows that the input Winter is selected in the model.

**Output 5.5.1** UCM Input Selection

**The HPFDIAGNOSE Procedure**

**UCM Input Selection**

| Input Variable | Selected | Functional Transform | d | D | Delay | Status |
|---|---|---|---|---|---|---|
| **x1** | NO | NONE | 0 | 0 | 11 | Insignificant |
| **summer** | NO | | | | | Not Causal |
| **winter** | YES | NONE | 0 | 0 | 4 | OK |

Output 5.5.2 shows that the forecast variable (Ozone) is explained by the irregular and season components, along with the single input variable.

**Output 5.5.2** Unobserved Components Model Specification

**Unobserved Components Model(UCM) Specification**

| Variable | Functional Transform | Component | Selected | Stochastic | Seasonality | Model Criterion | Statistic | Status |
|---|---|---|---|---|---|---|---|---|
| **ozone** | NONE | IRREGULAR | YES | YES | | RMSE | 1.0110 | OK |
| | | LEVEL | YES | NO | | | | |
| | | SLOPE | NO | | | | | |
| | | SEASON | YES | NO | 12 | | | |
| | | INPUT | 1 | | | | | |

## Example 5.6: Automatic Model Combination

This example illustrates the automatic model combination feature of the HPFDIAGNOSE procedure.

The following statements run PROC HPFDIAGNOSE to generate model combinations for the automatic models it formulates from its diagnosis of the input series. The combined forecast is produced using a simple average of the candidate forecasts. The COMBINE statement specifies an OLS encompassing test and tests to access forecast quality with an estimation region missing percentage of 25% and a horizon missing percentage of 50%.

```
proc hpfdiagnose
     data=sashelp.air
     rep=work.diagcomb1
     outest=diagest;
   id date interval=month;
   forecast air;
   esm;
   arimax;
   combine method=average encompass=ols misspercent=25 hormisspercent=50;
run;
```

The OUTEST= data set and model repository from HPFDIAGNOSE are fed into the HPFENGINE procedure to perform model selection and forecast generation as follows:

```
proc hpfengine data=sashelp.air
     rep=work.diagcomb1
     inest=diagest
     out=outcomb
     outfor=forcomb
     outest=estcomb
     outstatselect=selcomb
     print=(select estimates)
     lead=4;
   id date interval=month;
   forecast air;
run;
```

Output 5.6.1 shows that the combined forecast is selected from this run.

**Output 5.6.1** Model Selection Statistics

**The HPFENGINE Procedure**

| Model Selection Criterion = RMSE | | | |
|---|---|---|---|
| **Model** | **Statistic** | **Selected** | **Label** |
| **diag0** | 10.695241 | No | ARIMA: AIR ~ P = 1 D = (1,12) NOINT |
| **diag1** | 10.579085 | No | Winters Method (Multiplicative) |
| **diag2** | 9.571806 | Yes | COMBINE: AVERAGE(diag0,diag1) |

# References


Box, G. E. P. and Jenkins, G. M. (1976), *Time Series Analysis: Forecasting and Control*, Rev. Edition, San Francisco: Holden-Day.

Box, G. E. P., Jenkins, G. M., and Reinsel, G. C. (1994), *Time Series Analysis: Forecasting and Control*, 3rd Edition, Englewood Cliffs, NJ: Prentice-Hall.

Choi, B. (1992), *ARMA Model Identification*, New York: Springer-Verlag.

Cobb, G. W. (1978), "The Problem of the Nile: Conditional Solution to a Change Point Problem," *Biometrika*, 65, 243–251.

De Jong, P. and Penzer, J. (1998), "Diagnosing Shocks in Time Series," *Journal of the American Statistical Association*, 93, 796–806.

Dickey, D. A. and Fuller, W. A. (1979), "Distribution of the Estimators for Autoregressive Time Series with a Unit Root," *Journal of the American Statistical Association*, 74, 427–431.

Dickey, D. A., Hasza, D. P., and Fuller, W. A. (1984), "Testing for Unit Roots in Seasonal Time Series," *Journal of the American Statistical Association*, 79, 355–367.

Durbin, J. and Koopman, S. J. (2001), *Time Series Analysis by State Space Methods*, Oxford: Oxford University Press.

Harvey, A. C. (1989), *Forecasting, Structural Time Series Models, and the Kalman Filter*, Cambridge: Cambridge University Press.

Harvey, A. C. (2001), "Testing in Unobserved Components Models," *Journal of Forecasting*, 20, 1–19.

Hasza, D. P. and Fuller, W. A. (1979), "Estimation for Autoregressive Processes with Unit Roots," *Annals of Statistics*, 7, 1106–1120.

Hasza, D. P. and Fuller, W. A. (1984), "Testing for Nonstationary Parameter Specifications in Seasonal Time Series Models," *Annals of Statistics*, 10, 1209–1216.

Hillmer, S. C., Larcker, D. F., and Schroeder, D. A. (1983), "Forecasting Accounting Data: A Multiple Time-Series Analysis," *Journal of Forecasting*, 2, 389–404.

McKenzie, E. (1984), "General Exponential Smoothing and the Equivalent ARMA Process," *Journal of Forecasting*, 3, 333–344.

Tsay, R. S. and Tiao, G. C. (1984), "Consistent Estimates of Autoregressive Parameters and Extended Sample Autocorrelation Function for Stationary and Nonstationary ARMA Models," *Journal of the American Statistical Association*, 79, 84–96.

# Chapter 6
# The HPFDIAGSPEC Procedure

**Contents**


Overview: HPFDIAGSPEC Procedure . . . . . **157**
Getting Started: HPFDIAGSPEC Procedure . . . . . **158**
Syntax: HPFDIAGSPEC Procedure . . . . . **159**
Functional Summary . . . . . 160
PROC HPFDIAGSPEC Statement . . . . . 162
ARIMAX Statement . . . . . 162
COMBINE Statement . . . . . 164
DIAGNOSE Statement . . . . . 168
ESM Statement . . . . . 170
IDM Statement . . . . . 171
TRANSFORM Statement . . . . . 172
TREND Statement . . . . . 173
UCM Statement . . . . . 174
Details: HPFDIAGSPEC Procedure . . . . . **175**
Examples: HPFDIAGSPEC Procedure . . . . . **175**
Example 6.1: ARIMA Modeling Using Input Variables . . . . . 175
Example 6.2: Unobserved Components and ARIMA Models with Input Variables . . 178
Example 6.3: Automatic Model Combination . . . . . 179
References . . . . . **180**


## Overview: HPFDIAGSPEC Procedure

The HPFDIAGSPEC procedure creates a set of instructions or preferences that are used to identify time series models. This procedure supports many of the statements that the HPFDIAGNOSE procedure supports, but instead of executing these statements, PROC HPFDIAGSPEC creates a named specification file that is stored to a model repository catalog. The role of this procedure is analogous to the other model specification procedures (HPFESMSPEC, HPFARIMASPEC, HPFIDMSPEC, HPFUCMSPEC, and HPFEXMSPEC) and the HPFSELECT procedure, all of which function similarly to translate SAS statement syntax into a specification file that contains control information from the statements that are processed during the procedure invocation and then stores that specification in a model repository catalog.

The HPFDIAGSPEC procedure is used in conjunction with the HPFORECAST procedure. PROC HPFDIAGSPEC enables you to create diagnostic control preferences to be used by PROC HPFORECAST when PROC HPFORECAST is invoked to perform automatic time series forecasting.

## Getting Started: HPFDIAGSPEC Procedure

This section shows examples of how to generate diagnostic control preferences that can be used in PROC HPFORECAST to diagnose autoregressive integrated moving average (ARIMA), exponential smoothing model (ESM), and unobserved component model (UCM) models.

The following code creates a diagnostic control specification called DIAGSPECAR and stores it to the catalog Work.Myrep:

```
proc hpfdiagspec modelrepository=work.myrep
                 specname=diagspecar
                 label="Transform test and ARIMA models";
   transform;
   arimax;
run;
```

The TRANSFORM statement without further options indicates that you want to perform transformation tests by using defaults that are the same as in the TRANSFORM statement in the HPFDIAGNOSE procedure. For more information, see "TRANSFORM Statement" on page 172. The ARIMAX statement without further options indicates that you want to perform ARIMA model indentification by using defaults that are the same as in the ARIMAX statement in the HPFDIAGNOSE procedure. For more information, see the section "ARIMAX Statement" on page 162.

The following code creates a diagnostic control specification called DIAGSPECESM and stores it to the catalog Work.Myrep. The only difference between the following code and the preceding code is that the following code requests exponential smoothing model (ESM) identification instead of ARIMA model identification. The ESM statement without further options indicates that you want to perform ESM identification by using defaults that are the same as in the ESM statement in PROC HPFDIAGNOSE. For more information, see the section "ESM Statement" on page 170.

```
proc hpfdiagspec modelrepository=work.myrep
                 specname=diagspecesm
                 label="Transform test and ESM models";
   transform;
   esm;
run;
```

The following code creates a diagnostic control specification called DIAGSPECUCM and stores it to the catalog Work.Myrep. The only difference between this code and the preceding code is that the following code requests unobserved component model (UCM) identification instead of ESM identification. The UCM statement without further options indicates that you want to perform UCM identification with defaults that are the same as in the UCM statement in PROC HPFDIAGNOSE. For more information, see the section "UCM Statement" on page 174.

```
proc hpfdiagspec modelrepository=work.myrep
                 specname=diagspecucm
                 label="Transform test and UCM models";
   transform;
   ucm;
run;
```

The following code shows how to pass a diagnostic specification that is created by the HPFDIAGSPEC procedure to the HPFORECAST procedure. The PROC HPFDIAGSPEC step combines the previous examples into a single diagnostic control specification to perform the transform test and diagnose the best ESM, ARIMA, and UCM models for each time series from the data set Sashelp.Pricedata. ARIMA and UCM model identification steps are allowed to consider the predictor series from the variables Price and Discount.

```
proc hpfdiagspec modelrepository=work.myrep
                 specname=diagspec
                 label="Transform test + ARIMAX,ESM,UCM models";
   transform;
   arimax;
   esm;
   ucm;
run;

proc hpforecast data=sashelp.pricedata
     modelrepository=work.myrep
     globaldiagnose=diagspec
     mode=auto
     lead=12;
   output outfor=salefor
          outest=salepest
          outlog=errors;
   timeid date interval=month;
   forecast sale;
   input price discount;
   level regionName productName;
   end level;
   level regionName;
   end level;
run;
```

# Syntax: HPFDIAGSPEC Procedure

The following statements are available in the HPFDIAGSPEC procedure:

**PROC HPFDIAGSPEC** *options* ;
  **ARIMAX** *options* ;
  **COMBINE** *options* ;
  **ESM** *option* ;
  **IDM** *options* ;
  **TRANSFORM** *options* ;
  **TREND** *options* ;
  **UCM** *options* ;

If no model statement (ARIMAX, ESM, IDM, or UCM) is specified, then the HPFDIAGSPEC procedure creates commands to diagnose ARIMAX and exponential smoothing models and to test for intermittent demand based on the default intermittency threshold.

## Functional Summary

Table 6.1 summarizes the statements and options that control the HPFDIAGSPEC procedure.

**Table 6.1** HPFDIAGSPEC Functional Summary

| Description | Statement | Option |
|---|---|---|
| **Statements** | | |
| Specifies model combination options | COMBINE | |
| Specifies options for the log transform test and other functional transformation types | TRANSFORM | |
| Specifies options for the stationarity test | TREND | |
| Specifies ARIMA model options | ARIMAX | |
| Specifies exponential smoothing model options | ESM | |
| Specifies intermittent demand model options | IDM | |
| Specifies unobserved components model options | UCM | |
| **Model Repository Options** | | |
| Specifies the model repository | HPFDIAGSPEC | REPOSITORY= |
| **Transformation Test Options** | | |
| Specifies the autoregressive (AR) order for the log transformation test | TRANSFORM | P= |
| Specifies the type of the functional transformation | TRANSFORM | TYPE= |
| Specifies the method of the forecasts of the transformed series | TRANSFORM | TRANSOPT= |
| **Trend Test Options** | | |
| Specifies the simple differencing order | TREND | DIFF= |
| Specifies the seasonal differencing order | TREND | SDIFF= |
| Specifies the autoregressive (AR) order for the augmented unit root test | TREND | P= |
| **ARIMAX Model Options** | | |
| Specifies the autoregressive moving average (ARMA) order selection criterion | ARIMAX | CRITERION= |
| Specifies the range of the AR orders for obtaining the error series used in the MINIC method | ARIMAX | PERROR= |
| Specifies the range of the AR orders | ARIMAX | P= |
| Specifies the range of the MA orders | ARIMAX | Q= |
| Specifies the range of the denominator orders of the transfer function | ARIMAX | DEN= |
| Specifies the range of the numerator orders of the transfer function | ARIMAX | NUM= |

**Table 6.1** *continued*

| Description | Statement | Option |
|---|---|---|
| Specifies the tentative order identification method | ARIMAX | METHOD= |
| Specifies the identification order of the components | ARIMAX | IDENTIFYORDER= |
| Suppresses the intercept (constant) term | ARIMAX | NOINT |
| **Unobserved Components Model Option** | | |
| Specifies the components to test for inclusion in the UCM | UCM | COMPONENT= |
| **Exponential Smoothing Model Option** | | |
| Specifies the method of the ESM | ESM | METHOD= |
| **Model Combination Options** | | |
| Specifies the combination weight method | COMBINE | METHOD= |
| Specifies the encompassing test | COMBINE | ENCOMPASS= |
| Specifies the forecast combination criterion | COMBINE | CRITERION= |
| Specifies the percentage of missing forecast values | COMBINE | MISSPERCENT= |
| Specifies the percentage of missing horizon values | COMBINE | HORMISSPERCENT= |
| Specifies the combination of missing forecast values | COMBINE | MISSMODE= |
| **Significance Level Option** | | |
| Specifies the significance level for diagnostic tests | HPFDIAGSPEC, TRANSFORM, TREND, ARIMAX, UCM | SIGLEVEL= |
| **Input Variable Control Options** | | |
| Specifies the maximum number of input variables to be selected | DIAGNOSE | SELECTINPUT= |
| Specifies options for the transformation and differencing of the input variables | DIAGNOSE | TESTINPUT= |
| Specifies the maximum missing percentage of the input variables | DIAGNOSE | INPUTMISSINGPCT= |
| **Model Selection Options** | | |
| Specifies the model selection criterion | DIAGNOSE | CRITERION= |
| Specifies the forecast holdout sample size | DIAGNOSE | HOLDOUT= |

**Table 6.1** *continued*

| Description | Statement | Option |
|---|---|---|
| Specifies the forecast holdout sample percentage | DIAGNOSE | HOLDOUTPCT= |
| Specifies the threshold of forward selection of inputs | DIAGNOSE | ENTRYPCT= |
| **Data Prefilter Option** | | |
| Specifies how to handle missing and extreme values prior to diagnostic tests | DIAGNOSE | PREFILTER= |

The following statements describe the PROC HPFDIAGNOSE statement and then describe the other statements in alphabetical order.

## PROC HPFDIAGSPEC Statement

**PROC HPFDIAGSPEC MODELREPOSITORY=***catalog* **NAME=***SAS-name* < *options* > **;**

The PROC HPFDIAGSPEC statement invokes the HPFDIAGSPEC procedure. You can specify the following *options*:

**LABEL=***SAS-label*

**SPECLABEL=***SAS-label*

specifies a descriptive label for the diagnostic specification to be stored in the SAS catalog.

**MODELREPOSITORY=***catalog*

**MODELREP=***catalog*

**REPOSITORY=***catalog*

**REP=***catalog*

specifies the SAS catalog to store the diagnostic control information that is produced from the statements that the procedure processes. You can specify a two-part catalog name that identifies both the data library and catalog. If you specify a one-part name, it is taken as the catalog name and the default SAS data library is used. If the catalog does not previously exist in the data library, then it will be created. This option is required.

**NAME=***SAS-name*

**SPECNAME=***SAS-name*

names the diagnostic specification to be stored in the SAS catalog. This option is required.

## ARIMAX Statement

**ARIMAX** < *options* > **;**

The ARIMAX statement specifies options that are used to identify an appropriate ARIMAX model. The ARIMA model identification process is described in Chapter 5, "The HPFDIAGNOSE Procedure."

You can specify the following *options*:

**CRITERION=AIC | SBC**
specifies the information criterion for the tentative ARMA order selection.

You can specify the following values:

| | |
|---|---|
| **AIC** | uses Akaike's information criterion (AIC). |
| **SBC** | uses the Schwarz-Bayesian criterion (SBC). |

By default, CRITERION=SBC.

**DEN=(*number* : *number*)**
specifies the range of the denominator order of the transfer function. By default, DEN=(0:2).

**ESTMETHOD=CLS | ML | ULS**
specifies the method for choosing the tentative autoregressive moving average (ARMA) orders. For more information, see Chapter 5, "The HPFDIAGNOSE Procedure."

You can specify the following values:

| | |
|---|---|
| **CLS** | uses the conditional least squares method. |
| **ML** | uses the maximum likelihood method. |
| **ULS** | uses the unconditional least squares method. |

By default, ESTMETHOD=CLS.

**IDENTIFYORDER=ARIMA | REG | BOTH**
**IDENTIFY=ARIMA | REG | BOTH**
specifies the identification order when inputs are specified.

You can specify the following values:

| | |
|---|---|
| ARIMA | finds an ARIMA model for the error series first and then chooses significant inputs. |
| REG | finds a regression model first and then decides the AR and MA polynomial orders. |
| BOTH | finds models by using both methods and determines the better model. |

By default, IDENTIFYORDER=ARIMA.

**METHOD=ESACF | MINIC | SCAN**
specifies the method for choosing the tentative ARMA orders. For more information, see Chapter 5, "The HPFDIAGNOSE Procedure."

You can specify the following values:

| | |
|---|---|
| **ESACF** | uses the extended sample autocorrelation function. |
| **MINIC** | uses the minimum information criterion. |
| **SCAN** | uses the smallest canonical correlation analysis. |

By default, METHOD=MINIC.

**NOINT**

**NOCONSTANT**

suppresses the intercept (constant) term.

**NUM=(***number* **:** *number***)**

specifies the range of the numerator order of the transfer function. By default, NUM=(0:2).

**P=(***number* **:** *number***) (***number* **:** *number***)**

specifies the range of the nonseasonal and seasonal autoregressive (AR) orders. By default, P=(0:5)(0:2).

**PERROR=(***number* **:** *number***)**

specifies the range of the AR order for obtaining the error series that the MINIC method uses. The default is $(P_{\max} : P_{\max} + Q_{\max})$ where $P_{\max}$ is the upper limit of the nonseasonal range for the P= option and $Q_{\max}$ is the upper limit of the nonseasonal range for the Q= option.

**Q=(***number* **:** *number***) (***number* **:** *number***)**

specifies the range of the nonseasonal and seasonal moving average (MA) orders. By default, Q=(0:5)(0:2).

**REFINEPARMS=(***refine-options***)**

specifies how to refine insignificant parameters of the final model, identifies the factors to refine, and in which order.

You can specify the following values for *refine-options*:

**FACTOR=ALL** refines the parameters for all factors.

**FACTOR=ARMA** refines the parameters for the ARMA factor.

**FACTOR=INPUT** refines the parameters for the INPUT factor.

**FIRST=***factor* specifies which *factor* to refine first. You can specify ARMA or INPUT.

**SIGLEVEL=***value* specifies the cutoff value for refining insignificant parameters. The *value* should be between (0,1).

You can specify more than one FACTOR value by enclosing them in parentheses. For example, the option FACTOR=(ARMA INPUT) refines the parameters for ARMA and INPUT.

By default, REFINEPARMS=(FACTOR=ALL FIRST=ARMA SIGLEVEL=0.4).

**SIGLEVEL=***value*

specifies the significance level to use as a cutoff value to decide the AR and MA orders. The *value* should be in (0,1). This option overrides the value of SIGLEVEL= option in the DIAGNOSE statement.

## COMBINE Statement

**COMBINE** *options* **;**

The COMBINE statement specifies options that are used to automatically combine models when the diagnostic control specification is subsequently executed. For more information, see Chapter 5, "The HPFDIAGNOSE Procedure."

The following *options* affect the evaluation of the combined forecast that is produced by the combined model list.

**CRITERION=***option*

specifies the forecast combination criterion (statistic of fit) to be used when forecast candidates must be ranked to affect the forecast combination processing. This option is often used in conjunction with the ENCOMPASS= and METHOD=RANKWGT options. By default, CRITERION=RMSE. For valid values for *option*, see the section "Valid Statistic of Fit Names" on page 473.

**ENCOMPASS=NONE** | *test-name***<(***test-options***)>**

specifies whether to perform a forecast encompassing test and if so, which type of test. The encompassing test attempts to eliminate from consideration forecasts that fail to add significant information to the final forecast. By default, ENCOMPASS=NONE, which specifies that no encompassing test be performed.

You can specify the following values for *test-name*:

**HLN<(ALPHA=***number***)>** uses the Harvey-Leybourne-Newbold (HLN) test to estimate pairwise encompassing between candidate forecasts. Candidates are ranked from best to worst using the CRITERION= values. Iterating from best to worst, inferior candidates are tested with the best of the untested candidates for retention in the combined set. You specify the significance level for the test by including the ALPHA=*number* suboption. The default value is ALPHA=0.05 when the simple form ENCOMPASS=HLN is specified. The range is 0 to 1.

**OLS<(ALPHA=***number***)>** uses an OLS-based regression test to estimate pairwise encompassing between candidate forecasts. Candidates are ranked from best to worst using the CRITERION= values. Iterating from best to worst, inferior candidates are tested with the best of the untested candidates for retention in the combined set. You specify the significance level for the test by including the ALPHA=*number* suboption. The default value is ALPHA=0.05 when the simple form ENCOMPASS=OLS is specified. The range is 0 to 1.

**HORMISSPERCENT=***number*

specifies a threshold for the percentage of missing forecast values in the combination horizon that is used to exclude a candidate forecast from consideration in the final combination. The forecast horizon is the region of time in which multistep forecasts are generated. The value of *number* must be in the range 1 to 100. By default, no horizon missing percentage test is performed on candidate forecasts. This test and the MISSPERCENT test operate independent of each other. You can specify one or both of these tests.

**METHOD=***weight-method***<(***method-options***)>**

specifies the method for determining the combination weights that are used in the weighted average of the candidate forecasts in the combination list. The default method is METHOD=AVERAGE. You can specify the simple form METHOD=*weight-method* when you do not want to specify weight-specific *method-options*.

You can specify the following values for *weight-method* and *method-options*:

**AICC<(***AICC-opts***)>** bases computation of the combination weights on corrected AIC weights. For the mathematical details of this process, see Chapter 20, "Forecast Combination Computational Details." Frequently there is considerable disparity between the weights because of the exponential weighting scheme, so you can specify *AICC-opts* to affect the scaling and to cull

low-scoring candidates from consideration for computational efficiency. By default, all AICC scored candidate forecasts are combined.

You can specify the following values for *AICC-opts*:

**ABSWGT=***number* omits computed weights whose values are less than the specified value. The range is 0 to 1 inclusive. The remaining weights are normalized to sum to 1.

**BESTPCT=***number* retains the best $N$ of the candidates as a percentage of the total number weighted, where

$$N = \max\{\lfloor \frac{\textit{number} \cdot M}{100} \rfloor, 1\} \tag{6.1}$$

and $M$ denotes the number of candidate models in the combination after any specified forecast exclusion tests have been performed.

The $N$ remaining weights are normalized to sum to 1.

**BESTN=***N* retains the best $N$ of the candidates as a percentage of the total number weighted. The $N$ remaining weights are normalized to sum to 1.

**LAMBDA=***number* specifies the scale factor to be used in the computation of the AICC weights. The default is LAMBDA=1.0, which results in the usual Akaike weights.

**AVERAGE** computes the simple average of the forecasts that are selected for combination. This is the default.

**ERLS<(***NLP-opts***)>** bases computation of the combination weights on a constrained least squares problem to minimize the $\ell_2$ norm of the combined forecast residuals subject to the constraint that the weights sum to 1.

**LAD<(***LAD-opts***)>** bases computation of the combination weights on a least absolute deviations measure of fit for the combined forecast. A linear program is formulated according to the *LAD-opts* to minimize an objective function that is expressed in terms of absolute values of loss series subject to constraints that the weights sum to 1 and be nonnegative. You can specify the following *LAD-opts*:

**OBJTYPE=***obj-type* specifies the form of the objective function. You can specify these *obj-type* values:

**L1** specifies that the objective is an $\ell_1$ norm that involves loss series.

**LINF** specifies that the objective is an $\ell_\infty$ norm that involves the loss series.

**ERRTYPE=***err-type* specifies the form of the loss series in the objective function. You can specify these *err-type* values:

**ABS** specifies that loss series terms are deviations.

**APE** specifies that loss series terms are percentage deviations.

**RAE** specifies that loss series terms are relative error deviations.

**NERLS<(*NLP-opts*)>** bases computation of the combination weights on a constrained least squares problem to minimize the $\ell_2$ norm of the combined forecast residuals subject to the constraints that the weights sum to 1 and be nonnegative.

**NRLS<(*NLP-opts*)>** bases computation of the combination weights on a constrained least squares problem to minimize the $\ell_2$ norm of the combined forecast residuals subject to the constraints that the weights be nonnegative.

**OLS** computes the combination weights that result from the ordinary least squares problem to minimize the $\ell_2$ norm of the combined forecast residuals.

**RANKWGT(*W1,...,Wn*)** assigns weights by using the rank of the candidate forecasts at the time the combination is performed. The ranking is based on the candidate forecasts' fit statistic values as defined by the CRITERION= option. These weights must sum to 1. If they do not sum to 1, they are normalized and a warning is issued. The number of specified weight values must agree with the number of models that are expected from the ARIMAX, ESM, and UCM statements that are specified in the PROC HPFDIAGSPEC statement block. When the forecast combination from the diagnosed models is subsequently executed, the weights are assigned by ranking the associated forecasts from best to worst according to the CRITERION= option. The best forecast uses the first weight, W1, and so on. At execution time, the set of weights that are used is normalized to account for any discrepancy between the number of weights and the number of models that are diagnosed, for diagnosed models that fail to forecast, or for any diagnosed models that are omitted from the final combination because of exclusion tests that are specified in the COMBINE statement.

**RMSEWGT** bases computation of the combination weights on the RMSE statistic of fit for the forecast contributors. The weights are normalized to sum to 1. For more information, see Chapter 20, "Forecast Combination Computational Details."

**MISSMODE=MISSING | RESCALE**

specifies a method for treating missing values in the forecast combination. In a particular time slice across the combination ensemble, one or more combination contributors can have a missing value. This setting determines the treatment of those in the final combination for such time indices.

You can specify the following values:

**MISSING** generates a missing combined forecast at each time index with one or more missing contributors. This is the default for AICC weights, OLS weights, restricted least squares weights, and LAD weights. For METHOD=OLS or METHOD=NRLS, you cannot specify MISSMODE=RESCALE because the estimated weights are not constrained to sum to one.

**RESCALE** rescales the combination weights for the nonmissing contributors at each time index to sum to 1. This is the default for simple average, ranked user weights, ranked weights, and RMSE weights.

**MISSPERCENT=*number***

specifies a threshold for the percentage of missing forecast values in the combination estimation region that is used to exclude a candidate forecast from consideration in the final combination. The value of *number* must be in the range 1 to 100. By default, no missing percentage test is performed on candidate forecasts. This test and the HORMISSPERCENT test operate independent of each other. You can specify one or both of these tests.

**STDERR=***stderr-method***<(***stderr-options***)>**

**SEMODE=***stderr-method***<(***stderr-options***)>**

specifies the method for computing the prediction error variance series. This series is used to compute the prediction standard error, which in turn is used to compute confidence bands on the combined forecast.

You can use the simple form STDERR=*stderr-method* if you do not want to specify any method-specific options. You can specify the following values for *stderr-method*:

**DIAG** computes the prediction error variance by assuming that the forecast errors at time $t$ are uncorrelated so that the simple diagonal form of $\Sigma_t$ is used.

**ESTCORR** computes the prediction error variance by using estimates of $\rho_{i,j,t}$, which represent the sample cross-correlation between $e_{i,t}$ and $e_{j,t}$ over the time span $t = 1, \ldots, T$, where $T$ denotes the last time index of the actual series $y_t$. Of course, this option implies that the error series $e_{i,t}$ and $e_{j,t}$ are assumed to be jointly stationary.

**ESTCORR(TAU=$\tau$)** is similar to STDERR=ESTCORR except that the cross-correlation estimates are localized to a time window of $\tau$ steps. The time span $t = 1, \ldots, T$ is quantized into segments of $\tau$ steps working from $T$ backwards for in-sample cross-correlation estimates. The cross-correlation estimates from the interval $[T - \tau, T]$ are used for the period of multistep forecasts that extend beyond time $T$.

By default, STDERR=DIAG.

## DIAGNOSE Statement

**DIAGNOSE** *options* **;**

The DIAGNOSE statement defines control options that are used across the model families in the diagnostic control instructions that are created from the PROC HPFDIAGSPEC invocation. These options are analogous to the similarly named options in the PROC HPFDIAGNOSE statement. For more information, see the section "PROC HPFDIAGSPEC Statement" on page 162.

You can specify the following *options* in the DIAGNOSE statement.

**CRITERION=***option*

specifies the model selection criterion for selecting the best model. This option is often used in conjunction with the HOLDOUT= and HOLDOUTPCT= options. By default, CRITERION=RMSE. For more information, see the description of CRITERION= for PROC HPFDIAGNOSE.

**DELAYINPUT=***number*

specifies the delay lag for the inputs. If you do not specify this option, the delay lag for the inputs is appropriately chosen during the execution of the diagnostic specification.

**ENTRYPCT=***number*

specifies a threshold to check the percentage increment of the criterion between two candidate models. The ENTRYPCT=value should be in the range (0,100). By default, ENTRYPCT=0.1.

**HOLDOUT=***number*
specifies the size of the holdout sample to be used for model selection. The holdout sample is a subset of the dependent time series; it ends at the last nonmissing observation. When you use a holdout sample, the fit statistics for the model selection criterion (CRITERION= option) are computed using only the holdout sample time periods; otherwise, the model selection fit statistics are computed over the full in-sample region of the dependent time series. By default, HOLDOUT=0.

**HOLDOUTPCT=***value*
specifies the size of the holdout sample as a percentage of the length of the dependent time series. If HOLDOUT=5 and HOLDOUTPCT=10, the size of the holdout sample is $min(5, 0.1T)$ where $T$ is the length of the dependent time series after the beginning and ending missing values are removed. When you use a holdout sample, the fit statistics for the model selection criterion (as specified in the CRITERION= option) are computed using only the holdout sample time periods; otherwise, the model selection fit statistics are computed over the full in-sample region of the dependent time series. By default, HOLDOUTPCT=0.

**INPUTMISSINGPCT=***value*
specifies the size of the missing observation as a percentage of the length of any input time series. If INPUTMISSINGPCT=50, then any input time series that has more than 50% missing data is ignored during model identification. By default, INPUTMISSINGPCT=10.

**PREFILTER=BOTH | EXTREME | MISSING | YES**
specifies how missing and extreme values are handled prior to diagnostic tests.

You can specify the following values:

**BOTH** is equivalent to both YES and EXTREME.

**EXTREME** requests that extreme values be set to missing for a tentative ARIMA model and that extreme values be used for the final ARIMAX model diagnostics.

**MISSING** requests that smoothed values for missing data be applied for tentative order selection and that missing data be preserved for the final diagnostics.

**YES** requests that smoothed values for missing data be applied to overall diagnoses.

By default, PREFILTER=YES. If the input variables have missing values, they are always smoothed for the diagnostics.

**SELECTINPUT=ALL | SELECT |** *number*
specifies the maximum number of input variables to select.

You can specify the following values:

**ALL** selects the input variables that satisfy the criteria of noncollinearity and nonnegative delay.

**SELECT** selects the input variables that satisfy the criteria of noncollinearity, nonnegative delay, and smaller AIC.

*number* selects the best *number* of input variables that satisfy the criteria of noncollinearity and nonnegative delay.

By default, SELECTINPUT=SELECT.

**SIGLEVEL=***value*

specifies the cutoff value for all diagnostic tests (such as log transformation, stationarity, tentative ARMA order selection, and significance of UCM components). The *value* should be between 0,1). The SIGLEVEL options in the TRANSFORM, TREND, ARIMAX, and UCM statements control testing independently. By default, SIGLEVEL=0.05.

**TESTINPUT=BOTH | TRANSFORM | TREND**

You can specify the following values:

**BOTH** requests that log transform and trend testing of the input variables be applied independently of the variable to be forecast.

**TRANSFORM** requests that log transform testing of the input variables be applied independently of the variable to be forecast.

**TREND** requests that trend testing of the input variables be applied independently of the variable to be forecast.

If you do not specify this option, the same differencing is applied to the input variables that is used for the variable to be forecast, and no transformation is applied to the input variables.

## ESM Statement

**ESM** < *option* > **;**

The ESM statement specifies options that are used to identify an appropriate exponential smoothing model (ESM) based on the model selection criterion. The ESM model identification process is described in the PROC HPFDIAGNOSE chapter. For more information, see Chapter 5, "The HPFDIAGNOSE Procedure."

You can specify the following *option* in the ESM statement.

**METHOD=BEST | BESTN | BESTS**

You can specify the following values:

**BEST** determines the best smoothing model from these candidates: SIMPLE, DOUBLE, LINEAR, DAMPTREND, SEASONAL, WINTERS, and ADDWINTERS.

**BESTN** determines the best nonseasonal smoothing model from these candidates: SIMPLE, DOUBLE, LINEAR, and DAMPTREND.

**BESTS** determines the best seasonal smoothing model from these candidates: SEASONAL, WINTERS, and ADDWINTERS.

By default, METHOD=BEST.

## IDM Statement

**IDM** < *options* > **;**

The IDM statement controls the intermittency test and the generation of intermittent demand models.

When the diagnostic specification is executed, an intermittency test is performed first. If the dependent series is intermittent, an intermittent demand model (IDM) is fit to the data based on the model selection criterion. However, if the series is not intermittent, models for the other families that are specified in the PROC HPFDIAGSPEC statements are considered.

You can specify the following *options*:

**BASE=AUTO** | *value*
specifies the base value of the time series that is used to determine the demand series components. The demand series components are determined where there are departures from the base value.

You can specify the following values:

**AUTO** requests that the base value for the time series be determined automatically.

*value* requests that *value* be used as the base value.

By default, BASE=AUTO.

**INTERMITTENT=***number*
specifies a number greater than one that is used to determine whether a time series is intermittent. If the median demand interval is equal to or greater than this number, then the series is assumed to be intermittent. By default, INTERMITTENT=2.

**METHOD=AVERAGE | BEST | CROSTON**
specifies the model for the demand series components.

You can specify the following values for *idm-method*:

**AVERAGE** uses a single smoothing model to forecast the average demand component.

**BEST** uses both CROSTON and AVERAGE methods to select the best performing forecast for the demand component series.

**CROSTON** uses two smoothing models to forecast the demand interval component and the demand size component.

By default, METHOD=BEST.

**TRANSFORM=***transform*
specifies the type of functional transformation.

You can specify the following values for *transform*:

**AUTO** requests to automatically choose between NONE and LOG based on the CRITERION= option.

**BOXCOX<(*value*)>** requests that a Box-Cox transformation be used with parameter *value*, where *value* is between –5 and 5. TRANSFORM=BOXCOX is equivalent to TRANSFORM=BOXCOX(1).

**LOG** requests a logarithmic transformation be used.

**LOGISTIC** requests a logistic transformation be used.

**NONE** requests no transformation be used.

**SQRT** requests a square-root transformation be used.

By default, TRANSFORM=AUTO.

## TRANSFORM Statement

**TRANSFORM** <*options*> **;**

The TRANSFORM statement specifies diagnostic tests and options related to the functional transformation of the series.

You can specify the following *options*:

**P=***number*
specifies the autoregressive order for the log transform test. By default, P=$\min(2, [T/10])$, where $T$ is the number of observations.

**SIGLEVEL=***value*
specifies the significance level that is used to decide whether or not the series requires a log transformation. The *value* should be in the range (0,1). This option overrides the value of the SIGLEVEL= option in the DIAGNOSE statement.

**TRANSOPT=MEAN | MEDIAN**
specifies whether to produce mean or median forecasts. If no transformation is applied to the series, then the mean and median forecasts are identical.

You can specify the following values:

**MEAN** requests that the inverse transform produce mean forecasts.

**MEDIAN** requests that the inverse transform produce median forecasts.

By default, TRANSOPT=MEAN.

**TYPE=***transform*
specifies the type of functional transformation.

You can specify the following values for *transform*:

**AUTO** requests to automatically choose between TRANSFORM=NONE and TRANSFORM=LOG based on the CRITERION= option in the DIAGNOSE statement.

**BOXCOX<(*value*)>** requests that a Box-Cox transformation be used with parameter *value*, where *value* is between –5 and 5. TRANSFORM=BOXCOX is equivalent to TRANSFORM=BOXCOX(1).

**LOG** requests that a logarithmic transformation be used.

**LOGISTIC** requests that a logistic transformation be used.

**NONE** requests that no transformation be used.

**SQRT** requests that square-root transformation be used.

By default, TYPE=AUTO.

## TREND Statement

**TREND** < *options* > **;**

The TREND statement specifies how to test the dependent series for simple differencing, seasonal differencing, or both. For more information about stationarity testing that is used during model identification, see Chapter 5, "The HPFDIAGNOSE Procedure."

You can specify the following *options*:

**DIFF=AUTO | NONE | 1 | 2 | (0:*range*)**

You can specify the following values:

**AUTO** tests for simple differencing.

**NONE** specifies that no simple differencing be used.

**1** specifies the simple differencing order to be $(1 - B)y_t$.

**2** specifies the simple differencing order to be $(1 - B)^2 y_t$

**(0 :** *range***)** specifies the range of simple differencing orders to test. *range* can be 0, 1, or 2.

By default, DIFF=AUTO.

**P=***number*

specifies the autoregressive order for the augmented unit root tests and a seasonality test. By default, P=$\min(5, [T/10])$, where $T$ is the number of observations.

**SDIFF=AUTO | NONE | 1 | 2**

You can specify the following values for *sdiff-opt*:

**AUTO** tests for seasonal differencing.

**NONE** specifies that no seasonal differencing be used.

**1** specifies the seasonal differencing order to be $(1 - B^s)y_t$, where $s$ is the seasonal period.

**2** specifies the seasonal differencing order to be $(1 - B^s)^2 y_t$, where $s$ is the seasonal period.

By default, SDIFF=AUTO.

**SIGLEVEL=***value*
specifies the significance level that is used to decide whether the series needs differencing. The *value* should be in the range (0,1). This option overrides the value of SIGLEVEL= option in the DIAGNOSE statement.

## UCM Statement

**UCM** < *options* > **;**

The UCM statement specifies options that are used to identify an appropriate unobserved component model (UCM). The UCM identification process is described in Chapter 5, "The HPFDIAGNOSE Procedure."

You can specify the following *options*:

**COMPONENT=(***components***)**
You can specify one or more of the following values for *components*. If you specify multiple values, you must enclose them in parentheses.

**ALL** tests component variances for significance based on seasonality of the time series. When the series tests positive for seasonality, the IRREGULAR, LEVEL, SLOPE, and SEASON components are tested; otherwise, the IRREGULAR, LEVEL, SLOPE, and CYCLE components are tested.

**AUTOREG** tests whether an autoregressive component is significant in the model.

**CYCLE** tests whether two cycle components are significant in the model. When two CYCLE components are included, the LEVEL component is also included. When the series fails to test positive for seasonality, the CYCLE component is not tested.

**DEPLAG** tests whether a dependent lag component is significant in the model. A dependent lag of order 1 is tested for significance.

**IRREGULAR** tests whether an irregular component is significant in the model.

**LEVEL** tests whether a level component is significant in the model.

**SEASON** tests whether a season component is significant in the model. When the series fails to test positive for seasonality, the SEASON component is not tested.

**SLOPE** tests whether a slope component is significant in the model. When found to be significant, the LEVEL component is also included.

By default, COMPONENT=ALL.

**REFINEPARMS=(***refine-opts***)**
specifies how to refine insignificant components of the final model. The *refine-opts* identify components to refine and the order of refinement. For a UCM, the parameters of the model are the componentwise variance estimates. Refinement for a UCM performs a significance test on those componentwise variance estimates and excludes components that are not significant.

You can specify the following values for *refine-opts*:

**FACTOR=ALL** tests the significance of all components in the model.

**FACTOR=INPUT** refines the parameters for INPUT components.

**SIGLEVEL=***value* specifies the significance level for refining insignificant components. The *value* should be between in the range (0,1). This option overrides the SIGLEVEL= option in the UCM statement.

By default, REFINEPARMS=(FACTOR=ALL).

**SIGLEVEL=***value*
specifies the significance level to use for deciding which component variances are significant. The *value* should be in the range (0,1). This option overrides the value of SIGLEVEL= option in the DIAGNOSE statement.

# Details: HPFDIAGSPEC Procedure

For information about the computational details that are associated with the time series diagnostic and model identification process, see Chapter 5, "The HPFDIAGNOSE Procedure." For information about the integration of PROC HPFDIAGSPEC diagnostic specifications with the HPFORECAST procedure, see Chapter 12, "The HPFORECAST Procedure."

# Examples: HPFDIAGSPEC Procedure

## Example 6.1: ARIMA Modeling Using Input Variables

This example demonstrates how to create a diagnostic specification that requests testing of the transformation and differencing of the input variables independent of the variable to be forecast. For the purpose of vetting the added complexity of ARIMA models that include predictors, an exponential smoothing model is generated in the set of model candidates.

The following code uses PROC HPFDIAGSPEC to generate a diagnostic control specification named EXAMPLE1 that is stored in the model repository catalog Work.Rep:

```
proc hpfdiagspec rep=work.rep name=example1;
   diagnose testinput=both selectinput=all
            criterion=mape;
   transform;
   esm;
   arimax identifyorder=both method=scan;
run;
```

The DIAGNOSE statement requests that the following be done:

- both dependent and independent series be tested for transformation
- all suitable inputs that improve model prediction be preserved in the final model
- the mean absolute percentage error (MAPE) fit statistic be used as the criterion for selecting the best forecast from among the generated candidates

The TRANSFORM statement requests that a transformation test be performed by using the default option settings. The ESM statement requests that the best performing exponential smoothing model be included as a candidate in the generated model selection list.

The ARIMAX statement specifies that the following be done:

- both ARIMA model identification orders be used for each dependent series
- the smallest canonical correlation analysis method be used for tentative ARMA order selection

The following code demonstrates the use of PROC HPFORECAST to execute the EXAMPLE1 diagnostic control specification to automatically generate ARIMA and ESM models for the Conb series from the data set Sashelp.Citimon. The MODE=AUTO option in the PROC HPFORECAST statement specifies that this is an automatic modeling invocation. The REP= option specifies the model repository catalog that holds the EXAMPLE1 specification, and the GLOBALDIAGNOSE= option specifies the name of the diagnostic control specification, EXAMPLE1. The FORECAST statement specifies the dependent variables for the procedure invocation, and the INPUT statement specifies the independent (predictor) variables to be considered in the model identification process.

```
proc hpforecast data=sashelp.citimon(obs=141)
                mode=auto
                rep=work.rep
                globaldiagnose=example1;
   timeid date interval=month;
   output outstatselect=conboss
          outfor=conbfor;
   forecast conb;
   input cciutc eec eegp exvus fm1 fm1d82;
run;
```

The following code prints the results of the automatic forecasting process from the OUTSTATSELECT= data set.

```
proc print data=conboss noobs;
   var _model_ _selected_ _label_ mape;
run;
```

For each candidate model, PROC PRINT displays the generated model name (_Model_), a formulaic synopsis of the model (_Label_), its selected state (_Selected_), and the model's mean absolute percentage error (MAPE) fit statistic (Mape) that is computed over the historical region of the Conb series.

Output 6.1.1 shows the results from this PROC PRINT step.

**Output 6.1.1** Automatic Forecasting Results

| _MODEL_ | _SELECTED_ | _LABEL_ | MAPE |
|---|---|---|---|
| ARIMAX | NO | ARIMA: CONB ~ D = (1) NOINT + INPUT: Lag(8)Dif(1) EEGP | 2.60000 |
| REGARIMA | NO | ARIMA: CONB ~ P = 1 NOINT + INPUT1: CCIUTC DEN = 1 + ... + INPUT3: FM1D82 NUM = 1 DEN = 2 | 2.46135 |
| ESM | YES | Damped-Trend Exponential Smoothing | 2.39353 |

The following code demonstrates the same automatic forecasting process by using the HPFDIAGNOSE and HPFENGINE procedures:

```
proc hpfdiagnose data=sashelp.citimon(obs=141)
                 testinput=both
                 selectinput=all
                 rep=work.diagrep
                 outest=work.diagest
                 criterion=mape;
   id date interval=month;
   forecast conb;
   input cciutc eec eegp exvus fm1 fm1d82;
   transform;
   esm;
   arimax identifyorder=both method=scan outlier=(detect=no);
run;

proc hpfengine data=sashelp.citimon(obs=141)
               rep=work.diagrep
               inest=work.diagest
               outstatselect=conboss2
               outfor=conbfor2;
   id date interval=month;
   forecast conb;
   stochastic cciutc eec eegp exvus fm1 fm1d82;
run;

proc compare data=conbfor
     compare=conbfor2;
run;
```

Output 6.1.2 shows the results of a PROC COMPARE of the forecasts from PROC HPFORECAST and PROC HPFENGINE.

**Output 6.1.2** Comparison of HPFORECAST and HPFENGINE Forecasts

```
                    Observation Summary

                 Observation      Base  Compare

                 First Obs           1        1
                 Last  Obs         153      153

Number of Observations in Common: 153.
Total Number of Observations Read from WORK.CONBFOR: 153.
Total Number of Observations Read from WORK.CONBFOR2: 153.

Number of Observations with Some Compared Variables Unequal: 0.
Number of Observations with All Compared Variables Equal: 153.

NOTE: No unequal values were found. All values compared are exactly
      equal.
```

## Example 6.2: Unobserved Components and ARIMA Models with Input Variables

This example illustrates the creation of a diagnostic specification that applies UCM and ARIMA models to automatically forecast ozone concentration. Various predictors are considered in the model identification and selection process. They include an indicator variable for the summer months (Summer), one for the winter months (Winter), and a level shift variable (X1) for measurements that are dated 1960 or greater .

```
data ozone;
   input ozone @@;
   label ozone  = 'Ozone Concentration'
         x1     = 'Intervention for post 1960 period'
         summer = 'Summer Months Intervention'
         winter = 'Winter Months Intervention';
   date = intnx( 'month', '31dec1954'd, _n_ );
   format date monyy.;
   month = month( date );
   year = year( date );
   x1 = year >= 1960;
   summer = ( 5 < month < 11 ) * ( year > 1965 );
   winter = ( year > 1965 ) - summer;
datalines;

   ... more lines ...
```

The following statements use PROC HPFDIAGSPEC to construct a diagnostic control specification named EXAMPLE2 to be stored in the model repository catalog Work.Rep. The TRANSFORM statement requests that a transform test be performed on the dependent series. The UCM statement requests the generation of a well-suited UCM model, and the ARIMAX statement requests the generation of a well-suited ARIMA model. Then PROC HPFORECAST is used to execute the EXAMPLE2 specification on the ozone concentration data set (Work.Ozone).

```
proc hpfdiagspec rep=work.rep name=example2;
   transform;
   ucm;
   arimax;
run;

proc hpforecast data=ozone
                mode=auto
                rep=work.rep
                globaldiagnose=example2;
   timeid date interval=month;
   output outstatselect=ozoneoss
          outfor=ozonefor
          outlog=ozonelog;
   forecast ozone;
   input x1 summer winter;
run;
```

The following code prints the results of the automatic forecasting process. For each candidate model, it includes the generated model name, a formulaic synopsis of the model, its selected state, and the model's root mean squared error (RMSE) fit statistic over the historical region of the time series (RMSE is the default selection criterion for PROC HPFDIAGSPEC).

```
proc print data=ozoneoss noobs;
   var _model_ _selected_ _label_ rmse;
run;
```

Output 6.2.1 shows that the explanatory variable Summer is selected in the chosen model.

**Output 6.2.1** Ozone Example Model Selection Results

| _MODEL_ | _SELECTED_ | _LABEL_ | RMSE |
|---|---|---|---|
| ARIMAX | YES | ARIMA: ozone ~ P = 1 D = (1) Q = ((1)(12)) NOINT + INPUT: Lag(4)Dif(1) summer DEN = 1 | 0.91529 |
| UCM | NO | UCM: ozone = LEVEL + SEASON + Lag(4) winter + ERROR | 1.01103 |

## Example 6.3: Automatic Model Combination

This example illustrates the automatic model combination feature of the HPFDIAGSPEC procedure.

The following statements use PROC HPFDIAGSPEC to generate a diagnostic control specification named EXAMPLE3 to include model combinations for the automatic models that are formulated when the diagnostic specification is subsequently executed. The combined forecast is to be produced using a simple average of the candidate forecasts. The COMBINE statement specifies that an OLS encompassing test be performed and that tests to access forecast quality be performed with an estimation region missing percentage of 25% and a horizon missing percentage of 50%. The DIAGNOSE statement requests that model selection be performed using the MAPE fit statistic and that a holdout region of six observations be used to perform model selection. The ESM and ARIMAX statements specify that ESM and ARIMA models, respectively, be diagnosed for each dependent series.

```
proc hpfdiagspec rep=work.rep name=example3;
   diagnose criterion=mape
            holdout=6;
   esm;
   arimax;
   combine method=average encompass=ols misspercent=25 hormisspercent=50;
run;
```

The following statements show the use of PROC HPFORECAST to execute the EXAMPLE3 diagnostic specification. This specification automatically generates an ESM model, an ARIMA model, and a combination of those models in the model selection list for the dependent series Air from the well-known Box-Jenkins airline passenger data set Sashelp.Air:

```
proc hpforecast data=sashelp.air
                mode=auto
                rep=work.rep
                globaldiagnose=example3;
   timeid date interval=month;
   output outest=airest
          outstatselect=aiross
```

```
          outfor=airfor;
   forecast air;
run;
```

The following code prints the results of the automatic forecasting process. For each candidate model, it includes the generated model name, a formulaic synopsis of the model, its selected state, and the model's MAPE fit statistic over the historical region of the time series.

```
proc print data=aiross noobs;
   var _model_ _selected_ _label_ mape;
run;
```

Output 6.3.1 shows the results of the automatic modeling process for the Air series.

**Output 6.3.1** Airline Passenger Miles Model Selection Results

| _MODEL_ | _SELECTED_ | _LABEL_ | MAPE |
|---|---|---|---|
| ARIMAX | USED | ARIMA: AIR ~ P = 1 D = (1,12) NOINT | 3.21832 |
| ESM | USED | Winters Method (Multiplicative) | 3.08450 |
| COMBINE | YES | COMBINE: AIR ~ AVG(ARIMAX,ESM) | 2.98020 |
| ARIMAX | NO | ARIMA: AIR ~ P = 1 D = (1,12) NOINT | 3.21832 |
| ESM | NO | Winters Method (Multiplicative) | 3.08450 |

# References

Box, G. E. P. and Jenkins, G. M. (1976), *Time Series Analysis: Forecasting and Control*, Rev. Edition, San Francisco: Holden-Day.

Box, G. E. P., Jenkins, G. M., and Reinsel, G. C. (1994), *Time Series Analysis: Forecasting and Control*, 3rd Edition, Englewood Cliffs, NJ: Prentice-Hall.

# Chapter 7
# The HPFENGINE Procedure

## Contents


Overview: HPFENGINE Procedure . . . . . **182**
Getting Started: HPFENGINE Procedure . . . . . **183**
Syntax: HPFENGINE Procedure . . . . . **185**
- Functional Summary . . . . . 186
- PROC HPFENGINE Statement . . . . . 188
- ADJUST Statement . . . . . 200
- BY Statement . . . . . 201
- CONTROL Statement . . . . . 202
- EXTERNAL Statement . . . . . 203
- FORECAST Statement . . . . . 204
- ID Statement . . . . . 205
- INPUT Statement . . . . . 208
- SCORE Statement . . . . . 209
- STOCHASTIC Statement . . . . . 210

Details: HPFENGINE Procedure . . . . . **211**
- Accumulation . . . . . 212
- Missing Value Interpretation . . . . . 213
- Adjustment Operations . . . . . 214
- Diagnostic Tests . . . . . 215
- Model Selection . . . . . 216
- IDM Model Combinations . . . . . 217
- Transformations . . . . . 217
- Parameter Estimation . . . . . 217
- Missing Value Modeling Issues . . . . . 218
- Forecasting . . . . . 218
- Inverse Transformations . . . . . 218
- Statistics of Fit . . . . . 218
- Thread Usage in PROC HPFENGINE . . . . . 218
- Data Set Input/Output . . . . . 219
- ODS Table Names . . . . . 229
- ODS Graphics . . . . . 231

Examples: HPFENGINE Procedure . . . . . **233**
- Example 7.1: The TASK Option . . . . . 233
- Example 7.2: Different Types of Input . . . . . 236
- Example 7.3: Incorporating Events . . . . . 238
- Example 7.4: Using the SCORE Statement . . . . . 243

Example 7.5: HPFENGINE and HPFDIAGNOSE Procedures . . . . . 247
Example 7.6: The ADJUST Statement . . . . . 250
Example 7.7: Multiple Repositories . . . . . 252
Example 7.8: ODS Graphics . . . . . 254
Example 7.9: Combining Judgmental Forecasts . . . . . 258
References . . . . . **260**

# Overview: HPFENGINE Procedure

The HPFENGINE procedure provides an automatic way to generate forecasts for many time series or transactional data in one step. The procedure can automatically choose the best forecast model from a user-defined model list or a default model list. Specifications for the candidate forecast models are independent of any data series. You can generate the specifications or choose them from a default set. Supported model families include the following:

- smoothing
- intermittent demand
- unobserved component
- ARIMA

Additionally, you can provide user-defined forecasts with data drawn from a SAS data set or through the use of user-written functions.

All parameters associated with the forecast model are optimized based on the data. The HPFENGINE procedure selects the appropriate model for each data series based on one of several model selection criteria.

The procedure operates in a variety of modes. At its most comprehensive, all appropriate candidate models from a list are fit to a particular series, and the model that produces the best fit (based on a user-determined criterion) is determined. Forecasts are then produced from this model. It is also possible to skip the selection process and fit a particular model and produce subsequent forecasts. Finally, given a set of parameter estimates and model specifications, the procedure can bypass the fit stage entirely and calculate forecasts directly.

The HPFENGINE procedure writes the time series extrapolated by the forecasts, the series summary statistics, the forecasts and confidence limits, the parameter estimates, and the fit statistics to output data sets.

The HPFENGINE procedure can forecast both time series data (whose observations are equally spaced at a specific time interval) and transactional data (whose observations are not spaced at any particular time interval). For transactional data, the data are accumulated at a specified time interval to form a time series.

# Getting Started: HPFENGINE Procedure

In its simplest usage, the HPFENGINE procedure produces results similar to those of the HPF procedure:

```
proc hpfengine data=sashelp.air
               print=(select summary);
   id date interval=month;
   forecast air;
run;
```

The GLOBALSELECTION= and REPOSITORY= options assume their respective default values. Therefore automatic selection is performed for the AIR series in SASHELP.AIR by using the models specified in the BEST selection list. The selection list BEST is found, together with the smoothing models it references, in SASHELP.HPFDFLT.

The results of the automatic model selection are displayed in Output 7.1. A summary of the forecast is shown in Output 7.2.

**Figure 7.1** Selection Results

**The HPFENGINE Procedure**

| Model Selection Criterion = RMSE | | | |
|---|---|---|---|
| **Model** | **Statistic** | **Selected** | **Label** |
| **SMSIMP** | . | Removed | Simple Exponential Smoothing |
| **SMDAMP** | . | Removed | Damped-Trend Exponential Smoothing |
| **SMLIN** | . | Removed | Linear Exponential Smoothing |
| **SMADWN** | 12.245596 | No | Winters Method (Additive) |
| **SMWINT** | 10.579085 | Yes | Winters Method (Multiplicative) |
| **SMSEAS** | 14.169905 | No | Seasonal Exponential Smoothing |

**Figure 7.2** Forecast Summary

| Forecast Summary | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Variable** | **Value** | **JAN1961** | **FEB1961** | **MAR1961** | **APR1961** | **MAY1961** | **JUN1961** | **JUL1961** | **AUG1961** | **SEP1961** | **OCT1961** |
| **AIR** | Predicted | 445.2972 | 418.1426 | 464.0889 | 494.0261 | 504.9584 | 572.5947 | 662.7040 | 653.7742 | 545.8935 | 487.7147 |

| Forecast Summary | | |
|---|---|---|
| **Variable** | **NOV1961** | **DEC1961** |
| **AIR** | 415.2594 | 459.6067 |

The first four models in the selection list are nonseasonal smoothing models. The HPFENGINE procedure determined that the series AIR in SASHELP.AIR was seasonal and attempted to fit only seasonal models.

The multiplicative Winters method produced a fit with the smallest root mean squared error (RMSE).

As another example, consider the problem of comparing the performance of multiple ARIMA models for a particular series. This is done by using the HPFARIMASPEC procedure as follows to specify the models.

```
proc hpfarimaspec repository=sasuser.repository
                  name=arima1;
   dependent symbol=air q=1
             diflist=(1 12) noint transform=log;
run;

proc hpfarimaspec repository=sasuser.repository
                  name=arima2;
   dependent symbol=air q=(1,12)
             diflist=(1 12) noint transform=log;
run;
```

The model specifications are grouped together in a selection list. Selection lists are built with the HPFSELECT procedure.

```
proc hpfselect repository=sasuser.repository
               name=myselect;
   spec arima1 arima2;
run;
```

The HPFENGINE procedure uses the GLOBALSELECTION= option to reference the selection list that contains the two ARIMA models. Selection /results are shown in Output 7.3, and the forecast plot is shown in Figure 7.4.

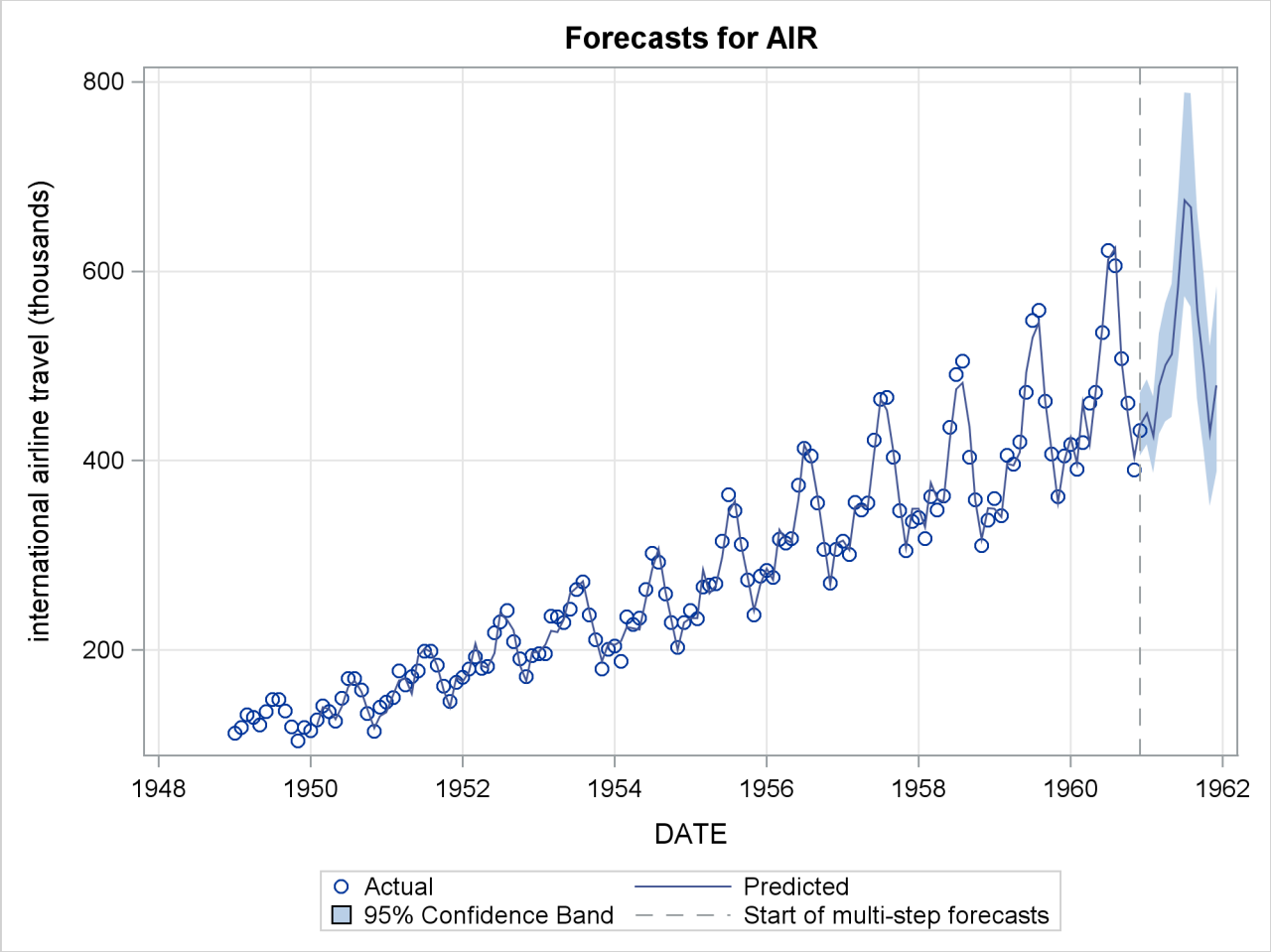
```
proc hpfengine data=sashelp.air
               repository=sasuser.repository
               globalselection=myselect
               print=(select forecasts)
               plot=forecasts;
   id date interval=month;
   forecast air;
run;
```

**Figure 7.3** Selection and Forecast Results

**The HPFENGINE Procedure**

| Model Selection Criterion = MAPE | | | |
|---|---|---|---|
| **Model** | **Statistic** | **Selected** | **Label** |
| **ARIMA1** | 3.2553815 | No | ARIMA: Log( AIR ) ~ D = (1,12) Q = 1 NOINT |
| **ARIMA2** | 2.9672282 | Yes | ARIMA: Log( AIR ) ~ D = (1,12) Q = (1,12) NOINT |

**Figure 7.4** Forecasts




# Syntax: HPFENGINE Procedure

The following statements are used with the HPFENGINE procedure:

**PROC HPFENGINE** *options* ;
**ADJUST** *variable = ( variable-list ) / options* ;
**BY** *variables* ;
**CONTROL** *variable-list / options* ;
**EXTERNAL** *variable-list / options* ;
**FORECAST** *variable-list / options* ;
**ID** *variable INTERVAL= interval options* ;
**INPUT** *variable-list / options* ;
**SCORE** ;
**STOCHASTIC** *variable-list / options* ;

## Functional Summary

Table 7.1 summarizes the statements and options that control the HPFENGINE procedure.

**Table 7.1** HPFENGINE Functional Summary

| Description | Statement | Option |
|---|---|---|
| **Statements** | | |
| Specifies BY group processing | BY | |
| Specifies variables to forecast | FORECAST | |
| Specifies the time ID variable | ID | |
| Specifies input variables | INPUT | |
| Specifies input variables | STOCHASTIC | |
| Specifies input variables | CONTROL | |
| Specifies input forecasts, bounds, PIs | EXTERNAL | |
| Requests creation of score files | SCORE | |
| **Model Selection Options** | | |
| Specifies location of model repository | PROC HPFENGINE | REPOSITORY= |
| Specifies model selection list | PROC HPFENGINE | GLOBALSELECTION= |
| **Data Sets** | | |
| Specifies the auxiliary input data sets | PROC HPFENGINE | AUXDATA= |
| Specifies the input data sets | PROC HPFENGINE | DATA= |
| Specifies the data set containing mapping and estimate data | PROC HPFENGINE | INEST= |
| Specifies the events data set | PROC HPFENGINE | INEVENT= |
| Specifies the output data set | PROC HPFENGINE | OUT= |
| Specifies the accumulation mode output data set | PROC HPFENGINE | OUTACCDATA= |
| Specifies the forecast component output data set | PROC HPFENGINE | OUTCOMPONENT= |
| Specifies the output data set to store mapping and estimate data | PROC HPFENGINE | OUTEST= |
| Specifies the forecast output data set | PROC HPFENGINE | OUTFOR= |
| Specifies the output data set to store independent variable time series | PROC HPFENGINE | OUTINDEP= |
| Specifies the detailed model information output data set | PROC HPFENGINE | OUTMODELINFO= |
| Specifies the forecast procedure run information output data set | PROC HPFENGINE | OUTPROCINFO= |
| Specifies the forecast model statistics-of-fit output data set | PROC HPFENGINE | OUTSTAT= |
| Specifies the candidate model statistics-of-fit output data set | PROC HPFENGINE | OUTSTATSELECT= |
| Specifies the summary output data set | PROC HPFENGINE | OUTSUM= |

**Table 7.1** *continued*

| Description | Statement | Option |
|---|---|---|
| **Accumulation Options** | | |
| Specifies the accumulation frequency | ID | INTERVAL= |
| Specifies the length of seasonal cycle | PROC HPFENGINE | SEASONALITY= |
| Specifies the interval alignment | ID | ALIGN= |
| Specifies the starting time ID value | ID | START= |
| Specifies the ending time ID value | ID | HORIZONSTART= |
| Specifies the starting time ID of forecast horizon | ID | END= |
| Specifies the date format | ID | FORMAT= |
| Specifies the accumulation statistic | ID, FORECAST, INPUT, STOCHASTIC, CONTROL, EXTERNAL | ACCUMULATE= |
| Specifies the missing value interpretation | ID, FORECAST, INPUT, STOCHASTIC, CONTROL, EXTERNAL | SETMISSING= |
| Specifies the zero value interpretation | ID, FORECAST, INPUT, STOCHASTIC, CONTROL, EXTERNAL | ZEROMISS= |
| Specifies the trim missing values | ID, FORECAST, INPUT, STOCHASTIC, CONTROL, EXTERNAL | TRIMMISS= |
| **Forecasting Horizon Options** | | |
| Specifies data to hold back | PROC HPFENGINE | BACK= |
| Modifies the holdback behavior for insufficient data | PROC HPFENGINE | FORCEBACK |
| Specifies forecast horizon or lead | PROC HPFENGINE | LEAD= |
| **Forecasting Control Options** | | |
| Specifies forecasting control options | PROC HPFENGINE | TASK= |
| Replaces missing values in OUTFOR= | FORECAST | REPLACEMISSING |

**Table 7.1** *continued*

| Description | Statement | Option |
|---|---|---|
| Does not replace missing values in OUTFOR= | FORECAST | NOREPLACE |
| **Scoring Options** | | |
| Specifies the location of the score repository | PROC HPFENGINE | SCOREREPOSITORY= |
| **Printing and Plotting Options** | | |
| Specifies graphical output | PROC HPFENGINE | PLOT= |
| Specifies printed output | PROC HPFENGINE | PRINT= |
| Specifies detailed printed output | PROC HPFENGINE | PRINTDETAILS |
| **Miscellaneous Options** | | |
| Specifies error printing options | PROC HPFENGINE | ERRORCONTROL= |
| Specifies exception handling | PROC HPFENGINE | EXCEPTIONS= |
| Specifies control for statistics-of-fit computation | PROC HPFENGINE | STAT= |
| Specifies that analysis variables are processed in sorted order | PROC HPFENGINE | SORTNAMES |

## PROC HPFENGINE Statement

**PROC HPFENGINE** *options* **;**

The PROC HPFENGINE statement invokes the HPFENGINE procedure. You can specify the following *options*:

**AUXDATA=***SAS-data-set*
names a SAS data set that contains auxiliary input data for the procedure to use for supplying explanatory variables in a forecast. See the section "AUXDATA= Data Set" on page 220 for more information.

**BACK=***n*
specifies the number of observations before the end of the data where the multistep forecasts are to begin. This option is often used to obtain performance statistics. See the PRINT= option details about printing performance statistics. The default is BACK=0.

**COMPONENTS=INTEGRATE**
requests that the component series in the OUTCOMPONENT= data set be generated so that they sum to the forecast series. This option affects only the components that are produced by ARIMA models that include differencing. By default, the ARIMA components are generated without the summability constraint.

**DATA=***SAS-data-set*
names the SAS data set that contains the input data for the procedure to forecast. If the DATA= option is not specified, the most recently created SAS data set is used.

**ERRORCONTROL=(SEVERITY=(***severity-options***) STAGE= (***stage-options***) MAXMESSAGE=***number***)**
enables finer control of message printing. The error severity level and HPFENGINE procedure processing stages are set independently. A logical 'and' is taken over all the specified options, and any message that tests true against the results of the 'and' is printed.

Available *severity-options* are as follows:

| | |
|---|---|
| LOW | specifies that only low severity, minor issues be printed. |
| MEDIUM | specifies that only medium-severity problems be printed. |
| HIGH | specifies that only severe errors be printed. |
| ALL | specifies that errors of all severity levels (LOW, MEDIUM, and HIGH) be printed. |
| NONE | specifies that no messages from the HPFENGINE procedure be printed. |

Available *stage-options* are as follows:

| | |
|---|---|
| PROCEDURELEVEL | specifies that only errors that occur during option processing and validation be printed. |
| DATAPREP | specifies that only errors that occur during the accumulation of data and the application of SETMISS= and ZEROMISS= options be printed. |
| SELECTION | specifies that only errors that occur during the model selection process be printed. |
| ESTIMATION | specifies that only errors that occur during the model parameter estimation process be printed. |
| FORECASTING | specifies that only errors that occur during the model evaluation and forecasting process be printed. |
| ALL | is the same as specifying all PROCEDURELEVEL, DATAPREP, SELECTION, ESTIMATION, and FORECASTING options. |

Examples are as follows:

```
errorcontrol=(severity=(high medium) stage=all);
```

prints high- and moderate-severity errors at any processing stage of PROC HPFENGINE.

```
errorcontrol=(severity=high stage=dataprep);
```

prints high-severity errors only during the data preparation.

```
errorcontrol=(severity=none stage=all);
```

turns off messages from PROC HPFENGINE.

```
errorcontrol=(severity=(high medium low)
              stage=(procedurelevel dataprep selection
                     estimation forecasting));
```

specifies the default behavior. Also the following statement specifies the default behavior:

```
errorcontrol=(severity=all stage=all)
```

**EXCEPTIONS=***except-option*

specifies the desired handling of arithmetic exceptions during the run. You can specify *except-option* as one of the following:

IGNORE specifies that PROC HPFENGINE stop on an arithmetic exception. No recovery is attempted. This is the default behavior if the EXCEPTIONS= option is not specified.

CATCH(ESM) specifies that PROC HPFENGINE generate a forecast based on using its default exponential smoothing model for the variable that produces the arithmetic exception in the current BY group. The ESM model is equivalent to the default model used by the HPFESMSPEC procedure with no modifications.

CATCH(RW) specifies that PROC HPFENGINE generate a forecast based on a zero-drift random walk model for the variable that produces the exception in the current BY group.

CATCH(MISSING) specifies that PROC HPFENGINE generate a forecast of missing values for the variable that produces the exception in the current BY group.

The order of CATCH handling corresponds to the order of the preceding list. If CATCH(ESM) handling produces an arithmetic exception, it attempts to generate a forecast by using CATCH(RW) semantics. Likewise, if CATCH(RW) handling produces an arithmetic exception, it generates a missing value forecast.

**FORCEBACK**

specifies alternate behavior for hold back when there is insufficient data to fit any model and honor the requested hold back sample size. The default behavior dynamically resets to no hold back samples for the current BY group if excluding the requested number of hold back samples leaves insufficient non-missing observations to fit any model. This allows PROC HPFENGINE to make an absolute effort to generate a statistical forecast for each BY group regardless of the condition of the input data set. Sometimes it can be desirable to sacrifice this behavior in order to preserve a consistent time base across the BY groups in the OUT= and OUTFOR= data sets. This is the purpose of FORCEBACK. It alters the default behavior of PROC HPFENGINE to ensure that a consistent hold back sample size is used across all BY groups of the input data set. If a series has insufficient observations to permit a model fit after excluding hold back samples, a missing value forecast is generated over the range of time ID's corresponding to the complete forecast horizon including the hold back region. Additionally, for both OUTSUM= and OUTEST= data sets, the _STATUS_ variable is set to indicate the excepted forecast produced by the presence of FORCEBACK.

**GLOBALSELECTION=***catalog-name*

specifies the name of a catalog entry that serves as a model selection list. This is the selection list used to forecast all series if no INEST= data set is provided. It is also the selection list used if individual model selections are missing in the INEST= data set when INEST= is provided. If REPOSITORY= is not present, GLOBALSELECTION defaults to BEST, specified in SASHELP.HPFDFLT.

**IGNORECHOOSE**

specifies that the CHOOSE= option in the HPFSELECT procedure be ignored when selecting a model in the candidate model list. The best model is selected regardless of the model chosen in the CHOOSE= option in the HPFSELECT procedure.

**INEST=***SAS-data-set*

contains information that maps forecast variables to models or selection lists, and data set variables to model variables. It can also contain parameter estimates used if the TASK=FORECAST or TASK=UPDATE options are present. INEST= is optional. See the description of the GLOBALSELECTION= option for more information.

**INEVENT=***SAS-data-set*

contains information that describes predefined events. This data set is usually created by the HPFEVENTS procedure. This option is only used if events are included in a model.

**LEAD=***n*

specifies the number of periods ahead to forecast (forecast lead or horizon). The default is LEAD=12.

The LEAD= value is relative to the last observation in the input data set and not to the last nonmissing observation of a particular series. Thus, if a series has missing values at the end, the actual number of forecasts computed for that series will be greater than the LEAD= value.

**OUT=***SAS-data-set*

names the output data set to contain the forecasts of the variables specified in the subsequent FORECAST statements. If an ID variable is specified, it will also be included in the OUT= data set. The values are accumulated based on the ACCUMULATE= option and forecasts are appended to these values. If the OUT= data set is not specified, a default output data set DATA*n* is created. If you do not want the OUT= data set created, then use OUT=_NULL_.

**OUTACCDATA=***SAS-data-set*

names the output data set to store accumulation mode used for each forecast variable specified for the HPFENGINE procedure. This data set is commonly used to transfer information to the HPFTEMPRECON procedure so it can properly affect constraint formulation for the benchmarking process.

**OUTCOMPONENT=***SAS-data-set*

names the output data set to contain the forecast components. The components included in the output depend on the model.

**OUTEST=***SAS-data-set*

contains information that maps forecast variables to model specifications, and data set variables to model variables and parameter estimates.

An OUTEST= data set will frequently be used as the INEST= data set for subsequent invocations of PROC HPFENGINE. In such a case, if the PROC HPFENGINE statement option TASK=FORECAST is used, forecasts are generated using the parameter estimates found in this data set and are not reestimated.

**OUTFOR=***SAS-data-set*
names the output data set to contain the forecast time series components (actual, predicted, lower confidence limit, upper confidence limit, prediction error, and prediction standard error). The OUTFOR= data set is particularly useful for displaying the forecasts in tabular or graphical form.

**OUTINDEP=***SAS-data-set*
names the output data set to contain input in the forecasting process. This information is useful if future values of input variables are automatically supplied by the HPFENGINE procedure. Such a case would occur if one or more input variables are listed in either the STOCHASTIC or the CONTROLLABLE statement and if there are missing future values of these input variables.

**OUTMODELINFO=***SAS-data-set*
names the output data set to contain detailed information about the selected forecast model. The data set has information such as the model family, presence or absence of inputs, events and outliers, and so forth.

**OUTPROCINFO=***SAS-data-set*
names the output data set to contain information in the SAS log, specifically the number of notes, errors, and warnings and the number of series processed, forecasts requested, and forecasts failed.

**OUTSTAT=***SAS-data-set*
names the output data set to contain the statistics of fit (or goodness-of-fit statistics). The OUTSTAT= data set is useful for evaluating how well the model fits the series. The statistics of fit are based on the entire range of the time series.

**OUTSTATSELECT=***SAS-data-set*
names the output data set to contain statistics of fit for all of the candidate models fit during model selection. The OUTSTATSELECT= data set is useful for comparing the performance of various models.

**OUTSUM=***SAS-data-set*
names the output data set to contain the summary statistics and the forecast summation. The summary statistics are based on the accumulated time series for the dependent variables acquired from the DATA= data set. The forecast summations are based on the LEAD= option setting. The OUTSUM= data set is particularly useful when forecasting large numbers of series and a summary of the results is needed.

**PLOT <(***global-plot-options***)> =***plot-request* *< (options) >*

**PLOT <(***global-plot-options***)> =(***plot-request* *< (options) >* **< ...** *plot-request* *< (options) >* **>)**
specifies the graphical output that you want. By default, the HPFENGINE procedure produces no graphical output.

Any specified *global-plot-options* are available to all of the requested plots. They are interpreted by the indidivual plots that are requested, so it is possible to specify *global-plot-options* that have no effect for a particular set of plot requests. Each of the following *global-plot-options* description includes the set of plots for which it has some effect.

You can specify the following *global-plot-options*:

**NLAGS=***number*

specifies the number of lags to include in various error series plots. This option affects the ACF, IACF, PACF, and WN plots. The default number of lags is given by

$$\mathrm{NLAGS} = \begin{cases} \min\{120, \max\{3 \times \mathrm{SEASONALITY}, 24\}\} & \text{if } \mathrm{SEASONALITY} > 1 \\ 24 & \text{if } \mathrm{SEASONALITY} = 0 \end{cases}$$

In all cases, NLAGS is automatically limited to half of the number of observations in the error series.

You can specify the following *plot-requests*:

ACF plots the prediction error autocorrelation function.

ALL is equivalent to specifying PLOT=(ACF COMPONENTS ERRORS FORECASTCYCLES FORECASTS FORECASTSONLY IACF MODELFORECASTS MODELS PERIODOGRAM SPECTRUM WN).

BASIC is equivalent to specifying PLOT=(CORR ERRORS MODELFORECASTS). In the context of IDM models, the stocking level plot is generated in place of the prediction error correlation panel plot.

CANDIDATES plots model (forecast) and error series for each candidate model that is fit to the dependent series.

COMPONENTS plots the forecast components.

CORR plots the prediction error series graphics panel that contains the ACF, IACF, PACF, and white noise probability plots. In the context of IDM models, PLOT=CORR produces no additional output.

ERRORS=<(SMOOTH=*method*)> plots prediction error time series. You can specify the following *method* values:

- NONE specifies that no smoothed error series overlay be applied to the plot.
- LOESS specifies that a LOESS fit be used to generate a smoothed error series overlay for the plot.

PLOT=ERRORS is equivalent to PLOT=ERRORS(SMOOTH=LOESS).

FORECASTCYCLES plots the forecast seasonal cycles.

FORECASTS plots the actual time series and its one-step-ahead forecast over the historical period; plots the forecast and its confidence bands over the forecast horizon.

FORECASTSONLY plots the forecast and its confidence bands over the forecast horizon only.

IACF plots the prediction error inverse autocorrelation function.

MODELFORECASTS plots the one-step-ahead model forecast and its confidence bands in the historical period; plots the forecast and its confidence bands over the forecast horizon.

MODELS plots the one-step-ahead model forecast and its confidence bands in the historical period.

PACF plots the prediction error partial autocorrelation function.

PERIODOGRAM plots periodogram of the prediction error series.

SPECTRUM=< (*spectrum-options*) > plots the periodogram and the smoothed periodogram of the prediction error series in a single graph.

You can specify the following *spectrum-options*:

ALPHA=*value* specifies the significance level for upper and lower confidence limits around the smoothed periodogram estimates of spectral density. By default, ALPHA=0.4.

CENTER=YES | NO specifies whether mean adjustment is desired for the error series before computation of the smoothed periodogram estimates of spectral density. By default, CENTER=NO.

PLOT=SPECTRUM is equivalent to PLOT=SPECTRUM(ALPHA=0.4 CENTER=NO).

WN plots white noise graphics.

For example, PLOT=FORECASTS plots the forecasts for each series; PLOT(NLAGS=12)=(ACF IACF) produces the prediction error autocorrelation and inverse autocorrelation plots for which the number of lags restricted to 12.

**PRINT=***option* | **(***options* **)**

specifies printed output. By default, the HPFENGINE procedure produces no printed output.

The following printing options are available:

ALL is the same as specifying PRINT=(ESTIMATES SELECT FORECASTS STATISTICS BIAS DESCSTATS). PRINT=(ALL CANDIDATES COMPONENTS PERFORMANCE PERFORMANCESUMMARY PERFORMANCEOVERALL) prints all the options listed.

BIAS prints model bias information.

CANDIDATES prints parameter estimates for each candidate model evaluated during the model selection step.

COMPONENTS prints forecast model components.

DESCSTATS prints descriptive statistics the series forecast series.

ESTIMATES prints parameter estimates for the selected model.

FORECASTS prints the forecasts.

PERFORMANCE prints the performance statistics for each forecast.

PERFORMANCESUMMARY prints the performance summary for each BY group.

PERFORMANCEOVERALL prints the performance summary for all of the BY groups.

SELECT prints the label and fit statistics for each model in the selection list.

STATISTICS prints the statistics of fit.

SUMMARY prints the forecast summary.

**PRINTDETAILS**
specifies that output requested with the PRINT= option be printed in greater detail.

**REPOSITORY=***catalog-name*
is a two-level SAS catalog name that specifies the location of the model repository. The REPOSITORY= option can also be specified as MODELREPOSITORY=, MODELREP=, or REP=. The default for this option is Sashelp.Hpfdflt.

**SCOREREPOSITORY=***catalog-name*
is a two-level SAS catalog name that specifies the location of the model score repository. Score files are written to this repository if the SCORE statement is used in the HPFENGINE procedure. There is no default score repository. The presence of a SCORE statement requires that the SCOREREPOSITORY= option also be present.

**SEASONALITY=***n*
specifies the length of the seasonal cycle. For example, SEASONALITY=3 means that every group of three observations forms a seasonal cycle. The SEASONALITY= option is applicable only for seasonal forecasting models. By default, the length of the seasonal cycle is 1 (no seasonality) or the length implied by the INTERVAL= option specified in the ID statement. For example, INTERVAL=MONTH implies that the length of the seasonal cycle is 12.

**SORTNAMES**
specifies that the variables specified in the FORECAST statement be processed in sorted order.

**STAT=ADJUST | NOADJUST**
specifies whether the statistics of fit be computed before or after the forecast dependent series is post-adjusted.

ADJUST — specifies that statistics of fit be computed after the forecast series is post-adjusted. This is the default.

NOADJUST — specifies that statistics of fit be computed before the forecast series is post-adjusted.

**TASK=***action*(*override-options*)
controls the model selection and parameter estimation process. The following *action* values are valid. All *override-options* are admissible for any *action*. See section "TASK= Option Details" on page 196 for details.

SELECT — performs model selection, estimates parameters of the selected model, and produces forecasts. This is the default.

SELECT(*override-options*) — performs model selection, estimates parameters of the selected model, produces forecasts, and potentially overrides settings in the model selection list. If a selection list does not specify a particular item and that item is specified with a TASK=SELECT option, the value as set in TASK=SELECT is used. If an option is specified in selection list, the corresponding value set in TASK=SELECT is not used unless the OVERRIDE option is also present. See section "TASK= Option Details" on page 196 for possible values of *override-options* and other details.

FIT — estimates parameters by using the model specified in the INEST= data set, and then forecasts. No model selection is performed. New parameter estimates are

saved if an OUTEST= data set is specified. See section "TASK= Option Details" on page 196 for details.

FIT(*override-options*) performs the same action as TASK=FIT supplemented by any *override-options* specified. See section "TASK= Option Details" on page 196 for details.

UPDATE estimates parameters by using the model specified in the INEST= data set, and then forecasts. TASK=UPDATE differs from TASK=FIT in that the parameters found in the INEST= data set are used as starting values in the estimation. No model selection is performed. New parameter estimates are saved if an OUTEST= data set is specified. See section "TASK= Option Details" on page 196 for details.

UPDATE(*override-options*) performs the same action as TASK=UPDATE supplemented by any *override-options* specified. See section "TASK= Option Details" on page 196 for details.

FORECAST forecasts using model and parameters specified in the INEST= data set. No parameter estimation nor model selection occurs. See section "TASK= Option Details" on page 196 for details.

FORECAST(*override-options*) performs the same action as TASK=FORECAST supplemented by any *override-options* specified. See section "TASK= Option Details" on page 196 for details.

## TASK= Option Details

This section describes the *override-options* that can be specified in any TASK=*action* context. Many of these settings are used only during the model selection phase of PROC HPFENGINE processing (that is, during the processing performed when you specify TASK=SELECT). However, there can be a need for them in any TASK=*action* context. PROC HPFENGINE can generate special placeholder forecasts for some of the time series it encounters during a TASK=SELECT run. These are generally the result of nominal data that prohibits PROC HPFENGINE from performing an evaluation of the model selection list for the series. As such, these forecasts represent excepted cases. Many of these cases are triggered from various *override-options* in effect during the TASK=SELECT run. For example, if you specified TASK=SELECT(MINOBS=6), any series with fewer than six nonmissing observations would trigger a forecast using the _MEAN_ model. Another example would be the _ZERO_ model that arises from use of the ENDZEROS= option. PROC HPFENGINE recognizes these excepted cases when processing the information from its INEST= data set that is saved as the OUTEST= data set from a previous PROC HPFENGINE run. When recognized in this way, PROC HPFENGINE attempts to perform model selection processing on those excepted cases regardless of the TASK=*action* requested. To deal with any such cases properly and ensure consistent behavior, you should specify the same set of *override-options* for your PROC HPFENGINE runs without regard to the *action* value specified.

You can specify these *override-options* in the context of any admissible *action* for TASK=.

ALPHA=*number* specifies the significance level to use in computing the confidence limits of the forecast. The ALPHA= value must be between 0 and 1. For example, ALPHA=0.05 produces 95% confidence intervals.

If a selection list does not specify ALPHA and ALPHA is specified in TASK=, the value that is specified takes effect. If ALPHA is specified in the selection list, the corresponding value specified in TASK= is not used unless the OVERRIDE option is also present.

CRITERION=*option* specifies the model selection criterion (statistic of fit) to be used to select from several candidate models. The following list shows the valid values for the CRITERION=*option* and its corresponding statistic of fit:

| | |
|---|---|
| SSE | sum of square error |
| MSE | mean square error |
| RMSE | root mean squared error |
| UMSE | unbiased mean squared error |
| URMSE | unbiased root mean squared error |
| MAXPE | maximum percent error |
| MINPE | minimum percent error |
| MPE | mean percent error |
| MAPE | mean absolute percent error |
| MDAPE | median absolute percent error |
| GMAPE | geometric mean absolute percent error |
| MINAPES | minimum absolute error percent of standard deviation |
| MAXAPES | maximum absolute error percent of standard deviation |
| MAPES | mean absolute error percent of standard deviation |
| MDAPES | median absolute error percent of standard deviation |
| GMAPES | geometric mean absolute error percent of standard deviation |
| MINPPE | minimum predictive percent error |
| MAXPPE | maximum predictive percent error |
| MPPE | mean predictive percent error |
| MAPPE | symmetric mean absolute predictive percent error |
| MDAPPE | median absolute predictive percent error |
| GMAPPE | geometric mean absolute predictive percent error |
| MINSPE | minimum symmetric percent error |
| MAXSPE | maximum symmetric percent error |
| MSPE | mean symmetric percent error |
| SMAPE | symmetric mean absolute percent error |
| MDASPE | median absolute symmetric percent error |
| GMASPE | geometric mean absolute symmetric percent error |
| MINRE | minimum relative error |
| MAXRE | maximum relative error |
| MRE | mean relative error |
| MRAE | mean relative absolute error |
| MDRAE | median relative absolute error |

| | |
|---|---|
| GMRAE | geometric mean relative absolute error |
| MAXERR | maximum error |
| MINERR | minimum error |
| ME | mean error |
| MAE | mean absolute error |
| MASE | mean absolute scaled error |
| RSQUARE | R square |
| ADJRSQ | adjusted R square |
| AADJRSQ | Amemiya's adjusted R square |
| RWRSQ | random walk R square |
| AIC | Akaike information criterion |
| AICC | finite sample corrected AIC |
| SBC | Schwarz Bayesian information criterion |
| APC | Amemiya's prediction criterion |

ENDZEROS=(MAXNUM=*number* MAXPCT=*percent* MINOBS=*number*) specifies criteria that control the use of a _ZERO_ model for a series. A _ZERO_ model is used when the number of trailing zero values in the series fails to indicate that a nonzero model is appropriate. (By default, no _ZERO_ model testing is performed.) If CHOOSE= is specified in the model selection list that is used for a series, these options have no effect and the chosen model is used.

You can specify the following suboptions:

MAXNUM=*number* specifies the count of trailing zero values needed in order to consider a nonzero model. If the series has *number* or fewer trailing zero observations, then a nonzero model is considered. For example, if you specify ENDZEROS=(MAXNUM=10), then a nonzero model is considered only if the series has 10 or fewer trailing zero values.

MAXPCT=*P* specifies the maximum percentage of trailing zero values relative to the number of nonzero values in the entire series. If the number of trailing zero observations in the series is fewer than or equal to *P*% of the nonmissing and nonzero values of the entire series, then a nonzero model is considered. The value of *P* is in the range (0,100). If you specify ENDZEROS=(MAXPCT=20), then a nonzero model is considered only if the number of trailing zero values is fewer than or equal to 20% of the number of nonmissing and nonzero values of the entire series.

MINOBS=*number* specifies the minimum series length that is required to perform the _ZERO_ model test. For example, if you specify MINOBS=8, then the _ZERO_ model test is not performed on any series whose length is fewer than eight observations. By default, MINOBS=0, which implies no minimum length exclusion.

If MAXNUM=0 and MAXPCT=100 or MAXPCT=0, then a nonzero model is always considered. You can specify a combination of the MAXNUM= and MAXPCT= options. If you specify ENDZEROS=(MAXNUM=10 MAXPCT=20), then a nonzero model is considered only if the series has 10 or fewer trailing zero values and if the number of trailing zero values is fewer than or equal to 20% of the number of nonmissing and nonzero values of the entire series.

When a series fails to use a nonzero model by the criteria specified in this option, the _ZERO_ model is used. In the OUTEST= data set, the _MODEL_ variable is set to _ZERO_, and the remaining data set variables that are associated with the parameter estimates are set to missing.

HOLDOUT=*n* specifies the size of the holdout sample to be used for model selection. The holdout sample is a subset of actual time series that ends at the last nonmissing observation.

HOLDOUTPCT=*number* specifies the size of the holdout sample as a percentage of the length of the time series. If HOLDOUT=5 and HOLDOUTPCT=10, the size of the holdout sample is $min(5, 0.1T)$, where $T$ is the length of the time series with beginning and ending missing values removed.

INTERMITTENT=*number* specifies a number greater than 1 that is used to determine whether or not a time series is intermittent. If the median demand interval is equal to or greater than this number, then the series is assumed to be intermittent.

MINOBS=(TREND=*n*) specifies that no trend model be fit to any series with fewer than *n* nonmissing values. Normally the models in a selection list are not subset by trend.

Incorporation of a trend is checked only for smoothing, UCM, and ARIMA models. For the smoothing case, only simple smoothing is a non-trend model. For UCM, the absence of a slope component qualifies it as a non-trend model. For ARIMA, there must be no differencing of the dependent variable for PROC HPFENGINE to consider it a non-trend model.

The value of *n* must be greater than or equal to 1. The default is TREND=1.

MINOBS=(SEASON=*n*) specifies that no seasonal model be fit to any series with fewer observations than *n* multiplied by the seasonal cycle length. The value of *n* must be greater than or equal to 1. The default is SEASON=2.

MINOBS=*n* specifies that any series with fewer than *n* nonmissing values not be fit using the models in the selection list, but instead be forecast as the mean of the observations in the series. The forecast is recorded as using the _MEAN_ model in the OUTEST= data set _MODEL_ variable. The value of *n* must be greater than or equal to 1. The default is MINOBS=1.

NOALTLIST disables the default action and sets the forecast to missing if no models can be fit from the initial selection list. (By default, if none of the models in a selection list can be successfully fit to a series, PROC HPFENGINE returns to the selection list Sashelp.Hpfdflt.Best and restarts the selection process.) There will be an observation in OUTSUM=, if requested, that corresponds to the variable and BY group in question, and the _STATUS_ variable will be nonzero.

OVERRIDE forces the use of any options listed.

SEASONTEST=*option* specifies the options related to the seasonality test.

The following values for the SEASONTEST= option are allowed:

NONE no test

(SIGLEVEL=*number*) specifies the significance probability value to use in testing whether seasonality is present in the time series. The value must be between 0 and 1. A smaller value of the SIGLEVEL= option means that stronger evidence of a seasonal pattern in the data is required before PROC HPFENGINE uses seasonal models to forecast the time series.

The default is SEASONTEST=(SIGLEVEL=0.01).

## ADJUST Statement

**ADJUST** *variable = ( variable-list ) / options* **;**

The ADJUST statement lists the numeric variables in the DATA= data set whose accumulated values are used to adjust the dependent values. Adjustments can be performed before or after forecasting (or both). A particular forecast variable can be referenced by multiple ADJUST statements.

The first variable specified is the variable to be adjusted. This variable must appear in a FORECAST statement. The numeric variables used as the source of the adjustments are listed following the parentheses. Options determine which adjustments are applied and when they are applied. More information about the use of adjustments is in the section "Details: HPFENGINE Procedure" on page 211.

The following options can be used with the ADJUST statement:

**OPERATION=(***preadjust***,** *postadjust* **)**

specifies how the adjustments are applied to the forecast variable. The *preadjust* option determines how the adjustment variables are applied to the dependent variable prior to forecasting. The *postadjust* option determines how the adjustment variables are applied to the forecast results.

Computations with missing values are handled differently in the adjustment statement than in other parts of SAS. If any of the adjustment operations result in a nonmissing dependent value being added to, subtracted from, divided by, or multiplied by a missing value, the nonmissing dependent value is left unchanged. Division by zero produces a missing value.

The following predefined adjustment operations are provided:

NONE No adjustment operation is performed. This is the default.

ADD Variables listed in the adjustment statement are added to the dependent variable.

SUBTRACT Variables listed in the adjustment statement are subtracted from the dependent variable.

MULTIPLY Dependent variable is multiplied by variables listed in the adjustment statement.

DIVIDE Dependent variable is divided by variables listed in the adjustment statement.

MIN Dependent variable is set to the minimum of the dependent variable and all variables listed in the adjustment statement.

MAX — Dependent variable is set to the maximum of the dependent variable and all variables listed in the adjustment statement.

It is important to note that pre-adjustment operations are applied in the order defined by the sequence of ADJUST statements but post-adjustment operations are applied in the reverse order. For example, suppose you specify the following sequence of ADJUST statements:

```
ADJUST AIR=(BIAS)  / OPERATION = (SUBTRACT,ADD);
ADJUST AIR=(SCALE) / OPERATION = (DIVIDE,MULTIPLY)
```

When PROC HPFENGINE performs its pre-adjustment, the value of BIAS is subtracted from the forecast variable AIR and that result is then divided by the value of SCALE to produce the pre-adjusted AIR series used by HPFENGINE to generate a forecast. Then that forecast series is post-adjusted using the specified *postadjust* operators in the reverse order of the ADJUST statements: the forecast series is multiplied by the value of SCALE and that result is added to the value of BIAS to produce the final forecast series output from HPFENGINE.

**ACCUMULATE=***option*

specifies how the data set observations are accumulated within each time period for the variables listed in the ADJUST statement. If the ACCUMULATE= option is not specified in the CONTROL statement, accumulation is determined by the ACCUMULATE= option of the ID statement. See the ID statement ACCUMULATE= option for more details.

**SETMISSING=***option* | *number*

specifies how missing values (either actual or accumulated) are assigned in the accumulated time series for variables listed in the ADJUST statement. If the SETMISSING= option is not specified in the ADJUST statement, missing values are set based on the SETMISSING= option of the ID statement. See the ID statement SETMISSING= option for more details.

**TRIMMISS=***option*

specifies how missing values (either actual or accumulated) are trimmed from the accumulated time series for variables listed in the ADJUST statement.

If the TRIMMISS= option is not specified in the ADJUST statement, missing values are set based on the TRIMMISS= option of the ID statement. See the ID statement TRIMMISS= option for more details.

**ZEROMISS=***option*

specifies how beginning and ending zero values (either actual or accumulated) are interpreted in the accumulated time series for variables listed in the ADJUST statement. If the ZEROMISS= option is not specified in the ADJUST statement, missing values are set based on the ZEROMISS= option of the ID statement. See the ID statement ZEROMISS= option for more details.

## BY Statement

**BY** *variables* **;**

A BY statement can be used with PROC HPFENGINE to obtain separate dummy variable definitions for groups of observations defined by the BY variables.

When a BY statement appears, the procedure expects the input data set to be sorted in order of the BY variables.

If your input data set is not sorted in ascending order, use one of the following alternatives:

- Sort the data by using the SORT procedure with a similar BY statement.
- Specify the option NOTSORTED or DESCENDING in the BY statement for the HPF procedure. The NOTSORTED option does not mean that the data are unsorted but rather that the data are arranged in groups (according to values of the BY variables) and that these groups are not necessarily in alphabetical or increasing numeric order.
- Create an index on the BY variables by using the DATASETS procedure.

For more information about the BY statement, see *SAS Language Reference: Concepts*. For more information about the DATASETS procedure, see the discussion in the *Base SAS Procedures Guide.*

## CONTROL Statement

**CONTROL** *variable-list / options* **;**

The CONTROL statement lists the numeric variables in the DATA= data set whose accumulated values are used as input in the forecasting process. The future values of the control variables in the forecast statement are determined by the EXTEND= option. Only input values used in a CONTROL statement are adjustable in the score evaluation subroutine HPFSCSUB.

The following options can be used with the CONTROL statement:

**ACCUMULATE=***option*

specifies how the data set observations are accumulated within each time period for the variables listed in the CONTROL statement. If the ACCUMULATE= option is not specified in the CONTROL statement, accumulation is determined by the ACCUMULATE= option of the ID statement. See the ID statement ACCUMULATE= option for more details.

**EXTEND=***option*

specifies how future values of the control variables are set. The following options are provided:

| | |
|---|---|
| NONE | Future values are set to missing. |
| AVERAGE | Future values are set to the mean of the values in the fit range. This is the default. |
| FIRST | Future values are set to the first value found in the fit range. |
| LAST | Future values are set to the last value found in the fit range. |
| MINIMUM | Future values are set to the minimum of the values in the fit range. |
| MAXIMUM | Future values are set to the maximum of the values in the fit range. |
| MEDIAN | Future values are set to the median of the values in the fit range. |
| STOCHASTIC | Future values are forecast using the best suited exponential smoothing model. See STOCHASTIC statement for details. |

**REPLACEMISSING**
specifies that embedded missing actual values over the fit range be replaced with values obtained by applying the method specified in the EXTEND= option. For EXTEND=STOCHASTIC, the replacement values are the one-step-ahead forecasts obtained over the fit range.

**SETMISSING=***option* | *number*
specifies how missing values (either actual or accumulated) are assigned in the accumulated time series for variables listed in the CONTROL statement. If the SETMISSING= option is not specified in the CONTROL statement, missing values are set based on the SETMISSING= option of the ID statement. See the ID statement SETMISSING= option for more details.

**TRIMMISS=***option*
specifies how missing values (either actual or accumulated) are trimmed from the accumulated time series for variables listed in the CONTROL statement.

If the TRIMMISS= option is not specified in the CONTROL statement, missing values are set based on the TRIMMISS= option of the ID statement. See the ID statement TRIMMISS= option for more details.

**ZEROMISS=***option*
specifies how beginning and/or ending zero values (either actual or accumulated) are interpreted in the accumulated time series for variables listed in the CONTROL statement. If the ZEROMISS= option is not specified in the CONTROL statement, missing values are set based on the ZEROMISS= option of the ID statement. See the ID statement ZEROMISS= option for more details.

## EXTERNAL Statement

**EXTERNAL** *variable-list / options* **;**

The EXTERNAL statement lists the numeric variables in the DATA= data set whose accumulated values are used as predicted values for an external model. It can also list prediction standard errors and lower and upper confidence intervals.

Any variables used in an EXMMAP in a selection list, as specified by PROC HPFSELECT, must appear in an EXTERNAL statement.

The following options can be used with the EXTERNAL statement:

**SETMISSING=***option* | *number*
specifies how missing values (either actual or accumulated) are assigned in the accumulated time series for variables listed in the EXTERNAL statement. If the SETMISSING= option is not specified in the EXTERNAL statement, missing values are set based on the SETMISSING= option of the ID statement. See the ID statement SETMISSING= option for more details.

**TRIMMISS=***option*
specifies how missing values (either actual or accumulated) are trimmed from the accumulated time series for variables listed in the EXTERNAL statement.

If the TRIMMISS= option is not specified in the EXTERNAL statement, missing values are set based on the TRIMMISS= option of the ID statement. See the ID statement TRIMMISS= option for more details.

**ZEROMISS=***option*

specifies how beginning and/or ending zero values (either actual or accumulated) are interpreted in the accumulated time series for variables listed in the ADJUST statement. If the ZEROMISS= option is not specified in the EXTERNAL statement, missing values are set based on the ZEROMISS= option of the ID statement. See the ID statement ZEROMISS= option for more details.

## FORECAST Statement

**FORECAST** *variable-list / options* **;**

The FORECAST statement lists the numeric variables in the DATA= data set whose accumulated values represent time series to be modeled and forecast.

A data set variable can be specified in only one FORECAST statement. Any number of FORECAST statements can be used.

The following options can be used with the FORECAST statement.

**ACCUMULATE=***option*

specifies how the data set observations are accumulated within each time period for the variables listed in the FORECAST statement. If the ACCUMULATE= option is not specified in the FORECAST statement, accumulation is determined by the ACCUMULATE= option of the ID statement. See the ID statement ACCUMULATE= option for more details.

**NOREPLACE**

specifies that the forecast value in the OUTFOR= data set be set to missing when a corresponding historical value is missing. For example, consider historical data spanning 01JAN1980 through 01DEC1985. If the dependent variable that corresponds to 01MAR1983 is missing, the forecast variable in the OUTFOR= data set at time ID 01MAR1983 is usually replaced by the one-step-ahead forecast. The NOREPLACE option prevents that one-step-ahead forecast from being written in the OUTFOR= data set. The inverse behavior of this option, applied to the OUT= data set, is REPLACEMISSING.

**REPLACEMISSING**

specifies that embedded missing actual values be replaced with one-step-ahead forecasts in the OUT= data set. The inverse behavior of this option, applied to the OUTFOR= data set, is NOREPLACE.

**SETMISSING=***option* | *number*

specifies how missing values (either actual or accumulated) are assigned in the accumulated time series for variables listed in the FORECAST statement. If the SETMISSING= option is not specified in the FORECAST statement, missing values are set based on the SETMISSING= option of the ID statement. See the ID statement SETMISSING= option for more details.

**TRIMMISS=***option*

specifies how missing values (either actual or accumulated) are trimmed from the accumulated time series for variables listed in the FORECAST statement. If the TRIMMISS= option is not specified in the FORECAST statement, missing values are set based on the TRIMMISS= option of the ID statement. See the ID statement TRIMMISS= option for more details.

**ZEROMISS=***option*

specifies how beginning and/or ending zero values (either actual or accumulated) are interpreted in the accumulated time series for variables listed in the FORECAST statement. If the ZEROMISS= option is not specified in the FORECAST statement, missing values are set based on the ZEROMISS= option of the ID statement. See the ID statement ZEROMISS= option for more details.

## ID Statement

**ID** *variable* < *options* > **;**

The ID statement names a numeric variable that identifies observations in the input and output data sets. The ID variable's values are assumed to be SAS date, time, or datetime values. In addition, the ID statement specifies the (desired) frequency associated with the actual time series. The ID statement options also specify how the observations are accumulated and how the time ID values are aligned to form the actual time series. The information specified affects all variables specified in subsequent FORECAST statements. If the ID statement is specified, the INTERVAL= option must also be specified. If an ID statement is not specified, the observation number (with respect to the BY group) is used as the time ID.

The following options can be used with the ID statement.

**ACCUMULATE=***option*

specifies how the data set observations are accumulated within each time period. The frequency (width of each time interval) is specified by the INTERVAL= option. The ID variable contains the time ID values. Each time ID variable value corresponds to a specific time period. The accumulated values form the actual time series, which is used in subsequent model fitting and forecasting.

The ACCUMULATE= option is particularly useful when there are zero or more than one input observations that coincide with a particular time period (for example, transactional data). The EXPAND procedure offers additional frequency conversions and transformations that can also be useful in creating a time series.

The following options determine how the observations are accumulated within each time period based on the ID variable and the frequency specified by the INTERVAL= option:

| | |
|---|---|
| NONE | No accumulation occurs; the ID variable values must be equally spaced with respect to the frequency. This is the default option. |
| TOTAL | Observations are accumulated based on the total sum of their values. |
| AVERAGE \| AVG | Observations are accumulated based on the average of their values. |
| MINIMUM \| MIN | Observations are accumulated based on the minimum of their values. |
| MEDIAN \| MED | Observations are accumulated based on the median of their values. |
| MAXIMUM \| MAX | Observations are accumulated based on the maximum of their values. |
| N | Observations are accumulated based on the number of nonmissing observations. |
| NMISS | Observations are accumulated based on the number of missing observations. |
| NOBS | Observations are accumulated based on the number of observations. |

| | |
|---|---|
| FIRST | Observations are accumulated based on the first of their values. |
| LAST | Observations are accumulated based on the last of their values. |
| STDDEV \| STD | Observations are accumulated based on the standard deviation of their values. |
| CSS | Observations are accumulated based on the corrected sum of squares of their values. |
| USS | Observations are accumulated based on the uncorrected sum of squares of their values. |

If the ACCUMULATE= option is specified, the SETMISSING= option is useful for specifying how accumulated missing values are treated. If missing values should be interpreted as zero, then SETMISSING=0 should be used. The section "Details: HPFENGINE Procedure" on page 211 describes accumulation in greater detail.

**ALIGN=***option*

controls the alignment of SAS dates that are used to identify output observations. The ALIGN= option accepts the following values: BEGINNING | BEG | B, MIDDLE | MID | M, and ENDING | END | E. The default is BEGINNING.

**END=***option*

specifies a SAS date, datetime, or time value that represents the end of the data. If the last time ID variable value is less than the END= value, the series is extended with missing values. If the last time ID variable value is greater than the END= value, the series is truncated. For example, END="&sysdate"D uses the automatic macro variable SYSDATE to extend or truncate the series to the current date.

**FORMAT=***option*

specifies a SAS format that is used for the DATE variable in the output data sets. The default format is the same as that of the DATE variable in the DATA= data set.

**HORIZONSTART=***option*

specifies a SAS date, datetime, or time value that represents the start of the forecast horizon. If the specified HORIZONSTART= date falls beyond the end of the historical data, then forecasts are computed from the last observation with a nonmissing dependent variable value until LEAD= intervals from the HORIZONSTART= data. Therefore, the effective forecast horizon for any particular BY group might differ from another due to differences in the lengths of the historical data across the BY groups, but all forecasts will end at the same date as determined by the HORIZONSTART= and LEAD= options.

An important feature of the HORIZONSTART= option is that it truncates values only in forecast variables. Any future values of input variables are retained.

**INTERVAL=***interval*

specifies the frequency of the input time series. For example, if the input data set consists of quarterly observations, then INTERVAL=QTR should be used. If the SEASONALITY= option is not specified, the length of the seasonal cycle is implied from the INTERVAL= option. For example, INTERVAL=QTR implies a seasonal cycle of length 4. If the ACCUMULATE= option is also specified, the INTERVAL= option determines the time periods for the accumulation of observations. See the *SAS/ETS User's Guide* for the intervals that can be specified.

**SETMISSING=***option* | *number*

specifies how missing values (either actual or accumulated) are assigned in the accumulated time series. If a number is specified, missing values are set to that number. If a missing value indicates an unknown value, this option should not be used. If a missing value indicates no value, a SETMISSING=0 should be used. You would typically use SETMISSING=0 for transactional data because the absence of recorded data usually implies no activity. The following options can also be used to determine how missing values are assigned:

| | |
|---|---|
| MISSING | Missing values are set to missing. This is the default option. |
| AVERAGE \| AVG | Missing values are set to the accumulated average value. |
| MINIMUM \| MIN | Missing values are set to the accumulated minimum value. |
| MEDIAN \| MED | Missing values are set to the accumulated median value. |
| MAXIMUM \| MAX | Missing values are set to the accumulated maximum value. |
| FIRST | Missing values are set to the accumulated first nonmissing value. |
| LAST | Missing values are set to the accumulated last nonmissing value. |
| PREVIOUS \| PREV | Missing values are set to the previous accumulated nonmissing value. Missing values at the beginning of the accumulated series remain missing. |
| NEXT | Missing values are set to the next accumulated nonmissing value. Missing values at the end of the accumulated series remain missing. |

If SETMISSING=MISSING is specified and the MODEL= option specifies a smoothing model, the missing observations are smoothed over. If MODEL=IDM is specified, missing values are assumed to be periods of no demand; that is, SETMISSING=MISSING is equivalent to SETMISSING=0.

**START=***option*

specifies a SAS date, datetime, or time value that represents the beginning of the data. If the first time ID variable value is greater than the START= value, the series is prefixed with missing values. If the first time ID variable value is less than the END= value, the series is truncated. This option and the END= option can be used to ensure that data associated with each BY group contain the same number of observations.

**TRIMMISS=***option*

specifies how missing values (either actual or accumulated) are trimmed from the accumulated time series for variables listed in the FORECAST statement. The following options are provided:

| | |
|---|---|
| NONE | No missing value trimming is applied. |
| LEFT | Beginning missing values are trimmed. |
| RIGHT | Ending missing values are trimmed. |
| BOTH | Both beginning and ending missing values are trimmed. This is the default. |

**ZEROMISS=***option*

specifies how beginning and/or ending zero values (either actual or accumulated) are interpreted in the accumulated time series. The following options can also be used to determine how beginning and/or ending zero values are assigned:

| | |
|---|---|
| NONE | Beginning and/or ending zeros are unchanged. This is the default. |
| LEFT | Beginning zeros are set to missing. |
| RIGHT | Ending zeros are set to missing. |
| BOTH | Both beginning and ending zeros are set to missing. |

If the accumulated series is all missing and/or zero, the series is not changed.

## INPUT Statement

**INPUT** *variable-list / options* **;**

The INPUT statement lists the numeric variables in the DATA= data set whose accumulated values are used as deterministic input in the forecasting process.

Future values for input variables must be supplied. If future values are unknown, consider using either the STOCHASTIC statement or the CONTROL statement. If it will be necessary to later modify future values using the forecast score function HPFSCSUB, use the CONTROL statement.

A data set variable can be specified in only one INPUT statement. Any number of INPUT statements can be used.

The following options can be used with the INPUT statement:

**ACCUMULATE=***option*

specifies how the data set observations are accumulated within each time period for the variables listed in the INPUT statement. If the ACCUMULATE= option is not specified in the INPUT statement, accumulation is determined by the ACCUMULATE= option of the ID statement. See the ID statement ACCUMULATE= option for more details.

**REQUIRED=YES | NO**

enables or disables a check of inputs to models. The kinds of problems checked include the following:

- errors in functional transformation
- an input consisting of only a constant value or all missing values
- errors introduced by differencing
- multicollinearity among inputs

If REQUIRED=YES, these checks are not performed and no inputs are dropped from a model. The model might subsequently fail to fit during parameter estimation or forecasting for any of the reasons in the preceding list.

If REQUIRED=NO, inputs are checked and those with errors, or those judged collinear, are dropped from the model for the current series and task only. No changes are kept in the model specification.

This option has no effect on models with no inputs.

The default is REQUIRED=YES.

**SETMISSING=***option* | *number*
specifies how missing values (either actual or accumulated) are assigned in the accumulated time series for the variables listed in the INPUT statement. If the SETMISSING= option is not specified in the INPUT statement, missing values are set based on the SETMISSING= option of the ID statement. See the ID statement SETMISSING= option for more details.

**TRIMMISS=***option*
specifies how missing values (either actual or accumulated) are trimmed from the accumulated time series for variables listed in the INPUT statement.

If the TRIMMISS= option is not specified in the INPUT statement, missing values are set based on the TRIMMISS= option of the ID statement. See the ID statement TRIMMISS= option for more details.

**ZEROMISS=***option*
specifies how beginning and/or ending zero values (either actual or accumulated) are interpreted in the accumulated time series for variables listed in the INPUT statement. If the ZEROMISS= option is not specified in the INPUT statement, missing values are set based on the ZEROMISS= option of the ID statement. See the ID statement ZEROMISS= option for more details.

## SCORE Statement

**SCORE** *options* **;**

The SCORE statement, used in conjunction with one or more FORECAST statements, causes the generation of score files. The score files are written to the catalog specified by the SCOREREPOSITORY= option. One score file is generated for each forecast variable within each BY group. Score file names are constructed by appending a sequence number to a base name to create a unique entry in the score repository. Sequence numbers start from 0. You can determine the name of the score file used for each forecast variable within each BY group by referring to the _SCORE_ column in the OUTEST= data set.

The SCORE statement supports the following option:

**BASENAME=***SAS-name*
prefixes the score file name with the specified value. For example, if you specify BASENAME=SALE, the score files are named SALE0, SALE1, .... The default BASENAME for the SCORE statement is 'SCOR'. If you forecast more than one variable in the PROC HPFENGINE run, score file names all use the same prefix regardless of the dependent variable. Use the OUTEST= data set to determine the score file generated for each variable and BY group combination.

Also note, the BASENAME= value is limited to 22 characters to ensure that the entire range of score file names can be generated if needed. If you specify a value that exceeds this limit, a warning is printed to the SAS log and the value is truncated.

For examples that demonstrate use of the SCORE statement, see Example 7.4.

## STOCHASTIC Statement

**STOCHASTIC** *variable-list / options* **;**

The STOCHASTIC statement lists the numeric variables in the DATA= data set whose accumulated values are used as stochastic input in the forecasting process.

Future values of stochastic inputs do not need to be provided. By default, they are automatically forecast using one of the following smoothing models:

- simple
- double
- linear
- damped trend
- seasonal
- Winters method (additive and multiplicative)

The model with the smallest in-sample MAPE is used to forecast the future values of the stochastic input.

A data set variable can be specified in only one STOCHASTIC statement. Any number of STOCHASTIC statements can be used.

The following options can be used with the STOCHASTIC statement:

**ACCUMULATE=***option*
specifies how the data set observations are accumulated within each time period for the variables listed in the STOCHASTIC statement. If the ACCUMULATE= option is not specified in the STOCHASTIC statement, accumulation is determined by the ACCUMULATE= option of the ID statement. See the ID statement ACCUMULATE= option for more details.

**SELECTION=***option*
specifies the selection list used to forecast the stochastic variables. The default is BEST, found in SASHELP.HPFDFLT.

**REQUIRED=YES | NO**
enables or disables a check of inputs to models. The kinds of problems checked include the following:

- errors in functional transformation
- an input consisting of only a constant value or all missing values
- errors introduced by differencing
- multicollinearity among inputs

If REQUIRED=YES, these checks are not performed and no inputs are dropped from a model. The model might subsequently fail to fit during parameter estimation or forecasting for any of the reasons in the preceding list.

If REQUIRED=NO, inputs are checked and those with errors, or those judged collinear, are dropped from the model for the current series and task only. No changes are kept in the model specification.

This option has no effect on models with no inputs.

The default is REQUIRED=YES.

**REPLACEMISSING**
specifies that embedded missing actual values be replaced with one-step-ahead forecasts in the STOCHASTIC variables.

**SETMISSING=***option* | *number*
specifies how missing values (either actual or accumulated) are assigned in the accumulated time series for variables listed in the STOCHASTIC statement. If the SETMISSING= option is not specified in the STOCHASTIC statement, missing values are set based on the SETMISSING= option of the ID statement. See the ID statement SETMISSING= option for more details.

**TRIMMISS=***option*
specifies how missing values (either actual or accumulated) are trimmed from the accumulated time series for variables listed in the STOCHASTIC statement.

If the TRIMMISS= option is not specified in the STOCHASTIC statement, missing values are set based on the TRIMMISS= option of the ID statement. See the ID statement TRIMMISS= option for more details.

**ZEROMISS=***option*
specifies how beginning and/or ending zero values (either actual or accumulated) are interpreted in the accumulated time series for variables listed in the FORECAST statement. If the ZEROMISS= option is not specified in the STOCHASTIC statement, missing values are set based on the ZEROMISS= option of the ID statement. See the ID statement ZEROMISS= option for more details.

# Details: HPFENGINE Procedure

The HPFENGINE procedure can be used to forecast time series data as well as transactional data. If the data are transactional, then the procedure must first accumulate the data into a time series before the data can be forecast. The procedure uses the following sequential steps to produce forecasts:

1. accumulation
2. missing value interpretation
3. pre-forecast adjustment
4. diagnostic tests
5. model selection
6. transformations
7. parameter estimation

8. forecasting
9. inverse transformation
10. post-forecast adjustment
11. statistics of fit

These steps are described in the following sections.

## Accumulation

If the ACCUMULATE= option is specified, data set observations are accumulated within each time period. The frequency (width of each time interval) is specified by the INTERVAL= option. The ID variable contains the time ID values. Each time ID value corresponds to a specific time period. Accumulation is particularly useful when the input data set contains transactional data, whose observations are not spaced with respect to any particular time interval. The accumulated values form the actual time series, which is used in subsequent analyses.

For example, suppose a data set contains the following observations:

```
19MAR1999    10
19MAR1999    30
11MAY1999    50
12MAY1999    20
23MAY1999    20
```

If the INTERVAL=MONTH is specified, all of the preceding observations fall within the three time periods of March 1999, April 1999, and May 1999. The observations are accumulated within each time period as follows:

If the ACCUMULATE=NONE option is specified, an error is generated because the ID variable values are not equally spaced with respect to the specified frequency (MONTH).

If the ACCUMULATE=TOTAL option is specified:

```
O1MAR1999    40
O1APR1999    .
O1MAY1999    90
```

If the ACCUMULATE=AVERAGE option is specified:

```
O1MAR1999    20
O1APR1999    .
O1MAY1999    30
```

If the ACCUMULATE=MINIMUM option is specified:

```
O1MAR1999    10
O1APR1999    .
O1MAY1999    20
```

If the ACCUMULATE=MEDIAN option is specified:

```
O1MAR1999    20
01APR1999    .
O1MAY1999    20
```

If the ACCUMULATE=MAXIMUM option is specified:

```
O1MAR1999    30
O1APR1999    .
O1MAY1999    50
```

If the ACCUMULATE=FIRST option is specified:

```
O1MAR1999    10
O1APR1999    .
O1MAY1999    50
```

If the ACCUMULATE=LAST option is specified:

```
O1MAR1999    30
O1APR1999    .
O1MAY1999    20
```

If the ACCUMULATE=STDDEV option is specified:

```
O1MAR1999    14.14
O1APR1999    .
O1MAY1999    17.32
```

As can be seen from the preceding examples, even though the data set observations contained no missing values, the accumulated time series might have missing values.

## Missing Value Interpretation

Sometimes missing values should be interpreted as unknown values. The forecasting models used by the HPFENGINE procedure can handle missing values effectively. But sometimes missing values are known (such as when missing values are created from accumulation), and no observations should be interpreted as no (zero) value. In the former case, the SETMISSING= option can be used to interpret how missing values are treated. The SETMISSING=0 option should be used when missing observations are to be treated as no (zero) values. In other cases, missing values should be interpreted as global values, such as minimum or maximum values of the accumulated series. The accumulated and interpreted time series is used in subsequent analyses.

## Adjustment Operations

Pre-adjustment variables can be used to adjust the dependent series prior to model parameter estimation, evaluation, and forecasting. After the predictions of the adjusted dependent series are obtained from the forecasting mode, the post-adjustment variables can be used to adjust these forecasts to obtain predictions that more closely match the original dependent series.

### Pre-adjustment (Before Forecasting) Step

If $y_t$ is the dependent series and $x_{i,t}$ for $i = 1, \ldots, M$ are the $M$ adjustment series, the adjusted dependent series $w_t$ is as follows:

$$\begin{aligned} w_{i,t} &= op_i^b(y_t, x_{i,t-k_i}) \text{ for } i = 1 \\ w_{i,t} &= op_i^b(w_{i-1,t}, x_{i,t-k_i}) \text{ for } 1 < i \leq M \\ w_t &= w_{i,t} \text{ for } i = M \end{aligned}$$

where $op_i^b$ represents the pre-adjustment operator and $k_i$ is the time shift for the adjustment series $x_{i,t}$.

As can be seen, the pre-adjustment operators are nested and applied sequentially from $i = 1, \ldots, M$.

Pre-adjustment is performed on the historical data only.

### Adjusted Forecast Step

$$\begin{aligned} w_t &= \hat{F}(w_t) + \varepsilon_t \text{ historical data} \\ \hat{w}_t &= \hat{F}(w_t) \text{ historical data and forecast horizon} \end{aligned}$$

where $\hat{F}()$ represents the fitted forecasting function.

### Post-adjustment (After Forecasting) Step

$$\begin{aligned} \hat{w}_{i,t} &= op_i^a(\hat{w}_t, x_{i,t-k_i}) \text{ for } i = M \\ \hat{w}_{i,t} &= op_i^a(\hat{w}_{i+1,t}, x_{i,t-k_i}) \text{ for } 1 \leq i < M \\ \hat{y}_t &= \hat{w}_{i,t} \end{aligned}$$

where $op_i^a$ represents the post-adjustment operator for the adjustment series $x_{i,t}$.

As can be seen, the post-adjustment operators are nested and applied sequentially from $i = M, \ldots, 1$, which is the reverse of the pre-adjustment step.

Post-adjustment is performed on the historical data as well as the forecast horizon.

### Notes

Typically the pre-adjustment operator $op_i^b$ and post-adjustment operator $op_i^a$ are inverses of each other; that is, $op_i^a = inverse(op_i^b)$.

For example, if the pre-adjustment operator is subtraction, the post-adjustment operator is addition, as shown in the following:

$$\begin{aligned} w_t &= y_t - \sum_{i=1}^{M} x_{i,t} \\ \hat{y}_t &= \hat{w}_t + \sum_{i=1}^{M} x_{i,t} \end{aligned}$$

For example, if the pre-adjustment operator is division, the post-adjustment operator is multiplication, as shown in the following:

$$\begin{aligned} w_t &= y_t \prod_{i=1}^{M} \frac{1}{x_{i,t}} \\ \hat{y}_t &= \hat{w}_t \prod_{i=1}^{M} x_{i,t} \end{aligned}$$

Pre-adjustment is often followed by post-adjustment, but the inverse operation is not required. It is acceptable to pre-adjust, but not post-adjust, and vice versa.

As an example, the following statement adds, before forecasting, the values contained in the variables firstadj and scndadj to the dependent variable air. After forecasting air, there is no post-adjustment. The variable air must be specified in a FORECAST statement.

```
ADJUST air=(firstadj scndadj) / OPERATION = (ADD,NONE);
```

## Diagnostic Tests

Diagnostic test control is specified in the model selection list and can be set by using the HPFSELECT procedure. The INTERMITTENT= option in the HPFSELECT procedure's DIAGNOSE statement sets the threshold for categorizing a series as intermittent or nonintermittent. Likewise, the SEASONTEST= option in the HPFSELECT procedure's DIAGNOSE statement sets the threshold for categorizing a series as seasonal or nonseasonal.

These diagnostic categorizations are used during the model selection process to ensure that only appropriate models are fit to a given series.

## Model Selection

PROC HPFENGINE applies a model selection list to each series it processes. When more than one candidate model is specified in a model selection list, forecasts for each candidate model are compared by using the model selection criterion specified via the CRITERION= option in the SELECT statement in the HPFSELECT procedure.

The selection criterion is computed using the multistep forecasts in the holdout sample range if the HOLDOUT= or HOLDOUTPCT= option is specified, or the one-step-ahead forecasts for the full range of the time series if the HOLDOUT= and HOLDOUTPCT= option is not specified.

The candidate model with the best selection criterion is selected to forecast the time series.

Not all model specifications in a model selection list are necessarily used as candidates during the selection process. Diagnostic tests might tag some model specifications as unsuitable. Generally, if a time series is judged as seasonal by the diagnostic tests, only seasonal models in the model selection list are candidates. Likewise, if a time series is nonseasonal, only nonseasonal models are fit. If the seasonal characteristics of a time series do not match those of any models in a selection list, the seasonal diagnostic test is turned off and the selection process is restarted.

Characteristics that make a model seasonal vary according to the family of model. Those characteristics are listed here by model family:

| | |
|---|---|
| Smoothing models | The Winters, additive Winters, and seasonal smoothing models are considered seasonal. Linear, simple, double, and damped trend models smoothing are considered nonseasonal. |
| ARIMA models | An ARIMA model is considered seasonal if there is a difference of order equal to the seasonal cycle length of the time ID. The presence of MA or AR terms with lags equal to the seasonal cycle length also qualifies a model as seasonal. The presence of a predefined seasonal input is another factor that tags a model as seasonal. |
| UCM | An unobserved component model is seasonal if a seasonal component is present, specified by the SEASON statement in the HPFUCMSPEC procedure, or if there is a predefined seasonal input. |

Similar subsetting of selection lists occurs after an intermittency diagnostic test. The HPFENGINE procedure avoids direct comparison of results from intermittent models and nonintermittent models. If the diagnostic tests deem a time series intermittent, only intermittent models are used as candidates during the selection process. If a series is nonintermittent, only nonintermittent series are used.

The toplogy of the model selection process can be more complex than this brief overview describes. Combined models and mixtures of lists with time series models in the overall selection process can be evaluated by the HPFENGINE procedure. This enables more complex model selection decision toplogies. Details of the full model selection process are described in Chapter 21, "Forecast Model Selection Graph Details." See Chapter 20, "Forecast Combination Computational Details," for mathematical details related to combined models. See also Chapter 15, "The HPFSELECT Procedure," for information related to defining these more complex model selection topologies.

## IDM Model Combinations

Intermittent demand model (IDM) forecasts present special problems when considering the combination of those IDM forecasts with forecasts from other non-IDM models, in addition to combinations of forecasts that arise from different IDM models. IDM models work in a transform domain of demand-indexed observations. The demand domain represents observations in the series that have values different from the IDM model's base value. Non-base demands are collapsed into contiguous observations for purposes of modeling the demand series. See Chapter 19, "Forecasting Process Details," for more information about IDM models.

For a given series, the model selection process in PROC HPFENGINE never mixes models from the IDM and non-IDM families. If IDM models are present in the list, those are run first. If one or more of them are successful in classifying the series as intermittent, then any non-IDM models in the list are automatically removed from consideration in the remainder of the selection list processing.

A combined model list that contains IDM models follows this same scheme. If any IDM model in the combination list is successful, the list is restricted to combinations of IDM model forecasts that use the same IDM base value. The first successful IDM model in the combination list defines the reference IDM base for the list. Any subsequent IDM model in the combined model list that uses a different base value is excluded from the combination.

A combination of base-compatible IDM forecasts is further restricted in terms of elements of the COMBINE statement functionality. These restrictions are related to operations that depend on the use of time-based residuals in some way. They affect the following COMBINE statement options:

- Combination weight methods are restricted to METHOD=AVG and METHOD=USERDEF.
- Encompassing tests are not permitted.
- Standard error modes for prediction errors are not permitted.

When a combined model list that uses IDM forecasts is processed in PROC HPFENGINE, checks are performed to ensure that the settings for the WEIGHT= and ENCOMPASS= options are consistent with the listed restrictions. If not, messages are issued to the SAS log to indicate that the respective option settings are being ignored and changed for the particular series instance and combined model list pairing.

## Transformations

If a forecasting model specifies a transformation of the dependent series, the time series is transformed prior to model parameter estimation and forecasting. Only strictly positive series can be transformed.

## Parameter Estimation

All parameters associated with the model are optimized based on the data with the default parameter restrictions imposed. The starting point for parameter estimation, either automatically chosen by the optimizer or supplied programmatically, can be controlled using the TASK= option.

If a forecasting model specifies a transformation of the dependent series, the transformed time series data are used to estimate the model parameters.

## Missing Value Modeling Issues

The treatment of missing values varies with the forecasting model. For the intermittent demand models, specified missing values are assumed to be periods of no demand. For other models, missing values after the start of the series are replaced with one-step-ahead predicted values, and the predicted values are applied to the smoothing equations. See the section "Forecasting" on page 218 for more information about how missing values are treated in the smoothing models.

You can also specify the treatment of missing values with the SETMISSING= option, which changes the missing values prior to modeling. The ZEROMISS= option replaces zero values with missing values at either the beginning or end of a series. In such cases the SETMISSING= option is applied after the ZEROMISS= option.

Even though all of the observed data are nonmissing, using the ACCUMULATE= option can create missing values in the accumulated series.

## Forecasting

After the model parameters are estimated, one-step-ahead forecasts are generated for the full range of the actual (optionally transformed) time series data, and multistep forecasts are generated from the end of the observed time series to the future time period after the last observation specified by the LEAD= option. If there are missing values at the end of the time series, the forecast horizon will be greater than that specified by the LEAD= option. Options such as HORIZONSTART= might implicitly add missing values to the end of a series and also cause forecast horizons greater than that specified by LEAD=.

## Inverse Transformations

If a forecasting model specifies a transformation of the dependent series, the forecasts of the transformed time series are inverse transformed. By default, the mean (expected) forecasts are generated. If the MEDIAN option is specified the model specification procedures, the median forecasts are generated.

## Statistics of Fit

The statistics of fit (or goodness-of-fit statistics) are computed by comparing the actual time series data and the generated forecasts. If the dependent series was transformed according to the model specification, the statistics of fit are based on the inverse transformed forecasts.

## Thread Usage in PROC HPFENGINE

The HPFENGINE procedure can make use of threads to speed up different aspects of model selection and forecasting. There are no special options beyond SAS system options to control thread usage in PROC HPFENGINE. You can use the OPTIONS statement to specify the following SAS options:

THREADS | NOTHREADS controls whether threading is allowed in PROC HPFENGINE.

CPUCOUNT= controls the maximum number of threads to be used PROC HPFENGINE.

When threading is allowed, the thread budget for the PROC HPFENGINE step is determined by the value of the CPUCOUNT= option. Threaded execution is considered only when the CPUCOUNT= value is greater than 1.

PROC HPFENGINE can thread certain phases of its computations within each BY group that it processes. When threading is allowed, PROC HPFENGINE can make use of threads in the following computational processes:

- FORECAST statement variables (also called FORECAST variable)
- model execution

At a coarse level, PROC HPFENGINE can use threads to process different FORECAST statement variables concurrently. If you specify more than one FORECAST variable in a single PROC HPFENGINE step and threading is allowed, then those variables are processed concurrently for the TASK= option that is specifed. When this coarse threading can be applied, it provides a maximum benefit in terms of reducing elapsed time with respect to nonthreaded execution for the additional CPU resources expended, and it produces gains without regard to the model selection list complexity or the TASK= mode that is specified for the step.

Within the scope of each FORECAST variable, PROC HPFENGINE can use threads as part of its model selection process. All of the time series models to be considered in the forecast model selection graph are submitted for concurrent execution. Subject to the availabilty of the CPUCPOUNT= resources after accounting for coarse-grained parallelism at the FORECAST variable level, the available threads execute time series models concurrently. When the models complete, the decision process as defined by the forecast model selection graph is evaluated to select the best performing model (forecast). If the selected model (forecast) uses a model combination, concurrent model execution can also occur during the generation of the final foreast. This obviously has the greatest opportunity for improvement when TASK=SELECT is performed for a model selection graph that has several time series models to be executed. Invocations that specify a TASK=FIT, UPDATE, or FORECAST option have much less opportunity for the use of threads at the FORECAST variable scope because the set of time series models to be executed is frequently only the one that was selected from a previous TASK=SELECT run.

The thread budget that is allowed for the PROC HPFENGINE step is dynamically shared between coarse threading of the FORECAST variables and threaded execution of the time series models.

## Data Set Input/Output

The following input data sets supply series data, selection list mapping information and/or parameter estimates, and event data base information, respectively:

- AUXDATA=
- DATA=

- INEST=
- INEVENT=

Additionally, the HPFENGINE procedure can create the following data sets:

- OUT=
- OUTACCDATA=
- OUTCOMPONENT=
- OUTEST=
- OUTFOR=
- OUTINDEP=
- OUTMODELINFO=
- OUTPROCINFO=
- OUTSTAT=
- OUTSTATSELECT=
- OUTSUM=

In general, if the forecasting process for a particular time series fails, the output that corresponds to this series is not recorded or is set to missing in the relevant output data set, and appropriate error and/or warning messages are recorded in the log.

### AUXDATA= Data Set

PROC HPFENGINE supports the AUXDATA= option to supply variables that are used in a forecast run but are not physically part of the primary data set supplied via the DATA= option. For example, you could use AUXDATA to share a data set with explanatory variables across multiple projects, or to separate out explanatory variables that are redundant below some level in a BY group hierarchy or that might not need BY-variable qualification at all. Unlike the DATA= option, you can specify the AUXDATA= option multiple times in the PROC statement to supply more than one auxiliary data set for PROC HPFENGINE to use during its run. For more information and examples, see Chapter 25, "Using Auxiliary Data Sets."

### DATA= Data Set

The DATA= data set supplies the input data for the procedure to forecast. It optionally contains any time ID variable, adjustment variables, explanatory variables, and BY group variables needed for PROC HPFENGINE to run successfully. If the DATA= option is not specified, the most recently created SAS data set is used. See also AUXDATA= for related information.

### INEST= Data Set

The INEST= data set supplies mapping information that associates individual time series with model selection lists. It can optionally include information about model parameter estimates. Parameter estimates might be required depending on the setting of the TASK= option. This data set is typically created by either the HPFDIAGNOSE procedure or a prior invocation of the HPFENGINE procedure with the OUTEST= option.

The INEST= data set is not required. If not present, a model selection list can be supplied for use by all dependent series by using the GLOBALSELECTION= option. In the absence of both the INEST= data set and the GLOBALSELECTION= option, a default selection list is used.

If the INEST= data set is supplied but there are some forecast variables that have no mapping in INEST=, those unmatched variables are forecast using model specifications found either by using the GLOBALSELECTION= option or in the default list.

The INEST= data set contains the variables specified in the BY statement as well as the following variables:

| | |
|---|---|
| _NAME_ | variable name |
| _SELECT_ | name of selection list |
| _MODEL_ | name of model |
| _MODELVAR_ | model variable mapping |
| _DSVAR_ | data set variable mapping |
| _VARTYPE_ | role of variable: DEPENDENT, INPUT, or EVENT |

The referenced selection lists, taken together with the data set to model variable mappings, drive the forecasting process.

If the HPFENGINE statement TASK= option is specified and set to either FORECAST or UPDATE, other variables should be present in INEST=. The additional variables are as follows:

| | |
|---|---|
| _TRANSFORM_ | transformation applied |
| _COMPONENT_ | model component (for example, AR, MA, trend, and so on) |
| _COMPMODEL_ | model portion of an intermittent demand component |
| _FACTOR_ | model factor |
| _LAG_ | lag of input |
| _SHIFT_ | shift |
| _PARM_ | parameter name |
| _LABEL_ | parameter label |
| _EST_ | parameter estimate |
| _STDERR_ | parameter estimate standard error |
| _TVALUE_ | parameter estimate *t* value |
| _PVALUE_ | parameter estimate *p*-value |
| _STATUS_ | indicates success/failure in estimating parameter. See "_STATUS_ Values" on page 229 for details. |

### INEVENT= Data Set

The INEVENT= contains information that describes predefined events. This data set is usually created by the HPFEVENTS procedure and is required only if events are included in a model. Details are found in the Chapter 9, "The HPFEVENTS Procedure,".

### OUT= Data Set

The OUT= data set contains the variables specified in the BY, ID, and FORECAST statements. If the ID statement is specified, the ID variables are aligned and extended based on the ALIGN= and INTERVAL= options. The values of the variables specified in the FORECAST statements are accumulated based on the ACCUMULATE= option, and missing values are interpreted based on the SETMISSING= option. If the REPLACEMISSING option is specified, embedded missing values are replaced by the one-step-ahead forecasts. If any of the forecasting steps fail for a particular variable, the variable values are extended by missing values.

### OUTACCDATA= Data Set

The OUTACCDATA contains one row for each forecast variable in the HPFENGINE run. It contains the following variables:

| | |
|---|---|
| _NAME_ | variable name |
| _ACCUMULATE_ | accumulation mode used for the variable |

### OUTCOMPONENT= Data Set

The contents of the OUTCOMPONENT set vary depending upon the forecast model. See Chapter 19, "Forecasting Process Details," for information about specific model components.

The OUTCOMPONENT= data set contains the variables specified in the BY statement as well as the following variables:

| | |
|---|---|
| _NAME_ | variable name |
| _COMP_ | name of the component |
| _TIME_ | time ID |
| _ACTUAL_ | dependent series value |
| _PREDICT_ | component forecast |
| _LOWER_ | lower confidence limit |
| _UPPER_ | upper confidence limit |
| _STD_ | prediction standard error |

### OUTEST= Data Set

The OUTEST= data set contains the variables specified in the BY statement as well as the following variables:

| | |
|---|---|
| _NAME_ | variable name |
| _SELECT_ | name of selection list |
| _MODEL_ | name of model |
| _MODELVAR_ | model variable mapping |
| _DSVAR_ | data set variable mapping |
| _TRANSFORM_ | transformation applied |
| _COMPONENT_ | model component (for example, AR, MA, trend, and so on) |
| _COMPMODEL_ | model portion of an intermittent demand component |
| _FACTOR_ | model factor |
| _LAG_ | lag of input |
| _SHIFT_ | shift |
| _PARM_ | parameter name |
| _LABEL_ | parameter label |
| _EST_ | parameter estimate |
| _STDERR_ | parameter estimate standard error |
| _TVALUE_ | parameter estimate *t* value |
| _PVALUE_ | parameter estimate *p*-value |
| _STATUS_ | indicates success/failure in estimating parameter. See "_STATUS_ Values" on page 229 for details. |

An OUTEST= data set is frequently used as the INEST= data set for subsequent invocations of PROC HPFENGINE. In such a case, if the option TASK=FORECAST is used, forecasts are generated using the parameter estimates found in this data set as opposed to being reestimated. If the option TASK=UPDATE is used, the parameters are estimated again, this time using the supplied estimates as starting values for the optimization process.

### OUTFOR= Data Set

The OUTFOR= data set contains the variables specified in the BY statement as well as the following variables:

| | |
|---|---|
| _NAME_ | variable name |
| _TIMEID_ | time ID values |
| PREDICT | predicted values |
| STD | prediction standard errors |
| LOWER | lower confidence limits |
| UPPER | upper confidence limits |
| ERROR | prediction errors |

If the forecasting step fails for a particular variable, no observations are recorded.

Procedures that define model specifications have TRANSFORM= and MEDIAN options used to apply functional transformations to series and the subsequent inverse transformation of forecast results. If the TRANSFORM= option is specified, the values in the preceding variables are the inverse transformed forecasts. If the MEDIAN option is specified, the median forecasts are stored; otherwise, the mean forecasts are stored.

### OUTINDEP= Data Set

The OUTINDEP= data set contains data used as input in the forecasting process. This information is useful if future values of input variables are automatically supplied by the HPFENGINE procedure. Such a case would occur if one or more input variables are listed in either the STOCHASTIC or CONTROLLABLE statement and if there are missing future values of these input variables.

The OUTINDEP= data set contains the variables specified in the BY statement as well as the following variables:

| | |
|---|---|
| _NAME_ | variable name |
| _TIMEID_ | time ID values |
| _X_ | values of the input variable _NAME_ |

### OUTMODELINFO= Data Set

The OUTMODELINFO= data set provides information about the selected forecast model and contains the following variables.

| | |
|---|---|
| _NAME_ | variable name |
| _MODEL_ | model specification name |
| _MODELTYPE_ | model specification type, either ARIMA, COMBINED, ESM, UCM, IDM, EXTERNAL, or INACTIVE |
| _DEPTRANS_ | name of transform applied to dependent variable or NONE if no transform |
| _SEASONAL_ | set to 1 if the model is seasonal, 0 otherwise |
| _TREND_ | set to 1 if the model has a trend, 0 otherwise |
| _INPUTS_ | set to 1 if one or more inputs are present, 0 otherwise |
| _EVENTS_ | set to 1 if one or more events are present, 0 otherwise |
| _OUTLIERS_ | set to 1 if one or more outliers are present, 0 otherwise |
| _STATUS_ | forecasting status. Nonzero values imply that no forecast was generated for the series. See "_STATUS_ Values" on page 229 for details. |
| _SOURCE_ | set to the model source based on the extended description element in the model's XML. This value corresponds to the SOURCE= attribute in the XML. The _SOURCE_ value for models generated from HPFDIAGNOSE is 'HPFDIAGNOSE'. The _SOURCE_ value for models generated from the HPF XML procedures is 'USERSPECIFED'. |

Characteristics that make a model seasonal vary according to the family of model. Those characteristics are listed here by model family.

| | |
|---|---|
| Smoothing models | The Winters, additive Winters, and seasonal smoothing models are considered seasonal. Linear, simple, double, and damped trend models smoothing are considered nonseasonal. |
| ARIMA models | An ARIMA model is considered seasonal if there is a difference with order equal to the seasonal cycle length of the time ID. The presence of MA or AR terms with lags equal to the seasonal cycle length also qualifies a model as seasonal. The presence of a predefined seasonal input is another factor that tags a model as seasonal. |
| Combined models | A combined model is considered seasonal if any of its contributing candidates is seasonal. |
| UCM | An unobserved component model is seasonal if there is a seasonal component present, specified by the SEASON statement in the HPFUCMSPEC procedure, or if there is a predefined seasonal input. |
| IDM | Intermittent demand models are always considered nonseasonal. |

Likewise, characteristics of a model that indicate a trend vary according to the family of model.

| | |
|---|---|
| Smoothing models | The simple and seasonal smoothing models do not have a trend. Linear, double, damped trend models, and multiplicative and additive Winters models do have a trend. |
| ARIMA models | An ARIMA model is considered to have a trend if there is a first- or second-order difference applied to the dependent variable. A predefined trend in an input statement also qualifies the model as having a trend component. |
| Combined models | A combined model has a trend if any of its contributing candidates has a trend component. |
| UCM | An unobserved component model has a trend if there is a slope component or a predefined trend in an input statement. |
| IDM | Intermittent demand models do have a trend. |

Additionally, a combined model has its other OUTMODELINFO attributes set as follows:

| | |
|---|---|
| _DEPTRANS_ | set to the common transform applied to all of the contributing candidate forecasts, or set to UNKNOWN if the contributing candidate forecasts use a mixture of transform types |
| _INPUTS_ | set to 1 if any of the contributing candidate forecasts has inputs, 0 otherwise |
| _EVENTS_ | set to 1 if any of the contributing candidate forecasts has events, 0 otherwise |
| _OUTLIERS_ | set to 1 if any of the contributing candidate forecasts has outliers, 0 otherwise |

### OUTPROCINFO= Data Set

The OUTPROCINFO= data set contains information about the run of the HPFENGINE procedure. The following variables are present:

| | |
|---|---|
| _SOURCE_ | set to the name of the procedure, in this case HPFENGINE |
| _NAME_ | name of an item being reported; can be the number of errors, notes, or warnings, number of forecasts requested, and so on |
| _LABEL_ | descriptive label for the item in _NAME_ |
| _STAGE_ | set to the current stage of the procedure; for HPFENGINE this is set to ALL |
| _VALUE_ | value of the item specified in _NAME_ |

### OUTSTAT= Data Set

The OUTSTAT= data set contains the variables specified in the BY statement as well as the following variables. The following variables contain observations related to the statistics-of-fit step:

| | |
|---|---|
| _NAME_ | variable name |
| _REGION_ | region in which the statistics are calculated. Statistics calculated in the fit region are indicated by FIT. Statistics calculated in the forecast region, which happens only if the BACK= option is greater than zero, are indicated by FORECAST. |
| DFE | degrees of freedom error |
| N | number of observations |
| NOBS | number of observations used |
| NMISSA | number of missing actuals |
| NMISSP | number of missing predicted values |
| NPARMS | number of parameters |
| TSS | total sum of squares |
| SST | corrected total sum of squares |
| SSE | sum of square error |
| MSE | mean square error |
| UMSE | unbiased mean square error |
| RMSE | root mean square error |
| URMSE | unbiased root mean square error |
| MAPE | mean absolute percent error |
| MAE | mean absolute error |
| MASE | mean absolute scaled error |
| RSQUARE | R square |
| ADJRSQ | adjusted R square |
| AADJRSQ | Amemiya's adjusted R square |
| RWRSQ | random walk R square |
| AIC | Akaike's information criterion |

| | |
|---|---|
| AICC | finite sample corrected AIC |
| SBC | Schwarz Bayesian information criterion |
| APC | Amemiya's prediction criterion |
| MAXERR | maximum error |
| MINERR | minimum error |
| MINPE | minimum percent error |
| MAXPE | maximum percent error |
| ME | mean error |
| MPE | mean percent error |
| MDAPE | median absolute percent error |
| GMAPE | geometric mean absolute percent error |
| MINPPE | minimum predictive percent error |
| MAXPPE | maximum predictive percent error |
| MSPPE | mean predictive percent error |
| MAPPE | symmetric mean absolute predictive percent error |
| MDAPPE | median absolute predictive percent error |
| GMAPPE | geometric mean absolute predictive percent error |
| MINSPE | minimum symmetric percent error |
| MAXSPE | maximum symmetric percent error |
| MSPE | mean symmetric percent error |
| SMAPE | symmetric mean absolute percent error |
| MDASPE | median absolute symmetric percent error |
| GMASPE | geometric mean absolute symmetric percent error |
| MINRE | minimum relative error |
| MAXRE | maximum relative error |
| MRE | mean relative error |
| MRAE | mean relative absolute error |
| MDRAE | median relative absolute error |
| GMRAE | geometric mean relative absolute error |
| MAPES | mean absolute error percent of standard deviation |
| MDAPES | median absolute error percent of standard deviation |
| GMAPES | geometric mean absolute error percent of standard deviation |

If the statistics-of-fit step fails for a particular variable, no observations are recorded.

## OUTSTATSELECT= Data Set

The OUTSTATSELECT= data set contains the same variables as the OUTSTAT= data set with the addition of the following:

_MODEL_ model specification name

_SELECT_ name of model selection list to which _MODEL_ belongs

_SELECTED_ whether or not this model was chosen to forecast the dependent series or used by the chosen forecast in the case when the chosen forecast is a combined model. Values set in the _SELECTED_ variable include:

- YES indicates the associated _MODEL_ is the primary model selected for the forecast
- USED indicates the associated _MODEL_ is used by the primary model in producing the final forecast
- USED_SELECT indicates the associated _MODEL_ is used by the primary model in the model selection region, but is not used in producing the final forecast
- NO indicates the associated _MODEL_ is neither selected nor used

## OUTSUM= Data Set

The OUTSUM= data set contains the variables specified in the BY statement as well as the variables listed in this section. The OUTSUM= data set records the summary statistics for each variable specified in a FORECAST statement and its multistep forecasts as determined by the LEAD= option specified.

Variables related to summary statistics are based on the ACCUMULATE= and SETMISSING= options:

_NAME_ variable name

_STATUS_ forecasting status. Nonzero values imply that no forecast was generated for the series. See "_STATUS_ Values" on page 229 for details.

NOBS number of observations

N number of nonmissing observations

NMISS number of missing observations

MIN minimum value

MAX maximum value

MEAN mean value

STDDEV standard deviation

Variables related to multistep forecast are based on the LEAD= option:

_LEAD*n*_ multistep forecast (*n* ranges from one to the value of the LEAD= option).

If the forecast step fails for a particular variable, the variables related to forecasting are set to missing.

### _STATUS_ Values

Values common to all _STATUS_ variables in various PROC HPFENGINE output data sets are listed here along with brief explanations of their meaning:

| | |
|---|---|
| 0 | The forecast was successfully completed. |
| 3000 | Model selection could not be completed. Forecast values are set to missing. |
| 3001 | Model selection could not be completed and NOALTLIST prohibits use of default exponential smoothing. Forecast values are set to missing. |
| 3002 | The forecast was completed subject to qualification that one or more input variables were omitted from the selected model. This can only occur in the context of ARIMAX or UCM models. |
| 3003 | The desired model could not be forecast. The forecast reverted to the default exponential smoothing model. |
| 3004 | The attempt to forecast the desired model produced an arithmetic exception. The forecast is generated by CATCH(ESM) processing. |
| 3005 | The attempt to forecast the desired model produced an arithmetic exception. The forecast is generated by CATCH(RW) processing. |
| 3006 | The attempt to forecast the desired model produced an arithmetic exception. The forecast is generated by CATCH(MISSING) processing. |
| 3007 | The mean value forecast is generated as a result of the MINOBS criterion. |
| 3008 | There were insufficient non-missing observations in the variable to be forecast. A missing value forecast is produced. |
| 3009 | There were insufficient non-zero observations in the variable to be forecast. A missing value forecast is produced. |

## ODS Table Names

**Table 7.2** ODS Tables Produced in PROC HPFENGINE

| ODS Table Name | Description | Specific Models |
|---|---|---|
| **ODS Tables Created by the PRINT=DESCSTATS Option** | | |
| Variable | forecast variable information | |
| DescStats | descriptive statistics | |
| **ODS Tables Created by the PRINT=SUMMARY Option** | | |
| Variable | forecast variable information | |
| ForecastSummary | forecast summary | |

**Table 7.2** *continued*

| ODS Table Name | Description | Specific Models |
|---|---|---|
| **ODS Tables Created by the PRINT=ESTIMATES Option** | | |
| Variable | forecast variable information | |
| ParameterEstimates | parameter estimates | |
| **ODS Tables Created by the PRINT=SELECT Option** | | |
| Variable | forecast variable information | |
| ModelSelection | model selection statistics | |
| **ODS Tables Created by the PRINT=FORECASTS Option** | | |
| Variable | forecast variable information | |
| Forecasts | forecast | |
| Demands | demands | IDM models only |
| DemandSummary | demand summary | IDM models only |
| **ODS Tables Created by the PRINT=STATISTICS Option** | | |
| Variable | forecast variable information | |
| FitStatistics | statistics of fit | |
| PerformanceStatistics | performance statistics | BACK= option only |
| **ODS Tables Created by the PRINT=BIAS Option** | | |
| Variable | forecast variable information | |
| BiasEstimates | bias test model parameter estimates | |
| TestUnbiasedness | bias test | |
| **ODS Tables Created by the PRINT=CANDIDATES Option** | | |
| Variable | forecast variable information | |
| ParameterEstimates | parameter estimates | |
| **ODS Tables Created by the PRINT=COMPONENTS Option** | | |
| Variable | forecast variable information | |
| ComponentEstimates | parameter estimates | |
| **ODS Tables Created by the PRINT=PERFORMANCE Option** | | |
| Variable | forecast variable information | |
| Performance | performance | BACK= option only |

**Table 7.2** *continued*

| ODS Table Name | Description | Specific Models |
|---|---|---|
| **ODS Tables Created by the PRINT=PERFORMANCESUMMARY Option** | | |
| Variable | forecast variable information | |
| PerformanceSummary | performance summary | BACK= option only |
| **ODS Tables Created by the PRINT=PERFORMANCEOVERALL Option** | | |
| Variable | forecast variable information | |
| PerformanceSummary | performance overall | BACK= option only |
| **ODS Tables Created by the PRINT=ALL Option** | | |
| DataSet | input data set | |
| Variable | forecast variable information | |
| DescStats | descriptive statistics | |
| Demands | demands | IDM models only |
| DemandSummary | demand summary | IDM models only |
| ModelSelection | model selection statistics | |
| ParameterEstimates | parameter estimates | |
| Forecasts | forecast | |
| BiasEstimates | bias test model parameter estimates | |
| TestUnbiasedness | bias test | |
| FitStatistics | statistics of fit | |
| PerformanceStatistics | performance statistics | BACK= option only |
| Performance | performance | BACK= option only |
| ForecastSummary | forecast summary | |

The ODS table ForecastSummary is related to all time series within a BY group. The other tables are related to a single series within a BY group.

## ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*).

Before you create graphs, ODS Graphics must be enabled (for example, with the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" in that chapter.

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" in that chapter.

This section describes the use of ODS for creating graphics with the HPFENGINE procedure.

## ODS Graph Names

PROC HPFENGINE assigns a name to each graph it creates using ODS. You can use these names to refer to the graphs when using ODS. The names are listed in Table 7.3.

You must specify the PLOT= option in the PROC HPFENGINE statement to select the desired plots. By default, no plots are generated.

**Table 7.3** ODS Graphs Produced by PROC HPFENGINE

| ODS Graph Name | Plot Description | Statement | PLOT= Option |
|---|---|---|---|
| CandidateErrorHoldoutPlot | Candidate model errors with holdout | PROC HPFENGINE | CANDIDATES |
| CandidateErrorPlot | Candidate model errors | PROC HPFENGINE | CANDIDATES |
| CandidateModelHoldoutPlot | Candidate models with holdout | PROC HPFENGINE | CANDIDATES |
| CandidateModelPlot | Candidate models | PROC HPFENGINE | CANDIDATES |
| ComponentEstimatesPlot | Component estimates | PROC HPFENGINE | COMPONENTS |
| DemandErrorsPlot | Average demand errors | PROC HPFENGINE | ERRORS |
| DemandForecastsPlot | Average demand forecasts | PROC HPFENGINE | FORECASTS |
| DemandIntervalHistogram | Demand interval histogram | PROC HPFENGINE | ALL |
| DemandIntervalPlot | Demand interval forecast plot | PROC HPFENGINE | ALL |
| DemandSizeHistogram | Demand size histogram | PROC HPFENGINE | MODELS |
| DemandSizePlot | Demand size forecast plot | PROC HPFENGINE | MODELS |
| ErrorACFNORMPlot | Standardized autocorrelation of prediction errors | PROC HPFENGINE | ACF |
| ErrorACFPlot | Autocorrelation of prediction errors | PROC HPFENGINE | ACF |
| ErrorCorrelationPlots | Prediction error panel | PROC HPFENGINE | CORR |
| ErrorHistogram | Prediction error histogram | PROC HPFENGINE | ERRORS |
| ErrorIACFNORMPlot | Standardized inverse autocorrelation of prediction errors | PROC HPFENGINE | IACF |
| ErrorIACFPlot | Inverse autocorrelation of prediction errors | PROC HPFENGINE | IACF |
| ErrorPACFNORMPlot | Standardized partial autocorrelation of prediction errors | PROC HPFENGINE | PACF |
| ErrorPACFPlot | Partial autocorrelation of prediction errors | PROC HPFENGINE | PACF |
| ErrorPeriodogram | Periodogram of prediction errors | PROC HPFENGINE | PERIODOGRAM |

**Table 7.3** *continued*

| ODS Graph Name | Plot Description | Statement | Option |
|---|---|---|---|
| ErrorSpectralDensityPlot | Spectral density estimates of prediction errors | PROC HPFENGINE | SPECTRUM |
| ErrorPlot | Prediction errors | PROC HPFENGINE | ERRORS |
| ErrorWhiteNoiseLogProbPlot | White noise log probability plot of prediction errors | PROC HPFENGINE | WN |
| ErrorWhiteNoisePlot | White noise plot of prediction errors | PROC HPFENGINE | ALL |
| ErrorWhiteNoiseProbPlot | White noise probability plot of prediction errors | PROC HPFENGINE | WN |
| ForecastSeasonalCyclePlot | Forecast seasonal cycles | PROC HPFENGINE | FORECASTCYCLES |
| ForecastsOnlyPlot | Forecasts only | PROC HPFENGINE | FORECASTSONLY |
| ForecastsPlot | Forecasts | PROC HPFENGINE | FORECAST |
| ModelForecastsPlot | Model and forecasts | PROC HPFENGINE | ALL |
| ModelPlot | Model only | PROC HPFENGINE | ALL |
| StockingAveragePlot | Stocking average | PROC HPFENGINE | FORECASTS |
| StockingLevelPlot | Stocking level | PROC HPFENGINE | FORECASTS |

# Examples: HPFENGINE Procedure

## Example 7.1: The TASK Option

The default selection list is used in this example. The first call to the HPFENGINE procedure, which follows, uses the default TASK = SELECT action. A model is selected from the default list, parameters are estimated, and a forecast is produced. Selection results are shown in Output 7.1.1.

```
proc hpfengine data=sashelp.citimon
               outfor=outfor
               print=select;
   id date interval=month;
   forecast eec;
run;
```

**Output 7.1.1** Selection and Forecast Results

**The HPFENGINE Procedure**

| Model Selection Criterion = RMSE | | | |
|---|---|---|---|
| **Model** | **Statistic** | **Selected** | **Label** |
| **SMSIMP** | . | Removed | Simple Exponential Smoothing |
| **SMDAMP** | . | Removed | Damped-Trend Exponential Smoothing |
| **SMLIN** | . | Removed | Linear Exponential Smoothing |
| **SMADWN** | 0.16293237 | No | Winters Method (Additive) |
| **SMWINT** | 0.17985708 | No | Winters Method (Multiplicative) |
| **SMSEAS** | 0.16291415 | Yes | Seasonal Exponential Smoothing |

The second call to the HPFENGINE procedure follows. It demonstrates how to use the same model specifications but alter the selection criterion and add a holdout. It is not necessary to create a new selection list to make these changes, since the TASK = SELECT option can override the settings in an existing list. Selection results are shown in Output 7.1.2.

```
proc hpfengine data=sashelp.citimon
               outfor=outfor
               outest=outest
               print=select
               task=select(criterion=mape holdout=24 override);
   id date interval=month;
   forecast eec;
run;
```

**Output 7.1.2** Selection and Forecast Results

**The HPFENGINE Procedure**

| Model Selection Criterion = MAPE | | | |
|---|---|---|---|
| **Model** | **Statistic** | **Selected** | **Label** |
| **SMSIMP** | . | Removed | Simple Exponential Smoothing |
| **SMDAMP** | . | Removed | Damped-Trend Exponential Smoothing |
| **SMLIN** | . | Removed | Linear Exponential Smoothing |
| **SMADWN** | 2.2881620 | Yes | Winters Method (Additive) |
| **SMWINT** | 2.9441207 | No | Winters Method (Multiplicative) |
| **SMSEAS** | 2.5775447 | No | Seasonal Exponential Smoothing |

Perhaps there are revisions to recent historical data and there is a need to forecast using the updated information but with the same model used to forecast earlier data. This is done easily with the TASK = UPDATE option. The following DATA step is used to simulate revision to the data.

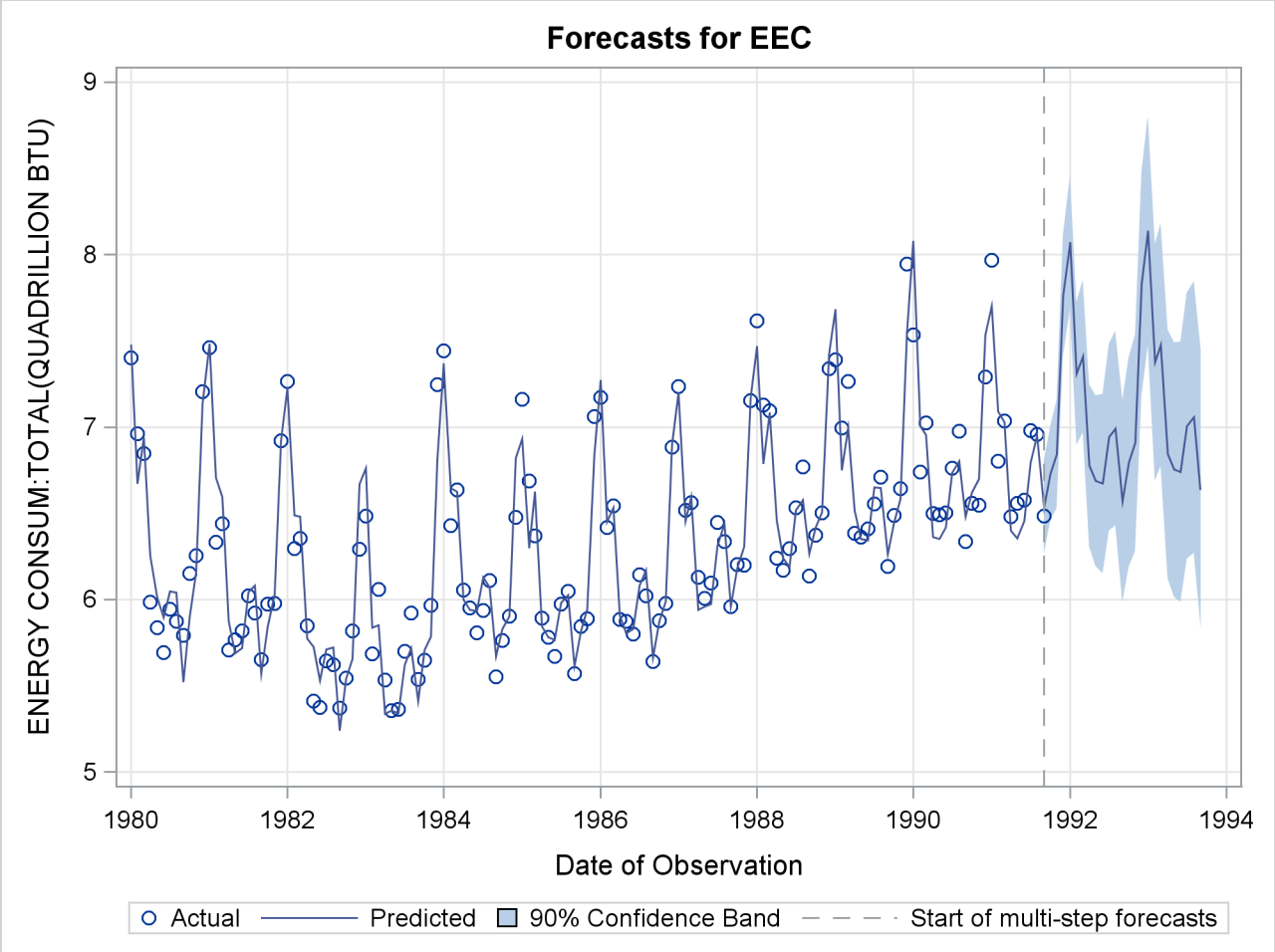
```
data citimon;
   set sashelp.citimon;
   if date ge '01JAN1991'd then eec = eec + 0.1;
run;
```

In the following statements, the HPFENGINE procedure is called with TASK=UPDATE. The additional instruction to change the confidence interval to 90model is the same as was used in the previous call of the HPFENGINE procedure. A reference to this model, together with estimates of its parameters, is found in the OUTEST= data set. The parameters are estimated again using the full range of data, and the prior parameter estimates are used as starting points in the optimization of the new estimates.

```
proc hpfengine data=citimon
               outfor=outfor
               inest=outest
               plot=forecasts
               lead=24
               task=update(alpha=.1 override);
   id date interval=month;
   forecast eec;
run;
```

Figure 7.1.3 shows the forecast plot from the PLOT=FORECASTS option.

**Output 7.1.3** Forecasts



## Example 7.2: Different Types of Input

This example demonstrates the use of different input types in the HPFENGINE procedure.

The following statements read Input Gas Rate and Output CO2 from a gas furnace. (Data values are not shown. See "Series J" in (Box, Jenkins, and Reinsel 1994) for the values.)

```
data seriesj;
   input x y @@;
   label x = 'Input Gas Rate'
         y = 'Output CO2';
   date = intnx( 'day', '01jan1950'd, _n_-1 );
   format Date DATE.;
datalines;
-0.109  53.8  0.000  53.6  0.178  53.5  0.339  53.5

... more lines ...
```

Begin by creating an ARIMA model specification using the HPFARIMASPEC procedure as follows. The new model specification is then placed into a model selection list by using the HPFSELECT procedure.

```
proc hpfarimaspec repository=sasuser.repository
                  name=arimasp;
   dependent symbol=Y p=2;
   input     symbol=X num=2 den=1 lag=3;
run;

proc hpfselect repository=sasuser.repository
               name=myselect;
   spec arimasp;
run;
```

There are no future values of the independent variables given in the seriesj data set. For this example a DATA step is used as follows to set the last few observations of the dependent value to missing. Values of the independent variable are left intact. This ensures that some future values of the independent variable are available for forecasting the dependent variable between 17OCT1950 and the end of the data set.

```
data seriesj_trunc;
   set seriesj;
   if (date >= '17oct1950'd) then y = .;
run;
```

In the following statements, the HPFENGINE procedure is called using the INPUT statement to identify the data set variable "x" as input. No missing future values, even if required, are computed for "x". The OUTINDEP= data set contains values of the input variable used in the forecast.

```
proc hpfengine data=seriesj_trunc
               outfor=outfor
               outindep=outindep
               repository=sasuser.repository
               globalselection=myselect
               lead=7;
   id date interval=day;
```

```
   forecast y;
   input    x;
run;

proc print data=outindep(where=(date >= '17oct1950'd)) label noobs;
var date x;
run;

proc print data=outfor(where=(date >= '17oct1950'd)) label noobs;
   var date predict upper lower;
run;
```

The user-supplied future values of the input variable used in the forecast as well as the forecasts themselves are shown in Output 7.2.1 and Output 7.2.2.

**Output 7.2.1** Future Values of X Supplied by the User

| date | Input Values |
|---|---|
| 17OCT1950 | 0.204 |
| 18OCT1950 | 0.253 |
| 19OCT1950 | 0.195 |
| 20OCT1950 | 0.131 |
| 21OCT1950 | 0.017 |
| 22OCT1950 | -0.182 |
| 23OCT1950 | -0.262 |

**Output 7.2.2** Forecasts for Y

| date | Predicted Values | Upper Confidence Limits | Lower Confidence Limits |
|---|---|---|---|
| 17OCT1950 | 57.0684 | 57.5116 | 56.6252 |
| 18OCT1950 | 56.4050 | 57.1793 | 55.6306 |
| 19OCT1950 | 55.3430 | 56.3381 | 54.3480 |
| 20OCT1950 | 54.1844 | 55.2923 | 53.0765 |
| 21OCT1950 | 53.2288 | 54.3756 | 52.0820 |
| 22OCT1950 | 52.6228 | 53.7750 | 51.4705 |
| 23OCT1950 | 52.3789 | 53.5315 | 51.2263 |

To demonstrate use of the STOCHASTIC statement, a DATA step is used as follows to eliminate future values of the input variable.

```
data seriesj_trunc;
   set seriesj;
   if (date < '17oct1950'd);
run;
```

In the following statements, the HPFENGINE procedure identifies the input variable using the STOCHASTIC statement and automatically forecasts the input variable.

```
proc hpfengine data=seriesj_trunc
               outfor=outfor
               outindep=outindep
               repository=sasuser.repository
```

```
                globalselection=myselect
                lead=7;
   id date interval=day;
   forecast   y;
   stochastic x;
run;

proc print data=outindep(where=(date >= '17oct1950'd)) label noobs;
var date x;
run;

proc print data=outfor(where=(date >= '17oct1950'd)) label noobs;
   var date predict upper lower;
run;
```

The future values of the input variable, automatically forecast, are shown in Output 7.2.3. The forecasts of the dependent variable are shown in Output 7.2.4.

**Output 7.2.3** Future Values of X Automatically Forecast

| date | Input Values |
|---|---|
| 17OCT1950 | 0.21222 |
| 18OCT1950 | 0.34557 |
| 19OCT1950 | 0.44535 |
| 20OCT1950 | 0.52002 |
| 21OCT1950 | 0.57588 |
| 22OCT1950 | 0.61769 |
| 23OCT1950 | 0.64897 |

**Output 7.2.4** Forecasts for Y

| date | Predicted Values | Upper Confidence Limits | Lower Confidence Limits |
|---|---|---|---|
| 17OCT1950 | 57.0684 | 57.5116 | 56.6252 |
| 18OCT1950 | 56.4050 | 57.1793 | 55.6306 |
| 19OCT1950 | 55.3430 | 56.3381 | 54.3480 |
| 20OCT1950 | 54.1798 | 55.2877 | 53.0719 |
| 21OCT1950 | 53.1713 | 54.3181 | 52.0244 |
| 22OCT1950 | 52.4125 | 53.5647 | 51.2602 |
| 23OCT1950 | 51.9055 | 53.0581 | 50.7528 |

## Example 7.3: Incorporating Events

This example creates an event called PROMOTION. The event is added as a simple regressor to each ARIMA specification in the selection list.

First a DATA step is used to generate a data set with a shift beginning at “01OCT1980”.

```
   data shifted;
      set sashelp.workers;
      if date >= '01oct80'd then Y = electric+100;
      else Y = electric;
      drop masonry electric;
   run;
```

Next the HPFEVENTS procedure is used to create an events database. The database will contain the definition of an event named "promotion," a level shift beginning at "01OCT1980."

```
proc hpfevents data=shifted lead=12;
   id date interval=month;
   eventdef   promotion='01oct80'd / TYPE=LS;
   eventdata  out= evdsout1;
   eventdummy out= evdumout1;
run;
```

Then two ARIMA model specification are created as follows. Both of them will be added to a selection list for use by the HPFENGINE procedure.

```
proc hpfarimaspec repository=sasuser.repository
                  name=sp1
                  label="ARIMA(0,1,2)(0,1,1)_12 No Intercept";
   dependent symbol=Y q=(1,2)(12) diflist=1 12 noint;
run;

proc hpfarimaspec repository=sasuser.repository
                  name=sp2
                  label="ARIMA(2,1,0)(1,1,0)_12 No Intercept";
   dependent symbol=Y p=(1, 2)(12) diflist=1 12 noint;
run;
```

The HPFSELECT procedure then creates a new selection list that contains the two ARIMA specifications and uses the EVENTMAP option to add the "promotion" event to each of them, as follows.

```
proc hpfselect repository=sasuser.repository
               name=myselect
               label="My Selection List";
   select select=mape;
   spec sp1 sp2 /
        eventmap(symbol=_NONE_ event=promotion);
run;
```

The HPFENGINE procedure fits both ARIMA models to the data with the "promotion" event, as follows.

```
proc hpfengine data=shifted
               globalselection=myselect
               repository=sasuser.repository
               inevent=evdsout1
               print=(select estimates)
               plot=forecasts;
   id date interval=month;
   forecast y;
run;
```

The results of the model selection are shown in Output 7.3.1. Parameter estimates for the selected model are shown in Output 7.3.2.

**Output 7.3.1** Model Selection Results

**The HPFENGINE Procedure**

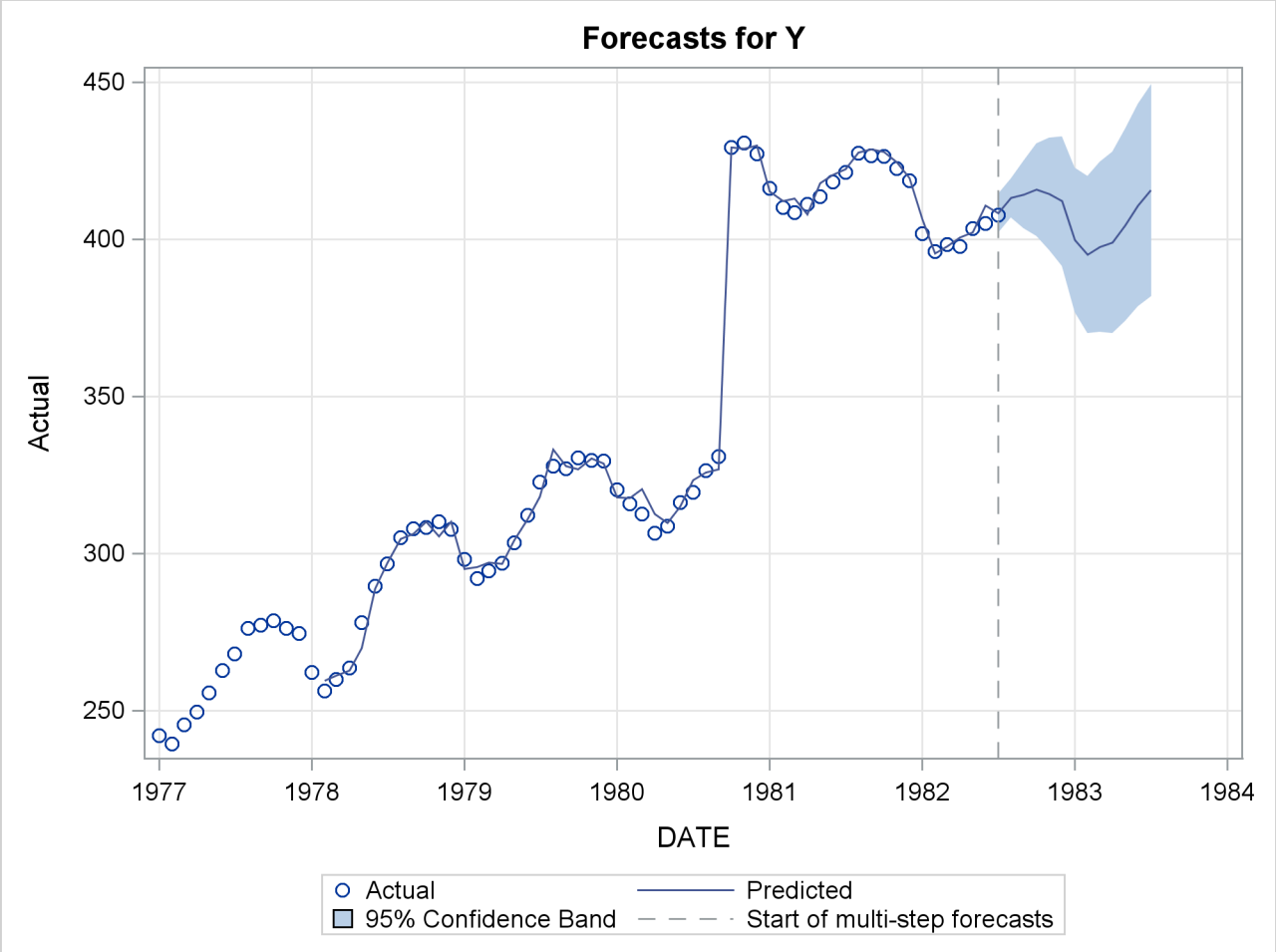
| Model Selection Criterion = MAPE | | | |
|---|---|---|---|
| **Model** | **Statistic** | **Selected** | **Label** |
| **SP1** | 0.70600173 | Yes | ARIMA(0,1,2)(0,1,1)_12 No Intercept |
| **SP2** | 0.76609476 | No | ARIMA(2,1,0)(1,1,0)_12 No Intercept |

**Output 7.3.2** Parameter Estimates of Selected Model

| Parameter Estimates for SP1 Model | | | | | |
|---|---|---|---|---|---|
| **Component** | **Parameter** | **Estimate** | **Standard Error** | **t Value** | **Approx Pr > \|t\|** |
| **Y** | **MA1_1** | -0.41281 | 0.14173 | -2.91 | 0.0053 |
| **Y** | **MA1_2** | -0.21826 | 0.14382 | -1.52 | 0.1354 |
| **Y** | **MA2_12** | 0.80211 | 0.11064 | 7.25 | <.0001 |
| **PROMOTION** | **SCALE** | 94.91032 | 2.91440 | 32.57 | <.0001 |

Figure 7.3.3 shows the forecast plot from the PLOT=FORECASTS option.

**Output 7.3.3** Forecasts



An alternate way to include inputs is to refer to the dummy variable in the data set created by the EVENTDUMMY statement in the HPFEVENTS procedure, as shown in the following statements.

The ARIMA models are different in this case because the event data are explicitly added as an input. In order to produce the same results as the EVENTMAP method of handling events, the input variables are differenced in the same manner as the dependent variable.

```
proc hpfarimaspec repository=sasuser.repository
                  name=sp1
                  label="ARIMA(0,1,2)(0,1,1)_12 No Intercept";
   dependent symbol=Y q=(1,2)(12) diflist=1 12 noint;
   input     symbol=promotion diflist=1 12;
run;

proc hpfarimaspec repository=sasuser.repository
                  name=sp2
                  label="ARIMA(2,1,0)(1,1,0)_12 No Intercept";
   dependent symbol=Y p=(1, 2)(12) diflist=1 12 noint;
   input     symbol=promotion diflist=1 12;
run;
```

Again, the HPFSELECT procedure creates a new selection list that contains the two ARIMA specifications, as follows. This time no EVENTMAP option is required.

```
proc hpfselect repository=sasuser.repository
               name=myselect
               label="My Selection List";
   select select=mape;
   spec sp1 sp2;
run;
```

The HPFENGINE procedure needs to find the "promotion" variable in the input data set. A DATA step is used as follows to merge the input variable with the data set that contains the dependent variable.

```
   data shifted;
      merge shifted evdumout1(drop=y);
      by date;
   run;
```

The HPFENGINE procedure fits both ARIMA models to the data with the "promotion" input, as follows. The results of the model selection are shown in Output 7.3.4. Notice that the results are the same as the results of using the EVENTMAP option in the HPFSELECT procedure. This technique of avoiding the EVENTMAP option and including the event is useful if events need to enter the forecast model through more complex transfer functions.

```
proc hpfengine data=shifted
               globalselection=myselect
               repository=sasuser.repository
               inevent=evdsout1
               print=(select estimates)
               plot=forecasts;
   id date interval=month;
   forecast y;
   input promotion;
run;
```

**Output 7.3.4** Model Selection Results

**The HPFENGINE Procedure**

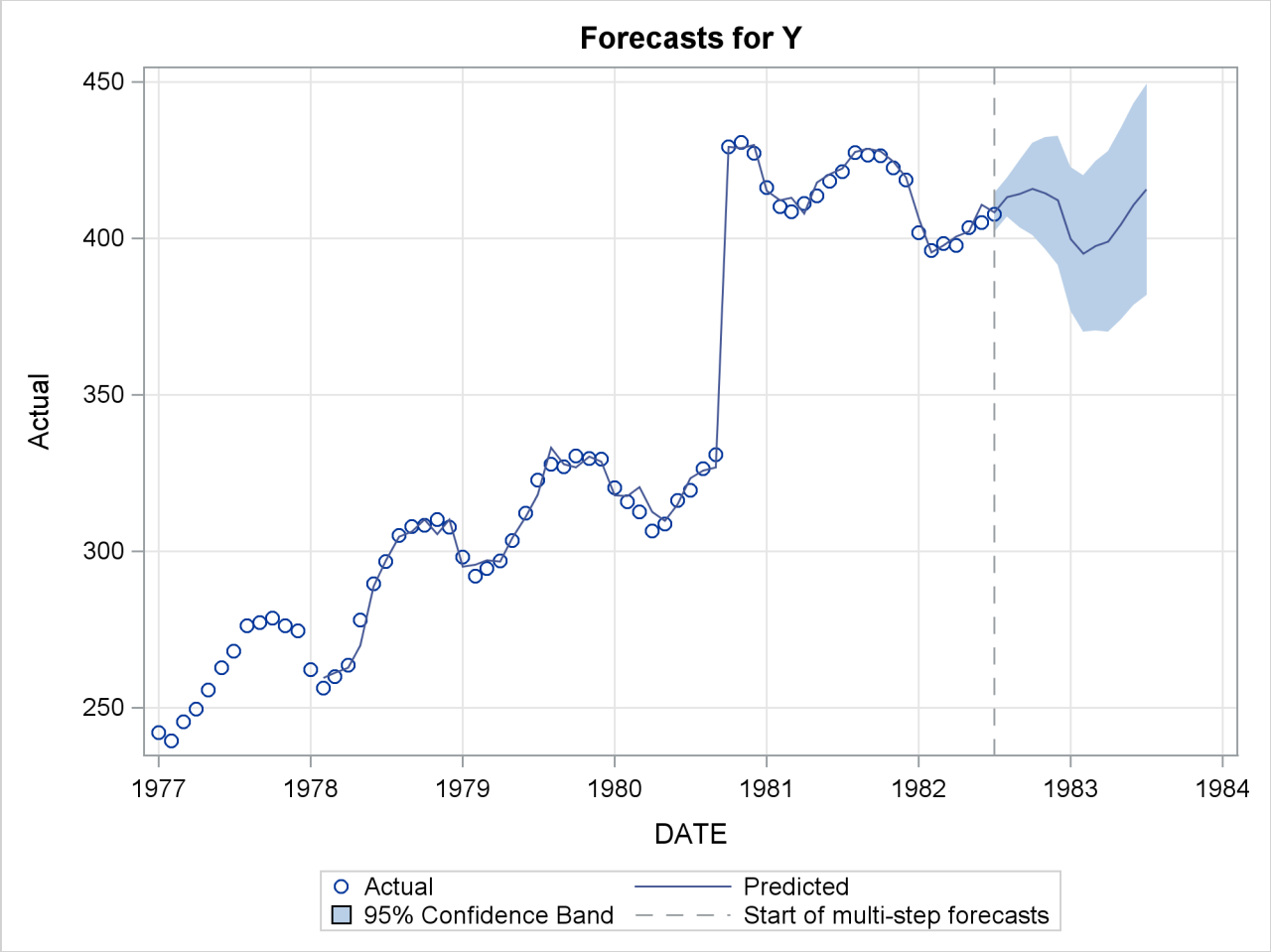
| Model Selection Criterion = MAPE | | | |
|---|---|---|---|
| **Model** | **Statistic** | **Selected** | **Label** |
| **SP1** | 0.70600173 | Yes | ARIMA(0,1,2)(0,1,1)_12 No Intercept |
| **SP2** | 0.76609476 | No | ARIMA(2,1,0)(1,1,0)_12 No Intercept |

Parameter estimates for the selected model are in Output 7.3.5. The forecast is displayed in Figure 7.3.6.

**Output 7.3.5** Parameter Estimates of Selected Model

| Parameter Estimates for SP1 Model | | | | | |
|---|---|---|---|---|---|
| **Component** | **Parameter** | **Estimate** | **Standard Error** | **t Value** | **Approx Pr > \|t\|** |
| **Y** | **MA1_1** | -0.41281 | 0.14173 | -2.91 | 0.0053 |
| **Y** | **MA1_2** | -0.21826 | 0.14382 | -1.52 | 0.1354 |
| **Y** | **MA2_12** | 0.80211 | 0.11064 | 7.25 | <.0001 |
| **promotion** | **SCALE** | 94.91032 | 2.91440 | 32.57 | <.0001 |

**Output 7.3.6** Forecasts




## Example 7.4: Using the SCORE Statement

This example demonstrates the use of the SCORE statement to create a forecast score file. Score files are used by the HPFSCSUB function to produce forecasts outside of the HPFENGINE procedure.

In this particular case price and sales data are present. A forecast score file is produced with price as a controllable input. The NLP procedure is used to maximize a simple expression of total revenue in the forecast horizon, in the way that the optimizer adjusts the price inputs of the forecast function. The input values found by the NLP procedure are then used in the HPFENGINE procedure again to create a forecast plot.

The following DATA step uses data from a single BY group in the SASUSER.PRICEDATA data set.

```
data pricedata;
   set sashelp.pricedata(where=(product=1));
   keep date sale price;
run;
```

An ARIMAX model specification is created as follows, together with a selection list that references this specification.

```
proc hpfarimaspec repository=work.repository name=arimax;
   forecast symbol=sale q=(12) diflist=12 noint;
   input symbol=price diflist=12;
run;

proc hpfselect repository=work.repository name=select;
   spec arimax;
run;
```

The HPFENGINE procedure then creates a forecast score file by using the controllable input variable price, as follows. The designation of this input as "controllable" means that the input can be manipulated in the HPFSCSUB function.

```
proc hpfengine data=pricedata
               repository=work.repository
               scorerepository=work.repository
               globalselection=select
               print=estimates
               plot=forecasts;
   id date interval=month;
   forecast sale;
   controllable price / extend=last;
   score;
run;
```

The parameter estimates of the model are shown in Output 7.4.1. A plot of the forecast results is produced and shown in Figure 7.4.2.

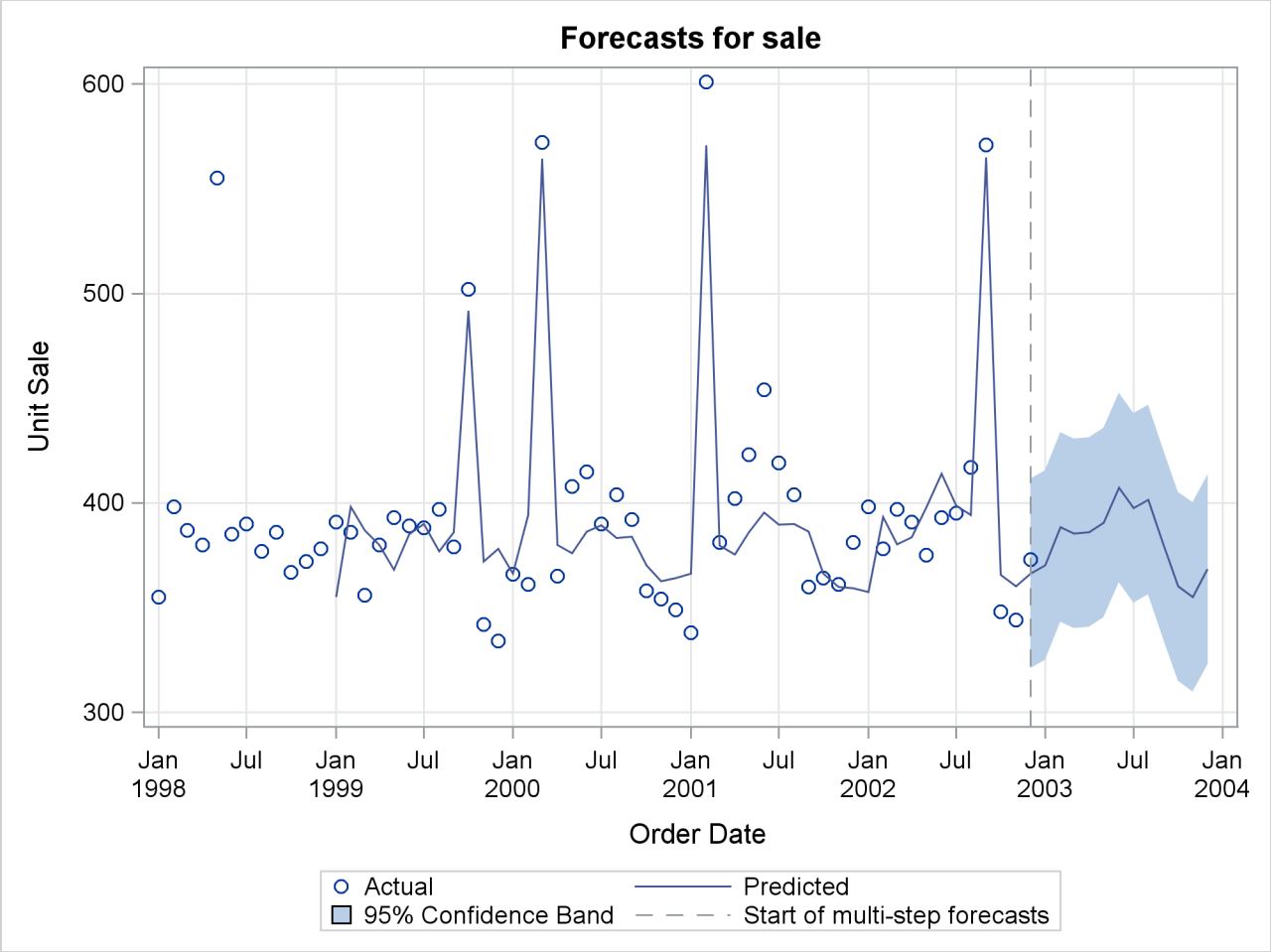
**Output 7.4.1** Parameter Estimates of Selected Model

**The HPFENGINE Procedure**

**Parameter Estimates for ARIMAX Model**

| Component | Parameter | Estimate | Standard Error | t Value | Approx Pr > \|t\| |
|---|---|---|---|---|---|
| sale | MA1_12 | 0.68351 | 0.13905 | 4.92 | <.0001 |
| price | SCALE | -23.82518 | 1.34147 | -17.76 | <.0001 |

**Output 7.4.2** Forecasts




The NLP procedure is used as follows to maximize an objective function whose definition depends on forecasts from the score file. Some lower bounds are set on the price for certain months.

```
filename score catalog "work.repository.scor0.xml";
proc nlp tech=nmsimp noprint out=outnlp(keep=p1-p6);
   max trev;
   parms p1-p6;
   bounds p3 p4 >= 20.0;
   bounds p1 p2 p5 p6 >= 52.3;
   initial = 52.3;
   q1 = .; q2 = .; q3 = .;
   q4 = .; q5 = .; q6 = .;
   call HPFSCSUB('score', 6, 'price', p1, p2, p3, p4, p5, p6,
                 'predict', q1, q2, q3, q4, q5, q6);
   trev = q1*p1 + q2*p2 + q3*p3 + q4*p4 + q5*p5 +q6*p6;
run;
```

Some manipulation of the output from the NLP procedure is needed before the HPFENGINE procedure is called again with the new future price values. In the following statements, the new price data are transposed and a date variable is added. The result is shown in Output 7.4.3.

```
proc transpose data=outnlp out=outnlp;
   var p1-p6;
run;

data outnlp;
   drop _name_;
   format date date9.;
   set outnlp(rename=(col1=price));
   date = intnx('month', '01dec02'd, _n_);
run;

proc print data=outnlp noobs;
run;
```

**Output 7.4.3** Future Values of Price

| date | price |
|---|---|
| 01JAN2003 | 52.3000 |
| 01FEB2003 | 52.3000 |
| 01MAR2003 | 34.2393 |
| 01APR2003 | 34.2512 |
| 01MAY2003 | 52.3000 |
| 01JUN2003 | 52.3000 |

The original data set is now extended, with proper date information, and merged with the results of the optimization, as follows.

```
data pricedataExtend;
   set pricedata;
   drop i;
   output;
   if _n_ ge 60 then do;
      sale = .;
      do i=1 to 6;
         date = intnx('month', '01dec02'd, i);
         output;
      end;
   end;
run;

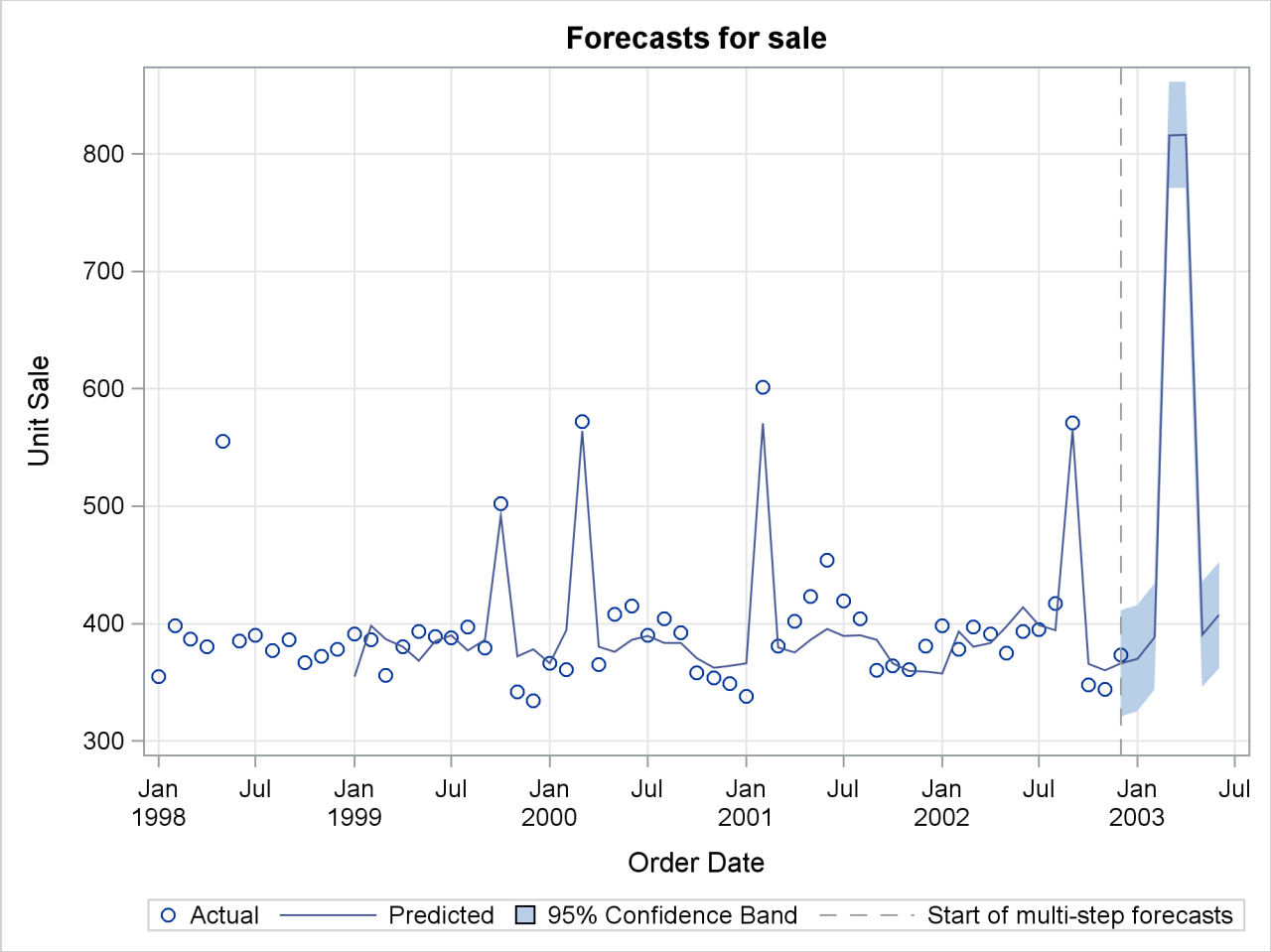
data pricedataExtend;
   merge pricedataExtend outnlp;
   by date;
run;
```

The HPFENGINE procedure then uses the optimized price input to forecast future sales, as follows. A plot of the forecasts is shown in Figure 7.4.4.

```
proc hpfengine data=pricedataExtend
               repository=work.repository
               globalselection=select
               plot=forecasts;
   id date interval=month;
```

```
   forecast sale;
   input price;
run;
```

**Output 7.4.4** Forecasts




## Example 7.5: HPFENGINE and HPFDIAGNOSE Procedures

The HPFDIAGNOSE procedure is often used in conjunction with the HPFENGINE procedure. This example demonstrates the most basic interaction between the two. In the following statements, model specifications are created by the HPFDIAGNOSE procedure and are those specifications are then fit to the data by using the HPFENGINE procedure.

```
proc hpfdiagnose data=sashelp.air
                 repository=work.repository
                 outest=est;
   id date interval=month;
   forecast air;
run;
proc hpfengine data=sashelp.air
               inest=est outest=outest
```

```
                repository=work.repository
                print=(select estimates summary)
                plot=forecasts;
   id date interval=month;
   forecast air;
run;
```

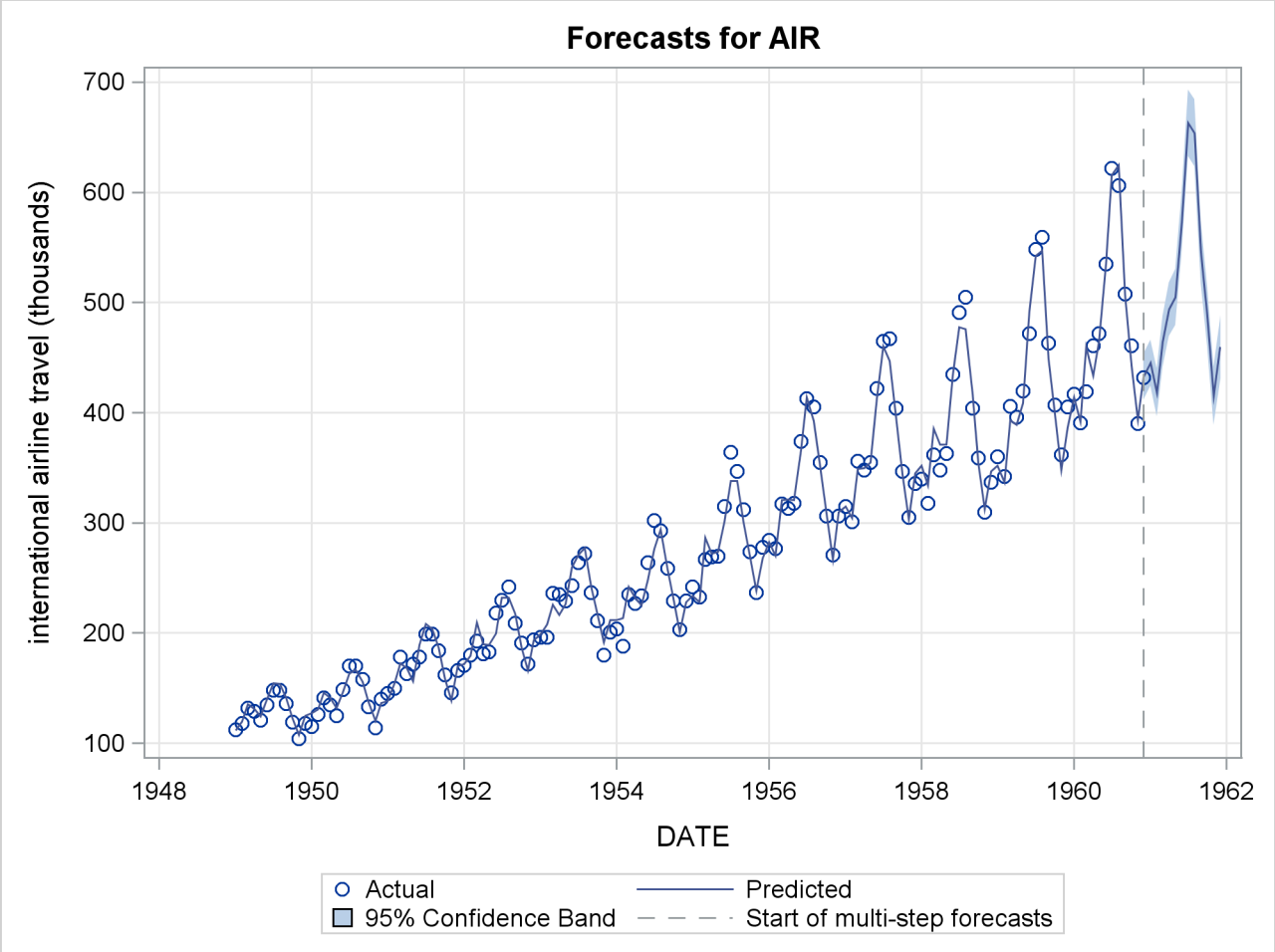
The HPFENGINE procedure output is shown in Output 7.5.1. Forecasts are shown in Figure 7.5.2.

**Output 7.5.1** Selection and Forecast Results

**The HPFENGINE Procedure**

| Model Selection Criterion = RMSE | | | |
|---|---|---|---|
| **Model** | **Statistic** | **Selected** | **Label** |
| **diag0** | 10.695241 | No | ARIMA: AIR ~ P = 1 D = (1,12) NOINT |
| **diag1** | 10.579085 | Yes | Winters Method (Multiplicative) |

**Output 7.5.2** Forecasts




It is also possible to compare the performance of model specifications that you create with those automatically generated by the HPFDIAGNOSE procedure.

In the following statements, two model specifications are created, together with a selection list to reference those specifications.

```
proc hpfarimaspec repository=work.repository name=myarima;
   forecast symbol=sale transform=log q=(1 12) diflist=(1 12) noint;
run;

proc hpfucmspec repository=work.repository name=myucm;
   forecast symbol=sale transform=log;
   irregular;
   level;
   season length=12;
run;

proc hpfselect repository=work.repository name=select;
   spec myarima myucm;
run;
```

The model selection list is passed to the HPFDIAGNOSE procedure by using the INSELECTNAME= option as follows. The new selection list ultimately created by the HPFDIAGNOSE procedure includes all model specifications in the INSELECTNAME= selection list together with automatically generated model specifications.

```
proc hpfdiagnose data=sashelp.air
                 inselectname=select
                 repository=work.repository
                 outest=est
                 criterion=mape;
   id date interval=month;
   forecast air;
run;
```

The HPFENGINE procedure selects the best-fitting model as follows. Selection results are shown in Figure 7.5.3, parameter estimates of the selected model are shown in Figure 7.5.4, and the forecast summary is shown in Figure 7.5.5.

```
proc hpfengine data=sashelp.air
               inest=est
               repository=work.repository
               print=(select estimates summary)
               plot=forecasts;
   id date interval=month;
   forecast air;
run;
```

**Output 7.5.3** Model Selection Results

**The HPFENGINE Procedure**

| Model Selection Criterion = MAPE | | | |
|---|---|---|---|
| **Model** | **Statistic** | **Selected** | **Label** |
| **diag3** | 3.1212192 | No | ARIMA: AIR ~ P = 1 D = (1,12) NOINT |
| **diag4** | 3.0845016 | No | Winters Method (Multiplicative) |
| **MYARIMA** | 2.9672282 | Yes | ARIMA: Log( SALE ) ~ D = (1,12) Q = (1,12) NOINT |
| **MYUCM** | 3.1842031 | No | UCM: Log( SALE ) = LEVEL + SEASON + ERROR |

**Output 7.5.4** Parameter Estimates of Selected Model

**Parameter Estimates for MYARIMA Model**

| Component | Parameter | Estimate | Standard Error | t Value | Approx Pr > \|t\| |
|---|---|---|---|---|---|
| AIR | MA1_1 | 0.24576 | 0.07364 | 3.34 | 0.0011 |
| AIR | MA1_12 | 0.50641 | 0.07676 | 6.60 | <.0001 |

**Output 7.5.5** Forecast Summary

**Forecast Summary**

| Variable | Value | JAN1961 | FEB1961 | MAR1961 | APR1961 | MAY1961 | JUN1961 | JUL1961 | AUG1961 | SEP1961 | OCT1961 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| AIR | Predicted | 450.3513 | 425.6158 | 478.6340 | 501.0466 | 512.5109 | 584.8311 | 674.9025 | 667.8935 | 558.3906 | 499.4696 |

**Forecast Summary**

| Variable | NOV1961 | DEC1961 |
|---|---|---|
| AIR | 430.1668 | 479.4592 |

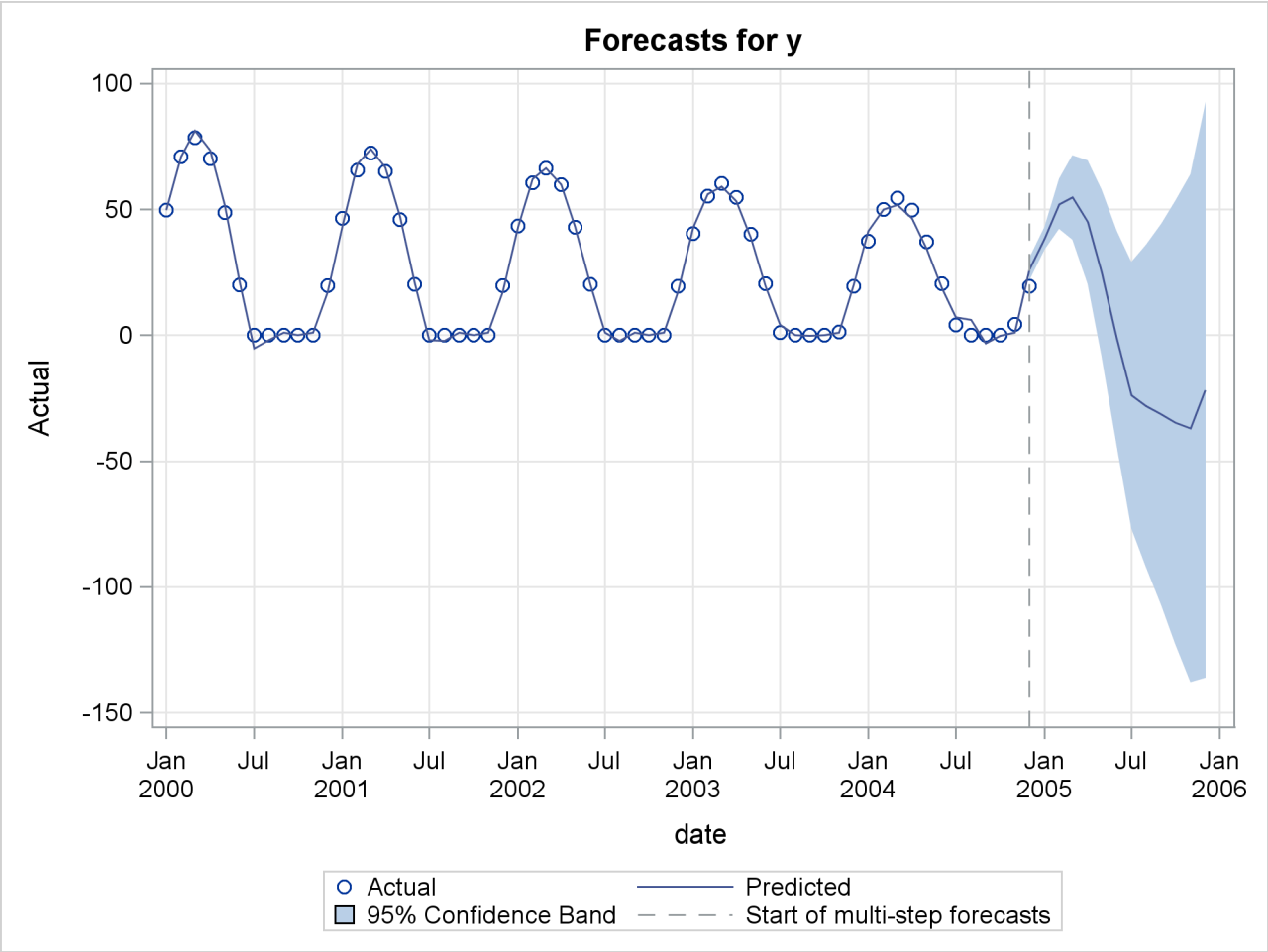
## Example 7.6: The ADJUST Statement

This example illustrates the use of adjustment operations. The following statements create seasonal test data with the in-season data trending downward. When the data are forecast, note that the out-of-season range in the forecast horizon is negative, due to the trend. (See the plot in Figure 7.6.1.)

```
data test;
   do i=1 to 60;
      y = (60-i/2)*sin(2*3.14*i/12) + 20;
      if y lt 0 then do;
         y =0;
         inseason = 0;
      end;
      else do;
         inseason = 1;
      end;
      date = intnx('month', '01jan2000'd, i-1);
      output;
   end;
   do i=61 to 72;
      y = .;
      date = intnx('month', '01jan2000'd, i-1);
      if i le 66 then inseason = 1;
      else inseason = 0;
      output;
   end;
run;

proc hpfengine data=test
               plot=forecasts;
   id date interval=month;
   forecast y;
run;
```

**Output 7.6.1** Forecasts

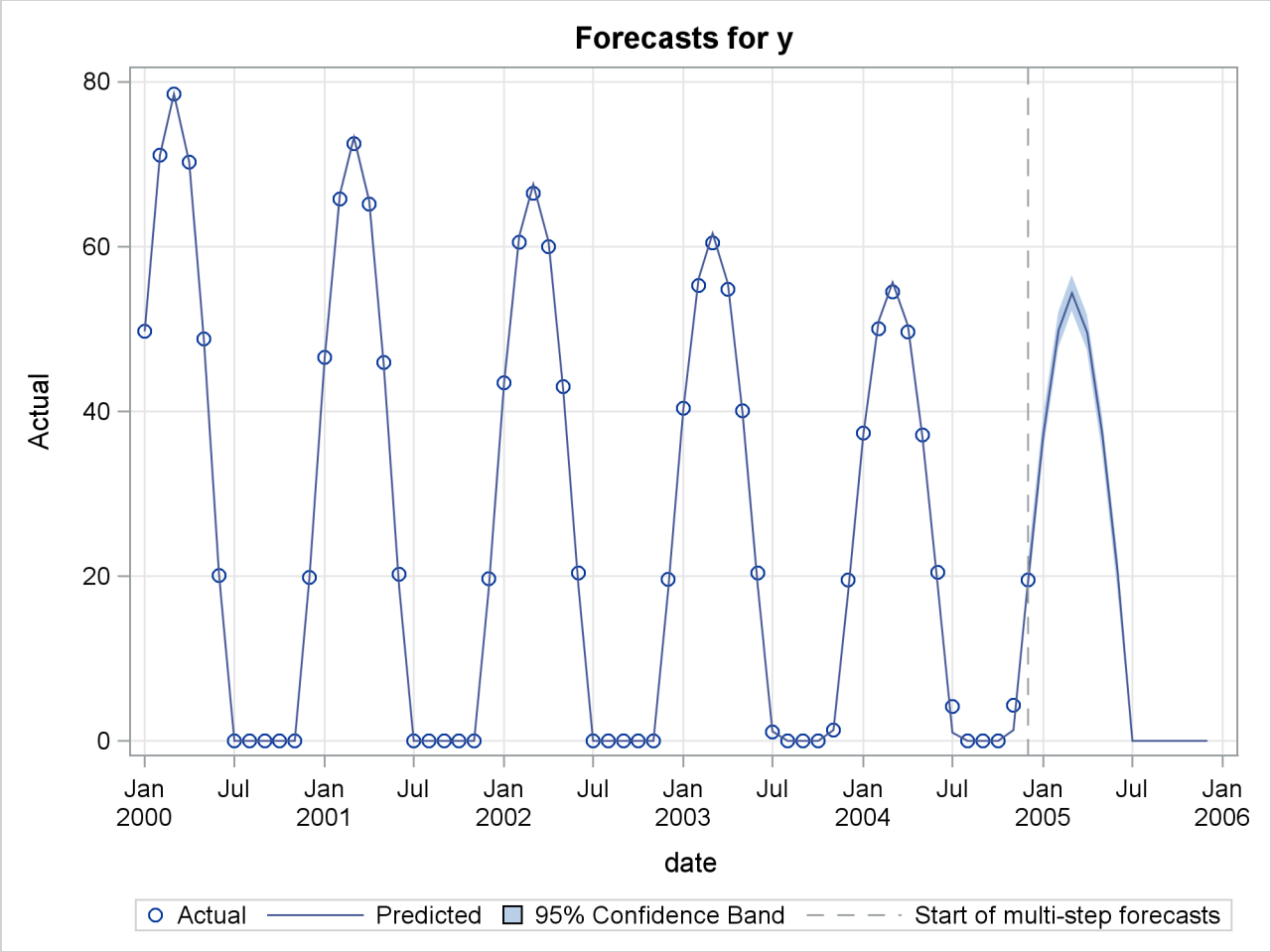



The seasonal indicator variable inseason can be used to fix this situation. Division by zero in the adjustment operation replaces the dividend by a missing value. Subsequent multiplication by zero sets the dividend to zero, especially helpful in the forecast horizon.

Running the HPFENGINE procedure again with the adjustment statement, as follows, produces the forecast shown in Output 7.6.2.

```
proc hpfengine data=test
               print=(select estimates)
               plot=forecasts;
   id date interval=month;
   forecast y;
   adjust y=(inseason) / operation=(divide, multiply);
run;
```

**Output 7.6.2** Forecasts




## Example 7.7: Multiple Repositories

This example demonstrates how to use the HPFENGINE procedure to access model specifications and model selection lists in multiple repositories.

In the following statements, three model specifications are created, one in SASUSER.REPOS1 and the other two in SASUSER.REPOS2.

```
proc hpfarimaspec repository=sasuser.repos1 name=myarima;
   forecast symbol=y transform=log q=(1 12) diflist=(1 12) noint;
run;

proc hpfucmspec repository=sasuser.repos2 name=myucm;
   forecast symbol=y transform=log;
   irregular;
```

```
   level;
   slope;
   season length=12;
run;

proc hpfesmspec repository=sasuser.repos2 name=mywinters;
   esm method=winters transform=log;
run;
```

A selection list is then created by using the HPFSELECT procedure and placed in the repository named SASUSER.REPOS3, as follows.

```
proc hpfselect repository=sasuser.repos3 name=mylist;
   spec myarima myucm mywinters;
run;
```

Because the HPFENGINE procedure has only one REPOSITORY option, it would not normally be possible for the procedure to locate the selection list and all three models. The CATNAME statement is used as follows to create a logical concatenation of multiple catalogs; this offers the solution to this problem.

```
catname sasuser.cataloglist (sasuser.repos1 sasuser.repos2 sasuser.repos3);
```

When the HPFENGINE procedure is called, the concatenated name is used in the REPOSITORY= option, as follows. The results of the model selection are shown in Output 7.7.1.

```
proc hpfengine data=sashelp.air
               repository=sasuser.cataloglist
               globalselection=mylist
               print=select;
   id date interval=month;
   forecast air;
run;
```

**Output 7.7.1** Model Selection Results

**The HPFENGINE Procedure**

| Model Selection Criterion = MAPE | | | |
|---|---|---|---|
| **Model** | **Statistic** | **Selected** | **Label** |
| **MYARIMA** | 2.9672282 | No | ARIMA: Log( Y ) ~ D = (1,12) Q = (1,12) NOINT |
| **MYUCM** | 3.2601060 | No | UCM: Log( Y ) = TREND + SEASON + ERROR |
| **MYWINTERS** | 2.7138783 | Yes | Log Winters Method (Multiplicative) |

## Example 7.8: ODS Graphics

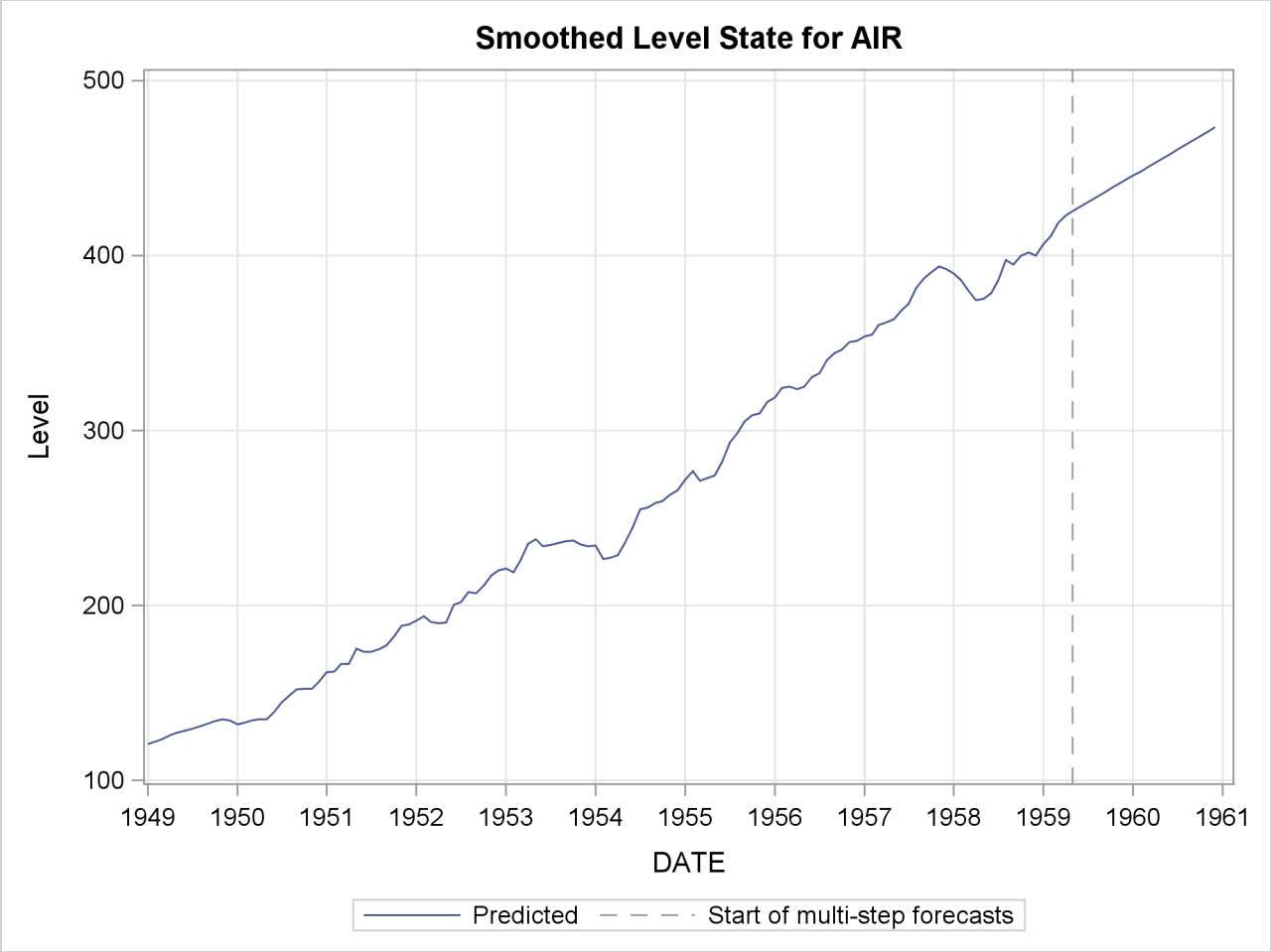
This example illustrates the use of ODS Graphics. The following statements use the SASHELP.AIR data set to automatically forecast the time series of international airline travel.

The graphical displays are requested by specifying the PLOT= option in the PROC HPFENGINE statement. In this case, all plots are requested. Figures Output 7.8.1 through Output 7.8.5 show a selection of the plots created.

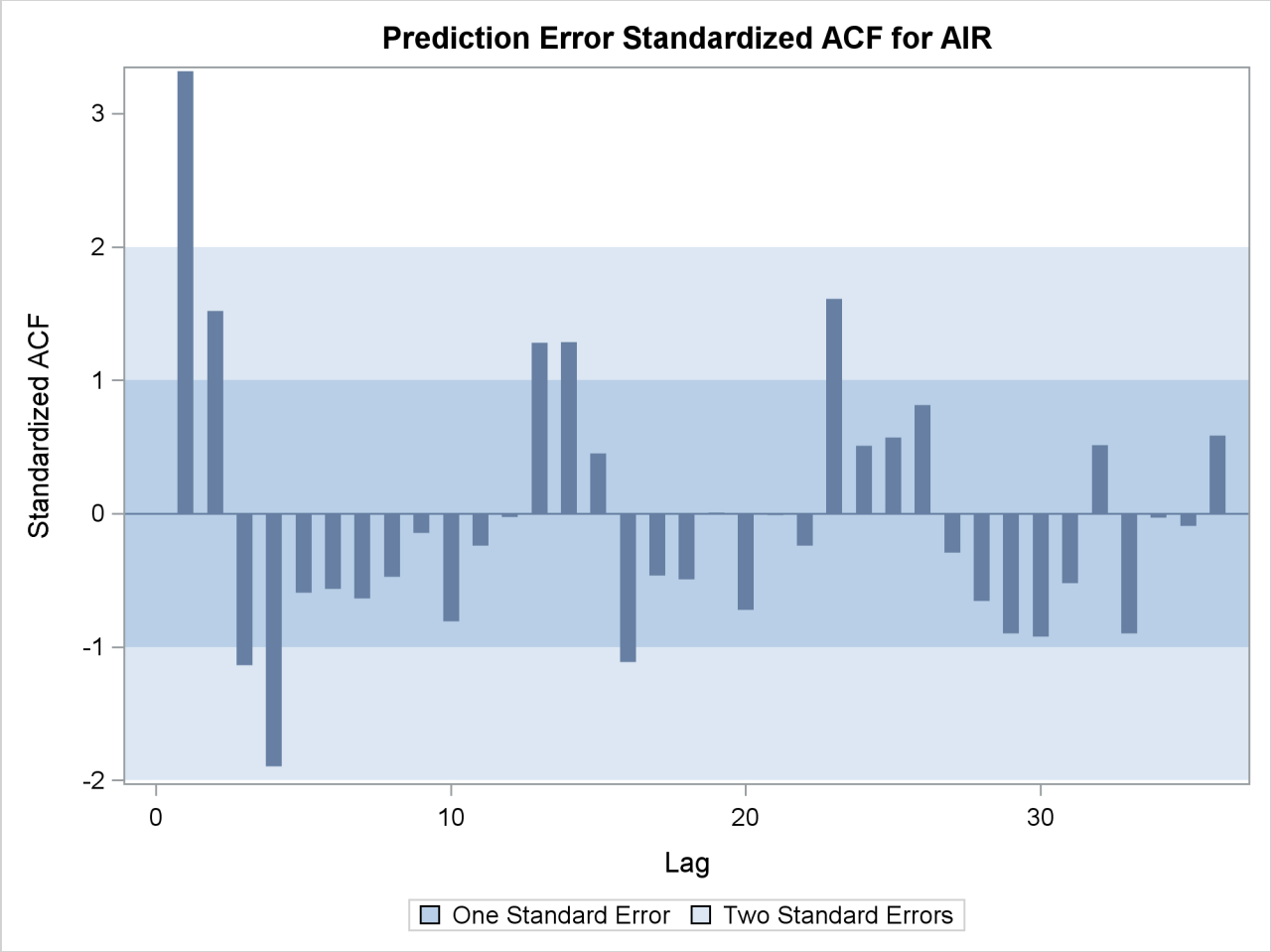
For information about the graphics available in the HPFENGINE procedure, see the "ODS Graphics" on page 231 section.

```
proc hpfengine data=sashelp.air
               out=_null_
               lead=20
               back=20
               plot=all;
   id date interval=month;
   forecast air;
run;
```

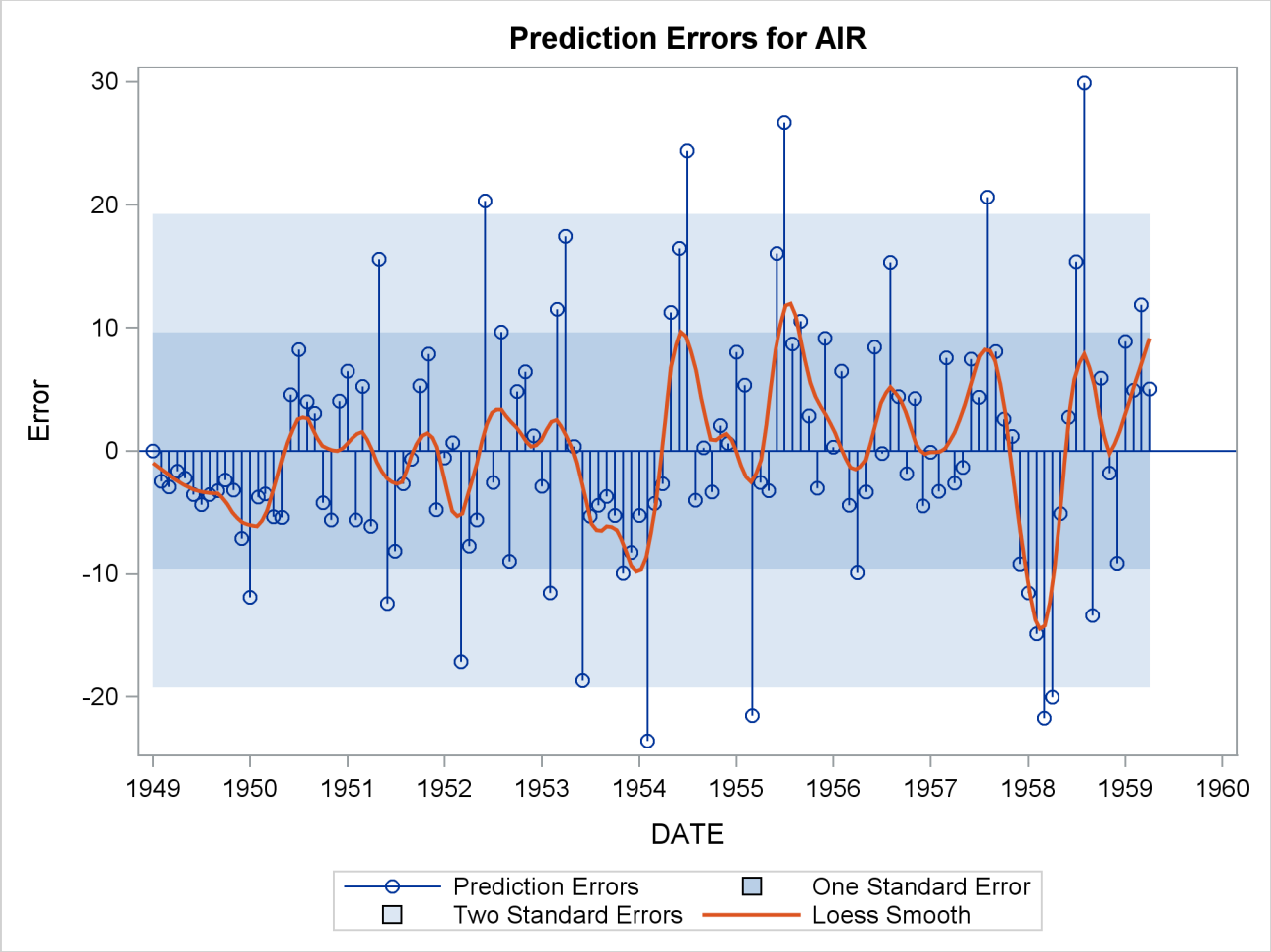
**Output 7.8.1** Component Estimates



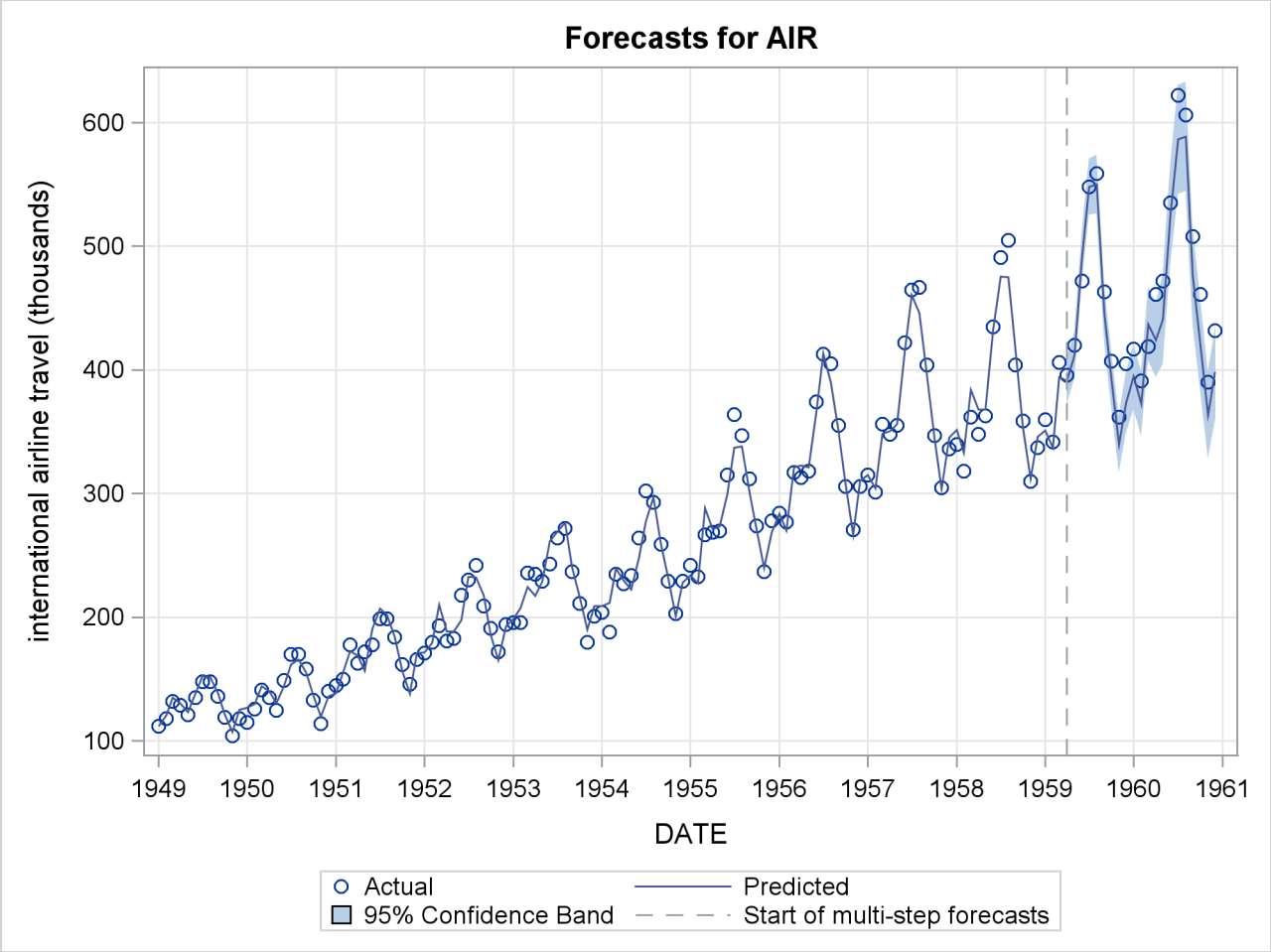
**Output 7.8.2** Standardized Autocorrelation of Prediction Errors



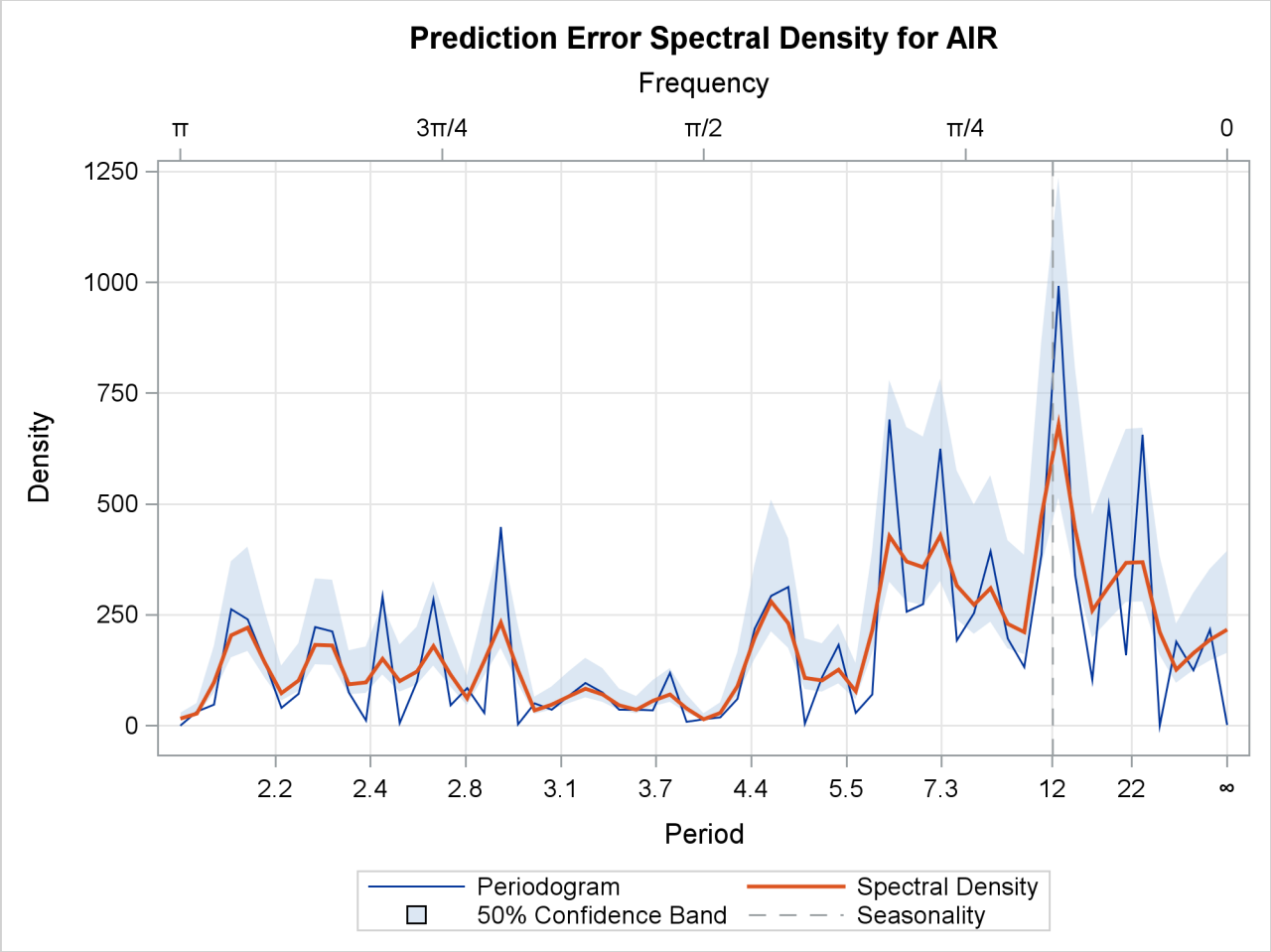
**Output 7.8.3** Prediction Errors



**Output 7.8.4** Forecasts



**Output 7.8.5** Prediction Error Spectral Density




## Example 7.9: Combining Judgmental Forecasts

Externally specified models enable forecasts from external sources to be included in the model selection process in the HPFENGINE procedure. Because ensemble models operate on forecasts, you might naturally expect a combination of forecasts from externally specified models to function in this setting. This simple example demonstrates such a scenario.

For the data set Sashelp.Pricedata, suppose you have sales forecasts that arise from the judgment of three independent sources. Suppose you want to compare a weighted combination of those judgmental forecasts against the sales forecast from the best exponential smoothing model (BESTESM), where the weighted combination assigns predetermined weights to the combination candidates. This example assumes that the external forecasts are contained in a separate data set, Perm.External, where the variables Salef1-Salef3 denote the three judgmental forecasts to be combined. This example also demonstrates the use of the AUXDATA= option for supporting auxiliary data set access.

First, you create three separate external models to identify each of the combined forecast contributors (named XF1, XF2, and XF3). Then you combine those with the combined model list COMBXF. Combination weights of 0.5, 0.3, and 0.2 are assigned to XF1, XF2, and XF3, respectively. You must map each external

model specification to the data set variable that supplies its forecast. Finally, you create the model selection list XFSELECT that uses MAPE to define the comparison between BESTESM and COMBXF over a holdout region of six months:

```
proc hpfesmspec rep=work.rep specname=bestesm;
   esm method=best;
run;

proc hpfexmspec rep=work.rep specname=xf1;
  exm;
run;

proc hpfexmspec rep=work.rep specname=xf2;
  exm;
run;

proc hpfexmspec rep=work.rep specname=xf3;
  exm;
run;

proc hpfselect rep=work.rep name=combxf label="USERDEF(XF1,XF2,XF3)";
   combine method=userdef(0.5,0.3,0.2);
   spec xf1 /exmmap(predict=salef1);
   spec xf2 /exmmap(predict=salef2);
   spec xf3 /exmmap(predict=salef3);
run;

proc hpfselect rep=work.rep name=xfselect;
   select holdout=6 criterion=mape;
   spec combxf bestesm;
run;
```

You now run the HPFENGINE procedure on the primary data set Sashelp.Pricedata with the external forecasts in Perm.External. PROC HPFENGINE constructs the forecast model selection graph as directed by the GLOBALSELECTION= option and runs it for each BY group in Sashelp.Pricedata. PROC HPFENGINE selects the best forecast for each BY group independently from the candidates that are evaluated in the forecast model selection graph.

```
proc hpfengine data=sashelp.pricedata
               auxdata=perm.external
               rep=work.rep
               globalselection=xfselect
               out=out3 outfor=for3
               outmodelinfo=minfo3
               outstatselect=oss3
               outstat=stat3
               outest=est3
               lead=12;
   by region line product;
   id date interval=month horizonstart='01jan2003'd end='01dec2003'd;
   forecast sale;
   external salef1 salef2 salef3;
run;
```

Sashelp.Pricedata contains 17 BY groups for the hierarchy Region, Line, Product. Figure 7.9.1 shows a summary of the MAPE statistics for the selected models over the BY groups on the basis of whether a combined judgmental forecast was selected.

**Output 7.9.1** MAPE Summary Statistics for Combined versus Best ESM

**Combined judgment vs. BESTESM**

**The MEANS Procedure**

**Analysis Variable : MAPE Mean Absolute Percent Error**

| CombSelected | N Obs | N | Mean | Std Dev | Minimum | Maximum |
|---|---|---|---|---|---|---|
| NO | 11 | 11 | 6.5339736 | 2.4822563 | 3.4681708 | 11.6676137 |
| YES | 6 | 6 | 6.1907580 | 2.2840940 | 4.3111761 | 9.9540557 |

# References

Box, G. E. P., Jenkins, G. M., and Reinsel, G. C. (1994), *Time Series Analysis: Forecasting and Control*, 3rd Edition, Englewood Cliffs, NJ: Prentice-Hall.

# Chapter 8
# The HPFESMSPEC Procedure

**Contents**


Overview: HPFESMSPEC Procedure . . . . . **261**
Getting Started: HPFESMSPEC Procedure . . . . . **261**
Syntax: HPFESMSPEC Procedure . . . . . **262**
Functional Summary . . . . . 262
PROC HPFESMSPEC Statement . . . . . 263
ESM Statement . . . . . 263
Details: HPFESMSPEC Procedure . . . . . **267**
Smoothing Model Parameter Specification Options . . . . . 267
Examples: HPFESMSPEC Procedure . . . . . **267**
Example 8.1: Various Kinds of ESM Model Specifications . . . . . 267
Example 8.2: Selecting the Best ESM Model . . . . . 271


## Overview: HPFESMSPEC Procedure

The HPFESMSPEC procedure creates model specifications files for exponential smoothing models (ESM).

You can specify many types of exponential models with this procedure. In particular, any model that can be analyzed using the HPF procedure can be specified.

## Getting Started: HPFESMSPEC Procedure

The following example shows how to create an exponential smoothing model specification file. In this example, a model specification for a Winters method is created.

```
proc hpfesmspec repository=sasuser.mymodels
                name=mywinters
                label="Winters Method";
   esm method=winters;
run;
```

The options in the PROC HPESMSPEC statement are used to specify the location of the specification file that will be output. Here the REPOSITORY= option specifies that the output file be placed in a catalog "sasuser.mymodels,", the NAME= option specifies that the name of the file be "mywinters.xml," and the LABEL= option specifies a label for this catalog member. The ESM statement in the procedure specifies the exponential smoothing model and the options used to control the parameter estimation process for the model.

# Syntax: HPFESMSPEC Procedure

The following statements are used with the HPFESMSPEC procedure.

**PROC HPFESMSPEC** *options* **;**
**ESM** *options* **;**

## Functional Summary

Table 8.1 summarizes the statements and options that control the HPFESMSPEC procedure.

**Table 8.1** HPFESMSPEC Functional Summary

| Description | Statement | Option |
|---|---|---|
| **Statements** | | |
| specifies the exponential smoothing model | ESM | |
| **Model Repository Options** | | |
| Specifies the model repository | PROC HPFESMSPEC | REPOSITORY= |
| Specifies the model specification name | PROC HPFESMSPEC | NAME= |
| Specifies the model specification label | PROC HPFESMSPEC | LABEL= |
| **Exponential Smoothing Model Options** | | |
| Specifies the model selection criterion | ESM | CRITERION= |
| Specifies the damping weight parameter initial value | ESM | DAMPPARM= |
| Specifies the damping weight parameter restrictions | ESM | DAMPREST= |
| Specifies the level weight parameter initial value | ESM | LEVELPARM= |
| Specifies the level weight parameter restrictions | ESM | LEVELREST= |
| Specifies median forecasts | ESM | MEDIAN |
| Specifies the time series forecasting model | ESM | METHOD= |
| Specifies that the smoothing model parameters are fixed values | ESM | NOEST |
| Specifies that stable parameter estimates are not required | ESM | NOSTABLE |
| Specifies the season weight parameter initial value | ESM | SEASONPARM= |
| Specifies the season weight parameter restrictions | ESM | SEASONREST= |
| Specifies the time series transformation | ESM | TRANSFORM= |

| Description | Statement | Option |
|---|---|---|
| Specifies the trend weight parameter initial value | ESM | TRENDPARM= |
| Specifies the trend weight parameter restrictions | ESM | TRENDREST= |

## PROC HPFESMSPEC Statement

**PROC HPFESMSPEC** *options* **;**

The following options can be used in the PROC HPFESMSPEC statement.

**LABEL=***"specification label"*

**SPECLABEL=***"specification label"*

specifies a descriptive label for the model specification to be stored in the SAS catalog or external file reference. The LABEL= option can also be specified as SPECLABEL=.

**NAME=***SAS-name*

**SPECNAME=***SAS-name*

names the model specification to be stored in the SAS catalog or external file reference. The NAME= option can also be specified as SPECNAME=.

**REPOSITORY=***SAS-catalog-name* | *SAS-file-reference*

names the SAS catalog or external file reference to contain the model specification. The REPOSITORY= option can also be specified as MODELREPOSITORY=, MODELREP=, or REP=.

## ESM Statement

**ESM** *options* **;**

The ESM statement is used to specify an exponential smoothing model.

The default specification selects the best exponential smoothing model without transformation (METHOD=BEST TRANSFORM=NONE).

The following options can be specified in the ESM statement.

**CRITERION=***option*

**SELECT=***option*

specifies the model selection criterion (statistic of fit) to be used to select from several candidate models. This option is often used in conjunction with the HOLDOUT= option. The CRITERION= option can also be specified as SELECT=. The default is CRITERION=RMSE.

The following list shows the valid values for the CRITERION= option and the statistics of fit these option values specify:

| | |
|---|---|
| SSE | sum of squared error |
| MSE | mean squared error |
| RMSE | root mean squared error |
| UMSE | unbiased mean squared error |
| URMSE | unbiased root mean squared error |
| MAXPE | maximum percent error |
| MINPE | minimum percent error |
| MPE | mean percent error |
| MAPE | mean absolute percent error |
| MDAPE | median absolute percent error |
| GMAPE | geometric mean absolute percent error |
| MAPES | mean absolute error percent of standard deviation |
| MDAPES | median absolute error percent of standard deviation |
| GMAPES | geometric mean absolute error percent of standard deviation |
| MINPPE | minimum predictive percent error |
| MAXPPE | maximum predictive percent error |
| MPPE | mean predictive percent error |
| MAPPE | symmetric mean absolute predictive percent error |
| MDAPPE | median absolute predictive percent error |
| GMAPPE | geometric mean absolute predictive percent error |
| MINSPE | minimum symmetric percent error |
| MAXSPE | maximum symmetric percent error |
| MSPE | mean symmetric percent error |
| SMAPE | symmetric mean absolute percent error |
| MDASPE | median absolute symmetric percent error |
| GMASPE | geometric mean absolute symmetric percent error |
| MINRE | minimum relative error |
| MAXRE | maximum relative error |
| MRE | mean relative error |
| MRAE | mean relative absolute error |
| MDRAE | median relative absolute error |
| GMRAE | geometric mean relative absolute error |
| MAXERR | maximum error |
| MINERR | minimum error |
| ME | mean error |

| | |
|---|---|
| MAE | mean absolute error |
| MASE | mean absolute scaled error |
| RSQUARE | R-square |
| ADJRSQ | adjusted R-square |
| AADJRSQ | Amemiya's adjusted R-square |
| RWRSQ | random walk R-square |
| AIC | Akaike information criterion |
| AICC | finite sample corrected AIC |
| SBC | Schwarz Bayesian information criterion |
| APC | Amemiya's prediction criterion |

**DAMPPARM=***number*

specifies the damping weight parameter initial value. See the section "Smoothing Model Parameter Specification Options" on page 267.

**DAMPREST=(***number***,** *number* **)**

specifies the damping weight parameter restrictions. See the section "Smoothing Model Parameter Specification Options" on page 267.

**LEVELPARM=***number*

specifies the level weight parameter initial value. See the section "Smoothing Model Parameter Specification Options" on page 267.

**LEVELREST=(***number***,** *number* **)**

specifies the level weight parameter restrictions. See the section "Smoothing Model Parameter Specification Options" on page 267.

**MEDIAN**

specifies that the median forecast values be estimated. Forecasts can be based on the mean or median. By default the mean value is provided. If no transformation is applied to the actual series with the TRANSFORM= option, the mean and median time series forecast values are identical.

**METHOD=***method-name*

specifies the forecasting model to be used to forecast the time series. A single model can be specified, or a group of candidate models can be specified. If a group of models is specified, the model used to forecast the accumulated time series is selected based on the CRITERION= option of the ESM statement and the HOLDOUT= option of the FORECAST statement. The default is METHOD=BESTN. The following forecasting models are provided:

| | |
|---|---|
| SIMPLE | simple (single) exponential smoothing |
| DOUBLE | double (Brown) exponential smoothing |
| LINEAR | linear (Holt) exponential smoothing |
| DAMPTREND | damped trend exponential smoothing |
| ADDSEASONAL | additive seasonal exponential smoothing |

| | |
|---|---|
| MULTSEASONAL | multiplicative seasonal exponential smoothing |
| SEASONAL | same as ADDSEASONAL |
| WINTERS | Winters multiplicative Method |
| ADDWINTERS | Winters additive Method |
| BEST | best candidate smoothing model (SIMPLE, LINEAR, DAMPTREND), (ADDSEASONAL, ADDWINTERS, WINTERS) |
| BESTN | best candidate nonseasonal smoothing model (SIMPLE, LINEAR, DAMPTREND) |
| BESTS | best candidate seasonal smoothing model (ADDSEASONAL, ADDWINTERS, WINTERS) |

**NOEST**
specifies that the smoothing model parameters are fixed values. To use this option, all of the smoothing model parameters must be explicitly specified. By default, the smoothing model parameters are optimized.

**NOSTABLE**
specifies that the smoothing model parameters are not restricted to the additive invertible region of the parameter space. By default, the smoothing model parameters are restricted to be inside the additive invertible region of the parameter space.

**SEASONPARM=***number*
specifies the season weight parameter initial value. See the section "Smoothing Model Parameter Specification Options" on page 267.

**SEASONREST=(***number***,** *number* **)**
specifies the season weight parameter restrictions. See the section "Smoothing Model Parameter Specification Options" on page 267.

**TRANSFORM=***option*
specifies the time series transformation to be applied to the time series. The following transformations are provided:

| | |
|---|---|
| NONE | no transformation. This option is the default. |
| LOG | logarithmic transformation |
| SQRT | square-root transformation |
| LOGISTIC | logistic transformation |
| BOXCOX(*n* ) | Box-Cox transformation with parameter number where number is between –5 and 5 |
| AUTO | Automatically choose between NONE and LOG based on model selection criteria. |

When the TRANSFORM= option is specified, the time series must be strictly positive. Once the time series is transformed, the model parameters are estimated using the transformed time series. The forecasts of the transformed time series are then computed, and finally the transformed time series forecasts are inverse transformed. The inverse transform produces either mean or median forecasts depending on whether the MEDIAN option is specified.

**TRENDPARM=***number*
specifies the trend weight parameter initial value. See the section "Smoothing Model Parameter Specification Options" on page 267.

**TRENDREST=(***number***,** *number* **)**
specifies the trend weight parameter restrictions. See the section "Smoothing Model Parameter Specification Options" on page 267.

# Details: HPFESMSPEC Procedure

## Smoothing Model Parameter Specification Options

The parameter options are used to specify smoothing model parameters. If the parameter restrictions are not specified, the default is (0.0001 0.9999), which implies that the parameters are restricted between 0.0001 and 0.9999. Parameters and their restrictions are required to be greater than or equal to –1 and less than or equal to 2. Missing values indicate no lower and/or upper restriction. If the parameter initial values are not specified, the optimizer uses a grid search to find an appropriate initial value.

# Examples: HPFESMSPEC Procedure

## Example 8.1: Various Kinds of ESM Model Specifications

The following statements illustrate typical uses of the ESM statement:

```
proc hpfesmspec repository=mymodels
                name=model1
                label="Default Specification";
   esm;
run;

proc hpfesmspec repository=mymodels
                name=model2
                label="Simple Exponential Smoothing";
   esm method=simple;
run;

proc hpfesmspec repository=mymodels
                name=model3
                label="Double Exponential Smoothing";
   esm method=double;
run;
```

```
proc hpfesmspec repository=mymodels
                name=model4
                label="Linear Exponential Smoothing";
   esm method=linear;
run;

proc hpfesmspec repository=mymodels
                name=model5
                label="Damp-Trend Exponential Smoothing";
   esm method=damptrend;
run;

proc hpfesmspec repository=mymodels
                name=model6
                label="Seasonal Exponential Smoothing";
   esm method=addseasonal;
run;

proc hpfesmspec repository=mymodels
                name=model7
                label="Winters Method";
   esm method=winters;
run;

proc hpfesmspec repository=mymodels
                name=model8
                label="Additive-Winters Method";
   esm method=addwinters;
run;

proc hpfesmspec repository=mymodels
                name=model9
                label="Best Smoothing Model";
   esm method=best;
run;

proc hpfesmspec repository=mymodels
                name=model10
                label="Best Non-Seasonal Smoothing Model";
   esm method=bestn;
run;

proc hpfesmspec repository=mymodels
                name=model11
                label="Best Seasonal Smoothing Model";
   esm method=bests;
run;

proc hpfesmspec repository=mymodels
                name=model12
                label="Log Simple Exponential Smoothing";
   esm method=simple transform=log;
run;
```

```
proc hpfesmspec repository=mymodels
                name=model13
                label="Log Double Exponential Smoothing";
   esm method=double transform=log;
run;

proc hpfesmspec repository=mymodels
                name=model14
                label="Log Linear Exponential Smoothing";
   esm method=linear transform=log;
run;

proc hpfesmspec repository=mymodels
                name=model15
                label="Log Damp-Trend Exponential Smoothing";
   esm method=damptrend transform=log;
run;

proc hpfesmspec repository=mymodels
                name=model16
                label="Log Seasonal Exponential Smoothing";
   esm method=addseasonal transform=log;
run;

proc hpfesmspec repository=mymodels
                name=model16
                label="Log Winters Method";
   esm method=winters transform=log;
run;

proc hpfesmspec repository=mymodels
                name=model18
                label="Log Additive-Winters Method";
   esm method=addwinters transform=log;
run;

proc hpfesmspec repository=mymodels
                name=model19
                label="Best Log Smoothing Model";
   esm method=best transform=log;
run;

proc hpfesmspec repository=mymodels
                name=model20
                label="Best Log Non-Seasonal Smoothing Model";
   esm method=bestn transform=log;
run;

proc hpfesmspec repository=mymodels
                name=model21
                label="Best Log Seasonal Smoothing Model";
   esm method=bests transform=log;
run;
```

```
title "Models Added to MYMODELS Repository";
proc catalog catalog=mymodels;
   contents;
run;
```

**Output 8.1.1** Listing of Models in MYMODELS Repository

**Models Added to MYMODELS Repository**

**Contents of Catalog WORK.MYMODELS**

| # | Name | Type | Create Date | Modified Date | Description |
|---|---|---|---|---|---|
| 1 | MODEL1 | XML | 03/26/2014 08:16:12 | 03/26/2014 08:16:12 | Default Specification |
| 2 | MODEL10 | XML | 03/26/2014 08:16:12 | 03/26/2014 08:16:12 | Best Non-Seasonal Smoothing Model |
| 3 | MODEL11 | XML | 03/26/2014 08:16:12 | 03/26/2014 08:16:12 | Best Seasonal Smoothing Model |
| 4 | MODEL12 | XML | 03/26/2014 08:16:12 | 03/26/2014 08:16:12 | Log Simple Exponential Smoothing |
| 5 | MODEL13 | XML | 03/26/2014 08:16:12 | 03/26/2014 08:16:12 | Log Double Exponential Smoothing |
| 6 | MODEL14 | XML | 03/26/2014 08:16:12 | 03/26/2014 08:16:12 | Log Linear Exponential Smoothing |
| 7 | MODEL15 | XML | 03/26/2014 08:16:12 | 03/26/2014 08:16:12 | Log Damp-Trend Exponential Smoothing |
| 8 | MODEL16 | XML | 03/26/2014 08:16:12 | 03/26/2014 08:16:12 | Log Winters Method |
| 9 | MODEL18 | XML | 03/26/2014 08:16:12 | 03/26/2014 08:16:12 | Log Additive-Winters Method |
| 10 | MODEL19 | XML | 03/26/2014 08:16:12 | 03/26/2014 08:16:12 | Best Log Smoothing Model |
| 11 | MODEL2 | XML | 03/26/2014 08:16:12 | 03/26/2014 08:16:12 | Simple Exponential Smoothing |
| 12 | MODEL20 | XML | 03/26/2014 08:16:12 | 03/26/2014 08:16:12 | Best Log Non-Seasonal Smoothing Model |
| 13 | MODEL21 | XML | 03/26/2014 08:16:12 | 03/26/2014 08:16:12 | Best Log Seasonal Smoothing Model |
| 14 | MODEL3 | XML | 03/26/2014 08:16:12 | 03/26/2014 08:16:12 | Double Exponential Smoothing |
| 15 | MODEL4 | XML | 03/26/2014 08:16:12 | 03/26/2014 08:16:12 | Linear Exponential Smoothing |
| 16 | MODEL5 | XML | 03/26/2014 08:16:12 | 03/26/2014 08:16:12 | Damp-Trend Exponential Smoothing |
| 17 | MODEL6 | XML | 03/26/2014 08:16:12 | 03/26/2014 08:16:12 | Seasonal Exponential Smoothing |
| 18 | MODEL7 | XML | 03/26/2014 08:16:12 | 03/26/2014 08:16:12 | Winters Method |
| 19 | MODEL8 | XML | 03/26/2014 08:16:12 | 03/26/2014 08:16:12 | Additive-Winters Method |
| 20 | MODEL9 | XML | 03/26/2014 08:16:12 | 03/26/2014 08:16:12 | Best Smoothing Model |

## Example 8.2: Selecting the Best ESM Model

This example illustrates how to define model specifications that automatically choose the best exponential smoothing model by using MAPE as the model selection criterion.

The following statements fit two forecast models (simple and log simple exponential smoothing) to the time series. The forecast model that results in the lowest MAPE is used to forecast the time series.

```
proc hpfesmspec repository=mymodels
                name=best_simple;
   esm method=simple transform=auto criterion=mape;
run;
```

The following statements fit two forecast models (seasonal and log seasonal exponential smoothing) to the time series. The forecast model that results in the lowest MAPE is used to forecast the time series.

```
proc hpfesmspec repository=mymodels
                name=best_seasonal;
   esm method=addseasonal transform=auto criterion=mape;
run;
```

The following statements fit 14 forecasting models (best and log best exponential smoothing) to the time series. The forecast model that results in the lowest MAPE is used to forecast the time series.

```
proc hpfesmspec repository=mymodels
                name=best;
   esm method=best transform=auto criterion=mape;
run;
```

# Chapter 9
# The HPFEVENTS Procedure

**Contents**


Overview: HPFEVENTS Procedure . . . . . **273**
Getting Started: HPFEVENTS Procedure . . . . . **275**
Syntax: HPFEVENTS Procedure . . . . . **279**
Functional Summary . . . . . 280
PROC HPFEVENTS Statement . . . . . 281
BY Statement . . . . . 281
EVENTCOMB Statement . . . . . 282
EVENTDATA Statement . . . . . 282
EVENTDEF Statement . . . . . 283
EVENTDUMMY Statement . . . . . 285
EVENTGROUP Statement . . . . . 286
EVENTKEY Statement . . . . . 287
ID Statement . . . . . 287
VAR Statement . . . . . 289
Details: HPFEVENTS Procedure . . . . . **289**
Event Definitions . . . . . 289
Using the EVENTKEY Statement . . . . . 299
Using the EVENTDS= System Option to Customize Holiday Predefined Date Keywords 302
Missing Value Interpretation . . . . . 303
Data Set Input/Output . . . . . 303
Printed Output . . . . . 305
Examples: HPFEVENTS Procedure . . . . . **305**
Example 9.1: Multiple Timing Values in a Single Event versus Using Multiple Events and the EVENTCOMB Statement . . . . . 305
Example 9.2: Using a DATA Step to Construct an Events Data Set . . . . . 307
Example 9.3: Preparing a Data Set for PROC HPFENGINE . . . . . 310
Example 9.4: Using Predefined SAS Event Keywords Directly in Other SAS Procedures 311
Example 9.5: Viewing Dummy Variables by Using the SGPLOT Procedure . . . . . 312
References . . . . . **315**


## Overview: HPFEVENTS Procedure

The HPFEVENTS procedure provides a way to create and manage events associated with time series for the purpose of analysis. The procedure can create events, read events from an events data set, write events to an events data set, and create dummy variables based on those events if date information is provided.

A SAS event is used to model any incident that disrupts the normal flow of the process that generated the time series. Examples of commonly used events include natural disasters, retail promotions, strikes, advertising campaigns, policy changes, and data recording errors.

An event has a reference name, a date or dates associated with the event, and a set of qualifiers. The event exists separate from any time series; however, the event can be applied to one or more time series. When the event is applied to a time series, a dummy variable is generated that can be used to analyze the impact of the event on the time series. You can use the HPFEVENTS procedure to apply an event or events to a time series, create one or more dummy variables, and save the dummy variables in a data set. However, it is not necessary to do this if the HPFENGINE or HPFDIAGNOSE procedure will be used to evaluate the time series and the events. PROC HPFENGINE and PROC HPFDIAGNOSE create and store the dummy variables in memory for you based on the definition created with PROC HPFEVENTS. You only need to supply the event definition data set created with the EVENTDATA OUT= statement to PROC HPFENGINE or PROC HPFDIAGNOSE.

There are many advantages of using PROC HPFEVENTS:

- Dummy variables generated by PROC HPFEVENTS are automatically extended, shortened, or changed as observations are added and deleted from a time series. Thus, a single EVENT definition can be used for several time series or for different spans of the same series.
- PROC HPFEVENTS can be used to define dummy variables that function equally well for time series of various intervals, such as weekly or monthly data. The same EVENT definition can model daily data or weekly totals.
- EVENT definitions can be stored in a data set. EVENT definitions can later be changed, new EVENTS added, or additional dummy variables generated from an existing data set.
- EVENT definitions stored in a data set can be passed directly to PROC HPFENGINE and PROC HPFDIAGNOSE. See Chapter 7, "The HPFENGINE Procedure," and Chapter 5, "The HPFDIAGNOSE Procedure," for details.
- SAS predefined EVENT definitions can be accessed directly from PROC HPFENGINE and PROC HPFDIAGNOSE. See the section "EVENTKEY Statement" on page 287 for a list of SAS predefined EVENT definitions. Example 9.4 illustrates this feature.
- PROC HPFEVENTS can generate a data set that can be used in other procedures such as PROC REG. See Chapter 85, "The REG Procedure" (*SAS/STAT User's Guide*). As data are added or deleted from the time series, PROC HPFEVENTS can automatically generate new dummy variables as required.
- PROC HPFEVENTS recognizes predefined variables and dates. Thus, events that involve holidays such as Easter and Thanksgiving can be modeled easily, even though the dates of the events change from year to year.

# Getting Started: HPFEVENTS Procedure

The HPFEVENTS procedure is simple to use. It provides results in output data sets that can be interpreted in other SAS procedures.

The following example creates events and dummy variables. Then it outputs the event definitions to a data set named EVDSOUT1 and dummy variables to a data set named EVDUMOUT1. More examples are shown in the section "Examples: HPFEVENTS Procedure" on page 305.

```
proc hpfevents data=sashelp.air ;
   var air;
   id date interval=month end='31Dec1952'D;
   eventdef laborday=labor / value=2 ;
   eventdef summer= '01Jun1900'D to '01Jun2005'D by year /
                  after=(duration=2) label='jun jul aug';
   eventdef yr1950= '01Jan1950'D / pulse=year ;
   eventdef levelshift= '01Jan1950'D / type=ls ;
   eventdef novdec= christmas / before=(duration=1)
                              pulse=month;
   eventdef first10obs= 1 to 10;
   eventdef everyotherobs= 1 to 200 by 2;
   eventdef saturday= '01Jan1950'D to '31Jan1950'D by week.7;
   eventkey ao15obs;
   eventkey ls01Jan1950D / after=(duration=5) ;
   eventkey garbage ;
   eventdata  out=evdsout1(label='list of events');
   eventdummy  out=evdumout1(label='dummy variables');
run;
```

Features of PROC HPFEVENTS illustrated by the preceding statements are as follows.

LABORDAY — PROC HPFEVENTS recognizes that "LABOR" is a date keyword. When PROC HPFEVENTS creates a dummy variable for this event, a timing value is generated for each Labor Day that falls in the span of the time series. Each observation that matches the date of Labor Day has a value of 2.

SUMMER — The do-list, '01Jun1900'D to '01Jun2005'D by year, will generate 106 timing values on the first of June for each year from 1900 to 2005. The pulse for each timing value will last for 3 observations, the observation matching June 1st and the two following. For monthly data, this should generate a pulse for June, July, and August of each year from 1900 to 2005. If you add PULSE=MONTH to this statement, then the EVENT always specifies June, July, and August, regardless of the interval of the data. If you specify only the timing value '01Jun1900'D and add PERIOD=YEAR, then you have the same effect for all years, even years before 1900 and after 2005.

YR1950 By specifying a timing value within the year 1950 and using PULSE=YEAR, this event is a pulse for any observations within the year 1950.

LEVELSHIFT TYPE=LS has, by default, AFTER=(DURATION=ALL). The pulse begins at the observation that matches January 1, 1950, and continues to the end of the series. A special missing value of "A" is shown in the _DUR_AFTER_ variable of the events definition data set to represent AFTER=(DURATION=ALL).

NOVDEC Compare this to the SUMMER event. Here the date keyword CHRISTMAS is used. CHRISTMAS produces the same result as a timing value of '25Decyyyy'D and PERIOD=YEAR. BEFORE=(DURATION=1) and PULSE=MONTH specifies the months of November and December for any year in the span of the series. If the intent is to specify certain months of the year, this is preferable to the syntax used in SUMMER.

FIRST10OBS Integers in the timing list always specify observation numbers. This dummy variable is always a pulse from observation 1 to observation 10, regardless of the value of the timing ID variable.

EVERYOTHEROBS Like FIRST10OBS, this dummy variable always specifies every other observation starting at observation 1 and ending at observation 199.

SATURDAY WEEK.7 in the do-list specifies Saturday dates. The do-list produces timing values that are the Saturdays in January of 1950. If the data are daily, there are 4 pulses in January of 1950, one on each Saturday. If the data are weekly, a pulse is formed for 4 successive observations in January of 1950. If the data are monthly and RULE= ADD (which is the default), then the observation for January 1950 counts the number of Saturdays in January.

AO15OBS AO15OBS is recognized as an event keyword. AO15OBS is a predefined event that means a pulse placed on the 15th observation.

LS01JAN1950D LS01JAN1950D is recognized as an event keyword. LS01JAN1950D is a predefined event that means a level shift beginning at '01Jan1950'D. The qualifier AFTER=(DURATION=5) modifies the predefined event.

GARBAGE EVENTKEY GARBAGE is ignored because garbage is not an event keyword. A warning is printed to the log.

The following statements display the EVENTDATA output data set that is shown in Figure 9.1.

```
proc print data=evdsout1;
run;
```

**Figure 9.1** Event Definition Data Set Showing All Variables Related to Event Definitions

| Obs | _NAME_ | _CLASS_ | _KEYNAME_ | _STARTDATE_ | _ENDDATE_ | _DATEINTRVL_ | _STARTDT_ | _ENDDT_ |
|---|---|---|---|---|---|---|---|---|
| 1 | laborday | SIMPLE | LABOR | . | . | | . | . |
| 2 | summer | SIMPLE | . | 01JUN1900 | 01JUN2005 | YEAR.6 | . | . |
| 3 | yr1950 | SIMPLE | . | 01JAN1950 | . | | . | . |
| 4 | levelshift | SIMPLE | . | 01JAN1950 | . | | . | . |
| 5 | novdec | SIMPLE | CHRISTMAS | . | . | | . | . |
| 6 | first10obs | SIMPLE | . | . | . | | . | . |
| 7 | everyotherobs | SIMPLE | . | . | . | | . | . |
| 8 | saturday | SIMPLE | . | 07JAN1950 | 28JAN1950 | WEEK1.7 | . | . |
| 9 | ao15obs | SIMPLE | . | . | . | | . | . |
| 10 | ls01Jan1950D | SIMPLE | . | 01JAN1950 | . | | . | . |

| Obs | _DTINTRVL_ | _STARTOBS_ | _ENDOBS_ | _OBSINTRVL_ | _TYPE_ | _VALUE_ | _PULSE_ | _DUR_BEFORE_ |
|---|---|---|---|---|---|---|---|---|
| 1 | . | . | . | . | POINT | 2 | . | 0 |
| 2 | . | . | . | . | POINT | 1 | . | 0 |
| 3 | . | . | . | . | POINT | 1 | YEAR | 0 |
| 4 | . | . | . | . | LS | 1 | . | 0 |
| 5 | . | . | . | . | POINT | 1 | MONTH | 1 |
| 6 | . | 1 | 10 | 1 | POINT | 1 | . | 0 |
| 7 | . | 1 | 199 | 2 | POINT | 1 | . | 0 |
| 8 | . | . | . | . | POINT | 1 | . | 0 |
| 9 | . | 15 | . | . | POINT | 1 | . | 0 |
| 10 | . | . | . | . | LS | 1 | . | 0 |

| Obs | _DUR_AFTER_ | _SLOPE_BEF_ | _SLOPE_AFT_ | _SHIFT_ | _TCPARM_ | _RULE_ | _PERIOD_ | _LABEL_ |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | GROWTH | GROWTH | 0 | 0.5 | ADD | . | . |
| 2 | 2 | GROWTH | GROWTH | 0 | 0.5 | ADD | . | jun jul aug |
| 3 | 0 | GROWTH | GROWTH | 0 | 0.5 | ADD | . | . |
| 4 | A | GROWTH | GROWTH | 0 | 0.5 | ADD | . | . |
| 5 | 0 | GROWTH | GROWTH | 0 | 0.5 | ADD | . | . |
| 6 | 0 | GROWTH | GROWTH | 0 | 0.5 | ADD | . | . |
| 7 | 0 | GROWTH | GROWTH | 0 | 0.5 | ADD | . | . |
| 8 | 0 | GROWTH | GROWTH | 0 | 0.5 | ADD | . | . |
| 9 | 0 | GROWTH | GROWTH | 0 | 0.5 | ADD | . | . |
| 10 | 5 | GROWTH | GROWTH | 0 | 0.5 | ADD | . | . |

The following statements display the EVENTDUMMY output data set that is shown in Figure 9.2.

```
proc print data=evdumout1(obs=10);
run;
```

**Figure 9.2** Event Dummy Data Set

| Obs | DATE | AIR | laborday | summer | yr1950 | levelshift | novdec | first10obs | everyotherobs |
|---|---|---|---|---|---|---|---|---|---|
| 1 | JAN1949 | 112 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 2 | FEB1949 | 118 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 3 | MAR1949 | 132 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 4 | APR1949 | 129 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 5 | MAY1949 | 121 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 6 | JUN1949 | 135 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 7 | JUL1949 | 148 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 8 | AUG1949 | 148 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 9 | SEP1949 | 136 | 2 | 0 | 0 | 0 | 0 | 1 | 1 |
| 10 | OCT1949 | 119 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

| Obs | saturday | ao15obs | ls01Jan1950D |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 |

Often a set of EVENT definitions has only a few variables that apply. For instance, in the preceding example, no datetime timing values are specified, so the variables _STARTDT_, _ENDDT_, and _DTINTRVL_ have all missing values. In such a case, you can specify the CONDENSE option in the EVENTDATA statement, and PROC HPFEVENTS automatically determines if any variables in the EVENT definition data set contain only the default values; those variables are not included in the output data set. In addition to the missing datetime values, in the following example, the variables _SLOPE_BEF_, _SLOPE_AFT_, _TCPARM_, _RULE_, and _PERIOD_ also contain default values and are omitted from the condensed data set that is shown in Figure 9.3. When PROC HPFEVENTS reads a data set, any variables not in the data set are automatically set to the default value. Thus, it is not necessary to specify CONDENSE when you use the EVENTDATA IN= option. PROC HPFEVENTS automatically reads condensed data sets. For more details about the CONDENSE option, see "EVENTDATA Statement OUT= Data Set" on page 304.

```
proc hpfevents data=sashelp.air ;
   id date interval=month end='31Dec1952'D;
   eventdata  in= evdsout1;
   eventdata  out= evdsout2 condense;
run;

proc print data=evdsout2;
run;
```

**Figure 9.3** Event Definition Data Set in Condensed Format

| Obs | _NAME_ | _KEYNAME_ | _STARTDATE_ | _ENDDATE_ | _DATEINTRVL_ | _STARTOBS_ | _ENDOBS_ |
|---|---|---|---|---|---|---|---|
| 1 | laborday | LABOR | . | . | . | . | . |
| 2 | summer | . | 01JUN1900 | 01JUN2005 | YEAR.6 | . | . |
| 3 | yr1950 | . | 01JAN1950 | . | . | . | . |
| 4 | levelshift | . | 01JAN1950 | . | . | . | . |
| 5 | novdec | CHRISTMAS | . | . | . | . | . |
| 6 | first10obs | . | . | . | . | 1 | 10 |
| 7 | everyotherobs | . | . | . | . | 1 | 199 |
| 8 | saturday | . | 07JAN1950 | 28JAN1950 | WEEK1.7 | . | . |
| 9 | ao15obs | . | . | . | . | 15 | . |
| 10 | ls01Jan1950D | . | 01JAN1950 | . | . | . | . |

| Obs | _OBSINTRVL_ | _TYPE_ | _VALUE_ | _PULSE_ | _DUR_BEFORE_ | _DUR_AFTER_ | _LABEL_ |
|---|---|---|---|---|---|---|---|
| 1 | . | POINT | 2 | . | 0 | 0 | . |
| 2 | . | POINT | 1 | . | 0 | 2 | jun jul aug |
| 3 | . | POINT | 1 | YEAR | 0 | 0 | . |
| 4 | . | LS | 1 | . | 0 | A | . |
| 5 | . | POINT | 1 | MONTH | 1 | 0 | . |
| 6 | 1 | POINT | 1 | . | 0 | 0 | . |
| 7 | 2 | POINT | 1 | . | 0 | 0 | . |
| 8 | . | POINT | 1 | . | 0 | 0 | . |
| 9 | . | POINT | 1 | . | 0 | 0 | . |
| 10 | . | LS | 1 | . | 0 | 5 | . |

# Syntax: HPFEVENTS Procedure

The following statements are used with the HPFEVENTS procedure:

**PROC HPFEVENTS** < *options* > ;
**BY** *variables* ;
**EVENTCOMB** *variable= variable-list / options* ;
**EVENTDATA** *options* ;
**EVENTDEF** *variable= do-list / options* ;
**EVENTDUMMY** *options* ;
**EVENTGROUP** *variable=( list )* ;
**EVENTKEY** < *variable=* > *event-keyword* < */ options* > ;
**ID** *variable INTERVAL= interval options* ;
**VAR** *variables* ;

## Functional Summary

Table 9.1 summarizes the statements and options that control the HPFEVENTS procedure.

**Table 9.1** HPFEVENTS Functional Summary

| Description | Statement | Option |
|---|---|---|
| **Statements** | | |
| Specifies BY-group processing | BY | |
| Specifies event combination | EVENTCOMB | |
| Specifies event definition | EVENTDEF | |
| Specifies group of events | EVENTGROUP | |
| Uses a predefined event definition | EVENTKEY | |
| Specifies event data set | EVENTDATA | |
| Specifies dummy variable data set | EVENTDUMMY | |
| Specifies the time ID variable | ID | |
| Specifies the variables to be copied to the dummy variable data set | VAR | |
| **Data Set Options** | | |
| Specifies the input data set | PROC HPFEVENTS | DATA= |
| Specifies an events input data set | EVENTDATA | IN= |
| Specifies an events output data set | EVENTDATA | OUT= |
| Specifies that the events output data set be condensed | EVENTDATA | CONDENSE |
| Specifies a dummy variable output data set | EVENTDUMMY | OUT= |
| Specifies a starting time ID value | ID | START= |
| Specifies an ending time ID value | ID | END= |
| Specifies the format of the ID variable | ID | FORMAT= |
| **Dummy Variable Format Options** | | |
| Extends dummy variables past the end of the series | PROC HPFEVENTS | LEAD= |
| Specifies missing value interpretation | ID | SETMISSING= |
| Specifies the frequency of the dummy variables | ID | INTERVAL= |
| Specifies interval alignment | ID | ALIGN= |
| **Miscellaneous Options** | | |
| Specifies that variables in output data sets are in sorted order | PROC HPFEVENTS | SORTNAMES |
| Limits error and warning messages | PROC HPFEVENTS | MAXERROR= |

## PROC HPFEVENTS Statement

**PROC HPFEVENTS** *options* **;**

The following options can be used in the PROC HPFEVENTS statement.

**DATA=***SAS-data-set*
names the SAS data set that contains the variables used in the VAR, ID, and BY statements. If the DATA= option is not specified, the most recently created SAS data set is used.

**LEAD=***n*
specifies the number of periods to extend the dummy variable beyond the time series. The default is LEAD=0.

The LEAD= value is relative to the last observation in the input data set and not to the last nonmissing observation of a particular series. Thus, if a series has missing values at the end, the actual number of dummy values beyond the last nonmissing value will be greater than the LEAD= value.

**MAXERROR=***number*
limits the number of warning and error messages produced during the execution of the procedure to the specified value. The default is MAXERROR=25. This option is particularly useful in BY-group processing, where it can be used to suppress the recurring messages.

**SORTNAMES**
specifies that the events and variables in the output data sets be printed in alphabetical order.

## BY Statement

**BY** *variables* **;**

A BY statement can be used with PROC HPFEVENTS to obtain separate dummy variable definitions for groups of observations defined by the BY variables.

When a BY statement appears, the procedure expects the input data set to be sorted in order of the BY variables.

If your input data set is not sorted in ascending order, use one of the following alternatives:

- Sort the data by using the SORT procedure with a similar BY statement.
- Specify the NOTSORTED or DESCENDING option in the BY statement for the HPF procedure. The NOTSORTED option does not mean that the data are unsorted but rather that the data are arranged in groups (according to values of the BY variables) and that these groups are not necessarily in alphabetical or increasing numeric order.
- Create an index on the BY variables by using the DATASETS procedure.

For more information about the BY statement, see *SAS Language Reference: Concepts*. For more information about the DATASETS procedure, see the discussion in the *Base SAS Procedures Guide.*

## EVENTCOMB Statement

**EVENTCOMB** *variable= variable-list /options* **;**

An EVENTCOMB statement can be used with PROC HPFEVENTS to create a new event from one or more events that have previously been defined.

The following options can be used with the EVENTCOMB statement.

**LABEL=***'SAS-label'*
specifies a label for the dummy variable for this event. 'SAS-label' is a quoted text string of up to 256 characters. The default label is 'Dummy Variable for Event <variable-name>,' where <variable-name> is the name specified in the EVENT statement. The label is also stored as a description in the EVENTDATA OUT= data set. If no label is specified, then "." is displayed in the EVENTDATA OUT= data set, but the default label is still used for the dummy variable.

**RULE=***value*
specifies the action to take when combining events. The RULE= option accepts the following values: ADD | MAX | MIN | MINNZ | MINMAG | MULT. The default is RULE=ADD.

Table 9.2 explains how the RULE= option is interpreted. Example 9.1 shows how PROC HPFEVENTS interprets the RULE= option in the EVENTDEF and EVENTCOMB statements.

## EVENTDATA Statement

**EVENTDATA** *options* **;**

An EVENTDATA statement can be used with PROC HPFEVENTS to input events from an events data set and to output events to an events data set. Either the IN= or the OUT= option must be specified.

The following options can be used with the EVENTDATA statement.

**CONDENSE**
specifies that the EVENTDATA OUT= data set be condensed; any variables that contain only default values are omitted from the data set. The EVENTDATA IN= option reads both condensed data sets and data sets that have not been condensed. For more details, see the "EVENTDATA Statement OUT= Data Set" on page 304 section.

**IN=***SAS-data-set*
names an input data set that contains event definitions to be used in the PROC HPFEVENTS procedure.

**OUT=***SAS-data-set*
names the output data set to contain the event definitions as specified in the EVENTDATA IN= data sets and the EVENTDEF, EVENTKEY, and EVENTCOMB statements. The OUT= data set can then be used in other SAS procedures to define events.

## EVENTDEF Statement

**EVENTDEF** *SAS-variable-name= timing-value-list / qualifier options* **;**

The EVENTDEF statement defines an event that can be included in forecasting models. The following options can be used with the EVENTDEF statement.

**AFTER=( <DURATION=value> <SLOPE=value> )**
specifies options that control the event definition after the timing value. The DURATION= and SLOPE= suboptions are used within the parentheses in the AFTER=( ) option. See the BEFORE= option for information about the DURATION= and SLOPE= suboptions.

**BEFORE=( <DURATION=value> <SLOPE=value> )**
specifies options that control the event definition before the timing value. The DURATION= and SLOPE= options are used within the parentheses in the BEFORE=( ) option.

**DURATION=***number*
specifies the event duration before the timing value when used in the BEFORE=( ) option or after the timing value when used in the AFTER=( ) option.

**SLOPE=GROWTH | DECAY**
specifies whether a ramp or temporary change type is growth or decay. The SLOPE= value in the BEFORE= option controls the slope before the timing value and the SLOPE= value in the AFTER= option controls the slope after the timing value. SLOPE= is ignored unless TYPE=RAMP, TYPE=TR, TYPE=TEMPRAMP, or TYPE=TC. SLOPE= is also ignored if the corresponding DURATION=0.

SLOPE=GROWTH is the default in all cases except TYPE=TC. For TYPE=TC, the default is BEFORE=(SLOPE=GROWTH) and AFTER=(SLOPE=DECAY).

The following statement specifies a ramp up, followed by a ramp down.

```
EVENTDEF E1= '01JAN1950'D / BEFORE=(DURATION=3 SLOPE=GROWTH)
                            AFTER=(DURATION=4 SLOPE=DECAY)
                            TYPE=RAMP;
```

The event dummy observations that immediately precede the timing value contain the following values: 0, $\frac{1}{3}$, $\frac{2}{3}$. The observation at the timing value has a value of 1. The observations immediately after the timing value are $\frac{3}{4}$, $\frac{2}{4}$, $\frac{1}{4}$, 0.

**LABEL=***'SAS-label'*
specifies a label for the dummy variable for this event. 'SAS-label' is a quoted text string of up to 256 characters. The default label is 'Dummy Variable for Event <variable-name>,' where <variable-name> is the name specified in the EVENT statement. The label is also stored as a description in the EVENTDATA OUT= data set. If no label is specified, then "." is displayed in the EVENTDATA OUT= data set, but the default label is still used for the dummy variable.

**PERIOD=***interval*

specifies the interval for the frequency of the event. For example, PERIOD=YEAR should produce a dummy value that is periodic in a yearly pattern. If the PERIOD= option is omitted, the event is not periodic. The PERIOD= option also does not apply to observation numbers, which are not periodic, or to date keywords, which have their own periodicity. See Chapter 4, "Date Intervals, Formats, and Functions" (*SAS/ETS User's Guide*) for the intervals that can be specified.

**PULSE=***interval*

specifies the interval to be used with the DURATION= option to determine the width of the event. The default pulse is one observation. When no DURATION= values are specified and the PULSE= option is specified, the DURATION= values are set to zero. See Chapter 4, "Date Intervals, Formats, and Functions" (*SAS/ETS User's Guide*) for the intervals that can be specified.

**RULE=***value*

specifies the action to take when the defined event has multiple timing values that overlap. When the timing values do not overlap, RULE= has no impact because if there is only one defined value for an observation, that value is always used. The RULE= option accepts the following values: ADD | MAX | MIN | MINNZ | MINMAG | MULT. The default is RULE=ADD. Table 9.2 explains how the RULE= option is interpreted when the value for an observation is defined by multiple timing values.

**Table 9.2** Definition of RULE= Option Values

| RULE= Option | Name | Definition |
|---|---|---|
| ADD | Add | Add the values |
| MAX | Maximum | Use the maximum value |
| MIN | Minimum | Use the minimum value |
| MINNZ | Minimum nonzero | Use the minimum nonzero value |
| MINMAG | Minimum magnitude | Use the value whose magnitude is the least |
| MULT | Multiply | Multiply the values |

Because the range of the event associated with a timing value might not include all the observations in the series, RULE= can be interpreted differently when you are using multiple timing values in one EVENTDEF statement versus defining a combination event by using the EVENTCOMB statement. Thus, the dummy variables TWOTIMING and TWOEVENTS defined in the following statements are different:

```
eventdef xmasrp= christmas / before=(slope=growth duration=3)
                             type=ramp rule=min ;
eventdef easterrp= easter  / before=(slope=growth duration=3)
                             type=ramp rule=min ;
eventdef twotiming= easter christmas /
                             before=(slope=growth duration=3)
                             type=ramp rule=min ;
eventcomb twoevents= easterrp xmasrp / rule=min ;
```

In the section "Examples: HPFEVENTS Procedure" on page 305, Example 9.1 shows how PROC HPFEVENTS interprets each of these statements.

**SHIFT=***number*

specifies the number of pulses to shift the timing value $\delta$. The default is not to shift the timing value ($\delta = 0$). When the SHIFT= option is used, all timing values in the list (including those generated by date keywords) are shifted. Thus, SHIFT= can be used with EASTER to specify ecclesiastical holidays that are based on Easter. For example, the following statement specifies Good Friday, which is defined as 2 days before Easter (Montes 2001a).

```
EVENTDEF GoodFriday= EASTER / SHIFT=-2 PULSE=DAY;
```

**TCPARM=***number*

specifies a parameter $0 \leq \phi \leq 1$ used in the growth/decay equation for TYPE=TC given in Table 9.8. The TCPARM= value is the rate of growth or decay. A larger TCPARM= value causes faster growth or decay. TCPARM is ignored unless TYPE=TC. The default value is 0.5.

**TYPE=***option*

specifies the type of event variable. Each type uses a different formula to create the dummy variables. The formula for each TYPE= option is dependent on the other qualifiers that are specified in the EVENTDEF options. The formula is applied to each timing value specified in the timing-value list. The TYPE= option accepts the following values: POINT | LS | RAMP | TR | TEMPRAMP | TC | LIN | LINEAR | QUAD | CUBIC | INV | INVERSE | LOG | LOGARITHMIC. Table 9.9 and Table 9.10 illustrate the basic shape for each TYPE= value. TYPE=POINT is the default type.

**VALUE=***number*

specifies the event indicator value $\nu$. The default event indicator value is one ($\nu = 1.0$). Table 9.8 provides details about the effect of the event indicator value on the dummy variables. However, for TYPE=POINT | LS | RAMP | TR | TC events that consist of a single timing value with finite duration, you can think of the event indicator value as the maximum amplitude: the values of the dummy variable should be bounded below by zero and above by the event indicator value. For trend events (TYPE = LINEAR | QUAD | CUBIC | INV | LOG ), the event indicator value is the coefficient of the term.

## EVENTDUMMY Statement

**EVENTDUMMY OUT=** *SAS-data-set* **;**

An EVENTDUMMY statement can be used with PROC HPFEVENTS to output dummy variables for events to a data set. The OUT= option must be specified.

The following option can be used with the EVENTDUMMY statement.

**OUT=***SAS-data-set*
names the output data set to contain the dummy variables for the specified events based on the ID information as specified in the ID statement. The OUT= data set also includes variables as specified in the VAR, BY, and ID statements.

## EVENTGROUP Statement

**EVENTGROUP** *variable=( variable-list )* < */ option* > **;**

An EVENTGROUP statement can be used with PROC HPFEVENTS to create an event group or make a predefined event group available for processing. The EVENTGROUP statement constructs a SAS complex event. A complex event is an event that is represented by multiple dummy variables. For example, seasonal effects usually require multiple dummy variables. The predefined SAS event groups are also available directly through PROC HPFENGINE. Each EVENTGROUP predefined group keyword has a predefined set of event keywords associated with the predefined group. The default SAS variable name for the predefined event is the predefined event keyword. However, you can specify a SAS variable name for the event.

For example, you can rename the DAYS predefined event group to TD using the following statement:

```
eventgroup td=days;
```

The following option can be used with the EVENTGROUP statement.

**LABEL=***'SAS-label'*
specifies a description stored in the EVENTDATA OUT= data set. If no label is specified, then "." is displayed in the EVENTDATA OUT= data set.

Table 9.3 describes the predefined SAS event group keywords. The SEASONAL group is a predefined complex event; the SEASONAL group is interpreted to be one of the other SEASONAL groups at the time that dummy variables are created based on the ID statement. The ID statement could be the ID statement associated with either PROC HPFEVENTS or PROC HPFENGINE.

**Table 9.3** Definitions for EVENTGROUP Predefined Event Group Keywords

| Variable Name | Description | Associated Event Keywords |
|---|---|---|
| SEASONAL | Seasonal | Depending on ID statement:<br>SECOND_1, . . . SECOND_60 or<br>MINUTE_1, . . . MINUTE_60 or<br>HOUR_1, . . . HOUR_24 or<br>SUNDAY, . . . SATURDAY or<br>WEEK_1, . . . WEEK_53 or<br>TENDAY_1, . . . TENDAY_36 or<br>SEMIMONTH_1, . . . SEMIMONTH_24 or<br>JANUARY, . . . DECEMBER or<br>QTR_1, QTR_2, QTR_3, QTR_4 or<br>SEMIYEAR_1, SEMIYEAR_2 |
| SECONDS | Seasonal | SECOND_1, . . . SECOND_60 |
| MINUTES | Seasonal | MINUTE_1, . . . MINUTE_60 |
| HOURS | Seasonal | HOUR_1, . . . HOUR_24 |
| DAYS | Seasonal | SUNDAY, . . . SATURDAY |
| WEEKDAYS | Seasonal | MONDAY, . . . FRIDAY<br>(FRIDAY includes SATURDAY and SUNDAY) |
| WEEKS | Seasonal | WEEK_1, . . . WEEK_53 |
| TENDAYS | Seasonal | TENDAY_1, . . . TENDAY_36 |
| SEMIMONTHS | Seasonal | SEMIMONTH_1, . . . SEMIMONTH_24 |
| MONTHS | Seasonal | JANUARY, . . . DECEMBER |
| QTRS | Seasonal | QTR_1, QTR_2, QTR_3, QTR_4 |
| SEMIYEARS | Seasonal | SEMIYEAR_1, SEMIYEAR_2 |
| CUBICTREND | Trend | LINEAR, QUAD, CUBIC |
| QUADTREND | Trend | LINEAR, QUAD |

Table 9.11 gives more detail about the seasonal and trend predefined SAS events that compose the event groups.

## EVENTKEY Statement

**EVENTKEY** < *variable=* > *event-keyword* < */ options* > **;**

An EVENTKEY statement can be used to alter a user-defined simple event or a predefined SAS event or to create a new event.

See the section "Using the EVENTKEY Statement" on page 299 for more information about the EVENTKEY statement.

## ID Statement

**ID** *variable* < *options* > **;**

The ID statement names a numeric variable that identifies observations in the input and output data sets. The ID variable's values are assumed to be SAS date, time, or datetime values. In addition, the ID statement specifies the (desired) frequency associated with the actual time series. The information specified affects all dummy variables output by using the EVENTDUMMY statements. If no dummy variables are requested, the ID statement has no impact on processing, because the EVENTDEF definitions are independent of the time identification values of a time series. If the ID statement is specified, the INTERVAL= option must also be specified. If an ID statement is not specified, the observation number (with respect to the BY group) is used as the time ID. When the observation number is used as the time ID, only event timing values that are based on observation numbers are applied to the time series to create dummy variables; timing values based on SAS date or datetime values are ignored.

The following options can be used with the ID statement.

**ALIGN=***option*

controls the alignment of SAS dates used to identify output observations. The ALIGN= option accepts the following values: BEGINNING | BEG | B, MIDDLE | MID | M, and ENDING | END | E. The default is BEGINNING.

**END=***option*

specifies a SAS date, datetime, or time value that represents the end of the data. If the last time ID variable value is less than the END= value, the series is extended with missing values. If the last time ID variable value is greater than the END= value, the series is truncated. For example, END="&sysdate"D uses the automatic macro variable SYSDATE to extend or truncate the series to the current date. This option and the START= option can be used to ensure that data associated with each BY group contains the same number of observations.

**FORMAT=***format*

specifies the SAS format for the time ID values. If the FORMAT= option is not specified, the default format is implied from the INTERVAL= option.

**INTERVAL=***interval*

specifies the frequency of the input time series. For example, if the input data set consists of quarterly observations, then INTERVAL=QTR should be used. See Chapter 4, "Date Intervals, Formats, and Functions" (*SAS/ETS User's Guide*), for the intervals that can be specified.

**SETMISSING=***option* | *number*

specifies how missing values are assigned in the time series that is copied to the dummy variable data set when no observation matches the time ID in the input data set. If a number is specified, missing values are set to the number. If a missing value indicates an unknown value, this option should not be used. If a missing value indicates no value, a SETMISSING=0 should be used. You would typically use SETMISSING=0 for transactional data because no recorded data usually implies no activity. The following options can also be used to determine how missing values are assigned:

| | |
|---|---|
| MISSING | Missing values are set to missing. This is the default option. |
| SKIP | If the observation for the time ID value is missing in the input data set, then the corresponding observation is skipped in the dummy variable data set. This option can be useful if dummy variables are to be used as predictor values and you want to apply the PROC HPFENGINE ACCUMULATION option to the dummy data. |

**START=***option*

specifies a SAS date, datetime, or time value that represents the beginning of the data. If the first time ID variable value is greater than the START= value, the series is prefixed with missing values. If the first time ID variable value is less than the START= value, the series is truncated. This option and the END= option can be used to ensure that data associated with each BY group contain the same number of observations.

## VAR Statement

**VAR** *variables* **;**

A VAR statement can be used with PROC HPFEVENTS to copy input variables to the output dummy variables data set. Only numeric variables can be specified. If the VAR statement is omitted, all numeric variables are selected except those that appear in a BY or ID statement.

# Details: HPFEVENTS Procedure

## Event Definitions

This section describes the use of the EVENTDEF statement.

Although an event occurs at one or more time values, the event definition is independent of the time ID; that is, the event is a function that operates on a time ID variable. Once defined, the event can be output using the OUT= option of the EVENTDATA statement. A dummy variable for the event can be output using the OUT= option of the EVENTDUMMY statement. The dummy variable is created by evaluating the event with respect to the time ID. If a time ID is not specified using the ID statement, then the BY-group observation number is used as the time ID. More than one EVENTDEF statement can be specified.

Once defined, an event is referenced using its SAS variable name. When the event is output using the EVENTDATA statement, the event is identified by its SAS variable name. When a dummy variable is created using the event definition, the dummy variable name is the same as the event SAS variable name.

Each event must have a unique SAS variable name. If two event definitions have the same name, the following rules apply. If two EVENTDEF statements exist using the same name, the later statement is used. If an event is defined in both an EVENTDEF statement and in a data set specified using the EVENTDATA statement, the definition in the EVENTDEF statement is used. Any event defined using an EVENTDEF or EVENTDATA statement is used rather than a predefined SAS event.

Each EVENTDEF statement must be defined using one or more event timing values. The timing values can be specified using a list. Each item in the list can be a SAS date keyword, an integer, a SAS date, a SAS datetime, or a do-list. For example, the following EVENTDEF statement specifies timing values that use each of these methods in the order listed.

```
EVENTDEF EVENT1= USINDEPENDENCE 10 '25Dec2000'D
                 '01Mar1990:15:03:00'DT
                 '01Jan2000'D to '01Mar2000'D by month;
```

The timing values are interpreted as follows: any July 4 in the series; the 10th observation; December 25, 2000; March 1, 1990 at 3:03PM; January 1, 2000; February 1, 2000; and March 1, 2000.

The following two EVENTDEF statements specify identical timing values.

```
EVENTDEF MYFIRSTEVENT= '01Jan2000'D to '01Mar2000'D by month;
EVENTDEF MYNEXTEVENT= ( '01Jan2000'D, '01Feb2000'D, '01Mar2000'D );
```

The timing value list can be enclosed in parentheses, and commas can separate the items in the list. Numbers are always interpreted as observation numbers. The do-list can be based on observation numbers, SAS dates, or SAS datetimes. However, the first and second values in the list must be of the same type. The SAS grammar always expects the type of the second value to be the same as the type of the first value, and tries to interpret the statement in that fashion. The following statement yields erratic results.

```
EVENTDEF BADEVENT= '01Jan2000'D to '01Mar2000:00:00:00'DT by month;
```

Either the HPFEVENTS procedure produces a list much longer than expected or the procedure does not have enough memory to execute. **NOTE:** You should never mix date, datetime, and integer types in a do-list.

Table 9.4 shows the SAS default date keywords that you can use in a timing value list and their definitions. For information about customizing the SAS date keywords, see the section "Using the EVENTDS= System Option to Customize Holiday Predefined Date Keywords" on page 302.

**Table 9.4** Default Holiday Date Keywords and Definitions

| Date Keyword | Definition |
|---|---|
| BOXING | December 26th |
| CANADA | July 1st |
| CANADAOBSERVED | July 1st, or July 2nd if July 1st is a Sunday |
| CHRISTMAS | December 25th |
| COLUMBUS | Second Monday in October |
| EASTER | Easter Sunday |
| FATHERS | Third Sunday in June |
| HALLOWEEN | October 31st |
| LABOR | First Monday in September |
| MEMORIAL | Last Monday in May |
| MLK | Third Monday in January |
| MOTHERS | Second Sunday in May |
| NEWYEAR | January 1st |
| THANKSGIVING | Fourth Thursday in November |
| THANKSGIVINGCANADA | Second Monday in October |
| USINDEPENDENCE | July 4th |
| USPRESIDENTS | Third Monday in February (since 1971) |
| VALENTINES | February 14th |
| VETERANS | November 11th |
| VETERANSUSG | U.S. government observed date for Monday–Friday schedule |
| VETERANSUSPS | U.S. government observed date for Monday–Saturday schedule (U.S. Post Office) |
| VICTORIA | Monday on or preceding May 24th |

The date of Easter is calculated using a method described by Montes (2001b).

Table 9.5 shows the seasonal date keywords that can be used in a timing value list and their definitions.

**Table 9.5** Seasonal Date Keywords and Definitions

| Date Keyword | Definition |
|---|---|
| SECOND_1, ... SECOND_60 | The specified second |
| MINUTE_1, ... MINUTE_60 | The beginning of the specified minute |
| HOUR_1, ... HOUR_24 | The beginning of the specified hour |
| SUNDAY, ... SATURDAY | All Sundays, and so on, in the time series |
| WEEK_1, ... WEEK_53 | The first day of the *n*th week of the year<br>PULSE=WEEK.n shifts this date for $n \neq 1$ |
| TENDAY_1, ... TENDAY_36 | The 1st, 11th, or 21st of the appropriate month |
| SEMIMONTH_1, ... SEMIMONTH_24 | The 1st or 16th of the appropriate month |
| JANUARY, ... DECEMBER | The 1st of the specified month |
| QTR_1, QTR_2, QTR_3, QTR_4 | The first date of the quarter<br>PULSE=QTR.n shifts this date for $n \neq 1$ |
| SEMIYEAR_1, SEMIYEAR_2 | The first date of the semiyear<br>PULSE=SEMIYEAR.n shifts this date for $n \neq 1$ |

When dummy variables are created, each timing value is evaluated with respect to the time ID. Take care to choose the event timing values that are consistent with the time ID. In particular, date and datetime timing values are ignored when the time ID is based on the observation number.

The qualifier options define a function to be applied at each timing value.

## Event Types

The TYPE= option in the EVENTDEF statement specifies the type of the event defined. These event types are POINT, LS, RAMP, TR (or TEMPRAMP), TC, LIN (or LINEAR), QUAD, CUBIC, INV (or INVERSE), and LOG (or LOGARITHMIC).

Table 9.6, Table 9.7, and Table 9.8 show the formulas used to calculate the dummy variables for the event. Table 9.7 shows the formula used for each type of event when the event extends infinitely both before and after the timing value. Table 9.8 shows the formula used for each type of event for finite duration values.

To calculate the dummy values, the timing values list is first expanded, if applicable, with respect to the PERIOD option and the time series ID values. Then, if SHIFT is not zero, the timing values are shifted according to the value of SHIFT and PULSE. In the formulas, $t_i$ is the observation specified by the $i$th (shifted) timing value in the expanded timing value list, VALUE=$v$, TCPARM=$\phi$, AFTER=(DURATION=$n$ SLOPE=$s_a$), BEFORE=(DURATION=$m$ SLOPE=$s_b$), and PULSE=$interval$. Table 9.6 shows how to calculate $t_b$ and $t_e$, which are based on the values of the DURATION= option. $t_b$ and $t_e$ are the beginning and ending observations of the event definition. (For TYPE=RAMP, the ramp persists beyond the top of the ramp, either before $t_b$ or after $t_e$.) For more information about matching SAS date values to observations, see Chapter 3, "Working with Time Series Data" (*SAS/ETS User's Guide*), and Chapter 4, "Date Intervals, Formats, and Functions" (*SAS/ETS User's Guide*).

When multiple timing values are evaluated (because either the PERIOD= option or multiple values were specified), observations within the range of the function are calculated according to the formulas in the tables. Observations not in the range of the functions are left undefined. When the range from two timing values overlap, the RULE= option applies. After all timing values have been evaluated, the undefined values are set to zero.

When one DURATION= value is finite and the other is infinite, this is equivalent to extending the finite portion of the event infinitely in one direction. This principle can be understood by examining the results of the following EVENTDEF statements:

```
eventdef monlygg= '01Jun1951'D  /  TYPE=RAMP
         BEFORE=(SLOPE=GROWTH DURATION=4);
eventdef minfgg= '01Jun1951'D  /  TYPE=RAMP
         BEFORE=(SLOPE=GROWTH DURATION=4)
         AFTER=(SLOPE=GROWTH DURATION=ALL) ;
eventdef minfgd= '01Jun1951'D  /  TYPE=RAMP
         BEFORE=(SLOPE=GROWTH DURATION=4)
         AFTER=(SLOPE=DECAY DURATION=ALL) ;
```

In the section "Examples: HPFEVENTS Procedure" on page 305, Example 9.5 shows how PROC HPFEVENTS interprets each of these statements.

**Table 9.6** Calculating the Beginning and Ending Observation for Events

| **BEFORE= (DURATION=value)** | **PULSE=value** | **Definition of $t_b$** |
|---|---|---|
| ALL | N/A | $t_b = 1$, the first observation in the data set,<br>or $t_b =$ the observation specified by START= |
| $m = 0$ | Not specified | $t_b = t_i$, the observation specified by the shifted timing value |
| $m > 0$ | Not specified | $t_b = t_i - m$ |
| $m \geq 0$ | *interval* | $t_b =$ the observation specified by the<br>date INTNX( *interval*, timing value, –*m*, 'begin' ) |
| **AFTER= (DURATION=value)** | **PULSE=value** | **Definition of $t_e$** |
| ALL | N/A | $t_e =$ the last observation in the data set,<br>or $t_e =$ the observation specified by END= |
| $n = 0$ | Not specified | $t_e = t_i$, the observation specified by the shifted timing value |
| $n > 0$ | Not specified | $t_e = t_i + n$ |
| $n \geq 0$ | *interval* | $t_e =$ the observation specified by the<br>date INTNX( *interval*, timing value, *n*, 'end' ) |

**Table 9.7** Event Types for Infinite Durations ($m$ =ALL and $n$ =ALL)

| Type | Description | Definition |
|---|---|---|
| POINT | Point or pulse | $\xi_{it} = \nu$, for all $t$ |
| LS | Level shift | $\xi_{it} = \nu$, for all $t$ |
| RAMP | Ramp | |
| | SLOPE=GROWTH | $\xi_{it} = \nu(t - t_i)$, for all $t$ |
| | SLOPE=DECAY | $\xi_{it} = \nu(t_i - t)$, for all $t$ |
| | $s_b$ = GROWTH | $\xi_{it} = \nu(t - t_i)$, if $t \le t_i$ |
| | $s_a$ = DECAY | $\xi_{it} = \nu(t_i - t)$, if $t_i \le t$ |
| | $s_b$ = DECAY | $\xi_{it} = \nu(t_i - t)$, if $t \le t_i$ |
| | $s_a$ = GROWTH | $\xi_{it} = \nu(t - t_i)$, if $t_i \le t$ |
| TEMPRAMP or TR | Temporary ramp | TEMPRAMP is the same as RAMP for infinite cases |
| TC | Temporary change | |
| | SLOPE=GROWTH | $\xi_{it} = \nu\phi^{(t_i - t)}$, for all $t$ |
| | SLOPE=DECAY | $\xi_{it} = \nu\phi^{(t - t_i)}$, for all $t$ |
| | $s_b$ = GROWTH | $\xi_{it} = \nu\phi^{(t_i - t)}$, if $t \le t_i$ |
| | $s_a$ = DECAY | $\xi_{it} = \nu\phi^{(t - t_i)}$, if $t_i \le t$ |
| | $s_b$ = DECAY | $\xi_{it} = \nu\phi^{(t - t_i)}$, if $t \le t_i$ |
| | $s_a$ = GROWTH | $\xi_{it} = \nu\phi^{(t_i - t)}$, if $t_i \le t$ |
| LINEAR or LIN | Linear trend SLOPE= does not apply | $\xi_{it} = \nu(t - t_i)$, for all $t$ |
| QUAD | Quadratic trend SLOPE= does not apply | $\xi_{it} = \nu(t - t_i)^2$, for all $t$ |
| CUBIC | Cubic trend SLOPE= does not apply | $\xi_{it} = \nu(t - t_i)^3$, for all $t$ |
| INVERSE or INV | Inverse trend SLOPE= does not | $\xi_{it} = \nu/t$, for all $t$ |

**Table 9.7** *continued*

| Type | Description | Definition |
|---|---|---|
| | apply | |
| LOGARITHMIC or LOG | Log trend<br>SLOPE= does not apply | $\xi_{it} = \nu \log(t)$, for all $t$ |

**Table 9.8** Event Types (for Finite $m$,$n$)

| Type | Description | Definition |
|---|---|---|
| POINT | Point or pulse | $\xi_{it} = \nu$, if $t_b \le t \le t_e$<br>$\xi_{it} =$ undefined, otherwise |
| LS | Level shift | $\xi_{it} = \nu$, if $t_b \le t \le t_e$<br>$\xi_{it} =$ undefined, otherwise |
| RAMP | Ramp | |
| | $m = n = 0$<br>PULSE=*interval* $\le$ width of an observation | $\xi_{it} = 0$, if $t = t_i$<br>$\xi_{it} =$ undefined, otherwise |
| | SLOPE=GROWTH | $\xi_{it} = \nu(t - t_b)/(t_e - t_b)$, if $t_b \le t \le t_e$<br>$\xi_{it} = \nu$, if $t > t_e$<br>$\xi_{it} =$ undefined, otherwise |
| | SLOPE=DECAY | $\xi_{it} = \nu$, if $t < t_b$<br>$\xi_{it} = \nu(t_e - t)/(t_e - t_b)$, if $t_b \le t \le t_e$<br>$\xi_{it} =$ undefined, otherwise |
| | $s_b =$ GROWTH<br>$s_a =$ DECAY<br>$m > 0, n > 0$ | $\xi_{it} = \nu(t - t_b)/(t_i - t_b)$, if $t_b \le t \le t_i$<br>$\xi_{it} = \nu(t_e - t)/(t_e - t_i)$, if $t_i \le t \le t_e$<br>$\xi_{it} =$ undefined, otherwise |
| | $s_b =$ DECAY<br>$s_a =$ GROWTH<br>$m > 0, n > 0$ | $\xi_{it} = \nu$, if $t < t_b$<br>$\xi_{it} = \nu(t_i - t)/(t_i - t_b)$, if $t_b \le t \le t_i$<br>$\xi_{it} = \nu(t - t_i)/(t_e - t_i)$, if $t_i \le t \le t_e$<br>$\xi_{it} = \nu$, if $t > t_e$ |

**Table 9.8** *continued*

| Type | Description | Definition |
|---|---|---|
| TEMPRAMP or TR | Temporary ramp<br>$m = n = 0$<br>PULSE=*interval* ≤ width of an observation | $\xi_{it} = 0$, if $t = t_i$<br>$\xi_{it} =$ undefined, otherwise |
| | SLOPE=GROWTH | $\xi_{it} = \nu(t - t_b)/(t_e - t_b)$, if $t_b \le t \le t_e$<br>$\xi_{it} =$ undefined, otherwise |
| | SLOPE=DECAY | $\xi_{it} = \nu(t_e - t)/(t_e - t_b)$, if $t_b \le t \le t_e$<br>$\xi_{it} =$ undefined, otherwise |
| | $s_b =$ GROWTH<br>$s_a =$ DECAY<br>$m > 0, n > 0$ | $\xi_{it} = \nu(t - t_b)/(t_i - t_b)$, if $t_b \le t \le t_i$<br>$\xi_{it} = \nu(t_e - t)/(t_e - t_i)$, if $t_i \le t \le t_e$<br>$\xi_{it} =$ undefined, otherwise |
| | $s_b =$ DECAY<br>$s_a =$ GROWTH<br>$m > 0, n > 0$ | $\xi_{it} = \nu(t_i - t)/(t_i - t_b)$, if $t_b \le t \le t_i$<br>$\xi_{it} = \nu(t - t_i)/(t_e - t_i)$, if $t_i \le t \le t_e$<br>$\xi_{it} =$ undefined, otherwise |
| TC | Temporary change<br>SLOPE=GROWTH | $\xi_{it} = \nu\phi^{(t_e - t)}$, if $t_b \le t \le t_e$<br>$\xi_{it} =$ undefined, otherwise |
| | SLOPE=DECAY | $\xi_{it} = \nu\phi^{(t - t_b)}$, if $t_b \le t \le t_e$<br>$\xi_{it} =$ undefined, otherwise |
| | $s_b =$ GROWTH<br>$s_a =$ DECAY<br>$m > 0, n > 0$ | $\xi_{it} = \nu\phi^{(t_i - t)}$, if $t_b \le t \le t_i$<br>$\xi_{it} = \nu\phi^{(t - t_i)}$, if $t_i \le t \le t_e$<br>$\xi_{it} =$ undefined, otherwise |
| | $s_b =$ DECAY<br>$s_a =$ GROWTH<br>$0 < m \le n$ | $\xi_{it} = \nu\phi^{((t_e - t_i) + (t - t_i))}$, if $t_b \le t \le t_i$<br>$\xi_{it} = \nu\phi^{(t_e - t)}$, if $t_i \le t \le t_e$<br>$\xi_{it} =$ undefined, otherwise |
| | $s_b =$ DECAY<br>$s_a =$ GROWTH<br>$0 < n \le m$ | $\xi_{it} = \nu\phi^{(t - t_b)}$, if $t_b \le t \le t_i$<br>$\xi_{it} = \nu\phi^{((t_i - t_b) + (t_i - t))}$, if $t_i \le t \le t_e$<br>$\xi_{it} =$ undefined, otherwise |
| LINEAR or LIN | Linear trend<br>SLOPE= does not apply | $\xi_{it} = \nu(t - t_i)$, if $t_b \le t \le t_e$ |
| QUAD | Quadratic trend<br>SLOPE= does not apply | $\xi_{it} = \nu(t - t_i)^2$, if $t_b \le t \le t_e$ |

**Table 9.8** *continued*

| Type | Description | Definition |
|---|---|---|
| CUBIC | Cubic trend<br>SLOPE= does not apply | $\xi_{it} = \nu(t - t_i)^3$, if $t_b \leq t \leq t_e$ |
| INVERSE or INV | Inverse trend<br>SLOPE= does not apply | $\xi_{it} = \nu/(t - t_b + 1)$, if $t_b \leq t \leq t_e$ |
| LOGARITHMIC or LOG | Log trend<br>SLOPE= does not apply | $\xi_{it} = \nu \log(t - t_b + 1)$, if $t_b \leq t \leq t_e$ |

Note that undefined values are set to zero after all timing values have been evaluated. See the RULE= option for details about evaluating overlapping timing values.

## Details of Event Specifications

The event always occurs at the timing value. You would specify 3 observations before, 1 observation at the timing value, and 4 after the timing value for a total of $3 + 1 + 4 = 8$ observations as follows:

```
EVENTDEF E1= '01JAN1950'D / BEFORE=(DURATION=3)
                            AFTER=(DURATION=4);
```

You would specify 3 weeks before, the week of the timing value, and 4 weeks after the timing value using a combination of the BEFORE=, AFTER=, and PULSE= options as follows:

```
EVENTDEF E1= '01JAN1950'D / BEFORE=(DURATION=3)
                            AFTER=(DURATION=4)
                            PULSE=WEEK;
```

DURATION=ALL implies that the event should be extended to the beginning (BEFORE=) or end (AFTER=) of the series. If only one DURATION= value is specified, the other value is assumed to be zero. When neither DURATION= value is specified and the PULSE= value is specified, both DURATION= values are set to zero. When neither DURATION= value is specified and the PULSE= value is not specified, then both DURATION= values are assigned default values based on the TYPE= option. For polynomial trend events (TYPE = LINEAR | QUAD | CUBIC), the default DURATION= value is ALL for both the BEFORE=( ) option and the AFTER=( ) option. For other events, the default value for the BEFORE=( ) option is always zero, and the default event duration for the AFTER=( ) option depends on the TYPE= option. Table 9.9 and Table 9.10 show default duration values by TYPE= value and how the basic event shape depends on the duration value. DURATION=ALL is represented in the event definition data set as a special missing value displayed as "A". For more information about special missing values, see SAS System Concepts in *SAS Language Reference: Concepts*.

**Table 9.9** Default DURATION= Values for Non-Trend Types

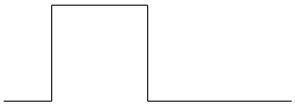
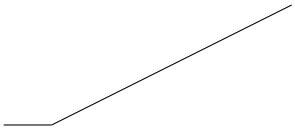
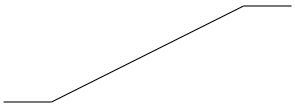
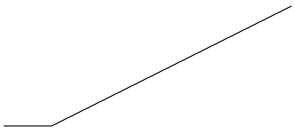
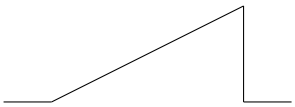
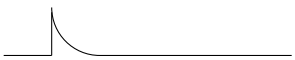
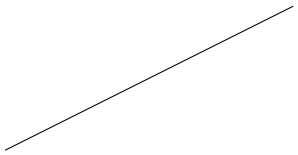
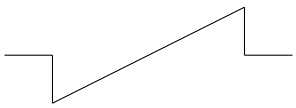
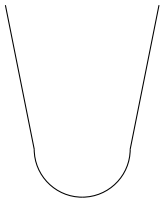
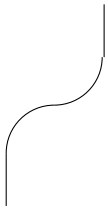
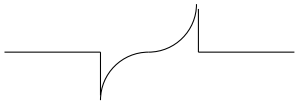
| Non-Trend TYPE= (BEFORE= Default Is 0) | AFTER= (DURATION=) Default | Default Shape | Shape When Finite AFTER= Duration > 0 |
|---|---|---|---|
| TYPE=POINT | Default is zero |  |  |
| TYPE=LS | Default is ALL (or end of series) |  |  |
| TYPE=RAMP | Default is ALL (or end of series) |  |  |
| TYPE= TEMPRAMP or TR | Default is ALL (or end of series) |  |  |
| TYPE=TC | Default is ALL (or end of series) |  |  |
| TYPE=INV or INVERSE | Default is ALL (or end of series) |  |  |
| TYPE=LOG or | Default is ALL |  |  |

**Table 9.9** *continued*

| Non-Trend TYPE= (BEFORE= Default Is 0) | AFTER= (DURATION=) Default | Default Shape | Shape When Finite AFTER= Duration > 0 |
|---|---|---|---|
| LOGARITHMIC | (or end of series) | | |

**Table 9.10** Default DURATION= Values for Trend Types

| Trend TYPE= | BEFORE= and AFTER= (DURATION=) Default | Default Shape | Shape When Finite BEFORE= and AFTER= Duration > 0 |
|---|---|---|---|
| TYPE=LINEAR TYPE=LIN | Default is ALL (or entire series) |  |  |
| TYPE=QUAD | Default is ALL (or entire series) |  |  |
| TYPE=CUBIC | Default is ALL (or entire series) |  |  |

## Using the EVENTKEY Statement

An EVENTKEY statement can be used with PROC HPFEVENTS to make a predefined SAS event available for processing. The EVENTKEY statement constructs a SAS simple event for each predefined SAS event keyword. The predefined SAS events are also available directly through PROC HPFDIAGNOSE and PROC HPFENGINE. Each EVENTKEY variable has a predefined set of timing values and qualifiers associated with the predefined event keyword. The options are the same as in the EVENTDEF statement and can be used to redefine the qualifiers that are associated with the predefined event. As shown in the section "Getting

Started: HPFEVENTS Procedure" on page 275, the default SAS variable name for the predefined event is the predefined event keyword. However, you can specify a SAS name for the event. For example, you can rename the CHRISTMAS predefined EVENT to XMAS by using the following statement:

```
eventkey xmas= christmas;
```

If you redefine the qualifiers associated with a predefined SAS event and do not rename the event, then that has the impact of redefining the predefined SAS event, because any user definition takes precedence over a SAS predefined definition. The following example produces an event FALLHOLIDAYS with a pulse of 1 day at Halloween and a pulse of 1 month at Thanksgiving.

```
eventkey thanksgiving / pulse=month;
eventcomb fallholidays= halloween thanksgiving;
```

Predefined SAS events are based on a SAS date keyword, a trend keyword, or an additive outlier or level shift based on a timing value. Table 9.11 describes how to construct a predefined SAS event keyword. It also gives the default qualifier options for those predefined events.

An EVENTKEY statement can be used in a similar manner to modify or clone a user-defined simple event. In the following example, the EVENTDEF statement is used to define a simple event named SPRING. The EVENTKEY statement is used to modify the SPRING event definition, and then the EVENTKEY statement is used to create a new event named SPRINGBREAK based on the previously defined user event named SPRING. So the example defines a total of two events, SPRING and SPRINGBREAK. The EVENTKEY statement can be used to modify the qualifiers; it cannot be used to modify the timing values.

```
eventdef spring = '20mar2005'd;
eventkey spring / pulse=day;
eventkey SPRINGBREAK = spring / pulse=week;
```

Suppose that the preceding events are stored in a data set named SPRINGHOLIDAYS. The first EVENTKEY statement in the following example clones SPRING as an event named FirstDayOfSpring. The second EVENTKEY statement changes the case of the SPRINGBREAK event name.

```
eventdata in=springholidays;
eventkey FirstDayOfSpring = spring;
eventkey Springbreak = springbreak;
```

Event names that refer to a previously defined event are not case sensitive. However, event names that are used to create a new event have the case preserved in the _NAME_ variable of the EVENTDATA OUT= data set and the variable name used in the EVENTDUMMY OUT= data set.

**Table 9.11** Definitions for EVENTKEY Predefined Event Keywords

| Variable Name or Variable Name Format | Description | Qualifier Options |
|---|---|---|
| AO<obs>OBS<br>AO<date>D | Outlier | TYPE=POINT VALUE=1<br>BEFORE=(DURATION=0) |

**Table 9.11** *continued*

| Variable Name or Variable Name Format | Description | Qualifier Options |
|---|---|---|
| AO<datetime>DT | | AFTER=(DURATION=0) |
| LS<obs>OBS<br>LS<date>D<br>LS<datetime>DT | Level shift | TYPE=LS VALUE=1<br>BEFORE=(DURATION=0)<br>AFTER=(DURATION=ALL) |
| TLS<obs>OBS<n><br>TLS<date>D<n><br>TLS<datetime>DT<n> | Temporary level shift | TYPE=LS VALUE=1<br>BEFORE=(DURATION=0)<br>AFTER=(DURATION=<n>) |
| NLS<obs>OBS<br>NLS<date>D<br>NLS<datetime>DT | Negative level shift | TYPE=LS VALUE=–1<br>BEFORE=(DURATION=0)<br>AFTER=(DURATION=ALL) |
| CBLS<obs>OBS<br>CBLS<date>D<br>CBLS<datetime>DT | U.S. Census Bureau level shift | TYPE=LS VALUE=–1<br>SHIFT=–1<br>BEFORE=(DURATION=ALL)<br>AFTER=(DURATION=0) |
| TC<obs>OBS<br>TC<date>D<br>TC<datetime>DT | Temporary change | TYPE=TC VALUE=1<br>BEFORE=(DURATION=0)<br>AFTER=(DURATION=ALL) |
| <date keyword> | Date pulse | TYPE=POINT VALUE=1<br>BEFORE=(DURATION=0)<br>AFTER=(DURATION=0)<br>PULSE=DAY |
| LINEAR<br>QUAD<br>CUBIC | Polynomial Trends | TYPE=LIN<br>TYPE=QUAD<br>TYPE=CUBIC<br>VALUE=1<br>BEFORE=(DURATION=ALL)<br>AFTER=(DURATION=ALL)<br>Default timing value is 0 observation |
| INVERSE<br>LOG | Trends | TYPE=INV<br>TYPE=LOG<br>VALUE=1<br>BEFORE=(DURATION=0)<br>AFTER=(DURATION=ALL)<br>Default timing value is 0 observation |

**Table 9.11** *continued*

| Variable Name or Variable Name Format | Description | Qualifier Options |
|---|---|---|
| <seasonal keywords> | Seasonal | TYPE=POINT<br>PULSE= depends on keyword<br>VALUE=1<br>BEFORE=(DURATION=0)<br>AFTER=(DURATION=0)<br>Timing values based on keyword |

The date keywords described in Table 9.4 that can be used in the EVENTDEF statement can be used as predefined SAS event keywords. The timing values are as defined in Table 9.4, and the default qualifiers are as shown in Table 9.11. Table 9.5 in the section on the EVENTDEF statement shows the seasonal keywords that can be used as predefined SAS event keywords. The default qualifiers for seasonal keywords are shown in Table 9.11. Table 9.12 gives a more detailed description of how date and observation numbers are encoded into AO, LS, TLS, NLS, CBLS, and TC type predefined events.

**Table 9.12** Details for Encoding Date Information into AO, LS, TLS, NLS, CBLS, and TC Type EVENTKEY Variable Names

| Variable Name Format | Example | Refers to |
|---|---|---|
| AO<int>OBS | AO15OBS | 15th observation |
| AO<date>D | AO01JAN2000D | '01JAN2000'D |
| AO<date>h<hr>m<min>s<sec>DT | AO01Jan2000h12m34s56DT | '01Jan2000:12:34:56'DT |
| TLS<int>OBS<n> | TLS15OBS10 | 15th observation |
| TLS<date>D<n> | TLS01JAN2000D10 | '01JAN2000'D |
| TLS<date>h<hr>m<min>s<sec>DT<n> | TLS01Jan2000h12m34s56DT10 | '01Jan2000:12:34:56'DT |

Several types of predefined level shifts are available. The parameter for the negative level shift is same as that for the level shift, but with opposite sign. If the parameter for a level shift is negative, then replacing the level shift with a negative level shift will result in a positive parameter value. The U.S. Census Bureau level shift is defined in the same manner as the level shift in the U.S. Bureau of the Census X-12-ARIMA Seasonal Adjustment program (U.S. Bureau of the Census 2001). The advantage of the U.S. Census Bureau level shift is that as historical observations are dropped and the point of the level shift is no longer within the span of the series, the constant term of the series does not change.

## Using the EVENTDS= System Option to Customize Holiday Predefined Date Keywords

You can use the DATEKEYS procedure and the EVENTDS= system option to construct user-defined date keywords. The following example constructs the user-defined date keyword GoodFriday.

```
proc datekeys;
   datekeydef GoodFriday = EASTER / SHIFT=-2 PULSE=DAY;
```

```
   datekeydata out=MyHolidays;
run;

options eventds=(MyHolidays);

proc hpfevents;
   eventkey GoodFriday;
   eventdata out=MyEvents condense;
run;
```

## Missing Value Interpretation

When the EVENTDUMMY statement is used to create dummy variables, you might need to specify the handling of missing observations in the input data set (that is, where the observation corresponding to a time ID is missing from the data set). In that case, the input data set does not contain a value for the variables to be copied to the EVENTDUMMY OUT= data set. Sometimes missing values should be interpreted as unknown values. The forecasting models used by the HPFENGINE procedure can effectively handle missing values. (See Chapter 7, "The HPFENGINE Procedure," for more details.) In this case, SETMISSING=MISSING can be used. But sometimes missing values are known, such as when no observations should be interpreted as no (zero) value. In this case, specify SETMISSING=0. In other cases, missing time IDs should be skipped, such as when the data are to be accumulated at a later time. In this case, specify SETMISSING=SKIP.

## Data Set Input/Output

The IN= option in the EVENTDATA statement specifies a data set that contains saved event definitions for input into the HPFEVENTS procedure.

PROC HPFEVENTS can create the OUT= data sets in the EVENTDATA and EVENTDUMMY statements. The OUT= data set in the EVENTDATA statement contains the event definitions that can be used as input to another SAS procedure. The OUT= data set in the EVENTDUMMY statement contains the variables that are listed in the BY statement, the ID variable, any variables that are defined by the VAR statement, and any dummy variables that are generated by the procedure.

### EVENTDATA Statement IN= Data Set

The IN= option in the EVENTDATA statement inputs a data set that is usually created by a previous run of PROC HPFEVENTS, where the OUT= option is specified in the EVENTDATA statement. The input data set can also be created in a prior DATA step, where the data set contains the same variables and attributes as those written to the OUT= data set in the EVENTDATA statement. If any variables that are listed in this OUT= data set are omitted from the data set to be read using the IN= option, then the default values for the omitted variables are used. This can occur, for example, when the CONDENSE option is specified in the PROC HPFEVENTS step where this OUT= data set is created. Whenever a condensed data set is used as input to the IN= option in the EVENTDATA statement, the default settings of the omitted variables are assumed.

If the IN= data set contains any extraneous variables that are not usually written to the OUT= data set in the EVENTDATA statement, then those variables are ignored.

All observations that are associated with a particular event should be recorded in contiguous observations in the data set. If a previously defined event is referenced in a subsequent observation in the IN= data set in the EVENTDATA statement, then PROC HPFEVENTS produces a warning to indicate that a duplicate event name was found, and the previous definition of the event is replaced by the definition in the latter observation(s).

### EVENTDATA Statement OUT= Data Set

The OUT= data set in the EVENTDATA statement contains the following variables. The default values for the CONDENSE option are also given. When all the observations in the variable are equal to the default value, the variable can be omitted from the event definition data set.

_NAME_ event variable name. _NAME_ is displayed with the case preserved. Because _NAME_ is a SAS variable name, the event can be referenced using any case. The _NAME_ variable is required; there is no default.

_CLASS_ class of event: SIMPLE, COMBINATION, PREDEFINED. The default for _CLASS_ is SIMPLE.

_KEYNAME_ one of the following: a date keyword (SIMPLE EVENT) or a predefined event variable name ( PREDEFINED EVENT) or an event name (COMBINATION event). All _KEYNAME_ values are displayed in uppercase. However, if the _KEYNAME_ value refers to an event name, then the actual name can be mixed case. The default for _KEYNAME_ is no keyname, designated by ".".

_STARTDATE_ either the date timing value or the first date timing value to use in a do- list. The default for _STARTDATE_ is no date, designated by a missing value.

_ENDDATE_ the last date timing value to use in a do-list. The default for _ENDDATE_ is no date, designated by a missing value.

_DATEINTRVL_ the interval for the date do-list. The default for _DATEINTRVL_ is no interval, designated by ".".

_STARTDT_ either the datetime timing value or the first datetime timing value to use in a do-list. The default for _STARTDT_ is no datetime, designated by a missing value.

_ENDDT_ the last datetime timing value to use in a do-list. the default for _ENDDT_ is no datetime, designated by a missing value.

_DTINTRVL_ the interval for the datetime do-list. The default for _DTINTRVL_ is no interval, designated by ".".

_STARTOBS_ either the observation number timing value or the first observation number timing value to use in a do-list. The default for _STARTOBS_ is no observation number, designated by a missing value.

_ENDOBS_ the last observation number timing value to use in a do-list. The default for _ENDOBS_ is no observation number, designated by a missing value.

_OBSINTRVL_ the interval length of the observation number do-list. The default for _OBSINTRVL_ is no interval, designated by ".".

_TYPE_ type of EVENT. The default for _TYPE_ is POINT.

_VALUE_ value for nonzero observation. The default for _VALUE_ is 1.0.

_PULSE_ interval that defines the units for the DURATION values. The default for _PULSE_ is no interval (one observation), designated by ".".

_DUR_BEFORE_ number of durations before the timing value. The default for _DUR_BEFORE_ is 0.

_DUR_AFTER_ number of durations after the timing value. The default for _DUR_AFTER_ is 0.

_SLOPE_BEFORE_ for TYPE=RAMP, TYPE=RAMPP, and TYPE=TC, determines whether the curve is GROWTH or DECAY before the timing value. The default for _SLOPE_BEFORE_ is GROWTH.

_SLOPE_AFTER_ for TYPE=RAMP, TYPE=RAMPP, and TYPE=TC, determines whether the curve is GROWTH or DECAY after the timing value. The default for _SLOPE_AFTER_ is GROWTH unless TYPE=TC; then the default is DECAY.

_SHIFT_ number of PULSE=intervals to shift the timing value. The shift can be positive (forward in time) or negative (backward in time). If PULSE= is not specified, then the shift is in observations. The default for _SHIFT_ is 0.

_TCPARM_ parameter for EVENT of TYPE=TC. The default for _TCPARM_ is 0.5.

_RULE_ rule to use when combining events or when timing values of an event overlap. The default for _RULE_ is ADD.

_PERIOD_ frequency interval at which the event should be repeated. If this value is missing, then the event is not periodic. The default for _PERIOD_ is no interval, designated by ".".

_LABEL_ label or description for the event. If you do not specify a label, then the default label value is displayed as ".". However, the default label used in an EVENTDUMMY OUT= data set is "Dummy Variable for Event <variable-name>". See the LABEL= option for more information about the default label.

## Printed Output

The HPFEVENTS procedure has no printed output other than warning and error messages as recorded in the log.

# Examples: HPFEVENTS Procedure

## Example 9.1: Multiple Timing Values in a Single Event versus Using Multiple Events and the EVENTCOMB Statement

This example illustrates how the HPFEVENTS procedure interprets multiple timing values that overlap. It also illustrates the results of the same timing values used in separate EVENTDEF statements that are combined using EVENTCOMB. Airline sales data are used for this illustration.

```
   data a(Label='Box-Jenkins Series G: International Airline Data');
     set sashelp.air;
     t = intnx( 'month', '01jan1949'd, _n_-1 );
     format t DATE.;
   run;

proc hpfevents data=sashelp.air ;
   var air;
   id date interval=month start='01Jan1949'D end='01Feb1950'D;
   eventdef xmasrp= christmas / before=(slope=growth duration=3)
                                type=ramp rule=min ;
   eventdef easterrp= easter / before=(slope=growth duration=3)
                               type=ramp rule=min ;
   eventdef twotiming= easter christmas /
                               before=(slope=growth duration=3)
                               type=ramp rule=min ;
   eventcomb twoevents= easterrp xmasrp / rule=min ;
   eventdata  out= evdsout1 (label='EASTER and CHRISTMAS Ramps');
   eventdummy  out= evdumout1 (label='Combining Timing Values');
run;

proc print data=evdumout1;
run;
```

**Output 9.1.1** Multiple Timing Values versus Multiple Events

| Obs | DATE | AIR | xmasrp | easterrp | twotiming | twoevents |
|---|---|---|---|---|---|---|
| 1 | JAN1949 | 112 | 0.00000 | 0.00000 | 0.00000 | 0.00000 |
| 2 | FEB1949 | 118 | 0.00000 | 0.33333 | 0.33333 | 0.00000 |
| 3 | MAR1949 | 132 | 0.00000 | 0.66667 | 0.66667 | 0.00000 |
| 4 | APR1949 | 129 | 0.00000 | 1.00000 | 1.00000 | 0.00000 |
| 5 | MAY1949 | 121 | 0.00000 | 1.00000 | 1.00000 | 0.00000 |
| 6 | JUN1949 | 135 | 0.00000 | 1.00000 | 1.00000 | 0.00000 |
| 7 | JUL1949 | 148 | 0.00000 | 1.00000 | 1.00000 | 0.00000 |
| 8 | AUG1949 | 148 | 0.00000 | 1.00000 | 1.00000 | 0.00000 |
| 9 | SEP1949 | 136 | 0.00000 | 1.00000 | 0.00000 | 0.00000 |
| 10 | OCT1949 | 119 | 0.33333 | 1.00000 | 0.33333 | 0.33333 |
| 11 | NOV1949 | 104 | 0.66667 | 1.00000 | 0.66667 | 0.66667 |
| 12 | DEC1949 | 118 | 1.00000 | 1.00000 | 1.00000 | 1.00000 |
| 13 | JAN1950 | 115 | 1.00000 | 0.00000 | 0.00000 | 0.00000 |
| 14 | FEB1950 | 126 | 1.00000 | 0.33333 | 0.33333 | 0.33333 |

In this example, the ramp for Christmas is defined for observations 9 through 14. When XMASRP is evaluated, the undefined values in observations 1 through 8 are replaced with zeros. The ramp for Easter is defined for the entire time series, as shown in the variable EASTERRP. When both timing values are used in one EVENTDEF statement for variable TWOTIMING, the values from the Easter ramp are used in observations 1 through 8, and the RULE=MIN is applied to observations 9 through 14. For the EVENTCOMB statement that defines the variable TWOEVENTS, the RULE=MIN option applies to all observations in the series.

## Example 9.2: Using a DATA Step to Construct an Events Data Set

This example uses the DATA step to automatically construct potential outliers related to the price data found in the data set SASHELP.PRICEDATA.

```
data orders(keep=date region line product sale);
   set sashelp.pricedata;
   format date monyy.;
run;
```

The following SAS statements construct an EVENTDATA IN= data set for potential outliers (identified as $sale > 450$). Only the _NAME_ and _STARTDATE_ variables are needed.

```
data outliers(keep=_name_ _startdate_  );
   set orders;
   if (sale > 450) then do;
      _name_ = trim('ao')||trim(left(put(year(date),8.)))||'_'
               ||trim(left(put(month(date),8.)));
      _startdate_=date;
      end;
   else delete;
   format _startdate_ monyy.;
run;
```

Next, identify which outliers apply to each product.

```
data product_event_list (keep= region line product _name_);
   set orders;
   if (sale > 450) then do;
      _name_ = trim('ao')||trim(left(put(year(date),8.)))||'_'
                ||trim(left(put(month(date),8.)));
      end;
   else delete;
run;
```

The potential outliers in the data set OUTL_REG1_LINE1_PROD1 apply to Region 1, Line 1, and Product 1.

```
data outl_reg1_line1_prod1;
   set product_event_list;
   if ((region ~= 1) | (line ~= 1) | (product ~= 1)) then delete;
run;
```

Dummy variables are created and duplicate outlier events are eliminated from the events definition data set.

```
proc hpfevents data=orders ;
   id date interval=month;
   by region line product;
   eventdata  in=outliers ;
   eventdata  out=outldatabase(label='outlier definitions')
              condense;
   eventdummy out=dummies(label='dummy variables');
run;
```

```
proc print data=outldatabase(obs=10);
run;

proc print data=outl_reg1_line1_prod1;
run;
```

Examining the data set OUTL_REG1_LINE1_PROD1 shows that you might want to look for outliers for May 1998, October 1999, March 2000, February 2001, June 2001, and September 2002.

**Output 9.2.1** Potential Outliers for Region 1, Line 1, Product 1

| Obs | region | line | product | _name_ |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | ao1998_5 |
| 2 | 1 | 1 | 1 | ao1999_10 |
| 3 | 1 | 1 | 1 | ao2000_3 |
| 4 | 1 | 1 | 1 | ao2001_2 |
| 5 | 1 | 1 | 1 | ao2001_6 |
| 6 | 1 | 1 | 1 | ao2002_9 |

PROC HPFEVENTS produced this data set, which is condensed and has the duplicate events eliminated.

**Output 9.2.2** Event Definition Data Set

| Obs | _NAME_ | _STARTDATE_ |
|---|---|---|
| 1 | ao1999_1 | 01JAN1999 |
| 2 | ao1999_11 | 01NOV1999 |
| 3 | ao2000_12 | 01DEC2000 |
| 4 | ao2002_1 | 01JAN2002 |
| 5 | ao1998_10 | 01OCT1998 |
| 6 | ao1999_8 | 01AUG1999 |
| 7 | ao2000_4 | 01APR2000 |
| 8 | ao2001_1 | 01JAN2001 |
| 9 | ao2001_12 | 01DEC2001 |
| 10 | ao2002_12 | 01DEC2002 |

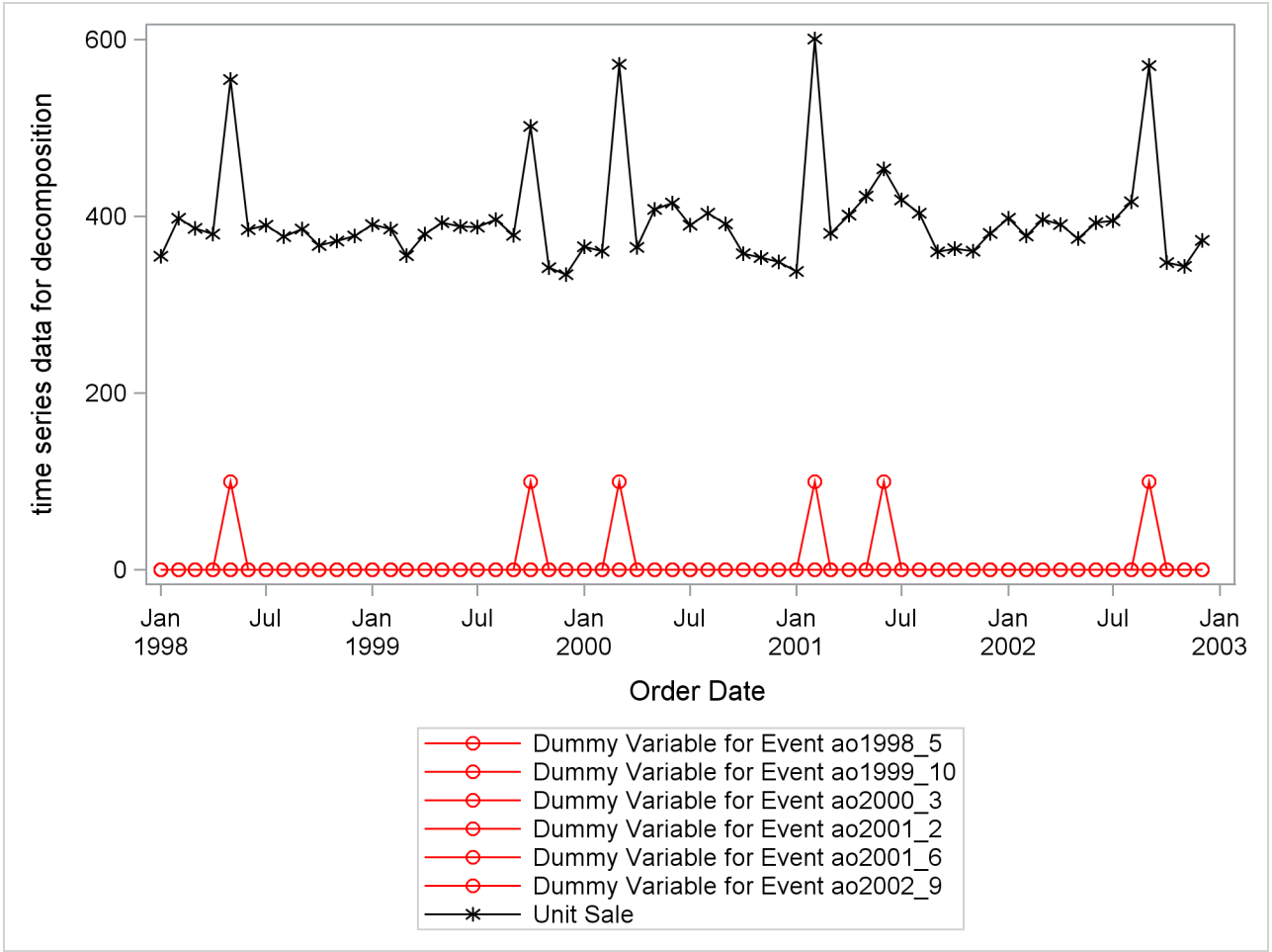
Select the observations related to Region 1, Line 1, Product 1 and scale the dummy variables that apply so that they are visible when plotted with the original data.

```
data pid1;
   set dummies;
   if ((region ~= 1) | (line ~= 1) | (product ~= 1)) then delete;
   else do;
      AO1998_5  = 100 * AO1998_5;
      AO1999_10 = 100 * AO1999_10;
      AO2000_3  = 100 * AO2000_3;
      AO2001_2  = 100 * AO2001_2;
      AO2001_6  = 100 * AO2001_6;
      AO2002_9  = 100 * AO2002_9;
   end;
run;
```

Use PROC SGPLOT to visually verify that these potential outliers are appropriate for the original data.

```
proc sgplot data=pid1;
   series x=date y=AO1998_5 / markers markerattrs=(symbol=circle color=red)
    lineattrs=(pattern=1 color=red);
   series x=date y=AO1999_10 / markers markerattrs=(symbol=circle color=red)
    lineattrs=(pattern=1 color=red);
   series x=date y=AO2000_3 / markers markerattrs=(symbol=circle color=red)
    lineattrs=(pattern=1 color=red);
   series x=date y=AO2001_2 / markers markerattrs=(symbol=circle color=red)
    lineattrs=(pattern=1 color=red);
   series x=date y=AO2001_6 / markers markerattrs=(symbol=circle color=red)
    lineattrs=(pattern=1 color=red);
   series x=date y=AO2002_9 / markers markerattrs=(symbol=circle color=red)
    lineattrs=(pattern=1 color=red);
   series x=date y=sale / markers markerattrs=(symbol=asterisk color=black)
    lineattrs=(pattern=1 color=black);
   yaxis label='time series data for decomposition';
run;
```

**Output 9.2.3** Plot of Amount and Dummy Variables



## Example 9.3: Preparing a Data Set for PROC HPFENGINE

This example illustrates how the HPFEVENTS procedure can be used to include events in the automatic forecasting of time series data. The data have been altered by adding a level shift of 100 beginning at October 1980. PROC HPFEVENTS is used to create an event named PROMOTION as a level shift occurring at October 1, 1980. PROC HPFENGINE identifies the parameter of the event PROMOTION as 97.6728, which is used in conjunction with the model named SP1 described as "ARIMA(0, 1, 1) No Intercept".

```
   data work_intv;
      set sashelp.workers;
      if date >= '01oct80'd then electric = electric+100;
      drop masonry;
   run;

* define event 'promotion';
proc hpfevents data=work_intv lead=12;
   id date interval=month;
   eventdef promotion=  '01oct80'd / TYPE=LS;
   eventdata out= evdsout1 (label='list of events');
run;

proc hpfarimaspec modelrepository=sasuser.mycat
                  specname=sp1
                  speclabel="ARIMA(0,1,1) No Intercept";
   dependent symbol=Y q=1 diflist=1 noint;
run;

proc hpfarimaspec modelrepository=sasuser.mycat
                   specname=sp2
                   speclabel="ARIMA(0,1,2)(0,1,1)_12 No Intercept";
   dependent symbol=Y q=(1,2)(12) diflist=1 12 noint;
run;

proc hpfselect modelrepository=sasuser.mycat
               selectname=myselect
               selectlabel="My Selection List";
   select select=mape holdout=12;
   spec sp1 sp2 /
        inputmap(symbol=y var=electric)
        eventmap(symbol=_none_ event=promotion)
        ;
run;

proc hpfengine data=work_intv lead=12 outest=outest
               globalselection=myselect
               modelrepository=sasuser.mycat
               inevent=evdsout1;
   id date interval=month;
   forecast electric / task = select;
run;

proc print data=outest; run;
```

**Output 9.3.1** Model Selection Using Events

| Obs | _NAME_ | _SELECT_ | _MODEL_ | _MODELVAR_ | _DSVAR_ | _VARTYPE_ | _TRANSFORM_ |
|---|---|---|---|---|---|---|---|
| 1 | ELECTRIC | MYSELECT | SP1 | Y | ELECTRIC | DEPENDENT | NONE |
| 2 | ELECTRIC | MYSELECT | SP1 | PROMOTION | PROMOTION | EVENT | NONE |

| Obs | _COMPONENT_ | _COMPMODEL_ | _FACTOR_ | _LAG_ | _SHIFT_ | _PARM_ | _LABEL_ | _SCORE_ |
|---|---|---|---|---|---|---|---|---|
| 1 | MA | | 1 | 1 | 0 | MA1_1 | ARIMA(0,1,1) No Intercept | |
| 2 | SCALE | | 0 | 0 | 0 | SCALE | ARIMA(0,1,1) No Intercept | |

| Obs | _EST_ | _STDERR_ | _TVALUE_ | _PVALUE_ | _STATUS_ |
|---|---|---|---|---|---|
| 1 | -0.5260 | 0.10798 | -4.8718 | .000007605 | 0 |
| 2 | 97.6728 | 4.28766 | 22.7800 | 2.0879E-32 | 0 |

## Example 9.4: Using Predefined SAS Event Keywords Directly in Other SAS Procedures

In Example 9.3, you can modify the SAS statements to use the EVENTS system directly without using PROC HPFEVENTS. Instead of creating an event named PROMOTION, you can use the SAS predefined event keyword LS01OCT1980D. This example uses the data set from Example 9.3 and the EVENT statement in PROC HPFDIAGNOSE to illustrate this method.

```
proc hpfdiag data=work_intv
             seasonality=12
             print=all;
   id date interval=month;
   forecast electric;
   event LS01OCT1980D;
   trend diff=1 sdiff=1;
   arimax;
run;
```

The output from PROC HPFDIAGNOSE shows that a model was selected that included the level shift occurring at October 1980.

**Output 9.4.1** Using the EVENT Statement in PROC HPFDIAGNOSE

**The HPFDIAGNOSE Procedure**

| Minimum Information Criterion | | | | | | |
|---|---|---|---|---|---|---|
| Lags | MA 0 | MA 1 | MA 2 | MA 3 | MA 4 | MA 5 |
| AR 0 | 5.883729 | 5.949528 | 6.02335 | 6.096772 | 6.170614 | 6.241918 |
| AR 1 | 5.949355 | 6.023199 | 6.096221 | 6.169567 | 6.243324 | 6.314554 |
| AR 2 | 6.023143 | 6.096346 | 6.170056 | 6.24107 | 6.314917 | 6.38698 |
| AR 3 | 6.096554 | 6.169837 | 6.240963 | 6.314829 | 6.387728 | 6.459893 |
| AR 4 | 6.170396 | 6.243647 | 6.314666 | 6.387965 | 6.461478 | 6.533759 |
| AR 5 | 6.241435 | 6.314684 | 6.386682 | 6.459847 | 6.533716 | 6.60647 |

**Output 9.4.1** *continued*

**ARIMA Model Specification**

| Variable | Functional Transform | Constant | p | d | q | P | D | Q | Seasonality | Model Criterion | Statistic | Status |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **ELECTRIC** | NONE | NO | 0 | 1 | 0 | 0 | 1 | 1 | 12 | RMSE | 13.6804 | OK |

**ARIMA Event Selection**

| Event Name | Selected | d | D | Status |
|---|---|---|---|---|
| **LS01OCT1980D** | YES | 1 | 1 | OK |

**ARIMA Model Specification After Adjusting for Events**

| Variable | Functional Transform | Constant | p | d | q | P | D | Q | Seasonality | Event | Model Criterion | Statistic | Status |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **ELECTRIC** | NONE | NO | 0 | 1 | 0 | 0 | 1 | 1 | 12 | 1 | RMSE | 3.3460 | OK |

## Example 9.5: Viewing Dummy Variables by Using the SGPLOT Procedure

This example illustrates how the HPFEVENTS procedure can be used to create dummy variables. These dummy variables can then be viewed using the SGPLOT procedure. This example also shows the behavior of ramp variables when used with the SLOPE= option.

```
proc hpfevents data=sashelp.air ;
   var air;
   id date interval=month start='01jan1951'd end='31dec1951'd;
   eventdef infgg= '01jun1951'd  /  type=ramp before=(slope=growth duration=all)
                                    after=(slope=growth duration=all);
   eventdef infgd= '01jun1951'd  /  type=ramp before=(slope=growth duration=all)
                                    after=(slope=decay duration=all) ;
   eventdef infdg= '01jun1951'd  /  type=ramp before=(slope=decay duration=all)
                                    after=(slope=growth duration=all)  ;
   eventdef infdd= '01jun1951'd  /  type=ramp before=(slope=decay duration=all)
                                    after=(duration=all slope=decay) ;
   eventdef minfgg= '01jun1951'd  /  type=ramp before=(slope=growth duration=4)
                                     after=(slope=growth duration=all) ;
   eventdef minfgd= '01jun1951'd  /  type=ramp before=(slope=growth duration=4)
                                     after=(slope=decay duration=all) ;
   eventdef minfdg= '01jun1951'd  /  type=ramp before=(slope=decay duration=4)
                                     after=(slope=growth duration=all);
   eventdef minfdd= '01jun1951'd  /  type=ramp before=(slope=decay duration=4)
                                     after=(slope=decay duration=all) ;
   eventdef monlygg= '01jun1951'd  /  type=ramp before=(slope=growth duration=4);
   eventdef monlygd= '01jun1951'd  /  type=ramp before=(slope=growth duration=4)
                                      after=(slope=decay);
   eventdef monlydg= '01jun1951'd  /  type=ramp before=(slope=decay duration=4)
                                      after=(slope=growth) ;
   eventdef monlydd= '01jun1951'd  /  type=ramp before=(slope=decay duration=4)
                                      after=(slope=decay);
   eventdef ninfgg= '01jun1951'd  /  type=ramp before=(slope=growth duration=all)
                                     after=(slope=growth duration=2) ;
   eventdef ninfgd= '01jun1951'd  /  type=ramp before=(slope=growth duration=all)
                                     after=(slope=decay duration=2) ;
```

```
   eventdef ninfdg= '01jun1951'd  /  type=ramp before=(slope=decay duration=all)
                                     after=(slope=growth duration=2) ;
   eventdef ninfdd= '01jun1951'd  /  type=ramp before=(slope=decay duration=all)
                                     after=(slope=decay duration=2) ;
   eventdef nonlygg= '01jun1951'd  /  type=ramp after=(slope=growth duration=2);
   eventdef nonlygd= '01jun1951'd  /  type=ramp after=(slope=decay duration=2)
                                      before=(slope=growth) ;
   eventdef nonlydg= '01jun1951'd  /  type=ramp before=(slope=decay)
                                      after=(slope=growth duration=2) ;
   eventdef nonlydd= '01jun1951'd  /  type=ramp before=(slope=decay)
                                      after=(slope=decay duration=2) ;
   eventdef mngg= '01jun1951'd  /  type=ramp before=(slope=growth duration=4)
                                   after=(slope=growth duration=2) ;
   eventdef mngd= '01jun1951'd  /  type=ramp before=(slope=growth duration=4)
                                   after=(slope=decay duration=2) ;
   eventdef mndg= '01jun1951'd  /  type=ramp before=(slope=decay duration=4)
                                   after=(slope=growth duration=2) ;
   eventdef mndd= '01jun1951'd  /  type=ramp before=(slope=decay duration=4)
                                   after=(slope=decay duration=2) ;
   eventdata  out= rampds (label='Ramps Using DURATION= and SLOPE=');
   eventdummy out= rampdummies (label='Dummy Variables for Ramps');
run;

proc print data=rampdummies(obs=10);
run;
```

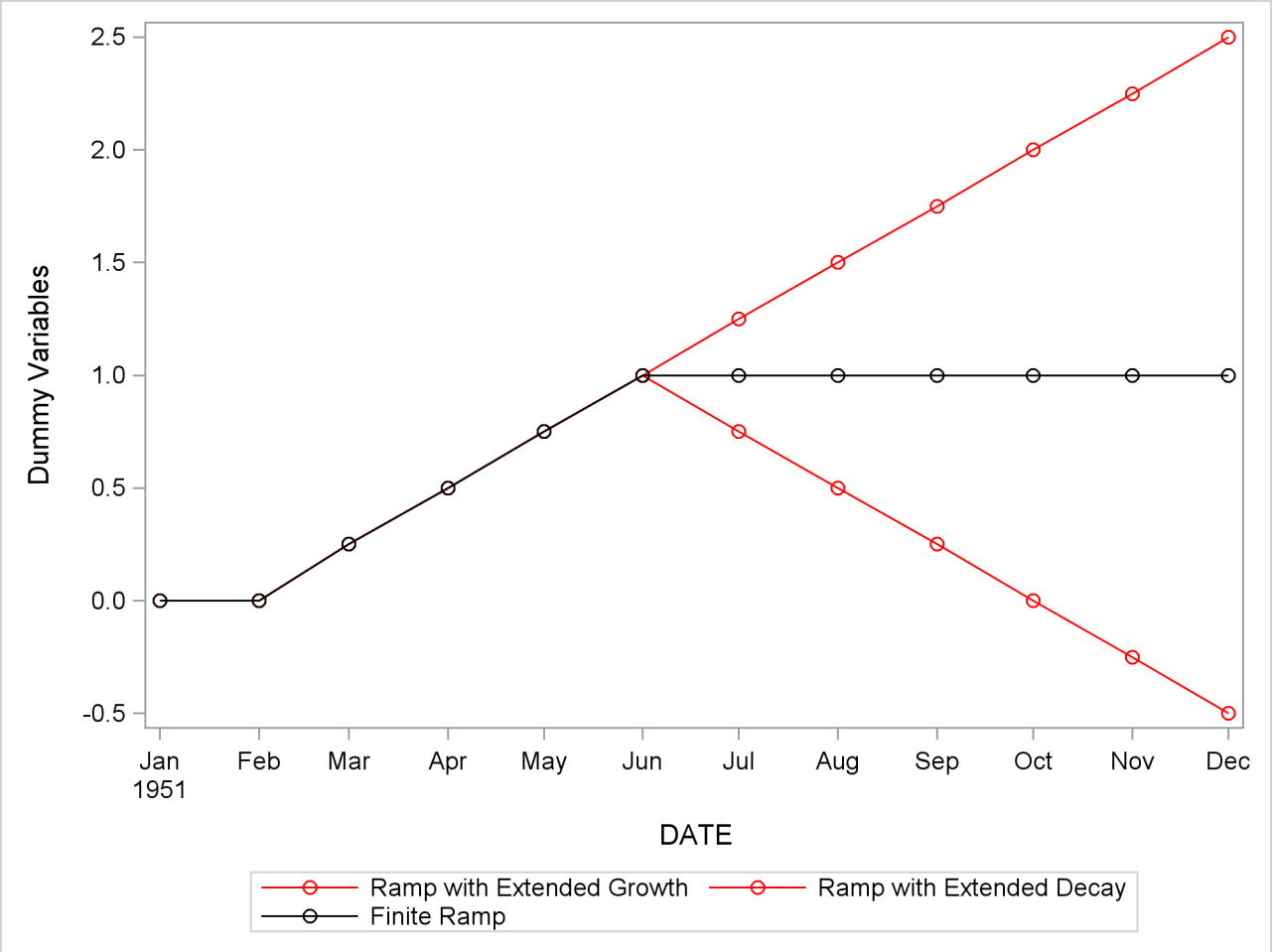
**Output 9.5.1** Ramp Dummy Variables

| Obs | DATE | AIR | infgg | infgd | infdg | infdd | minfgg | minfgd | minfdg | minfdd | monlygg | monlygd | monlydg | monlydd |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | JAN1951 | 145 | -5 | -5 | 5 | 5 | 0.00 | 0.00 | 1.00 | 1.00 | 0.00 | 0.00 | 1.00 | 1.00 |
| 2 | FEB1951 | 150 | -4 | -4 | 4 | 4 | 0.00 | 0.00 | 1.00 | 1.00 | 0.00 | 0.00 | 1.00 | 1.00 |
| 3 | MAR1951 | 178 | -3 | -3 | 3 | 3 | 0.25 | 0.25 | 0.75 | 0.75 | 0.25 | 0.25 | 0.75 | 0.75 |
| 4 | APR1951 | 163 | -2 | -2 | 2 | 2 | 0.50 | 0.50 | 0.50 | 0.50 | 0.50 | 0.50 | 0.50 | 0.50 |
| 5 | MAY1951 | 172 | -1 | -1 | 1 | 1 | 0.75 | 0.75 | 0.25 | 0.25 | 0.75 | 0.75 | 0.25 | 0.25 |
| 6 | JUN1951 | 178 | 0 | 0 | 0 | 0 | 1.00 | 1.00 | 0.00 | 0.00 | 1.00 | 1.00 | 0.00 | 0.00 |
| 7 | JUL1951 | 199 | 1 | -1 | 1 | -1 | 1.25 | 0.75 | 0.25 | -0.25 | 1.00 | 1.00 | 0.00 | 0.00 |
| 8 | AUG1951 | 199 | 2 | -2 | 2 | -2 | 1.50 | 0.50 | 0.50 | -0.50 | 1.00 | 1.00 | 0.00 | 0.00 |
| 9 | SEP1951 | 184 | 3 | -3 | 3 | -3 | 1.75 | 0.25 | 0.75 | -0.75 | 1.00 | 1.00 | 0.00 | 0.00 |
| 10 | OCT1951 | 162 | 4 | -4 | 4 | -4 | 2.00 | 0.00 | 1.00 | -1.00 | 1.00 | 1.00 | 0.00 | 0.00 |

| Obs | ninfgg | ninfgd | ninfdg | ninfdd | nonlygg | nonlygd | nonlydg | nonlydd | mngg | mngd | mndg | mndd |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | -2.5 | -1.5 | 2.5 | 3.5 | 0.0 | 1.0 | 0.0 | 1.0 | 0.00000 | 0.00 | 1.00 | 1.00000 |
| 2 | -2.0 | -1.0 | 2.0 | 3.0 | 0.0 | 1.0 | 0.0 | 1.0 | 0.00000 | 0.00 | 1.00 | 1.00000 |
| 3 | -1.5 | -0.5 | 1.5 | 2.5 | 0.0 | 1.0 | 0.0 | 1.0 | 0.16667 | 0.25 | 0.75 | 0.83333 |
| 4 | -1.0 | 0.0 | 1.0 | 2.0 | 0.0 | 1.0 | 0.0 | 1.0 | 0.33333 | 0.50 | 0.50 | 0.66667 |
| 5 | -0.5 | 0.5 | 0.5 | 1.5 | 0.0 | 1.0 | 0.0 | 1.0 | 0.50000 | 0.75 | 0.25 | 0.50000 |
| 6 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 | 1.0 | 0.66667 | 1.00 | 0.00 | 0.33333 |
| 7 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.83333 | 0.50 | 0.50 | 0.16667 |
| 8 | 1.0 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 | 1.00000 | 0.00 | 1.00 | 0.00000 |
| 9 | 1.0 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 | 1.00000 | 0.00 | 1.00 | 0.00000 |
| 10 | 1.0 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 | 1.00000 | 0.00 | 1.00 | 0.00000 |

```
proc sgplot data=rampdummies;
   series x=date y=minfgg / markers markerattrs=(color=red)
    lineattrs=(pattern=1 color=red) legendlabel='Ramp with Extended Growth';
   series x=date y=minfgd / markers markerattrs=(color=red)
    lineattrs=(pattern=1 color=red) legendlabel='Ramp with Extended Decay';
   series x=date y=monlygg / markers markerattrs=(color=black)
    lineattrs=(pattern=1 color=black) legendlabel='Finite Ramp';
   yaxis label='Dummy Variables';
run;
```

**Output 9.5.2** Plot of Finite and Extended Dummy Variables



# References

Montes, M. J. (2001a), *Calculation of the Ecclesiastical Calendar*, http://www.smart.net/~mmontes/ec-cal.html.

Montes, M. J. (2001b), *Nature (1876) Algorithm for Calculating the Date of Easter in the Gregorian Calendar*, http://www.smart.net/~mmontes/nature1876.html.

U.S. Bureau of the Census (2001), *X-12-ARIMA Seasonal Adjustment Program, Version 0.2.8*, U.S. Bureau of the Census, Washington, DC.

# Chapter 10
# The HPFEXMSPEC Procedure

**Contents**


Overview: HPFEXMSPEC Procedure . . . . . **317**
Getting Started: HPFEXMSPEC Procedure . . . . . **318**
Syntax: HPFEXMSPEC Procedure . . . . . **318**
Functional Summary . . . . . 318
PROC HPFEXMSPEC Statement . . . . . 319
EXM Statement . . . . . 319
Examples: HPFEXMSPEC Procedure . . . . . **321**
Example 10.1: Various EXM Model Specifications . . . . . 321


## Overview: HPFEXMSPEC Procedure

The HPFEXMSPEC procedure creates model specifications files for external models (EXM). External model specifications are used for forecasts that are provided external to the system. These external forecasts might have originated from an external statistical model from another software package, might have been provided by an outside organization (for example, a marketing organization or government agency), or might be based on judgment.

External forecasts might or might not provide prediction standard errors. If the prediction standard errors are not provided, they must be computed from the prediction errors and additional information. To properly compute the prediction standard errors, the autocovariances of model residuals and information about any transformations applied to the actual time series are needed. Since the autocovariances or transformations are not known to the system, this information must be specified by the user or approximated from the actual time series or the prediction errors.

External forecasts might or might not provide lower and upper confidence limits. If lower and upper confidence limits are not provided, they must be computed from the prediction standard errors.

The external model specification is a means by which the user can specify information about how external forecasts were created so that the prediction standard errors or confidence limits or both can be approximated when they are not provided with the external forecasts.

# Getting Started: HPFEXMSPEC Procedure

The following example shows how to create an external model specification file.

```
proc hpfexmspec repository=sasuser.mymodels
                name=myexternal
                label="My External Model";
   exm method=wn;
run;
```

The options in the PROC HPFEXMSPEC statement are used to specify the location of the specification file that will be output. Here the REPOSITORY= option specifies that the output file be placed in a catalog "sasuser.mymodels," the NAME= option specifies that the name of the file be "myexternal.xml," and the LABEL= option specifies a label for this catalog member. The EXM statement in the procedure specifies the external model and the options used to control the parameter estimation process for the model.

# Syntax: HPFEXMSPEC Procedure

The following statements are used with the HPFEXMSPEC procedure.

**PROC HPFEXMSPEC** *options* **;**
**EXM** *options* **;**

## Functional Summary

Table 10.1 summarizes the statements and options that control the HPFEXMSPEC procedure.

**Table 10.1** HPFEXMSPEC Functional Summary

| Description | Statement | Option |
|---|---|---|
| **Model Repository Options** | | |
| Specifies the model specification label | PROC HPFEXMSPEC | LABEL= |
| Specifies the model specification name | PROC HPFEXMSPEC | NAME= |
| Specifies the model repository | PROC HPFEXMSPEC | REPOSITORY= |
| **External Model Options** | | |
| Specifies median forecasts | EXM | MEDIAN |
| Specifies the method of creating forecast standard errors | EXM | METHOD= |
| Specifies the number of time lags used to compute the autocorrelations | EXM | NLAGPCT= |

| Description | Statement | Option |
|---|---|---|
| Specifies that the external model parameters be fixed values | EXM | NOEST |
| Specifies the number of parameters used to create the external forecast | EXM | NPARMS= |
| Specifies the prediction standard error for the external model | EXM | SIGMA= |
| Specifies the time series transformation | EXM | TRANSFORM= |

## PROC HPFEXMSPEC Statement

**PROC HPFEXMSPEC** *options* **;**

The following options can be used in the PROC HPFEXMSPEC statement.

**LABEL=***SAS-label*
labels the model specification to be stored in the SAS catalog or external file reference. The LABEL= option can also be specified as SPECLABEL=.

**NAME=***SAS-name*
names the model specification to be stored in the SAS catalog or external file reference. The NAME= option can also be specified as SPECNAME=.

**REPOSITORY=***SAS-catalog-name* | *SAS-file-reference*
names the SAS catalog or external file reference to contain the model specification. The REPOSITORY= option can also be specified as MODELREPOSITORY=, MODELREP=, or REP=.

## EXM Statement

**EXM** *options* **;**

The EXM statement specifies an external model that is used to generate the external forecasts. These options are not needed if the prediction standard errors and confidence limits are provided.

The following examples illustrate typical uses of the EXM statement.

```
/* default specification */
exm;

/* Actual Series Autocorrelation */
exm method=acf;

/* Prediction Error Autocorrelation */
exm method=erroracf;
```

The following options can be specified in the EXM statement.

**MEDIAN**
specifies that the median forecast values were used to generate the external forecasts. The external forecasts can have been based on the mean or median. By default the mean value is assumed. If no transformation was used by the external forecasting method, as specified by the TRANSFORM=NONE option, the mean and median prediction standard errors and confidence limits are identical.

**METHOD=***method-name*
specifies the external model to be used to approximate the prediction standard errors. The default is METHOD=ACF. The following forecasting models are provided:

| | |
|---|---|
| NONE | No prediction error autocorrelation is used. |
| WN | Prediction error autocorrelation is white noise. |
| ACF | Autocorrelation is used. |
| ERRORACF | Prediction error autocorrelation is used. |
| PERFECT | Perfect autocorrelation is assumed. |

**NLAGPCT=***number*
specifies the number of time lags as a percentage of the number of computed predictions errors. The default is NLAGPCT=0.25.

**NOEST**
specifies that the external model parameters are fixed values. To use this option, all of the external model parameters must be explicitly specified. By default, the external model parameters are optimized.

**NPARMS=***n*
specifies the number of parameters used by the external model to generate the forecasts. The default is NPARMS=0.

**SIGMA=***number*
specifies the prediction standard error for the external model. If the SIGMA= option is specified with the NOEST option, the prediction mean square error specified by the SIGMA= option is used. Otherwise, the prediction mean square error is computed from the prediction errors by using the NPARMS= option.

**TRANSFORM=***option*
specifies the time series transformation that was applied to the actual time series when generating the external forecast. The following transformations are provided:

| | |
|---|---|
| NONE | no transformation |
| LOG | logarithmic transformation |
| SQRT | square-root transformation |
| LOGISTIC | logistic transformation |
| BOXCOX(*n* ) | Box-Cox transformation with parameter number where number is between –5 and 5 |

When the TRANSFORM= option is specified, the actual time series must be strictly positive. After the time series is transformed, the model parameters are estimated using the transformed time series. The forecasts of the transformed time series are then computed, and finally the transformed time series forecasts are inverse transformed. The inverse transform produces either mean or median forecasts depending on whether the MEDIAN option is specified.

# Examples: HPFEXMSPEC Procedure

## Example 10.1: Various EXM Model Specifications

The following statements illustrate typical uses of the EXM statement:

```
proc hpfexmspec repository=mymodels
                name=model1
                label="Default Specification";
   exm;
run;

proc hpfexmspec repository=mymodels
                name=model2
                label="Actual Series Autocorrelation";
   exm method=acf;
run;

proc hpfexmspec repository=mymodels
                name=model3
                label="Prediction Error Autocorrelation";
   exm method=erroracf;
run;

title "Models Added to MYMODELS Repository";
proc catalog catalog=mymodels;
   contents;
run;
```

**Output 10.1.1** Listing of Models in MYMODELS Repository

**Models Added to MYMODELS Repository**

Contents of Catalog WORK.MYMODELS

| # | Name | Type | Create Date | Modified Date | Description |
|---|---|---|---|---|---|
| 1 | MODEL1 | XML | 03/26/2014 08:18:20 | 03/26/2014 08:18:20 | Default Specification |
| 2 | MODEL2 | XML | 03/26/2014 08:18:20 | 03/26/2014 08:18:20 | Actual Series Autocorrelation |
| 3 | MODEL3 | XML | 03/26/2014 08:18:20 | 03/26/2014 08:18:20 | Prediction Error Autocorrelation |

# Chapter 11
# The HPFIDMSPEC Procedure

**Contents**


Overview: HPFIDMSPEC Procedure . . . . . **323**
Getting Started: HPFIDMSPEC Procedure . . . . . **323**
Syntax: HPFIDMSPEC Procedure . . . . . **324**
Functional Summary . . . . . 324
PROC HPFIDMSPEC Statement . . . . . 325
IDM Statement . . . . . 326
Smoothing Model Suboptions for IDM Statement Options . . . . . 326
Details: HPFIDMSPEC Procedure . . . . . **330**
Smoothing Model Parameter Specification Options . . . . . 330
Smoothing Model Forecast Bounds Options . . . . . 331
Examples: HPFIDMSPEC Procedure . . . . . **331**
Example 11.1: Various Kinds of IDM Model Specifications . . . . . 331
Example 11.2: Automatically Choosing the Best Decomposed Demand Model . . . . 332


## Overview: HPFIDMSPEC Procedure

The HPFIDMSPEC procedure creates model specifications files for intermittent demand models (IDM).

You can specify many types of intermittent demand models with this procedure. In particular, any model that can be analyzed using the HPF procedure can be specified.

## Getting Started: HPFIDMSPEC Procedure

The following example shows how to create an intermittent demand model specification file. In this example, a model specification for Croston's method is created.

```
proc hpfidmspec repository=mymodels
                name=mycroston
                label="Croston Method";
   idm  interval=( method=simple )
        size=( method=simple ) ;
run;
```

The options in the PROC HPFIDMSPEC statement are used to specify the location of the specification file that will be output. Here the REPOSITORY= option specifies that the output file be placed in a catalog "sasuser.mymodels," the NAME= option specifies that the name of the file be "mycroston.xml," and the LABEL= option specifies a label for this catalog member. The IDM statement in the procedure specifies the intermittent demand model and the options used to control the parameter estimation process for the model.

# Syntax: HPFIDMSPEC Procedure

The following statements are used with the HPFIDMSPEC procedure.

**PROC HPFIDMSPEC** *options* **;**
**IDM** *options* **;**

## Functional Summary

Table 11.1 summarizes the statements and options that control the HPFIDMSPEC procedure.

**Table 11.1** HPFFIDMSPEC Functional Summary

| Description | Statement | Option |
|---|---|---|
| **Statements** | | |
| specifies the intermittent demand model | IDM | |
| **Model Repository Options** | | |
| Specifies the model specification label | PROC HPFIDMSPEC | LABEL= |
| Specifies the model specification name | PROC HPFIDMSPEC | NAME= |
| Specifies the model repository | PROC HPFIDMSPEC | REPOSITORY= |
| **Intermittent Demand Model Options** | | |
| Specifies the model for average demand | IDM | AVERAGE= |
| Specifies the base value | IDM | BASE= |
| Specifies the model for demand intervals | IDM | INTERVAL= |
| Specifies the model for demand sizes | IDM | SIZE= |

| Description | Statement | Option |
|---|---|---|
| **Exponential Smoothing Model Options** | | |
| Specifies the model selection criterion | IDM | CRITERION= |
| Specifies the damping weight parameter initial value | IDM | DAMPPARM= |
| Specifies the damping weight parameter restrictions | IDM | DAMPREST= |
| Specifies the level weight parameter initial value | IDM | LEVELPARM= |
| Specifies the level weight parameter restrictions | IDM | LEVELREST= |
| Specifies median forecasts | IDM | MEDIAN |
| Specifies the time series forecasting model | IDM | METHOD= |
| Specifies that the smoothing model parameters are fixed values | IDM | NOEST |
| Specifies that stable parameter estimates are not required | IDM | NOSTABLE |
| Specifies the time series transformation | IDM | TRANSFORM= |
| Specifies the trend weight parameter initial value | IDM | TRENDPARM= |
| Specifies the trend weight parameter restrictions | IDM | TRENDREST= |

## PROC HPFIDMSPEC Statement

**PROC HPFIDMSPEC** *options* **;**

The following options can be used in the PROC HPFIDMSPEC statement.

**LABEL=***SAS-label*
specified a descriptive label for the model specification to be stored in the SAS catalog or external file reference. The LABEL= option can also be specified as SPECLABEL=.

**NAME=***SAS-name*
names the model specification to be stored in the SAS catalog or external file reference. The NAME= option can also be specified as SPECNAME=.

**REPOSITORY=***SAS-catalog-name* | *SAS-file-reference*
names the SAS catalog or external file reference to contain the model specification. The REPOSITORY= option can also be specified as MODELREPOSITORY=, MODELREP=, or REP=.

## IDM Statement

**IDM** *options* **;**

The IDM statement is used to specify an intermittent demand model. An intermittent demand series can be analyzed in two ways: individually modeling both demand interval and size component, or jointly modeling these components by using the average demand component (demand size divided by demand interval). The IDM statement specifies the two smoothing models to be used to forecast the demand interval component (INTERVAL= option) and the demand size component (SIZE= option), or specifies the single smoothing model to be used to forecast the average demand component (AVERAGE= option). Therefore, two smoothing models (INTERVAL= and SIZE= options) must be specified or one smoothing model (AVERAGE= option) must be specified. Only one IDM statement can be specified.

The following options can be specified in the IDM statement.

**AVERAGE=(** *smoothing-model-options* **)**
specifies the smoothing model used to forecast the demand average component. See the section "Smoothing Model Suboptions for IDM Statement Options" on page 326.

**BASE=AUTO** | *number*
specifies the base value of the time series used to determine the demand series components. The demand series components are determined based on the departures from this base value. If a base value is specified, this value is used to determine the demand series components. If BASE=AUTO is specified, the time series properties are used to automatically adjust the time series. For the common definition of Croston's method, use BASE=0, which defines departures based on zero. The default is BASE=AUTO.

Given a time series $y_t$ and base value $b$ the time series is adjusted by the base value to create the base adjusted time series, $x_t = y_t - b$. Demands are assumed to occur when the base adjusted series is nonzero (or when the time series $y_t$ departs from the base value $b$).

When BASE=AUTO, the base value is automatically determined by the time series median, minimum, and maximum value and the INTERMITTENT= option value of the FORECAST statement.

**INTERVAL=(** *smoothing-model-options* **)**
specifies the smoothing model used to forecast the demand interval component. See the section "Smoothing Model Suboptions for IDM Statement Options" on page 326.

**SIZE=(** *smoothing-model-options* **)**
specifies the smoothing model used to forecast the demand size component. See the section "Smoothing Model Suboptions for IDM Statement Options" on page 326.

## Smoothing Model Suboptions for IDM Statement Options

The smoothing model options are specified as suboptions of the INTERVAL= option, SIZE= option, and AVERAGE= options. They describe how to forecast the demand interval, demand size, and average demand components, respectively.

The following describes the smoothing model options.

**BOUNDS=(***number***,** *number* **)**
specifies the component forecast bound. See the section "Smoothing Model Parameter Specification Options" on page 330.

**CRITERION=***option*

**SELECT=***option*
specifies the model selection criterion (statistic of fit) to be used to select from several candidate models. This option is often used in conjunction with the HOLDOUT= option specified in the FORECAST statement. The CRITERION= option can also be specified as SELECT=. The default is CRITERION=RMSE.

The following list shows the valid values for the CRITERION= option and the statistics of fit these option values specify:

| | |
|---|---|
| SSE | sum of square error |
| MSE | mean square error |
| RMSE | root mean square error |
| UMSE | unbiased mean square error |
| URMSE | unbiased root mean square error |
| MAXPE | maximum percent error |
| MINPE | minimum percent error |
| MPE | mean percent error |
| MAPE | mean absolute percent error |
| MDAPE | median absolute percent error |
| GMAPE | geometric mean absolute percent error |
| MAPES | mean absolute error percent of standard deviation |
| MDAPES | median absolute error percent of standard deviation |
| GMAPES | geometric mean absolute error percent of standard deviation |
| MINPPE | minimum predictive percent error |
| MAXPPE | maximum predictive percent error |
| MPPE | mean predictive percent error |
| MAPPE | symmetric mean absolute predictive percent error |
| MDAPPE | median absolute predictive percent error |
| GMAPPE | geometric mean absolute predictive percent error |
| MINSPE | minimum symmetric percent error |
| MAXSPE | maximum symmetric percent error |
| MSPE | mean symmetric percent error |
| SMAPE | symmetric mean absolute percent error |

| | |
|---|---|
| MDASPE | median absolute symmetric percent error |
| GMASPE | geometric mean absolute symmetric percent error |
| MINRE | minimum relative error |
| MAXRE | maximum relative error |
| MRE | mean relative error |
| MRAE | mean relative absolute error |
| MDRAE | median relative absolute error |
| GMRAE | geometric mean relative absolute error |
| MAXERR | maximum error |
| MINERR | minimum error |
| ME | mean error |
| MAE | mean absolute error |
| MASE | mean absolute scaled error |
| RSQUARE | R-square |
| ADJRSQ | adjusted R-square |
| AADJRSQ | Amemiya's adjusted R-square |
| RWRSQ | random walk R-square |
| AIC | Akaike information criterion |
| AICC | finite sample corrected AIC |
| SBC | Schwarz Bayesian information criterion |
| APC | Amemiya's prediction criterion |

**DAMPPARM=***number*

specifies the damping weight parameter initial value. See the section "Smoothing Model Parameter Specification Options" on page 330.

**DAMPREST=(***number***,** *number* **)**

specifies the damping weight parameter restrictions. SSee the section "Smoothing Model Parameter Specification Options" on page 330.

**LEVELPARM=***number*

specifies the level weight parameter initial value. See the section "Smoothing Model Parameter Specification Options" on page 330.

**LEVELREST=(***number***,** *number* **)**

specifies the level weight parameter restrictions. See the section "Smoothing Model Parameter Specification Options" on page 330.

**MEDIAN**

specifies that the median forecast values are to be estimated. Forecasts can be based on the mean or median. By default the mean value is provided. If no transformation is applied to the actual series with the TRANSFORM= option, the mean and median component forecast values are identical.

**METHOD=***method-name*

specifies the forecasting model to be used to forecast the demand component. A single model can be specified, or a group of candidate models can be specified. If a group of models is specified, the model used to forecast the accumulated time series is selected based on the CRITERION= option of the IDM statement and the HOLDOUT= option of the FORECAST statement. The default is METHOD=BESTN. The following forecasting models are provided:

| | |
|---|---|
| SIMPLE | simple (single) exponential smoothing |
| DOUBLE | double (Brown) exponential smoothing |
| LINEAR | linear (Holt) exponential smoothing |
| DAMPTREND | damped trend exponential smoothing |
| BESTN | best candidate nonseasonal smoothing model (SIMPLE, LINEAR, DAMPTREND) |

**NOEST**

specifies that the smoothing model parameters are fixed values. To use this option, all of the smoothing model parameters must be explicitly specified. By default, the smoothing model parameters are optimized.

**NOSTABLE**

specifies that the smoothing model parameters are not restricted to the additive invertible region of the parameter space. By default, the smoothing model parameters are restricted to be inside the additive invertible region of the parameter space.

**TRANSFORM=***option*

specifies the time series transformation to be applied to the demand component. The following transformations are provided:

| | |
|---|---|
| NONE | no transformation |
| LOG | logarithmic transformation |
| SQRT | square-root transformation |
| LOGISTIC | logistic transformation |
| BOXCOX(*n* ) | Box-Cox transformation with parameter number where number is between –5 and 5 |
| AUTO | Automatically choose between NONE and LOG based on model selection criteria. This option is the default. |

When the TRANSFORM= option is specified, the demand component must be strictly positive. After the demand component is transformed, the model parameters are estimated using the transformed component. The forecasts of the transformed component are then computed, and finally the transformed component forecasts are inverse transformed. The inverse transform produces either mean or median forecasts depending on whether the MEDIAN option is specified.

**TRENDPARM=***number*

specifies the trend weight parameter initial value. See the section "Smoothing Model Parameter Specification Options" on page 330.

**TRENDREST=(***number***,** *number* **)**
specifies the trend weight parameter restrictions. See the section "Smoothing Model Parameter Specification Options" on page 330.

The following shows the defaults values of the smoothing model options that are used when the options are not specified.

For the demand interval component the defaults are:

```
interval=( transform=auto
           method=bestn
           levelrest=(0.001 0.999)
           trendrest=(0.001 0.999)
           damprest =(0.001 0.999)
           criterion=rmse
           bounds=(1,.)
         )
```

For the demand size component the defaults are:

```
size=(    transform=auto
          method=bestn
          levelrest=(0.001 0.999)
          trendrest=(0.001 0.999)
          damprest =(0.001 0.999)
          criterion=rmse
        )
```

For the average demand component the defaults are:

```
average=( transform=auto
          method=bestn
          levelrest=(0.001 0.999)
          trendrest=(0.001 0.999)
          damprest =(0.001 0.999)
          criterion=rmse
        )
```

# Details: HPFIDMSPEC Procedure

## Smoothing Model Parameter Specification Options

The parameter options are used to specify smoothing model parameters. If the parameter restrictions are not specified the default is (0.001 0.999), which implies that the parameters are restricted between 0.001 and 0.999. Parameters and their restrictions are required to be greater than or equal to –1 and less than or equal to 2. Missing values indicate no lower and/or upper restriction. If the parameter initial values are not specified, the optimizer uses a grid search to find an appropriate initial value.

## Smoothing Model Forecast Bounds Options

Specifies the demand component forecast bounds. The forecast bounds restrict the component forecasts. The default for demand interval forecasts is BOUNDS=1. The lower bound for the demand interval forecast must be greater than one. The default for demand size forecasts is BOUNDS=(.,.) or no bounds. The demand size forecasts bounds are applied after the forecast is adjusted by the base value.

# Examples: HPFIDMSPEC Procedure

## Example 11.1: Various Kinds of IDM Model Specifications

The following statements illustrate typical uses of the IDM statement:

```
proc hpfidmspec repository=mymodels
                name=model1
                label="Default Specification";
   idm;
run;

proc hpfidmspec repository=mymodels
                name=model2
                label="Demand Interval model only specification";
   idm interval=( transform=log );
run;

proc hpfidmspec repository=mymodels
                name=model3
                label="Demand Size model only specification";
   idm size=( method=linear );
run;

proc hpfidmspec repository=mymodels
                name=model4
                label="Croston's Method";
   idm interval=( method=simple )
       size    =( method=simple );
run;

proc hpfidmspec repository=mymodels
                name=model5
                label="Log Croston's Method";
   idm interval=( method=simple transform=log )
       size    =( method=simple transform=log );
run;

proc hpfidmspec repository=mymodels
                name=model6
                label="Average demand model specification";
   idm average=(method=bestn);
run;
```

```
title "Models Added to MYMODELS Repository";
proc catalog catalog=mymodels;
   contents;
run;
```

The default specification uses both the INTERVAL= option and SIZE= option defaults for the decomposed (Croston's) demand model and the AVERAGE= option defaults for the average demand model.

The models added to the model repository are shown in Output 11.1.1.

**Output 11.1.1** Listing of Models in MYMODELS repository

**Models Added to MYMODELS Repository**

Contents of Catalog WORK.MYMODELS

| # | Name | Type | Create Date | Modified Date | Description |
|---|---|---|---|---|---|
| 1 | MODEL1 | XML | 03/26/2014 08:20:40 | 03/26/2014 08:20:40 | Default Specification |
| 2 | MODEL2 | XML | 03/26/2014 08:20:40 | 03/26/2014 08:20:40 | Demand Interval model only specification |
| 3 | MODEL3 | XML | 03/26/2014 08:20:40 | 03/26/2014 08:20:40 | Demand Size model only specification |
| 4 | MODEL4 | XML | 03/26/2014 08:20:40 | 03/26/2014 08:20:40 | Croston's Method |
| 5 | MODEL5 | XML | 03/26/2014 08:20:40 | 03/26/2014 08:20:40 | Log Croston's Method |
| 6 | MODEL6 | XML | 03/26/2014 08:20:40 | 03/26/2014 08:20:40 | Average demand model specification |
| 7 | MYCROSTON | XML | 03/26/2014 08:20:39 | 03/26/2014 08:20:39 | Croston Method |

## Example 11.2: Automatically Choosing the Best Decomposed Demand Model

This example illustrates how to automatically choose the decomposed demand model by using mean absolute percent error (MAPE) as the model selection criterion.

```
proc hpfidmspec repository=mymodels
                name=auto1
                label="Automatically Selected Best IDM Model";
   idm interval=( method=simple transform=auto criterion=mape )
       size    =( method=simple transform=auto criterion=mape );
run;
```

The preceding model statements cause PROC HPFENGINE to fit two forecast models (simple and log simple exponential smoothing) to both the demand interval and size components. The forecast model that results in the lowest in-sample MAPE for each component is used to forecast the component.

The following statements illustrate how to automatically choose the average demand model by using MAPE as the model selection criterion.

```
proc hpfidmspec repository=mymodels
                name=auto2
                label="Automatically Selected Best IDM Model";
   idm average=( method=simple transform=auto criterion=mape );
run;
```

The preceding statements cause PROC HPFENGINE to fit two forecast models (simple and log simple exponential smoothing) to the average demand component. The forecast model that results in the lowest in-sample MAPE is used to forecast the component.

Combining the preceding two examples, the following example illustrates how to automatically choose between the decomposed demand model and the average demand model by using MAPE as the model selection criterion:

```
proc hpfidmspec repository=mymodels
                name=auto3
                label="Automatically Selected Best IDM Model";
   idm interval=( method=simple transform=auto criterion=mape )
       size    =( method=simple transform=auto criterion=mape )
       average =( method=simple transform=auto criterion=mape );
run;
```

The preceding model specification causes PROC HPFENGINE to automatically select between the decomposed demand model and the average demand model as described previously. The forecast model that results in the lowest in-sample MAPE is used to forecast the series.

# Chapter 12
# The HPFORECAST Procedure

**Contents**


Overview: HPFORECAST Procedure . . . . . **337**
Getting Started: HPFORECAST Procedure . . . . . **339**
Syntax: HPFORECAST Procedure . . . . . **342**
Functional Summary . . . . . 343
PROC HPFORECAST Statement . . . . . 344
DISTBY Statement . . . . . 350
END LEVEL Statement . . . . . 351
FORECAST Statement . . . . . 351
INPUT Statement . . . . . 352
LEVEL Statement . . . . . 354
OUTPUT Statement . . . . . 355
PERFORMANCE Statement . . . . . 356
SET Statement . . . . . 359
TIMEID Statement . . . . . 360
Details: HPFORECAST Procedure . . . . . **363**
Parallel Time Series Processing . . . . . 363
High-Performance Analytics: Concepts and Topics for PROC HPFORECAST . . . . 364
Single-Machine Mode . . . . . 364
Distributed Mode . . . . . 365
Symmetric and Asymmetric Distributed Modes . . . . . 365
Controlling the Execution Mode with Environment Variables and Performance Statement Options . . . . . 366
Determining Single-Machine Mode or Distributed Mode . . . . . 367
Alongside-the-Database Execution . . . . . 368
Alongside-LASR Distributed Execution . . . . . 369
Running PROC HPFORECAST Alongside a SAS LASR Analytic Server in Distributed Mode . . . . . 369
Running PROC HPFORECAST in Asymmetric Mode . . . . . 371
Alongside-HDFS Execution . . . . . 372
Output Data Sets . . . . . 374
Data Preparation and Staging . . . . . 374
Threading . . . . . 376
Output Data Considerations . . . . . 380
Time Series Acquisition Details . . . . . 380
Accumulation . . . . . 381
Missing Value Interpretation . . . . . 382

Time Indexing in the Forecasting Process . . . 382
Differences between MODE=SIMPLE and MODE=AUTO . . . 384
Caveats: MODE=AUTO versus PROC HPFDIAGNOSE and PROC HPFENGINE . . . 385
Conceptual View of the Forecasting Process . . . 386
Model Diagnosis . . . 386
Model Selection . . . 387
Parameter Estimation . . . 388
Missing Value Modeling Issues . . . 388
Forecasting . . . 388
Inverse Transformations . . . 389
Statistics of Fit . . . 389
Output Data Set Details . . . 389
OUTEST= Data Set . . . 389
OUTFOR= Data Set . . . 390
OUTINDEP= Data Set . . . 391
OUTLOG= Data Set . . . 391
OUTMODELINFO= Data Set . . . 391
OUTSTAT= Data Set . . . 393
OUTSTATSELECT= Data Set . . . 395
OUTSUM= Data Set . . . 395
Printed Output . . . 396
PERFORM . . . 396
SUMMARY . . . 396
THREADS . . . 398
TIMING . . . 399
ODS Table Names . . . 399
Examples: HPFORECAST Procedure . . . **399**
Example 12.1: Automatic Forecasting of Time Series Data . . . 399
Example 12.2: Automatic Forecasting of Transactional Data . . . 401
Example 12.3: Forecasting Hierarchically Structured Data . . . 403
Example 12.4: Forecasting Intermittent Time Series Data . . . 404
Example 12.5: Using the DISTBY Statement in PROC HPFORECAST . . . 407
Example 12.6: Using PROC HPFORECAST with the Greenplum DBMS . . . 409
Example 12.7: Automatic Modeling with Greenplum . . . 412
Example 12.8: Using Automatic Modeling and User-Defined Models . . . 413
References . . . **416**

# Overview: HPFORECAST Procedure

The HPFORECAST procedure is a high-performance analytics procedure that forecasts time series. Like other high-performance analytics procedures, PROC HPFORECAST can either run on one machine (called single-machine mode execution) or take advantage of a distributed computing environment (called distributed mode).

**NOTE:** Distributed mode requires an additional license.

Distributed mode enables PROC HPFORECAST to distribute the forecasting task among one or more nodes. The time series data set must be partitioned by a set of variables that are used to discriminate the hierarchy levels of interest, which are then distributed to the nodes. In addition, each node can use multiple threads to carry out parallel operations on the time series data for the hierarchy levels that have been distributed to that node.

The HPFORECAST procedure combines the functionality of several other procedures. For simple forecasting needs, PROC HPFORECAST functions like PROC HPF, which uses only exponential smoothing models to perform forecasting. For complex forecasting needs, PROC HPFORECAST embodies the functionality of the HPFDIAGNOSE and HPFENGINE procedures to perform automatic time series model identification, model selection, and forecast generation.

The HPFORECAST procedure is designed to efficiently process large-scale, hierarchically structured, time-stamped data sets. The HPF, HPFDIAGNOSE, and HPFENGINE procedures can run only on one machine, but the HPFORECAST procedure can run in either single-machine mode or in distributed mode. Like the HPF, HPFDIAGNOSE, and HPFENGINE procedures, the HPFORECAST procedure can automatically generate time series from the time-stamped data and produce forecasts for them in one step. PROC HPFORECAST can process multiple hierarchy levels in one pass of the input data set, whereas the HPF, HPFDIAGNOSE, and HPFENGINE procedures can process only one hierarchy level (BY statement) in one pass of the input data set.

In order for the HPFORECAST procedure to execute in distributed mode, the time series data set must be partitioned by some set of variables that are used to discriminate hierarchy levels of interest and are distributed to the nodes. Each node can use multiple threads to carry out parallel operations on the time series for the hierarchy levels that are materialized on that node.

These parallel operations include the following:

- time series acquisition from the store
- time series model identification and selection
- time series forecasting for the selected model

Distributed processing that is applied to properly structured subsets of the time series data set coupled with symmetric multiprocessing (threading) on each node results in dramatic gains in performance for large-scale time series data sets. In distributed database management system (DBMS) environments, output data sets that are produced from the HPFORECAST procedure can be written back directly to the DBMS facility with maximum efficiency.

For more information about processing modes for PROC HPFORECAST, see the section "High-Performance Analytics: Concepts and Topics for PROC HPFORECAST" on page 364.

For simple applications, the time series modeling features of the HPFORECAST procedure include the following:

- For typical time series, the HPFORECAST procedure automatically chooses the best performing forecast from among the following smoothing models:
  - simple
  - linear
  - damped trend
  - seasonal (additive and multiplicative)
  - Winters method (additive and multiplicative)
- For intermittent time series (series in which a large number of values are zero-valued), the HPFORECAST procedure can test the series for intermittency. For such series, the HPFORECAST procedure automatically selects between the best performing forecast from Croston's method and the average demand model.

For complex applications, the time series modeling features of the HPFORECAST procedure include the following:

- automatic model identification from different model families that include ARIMAX models, exponential smoothing models (ESMs),and unobserved component models (UCMs), intermittent demand models (IDMs), and transformed versions of these model families
- automatic model combinations for those generated models
- mixing of user-defined (custom) time series models and combinations with automatically generated time series models and combinations
- automatic model selection that incorporates both automatically generated models and combinations with user-defined models and combinations
- automatic generation of forecasts for explanatory variables that are used in time series models and combinations

All parameters that are associated with the forecast models are optimized based on the historical data. Optionally, the HPFORECAST procedure can select the appropriate model for you by using holdout sample analysis that is based on one of several model selection criteria.

The HPFORECAST procedure writes the following information to output data sets:

- series summary statistics
- forecasts, prediction standard errors, and confidence limits
- parameter estimates
- final forecast fit statistics

- model selection fit statistics
- model execution logging
- ancillary information used by and produced by the automatic forecasting process

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

## Getting Started: HPFORECAST Procedure

The HPFORECAST procedure is both simple to use for someone who is new to forecasting and yet a powerful tool for the experienced professional forecaster who needs to automatically generate a large number of forecasts. It can provide results in output data sets that can be postprocessed by the SAS System, or in DBMS tables that can be accessed by a variety of software including the SAS System.

The following examples are more fully illustrated in the section "Examples: HPFORECAST Procedure" on page 399.

If you provide an input data set that contains numerous time series variables that are recorded at a specific frequency, the HPFORECAST procedure can automatically forecast the series as follows:

```
PROC HPFORECAST DATA=<input-data-set>;
   OUTPUT OUTFOR=<output-data-set>;
   OUTPUT OUTLOG=<log-data-set>;
   TIMEID <time-ID-variable> INTERVAL=<frequency>;
   FORECAST <time-series-variables>;
RUN;
```

For example, suppose that the input data set Sales contains numerous sales observations that are recorded by using the corresponding month of the sale as a SAS DATE value in the variable Date:

```
proc hpforecast data=sales;
   output outfor=nextyear;
   output outlog=errors;
   timeid date interval=month;
   forecast _ALL_;
run;
```

The preceding statements automatically select the best fitting model, generate forecasts for every numeric variable in the input data set (Sales) for the next 12 months, and store these forecasts in the output data set (Nextyear). The OUTLOG= option in the second OUTPUT statement names the data set Errors where PROC HPFORECAST will store model-specific or data-related error synopses for the time series that it processes. This data set replaces the use of the SAS log to report data-related and model-related failures that occur during the processing of time series data sets. The real benefits of this approach become obvious for large-scale, hierarchically-structured time series projects. Errors that halt the execution of PROC HPFORECAST are still

reported to the SAS log. You can specify other output data sets for storing the model parameter estimates, forecasts, fit statistics, and time series summary statistics.

If you provide an input data set that contains transactional variables that are not recorded at any specific frequency, the HPFORECAST procedure accumulates the data to a specific time interval and forecasts the accumulated series as follows:

```
PROC HPFORECAST DATA=<input-data-set>;
   OUTPUT OUTFOR=<output-data-set>;
   TIMEID <time-ID-variable> INTERVAL=<frequency>
          ACCUMULATE=<accumulation>;
   FORECAST <time-series-variables>;
RUN;
```

For example, the following statements create a forecast that is based on input data set, Websites, which contains three variables (Boats, Cars, and Planes) and the variable Time, which records the time of the website visit as a SAS DATETIME value:

```
proc hpforecast data=websites lead=7;
   output outfor=nextweek
          outlog=errors;
   timeid time interval=dtday accumulate=total;
   forecast boats cars planes;
run;
```

The preceding statements accumulate the data into a daily time series, automatically generate forecasts for the Boats, Cars, and Planes variables in the input data set (Websites) for the next seven days, and store the forecasts in the output data set (Nextweek).

As an introduction to the concept of hierarchically structured time series, consider viewing the same website data in a different way. Suppose that there is a variable Topic in each row of the data set Websites and its value is set as BOAT, CAR, or PLANE accordingly, and suppose that a variable Hits records the number of website visits that are associated with the Topic and timestamp values. You can use the Topic variable can be used to define a simple, one-level hierarchy for the Websites data set. When you define the data set this way, you can produce a forecast for the TOPIC level and the _TOP_ level in a single pass of the data set. As the name implies, the _TOP_ level aggregates all the observations for the BOAT, CAR, and PLANE values of the Topic variable into a single time series. The following statements use this approach to forecast the website data:

```
proc hpforecast data=websites lead=7;
   timeid time interval=dtday accumulate=total;
   forecast hits;
   output outfor=nextweek
          outlog=errors;
   level _top_;
   end level;
   level topic;
   end level;
run;
```

The OUTFOR= data set, Nextweek, has an extra variable Topic. The forecasts for the variable Hits are stored in the data set Nextweek and are qualified by the value of the Topic variable. Rows that correspond to the TOPIC level in the hierarchy have the value BOAT, CAR, or PLANE in the Topic variable. Rows that correspond to the _TOP_ level are qualified by setting the value of the Topic variable to blank.

You can use full automation of the HPFORECAST procedure on the previous hierarchically structured website data by specifying the MODE=AUTO option in the PROC HPFORECAST statement. This addition alone causes the HPFORECAST procedure to perform automatic model identification over a predefined set of time series model families, then to perform model selection over the models generated, and then to produce the final forecast for the selected model. This process happens independently for each level member of the requested hierarchies. The following statements use this approach to forecast the website data by using the default settings for the model families to be considered. For more information comparing MODE=SIMPLE and MODE=AUTO, see the section "Differences between MODE=SIMPLE and MODE=AUTO" on page 384.

```
proc hpforecast data=websites lead=7 mode=auto;
   timeid time interval=dtday accumulate=total;
   forecast hits;
   output outfor=nextweek
          outmodelinfo=modinfo
          outstatselect=modselect
          outlog=errors;
   level _top_;
   end level;
   level topic;
   end level;
run;
```

The OUTFOR= data set (Nextweek) has the same structure as in the previous PROC HPFORECAST example; however, the forecasts stored in it are produced by automatic model generation and selection process as previously described. When you specify MODE=AUTO, you can specify some additional output data sets to store results from the automatic forecasting process. The OUTSTATSELECT= data set (Modselect) stores the fit statistics for each of the models that are evaluated for each series during model selection. The OUTMODELINFO= data set (Modinfo) stores information about the model that is selected for each series. The OUTSTATSELECT= and OUTMODELINFO= data sets are very similar to the corresponding data sets from PROC HPFENGINE.

The HPFORECAST procedure can also forecast intermittent time series (series in which a large number of values are constant-valued). Traditional time series forecasting techniques are less effective in forecasting intermittent time series. Furthermore, these intermittent series might be interspersed with other series that exhibit continuous demand patterns over time, either at the same level or at higher levels of aggregation in the hierarchy, for which more conventional time series forecasting techniques are best. For example, suppose that the input data set Inventory contains a Product variable that has possible values TIRES, HUBCAPS, and LUGBOLTS, and that the demand for these products is recorded in a variable Demand. The time for each demand is recorded in the Date variable as a SAS DATE value. The demand series for products TIRES, HUBCAPS, and LUGBOLTS are to be tested for intermittency such that any series that has a median time between demands of two weeks or more is classified as intermittent. If the series is deemed intermittent, then PROC HPFORECAST compares forecasts from Croston's method and the average demand model, and uses the best performing model to produce the final demand forecast. If the demand series is not deemed intermittent, then PROC HPFORECAST uses the best performing exponential smoothing model to produce the final demand forecast. In either case, suppose that root mean squared error (RMSE) is the metric of choice for selecting the best performing forecast. The following statements forecast demand for the Product level and the _TOP_ level of the Inventory data set:

```
proc hpforecast data=inventory lead=4 intermittent=2;
   timeid date interval=week accumulate=total;
   forecast demand;
   output outfor=nextmonth / nooutall;
   output outlog=errors;
   level _top_;
   end;
   level product;
   end level;
run;
```

The preceding statements accumulate the total demand for inventory items on a weekly basis, and generate forecasts that recommend future stocking levels for four weeks ahead. The NOOUTALL option requests that only the forecasts for those four future weeks be stored the output data set Nextmonth. Of course, the demand forecasts at the _TOP_ level are devoid of product-specific information and would not be useful for stocking individual products, but they could provide information useful in other contexts for staffing, operating expense projections, and so on. You might also expect to see that filling in of demands occurs over the time index span at the _TOP_ level which naturally leads to a series that is nonintermittent in nature (very few zero-valued observations remain in the _TOP_ level series).

To include automatic model identification that includes intermittent demand models, simply add MODE=AUTO to the PROC HPFORECAST statement in this example.

## Syntax: HPFORECAST Procedure

The following statements are available in the HPFORECAST procedure:

**PROC HPFORECAST** *options* **;**
**TIMEID** *variable* **INTERVAL**=*interval* < *options* > **;**
**OUTPUT** *options* / *qualifiers* **;**
**FORECAST** *variable-list* / *options* **;**
**PERFORMANCE** < *performance-options* > **;**
**DISTBY** *variable-list* **;**
**LEVEL** *level-variables* / *options* **;**
**END LEVEL** **;**
**INPUT** *variable-list* / *options* **;**
**SET** *variable.property=type(value)* **;**

PROC HPFORECAST statements must adhere to the following precedence in the procedure statement block:

- You must specify the PROC HPFORECAST statement first.
- You can specify the TIMEID, FORECAST, INPUT, OUTPUT, DISTBY, or PERFORMANCE statements in any order after the PROC HPFORECAST statement and before the first LEVEL statement. You must specify a single TIMEID statement, and you must specify at least one FORECAST statement.
- You can specify one or more blocks that start with a LEVEL statement and end with an END LEVEL statement. If no LEVEL/END LEVEL blocks are specified, a default level of _TOP_ is generated.

- You can include SET statements in the context of LEVEL/END LEVEL blocks to set level-specific properties for the FORECAST and INPUT variables.

## Functional Summary

Table 12.1 summarizes the statements and options that control the HPFORECAST procedure.

**Table 12.1** PROC HPFORECAST Functional Summary

| Description | Statement | Option |
|---|---|---|
| **Statements** | | |
| Specifies hierarchy level | LEVEL | |
| Specifies variables to forecast | FORECAST | |
| Specifies the time ID variable | TIMEID | |
| **Data Set Options** | | |
| Specifies the input data set | PROC HPFORECAST | DATA= |
| Outputs forecasts only | OUTPUT | NOOUTALL |
| Specifies the parameter output data set | OUTPUT | OUTEST= |
| Specifies the forecast output data set | OUTPUT | OUTFOR= |
| Specifies the output data set to store independent variable time series | OUTPUT | OUTINDEP= |
| Specifies the error log output data set | OUTPUT | OUTLOG= |
| Specifies the detailed model information output data set | OUTPUT | OUTMODELINFO= |
| Specifies the statistics output data set | OUTPUT | OUTSTAT= |
| Specifies the selection statistics output data set | OUTPUT | OUTSTATSELECT= |
| Specifies the summary output data set | OUTPUT | OUTSUM= |
| **Accumulation and Seasonality Options** | | |
| Specifies the accumulation frequency | TIMEID | INTERVAL= |
| Specifies the length of seasonal cycle | PROC HPFORECAST | SEASONALITY= |
| Specifies the interval alignment | TIMEID | ALIGN= |
| Specifies the starting time ID value | TIMEID | START= |
| Specifies the ending time ID value | TIMEID | END= |
| Specifies the accumulation statistic | TIMEID, FORECAST | ACCUMULATE= |
| Specifies how missing values are to be interpreted | TIMEID, FORECAST | SETMISSING= |
| Specifies special zero value replacement semantics | TIMEID, FORECAST | ZEROMISS= |
| **Forecasting Control Options** | | |
| Specifies the forecast confidence limit width | PROC HPFORECAST | ALPHA= |
| Specifies the model selection criterion | PROC HPFORECAST | CRITERION= |
| Specifies intermittency threshold | PROC HPFORECAST | INTERMITTENT= |

| Description | Statement | Option |
|---|---|---|
| Specifies automatic forecasting mode | PROC HPFORECAST | MODE= |
| **Automatic Forecasting Options** | | |
| Specifies global diagnostic control specification | PROC HPFORECAST | GLOBALDIAGNOSE= |
| Specifies global custom model list | PROC HPFORECAST | GLOBALSELECTION= |
| Specifies automatic forecasting mode | PROC HPFORECAST | MODE= |
| Specifies location of model repository | PROC HPFORECAST | MODELREPOSITORY= |
| Specifies forecast variable custom model list | FORECAST | SELECTION= |
| Specifies forecast variable diagnostic controls | FORECAST | DIAGNOSE= |
| Specifies variable's level-specific properties | SET | |
| Specifies that automatic model generation be suppressed | PROC HPFORECAST | NODIAGNOSE |
| **Forecasting Horizon, Holdout, Holdback Options** | | |
| Specifies data to hold back | PROC HPFORECAST | BACK= |
| Specifies the forecast holdout sample size | PROC HPFORECAST | HOLDOUT= |
| Specifies the forecast holdout sample percent | PROC HPFORECAST | HOLDOUTPCT= |
| Specifies the forecast horizon or lead | PROC HPFORECAST | LEAD= |
| Specifies the forecast horizon starting time | TIMEID | HORIZONSTART= |
| **Printing Control Options** | | |
| Specifies the time ID format | TIMEID | FORMAT= |
| Specifies types of printed output | PROC HPFORECAST | PRINT= |
| Specifies that detailed printed output is desired | PROC HPFORECAST | PRINTDETAILS |
| **Miscellaneous Options** | | |
| Control over level processing | PROC HPFORECAST | NOFORECAST |

## PROC HPFORECAST Statement

**PROC HPFORECAST** *options* **;**

You can specify the following *options* in the PROC HPFORECAST statement.

**ALPHA=***number*

specifies the significance level to use in computing the confidence limits of the forecast, where *number* must be between 0 and 1. For example, ALPHA=0.05 produces 95% confidence intervals.

**BACK=***n*

specifies the number of observations before the end of the data where the multistep forecasts are to begin. The default is BACK=0.

**CRITERION=***option*

specifies the model selection criterion (statistic of fit) to be used to select from several candidate models. The following list shows the valid values for *option* and their corresponding statistics of fit:

| | |
|---|---|
| **SSE** | sum of square error |
| **MSE** | mean square error |
| **RMSE** | root mean squared error |
| **UMSE** | unbiased mean squared error |
| **URMSE** | unbiased root mean squared error |
| **MAXPE** | maximum percent error |
| **MINPE** | minimum percent error |
| **MPE** | mean percent error |
| **MAPE** | mean absolute percent error |
| **MDAPE** | median absolute percent error |
| **GMAPE** | geometric mean absolute percent error |
| **MINAPES** | minimum absolute error percent of standard deviation |
| **MAXAPES** | maximum absolute error percent of standard deviation |
| **MAPES** | mean absolute error percent of standard deviation |
| **MDAPES** | median absolute error percent of standard deviation |
| **GMAPES** | geometric mean absolute error percent of standard deviation |
| **MINPPE** | minimum predictive percent error |
| **MAXPPE** | maximum predictive percent error |
| **MPPE** | mean predictive percent error |
| **MAPPE** | symmetric mean absolute predictive percent error |
| **MDAPPE** | median absolute predictive percent error |
| **GMAPPE** | geometric mean absolute predictive percent error |
| **MINSPE** | minimum symmetric percent error |
| **MAXSPE** | maximum symmetric percent error |
| **MSPE** | mean symmetric percent error |
| **SMAPE** | symmetric mean absolute percent error |
| **MDASPE** | median absolute symmetric percent error |
| **GMASPE** | geometric mean absolute symmetric percent error |
| **MINRE** | minimum relative error |
| **MAXRE** | maximum relative error |
| **MRE** | mean relative error |
| **MRAE** | mean relative absolute error |

| | |
|---|---|
| **MDRAE** | median relative absolute error |
| **GMRAE** | geometric mean relative absolute error |
| **MAXERR** | maximum error |
| **MINERR** | minimum error |
| **ME** | mean error |
| **MAE** | mean absolute error |
| **MASE** | mean absolute scaled error |
| **RSQUARE** | R square |
| **ADJRSQ** | adjusted R square |
| **AADJRSQ** | Amemiya's adjusted R square |
| **RWRSQ** | random walk R square |
| **AIC** | Akaike information criterion |
| **AICC** | finite sample corrected AIC |
| **SBC** | Schwarz Bayesian information criterion |
| **APC** | Amemiya's prediction criterion |

For more information about the mathematical details for these fit statistics, see the section "Statistics of Fit" on page 600.

**DATA=***SAS-data-set*

names the SAS data set that contains the input data for the procedure to forecast. If the DATA= option is not specified, the most recently created SAS data set is used.

**ENDZEROS=(MAXNUM=***number* **MAXPCT=***percent* **MINOBS=***number***)**

specifies criteria that control the use of a _ZERO_ model for a series. A _ZERO_ model is used when the number of trailing zero values in the series fails to indicate that a nonzero model is appropriate. (By default, no _ZERO_ model testing is performed.) If CHOOSE= is specified in the model selection list that is used for a series, these options have no effect and the chosen model is used.

You can specify the following suboptions:

**MAXNUM=***number* specifies the count of trailing zero values needed in order to consider a nonzero model. If the series has *number* or fewer trailing zero observations, then a nonzero model is considered. For example, if you specify ENDZEROS=(MAXNUM=10), then a nonzero model is considered only if the series has 10 or fewer trailing zero values.

**MAXPCT=***P* specifies the maximum percentage of trailing zero values relative to the number of nonzero values in the entire series. If the number of trailing zero observations in the series is fewer than or equal to *P*% of the nonmissing and nonzero values of the entire series, then a nonzero model is considered. The value of *P* is in the range (0,100). If you specify ENDZEROS=(MAXPCT=20), then a nonzero model is considered only if the number of trailing zero values is fewer than or equal to 20% of the number of nonmissing and nonzero values of the entire series.

**MINOBS=***number* specifies the minimum series length that is required to perform the _ZERO_ model test. For example, if you specify MINOBS=8, then the _ZERO_ model test is not performed on any series whose length is fewer than eight observations. By default, MINOBS=0, which implies no minimum length exclusion.

If MAXNUM=0 and MAXPCT=100 or MAXPCT=0, then a nonzero model is always considered. You can specify a combination of the MAXNUM= and MAXPCT= options. If you specify ENDZEROS=(MAXNUM=10 MAXPCT=20), then a nonzero model is considered only if the series has 10 or fewer trailing zero values and if the number of trailing zero values is fewer than or equal to 20% of the number of nonmissing and nonzero values of the entire series.

When a series fails to use a nonzero model by the criteria specified in this option, the _ZERO_ model is used. In the OUTEST= data set, the _MODEL_ variable is set to _ZERO_, and the remaining data set variables that are associated with the parameter estimates are set to missing.

This option is ignored unless MODE=AUTO is specified.

**GLOBALDIAGNOSE=***spec-name*

**DIAGNOSE=***spec-name*

specifies the name of a diagnostic specification that PROC HPFDIAGSPEC generates and that contains the global diagnostic control options to be used for all forecast variables and time series. This option requires the use of MODE=AUTO. The diagnostic specification *spec-name* must be present in the model repository that is specified in the MODELREPOSITORY= option.

**GLOBALSELECTION=***spec-name*

**SELECTION=***spec-name*

specifies the name of a global model selection list that PROC HPFSELECT defines and that contains the custom models to be used for all forecast variables and time series. This option requires the use of MODE=AUTO. These models are included in the model selection list that is generated for each forecast variable and time series. The model selection list *spec-name* and all of the specifications that it refers to must be present in the model repository that is specified in the MODELREPOSITORY= option. If you want to suppress automatic model identification and use only the models that are specified by the GLOBALSELECTION= option, you can specify the NODIAGNOSE option.

**HOLDOUT=***n*

specifies the size of the holdout sample to be used for model selection. The holdout sample is a subset of the actual time series, and it ends at the last nonmissing observation. The holdout sample is based on the accumulated series. If the holdout sample is not specified, the full range of the accumulated time series is used for model selection.

For each candidate model, the holdout sample is excluded from the initial model fit, multistep forecasts are made within the holdout sample time range, and the statistic of fit that is specified in the CRITERION= option is computed by using observations in the holdout sample. Finally, the candidate model that performs best in the holdout sample, based on this statistic, is selected to forecast the actual time series.

The HOLDOUT= option is used only to select the best performing forecasting model from a list of candidate models. After the best model is selected, the full range of the actual time series (excluding any BACK= observations) is used for subsequent model fitting and forecasting. It is possible for one model to outperform another model in the holdout sample but perform less well when the entire range of the actual series is used.

**HOLDOUTPCT=***number*

specifies the size of the holdout sample as a percentage of the length of the time series. If HOLDOUT=5 and HOLDOUTPCT=10, the size of the holdout sample is $min(5, 0.1T)$, where $T$ is the length of the time series after the beginning and ending missing values are removed. The holdout sample count is computed only when HOLDOUTPCT= is specified; otherwise, HOLDOUT= exclusively determines whether a holdout sample is used.

**INTERMITTENT=***number*

specifies a number greater than one that is used to determine whether or not a time series is intermittent. The last 100 observations are used to determine whether the series is intermittent. If the median demand interval is greater than *number*, then the series is assumed to be intermittent, and the candidate models are Croston's method and the average demand model. If the series is determined not to be intermittent, candidate models are the family of smoothing models. By default, no intermittency test is performed.

**LEAD=***n*

specifies the number of periods ahead to forecast. This number is also called the forecast lead or horizon. The default is LEAD=12.

The LEAD= value is relative to the first observation in the forecast horizon and not to the last nonmissing observation of a particular series. Thus if a series has missing values at the end, the actual number of forecasts computed for that series might be greater than the LEAD= value. The start of the forecast horizon is affected by the BACK= option in the PROC HPFORECAST statement and the HORIZONSTART= option in the TIMEID statement. For more information, see the section "Time Indexing in the Forecasting Process" on page 382.

**MODE=SIMPLE | AUTO**

specifies the forecasting mode for the PROC HPFORECAST invocation. You can specify the following values:

**SIMPLE** restricts its functionality to consider only variants of exponential smoothing models and to optionally test for and use intermittent demand models when the INTERMITTENT= option is specified.

**AUTO** performs automatic time series model identification, model selection, and forecasting. This is the equivalent of a combination of PROC HPFDIAGNOSE and PROC HPFENGINE functionality into PROC HPFORECAST.

By default, MODE=SIMPLE. For more information about comparing MODE=SIMPLE and MODE=AUTO, see the section "Differences between MODE=SIMPLE and MODE=AUTO" on page 384.

**MODELREPOSITORY=***catalog-name*

**MODELREP=***catalog-name*

**REPOSITORY=***catalog-name*

**REP=***catalog-name*

is a two-level SAS catalog name that specifies the location of the model repository. The default for this option is Sashelp.Hpfdflt. This option is supported only when MODE=AUTO.

PROC HPFORECAST must copy the entire contents of the MODELREPOSITORY= catalog into a storage representation that is thread safe (even for single-machine mode). When you perform

distributed mode execution, this representation must be transmitted to all of the appliance nodes that support the distributed processing for the procedure invocation. Make sure that this catalog does not contain extraneous entries that are of no use for the PROC HPFORECAST step for which it is specified. Failure to minimize the size of this catalog to only the necessary entries can needlessly increase start-up time for the procedure invocation.

**NODIAGNOSE**

suppresses the automatic generation of models for MODE=AUTO. This option enables you to perform automatic forecasting by using only the user-defined models that are explicitly specified in the model lists that are applied to the variables specified in FORECAST statements. User-defined models are specified by using the GLOBALSELECTION= option in the PROC HPFORECAST statement, by using the SELECTION= option in the FORECAST statement, or by using the SELECTION property in the SET statement. The NODIAGNOSE option is ignored unless MODE=AUTO. Obviously, using this option is meaningful only if you have specified a selection list that contains user-defined models for all of the FORECAST statement variables over all the hierarchy levels to be forecast. If you have not specified such a selection list, PROC HPFORECAST uses exponential smoothing as a last resort.

By default, automatic models are always generated when MODE=AUTO, and any user-defined models that are specified for the FORECAST statement variables are automatically included in the generated model selection list as additional candidates.

**OVERRIDE=YES | NO**

specifies whether to force the use of any options specified in the PROC HPFORECAST statement instead of those that are explicitly defined by the XML specifications. These XML specifications are specified through the following statement-specific syntax:

**Table 12.2** Statements and XML Specification Syntax

| Statement | Syntax |
|---|---|
| PROC HPFORECAST | GLOBALDIAGNOSE=, GLOBALSELECTION= options |
| FORECAST | DIAGNOSE=, SELECTION= options |
| SET | DIAGNOSE, SELECTION properties |

You can specify the following values:

**YES** forces the use of options that are specified in the PROC HPFORECAST statement instead of the corresponding options from the XML specifications

**NO** uses the options that are contained in the XML specifications

By default, OVERRIDE=NO. This option is ignored unless MODE=AUTO.

**PRINT=*option* | (*options* )**

prints output according to the specified *option* or *options*. By default, the HPFORECAST procedure produces no printed output. The following printing options are available:

**ALLPERF** is the same as specifying PRINT=(PERFORM SUMMARY THREADS TIMING).

**PERFORM** prints the standard “Performance” table that all high-performance analytics procedures produce.

**SUMMARY** prints a summary of the processing that is performed during the PROC HPFORECAST invocation. For information about the ODS tables that are generated by this option, see the section "SUMMARY" on page 396.

**THREADS** prints per-thread statistics for threads that are used the PROC HPFORECAST processing. For more information about the ODS tables that are generated by this option, see the section "THREADS" on page 398.

**TIMING** prints the standard "Timing" table that all high-performance analytics procedures can produce. This table shows a summary of elapsed times to perform the sequential steps that comprise the PROC HPFORECAST invocation. For more information, see the section "TIMING" on page 399.

**PRINTDETAILS**
specifies that output requested by the PRINT= option be printed in greater detail.

**SEASONALITY=***number*
specifies the length of the seasonal cycle. For example, SEASONALITY=3 means that every group of three observations forms a seasonal cycle. The SEASONALITY= option applies only for seasonal forecasting models. By default, the length of a seasonal cycle is the length that is implied by the INTERVAL= option in the TIMEID statement. For example, INTERVAL=MONTH implies that the length of the seasonal cycle is 12.

## DISTBY Statement

**DISTBY** *variable-list* **;**

**NOTE:** This statement is ignored if you do not have a distributed-mode license.

The DISTBY statement specifies a set of one or more variables in the input data set that are used to actively distribute its observations to a set of grid nodes for distributed processing. This statement is used for input data sources that do not accommodate direct alongside execution of the distributed processing. All of the input observations that have the same value for the DISTBY variables are distributed to the same grid node. This data distribution occurs anew each time you invoke PROC HPFORECAST for this form of distributed processing.

When you request distributed mode execution in the PERFORMANCE statement and the input data source is not a distributed DBMS table, using the DISTBY statement is necessary to consistently distribute the input data set observations in order to admit time series processing for the levels of the hierarchy. There is a close relationship between the choice of variables in the DISTBY statement and the hierarchy levels that can be analyzed via the LEVEL statement blocks. For more information about hierarchical time series processing, see the section "Details: HPFORECAST Procedure" on page 363.

When you use the DISTBY statement, the input data set must be sorted by the same variables that you specify in the DISTBY statement. For more information about tradeoffs between distributed and single-machine execution modes, see the section "Determining Single-Machine Mode or Distributed Mode" on page 367.

## END LEVEL Statement

**END LEVEL ;**

This statement terminates a LEVEL block. You must terminate an open LEVEL block before starting another one.

## FORECAST Statement

**FORECAST** *variable-list / options* **;**

The FORECAST statement lists the numeric variables in the DATA= data set whose accumulated values represent time series to be modeled and forecast. The *options* specify various properties that are shared by all of the variables specified in the same FORECAST statement.

A data set variable can be specified in only one FORECAST statement. You can specify any number of FORECAST statements.

You can specify the following *options* in the FORECAST statement:

**ACCUMULATE=***option*
specifies how the data set observations are to be accumulated within each time period for the variables that are listed in the FORECAST statement. If you do not specify this option, accumulation is determined by the ACCUMULATE= option in the TIMEID statement.

**DIAGNOSE=***spec-name*
specifies the name of a diagnostic specification that PROC HPFDIAGSPEC generates and that defines the diagnostic control options to be used for the variables that are specified in this FORECAST statement. This option requires the use of MODE=AUTO. The diagnostic specification *spec-name* must be present in the model repository that you specify in the MODELREPOSITORY= option.

**SELECTION=***spec-name*
specifies the name of a model selection list that PROC HPFSELECT defines and that contains custom models to be used for the variables that are specified in this FORECAST statement. This option requires the use of MODE=AUTO. These models are included in the model selection list that is generated for each associated forecast variable. The *spec-name* and all of the specifications that it refers to must be present in the model repository that you specify in the MODELREPOSITORY= option.

**SETMISSING=***option* | *number*
specifies how missing values (either actual or accumulated) are to be assigned in the accumulated time series for the variables that are listed in the FORECAST statement. If you do not specify this option, missing values are set based on the SETMISSING= option in the TIMEID statement.

**ZEROMISS=***option*
specifies how beginning and ending zero values (either actual or accumulated) are interpreted in the accumulated time series for the variables that are listed in the FORECAST statement. If you do not specify this option, beginning and ending zero values are set based on the ZEROMISS= option in the TIMEID statement.

## INPUT Statement

**INPUT** *variable-list / options* **;**

**INDEPENDENT** *variable-list / options* **;**

**INDEP** *variable-list / options* **;**

**PREDICTOR** *variable-list / options* **;**

The INPUT statement lists the numeric variables in the DATA= data set whose accumulated values represent time series to be used as explanatory variables for the time series models. The *options* specify various properties that are shared by all the variables that are specified in the same INPUT statement. The INPUT statement is allowed only when MODE=AUTO.

A data set variable can be specified in only one INPUT statement, and it cannot be a variable that is already selected for use by another statement. You can specify any number of INPUT statements.

You can specify the following *options* in the INPUT statement:

**ACCUMULATE=***option*

specifies how the data set observations are to be accumulated within each time period for the variables that are listed in the INPUT statement. If you do not specify this option, accumulation is determined by the ACCUMULATE= option in the TIMEID statement.

**EXTEND=***extend*

specifies how future values are to be computed for the variables that are specified in the INPUT statement. You can specify the following *extend* values:

**AVERAGE** requests that future values be generated by using the average of the series over the historical period for the respective INPUT variables.

**FIRST** requests that future values be generated by using the first nonmissing value of the series over the historical period for the respective INPUT variables.

**LAST** requests that future values be generated by using the last nonmissing value of the series over the historical period for the respective INPUT variables.

**MAXIMUM** requests that future values be generated by using the maximum value of the series over the historical period for the respective INPUT variables.

**MEDIAN** requests that future values be generated by using the median value of the series over the historical period for the respective INPUT variables.

**MINIMUM** requests that future values be generated by using the minimum value of the series over the historical period for the respective INPUT variables.

**NONE** requests that no future values be generated automatically for the respective INPUT variables.

**STOCHASTIC** requests that future values be generated by using the forecasts that are produced from the best exponential smoothing model that is selected by using a criterion of in-sample RMSE over the historical period of the respective INPUT variables. This value is equivalent to the model that is produced when METHOD=BEST in the HPFESMSPEC procedure.

By default, EXTEND=STOCHASTIC.

**KEEP=YES | NO**

enables or disables a check of inputs to models. This option enables you to independently specify control for the associated input variables during the model selection phase of forecasting, whereas the REQUIRED= option enables you to specify hints about the input variables during the model generation phase. The following types of problems are checked:

- errors in functional transformation
- an input that consists of a constant value
- an input that has small variation approaching machine precision
- an input with all missing values
- errors introduced by differencing

You can specify the following values:

**YES** performs no checks and drops no inputs from a model. The model might subsequently fail to fit during parameter estimation or forecasting for any of the reasons in the preceding list.

**NO** checks inputs and drops from the model those that have errors (as defined by the preceding list) for the current series. No changes that result from dropped inputs are kept in the model specification.

This option has no effect on models that have no inputs.

By default, KEEP=YES. For automatically generated models that are produced from MODE=AUTO, the behavior is always to force KEEP=YES regardless of whether this option is included in your INPUT statements.

**REPLACEMISSING=YES | NO**

specifies whether missing values in the historical region of the time series be replaced with a value according to the EXTEND= option for each of the variables in the INPUT statement.

You can specify these values:

**YES** replaces missing values.

**NO** does not replace missing values.

By default, REPLACEMISSING=NO.

**REQUIRED=YES | NO | MAYBE < (POSITIVE | NEGATIVE) >**

provides a hint for the model identification step as to importance of the time series variables in the model families that support explanatory variables and in the user-defined models that require the use of explanatory variables. The specified value applies to each of the variables in the INPUT statement.

You can specify the following values:

**YES** includes the input variables in the model as long as the model does not fail to be diagnosed.

MAYBE includes the input variables in the model as long as their parameters are significant.

NO includes the input variables in the model as long as their parameters are significant and the increment of the value of criterion exceeds a threshold.

You can also specify either the POSITIVE or NEGATIVE in parentheses after the YES, MAYBE, or NO value. For example, specifying REQUIRED=YES(POSITIVE) drops the input variable from the model if its coefficient is negative, and specifying REQUIRED=YES(NEGATIVE) implies the opposite. The specification of POSITIVE or NEGATIVE does not mean that constraints are imposed during the estimation of the variable's coefficient in the model. This option functions the same as the REQUIRED= option in PROC HPFDIAGNOSE. By default, REQUIRED=NO.

**SELECTION=***spec-name*
specifies the name of a model selection list to be used to forecast future values for the time series variables that are specified in the INPUT statement. The difference between this option and EXTEND=STOCHASTIC is that the forecasts for the input variables are produced by evaluating best performing forecast from the model selection list versus the exponential smoothing models that are used when you specify EXTEND=STOCHASTIC.

**SETMISSING=***option* | *number*
specifies how missing values (either actual or accumulated) are to be assigned in the accumulated time series for the variables that are listed in the INPUT statement. If you do not specify this option, missing values are set based on the SETMISSING= option in the TIMEID statement.

**ZEROMISS=***option*
specifies how beginning and ending zero values (either actual or accumulated) are interpreted in the accumulated time series for the variables that are listed in the INPUT statement. If you do not specify this option, beginning and ending zero values are set based on the ZEROMISS= option in the TIMEID statement.

## LEVEL Statement

**LEVEL _TOP_** */ options* **;**

**LEVEL** *variables / options* **;**

You can specify LEVEL blocks in PROC HPFORECAST to obtain separate analyses for groups of observations that are defined by the LEVEL variables. Each LEVEL statement must be accompanied by an END LEVEL statement. Remember that the LEVEL blocks must follow the PROC HPFORECAST statement precedence (see "PROC HPFORECAST Statement" on page 344).

Any variable type (numeric or character) can be used in the set of variables that is specified in the LEVEL statement. The LEVEL blocks can be specified in any order; however, the hierarchy levels that are spanned by the collection of LEVEL blocks must be nested. For more information about admissible level nesting, see the section "Parallel Time Series Processing" on page 363.

LEVEL _TOP_ requests that the topmost level of the hierarchy be included in the analysis. The _TOP_ level always produces exactly one time series group that has no level variable qualification.

You can specify *options* in the LEVEL statement that specifically apply to time series variables that are acquired in the context of the associated hierarchy level. Time series for FORECAST variables inherit these options from the FORECAST statement context, whereas time series for other variables inherit these options

from the TIMEID statement context. Any *option* that is specified in the LEVEL statement overrides the inherited option setting for all of the time series variables that are acquired in the context of the LEVEL statement.

The LEVEL statement supports the following *options*:

**NOFORECAST**
suppresses forecasting for the time series that are acquired from the input data set for this level. This option establishes a starting level member set that can be used as the source for distributed level processing.

You can specify the NOFORECAST option for single-machine mode, but there is no justification for using it. Likewise, you can specify it together with a DISTBY statement for distributed mode execution when the input data set does not support alongside execution, but there is no justification for this usage either. For either of these use cases, you can directly specify the hierarchy levels to be forecast without wasting CPU and memory to acquire a starting level member set. For more information about the connection between time series distribution to the DBMS store and hierarchy level processing, see the section "Data Preparation and Staging" on page 374.

## OUTPUT Statement

**OUTPUT** *options / qualifiers* **;**

You can specify the following *options* in the OUTPUT statement:

**OUTEST=***SAS-data-set*
names the output data set to contain the model parameter estimates and the associated test statistics and probability values. The OUTEST= data set is particularly useful for evaluating the significance of the model parameters and understanding the model dynamics.

**OUTFOR=***SAS-data-set*
names the output data set to contain the forecast time series components (actual, predicted, lower confidence limit, upper confidence limit, prediction error, and prediction standard error).

**OUTLOG=***SAS-data-set*
names the output data set to contain the per-series error synopsis that is generated from model-specific or data-specific failures. The OUTLOG= data set largely replaces the use of the SAS log to display such errors. Only errors are reported to the OUTLOG= data set; therefore, a clean PROC HPFORECAST run produces an empty OUTLOG= data set.

**OUTSTAT=***SAS-data-set*
names the output data set to contain the statistics of fit (goodness-of-fit statistics). The OUTSTAT= data set is particularly useful in evaluating how well the model fits the series. The statistics of fit are based on the entire range of the time series regardless of whether the HOLDOUT= option is specified. One observation is always generated for the selected model's fit region. When you specify a nonzero BACK= option, a second observation is generated that contains fit statistics computed over the observations that correspond to the BACK= option.

**OUTSUM=***SAS-data-set*
names the output data set to contain the summary statistics and the forecast summation. The summary statistics are based on the accumulated time series. The OUTSUM= data set is particularly useful when you forecast large numbers of series and you need a summary of the results.

When MODE=AUTO in the PROC HPFORECAST statement, you can also specify the following *options* in the OUTPUT statement:

**OUTINDEP=***SAS-data-set*
names the output data set to contain time series for explanatory variables that are used in the final forecast. When these series contain missing values, this data set provides complete information for the time series that are actually used in the generation of the final forecast. Any missing values for the future values or for the historical period are replaced as specified in the associated INPUT statement options.

**OUTSTATSELECT=***SAS-data-set*
names the output data set to contain the fit statistics for the models that are evaluated during the model selection process. This data set contains the same fit statistics variables as found in the OUTSTAT= data set. They are evaluated over the in-sample region of the time series for each candidate model when no holdout region is used; otherwise, they are evaluated over the holdout region of the time series for each candidate model. The former condition is differentiated by a value of FIT in the _REGION_ column, and the latter condition is differentiated by a value of FORECAST in the _REGION_ column. The system-generated label for each candidate model is included in the _LABEL_ column.

**OUTMODELINFO=***SAS-data-set*
names the output data set to contain detailed information about the selected forecast model. The data set has information such as the model family, presence or absence of inputs, events and outliers, and so on.

You can specify the following *qualifiers* in the OUTPUT statement:

**NOOUTALL**
requests that only multistep forecasts be written to the OUTFOR= data set. By default, the OUTFOR data set contains the one-step-ahead forecasts in the historical (fit) region of the time series and the multistep forecasts in the forecast horizon. When you specify this option, only observations in the forecast horizon are saved. This option can substantially reduce the size of the OUTFOR= data set and simplify its use during postprocessing when you are interested only in the forecasts.

For more information about the different output data sets that PROC HPFORECAST produces, see the section "Output Data Set Details" on page 389.

## PERFORMANCE Statement

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

The PERFORMANCE statement defines performance parameters for multithreaded and distributed computing, passes variables that describe the distributed computing environment, and requests detailed results about the performance characteristics of PROC HPFORECAST.

You can also use the PERFORMANCE statement to control whether a PROC HPFORECAST executes in single-machine or distributed mode.

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

**COMMIT=***n*
requests that the procedure write periodic updates to the SAS log when observations are sent from the client to the appliance for distributed processing.

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

PROC HPFORECAST can send distribute data in blocks to the appliance. Whenever the number of observations sent exceeds an integer multiple of the COMMIT= size, a SAS log message is produced. The message indicates the actual number of observations distributed, and not an integer multiple of the COMMIT= size.

**DATASERVER=***"name"*
specifies the name of the server on Teradata systems as defined through the *hosts* file and as used in the LIBNAME statement for Teradata. For example, assume that the *hosts* file defines the server for Teradata as follows:

```
myservercop1  33.44.55.66
```

Then a LIBNAME specification would be as follows:

```
libname TDLib teradata server=myserver user= password= database= ;
```

A PERFORMANCE statement to induce running alongside the Teradata server would specify the following:

```
performance dataserver="myserver";
```

The DATASERVER= option is not required if you specify the GRIDMODE=option in the PERFORMANCE statement or if you set the GRIDMODE environment variable.

Specifying the DATASERVER= option overrides the GRIDDATASERVER environment variable.

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

**GRIDHOST=***"name"*

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

**GRIDMODE=SYM | ASYM**

**MODE=SYM | ASYM**

specifies whether the HPFORECAST procedure runs in symmetric (SYM) mode or asymmetric (ASYM) mode. The default is GRIDMODE=SYM. For more information about these modes, see the section "Symmetric and Asymmetric Distributed Modes" on page 365.

If this option is specified, it overrides the GRIDMODE environment variable.

**GRIDTIMEOUT=***s*

**TIMEOUT=***s*

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

**INSTALL=***"name"*

**INSTALLLOC=***"name"*

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

**NODES=ALL** | *n*

**NNODES=ALL** | *n*

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

Specifying NODES=0 indicates that you want to process the data in single-machine mode on the client machine. If the input data are not alongside the database, this is the default. The HPFORECAST procedure then performs the analysis on the client.

If the data are not read alongside the database, the NODES= option specifies the number of nodes on the appliance that are involved in the analysis. If the value of the NODES= option is greater than one, you must also include a DISTBY statement that specifies one or more variables to be used to distribute the rows from the data set to the appliance nodes.

If the data are read alongside the distributed database on the appliance, specifying a nonzero value for the NODES= option has no effect. The number of units of work in the distributed computing environment is then determined by the distribution of the data and cannot be altered.

**NTHREADS=***n*

**THREADS=***n*

specifies the number of threads for analytic computations and overrides the SAS system option THREADS | NOTHREADS. If you do not specify the NTHREADS= option, the number of threads is determined based on the number of CPUs on the host on which the analytic computations execute. The HPFORECAST procedure uses this as its thread budget on each machine where it executes.

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

**NOTE:** The SAS system options THREADS | NOTHREADS apply to the client machine on which the HPFORECAST procedure executes. They do not apply to the compute nodes in a distributed environment.

## SET Statement

**SET** *variable.property=type(value)* **;**

The SET statement updates a named attribute (which is identified by *property*) of the target *variable* with the result from the right-hand side of the assignment that is identified by *type* and *value*. The allowed set of *property* names depends on the classification of the *variable* (for example, FORECAST or INPUT). The data type of a right-hand side expression is inherently identified by the cast operator *type* whose value is identified by the *value* that is specified in the cast operator. The SET statement is allowed only when MODE=AUTO. You can include SET statements only within a LEVEL/END LEVEL block.

For example, consider the following PROC HPFORECAST statements:

```
forecast sales;
level sku;
  set sales.diagnose=spec(diagsales);
end level;
level _top_;
end level;
```

This sets the DIAGNOSE property of the FORECAST variable Sales to be the specification named DIAGSALES. The specification DIAGSALES must be a diagnostic control specification that PROC HPFDIAGSPEC generates and it must be contained in the SAS catalog that is specified in the MODELREPOSITORY= option of the PROC HPFORECAST statement. This tells PROC HPFORECAST to use the diagnostic control preferences in DIAGSALES to perform model identification for the FORECAST variable Sales. The scope of this diagnostic control is limited to the Sku level of the hierarchy. Diagnostic control preferences for the _TOP_ level of the hierarchy revert to the global diagnostic control settings for the PROC HPFORECAST invocation.

For FORECAST variables, you can specify the following *property* values:

**DIAGNOSE** identifies the diagnostic control specification to be used for the target *variable*. The *type* must be SPEC, and the *value* must be a PROC HPFDIAGSPEC specification. The default diagnostic preferences are always defined by the settings for the FORECAST variable.

**SELECTION** identifies the user-defined custom model selection list to be included for the target *variable*. The *type* must be SPEC. The *value* must be a PROC HPFSELECT specification.

**INPUTS** identifies a subset of the collective INPUT variables to be considered during model identification for the target *variable*. User-defined models are not constrained to use only this set of explanatory variables. They can refer to any variable that is specified in an INPUT statement. The *type* must be VARS. The *value* must be a variable list that contains one or more INPUT variables. By default, all declared INPUT variables are considered for each FORECAST variable.

For INPUT variables, you can specify the following *property* values:

**SELECTION** identifies the user-defined model selection list to be used to forecast future values for the target *variable*. The *type* must be SPEC, and the *value* must be a PROC HPFSELECT specification. The default behavior is always defined by the EXTEND= option setting from the INPUT statement.

**REPMISS** identifies the replace missing property for the target *variable*. The *type* must be BOOL, and the *value* must be YES or NO. The default is defined by the REPLACEMISSING option in the INPUT statement that specifies the input variables.

## TIMEID Statement

**TIMEID** *variable* **INTERVAL**=*interval* < *options* > **;**

The TIMEID statement names a numeric *variable* that identifies observations in the input data set. The values of the TIMEID *variable* are assumed to be SAS date, time, or datetime values. You must specify an *interval* argument in the INTERVAL= option to indicate the desired frequency of the accumulated time series. You can specify *options* to describe how to accumulate and how to align the timestamp values. These *options* apply to all variables that are specified in subsequent FORECAST statements unless the FORECAST statement includes a more specific option that takes precedence.

You can specify the following *options*:

**ACCUMULATE=***value*

specifies how to accumulate the data set observations within each time period. The frequency (width of each time interval) is specified by the INTERVAL= argument. The values of the TIMEID *variable* are the timestamp values, and each timestamp value corresponds to a specific time period with respect to the frequency used. The accumulated values form the actual time series, which is used in subsequent model fitting and forecasting.

The ACCUMULATE= option is particularly useful when there are multiple input observations with timestamp values that coincide with a particular time period (for example, transactional data).

You can specify the following *values*, which determine how to accumulate the observations within each time period, based on the TIMEID *variable* and the frequency specified by the INTERVAL= argument:

**AVERAGE | AVG** Accumulation of observations is based on the average of their values.

**CSS** Accumulation of observations is based on the corrected sum of squares of their values.

**MAXIMUM** Accumulation of observations is based on the maximum of their values.

**MINIMUM** Accumulation of observations is based on the minimum of their values.

**N** Accumulation of observations is based on the number of nonmissing observations.

**NMISS** Accumulation of observations is based on the number of missing observations.

**NOBS** Accumulation of observations is based on the number of observations.

**STDDEV | STD** Accumulation of observations is based on the standard deviation of their values.

| | |
|---|---|
| **TOTAL** | Accumulation of observations is based on the sum of their values. |
| **USS** | Accumulation of observations is based on the uncorrected sum of squares of their values. |

The SETMISSING= option is useful for specifying how accumulated missing values are treated. If missing values should be interpreted as zero, then SETMISSING=0 should be used. For more information, see the section "Accumulation" on page 381.

**ALIGN=***value*

controls the alignment of SAS dates that are used to identify output observations. You can specify the following *values*: BEGINNING | BEG, MIDDLE | MID, and ENDING | END. By default, ALIGN=BEGINNING.

**END=***value*

specifies a SAS date, datetime, or time value that represents the end of the data. If the largest timestamp value for the TIMEID *variable* is less than *value*, the series is extended with missing values. Observations from the input data set with values for the TIMEID *variable* greater than *value* are skipped. For example, END="&sysdate"D uses the automatic macro variable SYSDATE to extend or truncate the series to the current date. You can use START= and END= options to ensure that the time series that are associated with each BY group contain the same number of observations.

**FORMAT=***format*

specifies the SAS format for the time ID values. If you do not specify this option, the default format is implied from the *interval* that is specified in the INTERVAL= option.

**INTERVAL=***interval*

specifies the frequency of the input time series. For example, if the input data set consists of quarterly observations, then specify INTERVAL=QTR. The INTERVAL= option also determines the time periods for the accumulation of observations. For example, if the timestamp values in the input data set are recorded at a weekly frequency as SAS date values and you specify INTERVAL=QTR and ACCUMULATE=TOTAL, then the accumulated time series are created at a quarterly frequency and the quarterly observations are computed as the totals of the corresponding weekly observations from the input data set. If you do not specify the SEASONALITY= option, the length of the seasonal cycle is implied from the *interval* that is specified in the INTERVAL= option. For example, INTERVAL=QTR implies a seasonal cycle of length 4.

The basic *interval* values are YEAR, SEMIYEAR, QTR, MONTH, SEMIMONTH, TENDAY, WEEK, WEEKDAY, DAY, HOUR, MINUTE, SECOND. For more information about intervals that you can specify, see Chapter 4, "Date Intervals, Formats, and Functions" (*SAS/ETS User's Guide*),

**HORIZONSTART=***date*

specifies a SAS date, datetime, or time value that represents the start of the forecast horizon. If *date* falls beyond the end of the historical data, then the number of periods of multistep forecasting will be greater than the LEAD= value specified in the PROC HPFORECAST statement. The effective forecast horizon for any particular level member (BY group) might differ from another due to differences in the lengths of the historical data across the hierarchy levels, but all forecasts will end at the same date as determined by the HORIZONSTART=, BACK=, and LEAD= options. If you do not specify the HORIZONSTART= option, the forecast horizon of the series for each variable that is specified in a FORECAST statement is defined relative to the first time point beyond the end of the series. For more

information about how the HORIZONSTART=, BACK=, and LEAD= options interact, see the section "Time Indexing in the Forecasting Process" on page 382

**SETMISSING=***option* | *number*

specifies how to assign missing values (either actual or accumulated) in the accumulated time series. If you specify a *number*, missing values are set to *number*. If a missing value indicates an unknown value, you should not use this option. If a missing value indicates no value, you should specify SETMISSING=0. You usually specify SETMISSING=0 for transactional data because no recorded data usually implies no activity.

You can specify the following *options* to determine how missing values are assigned:

| | |
|---|---|
| **AVERAGE \| AVG** | Missing values are set to the accumulated average value. |
| **FIRST** | Missing values are set to the accumulated first nonmissing value. |
| **LAST** | Missing values are set to the accumulated last nonmissing value. |
| **MAXIMUM** | Missing values are set to the accumulated maximum value. |
| **MINIMUM** | Missing values are set to the accumulated minimum value. |
| **MISSING** | Missing values are set to missing. This is the default. |
| **NEXT** | Missing values are set to the next accumulated nonmissing value. Missing values at the end of the accumulated series remain missing. |
| **PREVIOUS \| PREV** | Missing values are set to the previous accumulated nonmissing value. Missing values at the beginning of the accumulated series remain missing. |

**START=***value*

specifies a SAS date, datetime, or time value that represents the beginning of the data. If the smallest timestamp value for the TIMEID *variable* is greater than *value*, missing values are added at the beginning of the series. Observations from the input data set whose values for the TIMEID *variable* are less than *value* are skipped. You can use START= and END= options to ensure that data that are associated with each BY group contain the same number of observations.

**ZEROMISS=***option*

specifies how to interpret beginning and ending zero values (either actual or accumulated) in the accumulated time series. You can specify these *option* values:

| | |
|---|---|
| **BOTH** | Both beginning and ending zeros are set to missing. |
| **LEFT** | Beginning zeros are set to missing. |
| **NONE** | Beginning and ending zeros are not changed. This is the default. |
| **RIGHT** | Ending zeros are set to missing. |

If the accumulated series is all zero valued, the series is not changed.

# Details: HPFORECAST Procedure

## Parallel Time Series Processing

Many SAS high-performance analytical procedures use partitioned data to affect a divide-and-conquer strategy to solve a single very large-scale problem. In many cases, the problem involves parameter estimation for a model that uses so many observations that it would be impossible to solve the problem on a single machine in a reasonable amount of time. For such a large amount of input, the amount of output from solving such a problem might be comparatively small. For example, you can fit a model that has 1,000 parameters on a data set that contains 10 thousand or 1 billion observations, but the amount of output is the same if you are interested only in the parameter estimates and their statistical properties. And it usually does not matter how the observations are distributed in the divide-and-conquer scheme. An equitable division is generally the only important factor.

A large-scale time series forecasting problem is completely different in nature. The time series are generally short. For example, two years of weekly retail data amounts to only 104 observations per series, but for a large retail operation there might be several million SKU-level series that have to be analyzed and forecast. The computational time needed to forecast any one series by using an exponential smoothing or ARIMA model is very short, perhaps only 100 microseconds on current technology. No one time series problem is too large to solve on a single CPU, and any increase in speed must come from using a scheme that enables the entire forecasting process over all those series to be performed in parallel. Further, the distribution scheme must ensure that observations that comprise a given time series are all co-located in the same partition of the data. Lastly, any scheme that enables more work to be performed in a single pass of the data source can provide benefits in terms of time reduction.

PROC HPFORECAST uses a divide-and-conquer strategy to affect parallel processing of the time series that arise from different BY groups over the span of the hierarchy levels to be considered during the procedure step. This constitutes what is frequently referred to as an embarrassingly parallel problem because the time series that are realized from each BY group are independent of each other. For many purposes (such as time series analysis, modeling, and forecasting), the BY groups can be processed in parallel by using multiple threads on a single multiprocessor system or by using different computers in a grid-based computing environment, or a combination of both.

At its core, large-scale hierarchically-structured time series forecasting tends to be an I/O intensive problem. SAS data sets, or more generally relational data sources, provide a rectangular storage model that presents a sequence of rows, where each row has level member qualification variables, a time stamp variable, and one or more analysis variables. Turning that row sequence into a time series requires some work. For a given hierarchy level, a collection of time series (a time series group) arises from this row sequence, where each collection member is qualified by a set of distinct values of the level qualification variables. These distinct groups are called members or level members (BY groups might be a more familiar SAS term).

For each distinct level member, the set of rows that comprise the group is processed to produce a time series data frame. This data frame is characterized by the following properties:

- a uniform time ID sequence that is computed from the time stamp values
- a set of one or more time series variables that arise from the analysis variables

The sequence of rows of timestamped data is effectively transposed into a set of arrays: one array per analysis variable and an array of the uniform time ID values. All are indexed by the same discrete set of time indices.

Data set output is always qualified by the level member's distinct values. Different output data sets have different schema (depending on the information stored), but they all contain the set of hierarchy variables for the leaf level that is specified in the statement block of the PROC HPFORECAST step. The leaf level is defined by the LEVEL statement that has the most grouping variables in its *variable* list. Output rows that are generated for leaf-level members are fully qualified by using the member's distinct values for its hierarchy variables. For roll-ups to higher levels, the qualification variables that are aggregated out are filled with missing values. This means that character variables are filled with blanks and numeric variables are set to the default missing value (“.”). For the time series forecasting problem, the amount of output can be larger than the amount of input (10 times or more is common), depending on the output data sets and the time spans that are requested. When the amount of input is large, the amount of output can easily dominate the execution time.

Using threads for parallel analysis and forecasting of the level member's time series data scales close to perfection for that part of the processing when viewed apart from the whole. However, the timing of the whole process flow from acquisition of the time series data frames, then analysis and forecasting, and then output of results has a common factor that appears in each stage of the processing flow: the number of BY groups. Given an inherently serial stage on the front end to acquire each data frame for each level member, and an inherently serial stage on the back end to persist results for each level member, the overall increase in speed from the parallel processing of the level members in the middle is constrained by those serial steps. For all practical purposes, the scalability analysis looks like the case of synchronous time series BY-group processing, which is dominated by the synchronous input step on the front end and the synchronous output step on the back end. The overall time for either analysis scales linearly according to the number of BY groups in the data set.

The real scalability gains in the parallel time series forecasting problem come most directly from dividing the time series data set (more specifically the BY groups that represent its time series) into several partitions that can be processed in parallel on different machines. Because these machines do not compete for shared resources, this partitioned data scheme scales close to perfection. Threading portions of the time series acquisition processing and the forecasting stage on each of those partitions adds a marginal gain, but it does not come close to the increase in speed that comes from partitioning the data. The problem to overcome is the per-series serial processing on the front and back ends of the overall process flow, and that is I/O related. So the N-fold reduction of I/O time from dividing the data provides the real gain for parallel time series processing. Further, that N-fold division from the N DBMS machines also applies to the output processing.

## High-Performance Analytics: Concepts and Topics for PROC HPFORECAST

This section describes the modes of execution for the HPFORECAST procedure. To run PROC HPFORECAST in distributed mode, you must have an additional license.

### Single-Machine Mode

Single-machine mode is a computing model in which multiple processors or multiple cores are controlled by a single operating system and can access shared resources, such as disks and memory. More simply, single-machine mode for the HPFORECAST procedure means multithreading on the client machine.

## Distributed Mode

Distributed mode is a computing model in which several nodes in a distributed computing environment participate in the calculations. In distributed mode, the HPFORECAST procedure performs the analytics on an appliance that consists of a cluster of nodes. This appliance can be one of the following:

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

Distributed mode has these variations:

- Client-data (or local-data) mode: The input data for the analytic task are not stored on the appliance or cluster but are distributed to the distributed computing environment by the SAS High-Performance Analytics infrastructure when the HPFORECAST procedure runs.
- Alongside-the-database mode: The data are stored in the distributed database and are read from the DBMS in parallel into the HPFORECAST procedure that runs on the database appliance.
- Alongside-HDFS mode: The data are stored in the Hadoop Distributed File System (HDFS) and are read in parallel from the HDFS. This mode is available if you install the SAS High-Performance Deployment of Hadoop on the appliance or when you configure a Cloudera 4 Hadoop deployment on the appliance to operate with the SAS High-Performance Analytics infrastructure. For more information about installing the SAS High-Performance Deployment of Hadoop, see the *SAS High-Performance Analytics Infrastructure: Installation and Configuration Guide*.
- Alongside-LASR mode: The data are loaded from a SAS LASR Analytic Server that runs on the appliance.

## Symmetric and Asymmetric Distributed Modes

The HPFORECAST procedure can run in asymmetric mode alongside the database. The primary reason for providing the asymmetric mode is to enable you to manage and house data on one appliance (the data appliance) and to run the HPFORECAST procedure on a second appliance (the computing appliance). You can also run in asymmetric mode on a single appliance that functions as both the data appliance and the computing appliance. This enables you to run alongside the database, where computations are done on a different set of nodes from the nodes that contain the data. The following subsections provide more details.

### *Symmetric Mode*

When the HPFORECAST procedure runs in symmetric distributed mode, the data appliance and the computing appliance must be the same appliance. Both the SAS Embedded Process and the HPFORECAST procedure executes in a process that runs on the same hardware where the DBMS process executes. This is called symmetric mode because the number of nodes on which the DBMS executes is the same as the number of nodes on which the HPFORECAST procedure executes. The initial data movement from the DBMS to the HPFORECAST procedure does not cross node boundaries.

#### Asymmetric Mode

When the HPFORECAST procedure runs in asymmetric distributed mode, the data appliance and computing appliance are usually distinct appliances. The HPFORECAST procedure executes in a process that runs on the computing appliance. The DBMS and a SAS Embedded Process run on the data appliance. Data are requested by a SAS data feeder that runs on the computing appliance and communicates with the SAS Embedded Process on the data appliance. The SAS Embedded Process transfers the data in parallel to the SAS data feeder that runs on each of the nodes of the computing appliance. This is called asymmetric mode because the number of nodes on the data appliance does not need to be the same as the number of nodes on the computing appliance.

### Controlling the Execution Mode with Environment Variables and Performance Statement Options

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

The important environment variables follow:

- *grid host* identifies the domain name system (DNS) or IP address of the appliance node to which the software connects to run in distributed mode.
- *installation location* identifies the directory where the distributed-mode software is installed on the appliance.
- *data server* identifies the database server on Teradata appliances as defined in the *hosts* file on the client. This data server is the same entry that you usually specify in the SERVER= entry of a LIBNAME statement for Teradata. For more information about specifying LIBNAME statements for Teradata and other relational databases, see the *SAS/ACCESS Interface* documentation for the specific database.
- *grid mode* specifies whether the HPFORECAST procedure executes in symmetric or asymmetric mode. Valid values for this variable are `'sym'` for symmetric mode and `'asym'` for asymmetric mode. The default is symmetric mode.

You can set an environment variable directly from the SAS program by using the OPTION SET= command. For example, the following statements define three variables for a Teradata appliance (the grid mode is the default symmetric mode):

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

Alternatively, you can set the parameters in the PERFORMANCE statement. For example:

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

The following statements define three variables that are needed to run asymmetrically on a computing appliance.

```
option set=GRIDHOST      ="compute_appliance.sas.com";
option set=GRIDINSTALLLOC="/opt/TKGrid";
option set=GRIDMODE      ="asym";
```

Alternatively, you can set the parameters in the PERFORMANCE statement. For example:

```
performance host      ="compute_appliance.sas.com"
            install   ="/opt/TKGrid"
            gridmode  ="asym"
```

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

Specifying a data server is necessary only on Teradata systems when you do not explicitly set the *gridmode* environment variable or specify the GRIDMODE= option in the PERFORMANCE statement. The data server specification depends on the entries in the (client) *hosts* file. The file specifies the server (suffixed by *cop* and a number) and an IP address. For example:

```
 myservercop1  33.44.55.66
```

The key variable that determines whether the HPFORECAST procedure executes in single-machine or distributed mode is the *grid host*. The installation location and data server are needed to ensure that a connection to the grid host can be made, given that a host is specified. This book assumes that the installation location and data server (if necessary) have been set by your system administrator.

The following sets of SAS statements are functionally equivalent:

```
proc hpforecast ...;
   performance host="hpa.sas.com";
   [...];
run;

option set=GRIDHOST="hpa.sas.com";
proc hpforecast ...;
   [...];
run;
```

## Determining Single-Machine Mode or Distributed Mode

The HPFORECAST procedure uses the following rules to determine whether it runs in single-machine mode or distributed mode:

- If a grid host is not specified, the analysis is carried out in single-machine mode on the client machine that runs the SAS session.
- If a grid host is specified, the behavior depends on whether the execution is alongside the database or alongside HDFS. If the data are local to the client (that is, not stored in the distributed database or HDFS on the appliance), you need to use the NODES= option in the PERFORMANCE statement

to specify the number of nodes on the appliance or cluster that you want to engage in the analysis. If the procedure executes alongside the database or alongside HDFS, you do not need to specify the NODES= option.

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

When output of a similar or greater magnitude is produced by the grid-side processing of the input data set and that output is also returned to the client to persist as output data sets, the overall data transfer time increases accordingly. Obviously, there must be a significant amount of computing time spent on the grid-side nodes to overcome the transfer overhead when client-side data transfer to and from the grid is included.

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

## Alongside-the-Database Execution

The HPFORECAST procedure interfaces with the distributed database management system (DBMS) on the appliance in a unique way. If the input data are stored in the DBMS and the grid host is the appliance that houses the data, the HPFORECAST procedure creates a distributed computing environment in which an analytic process is co-located with the nodes of the DBMS. Data then pass from the DBMS to the analytic process on each node. Instead of moving across the network and possibly back to the client machine, the data pass locally between the processes on each node of the appliance.

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

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

Before you can run PROC HPFORECAST alongside the database, you must distribute the data to the appliance. The following statements use the DATA step procedure to distribute the data set Sashelp.Pricedata into the database mydb on the hpa.sas.com appliance. In this example, the appliance houses a Greenplum database.

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


proc datasets lib=applianc nolist;
   delete pricedata;
run;
data applianc.pricedata(
        dbtype= (productName='char(9)')
```

```
            distributed_by= 'distributed by (productName)'
            bulkload= yes
            bl_format= 'TEXT'
            bl_null='\n'
            bl_host='hpa.sas.com'
            bl_port='8081'
            bl_protocol='gpfdist'
            bl_delete_datafile=YES
            bl_exception='rejected');

      set sashelp.pricedata;
   run;
```

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

Note that the libref for the output table points to the appliance. The data set options for Applianc.Pricedata are specific to the Greenplum LIBNAME engine. These options invoke the Greenplum bulk loader to distribute the records by the distinct values of the ProductName variable among the data segments of the appliance. Every database has its own scheme to accommodate bulk loading. See the *SAS/ACCESS Interface* documentation for the specific database.

When PROC HPFORECAST executes symmetrically alongside the database, any nonzero specification of the NODES= option in the PERFORMANCE statement is ignored. If the data are read alongside the database, the number of compute nodes is determined by the layout of the database and cannot be modified.

However, when PROC HPFORECAST executes asymmetrically alongside the database, the number of compute nodes that you specify in the PERFORMANCE statement can differ from the number of nodes across which the data are partitioned.

### Alongside-LASR Distributed Execution

You can execute high-performance analytical procedures in distributed mode alongside a SAS LASR Analytic Server. When high-performance analytical procedures execute in this mode, the data are preloaded in distributed form in memory that is managed by a LASR Analytic Server. The data on the nodes of the appliance are accessed in parallel in the process that runs the LASR Analytic Server, and they are transferred to the process where the high-performance analytical procedure runs. In general, each high-performance analytical procedure copies the data to memory that persists only while that procedure executes. Hence, when a high-performance analytical procedure runs alongside a LASR Analytic Server, both the high-performance analytical procedure and the LASR Analytic Server have a copy of the subset of the data that is used by the high-performance analytical procedure. The advantage of running high-performance analytical procedures alongside a LASR Analytic Server (as opposed to running alongside a DBMS table or alongside HDFS) is that the initial transfer of data from the LASR Analytic Server to the high-performance analytical procedure is a memory-to-memory operation that is faster than the disk-to-memory operation when the procedure runs alongside a DBMS or HDFS. When the cost of preloading a table into a LASR Analytic Server is amortized by multiple uses of these data in separate runs of high-performance analytical procedures, using the LASR Analytic Server can result in improved performance.

### Running PROC HPFORECAST Alongside a SAS LASR Analytic Server in Distributed Mode

This section provides an example of steps that you can use to start a SAS LASR Analytic Server instance, load data into it, and then run the HPFORECAST procedure alongside this LASR Analytic Server instance.

### Starting a SAS LASR Analytic Server Instance

The following statements create a SAS LASR Analytic Server instance and load it with the Sashelp.Pricedata data set so that it is partitioned by the distinct values of the variable ProductName. The data that are loaded into the LASR Analytic Server persist in memory across procedure boundaries until these data are explicitly deleted or until the server instance is terminated.

```
   option set=GRIDHOST="greenarrow.unx.sas.com";
   option set=GRIDINSTALLLOC="/opt/v940m1/laxnd/TKGrid";
   option set=GRIDDATASERVER="greenarrow";

proc lasr create path="/tmp/" port=9152;
   performance host="greenarrow.unx.sas.com" nodes=ALL;
run;

libname MyLasr sasiola port=9152 host="greenarrow.unx.sas.com" tag=hpfug;

data MyLasr.pricedata(partition=(ProductName));
     set sashelp.pricedata;
run;
```

For the PROC LASR step, the PORT= option specifies a network port number to use. The PATH= option specifies the directory in which the server and table signature files are to be stored. The specified directory must exist on each machine in the cluster. The NODES=ALL option in the PERFORMANCE statement requests that the LASR Analytic Server run on all the nodes on the appliance. You can start a LASR Analytic Server on a subset of the nodes on an appliance, but this might affect whether high-performance analytical procedures can run alongside the LASR Analytic Server. For more information, see the section "Alongside-LASR Distributed Execution on a Subset of the Appliance Nodes" on page 371.

The LIBNAME statement assigns the MyLasr libref to the LASR Analytic Server instance so that it can be accessed by SAS procedures and the DATA step. The DATA step demonstrates one way to stored partitioned time series data into a LASR server so that it can be used by the HPFORECAST procedure.

For more information about the LASR Analytic Server or using the SASIOLA engine, see the *SAS LASR Analytic Server: Reference Guide*.

### Running PROC HPFORECAST Alongside the SAS LASR Analytic Server Instance

The following code demonstrates how you can use the MyLasr libref to specify the input data for PROC HPFORECAST. This code performs automatic model identification, model selection, and forecasting for monthly sales (Sale). During model identification, it considers some likely influential predictor variables of interest (Discount and Pice). Three levels of hierarchy are forecast by this one PROC HPFORECAST step.

```
proc hpforecast data=MyLasr.PriceData
                mode=auto
                print=perform;
   timeid date interval=month;
   output outfor=MyLasr.SalesFor
          outlog=MyLasr.SalesLog;
   forecast sale/accumulate=total;
   input price/accumulate=average;
   input discount/accumulate=average;
   level ProductName RegionName;
   end level;
   level ProductName;
```

```
   end level;
   level _top_;
   end level;
run;
```

Because you previously specified the GRIDHOST= environment variable and the input data are held in distributed form in the associated server instance, this PROC HPFORECAST step runs in distributed mode alongside the LASR Analytic Server, as indicated in the "Performance Information" table shown in Figure 12.1.

**Figure 12.1** Performance Information

**The HPFORECAST Procedure**

| Performance Information | |
|---|---|
| **Host Node** | greenarrow.unx.sas.com |
| **Data Server** | greenarrow |
| **Execution Mode** | Distributed |
| **Grid Mode** | Symmetric |
| **Number of Compute Nodes** | 16 |
| **Number of Threads per Node** | 24 |

Two output tables that are added to the LASR Analytic Server instance from this PROC HPFORECAST step. Output data sets do not have to be created in the same server instance that holds the input data; you can use a different LASR Analytic Server instance to hold the output data. However, in order for the output data to be created in alongside mode, all the nodes that are used by the server instance that holds the input data must also be used by the LASR instance that holds the output data.

#### *Terminating a SAS LASR Analytic Server Instance*

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

```
proc lasr term port=9152;
run;
```

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

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

### Running PROC HPFORECAST in Asymmetric Mode

Asymmetric mode is commonly used when the data appliance and the computing appliance are distinct appliances. It is not necessary for distributed-mode PROC HPFORECAST software to be installed on the data appliance in order to be able to use the appliance as a data provider for the HPFORECAST procedure

when it runs in asymmetric mode on another appliance. However, it is essential that a SAS Embedded Process be installed on the data appliance and that the distributed-mode software be installed on the computing appliance.

There is no advantage to executing PROC HPFORECAST in asymmetric mode on one appliance, because data might have to be unnecessarily moved between nodes. The advantage of using different computing and data appliances is that the data appliance is not affected by the execution of PROC HPFORECAST except during the initial parallel data transfer. A potential disadvantage of this asymmetric mode of execution is that the performance can be limited by the bandwidth with which data can be moved between the appliances. However, because this data movement takes place in parallel from the nodes of the data appliance to the nodes of the computing appliance, this potential performance bottleneck can be overcome with appropriately provisioned hardware.

If you plan to make repeated use of the same data, another way to overcome the data transfer latency between distinct appliances is to use the LASR Analytic Server on the compute appliance to load the data once with the desired partitioning, and then reuse it as needed. Output tables from your PROC HPFORECAST runs can be stored in the same LASR instance. Other SAS processing can access those results by using the SASIOLA LIBNAME engine, and other SAS high-performance analytical procedures can access those results for their own distributed-mode processing. When you are finished with all related processing, you terminate the LASR server instance.

## Alongside-HDFS Execution

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

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

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

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

### *Alongside-HDFS Execution by Using the SASHDAT Engine*

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

Before you can run a procedure alongside HDFS, you must distribute the data to the cluster. The following statements use the SASHDAT engine to distribute the Sashelp.Pricedata data set to HDFS in a way that is analogous to the previous LASR Analytic Server example:

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

libname hdatLib sashdat path="/hps";

data hdatLib.PriceData(replace=yes partition=(ProductName)) ;
   set sashelp.pricedata;
run;
```

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

The following SAS code demonstrates the use of PROC HPFORECAST in alongside-HDFS mode. These statements are almost identical to the preceding PROC HPFORECAST example for the LASR server.

```
proc hpforecast data=hdatLib.PriceData
                mode=auto
                print=perform;
   timeid date interval=month;
   output outfor=hdatLib.SalesFor
          outlog=hdatLib.SalesLog;
   forecast sale/accumulate=total;
   input price/accumulate=average;
   input discount/accumulate=average;
   level ProductName RegionName;
   end level;
   level ProductName;
   end level;
   level _top_;
   end level;
run;
```

Figure 12.2 shows the "Performance Information" table, which shows that the procedure ran in distributed mode.

**Figure 12.2** Alongside-HDFS Execution Performance Information

**The HPFORECAST Procedure**

| Performance Information | |
|---|---|
| **Host Node** | hpagrid |
| **Data Server** | greenarrow |
| **Execution Mode** | Distributed |
| **Grid Mode** | Symmetric |
| **Number of Compute Nodes** | 13 |
| **Number of Threads per Node** | 24 |

Be aware that the SASHDAT LIBNAME engine supports write-only access to the HDFS store. If you generate PROC HPFORECAST output data sets to HDFS, as in this example, then you can read them (process them) only in the context of some other SAS high-performance analytics procedure.

```
proc print data=hdatLib.SalesFor;
run;
```

For example, the preceding PROC PRINT code to display the OUTFOR= data set from the earlier PROC HPFORECAST step results in the following error:

```
ERROR: The SASHDAT engine is a uni-directional engine. Data flows from the SAS client to the
       Hadoop Distributed File System. The engine cannot be used to fetch data from HDFS.
```

### Output Data Sets

In alongside-the-data execution mode, the data are read in distributed form, minimizing data movement for best performance. Similarly, when you write output data sets and PROC HPFORECAST executes in distributed mode, the data can be written in parallel into the distributed store.

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

## Data Preparation and Staging

The time series data engine (TSDE) component used by PROC HPFORECAST is adaptable to different data distributions that can be used to satisfy a given set of LEVEL blocks in a PROC HPFORECAST step. When PROC HPFORECAST runs in the single-machine mode, data preparation is not an issue. No special data partitioning or ordering is needed. The remainder of this discussion applies to distributed mode execution.

Distributed mode execution for data sources that use distributed data base technology provides a powerful way to process a large time series data set in parallel with very high scalability for many use cases. To make use of this, the data set must be distributed to the DBMS in such a way that all of the rows that comprise some hierarchy level of interest are atomically located on a single machine. Most distributed DBMS systems support some notion of a distribute-by key or variable set. This mechanism informs the DBMS that rows of the associated data set that have the same value in the distribute-by variables must reside on the same DBMS machine. This mechanism can be used to ensure that all the rows that comprise a given level member in a time series data set are stored on the same DBMS machine and will be delivered to the PROC HPFORECAST processing that co-locates with the DBMS table partition.

The rules that govern admissible data distribution and hierarchy levels in distributed mode are summarized here. Let $\mathcal{D}$ denote the set of distribute-by variables in a distributed data set:

$$\mathcal{D} = \{V_i\}_{i=1}^{N}$$

And let $\mathcal{L}$ denote the set of hierarchy levels of interest for a particular step where there are $M$ different levels in the set $\mathcal{L}$:

$$\mathcal{L} = \{L_j\}_{j=1}^{M}$$

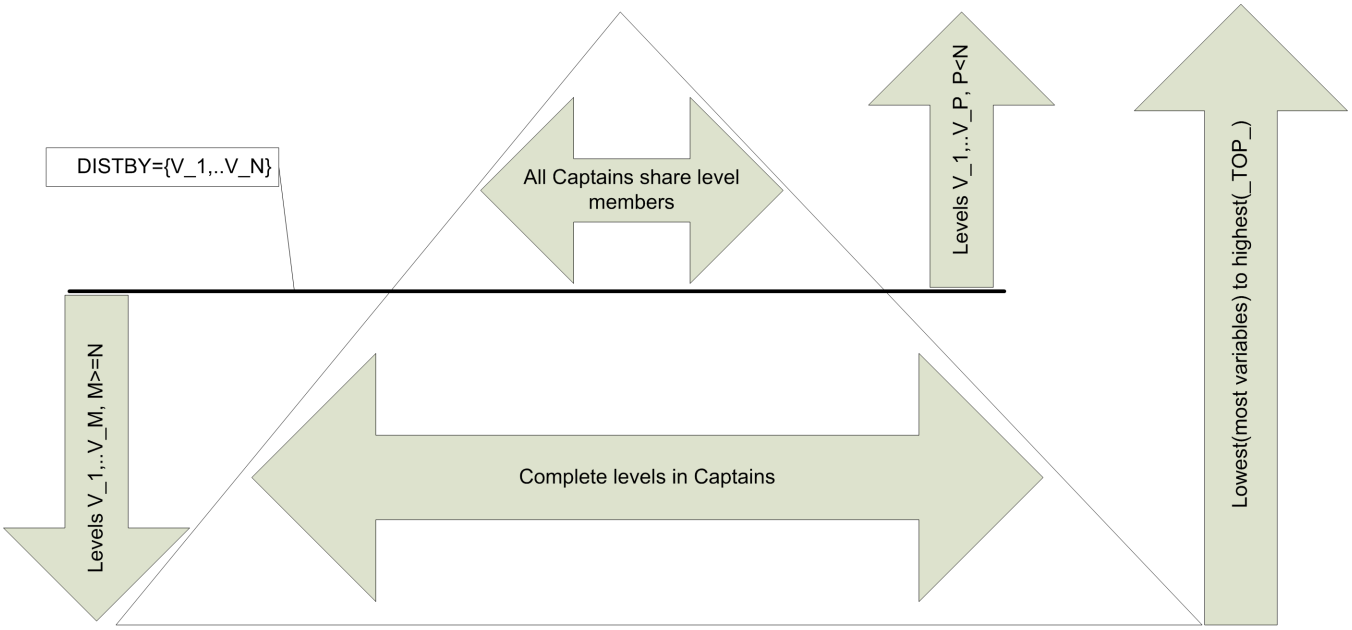
Each level $L_j$ is defined by an ordered set of variables,

$$L_j = (V_{j_1}, V_{j_2}, \ldots, V_{j_P})$$

with the requirement for strict nesting of $L_j$ in $L_{j+1}$. Nesting means that all of the variables in $L_j$ match in order with those in $L_{j+1}$. A level $L_j$ is admissible for local processing in the scope of a given captain for the distribution $\mathcal{D}$ provided that all of the variables in $\mathcal{D}$ are spanned by the leftmost variables in $L_j$. Any level $L_j$ that has a leftmost subset of its variables contained in $\mathcal{D}$ can be processed cooperatively by the PROC HPFORECAST captains sharing their partially complete results for their local level members. These results are shared to and by all captains, and each computes its own aggregation for those fully distributed level members so that these level members are duplicated across the captains. The end result of this exchange is that each captain forms a complete hierarchy member tree for those levels that are logically above the set $\mathcal{D}$ variables.

Figure 12.3 illustrates how the data distribution determines levels that are locally complete in the scope of each captain and how those levels are distributed across all captains.

**Figure 12.3** Distributed Time Series Hierarchy



For levels that require member distribution across the captains, you must include a LEVEL block that is admissible for local processing, in addition to those higher levels that are not locally complete in the scope of any single captain. For example, suppose you have a Cars data set that has a hierarchy of interest defined by the variables Make, Model, and Color, and suppose you distribute the data set via the variables Make and Model when you create the DBMS table. A PROC HPFORECAST run on the Make level alone is inadmissible for this distribution; it must be accompanied in the run by a LEVEL block that includes Make and Model, or Make, Model, and Color. The procedure invocation must include a leaf level that is locally complete to use as the source for the higher levels that might not be locally complete.

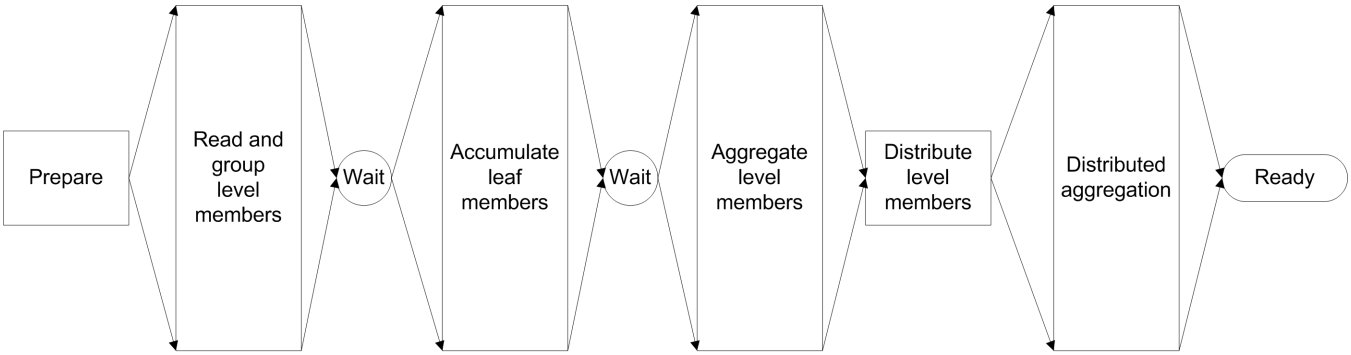
If you want to forecast hierarchy levels logically above the level of the DBMS table's distribution variables and you do not want to redistribute the data, you can use the NOFORECAST option in the LEVEL statement to suppress the generation of forecasts for the associated hierarchy level. The time series for this level are acquired by the time series data engine from the input table for the purpose of providing a starting point for distribution across the captain processes. No analysis or forecasting is performed for a LEVEL block that includes the NOFORECAST option, and no output data is produced for such a LEVEL block.

## Threading

PROC HPFORECAST makes use of threads when possible whether it runs in single-machine mode or distributed mode. In many ways, the single-machine mode execution pattern in the SAS process is mirrored in the distributed mode captain processes. Threading is used in two independent areas of processing. First, threads are used in various steps of time series acquisition in the time series data engine (TSDE). Then, threads are used to forecast the time series that are acquired during the first step. The use of threading during time series acquisition, and then during time series analysis and forecasting follows the same pattern in either mode.

Figure 12.4 depicts the flow of the steps that are involved in time series acquisition in the TSDE component. Note that there are four distinct that fan out to perform parallel processing.

**Figure 12.4** Time Series Data Engine Processing Flow




The function of each of these steps is briefly described here:

Prepare This step performs one-time setup operations that are necessary to acquire the time series from the input data set.

Read and group level members This step reads rows from the data source and groups the rows by the values of their level variables. Levels are enumerated in parallel when more than one LEVEL block is specified in the PROC HPFORECAST step. Rows are grouped to their respective level member at the leaf level of the hierarchy. When this step completes, the per-level member sets that are local to the associated process are enumerated. See ReadTime and GroupTime in the TSDEDetails ODS table.

Accumulate level members This step fans out in parallel over the leaf-level member set. Each thread operates on a different member of the member set, performs time series accumulation for the analysis variables, and saves the resulting time series along with the associated time ID array into a time series data frame. See AccTime in the TSDEDetails ODS table.

Aggregate level members This step fans out in parallel over the higher-level members that are local to the associated process. Levels are dispatched from the lowest non-leaf level to the highest non-leaf level that is local to the associated process. For a given level, each thread operates

on a different member of the level's member set and performs time series aggregation by using the leaf-level time series for the member's descendants as its source. Each level member generates a time series data frame that holds the member's time series variables and associated time ID. See AccTime in the TSDEDetails ODS table.

Distribute level members This step and the next apply only in distributed mode. The level members for each distributed level of the hierarchy that are local to a given captain are shared with its peer captains. When this step finishes, all captains have the same member sets for the distributed levels of the run. It remains for them to perform distributed aggregation of those members so that each has a completely formed time series for all of the contributing members. The time spent in this step is part of DistTime in the TSDEDetails ODS table.

Distributed aggregation This step fans out in parallel over the distributed levels of the hierarchy. The threads in each captain cooperatively aggregate each member from a distributed level's member set. The thread in each captain shares its partial result for a given distributed level member and simultaneously collects the contributions of all its peer captains, and then each captain applies those collected partial results to form a complete aggregation of the time series for the distributed level member. Each captain holds the aggregated result for the distributed level member. The time spent in this step is contained in DistTime in the TSDEDetails ODS table.

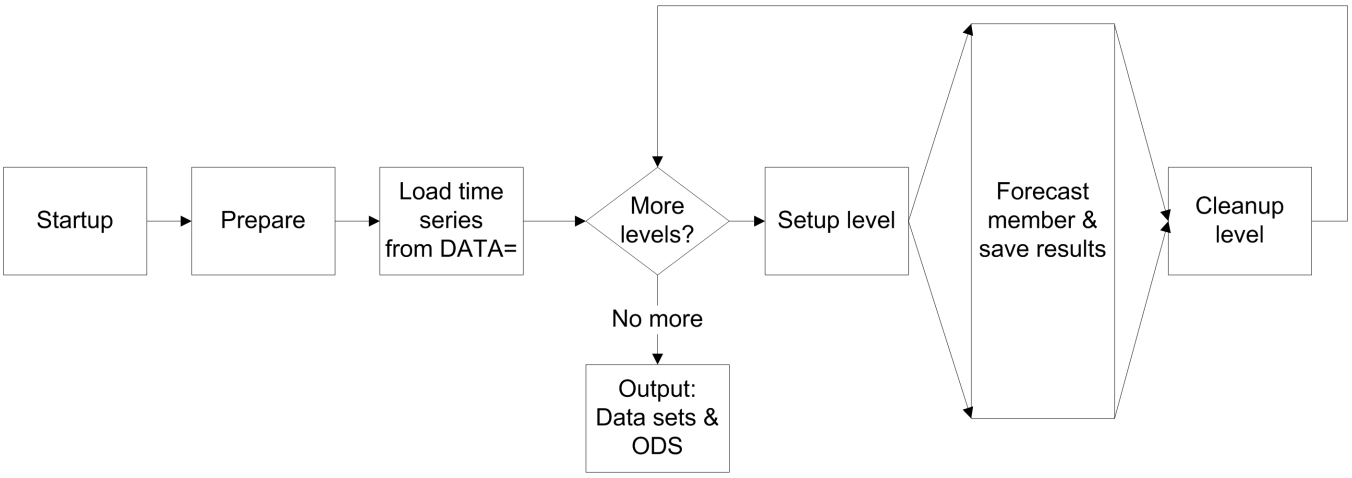
For single-machine mode, an optimization is triggered when the input data set is sorted by a set of variables that spans the set of variables that make up the leaf level of the hierarchy. This optimization is a memory-versus-time tradeoff that significantly reduces the transient memory that is required to acquire the time series for the leaf-level members. There are two ways to affect sorted-by information for the input data set:

- A PROC SORT step saves the sorted-by information for the BY variables in the metadata of its output data set.
- The SORTEDBY= data set option declares the sorted-by information for the input data set that is specified for the PROC HPFORECAST step.

When this optimization is triggered, the time series for the leaf-level members are accumulated as the rows are read from the input data set. No threads are used during the leaf-level processing in this special case. This direct accumulation strategy avoids the need to buffer the analysis variables from the input rows for subsequent parallel accumulation.

As mentioned previously, threads are also used to forecast the time series in parallel. The levels of the hierarchy are processed from lowest to highest. Within each level, a thread assumes responsibility for analyzing and forecasting all of the target (dependent) variables that are specified in the PROC HPFORECAST invocation. The available threads cooperatively process all of the members in a given level and then move on to the next higher level until all hierarchy levels that are specified in the PROC HPFORECAST invocation are completed. Figure 12.5 shows the control flow for the time series analysis and forecasting.

**Figure 12.5** PROC HPFORECAST Processing Flow




The function of each of these steps is briefly described here:

| | |
|---|---|
| Startup | This step initiates the services that are needed to perform the processing in the PROC HPFORECAST invocation. In single-machine mode mode, this is a simple and fast step. In distributed mode, this step involves launching the distributed processes that carry the PROC HPFORECAST processing out to the grid machines to run alongside the DBMS table partitions, and it can be several seconds in duration. See the ODS Timing table. |
| Prepare | This step performs one-time setup necessary to execute the processing that is specified in the PROC HPFORECAST invocation. See the Timing ODS table. |
| Load time series | This step expands into the TSDE process flow (Figure 12.4). See DataTime in the Summary ODS table. |
| Setup level | This step performs one-time setup that is necessary to forecast the targets for a given level of the hierarchy. The time spent in this step is part of ForecastTime in the Summary ODS table. |
| Forecast member | This step fans out in parallel over the member set for a given level of the hierarchy. Threads take members of the level to process. All targets for the member are analyzed and forecast by the same thread, and results for that target's forecast are saved to an internal store. The available threads cooperatively process all members of a given level |

before moving to the next level. The time spent in this step is part of ForecastTime in the Summary ODS table.

Cleanup level — This step is a rendezvous to clean up any resources that are no longer needed after the level completes. The time spent in this step is part of ForecastTime in the Summary ODS table. When this step completes, the next higher level of the hierarchy is initiated if there is one; otherwise, the forecasting portion of the PROC HPFORECAST invocation is complete.

Output — This step generates any requested output data sets and ODS tables. These are serial operations. Each output data set must be completed before the next one can begin. Output data sets are saved first, followed by the generation of any requested ODS tables. See OutputTime in the Summary ODS table and the Timing ODS table.

## Output Data Considerations

PROC HPFORECAST output data sets can be written back to the DBMS in distributed mode. Each captain process writes the output for any of the hierarchy levels that it processes. A special case occurs when there are hierarchy levels that are incompletely resolved to the scope of one single captain. Part of the level member distribution process described in the preceding section assigns each level member to a captain for the purpose of producing its output. All captains might proceed to model and forecast the associated time series for purposes of their own local computational needs, even though only one is designated to produce the output data set results for that level member.

It is common for many SAS programs to make assumptions about row order in data sets. You need to be careful when you use the results from PROC HPFORECAST that are written back to DBMS tables for distributed mode. The rows for a particular output data set are written in the scope of each captain process in such a way that all the rows for each level member are written in sequence, although the order of the level members does not reflect their sort order. For output data sets that store time series data, the rows are written in natural time ID sequence. However, when you read those output tables in a subsequent step of your SAS job, the row order that is read by SAS from the DBMS table is not guaranteed to preserve the time ID sequence. If it is important to process rows in groups by their qualifying level members, you must take steps in your SAS code to affect that grouping. When such a group represents a time series, you must further take steps in your SAS code to affect the ordering of the rows by the time ID value. Neither of these is ensured by the distributed DBMS when it returns rows from those PROC HPFORECAST output data sets.

## Time Series Acquisition Details

PROC HPFORECAST starts with a row-oriented data source that is specified by the DATA= option. It must transform the rows of that data source into time series for each level of the hierarchy from the LEVEL statements that are specified for the procedure invocation.

All time series variables (analysis variables) that are specified for processing during the PROC HPFORECAST invocation have an accumulation mode. The accumulation type for a given variable might come from the TIMEID statement, a FORECAST statement, or a LEVEL statement.

The conceptual model used for processing multiple levels in a single invocation can be viewed as follows:

- Each analysis variable is viewed as having a level-specific accumulation mode, although the variables might be constrained to use the same accumulation mode across all levels, depending on the variable's role.
- For the leaf level of the hierarchy, the observations that have timestamp values that resolve to the same time index are accumulated as directed by each analysis variable's ACCUMULATE= option value. A time series group is formed for each level member at the leaf level. This time series group contains the array of uniformly spaced time ID values, and an array of values for each analysis variable.
- For higher levels of the hierarchy, the leaf-level time series for each analysis variable is temporally aggregated, based on the parent-child relationships between the higher level members, and all their leaf-level children. All higher levels use the leaf-level time series groups as their data source, not the raw observations that are the source for the leaf-level time series groups.

## Accumulation

You can use the HPFORECAST procedure to forecast both time series data and transactional data. If the data are transactional, then the procedure must first accumulate the data into a time series before the data can be forecast.

All time series variables that are specified during the PROC HPFORECAST invocation have some accumulation mode. According to that accumulation mode, data set observations for the variable are accumulated within each time period. The frequency (width of each time interval) is specified by the INTERVAL= option. The TIMEID variable contains the data set observation's timestamp values. Each timestamp value corresponds to a specific time period. Accumulation is particularly useful when the input data set contains transactional data, whose observations are not spaced with respect to any particular time interval. The accumulated values form the actual time series that is used in subsequent analyses.

For example, suppose a data set contains the following observations:

```
19MAR1999    10
19MAR1999    30
11MAY1999    50
12MAY1999    20
23MAY1999    20
```

If INTERVAL=MONTH is specified, all the preceding observations fall within three time periods of March 1999, April 1999, and May 1999. By using different values of the ACCUMULATE= option, the following results are produced for this data:

ACCUMULATE=TOTAL

```
O1MAR1999    40
O1APR1999    .
O1MAY1999    90
```

ACCUMULATE=AVERAGE

```
O1MAR1999    20
O1APR1999    .
O1MAY1999    30
```

ACCUMULATE=MINIMUM

```
O1MAR1999    10
O1APR1999    .
O1MAY1999    20
```

ACCUMULATE=MAXIMUM

```
O1MAR1999    30
O1APR1999    .
O1MAY1999    50
```

ACCUMULATE=STDDEV

```
O1MAR1999    14.14
O1APR1999    .
O1MAY1999    17.32
```

As you can see from the preceding examples, even though the data set observations contained no missing values, the accumulated time series might have missing values.

### Missing Value Interpretation

Sometimes missing values should be interpreted as unknown values. The forecasting models used by the HPFORECAST procedure can effectively handle missing values (see the section "Missing Value Modeling Issues" on page 388). But sometimes missing values are known (such as when missing values are created from accumulation), and no observations should be interpreted as no (zero) value. In the former case, you can specify the SETMISSING= option to interpret how missing values are treated. Specify SETMISSING=0 when missing observations are to be treated as no (zero) values. In other cases, missing values should be interpreted as global values, such as minimum or maximum values of the accumulated series. The accumulated and interpreted time series is used in subsequent analyses.

## Time Indexing in the Forecasting Process

To make concise the time spans that are associated with time series processing in PROC HPFORECAST, the following notation is defined. These values should be viewed as defined within the scope of a particular time series group (level member) for a particular level of the hierarchy. Therefore, these values are contextual in nature. To avoid adding to the complexity of the notation, this level-member association is omitted. These values are computed for the accumulated time series.

$T$ denotes the total number of TIMEID values. Time series actual values (possibly missing) are defined for the index range $1 \ldots T$.

$t_j^b$ denotes the time index of the first nonmissing value for forecast variable $j$.

$t_j^e$ denotes the time index of the last nonmissing value for forecast variable $j$.

$t_j^f$ denotes the time index of the start of the forecast horizon for forecast variable $j$. The value used is determined by several factors that include the model family used, the value of the HORIZONSTART= option in the TIMEID statement, and the value of the BACK= option in the PROC HPFORECAST statement. When the HORIZONSTART= option is specified, this value is defined by

$$t_j^f = \min\{t_j^e + 1, t_{HZ}\}$$

where $t_{HZ}$ denotes the time index of the HORIZONSTART= timestamp that is resolved with respect to the first (smallest) time value for the associated series group. When the HORIZONSTART= option is not specified, this value is defined by

$$t_j^f = t_j^e + 1$$

$H$ denotes the number of periods of multistep forecasting, which is specified by the LEAD= option in the PROC HPFORECAST statement.

$B$ denotes the number of periods of out-of-sample performance evaluation for the final forecast. This is number is specified in the BACK= option in the PROC HPFORECAST statement.

$O_j$ denotes the number of holdout (out-of-sample) periods to be used to select the best performing forecast from among the possible candidates for forecast variable $j$. The number depends on the HOLDOUT= and HOLDOUTPCT= options in the PROC HPFORECAST statement, the length of the time series for forecast variable $j$, and whether the associated time series is classified as intermittent or not. When $O_j$ exceeds the available observations in the time series for variable $j$, $O_j$ is set to 0.

The time spans of interest for the forecasting process are defined as follows:

Initial Fit The fit region used for model selection for forecast variable $j$ is defined by

$$[1, t_j^f - B - O_j - 1]$$

Model Selection When holdout samples are used for model selection for forecast variable $j$, the model selection region is defined by

$$[t_j^f - B - O_j, t_j^f - B - 1]$$

When no holdout samples are used for model selection, the model selection region is defined by

$$[1, t_j^f - B - 1]$$

Final Fit — The fit region for the selected model for forecast variable $j$ is defined by

$$[1, t_j^f - B - 1]$$

Final Forecast — The forecast region for forecast variable $j$ is defined by

$$[t_j^f - B, t_j^f - B + H - 1]$$

Performance — The performance region for the selected model for forecast variable $j$ is defined by

$$[t_j^f - B, t_j^f - 1]$$

## Differences between MODE=SIMPLE and MODE=AUTO

For automatic time series forecasting, MODE=SIMPLE can be best compared most directly to the capabilities of the HPF procedure, whereas MODE=AUTO can be best compared to the joint capabilities of the HPFDIAGNOSE and HPFENGINE procedures. Specifying the MODE=SIMPLE option in the PROC HPFORECAST statement offers a limited set of time series models that are restricted to ESM and IDM models and a limited set of controls over the behavior of those models. Specifying the MODE=AUTO option in PROC HPFORECAST offers the full set of time series model families available in PROC HPFDIAGNOSE and PROC HPFENGINE. By specifying MODE=AUTO, you can access all of the controls for model identification available in PROC HPFDIAGNOSE, and you can use all of the time series model definition procedures that can be used with PROC HPFENGINE to create user-defined time series models.

For MODE=AUTO, the following procedures are of particular interest:

- PROC HPFDIAGSPEC enables you to create customized diagnostic control preferences. For more information, see Chapter 6, "The HPFDIAGSPEC Procedure."
- PROC HPFESMSPEC enables you to create customized ESM model specifications. For more information, see Chapter 8, "The HPFESMSPEC Procedure."
- PROC HPFARIMASPEC enables you to create customized ARIMA model specifications. For more information, see Chapter 4, "The HPFARIMASPEC Procedure."
- PROC HPFUCMSPEC enables you to create customized UCM model specifications. For more information, see Chapter 17, "The HPFUCMSPEC Procedure."
- PROC HPFIDMSPEC enables you to create customized IDM model specifications. For more information, see Chapter 11, "The HPFIDMSPEC Procedure."
- PROC HPFSELECT enables you to create customized model selection and model combination lists. For more information, see Chapter 15, "The HPFSELECT Procedure."

PROC HPFORECAST enables you to specify diagnostic control specifications in the following contexts:

Procedure — by the use of the GLOBALDIAGNOSE= option in the PROC HPFORECAST statement.

Variable — by the use of the DIAGNOSE= option in the FORECAST statement.

Level — by the use of the DIAGNOSE property in the SET statement for FORECAST variables.

PROC HPFORECAST enables you to specify user-defined model specifications in the following contexts:

| | |
|---|---|
| Procedure | by the use of the GLOBALSELECTION= option in the PROC HPFORECAST statement. |
| Variable | by the use of the SELECTION= option in the FORECAST and INPUT statements. |
| Level | by the use of the SELECTION property in the SET statement for FORECAST variables and INPUT variables. |

## Caveats: MODE=AUTO versus PROC HPFDIAGNOSE and PROC HPFENGINE

Using MODE=AUTO in the HPFORECAST procedure offers much, but not all, of the automation for time series modeling and forecasting that is present in the HPFDIAGNOSE and HPFENGINE procedures. It is useful to summarize where they are different:

OUTEST data set: The role and content of the OUTEST= data set from PROC HPFORECAST is most like the OUTEST= data set from PROC HPFENGINE. There is no OUTEST= data set that captures the state from the model identification step as performed by HPFDIAGNOSE apart from the model selection and forecasting step as performed by HPFENGINE.

TASK= option: There is no TASK= option in PROC HPFORECAST. When MODE=AUTO, there is no provision to impose fine-grained control over the automatic forecasting process as there is for the TASK= option in PROC HPFENGINE. Using MODE=AUTO in PROC HPFORECAST always performs the entire automatic forecasting process, which includes model identification, model selection, and forecasting based on the selected model.

INEST data set: No INEST= data set is allowed in PROC HPFORECAST. There are no TASK= modes as there are in PROC HPFENGINE; therefore, there is no reason to replay model selection parameter estimates from a previous PROC HPFORECAST invocation.

Events: PROC HPFORECAST does not support parametric events. This includes all uses of event dummies whether they arise from explicit or implicit use. Explicit use includes these areas:

- EVENT statement for PROC HPFDIAGNOSE
- event use in user-defined models from PROC HPFARIMASPEC or from PROC HPFUCMSPEC
- symbol mappings in PROC HPFSELECT via the EVENTMAP option in your SPEC statements

Implicit use arises from outlier detection in ARIMA models; therefore, outlier detection is suppressed in PROC HPFORECAST.

You can use PROC HPFEVENTS to generate event dummies as explicit variables that are contained in the input data set that you specify via the PROC HPFORECAST DATA= option, and then specify those variables as input variables by using the PROC HPFORECAST INPUT statement. For more information, see Chapter 9, "The HPFEVENTS Procedure"

OUTCOMPONENT data set: PROC HPFORECAST does not support the OUTCOMPONENT= data set.

OUTPROCINFO data set: PROC HPFORECAST provides a great deal of execution detail in the form of ODS tables from the various PRINT= options that can be specified. The information available in these ODS tables greatly exceeds the information available in the OUTPROCINFO= data set in the HPFDIAGNOSE and HPFENGINE procedures.

## Conceptual View of the Forecasting Process

The HPFORECAST procedure uses the following steps to produce forecasts for each time series group (level member) within each level. This process applies to each forecast variable that is specified in the PROC HPFORECAST invocation. The options that control the step are listed to the right:

**Table 12.3** PROC HPFORECAST Processing Steps and Control Options

| Step | Operation | Option | Statement |
|---|---|---|---|
| 1 | Model diagnosis | INTERMITTENT= | PROC HPFORECAST |
| | | MODE= | PROC HPFORECAST |
| | | MODELREPOSITORY= | PROC HPFORECAST |
| | | GLOBALDIAGNOSE= | PROC HPFORECAST |
| | | DIAGNOSE= | FORECAST |
| | | DIAGNOSE property | SET |
| 2 | Model selection | HOLDOUT=, | PROC HPFORECAST |
| | | HOLDOUTPCT= | PROC HPFORECAST |
| | | CRITERION= | PROC HPFORECAST |
| | | MODE= | PROC HPFORECAST |
| | | MODELREPOSITORY= | PROC HPFORECAST |
| | | GLOBALSELECTION= | PROC HPFORECAST |
| | | SELECTION= | FORECAST |
| | | SELECTION property | SET |
| 3 | Parameter estimation | | FORECAST |
| 4 | Forecasting | LEAD= | PROC HPFORECAST |
| 5 | Statistics of fit | OUTSTAT= | OUTPUT |
| | | OUTSTATSELECT= | OUTPUT |
| | | BACK= | PROC HPFORECAST |

Each of the steps shown in Table 12.3 is described in the following sections.

### Model Diagnosis

The MODE= option in the PROC HPFORECAST statement determines the level of diagnostic tests performed. If you specify MODE=SIMPLE, you are restricted to the use of exponential smoothing models, and you can optionally include intermittent demand models. If you specify MODE=AUTO, you can gain access to fully automatic model identification, model selection, and forecasting services. MODE=SIMPLE is the default.

When MODE=SIMPLE, the following diagnostic tests are available:

- testing the dependent series for intermittent demand. The INTERMITTENT= option in the PROC HPFORECAST statement sets the threshold that is used to categorize a series as intermittent or not

intermittent. If the INTERMITTENT= option is not specified, no intermittency test is performed on the series for the FORECAST variables, and only exponential smoothing models are considered.

- testing the dependent series for a seasonal pattern.

When MODE=AUTO, a complex array of diagnostic controls are available to control different aspects of model identification. The HPFORECAST procedure includes a default set of diagnostic control options, which you can use by simply specifying MODE=AUTO in the PROC HPFORECAST statement. You can also define your own diagnostic control preferences by using the HPFDIAGSPEC procedure. When you use PROC HPFDIAGSPEC to supply the diagnostic control preferences, those controls completely replace the default settings in the respective scope for which that set of diagnostic control preferences is used. For example, if you specify the DIAGNOSE= option in a FORECAST statement, those diagnostic control preferences apply to all the variables that are specified in that FORECAST statement. Diagnostic control preferences that are used for dependent variables in other FORECAST statements would either be the settings from the GLOBALDIAGNOSE= option in the PROC HPFORECAST statement (if that option is specified) or the built-in defaults. You can control all aspects of model diagnosis and identification through PROC HPFDIAGSPEC and the various PROC HPFORECAST language features that are designed to integrate with it.

These aspects include the following:

- testing dependent and independent series for transformation
- testing dependent series for stationarity and seasonality
- testing the dependent series for intermittent demand
- testing input series for multi-collinearity
- identification of well-suited ARIMAX models for the dependent series by using both ARIMA-REG and REG-ARIMA order
- identification of a well-suited ESM model for the dependent series
- identification of a well-suited UCM model for the dependent series
- automatic generation of a combined model for the ESM, ARIMA, and UCM models that are generated for the dependent series
- automatic inclusion of user-defined models into the model selection process
- specification of diagnostic preferences at the procedure scope, at FORECAST scope, and at LEVEL scope

## Model Selection

PROC HPFORECAST uses the model selection criterion (fit statistic) to compare forecasts for each candidate model. For MODE=SIMPLE, the fit statistic is determined by the CRITERION= option in the PROC HPFORECAST statement and defaults to CRITERION=RMSE. For MODE=AUTO, the fit statistic is determined by the CRITERION= setting in the dependent variable's diagnostic control specification, and can

be overridden by the CRITERION= option in the PROC HPFORECAST statement if the OVERRIDE=YES option is also specified.

PROC HPFORECAST computes the selection criterion either from the multistep forecasts in the holdout sample range (if the HOLDOUT= or HOLDOUTPCT= options are specified) or by the one-step-ahead forecasts for the full range of the time series (if the HOLDOUT= and HOLDOUTPCT= options are not specified). The candidate model that has the best selection criterion value is selected to forecast the time series. When a holdout sample is used, the selected model is estimated again by including the observations in the holdout sample in order to produce the final forecast.

You can also specify a custom (user-defined) model selection list for independent variables that are used either in the automatically generated models or in user-defined models. You can specify this user-defined list in the SELECTION= option in the INPUT statement or by using the SELECTION property in the SET statement for an input variable. A selection list carries its own CRITERION= option from the SELECT statement in PROC HPFSELECT, and that option is used to affect model selection during the generation of forecasts for independent variables when this mechanism is used. By default, PROC HPFORECAST generates forecasts for independent variables by using the intrinsic ESMBEST model, which always uses CRITERION=RMSE to select the best performing forecast over the historical region of the independent series. The OUTSTATSELECT= data set records model selection information for independent variables, and the OUTEST= data set records parameter estimates for the selected model for them.

## Parameter Estimation

All parameters that are associated with each model are optimized based on the data and default parameter restrictions. When a transformed model is estimated, the transformed time series data are used to estimate the model parameters. Model parameters can be estimated twice in the course of an automatic forecasting run: once for the model selection step which might include the use of a holdout sample, and then once again for the selected model with holdout samples restored to the selected model's fit region.

## Missing Value Modeling Issues

The treatment of missing values varies with the forecasting model. For the smoothing models, missing values after the start of the series are replaced with one-step-ahead predicted values, and the predicted values are applied to the smoothing equations. For more information about how missing values are treated in smoothing models, see Chapter 19, "Forecasting Process Details."

For intermittent demand models, missing values are assumed to be periods of no demand, regardless of where they appear in the time span of the actual series.

You can specify the treatment of missing values during time series acquisition in the SETMISSING= option, which changes the missing values in the accumulated series prior to modeling.

Even though all the observed data are nonmissing, using the ACCUMULATE= option can create missing values in the accumulated series.

## Forecasting

After the model parameters are estimated using data from the time series fit region, one-step-ahead forecasts are generated for the full range of the actual (optionally transformed) time series data, and multistep forecasts are generated from the end of the fit region to the end of the forecast horizon.

If there are missing values at the end of the time series, the forecast horizon might be greater than that specified by the LEAD= option.

### Inverse Transformations

When transformed models are included in the MODE=AUTO context, whether by model diagnostics or by user-defined models, the forecasts of the transformed time series are inverse transformed for purposes of saving forecasts and computing fit statistics. By default, the mean (expected) forecasts are generated. For automatically generated models, you can control this by using the MEDIAN option in the PROC HPFDIAGNOSE TRANSFORM statement; for user-defined models, you can control this in the respective model specification procedure. If MEDIAN is specified, the median forecasts are generated instead.

### Statistics of Fit

The statistics of fit (goodness-of-fit statistics) are computed from the actual time series data and the generated forecasts. When transformed models are used, the statistics of fit are based on the inverse transformed forecasts.

## Output Data Set Details

The HPFORECAST procedure creates the following output data sets for each of the following OUTPUT statement options that you specify:

- OUTEST=
- OUTFOR=
- OUTINDEP=
- OUTLOG=
- OUTMODELINFO=
- OUTSTAT=
- OUTSTATSELECT=
- OUTSUM=

In general, if the forecasting process for a particular time series fails, the output that corresponds to that series is set to missing in the relevant output data set. The OUTLOG= data set records an error synopsis for series that fail to forecast.

### OUTEST= Data Set

The OUTEST= data set contains the variables that are specified in the lowest hierarchy level for the PROC HPFORECAST invocation and the variables listed in this section. For variables that are listed in FORECAST statements, a set of rows is written to record the parameter estimates of the model that is used to affect the final forecast. The OUTEST= data set schema changes as a function of the forecasting mode used.

When MODE=SIMPLE, the following variables describe the parameter estimates:

| | |
|---|---|
| _EST_ | parameter estimate |
| _MODEL_ | forecasting model used |
| _NAME_ | forecast variable name |
| _PARM_ | parameter name |
| _PVALUE_ | parameter estimate *p*-value |
| _STDERR_ | standard error of parameter estimate |
| _TRANSFORM_ | transformation used |
| _TVALUE_ | parameter estimate *t* value |

When MODE=AUTO, the following variables describe the parameter estimates:

| | |
|---|---|
| _COMPMODEL_ | model portion of an intermittent demand component |
| _COMPONENT_ | model component (for example, AR, MA, trend, and so on) |
| _DSVAR_ | data set variable mapping |
| _EST_ | parameter estimate |
| _FACTOR_ | model factor |
| _LABEL_ | parameter label |
| _LAG_ | lag of input |
| _MODEL_ | name of model |
| _MODELVAR_ | model variable mapping |
| _NAME_ | variable name |
| _PARM_ | parameter name |
| _PVALUE_ | parameter estimate *p*-value |
| _SELECT_ | name of selection list |
| _SHIFT_ | shift |
| _TRANSFORM_ | transformation applied |
| _TVALUE_ | parameter estimate *t* value |
| _STATUS_ | indicates success or failure in estimating parameter. See also the OUTSUM= data set. |
| _STDERR_ | parameter estimate standard error |

If the parameter estimation step fails for a particular variable, a row that contains missing values is recorded.

## OUTFOR= Data Set

The OUTFOR= data set contains the variables that are specified in the lowest hierarchy level for the PROC HPFORECAST invocation, the time ID variable, and the variables listed in this section. For variables that are listed in FORECAST statements, a series of rows are written to the OUTFOR= data set that spans a range of time indices of the variable's time series. If the NOOUTALL option is specified in the associated OUTPUT statement, then rows are written over the range of multistep forecasting; otherwise, rows are written for the entire span of time indices for the FORECAST variable. The following variables contain observations that are related to the forecasting step:

| | |
|---|---|
| _NAME_ | forecast variable name |
| ACTUAL | actual values |
| ERROR | prediction errors |
| LOWER | lower confidence limits |
| PREDICT | predicted values |
| STD | prediction standard errors |
| UPPER | upper confidence limits |

If the forecasting step fails for a particular variable, a missing value forecast is generated over the span of time indices. If transforms are used to affect the forecasts, the values that are listed in this section are the inverse transform forecasts.

### OUTINDEP= Data Set

This data set stores the time series that are used as explanatory variables in the final forecast for any FORECAST variable. An explanatory series is recorded just once regardless of how many FORECAST variables might make use of it in the generation of their respective final forecasts. This data set contains variables that are specified for the lowest level of the hierarchy for the PROC HPFORECAST invocation.

| | |
|---|---|
| _NAME_ | variable name |
| _TIMEID_ | time ID values |
| _X_ | values of the input variable _NAME_ |

### OUTLOG= Data Set

The OUTLOG= data set contains the variables that are specified for the lowest level of the hierarchy for the PROC HPFORECAST invocation and the variables listed in this section. The OUTLOG= data set records the error synopsis for per-series data-specific or model-specific failures that occur during the PROC HPFORECAST step. Only time series that incur errors produce rows in the OUTLOG= data set. Variables that are reported in the error synopsis include:

| | |
|---|---|
| _MODEL_ | name of the model used |
| _MODELSTATUS_ | a terse description of the error that prohibited the model from executing |
| _NAME_ | forecast variable name |
| _STATUS_ | forecasting status. Nonzero values imply that no forecast was generated for the series. See also OUTSUM= data set. |

### OUTMODELINFO= Data Set

The OUTMODELINFO= data set contains the variables that are specified for the lowest level of the hierarchy in the PROC HPFORECAST invocation and the variables that are listed in this section.

| | |
|---|---|
| _DEPTRANS_ | name of transform that is applied to dependent variable, or NONE if no transform |
| _EVENTS_ | set to 1 if one or more events are present, 0 otherwise |
| _INPUTS_ | set to 1 if one or more inputs are present, 0 otherwise |
| _MODEL_ | model specification name |
| _MODELTYPE_ | model specification type (ARIMA, COMBINED, ESM, UCM, IDM, EXTERNAL, or INACTIVE) |
| _NAME_ | variable name |
| _OUTLIERS_ | set to 1 if one or more outliers are present, 0 otherwise |
| _SEASONAL_ | set to 1 if the model is seasonal, 0 otherwise |
| _SOURCE_ | set to the model source that is based on the extended description element in the model's XML. This value corresponds to the SOURCE= attribute in the XML. The _SOURCE_ value for models that PROC HPFORECAST generates is 'HPFORECAST'. The _SOURCE_ value for models that the HPF XML procedures generate is 'USERSPECIFIED'. |
| _STATUS_ | forecasting status. Nonzero values imply that no forecast was generated for the series. See also the OUTSUM= data set. |
| _TREND_ | set to 1 if the model has a trend, 0 otherwise |

Characteristics that make a model seasonal vary according to the family of model. The following list shows the characteristics of each by model family:

| | |
|---|---|
| ARIMA models | An ARIMA model is considered seasonal if there is a difference whose order equals the seasonal cycle length of the time ID. The presence of MA or AR terms whose lags are equal to the seasonal cycle length also qualifies a model as seasonal. The presence of a predefined seasonal input is another factor that qualifies a model as seasonal. |
| Combined models | A combined model is considered seasonal if any of its contributing candidates is seasonal. |
| IDM | Intermittent demand models are always considered nonseasonal. |
| Smoothing models | The Winters, additive Winters, and seasonal smoothing models are considered seasonal. Linear, simple, double, and damped trend models smoothing are considered nonseasonal. |
| UCM | An unobserved component model is seasonal if there is a seasonal component present, as specified in the SEASON statement in the HPFUCMSPEC procedure, or if there is a predefined seasonal input. |

Likewise, characteristics of a model that indicate a trend vary according to the family of model as follows:

| | |
|---|---|
| ARIMA models | An ARIMA model is considered to have a trend if a first- or second- order difference is applied to the dependent variable. A predefined trend in an input statement also qualifies the model as having a trend component. |
| Combined models | A combined model has a trend if any of its contributing candidates has a trend component. |

| | |
|---|---|
| IDM | Intermittent demand models do have a trend. |
| Smoothing models | The simple and seasonal smoothing models do not have a trend. Linear, double, damped trend models, and multiplicative and additive Winters models do have a trend. |
| UCM | An unobserved component model has a trend if there is a slope component or a predefined trend in an input statement. |

In addition, other OUTMODELINFO attributes are set as follows for combined models:

| | |
|---|---|
| _DEPTRANS_ | set to the common transform that is applied to all of the contributing candidate forecasts, or set to UNKNOWN if the contributing candidate forecasts use a mixture of transform types |
| _EVENTS_ | set to 1 if any of the contributing candidate forecasts has events, 0 otherwise |
| _INPUTS_ | set to 1 if any of the contributing candidate forecasts has inputs, 0 otherwise |
| _OUTLIERS_ | set to 1 if any of the contributing candidate forecasts has outliers, 0 otherwise |

## OUTSTAT= Data Set

The OUTSTAT= data set contains the variables that are specified for the lowest level of the hierarchy for the PROC HPFORECAST invocation and the variables listed in this section. For variables that are listed in FORECAST statements, the following variables contain observations that are related to the statistics of fit for the final forecast:

| | |
|---|---|
| _NAME_ | forecast variable name |
| _REGION_ | the region in which the statistics are calculated. Statistics that are calculated in the fit region are indicated by FIT. Statistics that are calculated in the forecast region (which happens only if the BACK= option is greater than 0) are indicated by FORECAST. |
| DFE | degrees-of-freedom for error |
| N | number of observations |
| NOBS | number of observations used |
| NMISSA | number of missing actuals |
| NMISSP | number of missing predicted values |
| NPARMS | number of parameters |
| TSS | total sum of squares |
| SST | corrected total sum of squares |
| SSE | sum-of-square error |
| MSE | mean square error |
| UMSE | unbiased mean square error |
| RMSE | root mean square error |
| URMSE | unbiased root mean square error |
| MAPE | mean absolute percent error |

| | |
|---|---|
| MAE | mean absolute error |
| MASE | mean absolute scaled error |
| RSQUARE | R-square |
| ADJRSQ | adjusted R-square |
| AADJRSQ | Amemiya's adjusted R-square |
| RWRSQ | random walk R-square |
| AIC | Akaike information criterion |
| AICC | finite sample corrected AIC |
| SBC | Schwarz Bayesian information criterion |
| APC | Amemiya's prediction criterion |
| MAXERR | maximum error |
| MINERR | minimum error |
| MINPE | minimum percent error |
| MAXPE | maximum percent error |
| ME | mean error |
| MPE | mean percent error |
| MDAPE | median absolute percent error |
| GMAPE | geometric mean absolute percent error |
| MINPPE | minimum predictive percent error |
| MAXPPE | maximum predictive percent error |
| MSPPE | mean predictive percent error |
| MAPPE | symmetric mean absolute predictive percent error |
| MDAPPE | median absolute predictive percent error |
| GMAPPE | geometric mean absolute predictive percent error |
| MINSPE | minimum symmetric percent error |
| MAXSPE | maximum symmetric percent error |
| MSPE | mean symmetric percent error |
| SMAPE | symmetric mean absolute percent error |
| MDASPE | median absolute symmetric percent error |
| GMASPE | geometric mean absolute symmetric percent error |
| MINRE | minimum relative error |
| MAXRE | maximum relative error |
| MRE | mean relative error |
| MRAE | mean relative absolute error |
| MDRAE | median relative absolute error |

| | |
|---|---|
| GMRAE | geometric mean relative absolute error |
| MINAPES | minimum absolute error percent of standard deviation |
| MAXAPES | maximum absolute error percent of standard deviation |
| MAPES | mean absolute error percent of standard deviation |
| MDAPES | median absolute error percent of standard deviation |
| GMAPES | geometric mean absolute error percent of standard deviation |

If the statistics-of-fit step fails for particular variable, a row with missing values is recorded. If a transform is used to affect the final forecast, the values that are listed in this section are computed based on the inverse transform forecasts. For more information about the mathematical details for these fit statistics, see "Statistics of Fit" on page 600.

## OUTSTATSELECT= Data Set

The OUTSTATSELECT= data set contains the same variables as the OUTSTAT= data set in addition to the following:

| | |
|---|---|
| _LABEL_ | system generated label for the candidate model |
| _MODEL_ | candidate model specification name |
| _SELECT_ | name of model selection list to which _MODEL_ belongs |
| _SELECTED_ | indication of whether or not this model was chosen to forecast the dependent series or used by the chosen forecast when the chosen forecast is a combined model. Values set in the _SELECTED_ variable include: |

YES indicates that the associated _MODEL_ is the primary model selected for the forecast.

USED indicates that the associated _MODEL_ is used by the primary model in producing the final forecast.

USED_SELECT indicates that the associated _MODEL_ is used by the primary model in the model selection region but is not used in producing the final forecast.

NO indicates that the associated _MODEL_ is neither selected nor used.

For more information about the mathematical details for these fit statistics, see the section "Statistics of Fit" on page 600.

## OUTSUM= Data Set

The OUTSUM= data set contains the variables that are specified for the lowest level of the hierarchy for the PROC HPFORECAST invocation and the variables listed in this section. The OUTSUM= data set records the summary statistics for each variable that is specified in a FORECAST statement. Variables that are related to summary statistics are based on the ACCUMULATE= and SETMISSING= options.

| | |
|---|---|
| _NAME_ | forecast variable name |
| _STATUS_ | forecasting status. Nonzero values imply that no forecast was generated for the series. |

| | |
|---|---|
| NOBS | number of observations |
| N | number of nonmissing observations |
| NMISS | number of missing observations |
| MIN | minimum value |
| MAX | maximum value |
| MEAN | mean value |
| STDDEV | standard deviation |

The following variables that are related to multistep forecasts are based on the specified LEAD= option value:

_LEAD*n*_ multistep forecast (*n* ranges from one to the value of the LEAD= option).

If the forecast step fails for a particular variable, the variables that are related to forecasts are set to missing. The OUTSUM= data set contains a summary of the (accumulated) time series and optionally its forecasts for all series.

## Printed Output

The HPFORECAST procedure optionally produces printed output to report information that is related to the execution of the procedure. By default, the procedure produces no printed output. All output is controlled by the PRINT= options that are associated with the PROC HPFORECAST statement or by options in the PERFORMANCE statement.

The following sections describe the ODS tables that are produced from printing options.

### PERFORM

PRINT=PERFORM prints the "Performance Information" table. A sample "Performance Information" table follows:

**Performance Information**

| | |
|---|---|
| Host Node | wintergreen.unx.sas.com |
| Data Server | wintergreen |
| Execution Mode | Distributed |
| Grid Mode | Symmetric |
| Number of Compute Nodes | 8 |
| Number of Threads per Node | 8 |

### SUMMARY

The PRINT=SUMMARY option produces ODS tables that summarize the work performed by PROC HPFORECAST. The Summary table summarizes information by process index (rank) that is related to time series forecasting operations in PROC HPFORECAST. When you specify the PRINTDETAILS option, you

also get the TSDESummary table which summarizes information by process index that is related to the time series data engine (TSDE) for PROC HPFORECAST.

The Summary table contains the following columns:

| | |
|---|---|
| Rank | shows the process index (rank), that defines the context for the remainder of these statistics. For distributed modes, ranks 1 through N identify the captain processes; rank N+1 identifies the general (the process that controls the captain processes). For single-machine mode, Rank is set to –1. |
| NumThreads | shows the number of threads that are applied to parallel forecasting. |
| NumLevels | shows the number of hierarchy levels that are processed. |
| NumSeriesRead | shows the number of series that are read from the data source. |
| NumSeriesForecast | shows the number of series that are forecast. This is the total of all levels. |
| NumSeriesFailed | shows the number of series that failed to forecast. This is the total of all levels. |
| DataTime | shows the elapsed time (in seconds) to load the time series. This is the total of all levels. |
| ForecastTime | shows the elapsed time (in seconds) to forecast the time series. This is the total of all levels. |
| OutputTime | shows the elapsed time (in seconds) to perform data set output. This is the total of all levels. |

The TSDESummary table contains the following columns:

| | |
|---|---|
| Rank | shows the process index (rank) that defines the context for the remainder of these statistics. For distributed modes, ranks 1 through N identify the captain processes; rank N+1 identifies the general process. For single-machine mode, Rank is set to –1. |
| NumThreads | shows the number of threads that are used to load the time series data. |
| NumLevels | shows the number of hierarchy levels that are processed. |
| NumMembers | shows the number of level members that are enumerated. This is the total of all levels. |
| ReadTime | shows the elapsed time (in seconds) to read data from the store. |
| GroupTime | shows the elapsed time (in seconds) for level member enumeration and binning of rows for local levels of the hierarchy. |
| AccTime | shows the elapsed time (in seconds) for level member time series accumulation and framing for all local levels of the hierarchy. |
| DistTime | shows the elapsed time (in seconds) for distributed level member enumeration and distributed accumulation. This represents the time for those levels that are not locally complete and require data exchanges with other machines in the grid framework in order to affect a complete time series realization for the level members. |
| MemPerGroup | shows an estimate for the amount of memory that is used per level member (in K-bytes). |

## THREADS

The PRINT=THREADS option produces detailed ODS tables for thread statistics. The ThreadStatistics table contains information about the threads that are used to perform forecasting. When you specify PRINTDETAILS, you also get the TSDEDetails table, which contains information about the time series data engine (TSDE) component that is used to acquire time series from the data source. For more information about how threads are used, see the section "Threading" on page 376. For distributed mode execution, the ThreadStats table contains one row per thread per grid process; for single-machine mode execution, there is just one row per thread.

The ThreadStatistics table contains the following columns:

| | |
|---|---|
| Rank | shows the process index (rank) that defines the context for the remainder of these statistics. For distributed modes, ranks 1 through N identify the captain processes; rank N+1 identifies the general process (the process that controls the captain processes). For single-machine mode, Rank is set to –1. |
| Thread | shows the thread number within the associated process. |
| NumMembers | shows the number of level members that are forecast by this thread. |
| NumRun | shows the number of time series that are processed by this thread. There is one target series per forecast variable per level member. Each thread processes all of the targets for each level member. This number should equal NumMembers times the number of FORECAST variables. |
| NumForecast | shows the number of time series that are successfully forecast by this thread. |
| NumFailed | shows the number of time series that failed to produce a forecast. The NumRun value should equal the sum of NumForecast and NumFailed. |
| NumFaulted | shows the number of unhandled faults in the thread. This number should always be 0. All faults should be handled in the forecasting code if the software is operating properly. |

The TSDEDetails table differentiates processing for time series acquisition by hierarchy level, in addition to the grid process rank index and thread within the process. The counts for each (Rank-Level-Thread) tuple reflect the work done in that specific context. The table contains the following columns:

| | |
|---|---|
| Rank | shows the process (rank) that defines the context for the remainder of these statistics. For distributed modes, ranks 1 through N identify the so-called captain processes; rank N+1 identifies the general. For single-machine mode, Rank is set to –1. |
| Level | shows the leaf variable name for the level of the hierarchy. _TOP_ indicates the top level of the hierarchy that has no associated grouping variables. |
| Thread | shows the thread number within the associated process. |
| NumKeys | shows the number of level members that are enumerated. |
| NumAdd | shows the number of input data rows that are stored. |
| NumAccumulate | shows the number of level members that are accumulated. |
| NumFaulted | shows the number of unhandled faults in the thread. This number should always be 0. All faults should be handled in the forecasting code if the software is operating properly. |

### TIMING

The PRINT=TIMING option in the PROC HPFORECAST statement prints the "Timing" table, which shows a summary of elapsed times to perform the sequential steps that comprise the PROC HPFORECAST invocation. The "Timing" table has the following columns:

| | |
|---|---|
| Task | identifies the step of the processing. |
| Time | shows the elapsed time that is measured for the step (in units of seconds). |
| RelTime | shows the percentage of total time that is represented by the displayed Time value. |

## ODS Table Names

Table 12.4 relates the PRINT= options to ODS tables:

**Table 12.4** ODS Tables Produced in PROC HPFORECAST

| ODS Table Name | Description | Printing Option |
|---|---|---|
| PerformanceInfo | Execution environment information | PRINT=PERFORM |
| Summary | PROC HPFORECAST summary | PRINT=SUMMARY |
| TSDESummary | Data engine summary | PRINT=SUMMARY PRINTDETAILS |
| ThreadStatistics | Forecasting thread statistics | PRINT=THREADS |
| TSDEDetails | Detailed data engine information | PRINT=THREADS PRINTDETAILS |
| Timing | Timing of PROC HPFORECAST steps | PRINT=TIMING |

# Examples: HPFORECAST Procedure

## Example 12.1: Automatic Forecasting of Time Series Data

This example illustrates how you can use the HPFORECAST procedure to automatically forecast time series data. Retail sales data is used for this illustration.

The following DATA step creates a data set from data that are recorded monthly at numerous points of sales. The data set, Sales, contains a variable Date that represents time and a variable for each sales item.

```
data sales;
   format date date9.;
   input date : date9. shoes socks laces dresses coats
         shirts ties belts hats blouses;
datalines;

   ... more lines ...
```

The following statements automatically forecast each of the monthly time series. They automatically select the best fitting model, generate forecasts for every numeric variable in the input data set (Sales) for the next 12 months, and store these forecasts in the output data set (Nextyear).

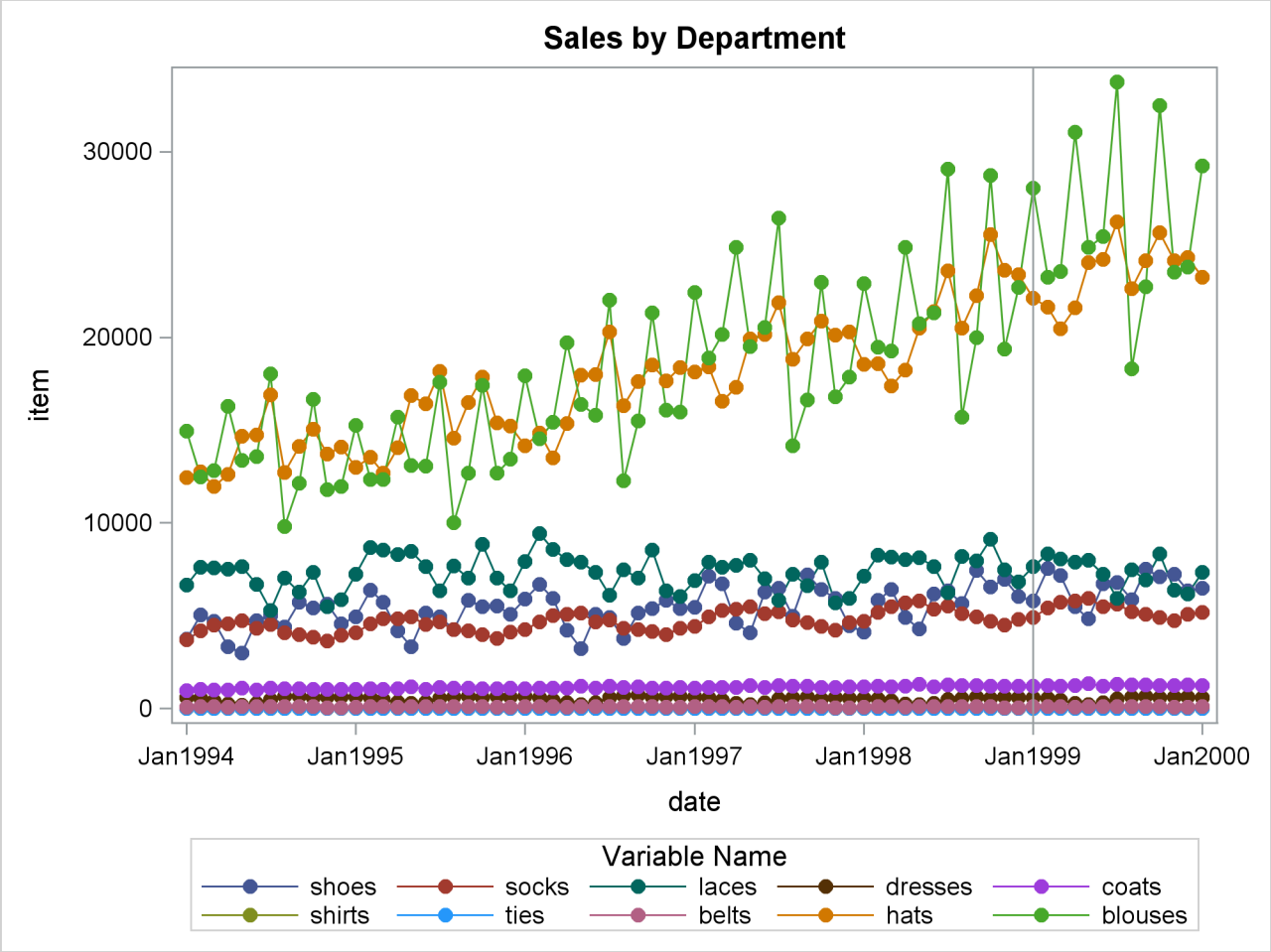
```
proc hpforecast data=sales;
   timeid date interval=month;
   forecast _ALL_;
   output outfor=nextyear
          outlog=errors;
run;
```

The following SGPLOT procedure statements plot the sales forecasts for the various departments on a single graph:

```
title1 "Sales by Department";
proc sgplot data=nextyear;
   series x=date y=predict / group=_name_
      markers markerattrs=(symbol=circlefilled) lineattrs=(pattern=1);
   xaxis values=('01JAN1994'd to '01DEC2000'd by year);
   yaxis label='item';
   refline '01JAN1999'd / axis=x;
run;
```

The SGPLOT procedure results are shown in Output 12.1.1. One-step-ahead forecasts over the period of the historical data are shown to the left of the vertical reference line, and multistep forecasts for the next 12 monthly periods are shown to the right.

**Output 12.1.1** Retail Sales Forecast Plots




## Example 12.2: Automatic Forecasting of Transactional Data

This example illustrates how you can use the HPFORECAST procedure to automatically forecast transactional data. Internet data is used for this illustration.

The data set Websites contains data that are recorded at several Internet websites. The Websites data set contains a variable Time and the variables Boats, Cars, and Planes, which represent Internet website data. Each value of the Time variable is recorded in ascending order, and the values of each of the other variables represent a transactional data series.

The following statements automatically forecast each of the transactional data series. They accumulate the data into a daily time series, automatically generate forecasts for the Boats, Cars, and Planes variables in the input data set (Websites) for the next week, and store the forecasts in the output data set (Nextweek).

```
proc hpforecast data=websites
                lead=7
                criterion=rmse;
   timeid time interval=dtday accumulate=total;
   forecast boats cars planes;
```

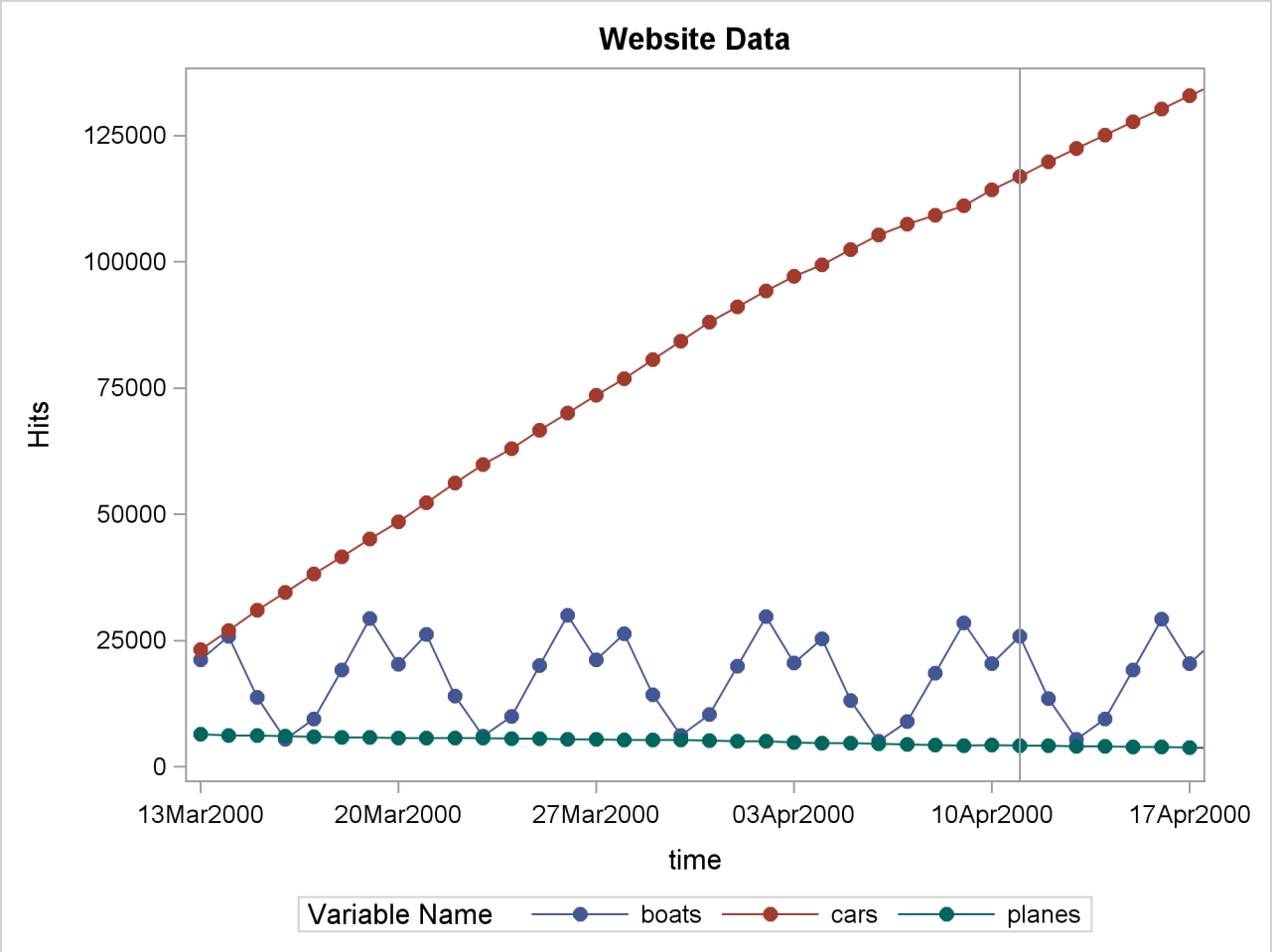
```
   output outfor=nextweek
          outlog=errors;
run;
```

The following SGPLOT procedure statements plot the forecasts that are related to the Internet data:

```
title1 "Website Data";
proc sgplot data=nextweek;
   series x=time y=predict / group=_name_
      markers markerattrs=(symbol=circlefilled) lineattrs=(pattern=1);
   xaxis values=('13MAR2000:00:00:00'dt to '18APR2000:00:00:00'dt by dtweek);
   yaxis label='Hits';
   refline '11APR2000:00:00:00'dt / axis=x;
run;
```

The SGPLOT procedure results are shown in Output 12.2.1. One-step-ahead forecasts over the period of the historical data are shown to the left of the vertical reference line, and multistep forecasts for the next seven days are shown to the right.

**Output 12.2.1** Internet Data Forecast Plots



## Example 12.3: Forecasting Hierarchically Structured Data

In the previous example, the HPFORECAST procedure was used on a data set where each of the analysis variables was present in each row. The FORECAST statement explicitly identified each of those variables to be automatically forecast during the procedure invocation. This example illustrates how you can use the HPFORECAST procedure on the same Internet data when they are stored as a hierarchically structured time series data set.

The Topic variable contains the topic that is associated with variable Hits, which records the number of hits on this topic at each time. The variable Topic takes on the values BOATS, CARS, ENGINES, and PLANES accordingly. The best performing forecast is selected based on the mean absolute percent error (MAPE) as the selection criterion over a holdout sample of five days. Forecasts are produced at the TOPIC level and at the _TOP_ level in a single pass of the data set. The following HPFORECAST procedure statements forecast each of the transactional data series based on these requirements:

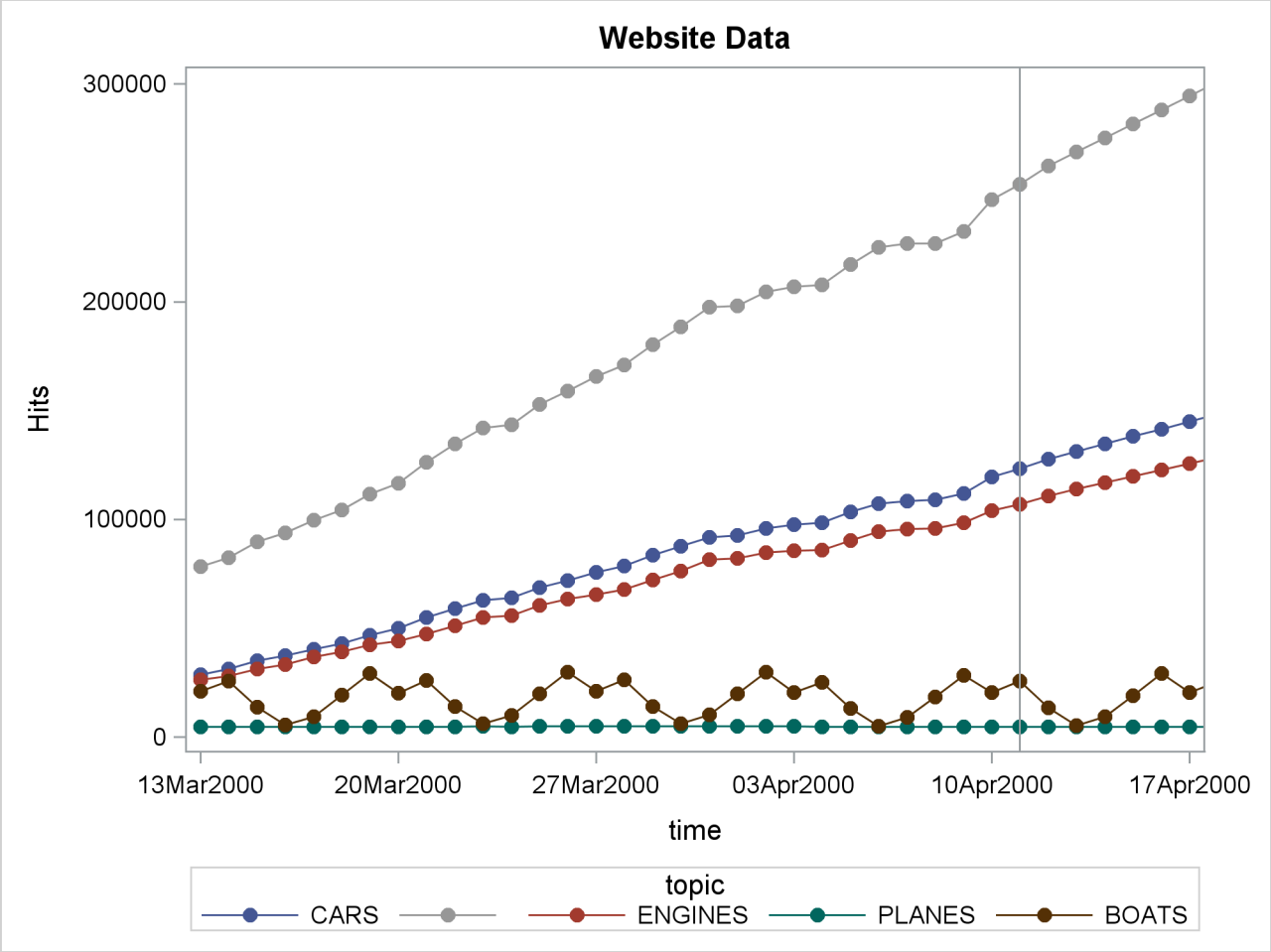
```
proc hpforecast data=websites
                criterion=mape
                holdout=5
                lead=7;
   timeid time interval=dtday accumulate=total;
   forecast hits;
   output outfor=nextweek
          outlog=errors;
   level topic;
   end level;
   level _top_;
   end level;
run;
```

The following SGPLOT procedure statements plot the forecasts that are related to the Internet data:

```
title1 "Website Data";
proc sgplot data=nextweek;
   series x=time y=predict / group=topic
      markers markerattrs=(symbol=circlefilled) lineattrs=(pattern=1);
   xaxis values=('13MAR2000:00:00:00'dt to '18APR2000:00:00:00'dt by dtweek);
   yaxis label='Hits';
   refline '11APR2000:00:00:00'dt / axis=x;
run;
```

The SGPLOT procedure results are shown in Output 12.3.1. One-step-ahead forecasts over the period of the historical data are shown to the left of the vertical reference line, and multistep forecasts for the next seven days are shown to the right. The unlabeled symbol in the legend is _TOP_, which is the forecast for the overall total.

**Output 12.3.1** Internet Data Forecast Plots




## Example 12.4: Forecasting Intermittent Time Series Data

This example illustrates how you can use the HPFORECAST procedure forecast intermittent time series data. Inventory demand is used for this illustration.

The following DATA step creates a data set from inventory data that are recorded at no particular frequency. The data set Inventory contains a variable Date (which represents time) and the demand variables Tires, Hubcaps, and Lugbolts (which represent inventory items). Each value of the Date variable is recorded in ascending order, and the values of each of the other variables represent a transactional data series.

```
data inventory;
   format date date9.;
   input date : date9. tires hubcaps lugbolts;
datalines;
01JUN1997  0  0  0
01JUL1997  0  1  5
01AUG1997  0  0  0

   ... more lines ...
```

The following HPFORECAST procedure statements test each of the transactional data series for intermittency. When a series qualifies, it is forecast by using an intermittent demand model; otherwise, it is forecast by using an exponential smoothing model. The best performing forecast for each series is selected based on the RMSE statistic of fit over the full range of the series.

```
proc hpforecast data=inventory
                lead=4
                intermittent=2.0
                criterion=rmse;
   timeid date interval=week accumulate=total;
   forecast tires hubcaps lugbolts;
   output outfor=nextmonth
          outlog=errors;
run;
```

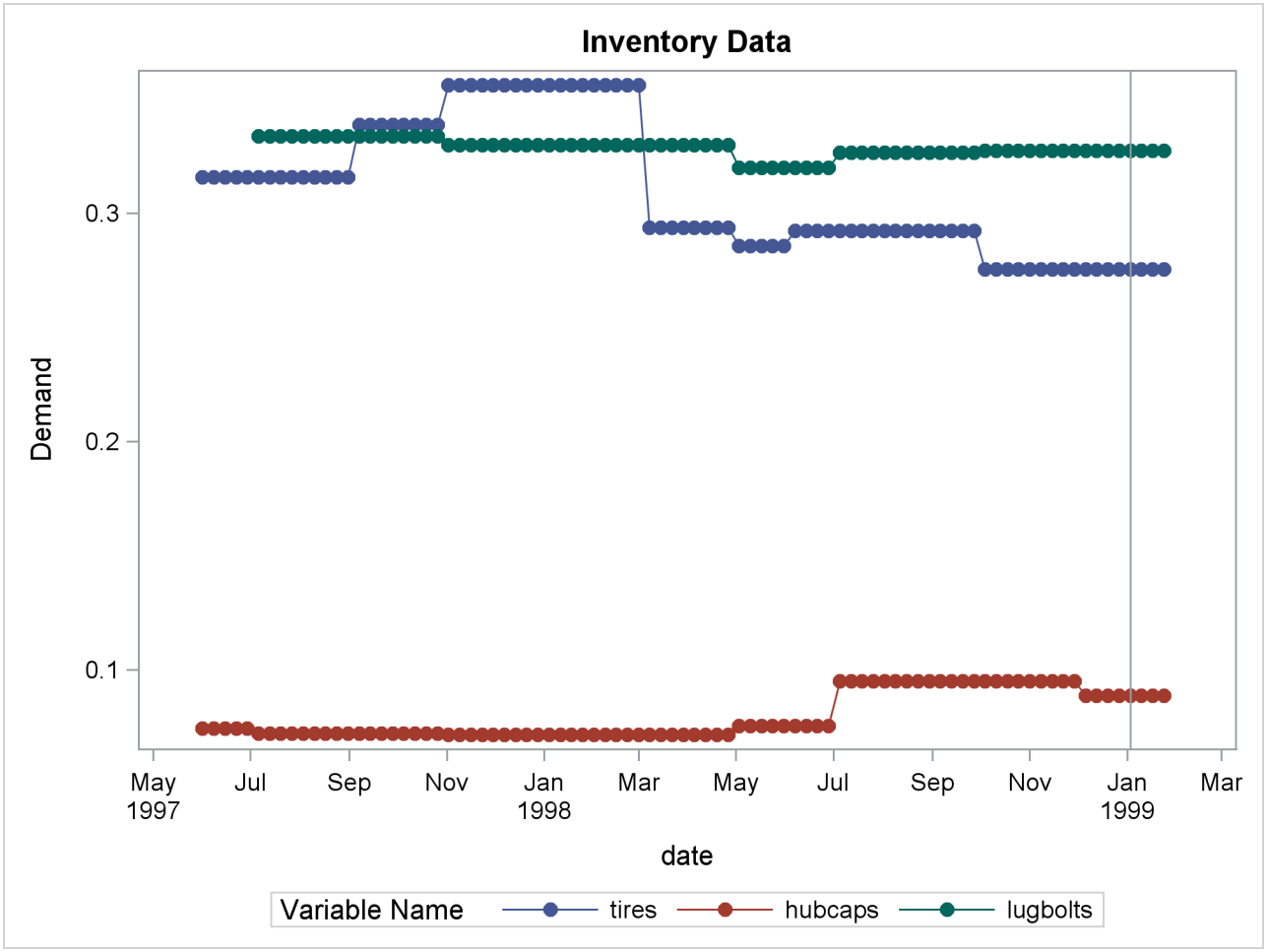
The preceding statements accumulate the data into a weekly time series, generate forecasts for the Tires, Hubcaps, and Lugbolts variables in the input data set (Inventory) for the four weekly periods, and store the forecasts in the output data set (Nextmonth).

The following SGPLOT procedure statements plot the forecasts that are related to the inventory data:

```
title1 "Inventory Data";
proc sgplot data=nextmonth;
   series x=date y=predict / group=_name_
          markers markerattrs=(symbol=circlefilled) lineattrs=(pattern=1);
   yaxis label='Demand';
   refline '03jan1999'd / axis=x;
run;
title1;
```

The SGPLOT procedure results are shown in Output 12.4.1. Forecasts over the period of the historical data are shown to the left of the vertical reference line, and forecasts for the next four weeks are shown to the right. These forecasts represent the average demand per time period. When integrated over the span between demands in the historical period, the forecast stocking level is determined at the time of the next demand and a corresponding inventory error can be computed. The time and quantity of a future demand is unknown, so the forecast is just the projected rate of inventory growth based on the historical demand.

**Output 12.4.1** Inventory Forecast Plots



## Example 12.5: Using the DISTBY Statement in PROC HPFORECAST

This example illustrates how you can use the HPFORECAST procedure to perform distributed mode processing of an input data set that is not stored in a distributed DBMS table.

**NOTE:** Distributed mode requires an additional license.

The following HPFORECAST procedure statements distribute the input data set Sashelp.Pricedata to four nodes of a grid computing appliance that is accessible to the SAS System. PROC HPFORECAST distributes the data by the values of the Productname variable. It generates forecasts for monthly sales (Sale) for three levels of the hierarchy that are defined by the LEVEL statement blocks: the (Productname, Regionname) level, the (Productname) level, and the _TOP_ level. The best performing forecast for each series is selected based on the MAPE fit statistic over a holdout sample of three months.

```
proc hpforecast data=sashelp.pricedata
                lead=4
                criterion=mape
                holdout=3
                print=(summary timing perform) printdetails;
   distby ProductName;
   performance nodes=4;
   timeid date interval=month accumulate=total;
   forecast sale;
   output outfor=salesforecast
          outlog=saleserrors;
   level ProductName RegionName;
   end level;
   level ProductName;
   end level;
   level _top_;
   end level;
run;
```

The preceding statements accumulate the data into a monthly time series for each level member over the three specified levels. For each level member, PROC HPFORECAST generates a forecast for variable Sale for the next four months and stores the forecasts in the output data set Salesforecast.

The following ODS tables are the result of the PRINT= options that are specified in the PROC HPFORECAST statement.

**Output 12.5.1** PRINT=PERFORM

**The HPFORECAST Procedure**

| Performance Information | |
|---|---|
| **Host Node** | greenarrow.unx.sas.com |
| **Data Server** | greenarrow |
| **Execution Mode** | Distributed |
| **Grid Mode** | Symmetric |
| **Number of Compute Nodes** | 4 |
| **Number of Threads per Node** | 24 |

**Output 12.5.2** PRINT=TIMING

| Procedure Task Timing | | |
|---|---|---|
| **Task** | **Seconds** | **Percent** |
| **Parse Statements** | 0.01 | 0.64% |
| **Start Session** | 1.36 | 87.18% |
| **Create Session** | 0.01 | 0.64% |
| **Configure Session** | 0.01 | 0.64% |
| **Prepare to Forecast** | 0.00 | 0.00% |
| **Load and Forecast Time Series** | 0.06 | 3.85% |
| **Output Data Sets** | 0.04 | 2.56% |
| **Output ODS** | 0.07 | 4.49% |

**Output 12.5.3** PRINT=SUMMARY

| Processing Summary | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Rank** | **Threads Used** | **Level Count** | **Series Count** | **Series Forecast** | **Failed Forecasts** | **Data Load Time (seconds)** | **Forecast Time (seconds)** | **Output Time (seconds)** |
| **1** | 3 | 3 | 11 | 10 | 0 | 0.0049 | 0.0340 | 0.0333 |
| **2** | 3 | 3 | 9 | 9 | 0 | 0.0045 | 0.0320 | 0.0317 |
| **3** | 3 | 3 | 9 | 8 | 0 | 0.0044 | 0.0174 | 0.0315 |
| **4** | 3 | 3 | 9 | 8 | 0 | 0.0049 | 0.0205 | 0.0316 |
| **5** | 10 | 0 | 0 | 0 | 0 | 0.0049 | 0 | 0.0365 |

**Output 12.5.4** PRINT=SUMMARY PRINTDETAILS

**Data Summary**

| Rank | Threads Used | Hierarchy Levels | Member Count | Read Time (seconds) | Group Time (seconds) | Accumulation Time (seconds) | Distributed Level Time (seconds) | Memory Per Group (Kb) |
|---|---|---|---|---|---|---|---|---|
| **1** | 3 | 3 | 11 | 0.000031 | 0.000729 | 0.0017 | 0.0019 | 1.0 |
| **2** | 3 | 3 | 9 | 0.000025 | 0.000481 | 0.0011 | 0.0024 | 1.0 |
| **3** | 3 | 3 | 9 | 0.000032 | 0.000511 | 0.0010 | 0.0023 | 1.0 |
| **4** | 3 | 3 | 9 | 0.000035 | 0.000521 | 0.0010 | 0.0028 | 1.0 |
| **5** | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0.0 |

## Example 12.6: Using PROC HPFORECAST with the Greenplum DBMS

This example repeats the same processing as the previous example, but now the input data are accessed from a Greenplum DBMS table. In this case, the HPFORECAST procedure automatically runs in distributed mode (symmetric) alongside the DBMS table segments.

**NOTE:** Distributed mode requires an additional license.

Before you run PROC HPFORECAST, you must load the table into the Greenplum appliance with a data partitioning that can support the specified LEVEL statements. Each distributed DBMS has a different way to perform this loading, and there are also different ways to approach this within the SAS System. The following sample code demonstrates one of the ways to load the table to support this example. This code uses special data set options that are supported by the Greenplum SAS/ACCESS engine and the Greenplum bulk loader *gpfdist*.

```
libname gplib greenplm
        server="hpa.sas.com"
        schema  =public
        user    =gpadmin
        password=XXXXXXX
        database=hps;

proc sql;
   drop table gplib.pricedata;
   drop table gplib.rejected;
quit;

proc sql;
   create table gplib.pricedata(
             dbtype            =(productName='char(9)')
             distributed_by    ='distributed by (productName)'
             bulkload          =yes
             bl_format         ='TEXT'
             bl_null           ='\n'
             bl_host           ='localhost'
             bl_port           ='8081'
             bl_protocol       ='gpfdist'
             bl_delete_datafile=NO
             bl_exception      ="rejected"
             bl_reject_limit   =100000)
```

```
   as select * from sashelp.pricedata;
run;

proc sql;
   connect to greenplm as gp(server="hpa.sas.com" user="gpadmin" password="XXXXXXX"
                          database="hps");
   execute (
       grant all privileges on public.pricedata TO sas_hpa_group;
   ) by gp;

   disconnect from gp;
quit;
```

The first PROC SQL step deletes any existing tables that are affected by the load from the Greenplum database. The second PROC SQL step loads the data set Sashelp.Pricedata into the Greenplum appliance so that its rows are distributed by the distinct values of the variable Productname. This allows any hierarchy that begins with the Productname variable to be processed from this single table load. The last PROC SQL step uses explicit pass-through SQL to grant access privileges to all members of **sas_hpa_group**.

The following SAS code generates the same forecasts as the previous example. It generates forecasts for monthly sales (Sale) for three levels of the hierarchy that are defined by the LEVEL statement blocks: the (Productname, Regionname) level, the (Productname) level, and the _TOP_ level. The best performing forecast for each series is selected based on the MAPE fit statistic over a holdout sample of three months. The GPLIB libref that is used in the example code is assumed to be assigned via the Greenplum SAS/ACCESS engine to access the **public** schema of the **hps** database.

```
proc hpforecast data=gplib.pricedata
                lead=4
                criterion=mape
                holdout=3
                print=(summary timing perform);
   timeid date interval=month accumulate=total;
   forecast sale;
   output outfor=gplib.salesforecast
          outlog=gplib.saleserrors;
   level ProductName RegionName;
   end level;
   level ProductName;
   end level;
   level _top_;
   end level;
run;
```

The preceding statements accumulate the data into a monthly time series for each level member over the three specified levels. For each level member, it generates a forecast for the variable Sale for the next four months and stores the forecasts in the Greenplum table Salesforecast.

The following ODS tables are the result of the PRINT= options that are specified in the PROC HPFORECAST statement. The PRINT=SUMMARY output indicates that some of the Greenplum DCA nodes did not receive any rows of the distributed table. Note also the start-up time that is required to launch the grid processes on the Greenplum DCA machines in the Timing table that is generated from the PRINT=TIMING option. This start-up time far exceeds the computational gain for small problems such as this example.

**Output 12.6.1** PRINT=PERFORM

**The HPFORECAST Procedure**

| Performance Information | |
|---|---|
| **Host Node** | greenarrow.unx.sas.com |
| **Data Server** | greenarrow |
| **Execution Mode** | Distributed |
| **Grid Mode** | Symmetric |
| **Number of Compute Nodes** | 16 |
| **Number of Threads per Node** | 24 |

**Output 12.6.2** PRINT=TIMING

| Procedure Task Timing | | |
|---|---|---|
| **Task** | **Seconds** | **Percent** |
| **Parse Statements** | 0.05 | 0.35% |
| **Start Session** | 3.19 | 22.08% |
| **Create Session** | 0.02 | 0.14% |
| **Configure Session** | 3.21 | 22.21% |
| **Prepare to Forecast** | 0.34 | 2.35% |
| **Load and Forecast Time Series** | 2.16 | 14.95% |
| **Output Data Sets** | 5.30 | 36.68% |
| **Output ODS** | 0.18 | 1.25% |

**Output 12.6.3** PRINT=SUMMARY

| Processing Summary | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Rank** | **Threads Used** | **Level Count** | **Series Count** | **Series Forecast** | **Failed Forecasts** | **Data Load Time (seconds)** | **Forecast Time (seconds)** | **Output Time (seconds)** |
| **1** | 16 | 2 | 0 | 0 | 0 | 2.0426 | 0.000017 | 5.2918 |
| **2** | 16 | 2 | 0 | 0 | 0 | 2.0426 | 0.000022 | 5.2918 |
| **3** | 16 | 2 | 0 | 0 | 0 | 2.0425 | 0.000021 | 5.2916 |
| **4** | 16 | 2 | 0 | 0 | 0 | 2.0428 | 0.000017 | 5.2918 |
| **5** | 16 | 2 | 0 | 0 | 0 | 2.0428 | 0.000021 | 5.2918 |
| **6** | 16 | 2 | 0 | 0 | 0 | 2.0428 | 0.000021 | 5.2919 |
| **7** | 16 | 2 | 0 | 0 | 0 | 2.0424 | 0.000018 | 5.2916 |
| **8** | 16 | 2 | 0 | 0 | 0 | 2.0423 | 0.000018 | 5.2920 |
| **9** | 16 | 2 | 0 | 0 | 0 | 2.0422 | 0.000017 | 5.2919 |
| **10** | 16 | 3 | 13 | 12 | 0 | 2.0424 | 0.0188 | 5.2919 |
| **11** | 16 | 3 | 7 | 6 | 0 | 2.0423 | 0.0165 | 5.2917 |
| **12** | 16 | 2 | 0 | 0 | 0 | 2.0424 | 0.000017 | 5.2918 |
| **13** | 16 | 2 | 0 | 0 | 0 | 2.0426 | 0.000016 | 5.2917 |
| **14** | 16 | 2 | 0 | 0 | 0 | 2.0427 | 0.000017 | 5.2918 |
| **15** | 16 | 3 | 5 | 5 | 0 | 2.0426 | 0.0299 | 5.2917 |
| **16** | 16 | 3 | 13 | 12 | 0 | 2.0426 | 0.0206 | 5.2920 |
| **17** | 16 | 0 | 0 | 0 | 0 | 2.0425 | 0 | 5.3057 |

## Example 12.7: Automatic Modeling with Greenplum

This example repeats the same processing as the previous example, but now MODE=AUTO is used to expand the scope of the time series modeling and forecasting capabilities beyond exponential smoothing models. This expansion shows how you can take advantage of full automation for model identification, model selection, and forecasting to run in a distributed setting alongside the DBMS table segments. MODE=AUTO is not restricted to use in distributed mode; this example simply demonstrates its use in that setting.

**NOTE:** Distributed mode requires an additional license.

Without any other option settings, a single step of PROC HPFORECAST MODE=AUTO performs the same model identification, model selection, and final forecast generation as if you repeated the following PROC HPFDIAGNOSE and PROC HPFENGINE steps for each level of the hierarchy:

```
proc hpfdiagnose ...
     rep=DiagRep
     outest=DiagEst ...;
   ...;
   arimax identifyorder=both outlier=(detect=no);
   esm;
run;

proc hpfengine ...
     rep=DiagRep
     inest=DiagEst ...;
   ...;
run;
```

The following code generates forecasts for monthly sales (Sale) for three levels of the hierarchy that are defined by the LEVEL statement blocks: the (Productname, Regionname) level, the (Productname) level, and the _TOP_ level. For each series for the Sale variable, PROC HPFORECAST performs automatic time series model generation for the model families of ESM and ARIMA by using the default (built-in) diagnostic control preferences. For ARIMA model identification, the INPUT statement requests that the predictor variables Price and Discount be considered for inclusion in the generated models if they prove to be significant in improving forecastability of sales. PROC HPFORECAST performs model selection for the generated candidate models by producing a forecast for the Sale and selecting the best performing model based on the MAPE fit statistic over a holdout sample of three months. The GPLIB libref that is used in the example code is assumed to be assigned via the Greenplum SAS/ACCESS engine to access the **public** schema of the **hps** database.

```
proc hpforecast data=gplib.pricedata
                mode=auto
                lead=4
                criterion=mape
                holdout=3
                override;
   timeid date interval=month accumulate=total;
   forecast sale;
   input price discount;
   output outfor=gplib.autoforecast
          outstatselect=gplib.autoselect
          outstat=gplib.autostat
```

```
          outmodelinfo=gplib.autoinfo
          outlog=gplib.autoerrors;
   level ProductName RegionName;
   end level;
   level ProductName;
   end level;
   level _top_;
   end level;
run;
```

The preceding code accumulates the data into a monthly time series for each level member over the three specified levels. For each level member, it generates a forecast for the variable Sale for the next four months and stores the forecasts in the Greenplum table Autoforecast. Fit statistics for the candidate models that are considered for each series are stored in the Greenplum table Autoselect, and fit statistics for the final forecast for each series are stored in the Greenplum table Autostat.

Output 12.7.1 shows a summary of model selection fit statistics by model type and predictor usage for the (ProductName, RegionName) level. You can see from this summary that all the generated ARIMA models include input (predictor) variables for this level of the hierarchy.

**Output 12.7.1** MAPE Selection Statistics for (ProductName,RegionName) Level

**The MEANS Procedure**

| Analysis Variable : MAPE MAPE | | | | | |
|---|---|---|---|---|---|
| _MODELTYPE_ | _INPUTS_ | N Obs | Mean | Maximum | Minimum |
| ARIMA | 1 | 11 | 5.2183971 | 16.8734015 | 0.3188330 |
| ESM | 0 | 6 | 3.1414346 | 5.2095116 | 1.0948547 |

Output 12.7.2 shows a summary of model selection fit statistics by model type and predictor usage for the ProductName level. You can see from this summary that all the generated ARIMA models include input (predictor) variables for this level of the hierarchy.

**Output 12.7.2** MAPE Selection Statistics for ProductName Level

**The MEANS Procedure**

| Analysis Variable : MAPE MAPE | | | | | |
|---|---|---|---|---|---|
| _MODELTYPE_ | _INPUTS_ | N Obs | Mean | Maximum | Minimum |
| ARIMA | 1 | 11 | 5.2183971 | 16.8734015 | 0.3188330 |
| ESM | 0 | 6 | 3.1414346 | 5.2095116 | 1.0948547 |

## Example 12.8: Using Automatic Modeling and User-Defined Models

This example demonstrates how you can blend user-defined models, model selection lists, and user-defined diagnostic control preferences into a single execution of PROC HPFORECAST.

The following SAS code uses PROC HPFDIAGSPEC to define a diagnostic specification (DSPEC0) that specifies options that control input testing and transformation testing and requests that only ESM models be

diagnosed. The CRITERION= option specifies that symmetric mean absolute percent error (SMAPE) be used for model selection.

```
proc hpfdiagspec rep=work.drep name=dspec0;
   diagnose siglevel=0.10
            testinput=both
            criterion=smape;
   transform type=auto;
   esm method=best;
run;
```

The following SAS code defines a diagnostic specification (DSPEC1) that specifies options that control testing of inputs. It also imposes limits on the number of inputs that can be included, requests that transformation testing be performed, requests that ESM and ARIMA models be diagnosed, and requests that mean absolute percent error (MAPE) be used for model selection.

```
proc hpfdiagspec rep=work.drep name=dspec1;
   diagnose siglevel=0.10
            testinput=both
            selectinput=3
            criterion=mape;
   transform type=auto;
   esm method=best;
   arimax identifyorder=both;
run;
```

The following SAS code defines a custom UCM model (MYUCM) and a custom ARIMA model (MYARIMA), and then defines a model combination (MYCOMB) of MYUCM and MYARIMA that uses RMSE weights for the combination method:

```
proc hpfucmspec rep=work.drep name=myucm;
   forecast symbol=y transform=log;
   irregular;
   level;
   slope;
   input symbol=price;
run;

proc hpfarimaspec rep=work.drep name=myarima;
   forecast var=y p=1 q=2 dif=(1) noint;
   input var=price;
run;

proc hpfselect rep=work.drep name=mycomb;
   combine method=rmsewgt;
   spec myucm myarima;
run;
```

The following SAS code defines two model selection lists: MYMODELS0, which is defined to select between models MYUCM and MYARIMA, and MYMODELS1, which is defined to select between the model combination MYCOMB and the models MYUCM and MYARIMA.

```
proc hpfselect rep=work.drep name=mymodels0;
   spec myucm myarima;
run;

proc hpfselect rep=work.drep name=mymodels1;
   spec mycomb myucm myarima;
run;
```

The following SAS code demonstrates how you can use these specifications in PROC HPFORECAST. The FORECAST statement for the variable Sale specifies diagnostic control preferences in DSPEC1 and the user-defined models in MYMODELS1. The DIAGNOSE and SELECTION properties of the variable Sale (with values DSPEC1 and MYMODELS1, respectively) are valid in all level contexts unless they are overridden by a SET statement in level scope. In this example, the DIAGNOSE and SELECTION properties for the FORECAST variable Sale are overridden in the (Region, Product) level by DSPEC0 and MYSPEC0, respectively.

```
proc hpforecast data=sashelp.pricedata
                mode=auto
                rep=work.drep
                lead=12;
   timeid date interval=month accumulate=total;
   output outstatselect=amoss
          outstat=amstat
          outfor=amfor
          outest=amest
          outlog=amlog;
   forecast sale / diagnose=dspec1 selection=mymodels1;
   input price / accumulate=average extend=stochastic;
   input discount / accumulate=average extend=stochastic;
   level region product;
      set sale.diagnose=spec(dspec0);
      set sale.selection=spec(mymodels0);
   end level;
   level _top_;
   end level;
run;
```

Suppose you want to forecast only with the user-defined models on this data set (omitting the automatic model generation). The following code demonstrates how you can use PROC HPFORECAST to generate forecasts that consider only the user-defined models. The following code differs from the previous SAS code in the addition of the NODIAGNOSE option in the PROC HPFORECAST statement, the omission of the DIAGNOSE= option in the FORECAST statement, and the omission of the SET statement for the DIAGNOSE property on the variable Sale. With these changes, the following PROC HPFORECAST invocation does not perform automatic model diagnosis for the time series in the specified levels. It evaluates the user-defined models that are contained in the MYMODELS0 list for the (Region, Product) level and the user-defined models that are contained in the MYMODELS1 list for all other levels (_TOP_ in this case).

```
proc hpforecast data=sashelp.pricedata
                mode=auto
                rep=work.drep
                lead=12
                nodiagnose;
```

```
   timeid date interval=month accumulate=total;
   output outstatselect=amoss
          outstat=amstat
          outfor=amfor
          outest=amest
          outlog=amlog;
   forecast sale / selection=mymodels1;
   input price / accumulate=average extend=stochastic;
   input discount / accumulate=average extend=stochastic;
   level region product;
      set sale.selection=spec(mymodels0);
   end level;
   level _top_;
   end level;
run;
```

## References

Pyle, D. (1999), *Data Preparation for Data Mining*, San Francisco: Morgan Kaufmann.

# Chapter 13
# The HPFRECONCILE Procedure

**Contents**


Overview: HPFRECONCILE Procedure . . . . . **417**
Getting Started: HPFRECONCILE Procedure . . . . . **418**
Syntax: HPFRECONCILE Procedure . . . . . **419**
Functional Summary . . . . . 420
PROC HPFRECONCILE Statement . . . . . 422
AGGBY Statement . . . . . 425
AGGDATA Statement . . . . . 426
BY Statement . . . . . 426
DISAGGDATA Statement . . . . . 427
ID Statement . . . . . 427
Details: HPFRECONCILE Procedure . . . . . **428**
Notation . . . . . 428
Top-Down Reconciliation . . . . . 429
Bottom-Up Reconciliation . . . . . 432
Data Set Input/Output . . . . . 434
Examples: HPFRECONCILE Procedure . . . . . **441**
Example 13.1: Reconciling a Hierarchical Tree . . . . . 441
Example 13.2: Aggregating Forecasts . . . . . 444
Example 13.3: Disaggregating Forecasts . . . . . 444
Example 13.4: Imposing Constraints . . . . . 446


## Overview: HPFRECONCILE Procedure

When the data are organized in a hierarchical fashion, there are often accounting constraints that link series at different levels of the hierarchy. For example, the total sales of a product by a retail company are the sum of the sales of the same product in all stores that belong to the company. It seems natural to require that the same constraints hold for the predicted values as well. Imposing such constraints during the forecasting process can be difficult or impossible. Therefore, the series are often forecast independently at different levels. The resulting forecasts, however, do not abide by the constraints that bind the original series. The after-the-fact process through which such constraints are enforced is called *reconciliation*.

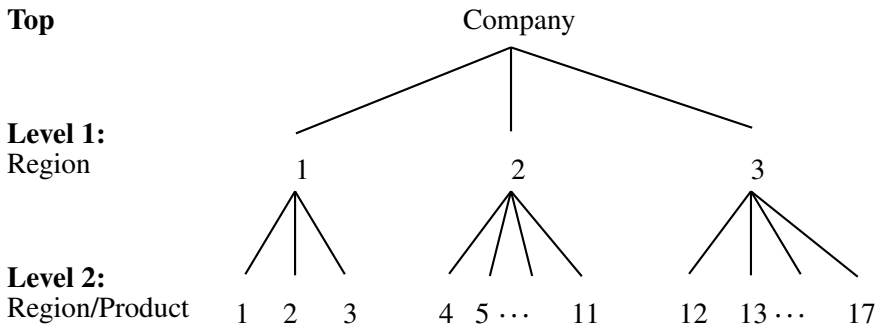
The HPFRECONCILE procedure reconciles forecasts of time series data at two different levels of a hierarchy. Optionally, the HPFRECONCILE procedure can disaggregate forecasts from upper-level forecasts or aggregate forecasts from lower-level forecasts. The procedure enables the user to specify the direction and the method of reconciliation, equality constraints, and bounds on the reconciled values.

# Getting Started: HPFRECONCILE Procedure

This section outlines the use of the HPFRECONCILE procedure.

Consider the following hierarchical structure of the SASHELP.PRICEDATA data set.

**Figure 13.1** Hierarchical Structure of SASHELP.PRICEDATA




Forecasts for the dependent variable sale are generated first at level 2, region / product , and then at level 1, region. The separate forecasts are then reconciled in a bottom-up manner by using the HPFRECONCILE procedure as shown in the following statements:

```
/* Forecast series at level 2 (region/product) */

* Step 1: model selection;
proc hpfdiagnose data=sashelp.pricedata
                 outest=lvl2est
                 modelrepository=work.mymodels
                 prefilter=both
                 criterion=mape;
   id date interval=month;
   by region product;
   forecast sale;
   input price;
run;

* Step 2: estimation and forecasting ;
proc hpfengine data=sashelp.pricedata
               inest=lvl2est
               out=_null_
               outest=lvl2fest
               modelrepository=mymodels
               outfor=lvl2for;
   id date interval=month;
   by region product;
   forecast sale / task=select ;
   stochastic price;
run;

* Forecast aggregated series at level 1 (region);
```

```
* Step 1: model selection;
proc hpfdiagnose data=sashelp.pricedata
                 outest=lvl1est
                 modelrepository=work.mymodels
                 prefilter=both
                 criterion=mape;
   id date interval=month notsorted;
   by region;
   forecast sale / accumulate=total;
   input price / accumulate=average;
run;

* Step 2: estimation and forecasting;
proc hpfengine data=sashelp.pricedata
               inest=lvl1est
               out=_null_
               outest=lvl1fest
               modelrepository=mymodels
               outfor=lvl1for;
   id date interval=month notsorted;
   by region;
   forecast sale / task=select accumulate=total;
   stochastic price /accumulate=average;
run;

* Reconcile forecasts bottom up with default settings;
proc hpfreconcile disaggdata=lvl2for
                  aggdata=lvl1for
                  direction=BU
                  outfor=lvl1recfor;
   id date interval=month;
   by region product;
run;
```

# Syntax: HPFRECONCILE Procedure

The HPFRECONCILE procedure is controlled by the following statements:

**PROC HPFRECONCILE** < *options* > ;
**AGGBY** *variables* < / *option* > ;
**AGGDATA** < *options* > ;
**BY** *variables* < / *option* > ;
**DISAGGDATA** < *options* > ;
**ID** *variable INTERVAL=interval* < / *options* > ;

## Functional Summary

The statements and options used with the HPFRECONCILE procedure are summarized in Table 13.1.

**Table 13.1** HPFRECONCILE Functional Summary

| Description | Statement | Option |
|---|---|---|
| **Statements** | | |
| specify the AGGBY variables | AGGBY | |
| name variables in the AGGDATA= data set | AGGDATA | |
| specify the BY variables | BY | |
| name variables in the DISAGGDATA= data set | DISAGGDATA | |
| specify the time ID variable | ID | |
| **Time ID Options** | | |
| specify the alignment of time ID values | ID | ALIGN= |
| specify the ending time ID value | ID | END= |
| specify the date format | ID | FORMAT= |
| specify the frequency | ID | INTERVAL= |
| specify whether to allow for irregularities in the ID variable frequency | ID | IRREGULAR= |
| specify the starting time ID value | ID | START= |
| **Data Set Options** | | |
| specify that the DISAGGDATA= data set is sorted by the BY variables | DISAGGDATA | BYVARSSORTED |
| specify the disaggregated input data set (child level in the hierarchy) | HPFRECONCILE | DISAGGDATA= |
| specify the aggregated input data set (parent level in the hierarchy) | HPFRECONCILE | AGGDATA= |
| specify the output data set that contains the reconciled forecasts | HPFRECONCILE | OUTFOR= |
| specify the output data set that contains information regarding the infeasible problems | HPFRECONCILE | OUTINFEASIBLE= |
| specify the output data set that contains summary information | HPFRECONCILE | OUTPROCINFO= |
| specify the data set that contains constraints on the reconciled forecasts | HPFRECONCILE | CONSTRAINT= |
| specify that constraints be forced on the *predict* variable in the OUTFOR= data set when it conflicts with the aggregation constraint | HPFRECONCILE | FORCECONSTRAINT |
| specify that the OUTFOR= data set contains the RECDIFF variable | HPFRECONCILE | RECDIFF |
| name the variable that contains the actual values in the DISAGGDATA= data set | DISAGGDATA | ACTUAL= |
| name the variable that contains the actual values in the AGGDATA= data set | AGGDATA | ACTUAL= |

| Description | Statement | Option |
|---|---|---|
| name the variable that contains the predicted values in the DISAGGDATA= data set | DISAGGDATA | PREDICT= |
| name the variable that contains the predicted values in the AGGDATA= data set | AGGDATA | PREDICT= |
| name the variable that contains the lower confidence limit in the DISAGGDATA= data set | DISAGGDATA | LOWER= |
| name the variable that contains the lower confidence limit in the AGGDATA= data set | AGGDATA | LOWER= |
| name the variable that contains the upper confidence limit in the DISAGGDATA= data set | DISAGGDATA | UPPER= |
| name the variable that contains the upper confidence limit in the AGGDATA= data set | AGGDATA | UPPER= |
| name the variable that contains the prediction error in the DISAGGDATA= data set | DISAGGDATA | ERROR= |
| name the variable that contains the prediction error in the AGGDATA= data set | AGGDATA | ERROR= |
| name the variable that contains the standard error in the DISAGGDATA= data set | DISAGGDATA | STD= |
| name the variable that contains the standard error in the AGGDATA= data set | AGGDATA | STD= |
| **Error Message Options** | | |
| specify the resolution of error and warning messages | HPFRECONCILE | ERRORTRACE= |
| **Analysis Options** | | |
| specify the aggregation method | HPFRECONCILE | AGGREGATE= |
| specify the confidence level | HPFRECONCILE | ALPHA= |
| specify method for confidence limits | HPFRECONCILE | CLMETHOD= |
| specify the reconciliation direction | HPFRECONCILE | DIRECTION= |
| specify the disaggregation function | HPFRECONCILE | DISAGGREGATION= |
| specify that `'.F'` missing values in PREDICT be treated as regular `'.'` missing values. | HPFRECONCILE | IGNOREMISSF |
| specify that zeros forecasts be left unchanged | HPFRECONCILE | LOCKZERO |
| specify the maximum number of iteration of the optimizer | HPFRECONCILE | MAXITER |
| specify that only the prediction be reconciled | HPFRECONCILE | PREDICTONLY |
| specify sign constraint on the reconciled series | HPFRECONCILE | SIGN= |
| specify the method of computing standard errors | HPFRECONCILE | STDMETHOD= |
| specify bounds for the standard error | HPFRECONCILE | STDDIFBD= |
| specify that the loss function be weighted by the inverse of the prediction variances | HPFRECONCILE | WEIGHTED |

## PROC HPFRECONCILE Statement

**PROC HPFRECONCILE** *options* **;**

The following options can be used in the PROC HPFRECONCILE statement.

### Options Related to the Input Data Sets

**AGGDATA=***SAS-data-set*

specifies the name of the SAS data set that contains the forecasts of the aggregated time series data. Typically, the AGGDATA= data set is generated by the OUTFOR= statement of the HPFENGINE procedure. If the AGGDATA= data set is not specified, only bottom-up reconciliation is allowed.

The AGGDATA= data set must contain a proper subset, possibly empty, of the BY variables present in the DISAGGDATA= data set. Such BY variables are called AGGBY variables. See the section "BY Statement" on page 426 for more details about BY and AGGBY variables.

See the section "AGGDATA= Data Set" on page 434 for more details about the AGGDATA= data set.

**CONSTRAINT=***SAS-data-set*

specifies the name of the SAS data set that contains the constraints for the reconciled series. See the section "CONSTRAINT= Data Set" on page 435 for more details.

**DISAGGDATA | DATA=***SAS-data-set*

specifies the name of the SAS data set that contains the forecast of the disaggregated time series data. Typically, the DISAGGDATA= data set is generated by the OUTFOR= statement of the HPFENGINE procedure.

If the DISAGGDATA= data set is not specified, the data set last opened is used. The dimensions of the DISAGGDATA= data set are greater than the dimensions of the AGGDATA= data set.

See the section "DISAGGDATA= Data Set" on page 435 for more details.

### Options Related to the Output Data Sets

**FORCECONSTRAINT**

specifies whether the user-specified constraints should be forced on the PREDICT variable in the OUTFOR= data set when the problem is infeasible because the constraints are incompatible with the aggregation constraint. The default is to leave the input unmodified.

**OUTFOR=***SAS-data-set*

specifies the name of the output SAS data set that contains the reconciled values.

**OUTINFEASIBLE=***SAS-data-set*

specifies the name of the SAS data set that contains a summary of the nodes for which reconciliation failed because of a conflict between the constraints imposed by the user and the aggregation constraint.

**OUTPROCINFO=***SAS-data-set*

names the output data set to contain the summary information of the processing done by PROC HPFRECONCILE. When you write a program to assess the status of the procedure's execution, it is easier to parse a data set than it is to look at or parse the SAS log.

**RECDIFF**
If the RECDIFF option is specified, the OUTFOR= data sets will contain a variable named RECDIFF that is the difference between the reconciled forecasts and the original forecasts.

## Options Related to Error Messages

**ERRORTRACE=***option*
specifies how often the error and warning messages should be printed to the log.

The following values are allowed:

| | |
|---|---|
| DATASET | Messages are printed only one time at the end of the procedure run. |
| AGGBY | Messages are printed for each AGGBY group. |
| ID | Messages are printed for each ID value. |

The default is ERRORTRACE=DATASET.

## Options Related to the Analysis

**AGGREGATE=TOTAL | AVERAGE**
specifies whether the dependent variable in the AGGDATA= data set is the total sum or the average over the BY groups of the dependent variable in the DISAGGDATA= data set. The default is AGGREGATE=TOTAL.

**ALPHA=***n*
specifies the level of the confidence limits when CLMETHOD=GAUSSIAN. The ALPHA= value must be between 0.0001 and 0.9999. When you specify ALPHA=$\alpha$, the upper and lower confidence limits will have a $1-\alpha$ confidence level. The default is ALPHA=0.05, which produces 95% confidence intervals.

**CLMETHOD=***option*
specifies the method used to compute confidence limits for the reconciled forecasts.

The following methods are provided:

| | |
|---|---|
| GAUSSIAN | The confidence intervals are computed by assuming that the forecasts are approximately Gaussian. |
| SHIFT | The confidence intervals are computed by re-centering the original confidence intervals around the new forecasts. |

The default value is CLMETHOD=SHIFT. See the section "Details: HPFRECONCILE Procedure" on page 428 for more information about the methods of computing confidence intervals.

**DIRECTION=** *reconciliation-direction*
specifies the reconciliation direction.

The following reconciliation values are allowed:

| | |
|---|---|
| BU | bottom-up reconciliation |
| TD | top-down reconciliation |

If the AGGDATA= data set is not specified, only DIRECTION=BU is allowed.

The default value is DIRECTION=BU.

See the section "Details: HPFRECONCILE Procedure" on page 428 for more information about the reconciliation directions available in PROC HPFRECONCILE.

**DISAGGREGATION=DIFFERENCE | PROPORTIONS**

specifies the type of loss function for top-down reconciliation.

DISAGGREGATION=PROPORTIONS is available only when all the forecasts at a given ID value share the same sign. See the section "Details: HPFRECONCILE Procedure" on page 428 for more information about the expressions of the loss function.

The default value is DISAGGREGATION=DIFFERENCE.

**IGNOREMISSF**

specifies that `'.F'` missing values in the PREDICT variable be treated as regular `'.'` missing values. If the IGNOREMISSF option is not specified, a `'.F'` missing value is interpreted as a failed forecast, and PROC HPFRECONCILE generates `'.F'` missing values for all forecasting variables in the OUTFOR= data set if that value is needed for computing the reconciled forecasts. If the IGNOREMISSF option is specified, observations that correspond to `'.F'` missing values are considered to belong to inactive series and therefore are not included in the reconciliation process.

**LOCKZERO**

specifies that zero values of the PREDICT variable in the DISAGGDATA= data be considered locked equalities. This option is available only when DIRECTION=TD. When the LOCKZERO option is active, a zero value for PREDICT in the DISAGGDATA= set implies a zero value for the corresponding observation in the OUTFOR= data set. However, if constraints are specified in the CONSTRAINT= data set for that observation, these constraints have precedence over the LOCKZERO option. Note that an unlocked equality constraint in the CONSTRAINT= data also has precedence over the LOCKZERO option. Similarly, an unlocked equality whose value is zero is not converted to a locked equality, even though the LOCKZERO option is specified.

**MAXITER=*k***

specifies the maximum number of predictor-corrector iterations performed by the interior point algorithm. The value *k* is an integer between 1 and the largest four-byte, signed integer, $2^{31} - 1$. The default value is MAXITER=100.

**PREDICTONLY**

specifies that only the predicted value be reconciled.

**SIGN=*option***

specifies the sign constraint on the reconciled series.

Valid values are as follows:

| | |
|---|---|
| MIXED | if the output series can have any sign. This is the default. |
| NONNEGATIVE \| POSITIVE | if the output series are supposed to be nonnegative. |
| NONPOSITIVE \| NEGATIVE | if the output series are supposed to be nonpositive. |

**STDMETHOD=***option*

specifies the method used to compute standard errors for the reconciled forecasts.

The following methods are provided:

| | |
|---|---|
| UNCHANGED | Reconciled standard errors are the original standard errors. |
| AGG | Reconciled standard errors are proportional to the original aggregated standard errors. |
| DISAGG | Reconciled standard errors are proportional to the original disaggregated standard errors. |

The default values are STDMETHOD=UNCHANGED. See the section "Details: HPFRECONCILE Procedure" on page 428 for more information about the methods of computing standard errors.

**STDDIFBD=***n*

specifies a positive number that defines boundaries for the percentage difference between the original standard error and the reconciled standard error. If the percentage difference is greater than the values specified in the STDDIFBD= option, the reconciled standard error will be equal to the boundary value. For example, if STDDIFBD=0.3, the reconciled standard errors will be within a 30% band of the original standard errors.

The default value is STDDIFBD=0.25.

**WEIGHTED**

specifies that the loss function for top-down reconciliation be weighted by the inverse of the variance of the input forecasts.

## AGGBY Statement

**AGGBY** *variables* **;**

When DIRECTION=BU and the AGGDATA= data set is not specified, the AGGBY statement can be used to specify the AGGBY variables, which is a proper subset of the BY variables that should appear in the OUTFOR= data set.

When DIRECTION=BU and neither the AGGDATA= data set nor the AGGBY statement is specified, the OUTFOR= data set contains no BY variables.

If the AGGDATA= data set is specified, the AGGBY statement is ignored. Note that when DIRECTION=TD, the AGGDATA= data set must be specified. Hence, the AGGBY statement is valid only when DIRECTION=BU.

## AGGDATA Statement

**AGGDATA** < *options* > **;**

The AGGDATA statement enables the user to specify custom names for forecasting variables in the AGGDATA= data set. The default names are ACTUAL, PREDICT, LOWER, UPPER, ERROR, and STD.

The following options can be specified in the AGGDATA statement.

**ACTUAL=***variable-name*
specifies the name of the variable in the AGGDATA= data set that contains the actual values.

**PREDICT=***variable-name*
specifies the name of the variable in the AGGDATA= data set that contains the predicted values.

**LOWER=***variable-name*
specifies the name of the variable in the AGGDATA= data set that contains the lower confidence limit values.

**UPPER=***variable-name*
specifies the name of the variable in the AGGDATA= data set that contains the upper confidence limit values.

**ERROR=***variable-name*
specifies the name of the variable in the AGGDATA= data set that contains the error values.

**STD=***variable-name*
specifies the name of the variable in the AGGDATA= data set that contains the standard error values.

## BY Statement

**BY** *variables* < *NOTSORTED* > **;**

The BY statement defines separate groups of observations for the DISAGGDATA= data set. BY variables can be either character or numeric.

All BY variables must exist in the DISAGGDATA= data set. Conversely, only a strict subset of or none of the BY variables must be present in the AGGDATA= data set. The BY variables that are present in the AGGDATA= data set are called AGGBY variables. Since the AGGBY variables form a proper subset of the BY variables, their number must be less than the number of BY variables. PROC HPFRECONCILE finds the AGGBY variables by comparing the variables in the BY statement with the variables in the AGGDATA= data set. When DIRECTION=BU and the AGGDATA= data set is not specified, the AGGBY statement can be used to specify AGGBY variables. See the section "AGGBY Statement" on page 425 for more details.

A group of observations with the same combination of values for the AGGBY variables is called an AGGBY group. The AGGBY groups must follow the same sorting order in both the DISAGGDATA= and the AGGDATA= data sets. However, some groups can be missing from either data set if the NOTSORTED option is not specified. When the NOTSORTED option is specified, all AGGBY groups must be present in both data sets and must follow the same order.

## DISAGGDATA Statement

**DISAGGDATA** < *options* > **;**

The DISAGGDATA statement enables you to specify names for forecasting variables in the DISAGGDATA= data set. The default names are ACTUAL, PREDICT, LOWER, UPPER, ERROR, and STD.

The following options can be specified in the DISAGGDATA statement.

**ACTUAL=***variable-name*
specifies the name of the variable in the DISAGGDATA= data set that contains the actual values.

**PREDICT=***variable-name*
specifies the name of the variable in the DISAGGDATA= data set that contains the predicted values.

**LOWER=***variable-name*
specifies the name of the variable in the DISAGGDATA= data set that contains the lower confidence limit values.

**UPPER=***variable-name*
specifies the name of the variable in the DISAGGDATA= data set that contains the upper confidence limit values.

**ERROR=***variable-name*
specifies the name of the variable in the DISAGGDATA= data set that contains the error values.

**STD=***variable-name*
specifies the name of the variable in the DISAGGDATA= data set that contains the standard error values.

**BYVARSSORTED**
specifies that the DISAGGDATA= data set be sorted by the BY variables. This option improves input/output performance when there is an index defined on the BY variables. Use of an index degrades the performance as compared to processing a sorted data set. If the BYVARSSORTED option is not specified, PROC HPFRECONCILE uses only the index for processing and disregards the sorting order. If you specify the BYVARSSORTED option in the DISAGGDATA statement, PROC HPFRECONCILE exploits the sorting order to achieve better performance and uses the index minimally.

## ID Statement

**ID** *variable INTERVAL=interval* < */options* > **;**

The ID statement names a numeric variable that identifies observations in the input and output data sets. The ID variable's values are assumed to be SAS date, time, or datetime values. In addition, the ID statement specifies the frequency associated with the time series. If the ID statement is specified, the INTERVAL= option must also be specified, and the ID variable must be present and must have the same frequency in both the DISAGGDATA= data set and the AGGDATA= data set. If an ID statement is not specified, then a number derived from the observation number, with respect to the BY group, is used as the time ID. The number is derived so as to align the last observations of all BY groups.

The following options can be used in the ID statement.

**ALIGN=***option*
controls the alignment of SAS dates used to identify output observations. Internal processing uses aligned versions of the values of START= and END= options (if specified) and values of ID variable in input observations. The ALIGN= option accepts the following values: BEGIN, MIDDLE, and END. BEGIN is the default.

**END=***option*
specifies a SAS date, datetime, or time value that represents the date at which the reconciliation should end. If the largest time ID variable value is less than the END= value, this option has no effect.

**FORMAT=***option*
specifies a SAS format used for the DATE variable in the output data sets. The default format is the same as that of the DATE variable in the DATA= data set.

**INTERVAL=***interval*
specifies the frequency of the input time series. The frequency must be the same for all input data sets. For example, if the input data sets consist of quarterly observations, then INTERVAL=QTR should be used. See the *SAS/ETS User's Guide* for the intervals that can be specified.

**IRREGULAR**
specifies whether to allow for irregularities in the ID variable frequency. By default, irregularities are not allowed. That is, all ID values that correspond to the INTERVAL= frequency must be present between the START= and END= values in both AGGDATA= and DISAGGDATA= data sets.

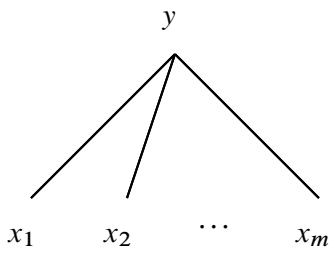
**START=***option*
specifies a SAS date, datetime, or time value that represents the time ID value at which the reconciliation should begin. This option can be used to limit the reconciliation process only to forecasts that are outside the historical period. For example, START="&sysdate"D uses the automatic macro variable SYSDATE to start the reconciliation at the current date.

# Details: HPFRECONCILE Procedure

## Notation

Assume a two-level hierarchical structure as depicted in Figure 13.2.

**Figure 13.2** Hierarchical Structure



Let $y_t$ be the values of the parent series at time $t$, and let $\mathbf{x}_t = [x_{1,t}, x_{2,t}, \ldots, x_{m,t}]'$ be the vector child series at time $t$, $t = 1, \ldots, T$. As usual, indicate by $\hat{y}_t$ and $\hat{\mathbf{x}}_t$ the pre-reconciliation forecasts of $y_t$ and $\mathbf{x}_t$, respectively, and denote by $\hat{\boldsymbol{\sigma}}_t = [\hat{\sigma}_{1,t}, \hat{\sigma}_{2,t}, \ldots, \hat{\sigma}_{m,t}]'$ the vector of prediction standard error for $\hat{\mathbf{x}}_t$. Denote by $\hat{\boldsymbol{\Sigma}}$ the diagonal matrix whose main diagonal is $\hat{\boldsymbol{\sigma}}_t^2$. Let the superscript tilde indicate the reconciled values, so that $\tilde{y}_t$ and $\tilde{\mathbf{x}}_t$ indicate the reconciled values of $\hat{y}_t$ and $\hat{\mathbf{x}}_t$, respectively. The number of child series $m$ can vary with $t$; however, for simplicity, it is considered fixed in the following discussion.

At each time $t$, the values of the series $x_{i,t}$, $i = 1 \ldots, m$, and $y_t$ are bound by an aggregation constraint. For example, if the $x_i$'s are the sales at store level for a retail company, then $y_t$ can be either the total sales at company level or the average sales per store. The aggregation constraint is $y_t = \sum_{i=1}^{m} x_{i,t}$, when you specify the AGGREGATE=TOTAL option of the PROC HPFRECONCILE statement. If instead you specify the AGGREGATE=AVERAGE option, the constraint is $y_t = \frac{1}{m} \sum_{i=1}^{m} x_{i,t}$.

If you need to have forecasts at both levels of the hierarchy, it is often more convenient to produce statistical forecasts separately for each series. However, the resulting forecasts do not abide by the aggregation constraint that binds the original series. The after-the-fact process through which the statistical forecasts are modified to enforce the aggregation constraint is called *reconciliation*.

By determining whether the upper-level forecasts or the lower-level forecasts are adjusted to meet the aggregation constraint, you can distinguish between bottom-up (BU) and top-down (TD) reconciliation. PROC HPFRECONCILE enables you to impose constraints on the individual reconciled forecasts. For example, you can require that $\tilde{x}_1 = 10$ and $\tilde{x}_2 \geq 15$.

## Top-Down Reconciliation

The goal of top-down (TD) reconciliation is to adjust the statistical forecasts $\hat{x}_{i,t}$ to obtain a new series $\{\tilde{x}_{i,t}\}$ of reconciled forecasts so that the sum of the reconciled forecasts at each fixed time $t$ is equal to $\hat{y}_t$ and the sum satisfies the constraints that you specify in the CONSTRAINT= data set.

The problem can be restated as follows: minimize with respect to $\tilde{\mathbf{x}}$ a quadratic loss function

$$L(\tilde{\mathbf{x}}_t; \hat{\mathbf{x}}_t)$$

subject to the following constraints:

- the *top-down constraint*

$$\sum_{i=1}^{m} \tilde{x}_{i,t} = \hat{y}_t$$

- the equality constraints

$$\tilde{x}_{i,t} = e_{i,t} \qquad i \in E_t$$

- the lower bounds

$$\tilde{x}_{i,t} \geq l_{i,t} \qquad i \in L_t$$

- the upper bounds

$$\tilde{x}_{i,t} \leq u_{i,t} \qquad i \in U_t$$

where $E_t$, $L_t$, and $U_t$ are subsets of $\{1, 2, \ldots, m\}$.

Equality constraints on the reconciled predicted values can be imposed with the EQUALITY variable of the CONSTRAINT= data set, or by using the WEIGHTED option with zero standard errors in the DISAGGDATA= data set. Bounds can be imposed either with the SIGN= option or with the LOWER and UPPER variables of the CONSTRAINT= data set.

PROC HPFRECONCILE employs an iterative interior point algorithm to solve the constrained quadratic optimization problem.

## Choice of Loss Function

The loss function takes the following functional forms:

- When DISAGGREGATION=DIFFERENCE, the loss function is

$$L(\tilde{\mathbf{x}}_t; \hat{\mathbf{x}}_t) = (\tilde{\mathbf{x}}_t - \hat{\mathbf{x}}_t)' \mathbf{W}^{-1} (\tilde{\mathbf{x}}_t - \hat{\mathbf{x}}_t)$$

- When DISAGGREGATION=PROPORTIONS, the loss function is

$$L(\tilde{\mathbf{x}}_t; \hat{\mathbf{x}}_t) = (\tilde{\mathbf{x}}_t - \hat{\mathbf{x}}_t)' \hat{\mathbf{X}}^{-\frac{1}{2}} \mathbf{W}^{-1} \overline{\hat{\mathbf{X}}^{-\frac{1}{2}}} (\tilde{\mathbf{x}}_t - \hat{\mathbf{x}}_t)$$

where $\mathbf{W}$ is a positive semidefinite matrix of weights independent of $\tilde{\mathbf{x}}_t$, $\hat{\mathbf{X}}^{-\frac{1}{2}}$ is a diagonal matrix with the square root of $\hat{\mathbf{x}}_t$ on the main diagonal, and $\overline{\hat{\mathbf{X}}^{-\frac{1}{2}}}$ is its complex conjugate.

If the WEIGHTED option is not specified, $\mathbf{W}$ is the identity matrix $\mathbf{I}$. If the WEIGHTED option is specified, $\mathbf{W} = \hat{\mathbf{\Sigma}}$, the diagonal matrix with the estimated variances $\hat{\sigma}^2_{i,t}$ of $\hat{x}_{i,t}$ on the main diagonal. If an observation has zero prediction standard error, $\hat{\sigma}_{j,t} = 0$, it is equivalent to imposing a locked equality constraint equal to the original forecast, $\tilde{x}_{j,t} = \hat{x}_{j,t}$. However, if a locked equality constraint is specified in the CONSTRAINT= data set for that observation, the locked equality constraint has precedence over the zero weight.

When DISAGGREGATION=DIFFERENCE, the loss function is defined for any value of $\hat{\mathbf{x}}_t$.

When DISAGGREGATION=PROPORTIONS, the loss function is defined only when all $\hat{x}_{i,t}$ are different from zero. The solutions can be extended to the zero cases by letting $\tilde{x}_{i,t} := 0$ when $\hat{x}_{i,t} = 0$, if there is at least one $\hat{x}_{j,t}$ that is different from zero. The case where all $\hat{x}_{j,t}$ are zero is handled by setting $\tilde{x}_{i,t} := \frac{\hat{y}_t}{m}$ when AGGREGATE=TOTAL and $\tilde{x}_{i,t} := \hat{y}_t$ when AGGREGATE=AVERAGE.

## Unconstrained Solutions

When the only constraint is the top-down constraint and $\mathbf{W} = \mathbf{I}$, the top-down problem admits intuitive solutions.

When DISAGGREGATION=DIFFERENCE, the loss function becomes

$$L(\tilde{\mathbf{x}}_t; \hat{\mathbf{x}}_t) = \sum_{i=1}^{m} (\hat{x}_{i,t} - \tilde{x}_{i,t})^2$$

This leads to the following solution

$$\tilde{x}_{i,t} = \hat{x}_{i,t} + \frac{\hat{r}_t}{m}$$

where $\hat{r}_t$ is the forecasting aggregate error—that is, when AGGREGATE=TOTAL,

$$\hat{r}_t := \hat{y}_t - \sum_{i=1}^{m} \hat{x}_{i,t}$$

and, when AGGREGATE=AVERAGE,

$$\hat{r}_t := m\hat{y}_t - \sum_{i=1}^{m} \hat{x}_{i,t}$$

Thus, when DISAGGREGATION=DIFFERENCE, the reconciled forecast $\tilde{x}_{i,t}$ is found by equally splitting the aggregation error $\hat{r}_t$ among the lower-level forecasts $\hat{x}_{i,t}$.

Notice that even if all statistical forecasts $\hat{x}_{i,t}$ are strictly positive, the reconciled forecasts $\tilde{x}_{i,t}$ need not be so if no bounds are specified via the SIGN= option. In particular, $\hat{x}_{i,t} = 0$ does not imply $\tilde{x}_{i,t} = 0$.

If DISAGGREGATION=PROPORTIONS, the loss function becomes

$$L(\tilde{\mathbf{x}}_t; \hat{\mathbf{x}}_t) = \sum_{i=1}^{m} \frac{\left(\hat{x}_{i,t} - \tilde{x}_{i,t}\right)^2}{|\hat{x}_{i,t}|}$$

This leads to the following solutions

$$\tilde{x}_{i,t} = \hat{x}_{i,t} + \frac{|\hat{x}_{i,t}|}{\sum_{j=1}^{m} |\hat{x}_{j,t}|}\hat{r}_t$$

When AGGREGATE=TOTAL and all the $\hat{x}_{j,t}$ have the same sign, the solution resolves to

$$\tilde{x}_{i,t} = \frac{\hat{x}_{i,t}}{\sum_{j=1}^{m} \hat{x}_{j,t}}\hat{y}_t$$

When AGGREGATE=AVERAGE and all the $\hat{x}_{j,t}$ have the same sign, the solution resolves to

$$\tilde{x}_{i,t} = \frac{\hat{x}_{i,t}}{\sum_{j=1}^{m} \hat{x}_{j,t}} m\hat{y}_t$$

Thus, the reconciled forecast $\tilde{x}_{i,t}$ is found by disaggregating the upper-level forecasts according to the proportion that $\hat{x}_{i,t}$ represents in the total sum of the lower-level forecasts.

## Missing Values

Missing values of type `'.F'` in the input are interpreted as failed forecasts. If a type `'.F'` missing value is detected in any of the variables that are needed to compute the reconciled prediction, then the reconciled prediction too takes the value `'.F'`. If the IGNOREMISSF option of the HPFRECONCILE statement is specified, the `'.F'` missing values are treated as `'.'` missing values.

When some of the predicted values $\hat{x}_{i,t}$ are missing, with any type of missing values different from `'.F'`, the missing values are replaced by the actual values $x_{i,t}$, if these are present. This is done to prevent bias between the aggregated and reconciled forecasts, which results from models in which missing values in the predictions are generated because of the presence of lagged variables. If the actual value is also missing, the series is excluded from the reconciliation process.

When you use the WEIGHTED option and the standard error is missing, the weight is assumed to be the average of the nonmissing variances. If all standard errors are missing, the weights are assumed to be all equal to one, which is equivalent to not using the WEIGHTED option.

### Standard Errors

When STDMETHOD=UNCHANGED, the reconciled standard error $\tilde{\sigma}_{i,t}$ of $\tilde{x}_{i,t}$ is equal to the original standard error $\hat{\sigma}_{i,t}$ of $\hat{x}_{i,t}$.

When STDMETHOD=DISAGG, the reconciled standard error is proportional to the original disaggregated standard error and is computed as follows:

$$\tilde{\sigma}_{i,t} = |w|\hat{\sigma}_{i,t}$$

where $w = \frac{\tilde{x}_{i,t}}{\hat{x}_{i,t}}$.

When STDMETHOD=AGG, the reconciled standard error of $\tilde{x}_{i,t}$ is proportional to the aggregated standard error,

$$\tilde{\sigma}_{i,t} = |\tilde{p}_{i,t}|\hat{\sigma}_t$$

where $\tilde{p}_{i,t} = \frac{\tilde{x}_{i,t}}{\tilde{y}_t}$ and $\hat{\sigma}_t$ is the standard deviation of $\hat{y}_t$.

If the selected method for the standard errors fails, PROC HPFRECONCILE tries to use a different method and displays a warning message in the log. For example, if STDMETHOD=DISAGG and the standard error is missing in the DISAGGDATA= data set, STDMETHOD=AGG is used instead, if possible. In such a case, the _RECONSTATUS_ variable identifies the observation that was not reconciled according to your preferences. You can also use the ERRORTRACE=ID option to display a message in the log that identifies the ID values for which the standard error was not reconciled according to your specification.

Care should be taken in interpreting standard errors when constraints are imposed on the reconciled forecasts. The presence of constraints renders the meaning of the reconciled standard errors ambiguous.

### Confidence Limits

When CLMETHOD=SHIFT, the reconciled confidence limits are computed by re-centering the original confidence limits around the reconciled predicted values.

When CLMETHOD=GAUSS, the reconciled confidence limits are computed by assuming that the series is Gaussian with standard error equal to the reconciled standard error.

If the selected method for the confidence limits fails, PROC HPFRECONCILE tries to use a different method and displays a warning message in the log. For example, if CLMETHOD=SHIFT and the confidence limits are missing in the DISAGGDATA= data set, CLMETHOD=GAUSS is used instead. In such a case, the _RECONSTATUS_ variable identifies the observation that was not reconciled according to your preferences. You can also use the ERRORTRACE=ID option to display a message in the log that identifies the ID values for which the confidence limits were not reconciled according to your specification.

Care should be used in interpreting confidence limits when constraints are imposed on the reconciled forecasts. The presence of constraints renders the meaning of reconciled confidence limits ambiguous.

## Bottom-Up Reconciliation

The goal of bottom-up (BU) reconciliation is to adjust $\hat{y}_t$ to obtain a new series $\{\tilde{y}_t\}$ of reconciled forecasts so that $\{\tilde{y}_t\}$ satisfies the aggregation constraint.

When AGGREGATE=TOTAL, this is done by setting

$$\tilde{y}_t = \sum_{i=1}^{m} \hat{x}_{i,t} \qquad t = 1, 2, \ldots$$

When AGGREGATE=AVERAGE, this is done by setting

$$\tilde{y}_t = \frac{1}{m} \sum_{i=1}^{m} \hat{x}_{i,t} \qquad t = 1, 2, \ldots$$

Because the bottom-up problem is exactly identified and admits a unique solution, additional constraints on $\tilde{y}_t$ specified in the CONSTRAINT= data set are either already satisfied by the solution or result in an infeasible problem that will be flagged by the _RECONSTATUS_ variable in the OUTFOR= data set.

## Missing Predicted Values

Missing values of type `'.F'` in the input are interpreted as failed forecasts. If a type `'.F'` missing value is detected in any of the variables that are needed to compute the reconciled prediction, then the reconciled prediction also takes the value `'.F'`. If the IGNOREMISSF option of the HPFRECONCILE statement is specified, the `'.F'` missing values are treated as `'.'` missing values.

When some of the predicted values $\hat{x}_{i,t}$ are missing, with missing value different from `'.F'`, the missing values are replaced by the actual values $x_{i,t}$, if these are present. This is done to prevent bias between the aggregated and reconciled forecasts, which results from models in which missing values in the predictions are generated because of the presence of lagged variables. However, if all predicted values $\hat{x}_{i,t}$ are missing for a given time ID $t$, then the reconciliation process is considered failed for this $t$, even though the actual values $x_{i,t}$ are not missing.

## Standard Errors

When STDMETHOD=UNCHANGED, the reconciled standard error $\tilde{\sigma}_t$ of $\tilde{y}_t$ is equal to the original standard error $\hat{\sigma}_t$ of $\hat{y}_t$.

When STDMETHOD=AGG, the reconciled standard error is proportional to the original aggregated standard error and is computed as follows:

$$\tilde{\sigma}_t = |w| \hat{\sigma}_t$$

where $w = \frac{\tilde{y}_t}{\hat{y}_t}$.

If STDMETHOD=DISAGG, the reconciled standard error $\tilde{\sigma}_t$ is

$$\tilde{\sigma}_t = \sqrt{\sum_{i=1}^{m} \hat{\sigma}_{i,t}^2}$$

when AGGREGATE=TOTAL, and

$$\tilde{\sigma}_t = \frac{1}{m} \sqrt{\sum_{i=1}^{m} \hat{\sigma}_{i,t}^2}$$

when AGGREGATE=AVERAGE.

If the selected method for the standard errors fails, PROC HPFRECONCILE tries to use a different method and displays a warning message in the log. For example, if STDMETHOD=AGG and the standard error is missing in the AGGDATA= data set, STDMETHOD=DISAGG is used instead, if possible. In such a case, the _RECONSTATUS_ variable identifies the observation that was not reconciled according to your preferences. You can also use the ERRORTRACE=ID option to display a message in the log that identifies the ID values for which the standard error was not reconciled according to your specification.

### Confidence Limits

When CLMETHOD=SHIFT, the reconciled confidence limits are computed by re-centering the original confidence limits around the reconciled predicted values.

When CLMETHOD=GAUSS, the reconciled confidence limits are computed by assuming that the series is Gaussian with standard error equal to the reconciled standard error.

If the selected method for the confidence limits fails, PROC HPFRECONCILE tries to use a different method and displays a warning message in the log. For example, if CLMETHOD=SHIFT and the confidence limits are missing in the AGGDATA= data set, CLMETHOD=GAUSS is used instead, if possible. In such a case, the _RECONSTATUS_ variable identifies the observation that was not reconciled according to your preferences. You can also use the ERRORTRACE=ID option to display a message in the log that identifies the ID values for which the confidence limits were not reconciled according to your specification.

## Data Set Input/Output

### AGGDATA= Data Set

The AGGDATA= data set contains either a proper subset of or none of the variables specified in the BY statement, the time ID variable in the ID statement (when this statement is specified), and the following variables:

| | |
|---|---|
| _NAME_ | variable name |
| PREDICT | predicted values |

The following variables can optionally be present in the AGGDATA= data set and are used when available. If not present, their value is assumed to be missing for computational purposes.

| | |
|---|---|
| ACTUAL | actual values |
| LOWER | lower confidence limits |
| UPPER | upper confidence limits |
| ERROR | prediction errors |
| STD | prediction standard errors |

Typically, the AGGDATA= data set is generated by the OUTFOR= option of the HPFENGINE procedure. See Chapter 7, "The HPFENGINE Procedure," for more details.

The AGGDATA= data set must be either sorted by the AGGBY variables and by the ID variable (when the latter is specified) or indexed on the AGGBY variables. Even when the data set is indexed, if the ID variable is specified, its values must be sorted in ascending order within each AGGBY group. See section "BY Statement" on page 426 for details about AGGBY variables and AGGBY groups.

You can specify custom names for the variables in the AGGDATA= data set by using the AGGDATA statement. See the section "AGGDATA Statement" on page 426 for more details.

## DISAGGDATA= Data Set

The DISAGGDATA= data set contains the variables specified in the BY statement, the variable in the ID statement (when this statement is specified), and the following variables:

| | |
|---|---|
| _NAME_ | variable name |
| PREDICT | predicted values |

The following variables can optionally be present in the DISAGGDATA= data set and are used when available. If not present, their value is assumed to be missing for computational purposes.

| | |
|---|---|
| ACTUAL | actual values |
| LOWER | lower confidence limits |
| UPPER | upper confidence limits |
| ERROR | prediction errors |
| STD | prediction standard errors |

Typically, the DISAGGDATA= data set is generated by the OUTFOR= option of the HPFENGINE procedure. See Chapter 7, "The HPFENGINE Procedure," for more details.

The DISAGGDATA= data set must be either sorted by the BY variables and by the ID variable when the latter is specified, or indexed on the BY variables. If the variable _NAME_ is present and has multiple values, then the index must be a composite index on BY variables and _NAME_, in that order. If _NAME_ is present and has only one value, then the index can contain only BY variables. Even when the data set is indexed, if the ID variable is specified, its values must be sorted in ascending order within each BY, or BY and _NAME_ group, as applicable. Indexing the DISAGGDATA= data set on the BY variables when it is already sorted by the BY variables leads to less efficient and less scalable operation if the available memory is not sufficient to hold the disaggregated data for the AGGBY group that is being processed. The amount of memory required depends on, among other things, the length of the series, the number of BY groups for each AGGBY group, and the number and format of the BY variables. For example, if there are four BY variables, each 16 characters long, 10,000 BY groups within each AGGBY group, and each series has length 100, then the minimum required memory for efficient processing is approximately 100 MB. If the memory is not sufficient, sorting the DISAGGDATA= data set, not indexing, is more efficient.

You can specify custom names for the variables in the DISAGGDATA= data set by using the DISAGGDATA statement. See the section "DISAGGDATA Statement" on page 427 for more details.

## CONSTRAINT= Data Set

The CONSTRAINT= data set specifies the constraints to be applied to the reconciled forecasts. It contains the BY variables for the level at which reconciled forecasts are generated. That is, it contains the AGGBY variables when DIRECTION=BU, and the variables specified in the BY statement when DIRECTION=TD. If the _NAME_ variable is present in the AGGDATA= and DISAGGDATA= data set, it must also be present in the CONSTRAINT= data set. Additionally, the CONSTRAINT= data set contains the variable in the ID statement (when this statement is specified), and the following variables:

| | |
|---|---|
| EQUALITY | an equality constraint for the predicted reconciled value |
| UNLOCK | a flag that specifies whether the equality constraint should be strictly enforced. Admissible values are as follows:<br>0 The equality constraint is locked.<br>1 The equality constraint is unlocked.<br>When EQUALITY is nonmissing and the UNLOCK flag is missing, the equality is treated as locked. |
| LOWERBD | lower bounds for the reconciled forecasts |
| UPPERBD | upper bounds for the reconciled forecasts |

Locked equality constraints are treated as constraints, and therefore their value is honored. Unlocked equalities are instead treated as regular forecasts and, in general, are changed by the reconciliation process.

A constraint is said to be *active* when the reconciled prediction lies on the constraint. By definition, locked equalities are always active constraints.

If the NOTSORTED option is specified in the BY statement, then any BY group in the CONSTRAINT= data set that is out of order with respect to the BY groups in the AGGDATA= or DISAGGDATA= data set is ignored without any error or warning message. If the NOTSORTED option is not specified, then the BY groups in the CONSTRAINT= data set must be in the same sorted order as the AGGBY groups in the AGGDATA= data set when DIRECTION=BU, and in the same sorted order as the BY groups in the DISAGGDATA= data set when DIRECTION=TD; otherwise processing stops at the first such occurrence of a mismatch.

### OUTFOR= Data Set

The OUTFOR= data set contains the following variables:

| | |
|---|---|
| _NAME_ | variable name |
| ACTUAL | actual values |
| PREDICT | predicted values |
| LOWER | lower confidence limits |
| UPPER | upper confidence limits |
| ERROR | prediction errors |
| STD | prediction standard errors |
| _RECONSTATUS_ | reconciliation status |

Additionally, it contains any other variable that was present in the input data set at the same level—that is, the DISAGGDATA= data set when DIRECTION=TD and the AGGDATA= data set when DIRECTION=BU.

When DIRECTION=BU and the AGGDATA= data set has not been specified, the OUTFOR= data set contains the variables in the previous list, the BY variables specified in the AGGBY statement, and the time ID variable in the ID statement.

If reconciliation fails with _RECONSTATUS_ between 1000 and 6000, PROC HPFRECONCILE copies the input values of the relevant variables to the OUTFOR= data set. If a variable is not present in the input data set, its value is set to missing in the OUTFOR= data set. The only exception to this rule is when the problem is infeasible and the FORCECONSTRAINT option is specified. See the section "The FORCECONSTRAINT Option" on page 438 for more details on the latter case.

The OUTFOR= data set is always sorted by the BY variables (and by the _NAME_ variable and time ID variable when these variables are present) even if input data sets are indexed and not sorted.

If the ID statement is specified, then the values of the ID variable in OUTFOR= data set are aligned based on the ALIGN= and INTERVAL= options specified on the ID statement. If ALIGN= option is not specified, then the values are aligned to the beginning of the interval.

If the RECDIFF option of the HPFRECONCILE statement has been specified, the OUTFOR= data sets also contains the following variable:

RECDIFF difference between the reconciled predicted value and the original predicted value

The _RECONSTATUS_ variable contains a code that specifies whether the reconciliation was successful or not. A corresponding message is also displayed in the log. You can use the ERRORTRACE= option to define how often the error and warning messages are displayed in the log. The _RECONSTATUS_ variable can take the following values:

0 Reconciliation was successful.

400 A unlocked equality constraint has been imposed.

500 A locked equality constraint has been imposed.

600 A lower bound is active.

700 An upper bound is active.

1000 The ID value is out of the range with respect to the START= and END= interval.

2000 There is insufficient data to reconcile.

3000 Reconciliation failed for the predicted value. This implies that it also failed for the confidence limits and standard error.

4000 Reconciliation failed for the standard error.

5000 Reconciliation failed for the confidence limits.

6000 The constrained optimization problem is infeasible.

7000 The option DISAGGREGATION=PROPORTION has been changed to DISAGGREGATION=DIFFERENCE for this observation because of a discordant sign in the input.

8000 The option STDMETHOD= provided by the user has been changed for this observation.

9000 The option CLMETHOD= provided by the user has been changed for this observation.

10000 The standard error hit the limits imposed by the STDDIFBD= option.

11000 Multiple warnings have been displayed in the log for this observation.

12000 The number of missing values in the STD variable in the DISAGGDATA= data set is different from the number of missing values in the union of the PREDICT and ACTUAL variables.

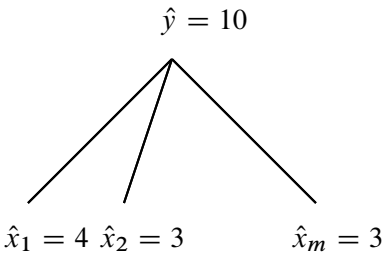
13000 The solution might be suboptimal. This means that the optimizer did not find an optimal solution, but the solution provided satisfies all constraints.

14000 A failed forecast ".F" has been detected in a relevant input variable.

### The FORCECONSTRAINT Option

The FORCECONSTRAINT option applies when there are conflicts between the aggregation constraint and one or more constraints that you specify using the CONSTRAINT= data set, the SIGN= option, or the WEIGHTED option with zero weights. By default, when reconciliation is impossible, PROC HPFRECONCILE copies the input to the OUTFOR= data set without modification. However, if the reconciliation is infeasible because of a conflict between the constraints you specified and the aggregation constraint, you can ask PROC HPFRECONCILE to impose your constraints on the output even though that results in a violation of the aggregation constraint. For example, assume the input is described by the diagram in Figure 13.3 and assume you want to impose the following constraints on the reconciled forecasts: $\tilde{x}_1 = 7,\ \tilde{x}_2 \geq 5,\ \tilde{x}_3 \geq 0$.

**Figure 13.3** FORCECONSTRAINT Option




The constraints are clearly in conflict the aggregation constraint $\sum \tilde{x}_i = \hat{y}$; therefore, PROC HPFRECONCILE will consider the problem infeasible. If you do not specify the FORCECONSTRAINT option, the predicted values in the OUTFOR= data set will equal the input predicted values (that is, $\tilde{x}_1 = 4,\ \tilde{x}_2 = 3,\ \tilde{x}_3 = 3$) and the _RECONSTATUS_ variable will take the value 6000. If you specify the FORCECONSTRAINT option, the OUTFOR= data set will contain the values $\tilde{x}_1 = 7,\ \tilde{x}_2 = 5,\ \tilde{x}_3 = 0$.

## OUTINFEASIBLE= Data Set

The OUTINFEASIBLE= data set contains summary information about the nodes in the hierarchy for which reconciliation is infeasible because the aggregation constraint is incompatible with the constraints supplied by the user.

The OUTINFEASIBLE= data set is always produced at the level of the AGGDATA= data set.

The OUTINFEASIBLE= data set contains the AGGBY variables present in the AGGDATA= data set, the time ID variable, when it is specified, and the following variables:

| | |
|---|---|
| _NAME_ | variable name |
| ISRECONCILED | takes value 1 when the node is reconciled, and value 0 when it is not |
| FINALPREDICT | the predicted value for the parent node |
| AGGCHILDPREDICT | the aggregated prediction of the children nodes |
| LOWERBD | the lower bound implied by the constraints on FINALPREDICT |
| UPPERBD | the upper bound implied by the constraints on FINALPREDICT |

If the ID statement is specified, then the values of the ID variable in OUTINFEASIBLE= data set are aligned based on the ALIGN= and INTERVAL= options specified in the ID statement. If ALIGN= option is not specified, then the values are aligned to the beginning of the interval.

### OUTNODESUM= Data Set

The OUTNODESUM= data set contains the BY variables in the AGGDATA= data set (or in the AGGBY statement if the AGGDATA= data set is not specified), the time ID variable in the ID statement when this statement is specified, and the following variables:

| | |
|---|---|
| _NAME_ | variable name |
| NONMISSCHLD | number of nonmissing children of the current AGGBY group |

### OUTPROCINFO= Data Set

The OUTPROCINFO= data set contains the following variables:

| | |
|---|---|
| _SOURCE_ | source procedure that produces this data set |
| _STAGE_ | stage of the procedure execution for which the summary variable is reported |
| _NAME_ | name of the summary variable |
| _LABEL_ | description of the summary variable |
| _VALUE_ | value of the summary variable |

For PROC HPFRECONCILE, the value of the _SOURCE_ variable is **HPFRECONCILE** and the value of the _STAGE_ variable is **ALL** for all observations. It contains observations that corresponds to each of the following values of the _NAME_.

| | |
|---|---|
| NOBS_RECON | total number of observations subject to reconciliation |
| NOBS_SUCCESS | number of observations with successful reconciliation |
| NOBS_PREDICTFAIL | number of observations for which reconciliation failed for PREDICT. This number does not include failures to reconcile due to an infeasible problem or a failed (".F") forecast. |
| NOBS_PROBLEMSTATUS | number of observations for which some problem was encountered. This is the number of observations in the OUTFOR= data set that have a _RECONSTATUS_ value greater or equal to 1000. |
| NOBS_INFEASIBLE | number of observations for which reconciliation is infeasible due to incompatible constraints |
| NOBS_SUBOPTIMAL | number of observations for which the optimizer did not find an optimal solution |
| NOBS_LOCKEQ | number of observations subject to a locked equality constraint |
| NOBS_USER_LOCKEQ | number of observations for which a locked equality was specified in the CONSTRAINT= data set |
| NOBS_LOWERBD | number of observations for which a lower bound was imposed |

| | |
|---|---|
| NOBS_USER_LOWERBD | number of observations for which a lower bound was specified in the CONSTRAINT= data set |
| NOBS_UPPERBD | number of observations for which an upper bound was imposed |
| NOBS_USER_UPPERBD | number of observations for which an upper bound was specified in the CONSTRAINT= data set |
| NOBS_ACTIVE_LOWERBD | number of observations for which a lower bound is active |
| NOBS_ACTIVE_UPPERBD | number of observations for which an upper bound is active |
| NOBS_FAILED_FORECAST | number of observations for which a failed forecast (".F") was written |
| NPROB_TOTAL | total number of possible problems. One reconciliation problem is possible for each distinct value of time ID variable that appears in AGGDATA= or DISAGGDATA= data sets. |
| NPROB_CHANGED_LOSS | number of reconciliation problems for which DISAGGREGATION= option was changed internally because the supplied or default option was not feasible |
| NPROB_CHANGED_CLMETHOD | number of reconciliation problems for which CLMETHOD= option was changed internally because the supplied or default option was not feasible |
| NPROB_CHANGED_STDMETHOD | number of reconciliation problems for which STDMETHOD= option was changed internally because the supplied or default option was not feasible |
| NPROB_RECON | total number of problems subject to reconciliation |
| NPROB_INFEASIBLE | number of infeasible reconciliation problems due to incompatible constraints |
| NPROB_SUBOPTIMAL | number of reconciliation problems for which the optimizer did not find an optimal solution |
| NPROB_OPTIMIZER | number of reconciliation problems solved by using the optimizer |
| ID_MIN | minimum value of time ID for which reconciliation was attempted |
| ID_MAX | maximum value of time ID for which reconciliation was attempted |
| NAGGBY | total number of AGGBY groups processed |
| AVGDBY_PERABY | average number of BY groups per AGGBY group |
| NBY_IRREGULAR_ID | number of BY groups processed partially because of irregular ID values |
| NAGGBY_IRREGULAR_ID | number of AGGBY groups processed partially because of irregular ID values |
| AVGACTIVEDBY_PERID | average number of active BY groups per time ID for which reconciliation was attempted |
| NCONS_READ | total number of constraints read in the CONSTRAINT= data set |
| NCONS_IN_VALID_ID_RANGE | total number of constraints in the CONSTRAINT= data set with time ID values in the [START=,END=] range |
| NCONS_USED | number of constraints used |

| | |
|---|---|
| CONS_ISANYUNMATCHED | If any of the constraints in the CONSTRAINT= data set is left unmatched and unprocessed, then _VALUE_ for this observation is set to 1; otherwise, it is set to 0. |
| RC | return code of PROC HPFRECONCILE |

# Examples: HPFRECONCILE Procedure

## Example 13.1: Reconciling a Hierarchical Tree

The HPFRECONCILE procedure reconciles forecasts between two levels of a hierarchy. It can also be used recursively for reconciling the whole hierarchy.

Consider the hierarchy structure for the SASHELP.PRICEDATA data set outlined in Figure 13.1. You can reconcile the hierarchy top down, starting from the top level 0 down to the bottom level 2. At each new iteration, the OUTFOR= data set of the previous reconciliation step becomes the AGGDATA= data set of the current step.

First, you need to compute the statistical forecasts for all levels. The statistical forecasts for level 1 and level 2 were already computed in the section "Getting Started: HPFRECONCILE Procedure" on page 418, so only the forecasts at the company levels are left to compute as shown in the following statements.

```
/* Forecast series at company level */

* Step 1: model selection;
proc hpfdiagnose data=sashelp.pricedata
                 outest=lvl0est
                 modelrepository=work.mymodels
                 prefilter=both
                 criterion=mape;
   id date interval=month notsorted;
   forecast sale / accumulate=total;
   input price / accumulate=average;
run;

* Step 2: estimation and forecasting;
proc hpfengine data=sashelp.pricedata
               inest=lvl0est
               out=_null_
               outest=lvl0fest
               modelrepository=mymodels
               outfor=lvl0for;
   id date interval=month notsorted;
   forecast sale / task=select accumulate=total;
   stochastic price /accumulate=average;
run;
```

First, you reconcile the top and region levels. The output data set lvl1recfor contains the reconciled forecasts at level 1. This data set becomes the AGGDATA= data set for the next step of TD reconciliation that involves level 1 and level 2. You can check that the reconciled forecasts at level 2 add up to the forecasts at level 0.

```
/* Reconcile forecasts top down from company to region */

proc hpfreconcile disaggdata=lvl1for
                  aggdata=lvl0for
                  direction=TD
                  outfor=lvl1recfor;
   id date interval=month;
   by region;
run;

/* Reconcile forecasts top down from region to region/product */

proc hpfreconcile disaggdata=lvl2for
                  aggdata=lvl1recfor
                  direction=TD
                  outfor=lvl2recfor;
   id date interval=month;
   by region product;
run;

/* Verify that level 2 forecasts add up to level 0 forecasts */

proc timeseries data=lvl2recfor out=toprec ;
   id date interval=month notsorted  accumulate=total;
   var predict;
run;

proc compare base=lvl0for compare=toprec criterion=0.00001;
   var predict;
run;
```

You can also reconcile the hierarchy from the bottom up. In such a case, the OUTFOR= data set of the previous step becomes the DISAGGDATA= data set of the current step.

Alternatively, you could choose to reconcile the hierarchy from the *middle out* from an intermediate level. In this case, you choose an intermediate level as a starting point, and reconcile all levels above from the bottom up, while reconciling all levels below from the top down. In the following SAS statements, the hierarchy of SASHELP.PRICEDATA is reconciled from the middle out, starting from level 1.

```
/* Reconcile forecasts bottom up from region to company */

proc hpfreconcile disaggdata=lvl1for
                  aggdata=lvl0for
                  direction=BU
                  outfor=lvl0recfor;
   id date interval=month;
   by region;
run;

/* Reconcile forecasts top down from region to region/product */

proc hpfreconcile disaggdata=lvl2for
                  aggdata=lvl1for
```

```
                direction=TD
                outfor=lvl2recfor;
   id date interval=month;
   by region product;
run;
```

You can use the external forecasts feature of the HPFENGINE procedure to generate summary statistics and statistics of fit for the reconciled forecasts, as shown in the following SAS statements for the company level.

First, an external model spec is generated using PROC HPFEXMSPEC. The characteristics of estimated models that determine the options for PROC HPFEXMSPEC can be found in the OUTEST= data set of the HPFENGINE call for the corresponding level. In this case, the lvl0fest data set shows that the estimated model has three parameters and that the dependent variable sales has not undergone any transformation.

```
/* Generate external model spec */

proc hpfexmspec modelrepository=mymodels
                specname=lvl0exm;
   exm transform=none nparms=3;
run;
```

Subsequently, a selection list that contains the external model is defined with PROC HPFSELECT as shown in the following statements.

```
/* Generate select list */

proc hpfselect modelrepository=mymodels
               selectname=lvl0selexm;
   spec lvl0exm/ exmmap(predict=predict lower=lower
                        upper=upper stderr=std);
run;
```

Finally, the EXTERNAL statement of the HPFENGINE procedure is used in conjunction with the FORECAST statement to generate the OUTSTAT= and OUTSUM= data sets that correspond to the reconciled forecasts input data set lvl0recfor and the model specifications contained in the external model lvl0exm as shown in the following statements.

```
/* Create OUTSTAT= and OUTSUM= data sets */
proc hpfengine data=lvl0recfor(rename=(actual=sales))
               out=_NULL_
               outstat=lvl0outstat
               outsum=lvl0outsum
               modelrepository=mymodels
               globalselection=lvl0selexm;
   id date interval=month notsorted;
   forecast sales;
   external predict lower
            upper std;
run;
```

## Example 13.2: Aggregating Forecasts

If you do not provide the AGGDATA= input data set, but provide only the DISAGGDATA= data set, PROC HPFRECONCILE aggregates the forecasts according to the BY variable that you specify in the AGGBY option. If you use the options STDMETHOD=DISAGG and CLMETHOD=GAUSS, you can obtain standard errors and confidence interval as well.

In this example, the forecasts at level 2 of Figure 13.1 are aggregated to find forecasts at level 1 for the SASHELP.PRICEDATA data set.

```
/* Aggregate region/product forecasts to region level */

proc hpfreconcile disaggdata=lvl2for
                  direction=BU
                  outfor=lvl1aggfor
                  stdmethod=disagg
                  clmethod=gauss;
   id date interval=month;
   by region product;
   aggby region;
run;
```

## Example 13.3: Disaggregating Forecasts

You can use the HPFRECONCILE procedure to disaggregate top-level forecasts according to proportions that you supply. This can be accomplished by creating a DISAGGDATA= data set that contains the proportions that you want to use in place of the PREDICT variable.

In this example, the level 1 forecasts of the variable sale in the SASHELP.PRICEDATA data set are disaggregated to level 2 according to the historical median proportions.

First, a combination of DATA steps and PROC UNIVARIATE is used to compute the median proportions and merge them with the level 2 OUTFOR= data set from PROC HPFENGINE as shown in the following statements.

```
/* Compute total sales per region */

proc timeseries data=sashelp.pricedata out=lvl1sales ;
   id date interval=month notsorted  accumulate=total;
   by region;
   var sale;
run;

/* Compute sale proportions */

proc sort data=sashelp.pricedata out=tmp;
   by region date;
run;

data lvl2prop;
   merge tmp lvl1sales(rename=(sale=totsale));
```

```
   by region date;
   prop = sale / totsale;
run;

/* Compute median sale proportions */

proc sort data=lvl2prop;
   by region product;
run;

proc univariate data=lvl2prop noprint;
   var prop;
   by region product;
   output out=lvl2medprop median=medprop;
run;

/* Merge median proportions with level2 OUTFOR */

data lvl2medfor;
   merge lvl2for lvl2medprop;
   by region product;
run;
```

Then PROC HPFRECONCILE is invoked, using the DISAGGDATA statement to specify that the variable medprop is to be used instead of the default PREDICT.

Note that the proportions do not need to sum to one. PROC HPFRECONCILE automatically rescales them to sum to one as shown in the following statements.

```
/* Disaggregate level1 forecasts according to median sale */

proc hpfreconcile disaggdata=lvl2medfor
                  aggdata=lvl1for
                  direction=TD
                  stdmethod=unchanged
                  clmethod=gauss
                  outfor=lvl2recmedfor;
   disaggdata predict=medprop;
   by region product;
run;
```

The variable medprop in the OUTFOR=lvl2recmedfor data set contains the disaggregated forecasts according to the proportions that you supplied.

In this case, the options STDMETHOD=UNCHANGED and CLMETHOD=GAUSS are used to obtain standard errors and confidence intervals. However, you need to be aware that they might not be reliable.

Alternatively, if you are interested in disaggregating the predicted values only, you can use the PREDICTONLY option as in the following statements.

```
/* Disaggregate level1 predict only */

proc hpfreconcile disaggdata=lvl2medfor
                  aggdata=lvl1for
                  direction=TD
                  predictonly
```

```
                  outfor=lvl2recmedfor;
   disaggdata predict=medprop;
   by region product;
run;
```

## Example 13.4: Imposing Constraints

You can impose constraints on the reconciled forecasts by using the CONSTRAINT= option or the SIGN= option.

This example revisits Example 13.1 and imposes different types of constraints on the reconciled forecasts. Suppose you want all reconciled forecasts to be nonnegative, and for the month of April 2003 you want the following:

1. Product 1 at Region 1 to have a locked equality of 400
2. Product 2 at Region 1 to have an unlocked equality of 400
3. Product 4 at Region 2 to be less or equal to 300

First, you need to create a CONSTRAINT= data set that contains the constraints you want for the date of April 2003.

```
/* Create constraint data set */

data constraint;
   length _name_ $32;
   input region product _name_ $ date MONYY7. equality
         unlock lowerbd upperbd;
datalines;
   1 1 sale Apr2003 400 0 . .
   1 2 sale Apr2003 400 1 . .
   2 4 sale Apr2003 . .   . 300
   ;
```

Then, you reconcile the two levels by using the SIGN=NONNEGATIVE option to impose the nonnegativity constraint and by using the CONSTRAINT= option to impose your constraints on the reconciled forecasts in April 2003. The PREDICTONLY option of the HPFRECONCILE statement restricts the reconciliation to the PREDICT variable as shown in the following statements.

```
/* Reconcile forecasts with constraints */

proc hpfreconcile disaggdata=lvl2for
                  aggdata=lvl1for
                  direction=TD
                  sign=nonnegative
                  constraint=constraint
                  outfor=lvl2recfor
                  predictonly;
   id date interval=month;
   by region product;
run;
```

# Chapter 14
# The HPFREPOSITORY Procedure (Experimental)

**Contents**


Overview: HPFREPOSITORY Procedure . . . . . **447**
Getting Started: HPFREPOSITORY Procedure . . . . . **448**
Syntax: HPFREPOSITORY Procedure . . . . . **449**
Functional Summary . . . . . 450
PROC HPFREPOSITORY Statement . . . . . 450
EVENTS Statement . . . . . 451
SPECIFICATIONS Statement . . . . . 451
TRAVERSE Statement . . . . . 451
VARIABLES Statement . . . . . 452
ODS Table Names . . . . . 452
Examples: HPFREPOSITORY Procedure . . . . . **453**
Example 14.1: Standard Use . . . . . 453
Example 14.2: Using the TRAVERSE Statement . . . . . 456


## Overview: HPFREPOSITORY Procedure

The HPFREPOSITORY procedure enables you to query a SAS Forecast Server Procedures repository and produce reports of its contents. A repository is a SAS catalog that holds the XML specfications that are needed by the HPFENGINE procedure to forecast time series, or that holds XML specifications that are generated by HPFENGINE to capture scoring information for the selected model for a given time series.

The XML specifications that are needed by the HPFENGINE procedure to forecast time series fall into two categories:

- those that represent instances of time series models
- those that represent lists of references to other XML specifications

The XML specifications that represent time series models include the following model families:

- exponential smoothing models (ESM)
- Box-Jenkins autoregressive integrated moving average models (ARIMA)

- unobserved component models (UCM)
- intermittent demand models (IDM)
- externally defined models (EXM)

An overview of the collection of time series model generation procedures can be found in Chapter 2, "Introduction."

The XML specifications that represent lists of references to other XML specifications include the following:

- model selection lists
- combined model lists

Further discussion of these concepts can be found in Chapter 15, "The HPFSELECT Procedure."

The XML specifications that represent scored time series forecasts include the following:

- exponential smoothing models (ESM)
- Box-Jenkins autoregressive integrated moving average models (ARIMA)
- unobserved component models (UCM)
- intermittent demand models (IDM)
- externally defined models (EXM)
- combined model lists

An overview of forecast scoring can be found in Chapter 7, "The HPFENGINE Procedure."

It is technically possible in some cases to use the same SAS catalog as both a model and score repository. However, SAS Institute advises against this practice.

## Getting Started: HPFREPOSITORY Procedure

This Getting Started example shows how to display the contents of a repository. Suppose the model repository WORK.MYMODELS contains three model specification files (A.XML, B.XML, C.XML) and a model selection list (MYSELECT.XML) that are created by the following SAS statements:

```
proc hpfarimaspec repository=work.mymodels name=a;
   forecast symbol=y p=12 diflist=(1 12) noint;
   estimate method=ml;
run;

proc hpfesmspec repository=work.mymodels name=b;
   esm method=winters;
run;
```

```
proc hpfucmspec repository=work.mymodels name=c;
   forecast symbol=y;
   irregular;
   level;
   slope;
   season length=12;
run;
proc hpfselect repository=work.mymodels
               name=myselect;
   select criterion=mape;
   spec a b c;
run;
```

The following statements display all classes of XML specifications in the model repository, as indicated by the PRINT=ALL option in the PROC HPFREPOSITORY statement:

```
proc hpfrepository repository=work.mymodels print=all;
run;
```

**Figure 14.1** WORK.MYMODELS Model Selection Lists

| Selection Lists |
|---|
| Selection Name |
| MYSELECT |

**Figure 14.2** WORK.MYMODELS Model Specifications

| Model Specifications | |
|---|---|
| Spec Name | Type |
| A | ARIMA |
| B | ESM |
| C | UCM |

# Syntax: HPFREPOSITORY Procedure

The following statements are used with the HPFREPOSITORY procedure:

**PROC HPFREPOSITORY** *options* ;
**EVENTS** *event-list* ;
**SPECIFICATIONS** *spec-list* ;
**VARIABLES** *var-list* ;
**TRAVERSE** *spec-list* ;

## Functional Summary

Table 14.1 summarizes the statements and options that control the HPFREPOSITORY procedure.

**Table 14.1** PROC HPFREPOSITORY Functional Summary

| Description | Statement | Option |
|---|---|---|
| **Model Repository Options** | | |
| Specifies the model repository | PROC HPFREPOSITORY | REPOSITORY= |
| **Filtering Statements** | | |
| Specifies event mappings to target | EVENTS | |
| Specifies model specifications to target | SPECIFICATIONS | |
| Specifies variable mappings to target | VARIABLES | |
| **Traversal Statements** | | |
| Specifies list specifications to be traversed | TRAVERSE | |

## PROC HPFREPOSITORY Statement

**PROC HPFREPOSITORY** *options* **;**

The following options can be used in the PROC HPFREPOSITORY statement:

**REPOSITORY=***SAS-catalog-name*
names the SAS catalog or directory to contain the model specification. The REPOSITORY= option can also be specified as MODELREPOSITORY=, MODELREP=, or REP=.

**PRINT=***print-opt* | **(***print-opts***)**
specifies the printed output desired.

The following printing options are available:

ALL is the same as specifying PRINT=(SELECTIONS COMBINATIONS SPECIFICATIONS).

COMBINATIONS prints the Combinations ODS table.

LISTS is the same as specifying PRINT=(SELECTIONS COMBINATIONS).

SCORES prints the Scores ODS table.

SELECTIONS prints the Selections ODS table.

SPECIFICATIONS prints the Specifications ODS table.

**PRINTDETAILS**
specifies that output that is requested with the PRINT= option be printed in greater detail. The meaning of this option depends on each of the output types in the PRINT= option. Subsequent examples show the types of information added to the respective ODS tables when you specify this option.

## EVENTS Statement

**EVENTS** *event-list* **;**

The EVENTS statement defines a set of event names of interest for possible references from other XML specifications. Any model selection list or combined model list that contains an event map entry that references one of these event names is included in the requested output. Lists that are included for output must satisfy all three sets of selection criteria defined by the SPECIFICATIONS, EVENTS, and VARIABLES statements that are specified in the procedure statement block.

The following two invocations of the HPFREPOSITORY procedure produce the same results:

```
proc hpfrepository repository=work.LevModRep print=selections;
   events x1;
   events x2;
   events x3;
run;

proc hpfrepository repository=work.LevModRep print=selections;
   events x1 x2 x3;
run;
```

## SPECIFICATIONS Statement

**SPECIFICATIONS** *spec-list* **;**

**SPECS** *spec-list* **;**

The SPECIFICATIONS statement defines a set of XML specification names of interest for possible references from other list specifications. Any model selection list or combined model list that contains a reference to one of these specification names is included in the requested output. Lists that are included for output must satisfy all three sets of selection criteria defined by the SPECIFICATIONS, EVENTS, and VARIABLES statements that are specified in the procedure statement block.

The following two invocations of the HPFREPOSITORY procedure produce the same results:

```
proc hpfrepository repository=work.LevModRep print=selections;
   specifications airline;
   specifications smwint;
   specifications smseas;
run;

proc hpfrepository repository=work.LevModRep print=selections;
   specifications airline smwint smseas;
run;
```

## TRAVERSE Statement

**TRAVERSE** *spec-list trav-opts* **;**

The TRAVERSE statement defines a set of XML specification names that you want to display. For each XML specification in *spec-list*, the forecast model selection graph that originates from the XML specification is displayed in the ODS Traversal table with the from/to path listed in the respective columns of the table. You can use the /PREORDER and /POSTORDER options to control the order of the display of the specifications in the Traversal table.

The TRAVERSE statement supports the following values for *trav-opts*:

**/PREORDER**
specifies that the information for each specification in the forecast model selection graph be displayed in the Traversal table before the information for each of the specifications that it references.

**/POSTORDER**
specifies that the information for each specification in the forecast model selection graph be displayed in the Traversal table after the information for each of the specifications that it references.

## VARIABLES Statement

**VARIABLES** *var-list* **;**

**VARS** *var-list* **;**

The VARIABLES statement defines a set of variable names of interest for possible references from other XML specifications. Any model selection list or combined model list that contains a variable map entry that references one of these variable names is included in the requested output. Lists that are included for output must satisfy all three sets of selection criteria defined by the SPECIFICATIONS, EVENTS, and VARIABLES statements that are specified in the procedure statement block.

The following two invocations of the HPFREPOSITORY procedure produce the same results:

```
proc hpfrepository repository=work.LevModRep print=selections;
   variables x1;
   variables x2;
   variables x3;
run;

proc hpfrepository repository=work.LevModRep print=selections;
   variables x1 x2 x3;
run;
```

## ODS Table Names

**Table 14.2** ODS Tables Produced in PROC HPFREPOSITORY

| ODS Table Name | Description | Printed By |
|---|---|---|
| Selections | Model selection lists | PRINT=SELECTIONS |
| Specifications | Model specifications | PRINT=SPECIFICATIONS |
| Combinations | Combined model lists | PRINT=COMBINATIONS |
| Scores | Scored forecasts | PRINT=SCORES |
| Traversal | Traversed lists | TRAVERSE statement |

# Examples: HPFREPOSITORY Procedure

## Example 14.1: Standard Use

This example shows a number of variations of HPFREPOSITORY procedure PRINT= options and how they interact with the SPECIFICATIONS, VARIABLES, and EVENTS statements to affect desired output. The addition of PRINTDETAILS includes information for the creation and modification dates for the XML specifications and the descriptive system label for each XML specification.

This set of query examples creates a repository named WORK.TSFS that contains XML specifications defined by the following SAS statements:

```
proc hpfidmspec modelrepository=work.tsfs name=myidmspec;
run;

proc hpfexmspec modelrepository=work.tsfs name=myextspec;
run;

proc hpfselect modelrepository=work.tsfs name=myselect;
   spec airline;
run;

proc hpfucmspec modelrepository=work.tsfs name=myucm;
   forecast symbol=y;
   irregular;
   level;
   slope;
   season length=12;
run;

proc hpfselect modelrepository=work.tsfs name=myselect;
   spec myucm;
run;

proc hpfselect modelrepository=work.tsfs name=inputselect;
   spec fakespec1;
   spec fakespec4 / inputmap(symbol=x1 var=var2);
run;

proc hpfselect modelrepository=work.tsfs name=eventselect;
   spec fakespec2 / eventmap(symbol=x1 event=event1);
   spec fakespec4 / eventmap(symbol=x1 event=event2);
run;

proc hpfselect modelrepository=work.tsfs name=inputeventsselect;
   spec fakespec3 / inputmap(symbol=x1 var=var1);
   spec fakespec4 / inputmap(symbol=x1 var=var2);
   spec fakespec5 / inputmap(symbol=x2 var=var3) eventmap(symbol=x3 event=event3);
run;
```

```
proc hpfselect modelrepository=work.tsfs name=combinelist;
   combine method=average;
   spec fakespec1;
   spec fakespec2 fakespec3 / inputmap(symbol=x1 var=var1);
   spec fakespec4 / inputmap(symbol=x1 var=var2);
   spec fakespec5 / inputmap(symbol=x2 var=var3) eventmap(symbol=x3 event=event3);
run;
```

The following statements display the model selection lists from the model repository WORK.TSFS in the ODS Selections table:

```
proc hpfrepository repository=work.tsfs print=selections printdetails;
run;
```

The PRINTDETAILS option adds the following columns to the Selections table:

- the Description column, which displays the system label generated for the listed XML specification
- the Created column, which displays the creation date for the listed XML specification
- the Modified column, which displays the modification date for the listed XML specification

**Output 14.1.1** ODS Selection Table Results

| Selection Lists | | | |
|---|---|---|---|
| Selection Name | Description | Created | Modified |
| EVENTSELECT | SELECT: FAKESPEC2 FAKESPEC4 | 26MAR2014:08:26:02 | 26MAR2014:08:26:02 |
| INPUTEVENTSSELECT | SELECT: FAKESPEC3 FAKESPEC4 FAKESPEC5 | 26MAR2014:08:26:03 | 26MAR2014:08:26:03 |
| INPUTSELECT | SELECT: FAKESPEC1 FAKESPEC4 | 26MAR2014:08:26:02 | 26MAR2014:08:26:02 |
| MYSELECT | SELECT: MYUCM | 26MAR2014:08:26:02 | 26MAR2014:08:26:02 |

The following statements display the XML specifications from the model repository WORK.TSFS in the ODS Specifications table:

```
proc hpfrepository repository=work.tsfs print=specifications printdetails;
run;
```

The PRINTDETAILS option adds the following columns to the Specifications table:

- the Description column, which displays the system label generated for the listed XML specification
- the Created column, which displays the creation date for the listed XML specification
- the Modified column, which displays the modification date for the listed XML specification

**Output 14.1.2** ODS Specifications Table Results

| Model Specifications | | | |
|---|---|---|---|
| **Spec Name** | **Type** | **Created** | **Modified** |
| MYEXTSPEC | EXT | 26MAR2014:08:26:02 | 26MAR2014:08:26:02 |
| MYIDMSPEC | IDM | 26MAR2014:08:26:02 | 26MAR2014:08:26:02 |
| MYUCM | UCM | 26MAR2014:08:26:02 | 26MAR2014:08:26:02 |

The following statements display the combined model lists from the model repository WORK.TSFS in the ODS Combinations table:

```
proc hpfrepository repository=work.tsfs print=combinations printdetails;
run;
```

The PRINTDETAILS option adds the following columns to the Combinations table:

- the Description column, which displays the system label generated for the listed XML specification
- the Created column, which displays the creation date for the listed XML specification
- the Modified column, which displays the modification date for the listed XML specification

**Output 14.1.3** ODS Combinations Table Results

| Combination Lists | | | |
|---|---|---|---|
| **Combination Name** | **Description** | **Created** | **Modified** |
| COMBINELIST | COMBINE: FAKESPEC1 FAKESPEC2 FAKESPEC3 FAKESPEC4 FAKESPEC5 | 26MAR2014:08:26:03 | 26MAR2014:08:26:03 |

The following statements display the model selection lists from the model repository WORK.TSFS in the ODS Selections table that reference any of the XML specifications FAKESPEC1, FAKESPEC2, and FAKESPEC3:

```
proc hpfrepository repository=work.tsfs print=selections;
   specifications fakespec1;
   specifications fakespec2;
   specifications fakespec3;
run;
```

**Output 14.1.4** ODS Selections Table Results with SPECIFICATIONS Statements

| Selection Lists |
|---|
| **Selection Name** |
| EVENTSELECT |
| INPUTEVENTSSELECT |
| INPUTSELECT |

The following statements display the model selection lists from the model repository WORK.TSFS in the ODS Selections table that reference either of the event definitions EVENT1 or EVENT2:

```
proc hpfrepository repository=work.tsfs print=selections;
   events event1 event2;
run;
```

**Output 14.1.5** ODS Selections Table Results with EVENTS Statements

| Selection Lists |
|---|
| Selection Name |
| EVENTSELECT |

The following statements display the model selection lists from the model repository WORK.TSFS in the ODS Selections table that reference either of the variables VAR1 or VAR3:

```
proc hpfrepository repository=work.tsfs print=selections;
   variables var1 var3;
run;
```

**Output 14.1.6** ODS Selections Table Results with VARIABLES Statements

| Selection Lists |
|---|
| Selection Name |
| INPUTEVENTSSELECT |

## Example 14.2: Using the TRAVERSE Statement

This example demonstrates the use of the TRAVERSE statement to display the forecast model selection graph structure that is generated from a list specification.

The following statements create a repository named WORK.REP. It contains a UCM model specification named T1, an ESM model specification named T2, an ARIMA model specification named T3, a combined model list named COMB3, and a model selection list named SELECT.

```
proc hpfucmspec rep=work.rep specname=t1;
   level;
   slope;
   irregular;
   season type=dummy length=s;
run;

proc hpfesmspec rep=work.rep specname=t2;
   esm method=bests;
run;

proc hpfarimaspec rep=work.rep specname=t3;
  forecast symbol=y dif=(1,s) q=(1)(1)s noint;
  estimate method=ml converge=.0001 delta=.0001 maxiter=150;
run;
```

```
proc hpfselect rep=work.rep name=comb3 label='Average(T1,T2,T3)';
  combine method=average;
  spec t1 t2 t3;
run;

proc hpfselect rep=work.rep name=select;
   spec comb3;
run;
```

The following statements generate the ODS Traversal table for the model selection list named SELECT in the default display order. This is equivalent to specifying the /POSTORDER option in the TRAVERSE statement. With this display order, the information for each specification in the forecast model selection graph is displayed after the information for each of the specifications that it references.

```
proc hpfrepository repository=work.rep printdetails;
   traverse select;
run;
```

**Output 14.2.1** Traversal Results for List SELECT (/POSTORDER)

| | | Traversal Set | | | |
|---|---|---|---|---|---|
| Root Spec Name | From Spec Name | Spec Name | Spec Type | Level | Label |
| SELECT | COMB3 | T1 | UCM | 1 | |
| SELECT | COMB3 | T2 | ESM | 1 | Best Seasonal Smoothing Method |
| SELECT | COMB3 | T3 | ARIMA | 1 | ARIMA: Y ~ D = (1,s) Q = ((1)(1)s) NOINT |
| SELECT | SELECT | COMB3 | COMBINED | 0 | Average(T1,T2,T3) |

The following statements use /PREORDER to generate the ODS Traversal table for the model selection list named SELECT. With this display order, the information for each specification in the forecast model selection graph is displayed before all of the specifications that it references.

```
proc hpfrepository repository=work.rep printdetails;
   traverse select/preorder;
run;
```

**Output 14.2.2** Traversal Results for List SELECT (/PREORDER)

| | | Traversal Set | | | |
|---|---|---|---|---|---|
| Root Spec Name | From Spec Name | Spec Name | Spec Type | Level | Label |
| SELECT | SELECT | COMB3 | COMBINED | 0 | Average(T1,T2,T3) |
| SELECT | COMB3 | T1 | UCM | 1 | |
| SELECT | COMB3 | T2 | ESM | 1 | Best Seasonal Smoothing Method |
| SELECT | COMB3 | T3 | ARIMA | 1 | ARIMA: Y ~ D = (1,s) Q = ((1)(1)s) NOINT |

# Chapter 15
# The HPFSELECT Procedure

**Contents**


Overview: HPFSELECT Procedure . . . . . **459**
Getting Started: HPFSELECT Procedure . . . . . **460**
Syntax: HPFSELECT Procedure . . . . . **462**
Functional Summary . . . . . 462
PROC HPFSELECT Statement . . . . . 463
COMBINE Statement . . . . . 464
DELETE Statement . . . . . 468
DIAGNOSE Statement . . . . . 469
FORECASTOPTIONS Statement . . . . . 470
SELECT Statement . . . . . 470
SPECIFICATION Statement . . . . . 471
Valid Statistic of Fit Names . . . . . 473
Examples: HPFSELECT Procedure . . . . . **475**
Example 15.1: The INPUTMAP Option . . . . . 475
Example 15.2: The EVENTMAP Option . . . . . 476
Example 15.3: The DIAGNOSE Statement . . . . . 478
Example 15.4: External Models and User-Defined Subroutines . . . . . 480
Example 15.5: Comparing Forecasts from Multiple External Sources . . . . . 481
Example 15.6: Input to User-Defined Subroutines . . . . . 483
Example 15.7: Changing an Existing Selection List . . . . . 485
Example 15.8: Using a Combined Forecast . . . . . 485


## Overview: HPFSELECT Procedure

The HPFSELECT procedure enables you to control the forecasting process by defining lists of candidate models, or more generally candidate specifications, to be evaluated by PROC HPFENGINE. These lists are defined in terms of time series models, model selection lists, or model combination lists. Abstractly, each list created by the HPFSELECT procedure is a list of specifications. Using lists created by the HPFSELECT procedure, you can control which forecasting model or models SAS Forecast Server Procedures software uses to forecast particular time series.

The HPFSELECT procedure creates a list specification file for each invocation and stores it in a repository for later use by the HPFENGINE procedure. Each list specification has semantics defined by the type of list you create. These semantics determine the behavior of the list when it is executed in the HPFENGINE procedure.

Each of these list files references specifications previously created and stored in a model repository by the HPFARIMASPEC, HPFESMSPEC, HPFEXMSPEC, HPFIDMSPEC, HPFUCMSPEC, or HPFSELECT procedures.

Abstractly, you should think of each specification in the list as producing a forecast when applied to a particular time series. The behavior of the list as defined by its type determines how those forecasts are used by the list and ultimately the forecast produced by the list itself.

The HPFSELECT procedure supports these types of lists:

Selection list
selects the best forecast of those produced by its list based on a specified statistic of fit criterion. Selection lists are also referred to as model selection lists. Selection lists are defined by the presence of the SELECT statement in the HPFSELECT procedure statement block. This is also the default list type when no other type-related statement is specified. You can also specify other options that control the forecasting model selection process such as the use of holdout samples to evaluate candidate forecast performance and various diagnostics applied to the target time series to characterize it for model selection.

Combination list
combines a subset of the forecasts produced by its list via a weighted average. Combination lists are specified via the presence of a COMBINE statement in the HPFSELECT procedure statement block. The COMBINE statement supports a variety of options to affect candidate forecast selection, to specify the method for determining the weights assigned to the forecasts in combination, and to control various other aspects of the final combined forecast results.

Further discussion of these concepts can be found in Chapter 21, "Forecast Model Selection Graph Details," for how they relate and interact in operation of the automated forecasting process in the context of the HPFENGINE procedure.

## Getting Started: HPFSELECT Procedure

The following example shows how to create a model selection list file. Suppose the model repository MYLIB.MYMODELS contains three model specification files (A.XML, B.XML, C.XML) created by the following SAS statements.

```
proc hpfarimaspec repository=mymodels name=a;
   forecast symbol=y p=12 diflist=(1 12) noint;
   estimate method=ml;
run;

proc hpfesmspec repository=mymodels name=b;
   esm method=winters;
run;
```

```
proc hpfucmspec repository=mymodels name=c;
   forecast symbol=y;
   irregular;
   level;
   slope;
   season length=12;
run;
```

The following statements create a model selection list that will tell the HPFENGINE procedure to automatically select from the A, B, and C models based on the mean absolute percentage error (MAPE).

```
proc hpfselect repository=mymodels
               name=myselect;
   spec a b c;
   select criterion=mape;
run;
```

The options in the PROC HPFSELECT statement specify the name and location of the model selection file that is created. The REPOSITORY= option specifies that the output file be placed in the catalog MYLIB.MYMODELS, and the NAME= option specifies that the name of the file be "myselect.xml." The SPEC statement specifies the list of candidate models. The SELECT statement specifies options that control how the HPFENGINE procedure selects from the candidate models when applying the selection list MYSELECT to actual time series data.

The new selection list is available for use by the HPFENGINE procedure, shown in the following SAS statements. The GLOBALSELECTION= option refers to the selection list by name. Selection results are shown in Figure 15.1.

```
proc hpfengine data=sashelp.air
               repository=mymodels
               globalselection=myselect
               print=select
               out=_null_;
   id date interval=month;
   forecast air;
run;
```

**Figure 15.1** Selection Results

**The HPFENGINE Procedure**

| Model Selection Criterion = MAPE | | | |
|---|---|---|---|
| **Model** | **Statistic** | **Selected** | **Label** |
| **A** | 3.1008419 | No | ARIMA: Y ~ P = 12 D = (1,12) NOINT |
| **B** | 3.0845016 | Yes | Winters Method (Multiplicative) |
| **C** | 4.3672632 | No | UCM: Y = TREND + SEASON + ERROR |

# Syntax: HPFSELECT Procedure

The following statements are used with the HPFSELECT procedure:

**PROC HPFSELECT** *options* ;
  **COMBINE** *options* ;
  **DELETE** *specification-list* ;
  **DIAGNOSE** *options* ;
  **FORECASTOPTIONS** *options* ;
  **SELECT** *options* ;
  **SPECIFICATION** *specification-list* < / *options* > ;

## Functional Summary

Table 15.1 summarizes statements and options that control the HPFSELECT procedure.

**Table 15.1** HPFSELECT Functional Summary

| Description | Statement | Option |
|---|---|---|
| **Statements** | | |
| Specifies the forecasting options | FORECASTOPTIONS | |
| **Model Repository Options** | | |
| Specifies the model repository | PROC HPFSELECT | REPOSITORY= |
| Specifies the model specification name | PROC HPFSELECT | NAME= |
| Specifies the model specification label | PROC HPFSELECT | LABEL= |
| **Forecasting Options** | | |
| Specifies the confidence limit width | FORECASTOPTIONS | ALPHA= |
| **Forecast Combination Options** | | |
| Specifies the combination weight method | COMBINE | METHOD= |
| Specifies the encompassing test | COMBINE | ENCOMPASS= |
| Specifies the forecast combination criterion | COMBINE | CRITERION= |
| Specifies the percentage of missing forecast values | COMBINE | MISSPERCENT= |
| Specifies the percentage of missing horizon values | COMBINE | HORMISSPERCENT= |
| Specifies the combination of missing forecast values | COMBINE | MISSMODE= |
| **Forecast Selection Options** | | |
| Specifies the forecast holdout sample size | SELECT | HOLDOUT= |
| Specifies the forecast holdout sample size as a percentage | SELECT | HOLDOUTPCT= |

**Table 15.1** *continued*

| Description | Statement | Option |
|---|---|---|
| Specifies the model selection criterion | SELECT | CRITERION= |
| **Model Specification Options** | | |
| Associates a symbol with data set variable | SPECIFICATION | INPUTMAP |
| Associates a symbol with an event | SPECIFICATION | EVENTAP |
| Associates external data with the specification | SPECIFICATION | EXMMAP |
| Associates an external subroutine with the specification | SPECIFICATION | EXMFUNC |
| Overrides specification labels | SPECIFICATION | LABEL |
| Validates model specifications | PROC HPFSELECT | VALIDATE |
| **Model Selection List Options** | | |
| Removes specifications from the list | DELETE | |
| Specifies an input selection list | PROC HPFSELECT | INSELECTNAME= |
| **Diagnostic Options** | | |
| Specifies the base value for an intermittent series | DIAGNOSE | IDMBASE= |
| Specifies the intermittency test threshold | DIAGNOSE | INTERMITTENT= |
| Specifies the seasonality test | DIAGNOSE | SEASONTEST= |

## PROC HPFSELECT Statement

**PROC HPFSELECT** *options* **;**

The following options can be used in the PROC HPFSELECT statement.

**INSELECTNAME=***SAS-catalog-name*
provides a selection list as input to the HPFSELECT procedure. The input selection list is specified as a three-level name. The INSELECTNAME= option can be used to add models to existing lists, remove models from existing lists, change selection options, and so forth.

**LABEL=***SAS-label*
specifies a descriptive label for the model selection to be stored in the SAS catalog or directory. The LABEL= option can also be specified as SELECTLABEL=.

**NAME=***SAS-name*
names the model selection file to be stored in the SAS catalog or directory. The NAME= option can also be specified as SELECTNAME=.

**REPOSITORY=***SAS-catalog-name* | *SAS-file-reference*
names the SAS catalog or directory to contain the model specification. The REPOSITORY= option can also be specified as MODELREPOSITORY=, MODELREP=, or REP=.

**VALIDATE**
checks the validity of lists used by and generated by the HPFSELECT procedure invocation. VALIDATE covers both the list in the INSELECTNAME= option, if specified, and the list generated by the HPFSELECT procedure statement block. Validation ensures that all of the specifications in the respective forecast model selection graphs exist, that they are valid HPF XML model or list specifications, and that no cycles are created in the selection graph. A failure to validate the INSELECTNAME= results in the INSELECTNAME= being ignored. Undefined repository entries in the HPFSELECT procedure statement block SPECIFICATION statements are omitted from the generated list. A failure to validate the list generated from the HPFSELECT procedure statement block produces no new XML list specification entry in the model repository. Specifications required to satisfy the references in the INSELECTNAME= list are not automatically copied to the model repository identified by the REPOSITORY= option. This can be one common cause of validation problems. For more details about the forecast model selection graph, see Chapter 21, "Forecast Model Selection Graph Details."

## COMBINE Statement

**COMBINE** *comb-options* **;**

The COMBINE statement defines combination semantics for the list of specifications that are defined in the SPECIFICATION statements.

The following examples illustrate typical uses of the COMBINE statement:

```
proc hpfselect rep=work.rep
     name=combavg
     label="Avergage(T1,T2,T3)";

combine method=average;
spec t1 t2 t3;
run;

proc hpfselect rep=work.rep
     name=combrwgt
     label="RankWeight(T1,T2,T3)";

combine method=rankwgt(0.5,0.3,0.2);
spec t1 t2 t3 / inputmap(symbol=x data=temp);
run;
```

The following *comb-options* can be specified:

**CRITERION=***option*
specifies the forecast combination criterion (statistic of fit) to be used when ranking forecast candidates in the context of the COMBINE statement. This option is often used in conjunction with the ENCOMPASS= and METHOD=RANKWGT options. If not specified, the default is determined by the HPFENGINE procedure. See the section "Valid Statistic of Fit Names" on page 473 for valid values.

**ENCOMPASS=NONE**

**ENCOMPASS=***test-name***(***test-options***)**
specifies whether a forecast encompassing test be performed, and if so which type of test. The encompassing test attempts to eliminate forecasts from consideration that fail to add significant information to the final forecast. The default is ENCOMPASS=NONE, which specifies that no encompassing test be performed.

You can specify the following values for *test-name* and *test-options*:

OLS(ALPHA=*number*) uses an OLS-based regression test to estimate pairwise encompassing between candidate forecasts. Candidates are ranked from best to worst using the CRITERION= values. Iterating from best to worst, inferior candidates are tested with the best of the untested candidates for retention in the combined set. The significance level for the test is given by specifying ALPHA=*number* option. The default value is ALPHA=0.05 when the simple form ENCOMPASS=OLS is specified. The range is 0 to 1.

HLN(ALPHA=*number*) uses the Harvey-Leybourne-Newbold (HLN) test to estimate pairwise encompassing between candidate forecasts. Candidates are ranked from best to worst using the CRITERION= values. Iterating from best to worst, inferior candidates are tested with the best of the untested candidates for retention in the combined set. The significance level for the test is given by specifying ALPHA=*number* option. The default value is ALPHA=0.05 when the simple form ENCOMPASS=HLN is specified. The range is 0 to 1.

**HORMISSPERCENT=***number*
specifies a threshold for the percentage of missing forecast values in the combination horizon used to exclude a candidate forecast from consideration in the final combination. By default, no horizon missing percentage test is performed on candidate forecasts. If specified, the admissible range is 1 to 100. The forecast horizon is the region of time in which multistep forecasts are generated. This test and the MISSPERCENT test operate independent of each other. One or both can be specified.

**METHOD=***weight-method***(***method-options***)**
specifies the method for determining the combination weights used in the weighted average of the candidate forecasts in the combination list. The default method is METHOD=AVERAGE. The simple form METHOD=*weight-method* can be used when no weight-specific options are desired.

The following values for *weight-method* and *method-options* can be specified:

AICC(*AICC-opts*) computes the combination weights based on corrected AIC weights. See Chapter 20, "Forecast Combination Computational Details," for the mathematical details of this process. Frequently there is considerable disparity between the weights due to the exponential weighting scheme, so options are allowed to affect the scaling and to cull low-scoring candidates from consideration for computational efficiency. By default, all AICC scored candidate forecasts are combined.

Possible values for *AICC-opts* include:

ABSWGT=*number* omits computed weights with values less than the specified value. The range is 0 to 1 inclusive. The remaining weights are normalized to sum to 1.

BESTPCT=*number* retains the best $N$ of the candidates as a percentage of the total number weighted, where

$$N = \max\{\lfloor \frac{number \cdot M}{100} \rfloor, 1\} \tag{15.1}$$

and $M$ denotes the number of candidate models in the combination after any specified forecast exclusion tests have been performed.

The $N$ remaining weights are normalized to sum to 1.

BESTN=*N* retains the best $N$ of the candidates as a percentage of the total number weighted. The $N$ remaining weights are normalized to sum to 1.

LAMBDA=*number* specifies the scale factor used in the computation of the AICC weights. The default is LAMBDA=1.0, which results in the usual Akaike weights.

AVERAGE computes the simple average of the forecasts selected for combination. This is the default.

ERLS(*NLP-opts*) computes the combination weights based on a constrained least squares problem to minimize the $\ell_2$ norm of the combined forecast residuals, subject to the constraint that the weights sum to 1.

LAD(*LAD-opts*) computes the weights based on a least absolute deviations measure of fit for the combined forecast. A linear program is formulated according to the *LAD-opts* to minimize an objective function expressed in terms of a absolute values of a loss series, subject to constraints that the weights sum to 1 and be nonnegative. Options permitted in *LAD-opts* include OBJTYPE and ERRTYPE.

The form of the objective can be specified by the OBJTYPE= option as:

OBJTYPE=L1 specifies that the objective is an $\ell_1$ norm involving the loss series.

OBJTYPE=LINF specifies that the objective is an $\ell_\infty$ norm involving the loss series.

The form of the loss series in the objective can be specified as:

ERRTYPE=ABS specifies that the loss series terms are deviations.

ERRTYPE=APE specifies that the loss series terms are percentage deviations.

ERRTYPE=RAE specifies that the loss series terms are relative error deviations.

NERLS(*NLP-opts*) computes the combination weights based on a constrained least squares problem to minimize the $\ell_2$ norm of the combined forecast residuals, subject to the constraints that the weights sum to 1 and be nonnegative.

NRLS(*NLP-opts*) computes the combination weights based on a constrained least squares problem to minimize the $\ell_2$ norm of the combined forecast residuals, subject to the constraints that the weights be nonnegative.

OLS computes the combination weights that result from the ordinary least squares problem to minimize the $\ell_2$ norm of the combined forecast residuals.

RANKWGT*(W1,...,Wn)* assigns weights using the rank of the candidate forecasts at the time the combination is performed as determined by the COMBINE statement CRITERION= values. These weights must sum to 1. If not, they are normalized and a warning is issued. The number of values specified must agree with the number of specification names declared in the SPECIFICATION statements in the PROC HPFSELECT statement block. The weights are assigned by ranking the candidate forecasts from best to worst. The best uses the first weight, W1, and so on. The set of weights used is normalized to account for candidates that fail to forecast or for candidates that are omitted from the final combination because of any of the COMBINE statement exclusion tests.

RMSEWGT computes the combination weights based on the RMSE statistic of fit for the forecast contributors. The weights are normalized to sum to 1. See Chapter 20, "Forecast Combination Computational Details," for details.

USERDEF*(W1,...,Wn)* assigns weights using the list of user-specified values. These weights must sum to 1. If not, they are normalized and a warning is issued. The number of values specified must agree with the number of specification names declared in the SPECIFICATION statements in the PROC HPFSELECT statement block. The weights correspond with the order of the names in the SPECIFICATION statements. The set of weights used is normalized to account for candidates that fail to forecast or for candidates that are omitted from the final combination because of any of the COMBINE statement exclusion tests.

**MISSMODE=***miss-method*

specifies a method for treating missing values in the forecast combination. In a given time slice across the combination ensemble, one or more combination contributors can have a missing value. This setting determines the treatment of those in the final combination for such time indices.

The following *miss-method* values are available:

RESCALE rescales the combination weights for the nonmissing contributors at each time index to sum to 1.

MISSING generates a missing combined forecast at each time index with one or more missing contributors.

The default behavior is determined by the weight method selected as follows:

MISSMODE=RESCALE is the default for simple average, user-specified weights, ranked user weights, ranked weights, and RMSE weights.

MISSMODE=MISSING is the default for AICC weights, OLS weights, restricted least squares weights, and LAD weights. For OLS and NRLS you cannot specify MISSMODE=RESCALE since the estimated weights are not constrained to sum to one.

**MISSPERCENT=***number*

specifies a threshold for the percentage of missing forecast values in the combination estimation region that is used to exclude a candidate forecast from consideration in the final combination. By default, no missing percentage test is performed on candidate forecasts. If specified, the admissible range is 1 to 100. This test and the HORMISSPERCENT test operate independent of each other. One or both can be specified.

**STDERR=***stderr-method***(***stderr-options***)**

**SEMODE=***stderr-method***(***stderr-options***)**

specifies the method for computing the prediction error variance series. This series is used to compute the prediction standard error, which in turn is used to compute confidence bands on the combined forecast.

The simple form STDERR=*stderr-method* can be used when no method-specific options are desired.

The following values for *stderr-method* and *stderr-options* can be specified:

STDERR=DIAG computes the prediction error variance by assuming the forecast errors at time $t$ are uncorrelated so that the simple diagonal form of $\Sigma_t$ is used. This is the default method for computing prediction error variance.

STDERR=ESTCORR computes prediction error variance by using estimates of $\rho_{i,j,t}$, the sample cross-correlation between $e_{i,t}$ and $e_{j,t}$ over the time span $t = 1, \ldots, T$, where $T$ denotes the last time index of the actual series $y_t$. This option implies, of course, that the error series $e_{i,t}$ and $e_{j,t}$ are assumed to be jointly stationary.

STDERR=ESTCORR(TAU=$\tau$) is similar to STDERR=ESTCORR except that the cross-correlation estimates are localized to a time window of $\tau$ steps. The time span $t = 1, \ldots, T$ is quantized into segments of $\tau$ steps working from $T$ backwards for in-sample cross-correlation estimates. The cross-correlation estimates from the interval $[T - \tau, T]$ are used for the period of multistep forecasts that extend beyond time $T$.

Computational details for combined forecasts can be found in Chapter 20, "Forecast Combination Computational Details."

## DELETE Statement

**DELETE** *specification-list* **;**

The DELETE statement is used to remove model specifications from a selection list. There can be any number of model specifications listed in a DELETE statement and any number of DELETE statements.

## DIAGNOSE Statement

**DIAGNOSE** *options* **;**

The DIAGNOSE statement is used to specify diagnostic options. DIAGNOSE options are used by the HPFENGINE procedure to subset a model selection list according to certain properties of a time series.

The following examples illustrate typical uses of the DIAGNOSE statement:

```
/* same as default options */
diagnose intermittent=2.0 seasontest=(siglevel=0.01);

/* no seasonality */
diagnose seasontest=(siglevel=0);
```

**IDMBASE=AUTO** | *number*

specifies the base value of the time series used to determine the demand series components for an intermittent demand model. The demand series components are determined based on the departures from this base value. If a base value is specified, this value is used to determine the demand series components. If IDMBASE=AUTO is specified, the time series properties are used to automatically adjust the time series. For the common definition of Croston's method use IDMBASE=0, which defines departures based on zero. The default is IDMBASE=AUTO.

Given a time series $y_t$ and base value $b$ the time series is adjusted by the base value to create the base-adjusted time series, $x_t = y_t - b$. Demands are assumed to occur when the base-adjusted series is nonzero (or when the time series $y_t$ departs from the base value $b$).

When IDMBASE=AUTO, the base value is automatically determined by the time series median, minimum, and maximum values and the INTERMITTENT= option value.

**INTERMITTENT=***number*

specifies a number greater than one that is used to determine whether or not a time series is intermittent. If the average demand interval is greater than this number, then the series is assumed to be intermittent. The default is INTERMITTENT=2.0.

**SEASONTEST=***option*

specifies the options related to the seasonality test.

The following values for the SEASONTEST= options are allowed:

NONE No test

(SIGLEVEL=number) Significance probability value to use in testing whether seasonality is present in the time series. The value must be between 0 and 1.

A smaller value of the SIGLEVEL= option means that stronger evidence of a seasonal pattern in the data is required before the HPFENGINE procedure will use seasonal models to forecast the time series. The default is SEASONTEST=(SIGLEVEL=0.01).

## FORECASTOPTIONS Statement

**FORECASTOPTIONS** *options* **;**

The FORECASTOPTIONS statement is used to specify forecasting options.

**ALPHA=***number*
specifies the significance level to use in computing the confidence limits of the forecast. The value of ALPHA= must be between 0 and 1. The default is ALPHA=0.05, which produces 95% confidence intervals.

## SELECT Statement

**SELECT** *options* **;**

The SELECT statement is used to specify forecast selection semantics for the list of specifications in the SPECIFICATION statements.

The following examples illustrate typical uses of the SELECT statement:

```
/* same as default options */
select criterion=rmse holdout=0 holdoutpct=0;

/* selection criterion mape with absolute holdout size 6 */
select criterion=mape holdout=6;
```

**CHOOSE=***specification*
specifies the name of a model specification that will be chosen by the HPFENGINE procedure. By default, HPFENGINE will select the model with the best fit in terms of the statistic set by the CRITERION= option. The CHOOSE= option overrides this automatic selection and causes HPFENGINE to generate forecasts by using the model indicated in the option.

**CRITERION=***option*
specifies the model selection criterion (statistic of fit) to be used to select from several candidate models. This option is often used in conjunction with the HOLDOUT= option. If not specified, the default is determined by the HPFENGINE procedure. See "Valid Statistic of Fit Names" on page 473 for the list of valid values for the CRITERION= option.

**HOLDOUT=***n*
specifies the size of the holdout sample to be used for model selection. The holdout sample is a subset of actual time series that ends at the last nonmissing observation. The default is zero (no holdout sample).

**HOLDOUTPCT=***number*
specifies the size of the holdout sample as a percentage of the length of the time series. If HOLDOUT=5 and HOLDOUTPCT=10, the size of the holdout sample is $min(5, 0.1T)$, where $T$ is the length of the time series with beginning and ending missing values removed. The default is 100 (100%), which means no restriction on the holdout sample size based on the series length.

## SPECIFICATION Statement

**SPECIFICATION** *specification-list* < / *options* > **;**

The SPECIFICATION statement is used to list model specifications. There can be any number of models specifications in the list and any number of SPECIFICATION statements. The SPECIFICATION statement can also be written as SPEC.

The following options can be used with the SPECIFICATION statement.

**EVENTMAP ( *symbol*= _NONE_ EVENT= *eventDef* < NODIFF > )**

**EVENTMAP ( *symbol*= *string* EVENT= *eventDef* )**

associates events with a model specification.

If SYMBOL=_NONE_ is used, the event specified in *eventDef* is added to the model as a simple regressor. By default, for an ARIMA model, any differencing applied to the dependent variable is applied in the same manner to the new event input. Specifying NODIFF indicates no differencing should be performed on the event.

If the SYMBOL string matches one of the symbols specified as input in either a UCM or ARIMA model, the event data will be used for the matching input. In this manner, events can enter models through complex transfer functions. The NODIFF option does not apply in this case, since differencing will be explicitly described in the input statement of the model.

If the event referenced by *eventDef* is a "predefined event" (see the HPFEVENT procedure), no INEVENT= option is required for the HPFENGINE procedure. Otherwise, the event named must be defined in an event data set by using the HPFEVENT procedure, and that data set must be given to the HPFENGINE procedure with the INEVENT= option. An error will occur during the HPFENGINE procedure model selection if *eventDef* is not found in an event data set.

Only UCM and ARIMA models are valid when EVENTMAP is used. If another model type, such as exponential smoothing, has an EVENTMAP option, the option is simply ignored.

**EXMMAP***(options)*

associates an external model specification with variable names in a DATA= data set, thus identifying the source of forecasts and optionally prediction standard errors, and the lower and upper confidence limits.

Available options are as follows:

*PREDICT=var*

identifies the variable to supply forecasts and is required.

*STDERR=var*

identifies the variable to supply the prediction standard error.

*LOWER=var*

identifies the variable to supply the lower confidence limit.

*UPPER=var*

identifies the variable to supply the upper confidence limit.

For example, if you want an external model "myexm" to use forecasts from the variable "yhat" in the DATA= data set passed to the HPFENGINE procedure, the appropriate statement would be as follows:

```
spec myexm / exmmap(predict=yhat);
```

If you also want to use prediction standard errors from the "std" variable in the same data set, use the following statement:

```
spec myexm / exmmap(predict=yhat stderr=std);
```

**EXMFUNC***(string)*

associates an external model specification with a user-defined subroutine. The string parameter is the signature of the subroutine and has the following form:

*subroutineName*( *parameters* )

where parameters indicate the order, number, and type of arguments in the user-defined subroutine. The parameter _PREDICT_ is required, indicating the return of forecast values. Optional parameters for return of other data from the user-defined subroutine include the following:

| | |
|---|---|
| _STDERR_ | prediction standard error |
| _LOWER_ | lower confidence limit |
| _UPPER_ | upper confidence limit |

The HPFENGINE procedure can also pass data into the user-defined subroutine by using the following options:

| | |
|---|---|
| _TIMEID_ | time ID |
| _SEASON_ | seasonal index |
| _ACTUAL_ | actual values |

As an example, suppose the signature of a user-defined subroutine is defined in the FCMP procedure as follows:

```
subroutine userdef1(act[*], pred[*]);
```

Also suppose this subroutine is mapped to the external model "myexm" by using the following statement:

```
spec myexm / exmfunc('userdef1(_actual_ _predict_ )');
```

Then the HPFENGINE procedure will pass the array of actuals to the subroutine userdef1, and the function will compute the forecasts and return them to the HPFENGINE procedure.

Next, consider the case where a user-defined subroutine requires actuals, time ID values, and seasonal indices in order to compute and return forecasts and prediction standard errors. The subroutine might be defined as follows:

```
subroutine complexsub(act[*], timeid[*],
                      seasons[*], pred[*], stderr[*]);
```

It would be mapped to an external model named “myexm” by using the following statement:

```
spec myexm / exmfunc(
     'complexsub(_actual_ _timeid_ _season_ _predict_ _stderr_)'
                  );
```

This syntax is demonstrated further in Example 15.4.

**INPUTMAP (SYMBOL=** *string* **VAR=** *variable***)**
associates the symbols in a model specification with variable names in a DATA= data set.

The SYMBOL= option should match a symbol defined in a model specification. The VAR= option should match a variable name in a data set. When the model selection list is used in conjunction with the HPFENGINE procedure, the DATA= option specifies the input data set that contains the variable.

Mappings are needed because model specifications are generic. For example, suppose a model specification associates the symbol Y with the dependent variable. If you want to use this model to forecast the variable OZONE, you map Y to OZONE with INPUTMAP(SYMBOL=Y VAR=OZONE). If you later want to use the same specification to forecast SALES, you map Y to SALES with INPUTMAP(SYMBOL=Y VAR=SALES).

The INPUTMAP option is not required. By default, the HPFENGINE procedure attempts to locate variables in its DATA= data set that match the symbols in each specification listed in the selection list.

**LABEL=***SAS-label*
overrides the label in the model specification and causes the new label to print in the model selection list in the HPFENGINE procedure. This option is useful if you have the same model specification listed more than once and want to distinguish between them in the HPFENGINE procedure output.

As an example, consider the following case of a single external model used twice in the selection list, with each occurrence mapped to a different external forecast:

```
spec myexm / exmmap(predict=yhatRALEIGH)
             label="External Model: Raleigh Forecasts";
spec myexm / exmmap(predict=yhatATLANTA)
             label="External Model: Atlanta Forecasts"
```

In the model selection list output from the HPFENGINE procedure, the new labels appear, rather than the label in “myexm” repeated twice.

## Valid Statistic of Fit Names

Following is the list of valid values for specifying a desired statistic of fit:

| | |
|---|---|
| SSE | sum of squares error |
| MSE | mean squared error |
| RMSE | root mean squared error |
| UMSE | unbiased mean squared error |

| | |
|---|---|
| URMSE | unbiased root mean squared error |
| MAXPE | maximum percent error |
| MINPE | minimum percent error |
| MPE | mean percent error |
| MAPE | mean absolute percent error |
| MDAPE | median absolute percent error |
| GMAPE | geometric mean absolute percent error |
| MINPPE | minimum predictive percent error |
| MAXPPE | maximum predictive percent error |
| MPPE | mean predictive percent error |
| MAPPE | symmetric mean absolute predictive percent error |
| MDAPPE | median absolute predictive percent error |
| GMAPPE | geometric mean absolute predictive percent error |
| MINSPE | minimum symmetric percent error |
| MAXSPE | maximum symmetric percent error |
| MSPE | mean symmetric percent error |
| SMAPE | symmetric mean absolute percent error |
| MDASPE | median absolute symmetric percent error |
| GMASPE | geometric mean absolute symmetric percent error |
| MINRE | minimum relative error |
| MAXRE | maximum relative error |
| MRE | mean relative error |
| MRAE | mean relative absolute error |
| MDRAE | median relative absolute error |
| GMRAE | geometric mean relative absolute error |
| MAXERR | maximum error |
| MINERR | minimum error |
| ME | mean error |
| MAE | mean absolute error |
| MASE | mean absolute scaled error |
| RSQUARE | R-square |
| ADJRSQ | adjusted R-square |
| AADJRSQ | Amemiya’s adjusted R-square |
| RWRSQ | random walk R-square |
| AIC | Akaike information criterion |

| | |
|---|---|
| AICC | finite sample corrected AIC |
| SBC | Schwarz Bayesian information criterion |
| APC | Amemiya's prediction criterion |
| MAPES | mean absolute error percent of standard deviation |
| MDAPES | median absolute error percent of standard deviation |
| GMAPES | geometric mean absolute error percent of standard deviation |

# Examples: HPFSELECT Procedure

## Example 15.1: The INPUTMAP Option

In this example, the HPFUCMSPEC procedure is used to define a UCM specification. The dependent variable is assigned the symbol Y, and an input is assigned the symbol X1. You ultimately want to forecast the series contained in the variable MASONRY with the input ELECTRIC. As demonstrated in the following SAS statements, the INPUTMAP option in the HPFSELECT procedure is used to tell the HPFENGINE procedure that MASONRY should replace Y and ELECTRIC should replace X1.

```
proc hpfucmspec repository=mymodels
                name=myucm
                label="My UCM spec";
   dependent symbol=Y;
   irregular;
   level;
   slope;
   season length=12;
   input symbol=X1;
run;

proc hpfselect repository=mymodels
               name=myselect;
   spec myucm /
   inputmap(symbol=Y var=MASONRY)
   inputmap(symbol=X1 var=ELECTRIC);
run;
```

The following call to the HPFENGINE procedure creates forecasts by using the UCM model with correct variable mappings. The model selection table that is created is shown in Output 15.1.1.

```
proc hpfengine data=sashelp.workers
               out=_null_
               repository=mymodels
               globalselection=myselect
               print=select;
   id date interval=month;
   forecast masonry;
   stochastic electric;
run;
```

**Output 15.1.1** Selection Results

**The HPFENGINE Procedure**

| Model Selection Criterion = MAPE | | | |
|---|---|---|---|
| **Model** | **Statistic** | **Selected** | **Label** |
| **MYUCM** | 2.0744810 | Yes | My UCM spec |

As shown in the following SAS statements, the same result could be achieved by making the symbol in the model specification match the variables in the HPFENGINE procedure's DATA= data set. No INPUTMAP option is required.

```
proc hpfucmspec repository=mymodels
                name=myucm
                label="My UCM spec";
   dependent symbol=MASONRY;
   irregular;
   level;
   slope;
   season length=12;
   input symbol=ELECTRIC;
run;

proc hpfselect repository=mymodels
               name=myselect
               label="My Selection List";
   spec myucm;
run;
```

The disadvantage here is that the model specification and data set are tightly linked.

## Example 15.2: The EVENTMAP Option

Events are dynamically added as simple regressors to UCM and ARIMA models by using the EVENTMAP option in the SPECIFICATIONS statement.

In this example, you first create an ARIMA model and then create a selection list that directs the HPFENGINE procedure to choose between this model without an event and this model with the event. The following SAS statements illustrate this process. The results are shown in Output 15.2.1.

```
proc hpfevents data=sashelp.air;
   eventdef summer = (june july august);
   eventdata out=eventDB;
run;

proc hpfarimaspec repository=sasuser.repository name=arima;
   forecast symbol=air q=(1 12) transform=log;
run;
```

```
proc hpfselect repository=sasuser.repository name=select;
   spec arima;
   spec arima / eventmap(symbol=_none_ event=summer);
run;

proc hpfengine data=sashelp.air
               repository=sasuser.repository
               outest=outest1
               globalselection=select
               print=(select estimates)
               inevent=eventDB;
   id date interval=month;
   forecast air;
run;
```

**Output 15.2.1** Selection and Estimation Results

**The HPFENGINE Procedure**

**Model Selection Criterion = MAPE**

| Model | Statistic | Selected | Label |
|---|---|---|---|
| **ARIMA** | 20.048903 | No | ARIMA: Log( AIR ) ~ Q = (1,12) |
| **ARIMA** | 19.654381 | Yes | ARIMA: Log( AIR ) ~ Q = (1,12) |

**Parameter Estimates for ARIMA Model**

| Component | Parameter | Estimate | Standard Error | t Value | Approx Pr > \|t\| |
|---|---|---|---|---|---|
| **AIR** | **CONSTANT** | 5.47071 | 0.04202 | 130.21 | <.0001 |
| **AIR** | **MA1_1** | -0.80089 | 0.02337 | -34.27 | <.0001 |
| **AIR** | **MA1_12** | -0.27008 | 0.02374 | -11.37 | <.0001 |
| **SUMMER** | **SCALE** | 0.15504 | 0.05859 | 2.65 | 0.0091 |

In the following statements, you calculate the same results but this time explicitly create a model that includes the input in its specification. Then the event is added to the data set.

```
data air(keep=date air summer);
   set sashelp.air;
   summer = 0;
   if month(date) eq 6 or
      month(date) eq 7 or
      month(date) eq 8 then summer = 1;
run;

proc hpfarimaspec repository=sasuser.repository name=arimasummer;
   forecast symbol=air q=(1 12) transform=log;
   input symbol=summer;
run;

proc hpfselect repository=sasuser.repository name=select;
   spec arima arimasummer;
run;
```

Note that the results from the following PROC HPFENGINE run, shown in Output 15.2.2, are the same as the results of the simpler usage with the EVENTMAP option.

```
proc hpfengine data=air
               repository=sasuser.repository
               outest=outest2
               globalselection=select
               print=(select estimates);
   id date interval=month;
   forecast air;
   input summer;
run;
```

**Output 15.2.2** Selection and Estimation Results

**The HPFENGINE Procedure**

| Model Selection Criterion = MAPE | | | |
|---|---|---|---|
| **Model** | **Statistic** | **Selected** | **Label** |
| **ARIMA** | 20.048903 | No | ARIMA: Log( AIR ) ~ Q = (1,12) |
| **ARIMASUMMER** | 19.654381 | Yes | ARIMA: Log( AIR ) ~ Q = (1,12) + INPUT: SUMMER |

| Parameter Estimates for ARIMASUMMER Model | | | | | |
|---|---|---|---|---|---|
| **Component** | **Parameter** | **Estimate** | **Standard Error** | **t Value** | **Approx Pr > \|t\|** |
| **AIR** | **CONSTANT** | 5.47071 | 0.04202 | 130.21 | <.0001 |
| **AIR** | **MA1_1** | -0.80089 | 0.02337 | -34.27 | <.0001 |
| **AIR** | **MA1_12** | -0.27008 | 0.02374 | -11.37 | <.0001 |
| **summer** | **SCALE** | 0.15504 | 0.05859 | 2.65 | 0.0091 |

## Example 15.3: The DIAGNOSE Statement

The DIAGNOSE statement enables control of the diagnostics used by the HPFENGINE procedure to subset the model selection list. Two ESM model specifications are created in this example: one seasonal and one nonseasonal. They are placed together in a selection list, with the seasonality test value allowed to remain at its default in this first case. The diagnostics in the HPFENGINE procedure judge the dependent series as seasonal and therefore exclude the nonseasonal model from consideration. The following statements illustrate the behavior with the explicit use of the DIAGNOSE statement. The results are shown in Output 15.3.1.

```
proc hpfesmspec repository=sasuser.repository name=logdouble;
   esm method=double transform=log;
run;

proc hpfesmspec repository=sasuser.repository name=logwinters;
   esm method=winters transform=log;
run;

proc hpfselect repository=sasuser.repository name=select;
   spec logdouble logwinters;
run;
```

```
proc hpfengine data=sashelp.air
               repository=sasuser.repository
               outest=outest1
               globalselection=select
               print=all;
   id date interval=month;
   forecast air;
run;
```

**Output 15.3.1** Seasonality Test at Significance Level of 0.01

**The HPFENGINE Procedure**

| Model Selection Criterion = MAPE | | | |
|---|---|---|---|
| **Model** | **Statistic** | **Selected** | **Label** |
| **LOGDOUBLE** | . | Removed | Log Double Exponential Smoothing |
| **LOGWINTERS** | 2.7138783 | Yes | Log Winters Method (Multiplicative) |

In the following statements, the same two exponential smoothing models are used in a selection list again, but this time the seasonality test is disabled. Both models are fit to the series as a result. The results are shown in Output 15.3.2.

```
proc hpfselect repository=sasuser.repository name=select;
   spec logdouble logwinters;
   diagnose seasontest=none;
run;

proc hpfengine data=sashelp.air
               repository=sasuser.repository
               outest=outest1
               globalselection=select print=select;
   id date interval=month;
   forecast air;
run;
```

**Output 15.3.2** No Seasonality Test Performed

**The HPFENGINE Procedure**

| Model Selection Criterion = MAPE | | | |
|---|---|---|---|
| **Model** | **Statistic** | **Selected** | **Label** |
| **LOGDOUBLE** | 10.167638 | No | Log Double Exponential Smoothing |
| **LOGWINTERS** | 2.713878 | Yes | Log Winters Method (Multiplicative) |

Finally, in the following statements, the seasonality test significance level is set to zero so that the series will not be judged as seasonal. In Output 15.3.3, note that the nonseasonal model is fit to the series, but the seasonal model is removed.

```
proc hpfselect repository=sasuser.repository name=select;
   spec logdouble logwinters;
   diagnose seasontest=(siglevel=0);
run;
```

```
proc hpfengine data=sashelp.air
               repository=sasuser.repository
               outest=outest1
               globalselection=select
               print=select;
   id date interval=month;
   forecast air;
run;
```

**Output 15.3.3** All Series Treated as Nonseasonal

**The HPFENGINE Procedure**

| Model Selection Criterion = MAPE | | | |
|---|---|---|---|
| **Model** | **Statistic** | **Selected** | **Label** |
| **LOGDOUBLE** | 10.167638 | Yes | Log Double Exponential Smoothing |
| **LOGWINTERS** | . | Removed | Log Winters Method (Multiplicative) |

## Example 15.4: External Models and User-Defined Subroutines

The EXMFUNC option enables you to associate an external model specification with a user-defined subroutine. In this example, you define a user subroutine and store it in a catalog. The following statements create a simple three-point moving average.

```
proc fcmp outlib=work.hpfengine.funcs;
   subroutine move_avg3(act[*], pred[*]);
      outargs pred;
      actlen = DIM(act);
      predlen = DIM(pred);
      pred[1] = 0;
      pred[2] = act[1]/3.0;
      pred[3] = (act[1] + act[2])/3.0;
      do i=4 to actlen+1;
         pred[i] = (act[i-1] + act[i-2] + act[i-3])/3.0;
      end;
      do i=actlen+2 to predlen;
         pred[i] = (pred[i-1] + pred[i-2] + pred[i-3])/3.0;
      end;
   endsub;
run;

options cmplib=work.hpfengine;
```

Now, just for comparison, you use the HPFARIMASPEC procedure to make a model specification that will produce the same three-point moving average.

```
proc hpfarimaspec repository=sasuser.repository name=arima;
   forecast symbol=y noint p=3
            ar=(0.333333333 0.333333333 0.333333333);
   estimate noest;
run;
```

Next, you use the HPFEXMSPEC procedure to create an external model specification and the HPFSELECT procedure to make a selection list with the external model, referring to the user-defined subroutine, and the ARIMA model.

```
proc hpfexmspec repository=sasuser.repository name=myexm;
   exm;
run;

proc hpfselect repository=sasuser.repository name=select;
   diagnose seasontest=none;
   spec arima;
   spec myexm / exmfunc('move_avg3(_actual_ _predict_ )')
                label="External Model from move_avg3";
run;
```

Finally, you use the HPFENGINE procedure to forecast a series by using the selection list just created. As expected, both models produce the same forecast, as indicated by the same selection fit statistic.

```
proc hpfengine data=sashelp.air
               out=_null_
               repository=sasuser.repository
               globalselection=select
               print=select;
   id date interval=month;
   forecast air;
run;
```

The output is shown in Output 15.4.1.

**Output 15.4.1** External Model with User-Defined Subroutine versus ARIMA Model

**The HPFENGINE Procedure**

| Model Selection Criterion = MAPE | | | |
|---|---|---|---|
| **Model** | **Statistic** | **Selected** | **Label** |
| **ARIMA** | 13.455362 | No | ARIMA: Y ~ P = 3 NOINT |
| **MYEXM** | 13.455362 | Yes | External Model from move_avg3 |

## Example 15.5: Comparing Forecasts from Multiple External Sources

The EXMMAP option enables you to associate an external model specification with variables in a data set. This example uses the EXPAND procedure and the ARIMA procedure to generate data for the external forecasts. In practice these external data might come from judgmental forecasts or other systems. The external forecasts must be in the same data set as the series you are modeling. The following statements generate external data.

```
data temp;
   set sashelp.air;
   drop i;
   * introduce some missing for EXPAND to fill in;
   if mod(_n_, 5) eq 0 then air = .;
   if mod(_n_+1, 5) eq 0 then air = .;
```

```
      if mod(_n_+2, 5) eq 0 then air = .;
      output;

      if date eq '01dec1960'd then do;
         do i=1 to 4;
            date = intnx('month', '01dec1960'd, i);
            air = .;
            output;
         end;
      end;
   run;

proc expand data=temp extrapolate
            out=expandout;
   id date;
   convert air=interp;
run;

data temp;
   set sashelp.air;
   logair = log(air);
run;

proc arima data=temp;
   identify var=logair(1,12) noprint;
   estimate q=(1)(12) noconstant method=ml noprint;
   forecast out=arimaout(rename=(forecast=airline))
            lead=4 id=date
            interval=month noprint;
run;

data arimaout;
   set arimaout;
   airline = exp(airline);
run;

data temp;
   keep date interp airline air;
   merge expandout arimaout sashelp.air;
   by date;
run;
```

Next, you create a smoothing model to add to the selection, demonstrating that you can compare multiple external forecasts not only with one another but also with other statistical models. After the models are defined, they are added to a selection list. Notice that the same external model can be associated with different external forecasts and that the LABEL= option can be used to help differentiate between the two. Finally, the HPFENGINE procedure is called, and you see that the forecast originally generated by the ARIMA procedure best fits the historical data.

```
proc hpfesmspec repository=sasuser.repository name=esm;
   esm method=seasonal transform=auto;
run;

proc hpfexmspec repository=sasuser.repository name=exm;
```

```
   exm;
run;

proc hpfselect repository=sasuser.repository name=select;
   spec esm;
   spec exm / exmmap(predict=interp)
              label="Interpolation from PROC EXPAND";
   spec exm / exmmap(predict=airline)
              label="Forecasts from PROC ARIMA";
run;

proc hpfengine data=temp
               out=_null_
               repository=sasuser.repository
               globalselection=select
               lead=4
               print=select;
   id date interval=month;
   forecast air;
   external interp airline;
run;
```

The output is shown in Output 15.5.1.

**Output 15.5.1** Two External Forecasts and a Smoothing Model

**The HPFENGINE Procedure**

| Model Selection Criterion = MAPE | | | |
|---|---|---|---|
| **Model** | **Statistic** | **Selected** | **Label** |
| **ESM** | 3.1909590 | No | Log Seasonal Exponential Smoothing |
| **EXM** | 5.5023045 | No | Interpolation from PROC EXPAND |
| **EXM** | 2.9239173 | Yes | Forecasts from PROC ARIMA |

## Example 15.6: Input to User-Defined Subroutines

This example illustrates the different types of data that the HPFENGINE procedure can pass to a user-defined subroutine. Consider the following subroutine definition.

```
proc fcmp outlib=work.hpfengine.funcs;
   subroutine testsub(timeid[*], act[*], seasons[*],  pred[*]);
      outargs pred;
      actlen = DIM(act);
      predlen = DIM(pred);
      format date monyy.;

      * print the input;
      do i=6 to 18;
         date = timeid[i]; actual=act[i]; season=seasons[i];
         put i= date= actual= season=;
      end;

      * just return mean;
```

```
      mean = 0.0;
      do i=1 to actlen;
         mean = mean + act[i];
      end;
      mean = mean / actlen;

      do i=1 to predlen;
         pred[i] = mean;
      end;

   endsub;
run;
```

Suppose you have an external model specification and invocation of the HPFSELECT procedure such as the following.

```
proc hpfexmspec repository=sasuser.repository name=myexm1;
   exm;
run;

proc hpfselect repository=sasuser.repository name=select;
   diagnose seasontest=none;
   spec myexm1 / exmfunc('testsub(_timeid_ _actual_ _season_ _predict_)');
run;
```

Then the following call to the HPFENGINE procedure invokes the user-defined subroutine named TESTSUB.

```
options cmplib = work.hpfengine;
proc hpfengine data=sashelp.air
               out=_null_
               outfor=outfor
               repository=sasuser.repository
               globalselection=select;
   id date interval=month;
   forecast air;
run;
```

A partial listing of the log output from the run of the HPFENGINE procedure follows. The seasonal cycle length is 12, and thus the zero-based seasonal index repeats 0 through 11. Though not used in this simple subroutine, all these data are available when you are computing forecasts.

```
i=6 date=JUN49 actual=135 season=5
i=7 date=JUL49 actual=148 season=6
i=8 date=AUG49 actual=148 season=7
i=9 date=SEP49 actual=136 season=8
i=10 date=OCT49 actual=119 season=9
i=11 date=NOV49 actual=104 season=10
i=12 date=DEC49 actual=118 season=11
i=13 date=JAN50 actual=115 season=0
i=14 date=FEB50 actual=126 season=1
i=15 date=MAR50 actual=141 season=2
i=16 date=APR50 actual=135 season=3
i=17 date=MAY50 actual=125 season=4
i=18 date=JUN50 actual=149 season=5
```

## Example 15.7: Changing an Existing Selection List

The following SAS statements delete three specifications from a copy of SASHELP.HPFDFLT.BEST and add a new user-defined specification. The resulting list is named is SASUSER.REPOSITORY.SELECT.

```
proc hpfarimaspec repository=sasuser.repository name=arima;
   forecast symbol=y q=(1 12) dif=(1 12) noint transform=log;
run;

proc hpfselect name=select repository=sasuser.repository
     inselectname=sashelp.hpfdflt.best;
   delete smsimp smdamp smwint;
   spec arima;
run;
```

The following CATNAME statement gives the HPFENGINE procedure access to the model specifications in both catalogs SASHELP.HPFDFLT and SASUSER.REPOSITORY. The HPFENGINE procedure produces the output shown in Output 15.7.1.

```
catname twocats (sashelp.hpfdflt sasuser.repository);

proc hpfengine data=sashelp.air repository=work.twocats
     globalselection=select print=select out=_null_;
   id date interval=month;
   forecast air;
run;
```

**Output 15.7.1** Modified Selection List

**The HPFENGINE Procedure**

| Model Selection Criterion = MAPE | | | |
|---|---|---|---|
| **Model** | **Statistic** | **Selected** | **Label** |
| **ARIMA** | 2.9672282 | Yes | ARIMA: Log( Y ) ~ D = (1,12) Q = (1,12) NOINT |
| **SMLIN** | . | Removed | Linear Exponential Smoothing |
| **SMADWN** | 3.5343216 | No | Winters Method (Additive) |
| **SMSEAS** | 3.5196140 | No | Seasonal Exponential Smoothing |

## Example 15.8: Using a Combined Forecast

In this example, the goal is to select between the forecasts from the classic ARIMA AIRLINE model and a combination of the forecasts from the AIRLINE model and the best seasonal exponential smoothing model (ESM). The following statements demonstrate the use of the COMBINE statement to define a model combination list, COMB2, and then use COMB2 in the context of a root selection list, SELECTCOMB, that is used to generate a forecast. SELECTCOMB is defined to select between the AIRLINE and COMB2 specifications by using forecast performance over a holdout sample of the last 6 months.

The COMBINE statement explicitly defines the method for determining the combination weights via the METHOD=AVERAGE option. In this case, omitting METHOD=AVERAGE produces the same result since that is the default weight method.

```
proc hpfarimaspec rep=work.rep specname=airline;
  forecast symbol=y dif=(1,s) q=(1)(1)s noint;
  estimate method=ml converge=.0001 delta=.0001 maxiter=150;
run;

proc hpfesmspec rep=work.rep name=bests;
  esm method=bests;
run;

proc hpfselect rep=work.rep name=comb2 label="AVG(AIRLINE,BESTS)";
  combine method=average;
  spec airline bests;
run;

proc hpfselect rep=work.rep name=selectcomb;
  select holdout=6;
  spec airline comb2;
run;
```

Run the selection list SELECTCOMB to choose between the better of the combined forecast and the AIRLINE model reference forecast:

```
proc hpfengine data=sashelp.air
    rep=work.rep globalselection=selectcomb
    out=out8 outfor=for8
    outmodelinfo=minfo8
    outstatselect=statsel8
    outstat=stat8
    outsum=sum8
    outcomponent=comp8
    lead=12 print=(all) plot=(components);
  id date interval=month;
  forecast air;
run;
```

From the model selection results in Output 15.8.1, the combination of the AIRLINE model forecast with the ESM forecast produces a better selection statistic of fit than the AIRLINE model alone.

**Output 15.8.1** Model Selection Results

**The HPFENGINE Procedure**

| Model Selection Criterion = MAPE | | | |
|---|---|---|---|
| **Model** | **Statistic** | **Selected** | **Label** |
| **AIRLINE** | 3.8095583 | No | ARIMA: Y ~ D = (1,s) Q = ((1)(1)s) NOINT |
| **COMB2** | 2.3574267 | Yes | AVG(AIRLINE,BESTS) |

Output 15.8.2 through Output 15.8.4 show the parameter estimates for all of the models that contribute to the combination in addition to those for the combined forecast proper.

**Output 15.8.2** Parameter Estimates for AIRLINE Model

**Parameter Estimates for AIRLINE Model**

| Component | Parameter | Estimate | Standard Error | t Value | Approx Pr > \|t\| |
|---|---|---|---|---|---|
| AIR | MA1_1 | 0.30868 | 0.08433 | 3.66 | 0.0003 |
| AIR | MA2_12 | 0.10744 | 0.10190 | 1.05 | 0.2917 |

**Output 15.8.3** Parameter Estimates for BESTS Model

**Parameter Estimates for BESTS Model**

| Component | Parameter | Estimate | Standard Error | t Value | Approx Pr > \|t\| |
|---|---|---|---|---|---|
| AIR | Level Weight | 0.30728 | 0.03153 | 9.74 | <.0001 |
| AIR | Trend Weight | 0.0010000 | 0.0030237 | 0.33 | 0.7413 |
| AIR | Seasonal Weight | 0.87493 | 0.07769 | 11.26 | <.0001 |

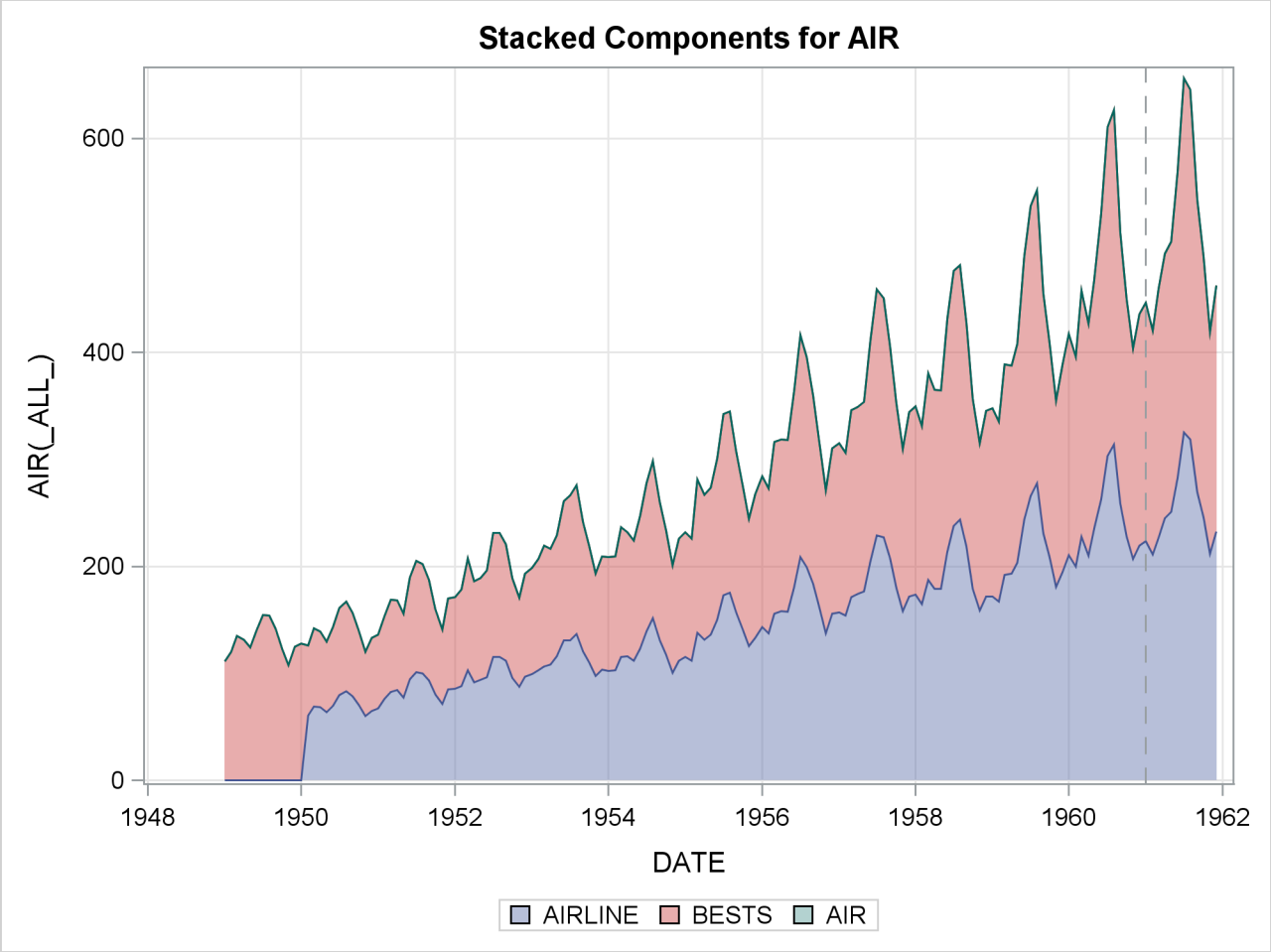
**Output 15.8.4** Parameter Estimates for COMB2 Model

**Parameter Estimates for COMB2 Model**

| Component | Parameter | Estimate | Standard Error | t Value | Approx Pr > \|t\| |
|---|---|---|---|---|---|
| AIRLINE | WEIGHT | 0.50000 | . | . | . |
| BESTS | WEIGHT | 0.50000 | . | . | . |

The weighted forecast components that contribute to the final combined forecast are plotted in a stack on the same set of axes. The weighted forecast components are plotted from bottom to top in the order of their combination to yield the final combined forecast.

**Output 15.8.5** Combined Forecast Components



# Chapter 16
# The HPFTEMPRECON Procedure

**Contents**


Overview: HPFTEMPRECON Procedure . . . . . **489**
Comparing the HPFTEMPRECON and HPFRECONCILE Procedures . . . . . 490
Getting Started: HPFTEMPRECON Procedure . . . . . **490**
Syntax: HPFTEMPRECON Procedure . . . . . **491**
Functional Summary . . . . . 491
PROC HPFTEMPRECON Statement . . . . . 493
BY Statement . . . . . 495
BENCHID Statement . . . . . 496
ID Statement . . . . . 498
RECONCILE Statement . . . . . 500
Details: HPFTEMPRECON Procedure . . . . . **501**
Mathematical Foundation . . . . . 501
Input and Output Data Sets . . . . . 502
Threading Details . . . . . 507
Examples: HPFTEMPRECON Procedure . . . . . **507**
Example 16.1: Reconciling Monthly Forecasts to Quarterly Forecasts . . . . . 507
Example 16.2: Reconciling Multiple Variables . . . . . 513
Example 16.3: Using Threads . . . . . 516
References . . . . . **519**


## Overview: HPFTEMPRECON Procedure

Forecasters often need to produce forecasts for a certain time series at more than one frequency. For example, a company that provides warranty repairs for appliances needs to forecast the number of daily calls for staffing and operational planning. The company also needs to forecast service calls at the monthly frequency to plan long-term expansion and financial planning such as the purchase of more vehicles or the hiring of new staff.

A common practice is to generate the forecasts at the two time intervals independently so as to choose the best model for each frequency. That can result in forecasts that do not agree, in the sense that the accumulation over time of the high-frequency forecasts does not equal the forecasts generated by the model for the low-frequency data.

The HPFTEMPRECON procedure reconciles the high-frequency forecasts to the low-frequency forecasts in such a way that the accumulation of the reconciled high-frequency forecasts is equal to the low-frequency forecasts.

Typically, the forecasts for the two frequencies that are input to the HPFTEMPRECON procedure are generated using the two runs of the HPFENGINE procedure with different values for the INTERVAL= options in the ID statement.

## Comparing the HPFTEMPRECON and HPFRECONCILE Procedures

The HPFTEMPRECON and the HPFRECONCILE procedures are related to each other in that both deal with reconciling two sets of forecasts that are generated independently. That is, they restore additivity under some form of aggregation in a hierarchy of forecasts. However, the hierarchies they deal with span different dimensions. The HPFRECONCILE procedure reconciles heterogeneous series to the aggregated parent in a hierarchy of forecasts for a given time interval that is equal for both the parent and the children series. For example, the parent node of the hierarchy is the given by the forecasts for the series of all electronic products, while the children nodes are the forecasts for the series of each element that compose the class of electronic products, such as TVs, DVD players, microwave ovens, and so on. For the HPFRECONCILE procedure the children and the parent forecasts need to be at the same frequency. The HPFTEMPRECON procedure, instead, reconciles forecasts for the same item at two different time frequencies whose intervals are nested in one another. In other words, it deals with a hierarchy of forecasts in the time dimension. For example, it reconciles monthly forecasts for the airline passenger data to the quarterly forecasts for the same series. For this reason, the HPFTEMPRECON procedure not only requires two input data sets, but also it requires that the two frequencies of the forecasts be specified in two separate statements: the ID statement for the high-frequency data, and the BENCHID statement for the low-frequency data.

Aggregation in the time dimension in which HPFTEMPRECON operates is henceforth referred to as *accumulation* to distinguish it from the aggregation across items in a hierarchy in which HPFRECONCILE operates.

# Getting Started: HPFTEMPRECON Procedure

In the following example, the monthly forecasts contained in the outformon data set are adjusted to match, when accumulated, the quarterly forecasts in the outforqtr data set. The monthly reconciled forecasts are saved in the recfor data set. The relative statistics of fit are saved in the recstat data set.

Procedures User's Guide,

```
proc hpftemprecon
     data=outformon
     benchdata=outforqtr
     outfor=recfor
     outstat=recstat;
   id date interval=month;
   benchid date interval=qtr;
run;
```

Note that the HPFTEMPRECON procedure requires two input data sets because it reconciles forecasts at two different frequencies. The DATA= option in the PROC HPFTEMPRECON statement specifies the data set whose PREDICT variable contains the forecasts for the high-frequency data. The BENCHDATA= option specifies the data set whose PREDICT variable contains the forecasts for the low-frequency data. Likewise,

there are two statements for specifying the time characteristics of the series. The ID statement specifies the variable in the DATA= data set that contains the time stamps and the interval frequency for the high-frequency data. The BENCHID statement specifies the variable in the BENCHDATA= data set that contains the time stamps and the interval frequency for the low-frequency data.

The PREDICT in the OUTFOR= data set contains the reconciled forecasts for the high-frequency data. The frequency of the reconciled forecasts is the same as that of the high-frequency data, as specified by the INTERVAL= option in the ID statement. When accumulated to the low frequency specified by the INTERVAL= option in the BENCHID statement, the reconciled forecasts are equal to the values of the PREDICT variable in the BENCHDATA= data set.

# Syntax: HPFTEMPRECON Procedure

The HPFTEMPRECON procedure is controlled by the following statements:

**PROC HPFTEMPRECON** < *options* > ;
**BY** *variables* < / *options* > ;
**BENCHID** *variable* **INTERVAL**=*interval* < / *options* > ;
**ID** *variable* **INTERVAL**=*interval* < / *options* > ;
**RECONCILE** *indicator* < =*benchmark* >< / *options* > ;

## Functional Summary

Table 16.1 summarizes the statements and options used with the HPFTEMPRECON procedure.

**Table 16.1** HPFTEMPRECON Functional Summary

| Description | Statement | Option |
|---|---|---|
| **Statements** | | |
| Specifies the BY variables | BY | |
| Specifies the time ID variable of the high-frequency series | ID | |
| Specifies the time ID variable of the low-frequency series | BENCHID | |
| Specifies custom variables names | RECONCILE | |
| **Time ID Options** | | |
| Specifies the alignment of time ID values | ID | ALIGN= |
| Specifies the ending time ID value | ID | END= |
| Specifies the date format | ID | FORMAT= |
| Specifies the frequency | ID | INTERVAL= |
| Specifies the starting time ID value | ID | START= |
| Specifies the missing value interpretation | ID | SETMISSING= |
| Specifies the zero value interpretation | ID | ZEROMISS= |

| Description | Statement | Option |
|---|---|---|
| Specifies the trim missing values | ID | TRIMMISS= |
| **Benchmark Time BENCHID Options** | | |
| Specifies the alignment of time BENCHID values | BENCHID | ALIGN= |
| Specifies the ending time BENCHID value | BENCHID | END= |
| Specifies the date format | BENCHID | FORMAT= |
| Specifies the frequency | BENCHID | INTERVAL= |
| Specifies the starting time BENCHID value | BENCHID | START= |
| Specifies the missing value interpretation | BENCHID | SETMISSING= |
| Specifies the zero value interpretation | BENCHID | ZEROMISS= |
| Specifies the trim missing values | BENCHID | TRIMMISS= |
| **Data Set Options** | | |
| Specifies the accumulation options for individual dependent variables | HPFTEMPRECON | ACCDATA= |
| Specifies the input data set of the low-frequency forecasts (benchmarks) | HPFTEMPRECON | BENCHDATA= |
| Specifies that the OUTFOR= data set contains the BENCHDIFF variable | HPFTEMPRECON | BENCHDIFF |
| Specifies the input data set of the high-frequency forecasts | HPFTEMPRECON | DATA= |
| Specifies the output data set to contain the reconciled forecasts | HPFTEMPRECON | OUTFOR= |
| Specifies the output data set to contain summary information | HPFTEMPRECON | OUTPROCINFO= |
| Specifies the output data set to contain statistics of fit. | HPFTEMPRECON | OUTSTAT= |
| **Analysis Options** | | |
| Specifies the type of relationship between the low-frequency and high-frequency data | HPFTEMPRECON | BENCHACCUMULATION= |
| Specifies whether the portion of the data in the DATA= data set that are not covered by benchmarks in the BENCHDATA= data set should be used in the reconciliation process | HPFTEMPRECON | BENCHCONS= |
| Specifies the bias correction | HPFTEMPRECON | BIASCORRECTION= |
| Specifies the maximum number of iterations of the quadratic solver | HPFTEMPRECON | MAXITER |
| Specifies that only the PREDICT variable be modified | HPFTEMPRECON | PREDICTONLY |
| Specifies the exponent of the scale factor | HPFTEMPRECON | SCALEEXP= |

| Description | Statement | Option |
|---|---|---|
| Specifies the value of the smoothing parameter | HPFTEMPRECON | SMOOTH= |
| Specifies the sign bound on the benchmarked series | HPFTEMPRECON | SIGN= |

## PROC HPFTEMPRECON Statement

**PROC HPFTEMPRECON** *options* **;**

The following *options* can be used in the PROC HPFTEMPRECON statement.

### Options Related to the Input Data Sets

**ACCDATA=***SAS-data-set*
specifies the name of the SAS data set that contains the BENCHACCUMULATION= option for each dependent variable listed in the _NAME_ variable. This option is useful when multiple FORECAST statements with different values of the ACCUMULATE= option are used in the HPFENGINE procedure.

See the section "ACCDATA= Data Set" on page 502 for more details about the ACCDATA= data set.

**BENCHDATA=***SAS-data-set*
specifies the name of the SAS data set that contains the low-frequency series (benchmarks).

See the section "BENCHDATA= Data Set" on page 503 for more details about the BENCHDATA= data set.

**DATA=***SAS-data-set*
specifies the name of the SAS data set that contains the high-frequency series. See the section "DATA= Data Set" on page 503 for more details about the DATA= data set.

### Options Related to the Output Data Sets

**BENCHDIFF**
specifies that the OUTFOR= data sets contain a variable named BENCHDIFF which is the difference between the reconciled predicted value and the input predicted value.

**OUTFOR=***SAS-data-set*
names the output SAS data set to contain the reconciled values.

See the section "OUTFOR= Data Set" on page 504 for more details about the OUTFOR= data set.

**OUTPROCINFO=***SAS-data-set*
names the output data set to contain the summary information of the processing done by PROC HPFTEMPRECON. When you write a program to assess the status of the procedure's execution, it is easier to parse a data set than it is to look at or parse the SAS log.

See the section "OUTPROCINFO= Data Set" on page 506 for more details about the OUTPROCINFO= data set.

**OUTSTAT=***SAS-data-set*
names the output data set to contain the statistics of fit (or goodness-of-fit statistics). The OUTSTAT= data set is useful for evaluating how well the model fits the series. The statistics of fit are based on the entire range of the time series. See the section "OUTSTAT= Data Set" on page 506 for more details about the OUTSTAT= data set.

## Options Related to Analysis

**BENCHACCUMULATION=***option*
specifies how the data of the high-frequency series is accumulated within each BENCHID time period of the low-frequency series.

The following *options* determine how the high-frequency time series observations are accumulated within each time period based on the BENCHID variable and the frequency specified by the BENCHINTERVAL= option:

| | |
|---|---|
| TOTAL | Observations are accumulated based on the total sum of their values. |
| AVERAGE \| AVG | Observations are accumulated based on the average of their values. |

See also the ACCDATA= option for an alternative way of specifying the BENCHACCUMULATION= option for individual dependent variables identified by the _NAME_ variable.

**BIASCORRECTION=NONE | ADDITIVE | MULTIPLICATIVE**
**BIAS=NONE | ADDITIVE | MULTIPLICATIVE**
specifies the type of bias correction to apply to the indicator series. The default is no bias correction (that is, BIASCORRECTION=NONE).

See the section "Mathematical Foundation" on page 501 for more details about the BIASCORRECTION= option.

**BENCHCONS=STRICT | NOSTRICT**
specifies whether the full range of the data in the DATA= data set or only the range covered by the data in BENCHDATA= should be used in the reconciliation process. The default is BENCHCONS=NOSTRICT, which corresponds to all data being used.

**MAXITER=***k*
specifies the maximum number of iterations performed by the quadratic solver. The value *k* is an integer between 1 and the largest four-byte, signed integer, $2^{31} - 1$. The default value is MAXITER=100.

**PREDICTONLY**
specifies that only the predicted value be modified. If the PREDICTONLY option is not specified, the confidence limits are centered around the new forecasts again.

**SCALEEXP=***n*

**EXP=***n*

specifies the value of the exponent of the scaling factor. *n* is a number between 0 and 1. The default is SCALEEXP=0.

See the section "Mathematical Foundation" on page 501 for more details about the SCALEEXP= option.

**SIGN=***option*

specifies the sign constraint on the reconciled series.

Valid values are as follows:

| | |
|---|---|
| MIXED | if the output series can have any sign. This is the default. |
| NONNEGATIVE \| POSITIVE | if the output series need to be nonnegative. |
| NONPOSITIVE \| NEGATIVE | if the output series need to be nonpositive. |

**SMOOTH=***n*

specifies the value of the autoregressive smoothing parameter. *n* is a number between 0 and 1. The default is SMOOTH=0.9.

See the section "Mathematical Foundation" on page 501 for more details about the SMOOTH= option.

## BY Statement

**BY** *variables* **;**

A BY statement can be used with PROC HPFTEMPRECON to obtain separate analyses for groups of observations defined by the BY variables.

When a BY statement appears, the procedure expects the input data set to be sorted in order of the BY variables.

If your input data set is not sorted in ascending order, use one of the following alternatives:

- Sort the data using the SORT procedure with a similar BY statement.
- Specify the BY statement option NOTSORTED or DESCENDING in the BY statement for the HPFTEMPRECON procedure. The NOTSORTED option does not mean that the data are unsorted but rather that the data are arranged in groups (according to values of the BY variables) and that these groups are not necessarily in alphabetical or increasing numeric order.
- Create an index on the BY variables using the DATASETS procedure.

For more information about the BY statement, see *SAS Language Reference: Concepts*. For more information about the DATASETS procedure, see the discussion in the *Base SAS Procedures Guide.*

## BENCHID Statement

**BENCHID** *variable* < *options* > **;**

The BENCHID statement names a numeric variable that identifies observations in the BENCHDATA= data set. The BENCHID variable's values are assumed to be SAS date, time, or datetime values. The BENCHID statement specifies the frequency associated with the low-frequency time series. The BENCHID statement *options* also specify how the BENCHID values are aligned to form the low-frequency time series. The INTERVAL= option must also be specified in the BENCHID statement. The time interval specified in the BENCHID option must be coarser than the time interval specified in the ID option.

The following *options* can be used with the BENCHID statement:

**ALIGN=***option*
controls the alignment of SAS dates used to identify output observations. The ALIGN= option accepts the following values: BEGINNING | BEG | B, MIDDLE | MID | M, and ENDING | END | E. The default is BEGINNING.

**END=***option*
specifies a SAS date, datetime, or time value that represents the end of the data. If the last time BENCHID variable value is less than the END= value, the series is extended with missing values. If the last time BENCHID variable value is greater than the END= value, the series is truncated. For example, END="&sysdate"D uses the automatic macro variable SYSDATE to extend or truncate the series to the current date.

**FORMAT=***option*
specifies a SAS format used for the DATE variable in the output data sets. The default format is the same as that of the DATE variable in the DATA= data set.

**INTERVAL=***interval*
specifies the frequency of the low-frequency time series. For example, if the BENCHDATA= data set consists of quarterly observations, then INTERVAL=QTR should be used. Only one observation per time interval is allowed.

The frequency specified by the INTERVAL= option in the BENCHID statement must be lower than the frequency specified by the INTERVAL option in the ID statement. Additionally, the time ID intervals must be fully nested in the time BENCHID intervals. For example, the ID statement can have INTERVAL=QTR and the BENCHID statement can have INTERVAL=YEAR, but INTERVAL=WEEK and INTERVAL=MONTH are not allowed because weekly intervals are not fully nested in monthly intervals, since a week can span over two months.

See Chapter 4, "Date Intervals, Formats, and Functions" (*SAS/ETS User's Guide*), for the intervals that can be specified.

**SETMISSING=***option* | *number*
specifies how missing values are assigned in the time series of low-frequency forecasts. If a number is specified, missing values are set to that number. If a missing value indicates an unknown value, this option should not be used. If a missing value indicates no value, a SETMISSING=0 should be used. You typically use SETMISSING=0 for transactional data because the absence of recorded data usually implies no activity. The following *options* can also be used to determine how missing values are assigned:

| | |
|---|---|
| MISSING | Missing values are set to missing. This is the default option. |
| AVERAGE \| AVG | Missing values are set to the average value. |
| MINIMUM \| MIN | Missing values are set to the minimum value. |
| MEDIAN \| MED | Missing values are set to the median value. |
| MAXIMUM \| MAX | Missing values are set to the maximum value. |
| FIRST | Missing values are set to the first nonmissing value. |
| LAST | Missing values are set to the last nonmissing value. |
| PREVIOUS \| PREV | Missing values are set to the previous nonmissing value. Missing values at the beginning of the series remain missing. |
| NEXT | Missing values are set to the next nonmissing value. Missing values at the end of the series remain missing. |

**START=***option*

specifies a SAS date, datetime, or time value that represents the beginning of the data. If the first time BENCHID variable value is greater than the START= value, missing values are added at the beginning of the series. If the first BENCHID variable value is less than the END= value, the series is truncated. This option and the END= option can be used to ensure that data associated with each BY group contain the same number of observations.

**TRIMMISS=***option*

specifies how missing values are trimmed from the time series of low-frequency forecasts. The following *options* are available:

| | |
|---|---|
| NONE | No missing values are trimmed. |
| LEFT | Beginning missing values are trimmed. |
| RIGHT | Ending missing values are trimmed. |
| BOTH | Both beginning and ending missing values are trimmed. This is the default. |

**ZEROMISS=***option*

specifies how beginning and ending zero values are interpreted in the time series of low-frequency forecasts. The following *options* can also be used to determine how beginning and ending zero values are assigned:

| | |
|---|---|
| NONE | Beginning and ending zeros are unchanged. This is the default. |
| LEFT | Beginning zeros are set to missing. |
| RIGHT | Ending zeros are set to missing. |
| BOTH | Both beginning and ending zeros are set to missing. |

If all of the series values are missing or zero, the series is not changed.

## ID Statement

**ID** *variable* < *options* > **;**

The ID statement names a numeric variable that identifies observations in the input and output data sets for the high-frequency series in the DATA= data set and the reconciled series in the OUTFOR= data set. The ID variable's values are assumed to be SAS date, time, or datetime values. The ID statement also specifies the frequency associated with the actual time series. The ID statement *options* specify how the ID values are aligned to form the actual time series. The INTERVAL= option must be specified in the ID statement. There can be only one observation in the DATA= data set associated with each time ID interval.

The following *options* can be used with the ID statement:

**ALIGN=***option*

controls the alignment of SAS dates used to identify output observations. The ALIGN= option accepts the following values: BEGINNING | BEG | B, MIDDLE | MID | M, and ENDING | END | E. The default is BEGINNING.

**END=***option*

specifies a SAS date, datetime, or time value that represents the end of the data. If the last time ID variable value is less than the END= value, the series is extended with missing values. If the last time ID variable value is greater than the END= value, the series is truncated. For example, END="&sysdate"D uses the automatic macro variable SYSDATE to extend or truncate the series to the current date.

**FORMAT=***option*

specifies a SAS format used for the DATE variable in the output data sets. The default format is the same as that of the DATE variable in the DATA= data set.

**INTERVAL=***interval*

specifies the frequency of the indicator time series. For example, if the input data set consists of quarterly observations, then INTERVAL=QTR should be used.

The frequency specified by the INTERVAL= option of the ID statement must be higher than the frequency specified by the BENCHINTERVAL option in the BENCHID statement. Additionally, the time ID intervals must be fully nested in the time BENCHID intervals. For example, the ID statement can have INTERVAL=QTR and the BENCHID statement can have INTERVAL=YEAR, but INTERVAL=WEEK and INTERVAL=MONTH are not allowed because weekly intervals are not fully nested in monthly intervals since one week can span over two months.

See Chapter 4, "Date Intervals, Formats, and Functions" (*SAS/ETS User's Guide*), for the intervals that can be specified.

**SETMISSING=***option* | *number*

specifies how missing values are assigned in the time series of high-frequency forecasts. If a *number* is specified, missing values are set to that number. If a missing value indicates an unknown value, this option should not be used. If a missing value indicates no value, a SETMISSING=0 should be used. You typically use SETMISSING=0 for transactional data because the absence of recorded data usually implies no activity. The following *options* can also be used to determine how missing values are assigned:

| | |
|---|---|
| MISSING | Missing values are set to missing. This is the default option. |
| AVERAGE \| AVG | Missing values are set to the average value. |
| MINIMUM \| MIN | Missing values are set to the minimum value. |
| MEDIAN \| MED | Missing values are set to the median value. |
| MAXIMUM \| MAX | Missing values are set to the maximum value. |
| FIRST | Missing values are set to the first nonmissing value. |
| LAST | Missing values are set to the last nonmissing value. |
| PREVIOUS \| PREV | Missing values are set to the previous nonmissing value. Missing values at the beginning of the series remain missing. |
| NEXT | Missing values are set to the next nonmissing value. Missing values at the end of the series remain missing. |

**START=***option*

specifies a SAS date, datetime, or time value that represents the beginning of the data. If the first time ID variable value is greater than the START= value, missing values are added at the beginning of the series. If the first time ID variable value is less than the END= value, the series is truncated. This option and the END= option can be used to ensure that data associated with each BY group contain the same number of observations.

**TRIMMISS=***option*

specifies how missing values are trimmed from the time series of high-frequency forecasts. The following options are provided:

| | |
|---|---|
| NONE | No missing values trimmed. |
| LEFT | Beginning missing values are trimmed. |
| RIGHT | Ending missing values are trimmed. |
| BOTH | Both beginning and ending missing values are trimmed. This is the default. |

**ZEROMISS=***option*

specifies how beginning and ending zero values are interpreted in the time series of high-frequency forecasts. The following *options* can also be used to determine how beginning and ending zero values are assigned:

| | |
|---|---|
| NONE | Beginning and ending zeros are unchanged. This is the default. |
| LEFT | Beginning zeros are set to missing. |
| RIGHT | Ending zeros are set to missing. |
| BOTH | Both beginning and ending zeros are set to missing. |

If all of the values in the accumulated series are missing or zero, the series is not changed.

## RECONCILE Statement

**RECONCILE** *indicator< =benchmark> </options>* **;**

The RECONCILE statement enables you to specify custom names for a group of variables to be reconciled and identify the role each of those variables plays in the reconciliation process. Multiple reconcile statements can be specified. Each RECONCILE statement specifies one variable in the DATA= data set to be reconciled, and, optionally, the corresponding variable in the BENCHDATA= data set that should be used for reconciling that particular variable. The *indicator* argument is required in the RECONCILE statement. It specifies the name of the variable in the DATA= data set that contains the high-frequency predicted values to be reconciled. The optional *=benchmark* argument specifies the name of the variable in the BENCHDATA= data set that contains the low-frequency predicted values used as benchmarks in the reconciliation process.

You can also specify in the RECONCILE statement the variables in the DATA= data set that should be associated with the variable to be reconciled. You identify the roles of these variables and the corresponding data set variable name through the *role=variable options*.

**ACTUAL=***variable-name*
specifies the name of the variable in the DATA= data set that contains the actual values.

**LOWER=***variable-name*
specifies the name of the variable in the DATA= data set that contains the lower confidence limit values.

**UPPER=***variable-name*
specifies the name of the variable in the DATA= data set that contains the upper confidence limit values.

**STD=***variable-name*
specifies the name of the variable in the DATA= data set that contains the standard error values.

When the RECONCILE statement is not specified, the default names are PREDICT for the indicator variable, PREDICT for the benchmark variable in the BENCHDATA= data set, ACTUAL for the actual values, LOWER for the lower confidence limit, UPPER for the upper confidence limit, and STD for the standard error.

See Example 16.2 for an example of how to use the RECONCILE statement when you want to reconcile multiple indicator variables in the same DATA= data set.

When you use PROC HPFTEMPRECON to reconcile forecasts with the OUTFOR= data set results from various SAS High Performance Forecasting procedures, it is not necessary, or even desirable, to use the RECONCILE statement. You only need to identify the DATA= and BENCHDATA= data sets and the corresponding time ID information for those data sets via ID and BENCHID statements, respectively. You can optionally specify a BY-clause in the PROC HPFTEMPRECON invocation if BY-group qualification was used when the original forecasts were produced.

In this context, it is important to remember that neither BENCHID nor ID support accumulation over duplicate time ID's in a given series. Thus for proper operation, it becomes important to specify the same BY-variable qualification in the PROC HPFTEMPRECON run as was used to construct the original low-frequency and high-frequency forecast data sets. This ensures seamless operation of PROC HPFTEMPRECON with the other SAS High Performance Forecasting procedures with a minimum of SAS code. For more information about accumulation, see the HPFTEMPRECON BENCHACCUMULATION option.

# Details: HPFTEMPRECON Procedure

## Mathematical Foundation

Forecasters often deal with data that are accumulated at different time intervals. For example, long-term forecasts are determined using yearly data, while short-term forecasts are determined using weekly data. If the forecasts at the two time intervals are generated independently one from the other, the sum of the higher-frequency forecasts within each time interval of the lower-frequency series do not necessarily add up to the lower-frequency forecasts. The HPFTEMPRECON procedure reconciles the high-frequency forecasts to the low-frequency forecasts in such a way that the accumulation of the reconciled high-frequency forecasts is equal to the low-frequency forecasts.

The process of adjusting more frequent data to match less frequent but more reliable data is known in the statistical literature as *benchmarking*. Denton (1971) provided the first general framework for benchmarking based on the minimization of a quadratic loss function. A recent and comprehensive review of the topic can be found in Dagum and Cholette (2006). PROC HPFTEMPRECON follow an approach similar to the one outlined in Quenneville et al. (2006). The lower-frequency forecasts are also referred to as the *benchmark forecasts*. The higher-frequency forecasts are also referred to as the *indicator forecasts*. The benchmarking procedure can be applied more generally to any two series that are measured at different time intervals. Therefore, this chapter more generally refers to the *benchmark series* and *indicator series* to indicate the forecasts involved in the benchmarking.

Denote the indicator series by $x_t$ with $t = 1, \ldots, T$, where t is associated with a date that corresponds to the time ID variable of the indicator series. Denote the benchmark series by $a_m$, $m = 1 \ldots, M$. The benchmarks have a starting date $t_{1,m}$ and ending date $t_{2,m}$ such that $1 \leq t_{1,m} \leq t_{2,m} \leq T$.

The bias is defined as the expected discrepancy between the benchmark and the indicator series. You can decide whether to adjust the original indicator series to account for the bias using the BIASCORRECTION= option. Denote by $s_t$ the bias-adjusted indicator series. When no adjustment for bias is performed, $s_t := x_t$.

The (*additive bias*) correction is given by:

$$b = \frac{\sum_{m=1}^{M} a_m - \sum_{m=1}^{M} \sum_{t=t_{1,m}}^{t_{2,m}} x_t}{\sum_{m=1}^{M} \sum_{t=t_{1,m}}^{t_{2,m}} 1}$$

In this case, the bias-adjusted indicator series is $s_t := b + x_t$.

The (*multiplicative bias*) correction is given by:

$$b = \frac{\sum_{m=1}^{M} a_m}{\sum_{m=1}^{M} \sum_{t=t_{1,m}}^{t_{2,m}} x_t}$$

In this case, the bias-adjusted indicator series is $s_t := b \cdot x_t$.

Note that the multiplicative bias is not defined when the denominator is zero. When such an event occurs and BIASCORRECTION=MULTIPLICATIVE, the required bias correction is not applied. Observations for which the bias correction cannot be applied are identified by the corresponding code in the _RECFLAGS_ variable in the OUTFOR= data set.

Let $\boldsymbol{s} := [s_1, s_2, \ldots, s_T]'$ be the vector of the bias-corrected indicator series, and let $\boldsymbol{\theta} := [\theta_1, \theta_2, \ldots, \theta_T]'$ be the vector of its reconciled values. Let $D$ be the $T \times T$ diagonal matrix whose main-diagonal elements are $d_{t,t} = |s_t|^{\lambda}$, $t = 1, \ldots, T$. Indicate by $V$ the tridiagonal symmetric matrix whose main-diagonal elements are $v_{1,1} = v_{T,T} = 1$ and $v_{t,t} = 1 + \rho^2$, $t = 2, \ldots, T-1$, and whose sub- and super-diagonal elements are $v_{t,t+1} = v_{t+1,t} = -\rho$, $t = 1, \ldots, T-1$. Define $Q := D^{+}VD^{+}$ and $\boldsymbol{c} := -Q\boldsymbol{s}$, where $D^{+}$ indicates the Moore-Penrose pseudo-inverse of $D$.

The benchmarked (or reconciled) series is given by the values $\theta_t$, $t = 1, \ldots, T$ that minimize the quadratic function

$$f(\boldsymbol{\theta}; \lambda, \rho) = \frac{1}{2}\boldsymbol{\theta}' Q \boldsymbol{\theta} + \boldsymbol{c}'\boldsymbol{\theta}$$

under the constraints

$$\sum_{t=t_{1,m}}^{t_{2,m}} \theta_t = a_m, \qquad m = 1, \ldots, M$$

where $0 \leq \rho \leq 1$ and $\lambda \in \Re$ are parameters that you define with the SMOOTH= and SCALEXP= options, respectively, in the PROC HPFTEMPRECON statement. When $\boldsymbol{s}$ does not contain zeros, the preceding target function is equivalent to the one proposed by Quenneville et al. (2006). The parameter $\rho$ is a smoothing parameter that controls movement preservation. The closer $\rho$ is to one, the more the original series movement is preserved. The parameter $\lambda$ typically takes values 0, 0.5, or 1 and controls the *proportionality* of the target function. For $\lambda = 0$, you have an additive benchmarking model. Note that when $\lambda = 0$ and $s_t = 0$, zero to the power zero is assumed to be one to guarantee that the target function be defined. For $\lambda = 0.5$ and $\rho = 0$, you have a pro-rating benchmarking model. The two first-difference versions of the penalty functions proposed by Denton (1971), as modified by Cholette (1984), can be retrieved as special cases by setting $\rho = 1$ and $\lambda = 0$, or $\lambda = 1$.

When $0 \leq \rho < 1$, the preceding minimization problem is equivalent to a constrained regression problem where the error between the bias-adjusted indicator and the benchmarked series follows an AR(1) process with autoregressive parameter proportional to $\rho$. See Quenneville et al. (2006) for more details.

## Input and Output Data Sets

### ACCDATA= Data Set

The ACCDATA= data set contains the BENCHACCUMULATION= option values for each dependent variable listed in the _NAME_ variable. The ACCDATA= data set is useful when the input data sets are generated by the HPFENGINE procedure with multiple FORECAST statements that do not have equal values of the ACCUMULATE= option.

The ACCDATA= data set must contain the following variables:

_NAME_ the variable name

_ACCUMULATE_ the value of the BENCHACCUMULATION= option to be used for the specified _NAME_ value

If not all possible values of _NAME_ are listed in the ACCDATA= data set, the BENCHACCUMULATION= option is determined by the value specified by the BENCHACCUMULATION= option in the HPFTEMPRECON statement.

## BENCHDATA= Data Set

The BENCHDATA= data set contains the low-frequency data. The frequency at which the low-frequency series is measured must be lower than the frequency at which the high-frequency series in the DATA= data set is observed. Additionally, the intervals at which the high-frequency data is measured must be fully nested in one interval of the low-frequency data. That is, one interval of the high-frequency data cannot span over two intervals of the low-frequency data. For example, days are nested in weeks, but weeks are not nested in months because one week can span over two months. At most one observation is allowed for each time interval in the BENCHDATA= data set. If there are two or more observations for the same time interval, only the first one is used for the reconciliation process.

See the sections "BENCHID Statement" on page 496 and "ID Statement" on page 498 for more details about the time intervals.

Typically, the data set (which is specified in the BENCHDATA= option in the PROC HPFTEMPRECON statement) is generated by the OUTFOR= option in a previous run of PROC HPFENGINE that produces the forecasts for the low-frequency data. The INTERVAL= option in the BENCHID statement in the current PROC HPFTEMPRECON run is also equal to the INTERVAL= option in the ID statement in the previous PROC HPFENGINE run. See Example 16.1 for an example of a typical run of HPFTEMPRECON which follows two runs of PROC HPFENGINE. See also Chapter 7, "The HPFENGINE Procedure," for more details about the HPFENGINE procedure.

The BENCHDATA= data set must contain the BY variables, the BENCHID variable, and the following variables:

| | |
|---|---|
| _NAME_ | variable name |
| PREDICT | predicted values |

The following variables can optionally be present in the BENCHDATA= data set and are used when available. If they are not present, their value is assumed to be missing for computational purposes.

| | |
|---|---|
| ACTUAL | actual values |
| LOWER | lower confidence limits |
| UPPER | upper confidence limits |
| ERROR | prediction errors |
| STD | prediction standard errors |

The BENCHDATA= data set must be either sorted by the BY variables and by the BENCHID variable, or indexed on the same variables.

You can specify custom names for the variables in the BENCHDATA= data set by using the SAS RENAME= data set option or by using the RECONCILE statement.

## DATA= Data Set

The DATA= data set contains the high-frequency data. The frequency at which the high-frequency series are measured must be higher than the frequency at which the low-frequency series in the BENCHDATA= data set are observed. Additionally, the intervals at which the high-frequency data are specified must not span over more than one interval of the low-frequency data. Only one observation is allowed for each time

interval. If there are two or more observations in the same time interval, only the first one is used for the reconciliation process, and the observations that are not reconciled are identified by a type 'D' missing value for the PREDICT variable in the OUTFOR= data set, where the type 'D' missing value indicates a duplicate in the time ID sequence.

See the sections "BENCHID Statement" on page 496 and "ID Statement" on page 498 for more details about the time intervals.

Typically, the data set (which is specified in the DATA= option in the PROC HPFTEMPRECON statement) is generated by the OUTFOR= option in a previous run of PROC HPFENGINE that produces the forecasts for the high-frequency data. The INTERVAL= option in the ID statement in the current PROC HPFTEMPRECON run is also equal to the INTERVAL= option in the ID statement in the previous PROC HPFENGINE run. See Example 16.1 for an example of a typical run of HPFTEMPRECON which follows two runs of PROC HPFENGINE. See also Chapter 7, "The HPFENGINE Procedure," for more details about the HPFENGINE procedure.

The DATA= data set must contain the BY variables, the ID variable, and the following variables:

| | |
|---|---|
| _NAME_ | variable name |
| PREDICT | predicted values |

The following variables can optionally be present in the DATA= data set and are used when available. If they are not present, their value is assumed to be missing for computational purposes.

| | |
|---|---|
| ACTUAL | actual values |
| LOWER | lower confidence limits |
| UPPER | upper confidence limits |
| ERROR | prediction errors |
| STD | prediction standard errors |

The DATA= data set must be either sorted by the BY variables and by the ID variable, or indexed on the same variables.

You can specify custom names for the variables in the DATA= data set by using the SAS RENAME data set option or by using the RECONCILE statement.

### OUTFOR= Data Set

The OUTFOR= data set contains the BY variables, the time ID variable, and the following variables:

| | |
|---|---|
| _NAME_ | variable name |
| ACTUAL | actual values |
| PREDICT | predicted values |
| LOWER | lower confidence limits |
| PPER | upper confidence limits |
| ERROR | prediction errors |
| STD | prediction standard errors |

_RECFLAGS_ benchmarking status flags. This variable contains 32-bit flags in hexadecimal format that identify various characteristics of the corresponding value in the PREDICT column. These characteristics can be informational, or they represent errors encountered during the formulation and solution of the optimization problem used to generate the reconciled forecast, or sometimes both. The notation 0x*n* is used to indicate that *n* is to be interpreted as a hexadecimal digit.

These bit-flags can occur in various combinations depending on need. A value of 0x0 in _RECFLAGS_ indicates that the corresponding PREDICT value was successfully reconciled without qualification.

There are two categories of _RECFLAGS_ bits: some represent failures and some represent informational conditions. When a failure occurs, the benchmarked series in the OUTFOR data set is the original indicator series. The following list defines the flag values associated with failure conditions:

0x00000001 A generic error condition occurred.

0x00000002 Insufficient memory was available.

0x00000004 Input data inconsistency prohibited formulation of the optimization problem.

0x00000008 A quadratic solver error occured.

0x00000010 A computational error occurred when the optimization problem was being solved.

0x00000020 A matching BY group was not found in the BENCHDATA= data set.

0x00000030 Not enough data to formulate the problem.

Informational flags are defined as follows:

0x00010000 A missing predicted value in DATA= data set was replaced by actual value.

0x00020000 An observation has no BENCHDATA= parent observation.

0x00030000 Bias correction was not applied.

If you want to interpret these flag values in SAS code, it might be helpful to consult the *SAS Language Reference: Dictionary* for the DATA step functions such as BOR, BAND, and BRSHIFT.

BENCHDIFF difference between the reconciled series and the input series predicted value. This variable is generated only when the BENCHDIFF option is specified in the HPFTEMPRECON statement.

The OUTFOR= data set is always sorted by the BY variables, the _NAME_ variable, and the time ID variable, even if input data sets are indexed and not sorted.

The values of the ID variable in the OUTFOR= data set are aligned based on the ALIGN= and INTERVAL= options specified in the ID statement. If the ALIGN= option is not specified, then the values are aligned to the beginning of the interval.

When more than one observation is present in one time interval in the input DATA= data set, only the first observation is used in the reconciliation process and the observations that are not reconciled are identified by a type 'D' missing value for the PREDICT variable in the OUTFOR= data set, where the type 'D' missing value indicates a duplicate in the time ID sequence.

## OUTPROCINFO= Data Set

The OUTPROCINFO= data set contains information about the run of the HPFTEMPRECON procedure. The following variables are present:

_SOURCE_ the name of the procedure, in this case HPFTEMPRECON.

_NAME_ name of an item being reported; can be the number of errors, notes, or warnings, number of benchmarks requested, and so on. Table 16.2 summarizes the values and meanings of the _NAME_ variable in PROC HPFTEMPRECON. When threads are used, the value of the row that corresponds to _STAGE_='THREAD' defines the number of threads used for solving reconciliation solver instances. Subsequent rows groups (which correspond to _STAGE_='THREAD'_*n*) record per-thread buffer statistics, where *n*=1, . . ., NCPUCOUNTS.

**Table 16.2** Values of the _NAME_ Variable

| _NAME_ | _STAGE_ | Meaning |
|---|---|---|
| SERIES | ALL | Number of series processed |
| REQUESTED | ALL | Number of reconciliations requested |
| FAILED | ALL | Number of reconciliations that failed |
| MODE | ALL | Thread mode used |
| NBUFIN | ALL | Number of input buffers serviced |
| NBUFOUT | ALL | Number of output buffers serviced |
| MINBUFIN | ALL | Minimum input buffer size processed |
| MINBUFOUT | ALL | Minimum output buffer size processed |
| MAXBUFIN | ALL | Maximum input buffer size processed |
| MAXBUFOUT | ALL | Maximum output buffer size processed |
| NTHREADS | THREAD | Number of threads used |
| NBUFIN | THREAD_*n* | Number of input buffers serviced |
| NBUFOUT | THREAD_*n* | Number of output buffers serviced |
| MINBUFIN | THREAD_*n* | Minimum input buffer size processed |
| MINBUFOUT | THREAD_*n* | Minimum output buffer size processed |
| MAXBUFIN | THREAD_*n* | Maximum input buffer size processed |
| MAXBUFOUT | THREAD_*n* | Maximum output buffer size processed |

_LABEL_ descriptive label for the item in _NAME_.

_STAGE_ current stage of the procedure. For execution statistics gained from the procedure as a whole, this is set to 'ALL'. For execution statistics related to a particular thread, it is set to 'THREAD'_*n*, where *n* is the ordered number of the thread in question.

_VALUE_ value of the item specified in _NAME_.

## OUTSTAT= Data Set

See the section "OUTSTAT= Data Set" on page 226 in Chapter 7, "The HPFENGINE Procedure," for details about the OUTSTAT= data set.

## Threading Details

PROC HPFTEMPRECON can multithread to reduce the elapsed time required to reconcile forecasts. For each indicator/benchmark series pairing in the reconciliation run, PROC HPFTEMPRECON must formulate and solve an instance of a mathematical program (MP) to generate reconciled forecasts (see the section "Mathematical Foundation" on page 501). Given a reconciliation run with a large number of BY groups or a run with several RECONCILE statements, or both in combination, substantial leverage can be gained by the use of multiprocessing to generate and solve the sequence of optimization problems. This can result in substantially reduced solution time with adequate CPU and memory resources on the system that runs PROC HPFTEMPRECON.

You can use SAS system options to control threading in PROC HPFTEMPRECON.You enable threading with the THREADS option, or disable it with the NOTHREAD option. The system option CPUCOUNT= sets the maximum number of threads allowed. CPUCOUNT=1 is equivalent to no threading. If the BY statement is not specified, threading is disabled regardless of the value of these options, since the cost of using threads would outweigh the benefits when there is only one optimization program to solve. Other settings of CPUCOUNT in the presence of a BY statement choose the most appropriate threading strategy to use the available CPU resources.

See *SAS System Options: Reference* for more information about the THREADS | NOTHREADS and CPUCOUNT= system options. See also Example 16.3 for an example of using threads with PROC HPFTEMPRECON.

# Examples: HPFTEMPRECON Procedure

## Example 16.1: Reconciling Monthly Forecasts to Quarterly Forecasts

In this example, monthly forecasts for the data set Sashelp.Air are reconciled to the quarterly forecasts of the same series.

First, PROC HPFESMSPEC generates an exponential smoothing model specification which was selected by the HPFSELECT procedure. See Chapter 8 and Chapter 15 for more details about the HPFESMSPEC and HPFSELECT procedures, respectively.

```
proc hpfesmspec
      rep=work.repo
      specname=myesm;
   esm;
run;

proc hpfselect
      rep=work.repo
      name=myselect;
   spec myesm;
run;
```

Then, forecasts are generated with PROC HPFENGINE at the monthly and the quarterly frequencies using the selected model specification.

```
proc hpfengine
      data=sashelp.air
      rep=work.repo
      globalselection=myselect
      out=outmon
      outfor=outformon
      outmodelinfo=outmodmon;
   id date interval=month;
   forecast air;
run;

proc hpfengine
      data=sashelp.air
      rep=work.repo
      globalselection=myselect
      out=outqtr
      outfor=outforqtr
      outmodelinfo=outmodqrt;
   id date interval=qtr accumulate=total;
   forecast air;
run;
```

Finally, the monthly forecasts are reconciled to the quarterly forecasts using PROC HPFTEMPRECON with default values for the smoothing and scale exponent parameters.

```
proc hpftemprecon
      data=outformon
      benchdata=outforqtr
      outfor=benfor1
      outstat=benstat1;
   id date interval=month;
   benchid date interval=qtr;
run;
```

Output 16.1.1 displays the first 20 rows of the output data set BENFOR1, which contains the reconciled forecasts.

```
title 'BENFOR1';
proc print data=benfor1(obs=20); run;
```

**Output 16.1.1** Reconciled Forecasts, Default Parameters

**BENFOR1**

| Obs | _NAME_ | DATE | ACTUAL | PREDICT | LOWER | UPPER | ERROR | STD | _RECFLAGS_ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | AIR | JAN1949 | 112 | 109.596 | 88.642 | 130.550 | 2.4038 | 10.6910 | 00000000 |
| 2 | AIR | FEB1949 | 118 | 118.292 | 97.338 | 139.246 | -0.2920 | 10.6910 | 00000000 |
| 3 | AIR | MAR1949 | 132 | 134.101 | 113.147 | 155.055 | -2.1010 | 10.6910 | 00000000 |
| 4 | AIR | APR1949 | 129 | 131.261 | 110.307 | 152.215 | -2.2608 | 10.6910 | 00000000 |
| 5 | AIR | MAY1949 | 121 | 125.618 | 104.664 | 146.572 | -4.6184 | 10.6910 | 00000000 |
| 6 | AIR | JUN1949 | 135 | 142.663 | 121.709 | 163.617 | -7.6630 | 10.6910 | 00000000 |
| 7 | AIR | JUL1949 | 148 | 158.532 | 137.578 | 179.486 | -10.5322 | 10.6910 | 00000000 |
| 8 | AIR | AUG1949 | 148 | 158.191 | 137.237 | 179.145 | -10.1907 | 10.6910 | 00000000 |
| 9 | AIR | SEP1949 | 136 | 144.719 | 123.765 | 165.673 | -8.7193 | 10.6910 | 00000000 |
| 10 | AIR | OCT1949 | 119 | 124.365 | 103.411 | 145.319 | -5.3648 | 10.6910 | 00000000 |
| 11 | AIR | NOV1949 | 104 | 109.137 | 88.183 | 130.091 | -5.1366 | 10.6910 | 00000000 |
| 12 | AIR | DEC1949 | 118 | 127.854 | 106.900 | 148.808 | -9.8543 | 10.6910 | 00000000 |
| 13 | AIR | JAN1950 | 115 | 133.405 | 112.451 | 154.359 | -18.4047 | 10.6910 | 00000000 |
| 14 | AIR | FEB1950 | 126 | 138.363 | 117.408 | 159.317 | -12.3625 | 10.6910 | 00000000 |
| 15 | AIR | MAR1950 | 141 | 153.119 | 132.165 | 174.073 | -12.1191 | 10.6910 | 00000000 |
| 16 | AIR | APR1950 | 135 | 147.403 | 126.449 | 168.357 | -12.4029 | 10.6910 | 00000000 |
| 17 | AIR | MAY1950 | 125 | 136.212 | 115.258 | 157.166 | -11.2116 | 10.6910 | 00000000 |
| 18 | AIR | JUN1950 | 149 | 148.854 | 127.900 | 169.808 | 0.1464 | 10.6910 | 00000000 |
| 19 | AIR | JUL1950 | 170 | 163.688 | 142.734 | 184.642 | 6.3119 | 10.6910 | 00000000 |
| 20 | AIR | AUG1950 | 170 | 166.391 | 145.436 | 187.345 | 3.6095 | 10.6910 | 00000000 |

Output 16.1.2 displays the BENSTAT1 data set, which contains the statistics of fit for the reconciled forecasts.

```
title 'BENSTAT1';
proc print data=benstat1 noobs; run;
```

**Output 16.1.2** Reconciled Forecasts Statistics of Fit

**BENSTAT1**

| _NAME_ | _REGION_ | DFE | NMISSA | NOBS | N | NPARMS | NMISSP | TSS | SST | SSE | MSE |
|---|---|---|---|---|---|---|---|---|---|---|---|
| AIR | FORECAST | 144 | 12 | 156 | 156 | 0 | 0 | 13371737 | 2058044.16 | 22969.94 | 159.513 |

| RMSE | UMSE | URMSE | MAPE | MAE | RSQUARE | ADJRSQ | AADJRSQ | RWRSQ | AIC | AICC | SBC |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 12.6299 | 159.513 | 12.6299 | 3.92521 | 9.95548 | 0.98884 | 0.98892 | 0.98884 | 0.85901 | 730.386 | 730.386 | 730.386 |

| APC | MAXERR | MINERR | MAXPE | MINPE | ME | MPE | MDAPE | GMAPE | MINPPE | MAXPPE | MPPE |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 159.513 | 33.0807 | -35.4894 | 10.7302 | -18.3817 | 0.035721 | -0.74054 | 3.24930 | 2.57881 | -15.5275 | 12.0200 | -0.49959 |

| MAPPE | MDAPPE | GMAPPE | MINSPE | MAXSPE | MSPE | SMAPE | MDASPE | GMASPE | MINRE | MAXRE | MRE |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 3.84944 | 3.31582 | 2.56290 | -16.8345 | 11.3386 | -0.61790 | 3.88241 | 3.30298 | 2.57007 | -25.2160 | 11.9697 | -0.17253 |

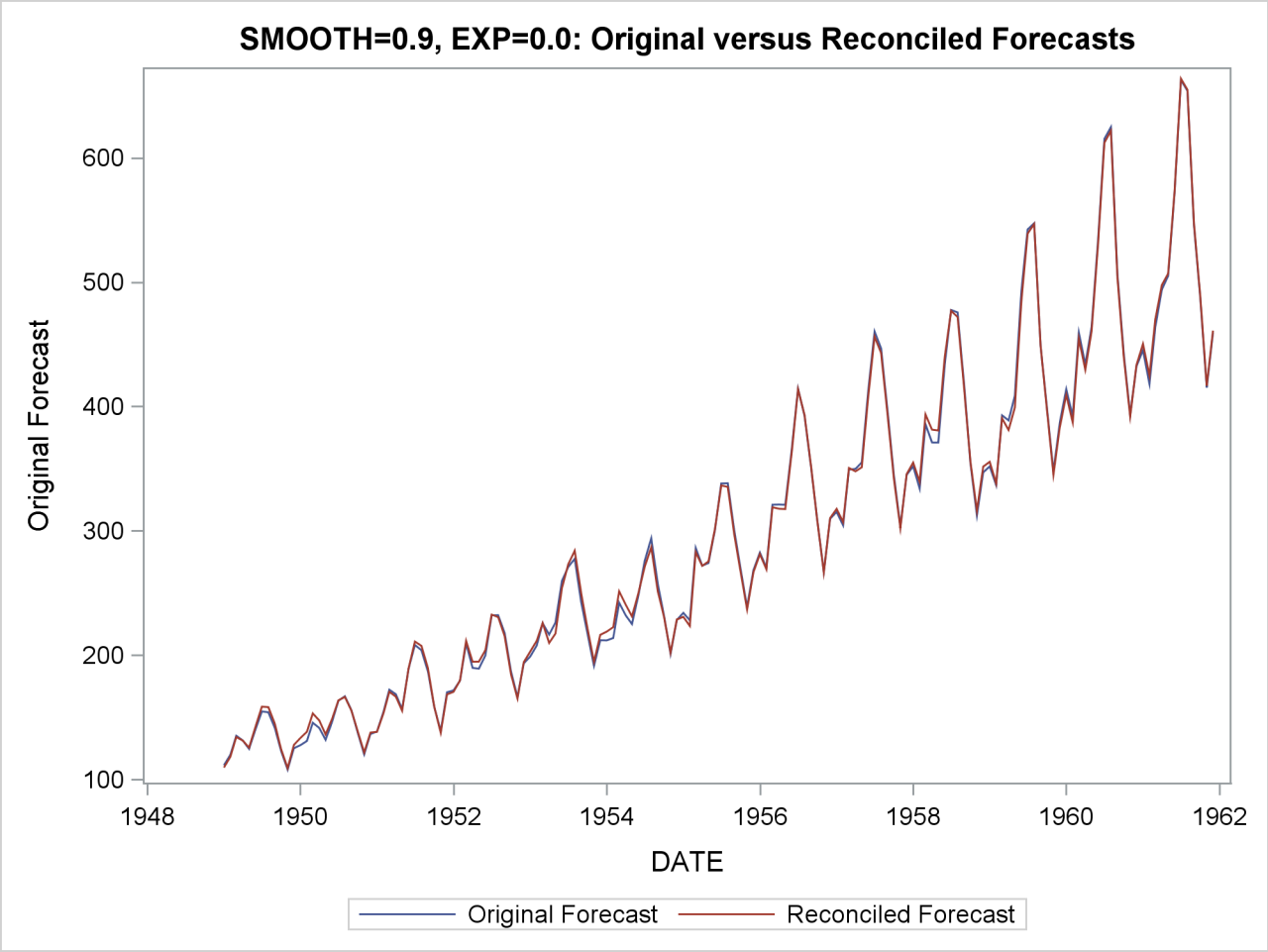
| MRAE | MDRAE | GMRAE | MASE | MINAPES | MAXAPES | MAPES | MDAPES | GMAPES |
|---|---|---|---|---|---|---|---|---|
| 0.99869 | 0.40464 | 0.36093 | 0.38497 | 0.12206 | 29.5828 | 8.29856 | 7.42818 | 5.48651 |

You can vary the reconciled forecasts by selecting the values of the SMOOTH= and EXP= options. Figure 16.1.3, Figure 16.1.4, and Figure 16.1.5 show the original forecasts versus the reconciled forecasts for different combinations of the two parameters. See the section "Mathematical Foundation" on page 501 for more details about the SMOOTH= and EXP= options.

```
data combined1;
   label opredict="Original Forecast";
   label bpredict="Reconciled Forecast";
   merge outformon(rename=(predict=opredict))
      benfor1(rename=(predict=bpredict));
   by _name_ date;
run;

title "SMOOTH=0.9, EXP=0.0: Original versus Reconciled Forecasts";
proc sgplot data=combined1;
   series x=date y=opredict;
   series x=date y=bpredict;
run;
```

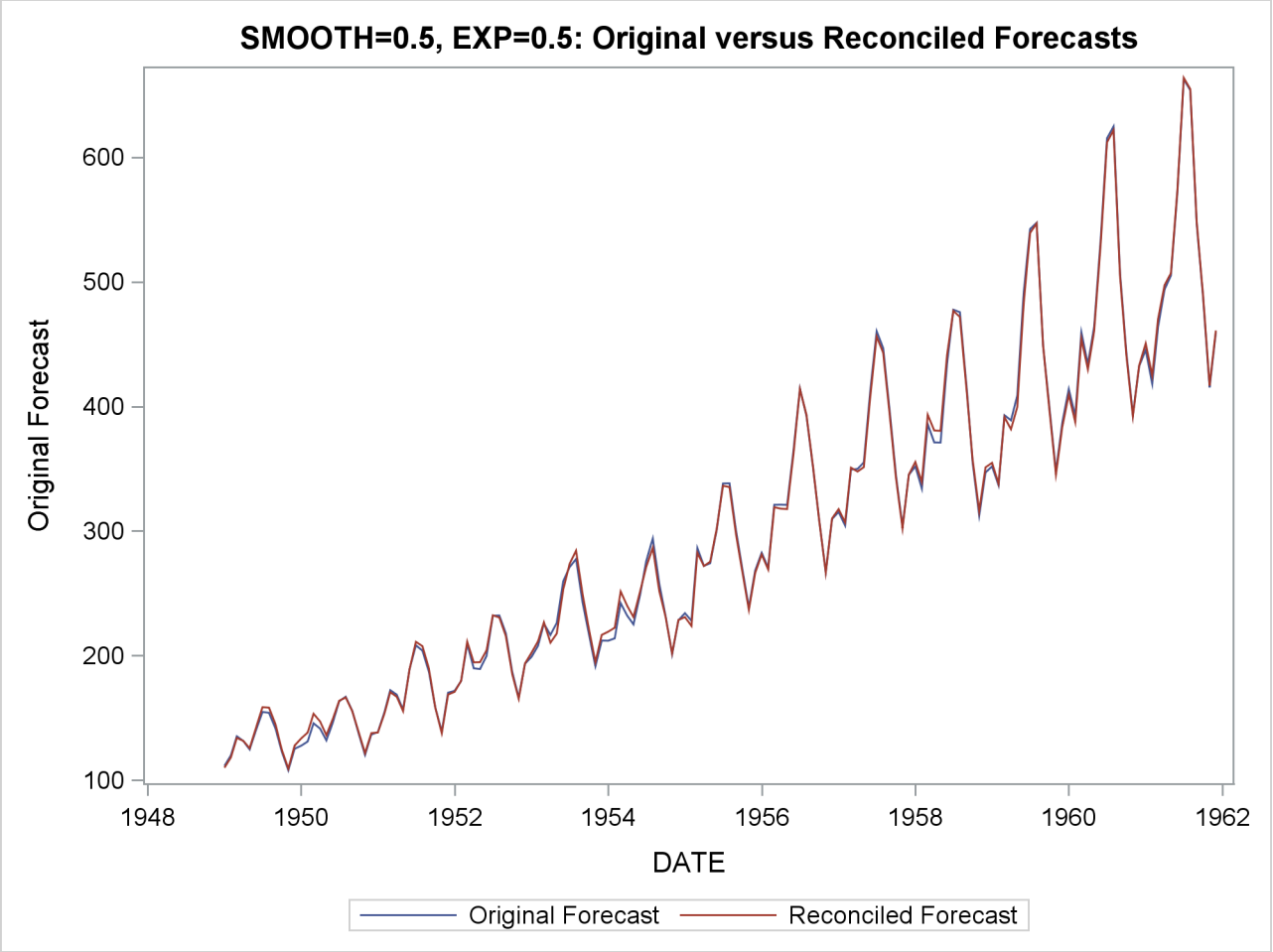
**Output 16.1.3** Original versus Reconciled Forecasts, SMOOTH=0.9, EXP=0



```
proc hpftemprecon
      data=outformon
      benchdata=outforqtr
      outfor=benfor2
      outstat=benstat2
      exp=0.5
      smooth=0.5;
   id date interval=month;
   benchid date interval=qtr;
run;

title "SMOOTH=0.5, EXP=0.5: Original versus Reconciled Forecasts";
data combined2;
   label opredict="Original Forecast";
   label bpredict="Reconciled Forecast";
   merge outformon(rename=(predict=opredict))
      benfor2(rename=(predict=bpredict));
   by _name_ date;
run;
proc sgplot data=combined2;
   series x=date y=opredict;
   series x=date y=bpredict;
run;
```

**Output 16.1.4** Original versus Reconciled Forecasts, SMOOTH=0.5, EXP=0.5

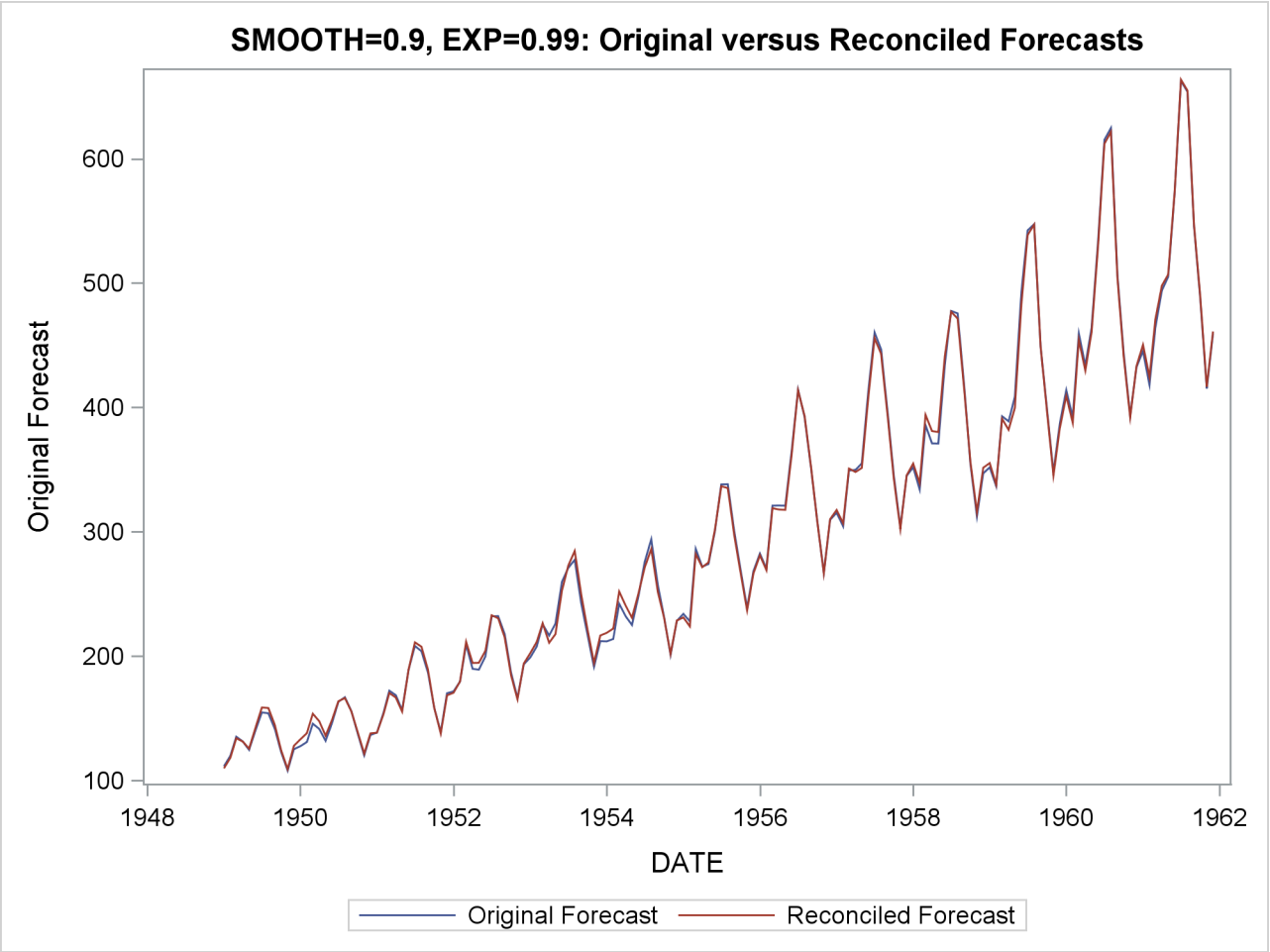



```
proc hpftemprecon
   data=outformon
      benchdata=outforqtr
      outfor=benfor3
      outstat=benstat3
      exp=0.99
      smooth=0.9;
   id date interval=month;
   benchid date interval=qtr;
run;

title "SMOOTH=0.9, EXP=0.99: Original versus Reconciled Forecasts";
data combined3;
   label opredict="Original Forecast";
   label bpredict="Reconciled Forecast";
   merge outformon(rename=(predict=opredict))
      benfor3(rename=(predict=bpredict)); by _name_ date;
run;

proc sgplot data=combined3;
   series x=date y=opredict;
   series x=date y=bpredict;
run;
```

**Output 16.1.5** Original versus Reconciled Forecasts, SMOOTH=0.9, EXP=0.99




## Example 16.2: Reconciling Multiple Variables

This example shows how you can use two RECONCILE statement to reconcile two variables in the DATA= data set to the same benchmark PREDICT variable in the BENCHDATA data set.

Suppose you collected from an expert judgmental forecasts in the forecasting horizon for the airline data set. You entered the forecasts in the judpred in the data set judgmental as follows:

```
data judgmental;
  attrib DATE length=8 format=MONYY7.;
  attrib judpred length=8;

  infile datalines dsd;
  input DATE @;
  input judpred @;
datalines4;
366,456
397,409
425,452
456,489
486,498
517,548
547,660
578,666
609,534
639,495
670,416
700,472
;;;;
```

You want to see how the reconciled statistical forecasts you derived in Example 16.1 compare to the reconciled judgmental forecasts when the quarterly statistical forecasts are used as benchmark. You can accomplish that by merging the two data predicted values in one data set and using two RECONCILE statements in PROC HPFTEMPRECON.

```
data judgmental;
   merge judgmental outformon(where=(date>"01dec1960"d));
   by date;
   keep date predict judpred;
run;


proc hpftemprecon
      data=judgmental
      benchdata=outforqtr
      outfor=benfor4
      outstat=benstat4
      exp=0.99
      smooth=0.9;
   id date interval=month;
   benchid date interval=qtr;
   reconcile predict=predict;
   reconcile judpred=predict;
run;
```

The first RECONCILE statement instructs PROC HPFTEMPRECON to reconcile the statistical forecasts PREDICT, and the second one reconciles the judgmental forecasts. The variable PREDICT on the right-hand side of the equality in both RECONCILE statements indicates that both statistical and judgmental monthly forecasts are to be reconciled against the quarterly statistical forecasts contained in the variable PREDICT in the data set outforqtr.

The OUTFOR= data set benfor4 contains the reconciled forecasts for both variables, stacked according to the order in which the RECONCILE statements appear. Similar to forecasts produced by different FORECAST statements of HPFENGINE, the _NAME_ variable is used to discriminate among variables.

```
proc print data=benfor4; run;
```

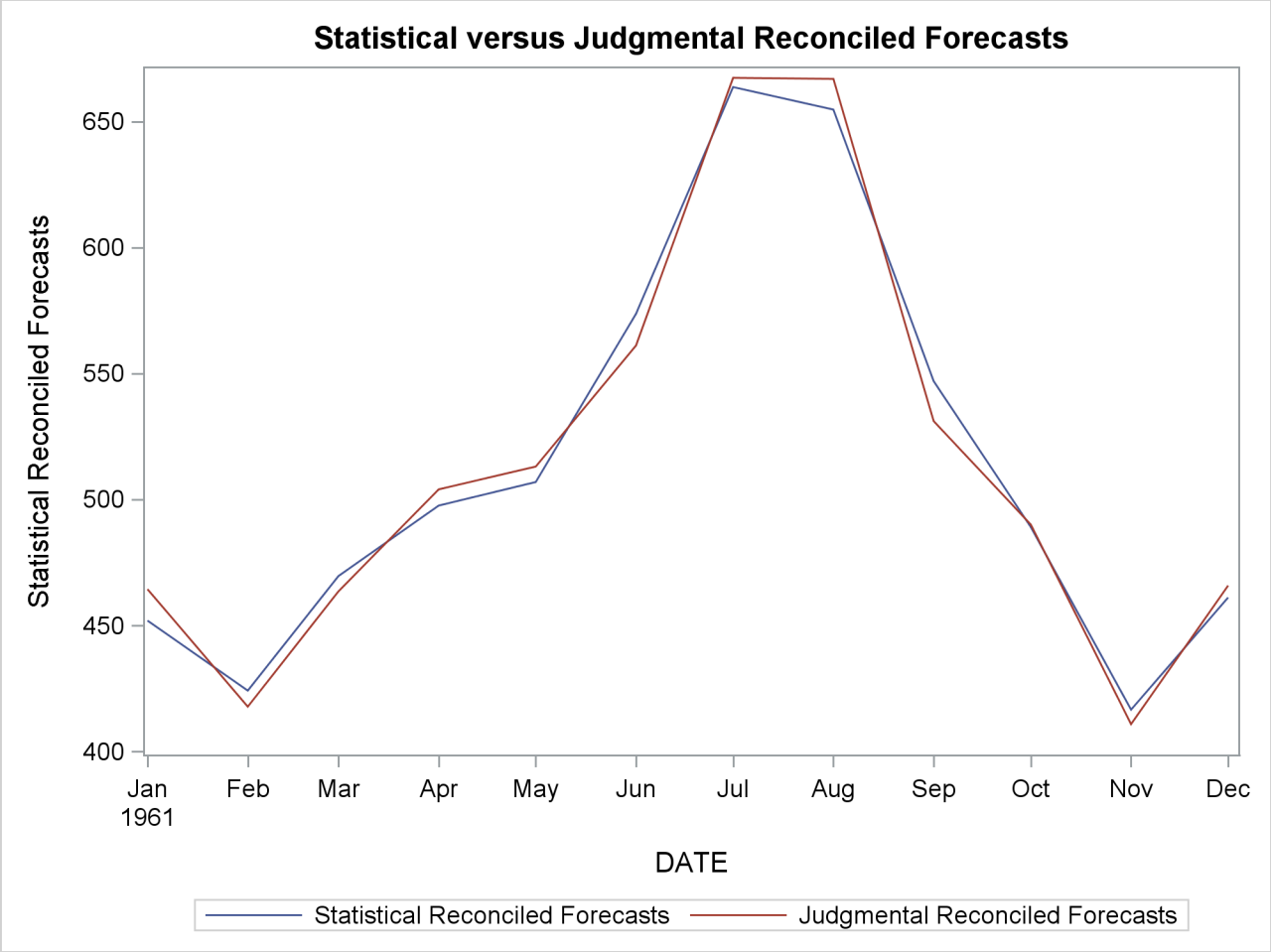
**Output 16.2.1** Reconciled Statistical and Judgmental Forecasts

**SMOOTH=0.9, EXP=0.99: Original versus Reconciled Forecasts**

| Obs | _NAME_ | DATE | ACTUAL | PREDICT | LOWER | UPPER | ERROR | STD | _RECFLAGS_ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | PREDICT | JAN1961 | . | 451.954 | . | . | . | . | 00000000 |
| 2 | PREDICT | FEB1961 | . | 424.164 | . | . | . | . | 00000000 |
| 3 | PREDICT | MAR1961 | . | 469.612 | . | . | . | . | 00000000 |
| 4 | PREDICT | APR1961 | . | 497.599 | . | . | . | . | 00000000 |
| 5 | PREDICT | MAY1961 | . | 506.940 | . | . | . | . | 00000000 |
| 6 | PREDICT | JUN1961 | . | 573.717 | . | . | . | . | 00000000 |
| 7 | PREDICT | JUL1961 | . | 663.692 | . | . | . | . | 00000000 |
| 8 | PREDICT | AUG1961 | . | 654.747 | . | . | . | . | 00000000 |
| 9 | PREDICT | SEP1961 | . | 546.957 | . | . | . | . | 00000000 |
| 10 | PREDICT | OCT1961 | . | 489.079 | . | . | . | . | 00000000 |
| 11 | PREDICT | NOV1961 | . | 416.598 | . | . | . | . | 00000000 |
| 12 | PREDICT | DEC1961 | . | 461.118 | . | . | . | . | 00000000 |
| 13 | judpred | JAN1961 | . | 464.419 | . | . | . | . | 00000000 |
| 14 | judpred | FEB1961 | . | 417.781 | . | . | . | . | 00000000 |
| 15 | judpred | MAR1961 | . | 463.530 | . | . | . | . | 00000000 |
| 16 | judpred | APR1961 | . | 504.038 | . | . | . | . | 00000000 |
| 17 | judpred | MAY1961 | . | 513.083 | . | . | . | . | 00000000 |
| 18 | judpred | JUN1961 | . | 561.135 | . | . | . | . | 00000000 |
| 19 | judpred | JUL1961 | . | 667.356 | . | . | . | . | 00000000 |
| 20 | judpred | AUG1961 | . | 666.931 | . | . | . | . | 00000000 |
| 21 | judpred | SEP1961 | . | 531.109 | . | . | . | . | 00000000 |
| 22 | judpred | OCT1961 | . | 490.083 | . | . | . | . | 00000000 |
| 23 | judpred | NOV1961 | . | 410.858 | . | . | . | . | 00000000 |
| 24 | judpred | DEC1961 | . | 465.855 | . | . | . | . | 00000000 |

Figure 16.2.2 displays how the statistical reconciled forecasts compare to the judgmental reconciled forecasts.

**Output 16.2.2** Statistical and Judgmental Reconciled Forecasts




## Example 16.3: Using Threads

As discussed in section "Threading Details" on page 507, if you have a BY statement or multiple RECONCILE statements, it can be beneficial to enable multithreading so that independent parallel threads can handle simultaneous optimization problems. This example illustrates the uses of multithreading to reconcile monthly forecasts to quarterly forecasts for the variable sale in Sashelp.Pricedata.

First, select a model and generate forecasts at the two time intervals of interest for each combination of values of the variables Region, Line, and product.

```
/* MONTHLY FORECASTS */
*Step 1: model selection;
proc hpfdiagnose data=sashelp.pricedata
   outest=monthest
   modelrepository=work.mycat
   prefilter=both
   criterion=mape;
   id date interval=month;
   by region line product;
   forecast sale;
   input price;
run;

*Step 2: estimation and forecasting ;
proc hpfengine data=sashelp.pricedata inest=monthest
   out=_null_ outest=monthfest
   modelrepository=work.mycat outfor=monthfor;
   id date interval=month;
   by region line product;
   forecast sale / task=select ;
   stochastic price;
run;


/*  QUARTERLY FORECASTS */
*Step 1: model selection;
proc hpfdiagnose data=sashelp.pricedata
   outest=qtrest
   modelrepository=work.mycat
   prefilter=both
   criterion=mape;
   id date interval=qtr accumulate = total;
   by region line product;
   forecast sale;
   input price;
run;

*Step 2: estimation and forecasting ;
proc hpfengine data=sashelp.pricedata inest=qtrest
   out=_null_ outest=qtrfest
   modelrepository=work.mycat outfor=qtrfor;
   id date interval=qtr accumulate = total;
   by region line product;
   forecast sale / task=select ;
   stochastic price;
run;
```

Suppose your computer has eight cores and you want to enable multithreading but you want to limit the number of threads of PROC HPFTEMPRECON to four in order to reserve system resources for other tasks you want to run. You enable multithreading with the THREADS system options and set the maximum number of threads with the CPUCOUNT= option.

```
options threads cpucount=4;
```

Finally, proceed to reconcile the forecasts with PROC HPFTEMPRECON as you normally would. PROC HPFTEMPRECON internal logic decides the best multithreading strategy for each situation.

```
 proc hpftemprecon
  data=monthfor
  benchdata=qtrfor
  outfor=benchfor
  outstat=benchstat
  outprocinfo=outprocinfo;
id date interval=month;
by region line product;
benchid date interval=qtr;
run;
```

The OUTPROCINFO= data set contains statistics about the procedure run. When the variable _STAGE_ is set to `'ALL'`, the statistics refers to the procedure run as a whole. When _STAGE_ is set to `'THREAD'`_*n* the relative statistics refers to the *n*th thread only.

```
proc print data=outprocinfo;
run;
```

**Output 16.3.1** PROC HPFTEMPRECON Run Information

**Statistical versus Judgmental Reconciled Forecasts**

| Obs | _SOURCE_ | _NAME_ | _LABEL_ | _STAGE_ | _VALUE_ |
|---|---|---|---|---|---|
| 1 | HPFTEMPRECON | SERIES | Number of series processed | ALL | 17 |
| 2 | HPFTEMPRECON | REQUESTED | Number of reconciliations requested | ALL | 17 |
| 3 | HPFTEMPRECON | FAILED | Number of reconciliations failed | ALL | 0 |
| 4 | HPFTEMPRECON | MODE | Thread mode used | ALL | 2 |
| 5 | HPFTEMPRECON | NBUFIN | Number of input buffers serviced | ALL | 17 |
| 6 | HPFTEMPRECON | NBUFOUT | Number of output buffers generated | ALL | 17 |
| 7 | HPFTEMPRECON | MINBUFIN | Minimum input buffer size processed | ALL | 5832 |
| 8 | HPFTEMPRECON | MINBUFOUT | Minimum output buffer size generated | ALL | 5832 |
| 9 | HPFTEMPRECON | MAXBUFIN | Maximum input buffer size processed | ALL | 5832 |
| 10 | HPFTEMPRECON | MAXBUFOUT | Maximum output buffer size generated | ALL | 5832 |
| 11 | HPFTEMPRECON | NTHREADS | Number of threads used | THREAD | 3 |
| 12 | HPFTEMPRECON | NBUFIN | Number of input buffers serviced | THREAD_1 | 6 |
| 13 | HPFTEMPRECON | NBUFOUT | Number of output buffers generated | THREAD_1 | 6 |
| 14 | HPFTEMPRECON | MINBUFIN | Minimum input buffer size processed | THREAD_1 | 5832 |
| 15 | HPFTEMPRECON | MINBUFOUT | Minimum output buffer size generated | THREAD_1 | 5832 |
| 16 | HPFTEMPRECON | MAXBUFIN | Maximum input buffer size processed | THREAD_1 | 5832 |
| 17 | HPFTEMPRECON | MAXBUFOUT | Maximum output buffer size generated | THREAD_1 | 5832 |
| 18 | HPFTEMPRECON | NBUFIN | Number of input buffers serviced | THREAD_2 | 5 |
| 19 | HPFTEMPRECON | NBUFOUT | Number of output buffers generated | THREAD_2 | 5 |
| 20 | HPFTEMPRECON | MINBUFIN | Minimum input buffer size processed | THREAD_2 | 5832 |
| 21 | HPFTEMPRECON | MINBUFOUT | Minimum output buffer size generated | THREAD_2 | 5832 |
| 22 | HPFTEMPRECON | MAXBUFIN | Maximum input buffer size processed | THREAD_2 | 5832 |
| 23 | HPFTEMPRECON | MAXBUFOUT | Maximum output buffer size generated | THREAD_2 | 5832 |
| 24 | HPFTEMPRECON | NBUFIN | Number of input buffers serviced | THREAD_3 | 6 |
| 25 | HPFTEMPRECON | NBUFOUT | Number of output buffers generated | THREAD_3 | 6 |
| 26 | HPFTEMPRECON | MINBUFIN | Minimum input buffer size processed | THREAD_3 | 5832 |
| 27 | HPFTEMPRECON | MINBUFOUT | Minimum output buffer size generated | THREAD_3 | 5832 |
| 28 | HPFTEMPRECON | MAXBUFIN | Maximum input buffer size processed | THREAD_3 | 5832 |
| 29 | HPFTEMPRECON | MAXBUFOUT | Maximum output buffer size generated | THREAD_3 | 5832 |

# References

Cholette, P. A. (1984), "Adjusting Sub-annual Series to Yearly Benchmarks," *Survey Methodology*, 10, 35–49.

Dagum, E. B. and Cholette, P. A. (2006), *Benchmarking, Temporal Distribution, and Reconciliation Methods for Time Series*, volume 186 of *Lecture Notes in Statistics*, Springer-Verlag.

Denton, F. T. (1971), "Adjustment of Monthly or Quarterly Series to Annual Totals: An Approach Based on Quadratic Minimization," *Journal of the American Statistical Association*, 82, 99–102.

Quenneville, B., Fortier, S., Chen, Z.-G., and Latendresse, E. (2006), "Recent Developments in Benchmarking to Annual Totals in X-12-ARIMA and at Statistics Canada," in *Proceedings of the Eurostat Conference on Seasonality, Seasonal Adjustment, and Their Implications for Short-Term Analysis and Forecasting*, Luxembourg: Eurostat.

# Chapter 17
# The HPFUCMSPEC Procedure

**Contents**


Overview: HPFUCMSPEC Procedure . . . . . **521**
Getting Started: HPFUCMSPEC Procedure . . . . . **522**
Syntax: HPFUCMSPEC Procedure . . . . . **523**
Functional Summary . . . . . 523
PROC HPFUCMSPEC Statement . . . . . 525
AUTOREG Statement . . . . . 525
BLOCKSEASON Statement . . . . . 526
CYCLE Statement . . . . . 527
DEPLAG Statement . . . . . 528
FORECAST Statement . . . . . 529
INPUT Statement . . . . . 530
IRREGULAR Statement . . . . . 531
LEVEL Statement . . . . . 532
SEASON Statement . . . . . 533
SLOPE Statement . . . . . 534
Examples: HPFUCMSPEC Procedure . . . . . **535**
Example 17.1: Some HPFUCMSPEC Syntax Illustrations . . . . . 535
Example 17.2: How to Include a UCM in a Model Selection List . . . . . 536
Example 17.3: How to Create a Generic Seasonal Model Specification . . . . . 538
References . . . . . **540**


## Overview: HPFUCMSPEC Procedure

The HPFUCMSPEC procedure is used to create an unobserved component model (UCM) specification file. The output of this procedure is an XML file that stores the intended UCM specification. This XML specification file can be used for different purposes—for example, it can be used to populate the model repository used by the HPFENGINE procedure (see Chapter 7, "The HPFENGINE Procedure"). You can specify any UCM that can be analyzed by using the UCM procedure; see Chapter 34, "The UCM Procedure" (*SAS/ETS User's Guide*). Moreover, the model specification can include series transformations such as log or Box-Cox transformations. Apart from minor modifications to accommodate series transformations, the model specification syntax of the HPFUCMSPEC procedure is similar to that of the UCM procedure.

# Getting Started: HPFUCMSPEC Procedure

The following example shows how to create a UCM specification file. In this example the specification for a basic structural model (BSM) with one input is created.

```
proc hpfucmspec repository=mymodels
                name=BSM1
                label="Basic structural model with one input";
   forecast symbol=Y transform=log;
   input symbol=X;
   irregular;
   level;
   slope variance=0 noest;
   season length=12 type=trig;
run;
```

The options in the PROC HPFUCMSPEC statement are used to specify the location of the specification file that will be output. Here the REPOSITORY= option specifies that the output file be placed in a catalog MYMODELS, the NAME= option specifies that the name of the file be BSM1.xml, and the LABEL= option specifies a label for this catalog member. The other statements in the procedure specify the UCM.

The model specification begins with the FORECAST statement that specifies a transformation, such as a log or Box-Cox, for the variable that is to be forecast. In some cases, the forecast variable is also called the *dependent* variable or the *response* variable. Here, the FORECAST statement specifies a log transformation for the series being forecast. The SYMBOL= option in the FORECAST statement can be used to provide a convenient name for the forecast variable. This name is only a placeholder, and a proper data variable will be associated with this name when this model specification is used in actual data analysis. Next, the INPUT statement specifies transformations, such as log or Box-Cox, as well as lagging and differencing that are associated with the input variable. In this case the input variable enters the model as a simple regressor. Here again the SYMBOL= option can be used to supply a convenient name for the input variable. If a model contains multiple input variables, then each input variable has to be specified with a separate INPUT statement.

After the forecast and input series transformations are described, the components in the model are specified using different component statements. In the above example the model contains three components: an *irregular* component, a *local linear trend* with fixed *slope*, and a *trigonometric seasonal* with season length 12.

# Syntax: HPFUCMSPEC Procedure

The HPFUCMSPEC procedure uses the following statements.

**PROC HPFUCMSPEC** *options* ;
**AUTOREG** *options* ;
**BLOCKSEASON** *options* ;
**CYCLE** *options* ;
**DEPLAG** *options* ;
**FORECAST** *options* ;
**INPUT** *options* ;
**IRREGULAR** *options* ;
**LEVEL** *options* ;
**SEASON** *options* ;
**SLOPE** *options* ;

## Functional Summary

Table 17.1 the statements and options that control the HPFUCMSPEC procedure.

**Table 17.1** HPFUCMSPEC Functional Summary

| Description | Statement | Option |
|---|---|---|
| **Model Repository Options** | | |
| Specifies the model repository | PROC HPFUCMSPEC | REPOSITORY= |
| Specifies the model specification name | PROC HPFUCMSPEC | NAME= |
| Specifies the model specification label | PROC HPFUCMSPEC | LABEL= |
| **Options for Specifying Symbolic Series Names** | | |
| Specifies a symbolic name for the response series | FORECAST | SYMBOL= |
| Specifies a symbolic name for the input series | INPUT | SYMBOL= |
| **Options for Specifying the Model** | | |
| Specifies the response series transformation | FORECAST | TRANSFORM= |
| Specifies the input series transformation | INPUT | TRANSFORM= |
| Specifies the input series differencing orders | INPUT | DIF= |
| Specifies the input series lagging order | INPUT | DELAY= |
| Specifies the initial value for the disturbance variance of the irregular component | IRREGULAR | VARIANCE= |
| Fixes the value of the disturbance variance of the irregular component to the specified initial value | IRREGULAR | NOEST |
| Specifies the initial value for the disturbance variance of the level component | LEVEL | VARIANCE= |

| Description | Statement | Option |
|---|---|---|
| Fixes the value of the disturbance variance of the level component to the specified initial value | LEVEL | NOEST |
| Specifies the initial value for the disturbance variance of the slope component | SLOPE | VARIANCE= |
| Fixes the value of the disturbance variance of the slope component to the specified initial value | SLOPE | NOEST |
| Specifies the season length of a seasonal component | SEASON | LENGTH= |
| Specifies the type of a seasonal component | SEASON | TYPE= |
| Specifies the initial value for the disturbance variance of a seasonal component | SEASON | VARIANCE= |
| Fixes the value of the disturbance variance of the seasonal component to the specified initial value | SEASON | NOEST |
| Specifies the block size of a block seasonal component | BLOCKSEASON | BLOCKSIZE= |
| Specifies the number of blocks of a block seasonal component | BLOCKSEASON | NBLOCKS= |
| Specifies the relative position of the first observation within the block of a block seasonal component | BLOCKSEASON | OFFSET= |
| Specifies the initial value for the disturbance variance of a block seasonal component | BLOCKSEASON | VARIANCE= |
| Fixes the value of the disturbance variance of the block seasonal component to the specified initial value | BLOCKSEASON | NOEST |
| Specifies the initial value for the period of a cycle component | CYCLE | PERIOD= |
| Specifies the initial value for the damping factor of a cycle component | CYCLE | RHO= |
| Specifies the initial value for the disturbance variance of the cycle component | CYCLE | VARIANCE= |
| Fixes the values of the parameters of the cycle component to the specified initial values | CYCLE | NOEST= |
| Specifies the initial value for the damping factor of the autoregressive component | AUTOREG | RHO= |
| Specifies the initial value for the disturbance variance of the autoregressive component | AUTOREG | VARIANCE= |
| Fixes the values of the parameters of the autoregressive component to the specified initial values | AUTOREG | NOEST= |
| Specifies the lags of the response series to be included in the model | DEPLAG | LAGS= |

| Description | Statement | Option |
|---|---|---|
| Specifies the initial values for the lag coefficients for the response lags | DEPLAG | PHI= |
| Fixes the values of lag coefficients to the specified initial values | DEPLAG | NOEST |

## PROC HPFUCMSPEC Statement

**PROC HPFUCMSPEC** *options* **;**

The following options can be used in the PROC HPFUCMSPEC statement.

**LABEL=***SAS-label*
specifies a descriptive label for the model specification to be stored in the SAS catalog or external file reference. The LABEL= option can also be specified as SPECLABEL=.

**NAME=***SAS-name*
names the model specification to be stored in the SAS catalog or external file reference. The NAME= option can also be specified as SPECNAME=.

**REPOSITORY=***SAS-catalog-name*
**REPOSITORY=***SAS-file-reference*
names the SAS catalog or external file reference to contain the model specification. The REPOSITORY= option can also be specified as MODELREPOSITORY=, MODELREP=, or REP=.

## AUTOREG Statement

**AUTOREG** < *options* > **;**

The AUTOREG statement specifies an *autoregressive* component of the model. An autoregressive component is a special case of cycle that corresponds to the frequency of zero or $\pi$. It is modeled separately for easier interpretation. A stochastic equation for an autoregressive component $r_t$ can be written as follows:

$$r_t = \rho r_{t-1} + \nu_t, \quad \nu_t \sim i.i.d.\ N(0, \sigma_\nu^2)$$

The damping factor $\rho$ can take any value in the interval (–1, 1), including –1 but excluding 1. If $\rho = 1$, the autoregressive component cannot be distinguished from the random walk level component. If $\rho = -1$, the autoregressive component corresponds to a seasonal component with season length 2 or a nonstationary cycle with period 2. If $|\rho| < 1$, then the autoregressive component is stationary. The following examples illustrate the AUTOREG statement:

```
autoreg;
```

This statement includes an autoregressive component in the model. The damping factor $\rho$ and the disturbance variance $\sigma_\nu^2$ are estimated from the data.

**NOEST=RHO**

**NOEST=VARIANCE**

**NOEST=( RHO VARIANCE )**

fixes the values of $\rho$ and $\sigma_v^2$ to those specified in RHO= and VARIANCE= options.

**RHO=*value***

supplies an initial value for the damping factor $\rho$ during the parameter estimation process. The value of $\rho$ must be in the interval (–1, 1), including –1 but excluding 1.

**VARIANCE=*value***

supplies an initial value for the disturbance variance $\sigma_v^2$ during the parameter estimation process. Any nonnegative value, including zero, is an acceptable starting value.

## BLOCKSEASON Statement

**BLOCKSEASON** *NBLOCKS= integer BLOCKSIZE= integer < options >* **;**

The BLOCKSEASON or BLOCKSEASONAL statement is used to specify a seasonal $\gamma_t$ that has a special block structure. The seasonal $\gamma_t$ is called a *block seasonal* of block size $m$ and number of blocks $k$ if its season length $s$ can be factored as $s = m * k$ and its seasonal effects have a block form—that is, the first $m$ seasonal effects are all equal to some number $\tau_1$, the next $m$ effects are all equal to some number $\tau_2$, and so on. This type of seasonal structure can be appropriate in some cases. For example, consider a series that is recorded on an hourly basis. Further assume that, in this particular case, the hour-of-the-day effect and the day-of-the-week effect are additive. In this situation the hour-of-the-week seasonality, having a season length of 168, can be modeled as a sum of two components. The hour-of-the-day effect is modeled using a simple seasonal of season length 24, while the day-of-the-week effect is modeled as a block seasonal that has the days of the week as blocks. This day-of-the-week block seasonal will have seven blocks, each of size 24. A block seasonal specification requires, at the minimum, the block size $m$ and the number of blocks in the seasonal $k$. These are specified using the BLOCKSIZE= and NBLOCKS= options, respectively. In addition, you might need to specify the position of the first observation of the series by using the OFFSET= option, if it is not at the beginning of one of the blocks. In the example just considered, this will correspond to a situation where the first series measurement is not at the start of the day. Suppose that the first measurement of the series corresponds to the hour between 6:00 and 7:00 a.m., which is the seventh hour within that day or at the seventh position within that block. This is specified as OFFSET=7.

The other options of this statement are very similar to the options in the SEASONAL statement. For example, a block seasonal can also be of one of the two types, DUMMY or TRIGONOMETRIC. There can be more than one block seasonal component in the model, each specified using a separate BLOCKSEASON statement. No two block seasonals in the model can have the same NBLOCKS= and BLOCKSIZE= specifications. The following example illustrates the use of the BLOCKSEASON statement to specify the additive, hour-of-the-week seasonal model:

```
season length=24 type=trig;
blockseason nblocks=7 blocksize=24;
```

The following options can be specified in the BLOCKSEASON statement.

**BLOCKSIZE=*integer***
specifies the block size, $m$ . This is a required option in this statement. The block size can be any integer larger than or equal to two. Typical examples of block sizes are 24 (which corresponds to the hours of the day when a day is being used as a block in hourly data) or 60 (which corresponds to the minutes in an hour when an hour is being used as a block in data recorded by minutes), and so on.

**NBLOCKS=*integer***
specifies the number of blocks, $k$. This is a required option in this statement. The number of blocks can be any integer larger than or equal to two.

**NOEST**
fixes the value of the disturbance variance parameter to the value specified in the VARIANCE= option.

**OFFSET=*integer***
specifies the position of the first measurement within the block, if the first measurement is not at the start of a block. The OFFSET= value must be between one and the block size. The default value is one. The *first measurement* refers to the start of the series.

**TYPE=DUMMY | TRIG**
specifies the type of the seasonal component. The default type is DUMMY.

**VARIANCE=*value***
supplies an initial value for the disturbance variance $\sigma_\omega^2$ in the $\gamma_t$ equation, at the start of the parameter estimation process. Any nonnegative value, including zero, is an acceptable starting value.

## CYCLE Statement

**CYCLE** < *options* > **;**

The CYCLE statement is used to specify a *cycle* component $\psi_t$ in the model. The stochastic equation that governs a cycle component of period $p$ and damping factor $\rho$ is as follows:

$$\begin{bmatrix} \psi_t \\ \psi_t^* \end{bmatrix} = \rho \begin{bmatrix} \cos\lambda & \sin\lambda \\ -\sin\lambda & \cos\lambda \end{bmatrix} \begin{bmatrix} \psi_{t-1} \\ \psi_{t-1}^* \end{bmatrix} + \begin{bmatrix} \nu_t \\ \nu_t^* \end{bmatrix}$$

where $\nu_t$ and $\nu_t^*$ are independent, zero-mean Gaussian disturbances with variance $\sigma_\nu^2$ and $\lambda = 2 * \pi / p$ is the angular frequency of the cycle. Any $p$ strictly larger than 2 is an admissible value for the period, and the damping factor $\rho$ can be any value in the interval (0, 1), including 1 but excluding 0. The cycles with the frequency zero and $\pi$, which correspond to the periods equal to infinity and two respectively, can be specified using the AUTOREG statement. The values of $\rho$ smaller than 1 give rise to a stationary cycle, while $\rho = 1$ gives rise to a nonstationary cycle. As a default, values of $\rho$, $p$, and $\sigma_\nu^2$ are estimated from the data. However, if necessary, you can fix the values of some, or all, of these parameters.

There can be multiple cycles in a model, each specified using a separate CYCLE statement. Currently, you can specify up to 50 cycles in a model.

The following examples illustrate the use of the CYCLE statement:

```
cycle;
cycle;
```

These statements request that two cycles be included in the model. The parameters of each of these cycles is estimated from the data.

```
cycle rho=1 noest=rho;
```

This statement requests inclusion of a nonstationary cycle in the model. The cycle period $p$ and the disturbance variance $\sigma_\nu^2$ are estimated from the data. In the following statement a nonstationary cycle with fixed period of 12 is specified. Moreover, a starting value is supplied for $\sigma_\nu^2$.

```
cycle period=12 rho=1 variance=4 noest=(rho period);
```

The following options can be specified in the CYCLE statement.

**NOEST=PERIOD**

**NOEST=RHO**

**NOEST=VARIANCE**

**NOEST= ( < RHO > < PERIOD > < VARIANCE > )**
fixes the values of the component parameters to those specified in the RHO=, PERIOD=, and VARIANCE= options. This option enables you to fix any combination of parameter values.

**PERIOD=*value***
supplies an initial value for the cycle period during the parameter estimation process. Period value must be strictly larger than 2.

**RHO=*value***
supplies an initial value for the damping factor in this component during the parameter estimation process. Any value in the interval (0, 1), including one but excluding zero, is an acceptable initial value for the damping factor.

**VARIANCE=*value***
supplies an initial value for the disturbance variance parameter $\sigma_\nu^2$ to be used during the parameter estimation process. Any nonnegative value, including zero, is an acceptable starting value.

## DEPLAG Statement

**DEPLAG** *LAGS = order* < *PHI* > = *value* . . . < *NOEST* > **;**

The DEPLAG statement is used to specify the lags of the forecast variable to be included as predictors in the model. The following example illustrates the use of DEPLAG statement:

```
deplag lags=2;
```

If the forecast series is denoted by $y_t$, this statement specifies the inclusion of $\phi_1 y_{t-1} + \phi_2 y_{t-2}$ in the model. The parameters $\phi_1$ and $\phi_2$ are estimated from the data. The following statement requests including $\phi_1 y_{t-1} + \phi_2 y_{t-4} - \phi_1\phi_2 y_{t-5}$ in the model. The values of $\phi_1$ and $\phi_2$ are fixed at 0.8 and –1.2.

```
deplag lags=(1)(4) phi=0.8 -1.2 noest;
```

The dependent lag parameters are not constrained to lie in any particular region. In particular, this implies that a UCM that contains only an *irregular* component and dependent lags, resulting in a traditional autoregressive model, is not constrained to be a stationary model. In the DEPLAG statement if an initial value is supplied for any one of the parameters, the initial values must be supplied for all other parameters also.

**LAGS=***order*

**LAGS=(***lag, . . . , lag* **) . . . (***lag, . . . , lag* **)**

**LAGS=(***lag, . . . , lag* **)<** $s_1$ **> . . . (***lag, . . . , lag* **)<** $s_k$ **>**

defines a model with specified lags. This is a required option in this statement. LAGS=($l_1, l_2, \ldots, l_k$) defines a model with specified lags of the forecast variable included as predictors. LAGS= *order* is equivalent to LAGS=(1, 2, . . . , *order* ).

A concatenation of parenthesized lists specifies a factored model. For example, LAGS=(1)(12) specifies that the lag values, 1, 12, and 13, corresponding to the following polynomial in the backward shift operator, be included in the model:

$$(1 - \phi_{1,1}B)(1 - \phi_{2,1}B^{12})$$

Note that, in this case, the coefficient of the thirteenth lag is constrained to be the product of the coefficients of the first and twelfth lags.

You can also specify a multiplier after a parenthesized list. For example, LAGS=(1)(1)12 is equivalent to LAGS=(1)(12), and LAGS=(1,2)4(1)12(1,2)24 is equivalent to LAGS=(4,8)(12)(24,48).

**NOEST**

fixes the values of the parameters to those specified in PHI= options.

**PHI=*value* . . .**

lists starting values for the coefficients of the lagged forecast variable.

## FORECAST Statement

**FORECAST** *options* **;**

The FORECAST statement specifies the symbolic name that represents the series to be forecast and also specifies an optional transformation to be applied to the series. The symbolic name is used in later steps to associate actual time series variables with the model specification when the specification is applied to data.

The following options can be specified in the FORECAST statement.

**(SYMBOL|VAR)=** *variable*

specifies a symbolic name for the forecast series. This symbol specification is optional. If the SYMBOL= option is not specified, *Y* is used as a default symbol.

**TRANSFORM=** *option*

specifies the transformation to be applied to the time series. The following transformations are provided:

| | |
|---|---|
| NONE | no transformation |
| LOG | logarithmic transformation |
| SQRT | square-root transformation |
| LOGISTIC | logistic transformation |
| BOXCOX(*n* ) | Box-Cox transformation with parameter number where number is between –5 and 5 |

When the TRANSFORM= option is specified, the time series must be strictly positive.

## INPUT Statement

**INPUT** *options* **;**

The INPUT statements specify the inputs in the model. A separate INPUT statement is needed for each of the inputs. In this statement you can specify the delay order, the differencing orders, and the Box-Cox type transformations associated with the input variable under consideration.

The following options can be specified in the INPUT statement.

**DELAY=***order*

specifies the delay (or lag) order for the input series.

**DIF=***order*

**DIF=** *( order1, order2, . . . )*

specifies the differencing orders for the input series.

**PREDEFINED=***option*

associates a predefined trend or a set of seasonal dummy variables with this transfer function. The SYMBOL= and PREDEFINED= options are mutually exclusive.

In the following list of options, let $t$ represent the observation count from the start of the period of fit for the model, and let $X_t$ be the value of the time trend variable at observation $t$.

| | |
|---|---|
| LINEAR | a linear trend, with $X_t = t - c$ |
| QUADRATIC | a quadratic trend, with $X_t = (t - c)^2$ |
| CUBIC | a cubic trend, with $X_t = (t - c)^3$ |
| INVERSE | an inverse trend, with $X_t = 1/t$ |
| SEASONAL | seasonal dummy variables. For a seasonal cycle of length $s$, the seasonal dummy regressors include $X_{i,t} : 1 \le i \le (s-1), 1 \le t \le n$ for models that include a level component, and $X_{i,t} : 1 \le i \le (s), 1 \le t \le n$ for models that do not include a level component. |

Each element of a seasonal dummy regressor is either zero or one, based on the following rule:

$$X_{i,t} = \begin{cases} 1 & \text{when } i = t \\ 0 & \text{otherwise} \end{cases}$$

**(SYMBOL|VAR)=***variable*

specifies a symbolic name for the input series. This symbol specification is optional. If the SYMBOL= option is not specified, then *X* is used as a default symbol. If there are multiple INPUT statements, then an attempt is made to generate a unique set of input symbols.

**TRANSFORM=***option*

specifies the transformation to be applied to the time series. The following transformations are provided:

| | |
|---|---|
| NONE | no transformation |
| LOG | logarithmic transformation |
| SQRT | square-root transformation |
| LOGISTIC | logistic transformation |
| BOXCOX(*n*) | Box-Cox transformation with parameter number where number is between –5 and 5 |

When the TRANSFORM= option is specified, the time series must be strictly positive.

## IRREGULAR Statement

**IRREGULAR** < *options* > **;**

The IRREGULAR statement is used to include an irregular component in the model. There can be at most one IRREGULAR statement in the model specification. The irregular component corresponds to the overall random error $\epsilon_t$ in the model; it is modeled as a sequence of independent, zero-mean Gaussian random variables with variance $\sigma_\epsilon^2$. The options in this statement enable you to specify the value of $\sigma_\epsilon^2$ and to output the forecasts of $\epsilon_t$. As a default, $\sigma_\epsilon^2$ is estimated using the data and the component forecasts are not saved or displayed. A few examples of the IRREGULAR statement are given next. In the first example the statement is in its simplest form, resulting in the inclusion of an irregular component with unknown variance.

```
irregular;
```

The following statement provides a starting value for $\sigma_\epsilon^2$, to be used in the nonlinear parameter estimation process.

```
irregular variance=4;
```

The following options can be specified in the IRREGULAR statement.

**NOEST**
fixes the value of $\sigma_\epsilon^2$ to the value specified in the VARIANCE= option.

**VARIANCE=***value*
supplies an initial value for $\sigma_\epsilon^2$ during the parameter estimation process. Any nonnegative value, including zero, is an acceptable starting value.

## LEVEL Statement

**LEVEL** < *options* > **;**

The LEVEL statement is used to include a level component in the model. The level component, either by itself or together with a slope component, forms the trend component $\mu_t$ of the model. If the slope component is absent, the resulting trend is a random walk (RW) specified by the following equations:

$$\mu_t = \mu_{t-1} + \eta_t, \quad \eta_t \sim i.i.d.\ N(0, \sigma_\eta^2)$$

If the slope component is present, signified by the presence of a SLOPE statement (see "SLOPE Statement" on page 534), a locally linear trend (LLT) is obtained. The equations of LLT are as follows:

$$\begin{aligned} \mu_t &= \mu_{t-1} + \beta_{t-1} + \eta_t, \quad \eta_t \sim i.i.d.\ N(0, \sigma_\eta^2) \\ \beta_t &= \beta_{t-1} + \xi_t, \qquad \xi_t \sim i.i.d.\ N(0, \sigma_\xi^2) \end{aligned}$$

In either case, the options in the LEVEL statement are used to specify the value of $\sigma_\eta^2$ and to request forecasts of $\mu_t$. The SLOPE statement is used for similar purposes in the case of slope $\beta_t$. The following examples illustrate the use of LEVEL statement. Assuming that a SLOPE statement is not added subsequently, a simple random walk trend is specified by the following statement:

```
level;
```

The following statements specify a locally linear trend with value of $\sigma_\eta^2$ fixed at 4. The value of $\sigma_\xi^2$, the disturbance variance in the slope equation, will be estimated from the data.

```
level variance=4 noest;
slope;
```

The following options can be specified in the LEVEL statement.

**NOEST**
fixes the value of $\sigma_\eta^2$ to the value specified in the VARIANCE= option.

**VARIANCE=***value*

supplies an initial value for $\sigma_\eta^2$, the disturbance variance in the $\mu_t$ equation, at the start of the parameter estimation process. Any nonnegative value, including zero, is an acceptable starting value.

## SEASON Statement

**SEASON** < *options* > **;**

The SEASON or the SEASONAL statement is used to specify a seasonal component $\gamma_t$ in the model. A seasonal component can be one of the two types, DUMMY or TRIGONOMETRIC. A DUMMY type seasonal with season length $s$ satisfies the following stochastic equation:

$$\sum_{i=0}^{s-1} \gamma_{t-i} = \omega_t, \qquad \omega_t \sim i.i.d.\ N(0, \sigma_\omega^2)$$

The equations for a TRIGONOMETRIC type seasonal are as follows:

$$\gamma_t = \sum_{j=1}^{[s/2]} \gamma_{j,t}$$

where $[s/2]$ equals $s/2$ if $s$ is even and equals $(s-1)/2$ if it is odd. The sinusoids $\gamma_{j,t}$ have frequencies $\lambda_j = 2\pi j/s$ and are specified by the matrix equation

$$\begin{bmatrix} \gamma_{j,t} \\ \gamma^*_{j,t} \end{bmatrix} = \begin{bmatrix} \cos\lambda_j & \sin\lambda_j \\ -\sin\lambda_j & \cos\lambda_j \end{bmatrix} \begin{bmatrix} \gamma_{j,t-1} \\ \gamma^*_{j,t-1} \end{bmatrix} + \begin{bmatrix} \omega_{j,t} \\ \omega^*_{j,t} \end{bmatrix}$$

where the disturbances $\omega_{j,t}$ and $\omega^*_{j,t}$ are assumed to be independent and, for fixed $j$, $\omega_{j,t}$ and $\omega^*_{j,t} \sim N(0, \sigma_\omega^2)$. If $s$ is even, then the equation for $\gamma^*_{s/2,t}$ is not needed and $\gamma_{s/2,t}$ is given by

$$\gamma_{s/2,t} = -\gamma_{s/2,t-1} + \omega_{s/2,t}$$

Note that, whether the seasonal type is DUMMY or TRIGONOMETRIC, there is only one parameter, the disturbance variance $\sigma_\omega^2$, in the seasonal model.

There can be more than one seasonal component in the model, necessarily with different season lengths. Each seasonal component is specified using a separate SEASON statement. A model with multiple seasonal components can easily become quite complex and can need large amounts of data and computing resources for its estimation and forecasting. Currently, at most three seasonals can be included in a model. The following examples illustrate the use of SEASON statement:

```
season length=4;
```

This statement specifies a DUMMY type (default) seasonal component with season length 4, corresponding to the quarterly seasonality. The disturbance variance $\sigma_\omega^2$ is estimated from the data. The following statement specifies a trigonometric seasonal with monthly seasonality. It also provides a starting value for $\sigma_\omega^2$.

```
season length=12 type=trig variance=4;
```

The following options can be specified in the SEASON statement.

**LENGTH=***integer*
This option is used to specify the season length $s$. The season length can be any integer larger than or equal to 2, or it can be "s," indicating a placeholder that will be substituted later with an appropriate value. The specification of season length is optional; the default is LENGTH=s. The use of specifications with a placeholder for season lengths is further explained in Example 17.3. Typical examples of season lengths are 12, corresponding to the monthly seasonality, or 4, corresponding to the quarterly seasonality.

**NOEST**
fixes the value of the disturbance variance parameter to the value specified in the VARIANCE= option.

**TYPE=DUMMY | TRIG**
specifies the type of the seasonal component. The default type is DUMMY.

**VARIANCE=***value*
supplies an initial value for the disturbance variance $\sigma_\omega^2$ in the $\gamma_t$ equation, at the start of the parameter estimation process. Any nonnegative value, including zero, is an acceptable starting value.

## SLOPE Statement

**SLOPE** < *options* > **;**

The SLOPE statement is used to include a slope component in the model. The slope component cannot be used without the level component. The level and slope specifications jointly define the trend component of the model. A SLOPE statement without the accompanying LEVEL statement is ignored. The equations of the trend, defined jointly by the level $\mu_t$ and slope $\beta_t$, are as follows:

$$\begin{aligned} \mu_t &= \mu_{t-1} + \beta_{t-1} + \eta_t, \quad \eta_t \sim i.i.d.\ N(0, \sigma_\eta^2) \\ \beta_t &= \beta_{t-1} + \xi_t, \qquad\quad \xi_t \sim i.i.d.\ N(0, \sigma_\xi^2) \end{aligned}$$

The SLOPE statement is used to specify the value of the disturbance variance $\sigma_\xi^2$ in the slope equation and to request forecasts of $\beta_t$. The following statements request that a locally linear trend be used in the model. The disturbance variances $\sigma_\eta^2$ and $\sigma_\xi^2$ are estimated from the data.

```
level;
slope;
```

You can request a locally linear trend with fixed slope using the following statements:

```
level;
slope variance=0 noest;
```

The following options can be specified in the SLOPE statement.

**NOEST**
fixes the value of the disturbance variance $\sigma_\xi^2$ to the value specified in the VARIANCE= option.

**VARIANCE=***value*
supplies an initial value for the disturbance variance $\sigma_\xi^2$ in the $\beta_t$ equation, at the start of the parameter estimation process. Any nonnegative value, including zero, is an acceptable starting value.

# Examples: HPFUCMSPEC Procedure

## Example 17.1: Some HPFUCMSPEC Syntax Illustrations

The following statements illustrate the HPFUCMSPEC syntax for some of the commonly needed modeling activities. Suppose that a variety of UCMs are to be fitted to a data set that contains a sales series as the forecast variable and several promotional events as predictor series. In all these cases the model repository work.mymodels is kept the same, and the models are named as *model1, model2,* and so on to ensure uniqueness. Note that in a given repository, the models must have unique names. The symbols for the forecast and input variables are *sales* and *promo1, promo2,* and so on, respectively.

```
/* BSM with two inputs */
proc hpfucmspec repository=mymodels
                name=model1;
   forecast symbol=sales transform=log;
   input symbol=promo1 delay=3;
   input symbol=promo2 dif=1;
   irregular;
   level;
   slope variance=0 noest; /* non-varying slope */
   season length=12 type=trig;
run;

/* Model with one cycle and Box-Cox transform */
proc hpfucmspec repository=mymodels
                name=model2;
   forecast symbol=sales transform=BoxCox(0.8);
   irregular;
   level;
   slope;
   cycle rho=1 noest=(rho); /* fixed damping factor */
run;
```

```
/* Unsaturated monthly seasonal */
proc hpfucmspec repository=mymodels
                name=model3;
   forecast symbol=sales transform=log;
   irregular;
   level;
   slope;
   cycle period=12 rho=1 noest=(period rho);
   cycle period=6  rho=1 noest=(period rho);
   cycle period=4  rho=1 noest=(period rho);
run;

/* Supply starting values for the parameters */
proc hpfucmspec repository=mymodels
                name=model4;
   forecast symbol=sales transform=log;
   irregular;
   level;
   slope variance=10;
   cycle period=12 rho=0.9 noest=(period);
   cycle period=6 noest=(period);
run;

title "Models Added to MYMODELS Repository";
proc catalog catalog=mymodels;
   contents;
run;
```

**Output 17.1.1** Listing of Models in MYMODELS Repository

**Models Added to MYMODELS Repository**

**Contents of Catalog WORK.MYMODELS**

| # | Name | Type | Create Date | Modified Date | Description |
|---|---|---|---|---|---|
| 1 | BSM1 | XML | 03/26/2014 08:31:18 | 03/26/2014 08:31:18 | Basic structural model with one input |
| 2 | MODEL1 | XML | 03/26/2014 08:31:19 | 03/26/2014 08:31:19 | |
| 3 | MODEL2 | XML | 03/26/2014 08:31:19 | 03/26/2014 08:31:19 | |
| 4 | MODEL3 | XML | 03/26/2014 08:31:19 | 03/26/2014 08:31:19 | |
| 5 | MODEL4 | XML | 03/26/2014 08:31:19 | 03/26/2014 08:31:19 | |

## Example 17.2: How to Include a UCM in a Model Selection List

One of the primary uses of the HPFUCMSPEC procedure is to add candidate UCMs to a model selection list that can be used by the HPFENGINE procedure (see Chapter 7, "The HPFENGINE Procedure"). The HPFUCMSPEC procedure is used to create the UCM specifications and the HPFSELECT procedure is used to add the specifications to a model selection list (see Chapter 15, "The HPFSELECT Procedure"). This example illustrates this scenario.

Here a series that consists of the yearly river flow readings of the Nile, recorded at Aswan (Cobb 1978), is studied. The data consists of readings from the years 1871 to 1970. This series is known to have had a shift in the level starting at the year 1899, and the years 1877 and 1913 are suspected to be outlying points.

The following DATA step statements read the data in a SAS data set and create dummy inputs for the shift in 1899 and the unusual years 1877 and 1913.

```
data nile;
   input riverFlow @@;
   year = intnx( 'year', '1jan1871'd, _n_-1 );
   format year year4.;
   if year >= '1jan1899'd then Shift1899 = 1.0;
   else Shift1899 = 0;
   if year = '1jan1913'd then Event1913 = 1.0;
   else Event1913 = 0;
   if year = '1jan1877'd then Event1877 = 1.0;
   else Event1877 = 0;
datalines;
   1120  1160  963  1210  1160  1160  813  1230   1370  1140
   995   935   1110 994   1020  960   1180 799    958   1140

... more lines ...
```

Three candidate models are specified, *m1, m2*, and *m3*. Out of these three models, *m1* is the simplest, which ignores the background information. Out of the other two models, *m2* uses only the shift in 1899, while *m3* uses all the three inputs. The following syntax shows how to specify these models and how to create a selection list that combines them with the HPFSELECT procedure. In the HPFSELECT procedure note the use of INPUTMAP option in the SPEC statement. It ties the symbolic variable names used in the HPFARIMASPEC procedure with the actual variable names in the data set. If the symbolic names were appropriate to start with, then the INPUTMAP option is not necessary.

```
proc hpfucmspec repository=mymodels
                name=m1;
   forecast symbol=y;
   irregular;
   level;
run;

proc hpfucmspec repository=mymodels
                name=m2;
   forecast symbol=y;
   irregular;
   level;
   input symbol=x1;
run;

proc hpfucmspec repository=mymodels
                name=m3;
   forecast symbol=y;
   irregular;
   level;
   input symbol=x1;
   input symbol=x2;
   input symbol=x3;
run;
```

The follow statements create a selection list that includes model specifications m1, m2 and m3.

```
proc hpfselect repository=mymodels
               name=myselect;

   spec m1 / inputmap(symbol=y var=riverFlow);

   spec m2 / inputmap(symbol=y var=riverFlow)
             inputmap(symbol=x1 var=Shift1899);

   spec m3 / inputmap(symbol=y var=riverFlow)
             inputmap(symbol=x1 var=Shift1899)
             inputmap(symbol=x2 var=Event1877)
             inputmap(symbol=x3 var=Event1913);
run;
```

This selection list can now be used in the HPFENGINE procedure for various types of analyses. The following syntax shows how to compare these models based on the default comparison criterion, mean absolute percentage error (MAPE). As expected, the model *m3* turns out to be the best of the three compared (see Figure 17.2.1).

```
proc hpfengine data=nile
               repository=mymodels
               globalselection=myselect
               lead=0
               print=(select);
   forecast riverFlow;
   input    Shift1899;
   input    Event1877;
   input    Event1913;
run;
```

**Output 17.2.1** Model Selection Based on the MAPE Criterion

**Models Added to MYMODELS Repository**

**The HPFENGINE Procedure**

| Model Selection Criterion = MAPE | | | |
|---|---|---|---|
| **Model** | **Statistic** | **Selected** | **Label** |
| **M1** | 13.096557 | No | UCM: Y = LEVEL + ERROR |
| **M2** | 11.978730 | No | UCM: Y = LEVEL + X1 + ERROR |
| **M3** | 10.532464 | Yes | UCM: Y = LEVEL + X1 + ...+ X3 + ERROR |

## Example 17.3: How to Create a Generic Seasonal Model Specification

In the case of many seasonal model specifications, it is possible to describe a generic specification that is applicable in a variety of situations just by changing the season length specifications at appropriate places. As an example consider the basic structural model, which is very useful for modeling seasonal data. The basic structural model for a monthly series can be specified using the following statements.

```
proc hpfucmspec repository=mymodels
                name=MonthlyBSM
                label=
      "Basic Structural Model For A Series With Season Length 12";
   forecast symbol=Y transform=log;
   irregular;
   level;
   slope;
   season type=trig length=12;
run;
```

It is easy to see that the same syntax is applicable to a quarterly series if the length in the SEASON specification is changed from 12 to 4. A generic specification that allows for late binding of season lengths can be generated by the following syntax:

```
proc hpfucmspec repository=mymodels
                name=GenericBSM
                label="Generic Basic Structural Model";
   forecast symbol=Y transform= log;
   irregular;
   level;
   slope;
   season type=trig length=s;
run;
```

In this syntax the length in the SEASON specification is changed from 12 to "s." This syntax creates a template for the basic structural model that is applicable to different season lengths. When the HPFENGINE procedure (which actually uses such model specifications to estimate the model and produce the forecasts) encounters such a "generic" specification, it automatically creates a proper specification by replacing the season length placeholder with the value implied by the ID variable or its SEASONALITY= option. The following example illustrates the use of this generic specification. It shows how the same specification can be used for monthly and quarterly series. The parameter estimates for monthly and quarterly series are given in Figure 17.3.1 and Figure 17.3.2, respectively.

```
   /* Create a selection list that contains
      the Generic Airline Model */
proc hpfselect repository=mymodels
               name=genselect;
   spec GenericBSM;
run;

proc hpfengine data=sashelp.air
               repository=mymodels
               globalselection=genselect
               print=(estimates);
   id date interval=month;
   forecast air;
run;
```

**Output 17.3.1** Parameter Estimates for the Monthly Series

**Models Added to MYMODELS Repository**

**The HPFENGINE Procedure**

**Parameter Estimates for GENERICBSM Model**

| Component | Parameter | Estimate | Standard Error | t Value | Approx Pr > \|t\| |
|---|---|---|---|---|---|
| IRREGULAR | ERROR VARIANCE | 0.0002344 | 0.0001079 | 2.17 | 0.0298 |
| LEVEL | ERROR VARIANCE | 0.0002983 | 0.0001057 | 2.82 | 0.0048 |
| SLOPE | ERROR VARIANCE | 8.4792E-13 | 6.2271E-10 | 0.00 | 0.9989 |
| SEASON | ERROR VARIANCE | 3.55769E-6 | 1.32347E-6 | 2.69 | 0.0072 |

```
   /* Create a quarterly series illustrating accumulating
      the monthly Airline series to quarterly */
proc timeseries data=sashelp.air out=Qair;
   id date interval=quarter;
   var air / accumulate=total;
run;

proc hpfengine data=Qair
               repository=mymodels
               globalselection=genselect
               print=(estimates);
   id date interval=quarter;
   forecast air;
run;
```

**Output 17.3.2** Parameter Estimates for the Quarterly Series

**Models Added to MYMODELS Repository**

**The HPFENGINE Procedure**

**Parameter Estimates for GENERICBSM Model**

| Component | Parameter | Estimate | Standard Error | t Value | Approx Pr > \|t\| |
|---|---|---|---|---|---|
| IRREGULAR | ERROR VARIANCE | 4.57317E-8 | 5.58257E-8 | 0.82 | 0.4127 |
| LEVEL | ERROR VARIANCE | 0.0006273 | 0.0001773 | 3.54 | 0.0004 |
| SLOPE | ERROR VARIANCE | 2.42795E-9 | . | . | . |
| SEASON | ERROR VARIANCE | 0.00002010 | 9.68227E-6 | 2.08 | 0.0379 |

# References

Cobb, G. W. (1978), "The Problem of the Nile: Conditional Solution to a Change Point Problem," *Biometrika*, 65, 243–251.

# Part III

# Forecasting Details

# Chapter 18
# Forecasting Process Summary

**Contents**


Background . . . . . **544**
Transactional Data . . . . . 544
Time Series Data . . . . . 545
Forecasting Models . . . . . 545
Forecasts . . . . . 549
Forecast Function (Scoring) . . . . . 550
Statistics of Fit . . . . . 551
Automatic Forecasting Process . . . . . **552**
Accumulation Step . . . . . 552
Interpretation Step . . . . . 552
Adjustment Step . . . . . 553
Diagnostic Step . . . . . 553
Model Selection Step . . . . . 554
Parameter Estimation Step . . . . . 554
Forecasting Step . . . . . 554
Evaluation Step . . . . . 555
Performance Step . . . . . 555
Forecast Function (Score File) Generation Step . . . . . 556
Automatic Forecasting Data . . . . . **556**
Automatic Forecasting Data Flow . . . . . 556
Forecast Scoring Data Flow . . . . . 556
Automatic Forecasting Information . . . . . **557**
Model Specification . . . . . 557
Model Selection List . . . . . 560
Selected Model Specification . . . . . 561
Fitted Model . . . . . 561
Forecast Function (Score File) . . . . . 561
Automatic Forecasting Information Flow . . . . . 562
Forecast Scoring Information Flow . . . . . 563
Automatic Forecasting Repositories . . . . . **563**
Event Repository . . . . . 563
Model Specification Repository . . . . . 564
Fitted Model Repository . . . . . 566
Forecast Results Repository . . . . . 567
Score Repository . . . . . 567

Automatic Forecasting System Flow . . . . . 567
Forecast Scoring System Flow . . . . . 569
Automatic Forecasting Archives . . . . . **570**
References . . . . . **571**

# Background

This chapter provides a brief theoretical background description of automatic forecasting. An introductory discussion of automatic forecasting topics can be found in Makridakis, Wheelwright, and Hyndman (1997); Brockwell and Davis (1996); Chatfield (2000). A more detailed discussion of time series analysis and forecasting can be found in Box, Jenkins, and Reinsel (1994); Hamilton (1994); Fuller (1995); Harvey (1994).

This chapter also provides a summary of the SAS Forecast Server Procedures forecasting process. Forecasting steps, data and information flows, and information repositories are explained in this chapter.

## Transactional Data

*Transactional data* are time-stamped data that are collected over time at no particular frequency. Some examples of transactional data are:

- Internet data
- point-of-sale (POS) data
- inventory data
- call center data
- trading data

Businesses often want to analyze transactional data for trends and seasonal variation. To analyze transactional data for trends and seasonality, statistics must be computed for each time period and season of concern. The frequency and the season might vary with the business problem. Various statistics can be computed for each time period and season. For example:

- web visits by hour and by hour of day
- sales per month and by month of year
- inventory draws per week and by week of month
- calls per day and by day of week
- trades per weekday and by weekday of week

## Time Series Data

*Time series data* are time-stamped data that are collected over time at a particular frequency. Some examples of time series data are:

- web visits per hour
- sales per month
- inventory draws per week
- calls per day
- trades per weekday

As you can see, the frequency that is associated with the time series varies with the problem at hand. The frequency or *time interval* can be hourly, daily, weekly, monthly, quarterly, yearly, or many other variants of the basic time intervals. The choice of frequency is an important modeling decision. This decision is especially true for automatic forecasting. For example, if you want to forecast the next four weeks, it is best to use weekly data rather than daily data. The forecast horizon in the former case is 4, while in the latter case it is 28.

Associated with each time series is a seasonal cycle or *seasonality*. For example, the length of seasonality for a monthly time series is usually assumed to be 12 because there are 12 months in a year. Likewise, the seasonality of a daily time series is usually assumed to be 7. The usual seasonality assumption might not always hold. For example, if a particular business' seasonal cycle is 14 days long, the seasonality is 14, not 7.

Time series that consist of mostly zero values (or a single value) are called interrupted or *intermittent time series*. These time series are mainly constant-valued except for relatively few occasions. Intermittent time series must be forecast dif ferently from non-intermittent time series.

## Forecasting Models

A skilled analyst can choose from a number of forecasting models. For automatic forecasting of large numbers of time series, only the most robust models should be used. The goal is not to have the analyst manually choose the very best model for forecasting each time series. The goal is to provide a list of *candidate models* that will forecast the large majority of the time series well. In general, when analysts have a large number of time series to forecast, they should use automatic forecasting for the low-valued forecasts; then they can spend a larger portion of their time dealing with high-valued forecasts or with low-valued forecasts that are problematic.

The candidate models that are used here are considered the most robust in the forecasting literature, and these models have proven their effectiveness over time. These models consider the local level, local trend, and local seasonal components of the time series. The term *local* describes the fact that these components evolve with time. For example, the local trend component might not be a straight line, but a trend line whose slope changes with time. In each of these models, there is an error or random component that models the uncertainty.

The components that are associated with these models are useful not only for forecasting but also for describing how the time series evolves over time. The forecasting model decomposes the series into its various components. For example, the local trend component describes the trend (up or down) at each point in time, and the final trend component describes the expected future trend. These forecasting models can also indicate departures from previous behavior or can be used to cluster time series.

The parameter estimates (*weights* or *component variances*) describe how fast the component is changing with time. Weights or component variances near 0 indicate a relative constant component; weights near 1 or large component variances indicate a relatively variable component. For example, a seasonal weight near 0 or a component variance near 0 represents a stable seasonal component, and a seasonal weight near 1 or a large component variance represents an unstable seasonal component. Parameter estimates should be optimized for each time series for best results.

### Local Level Models

The local level models forecast time series whose level (or mean) component varies with time. These models predict the local level for future periods. Conceptually, this can be expressed as

(series) = (local level) + (error)

Examples of local level models are simple exponential smoothing and local level unobserved component model. This model has one parameter (level), which describes how the local level evolves. The forecasts for the future periods are simply the final local level which is a constant.

### Local Trend Models

The local trend models forecast time series whose level or trend (slope) components vary with time. These models predict the local level and trend for future periods. Conceptually, this can be expressed as

(series) = (local level) + (local trend) + (error)

Examples of local trend models are double (Brown), linear (Holt), damped-trend exponential smoothing, and local trend unobserved component model. The double model has one parameter (defining both level and trend weights), the linear model has two parameters (level and trend weights), and the damped-trend model has three parameters (level, trend, and damping weights). The damping weight parameter dampens the trend over time. The forecasts for the future periods are a combination of the final local level and the final local trend.

### Local Seasonal Models

The local seasonal models forecast time series whose level or seasonal components vary with time. These models predict the local level and season for future periods. Conceptually, this can be expressed as

(series) = (local level) + (local season) + (error)

Examples of local seasonal models are seasonal exponential smoothing and the local seasonal unobserved component model. The seasonal model has two parameters (level and seasonal). The forecasts for the future periods are a combination of the final local level and the final local season.

## Local Models

The local models forecast time series whose level, trend, or seasonal components vary with time. These models predict the local level, trend, and seasonal component for future periods. Conceptually, this can be expressed in either an additive form,

(series) = (local level) + (local trend) + (local season) + (error)

or in a multiplicative form,

(series) = ((local level) + (local trend)) x (local season) + (error)

Examples of local models are the Winters method (additive or multiplicative) and the basic structural model. These models have three parameters (level, trend, and seasonal). The forecasts for the future periods are a combination of the final local level, the final local trend, and final local season.

## ARIMA Models

The autoregressive integrated moving average (ARIMA) models forecast time series whose level, trend, or seasonal properties vary with time. These models predict the future values of the time series by applying nonseasonal or seasonal polynomial filters to the disturbances. Using different types of polynomial filters permits the modeling of various properties of the time series. Conceptually, these models can be expressed as

(series) = disturbance filter (error)

Examples of ARIMA models are the exponentially weighted moving average (EWMA), moving average (MA) processes, integrated moving average (IMA) processes, autoregressive (AR) processes, integrated autoregressive (IAR) processes, and autoregressive moving average (ARMA) processes.

## Causal Models

Causal time series models forecast time series data that are influenced by causal factors. Input variables (regressor or predictor variables) and calendar events (indicator, dummy, or intervention variables) are examples of causal factors. These independent (exogenous) time series causally influence the dependent (response, endogenous) time series and therefore can aid the forecasting of the dependent time series.

Examples of causal time series models are autoregressive integrated moving average with exogenous inputs (ARIMAX), which are also known as transfer function models or dynamic regression models, and unobserved component models (UCM), which are also known as state-space models and structural time series models. Conceptually, an ARIMAX model of this form can be expressed as

(series) = transfer function filter (causal factors) + disturbance filter (error)

while a UCM model of this form can be expressed as

(series) = (local level) + (local trend) + (local season) + (causal factors) + (error)

These regression models are *dynamic* because they take into account the autocorrelation between observations that are recorded at different times. Dynamic regression includes and extends multiple linear regression (static regression).

Input variables are usually continuous-valued time series. They represent causal factors that influence the dependent time series throughout the time range. Examples of input variables are prices, temperatures, and other economic or natural factors. Input variables are contained in the time series data set.

Calendar events can be represented by indicator variables that are usually discrete-valued. They indicate when the causal factor influences the dependent time series. Zero values usually indicate the absence of the event, and nonzero values usually indicate the presence of the event. These dummy regressors can consist of pulses (points), steps (shifts), ramps, and temporary changes and combinations of these primitive shapes. The values of the indicator variable depend on the time interval. For example, if the calendar event is New Year's Day and the time interval is monthly, a pulse indicator variable is nonzero for each January and zero otherwise.

In addition to the causal factors, the causal model can contain components that are described in preceding sections: local level, local trend, and local seasonal. Causal models decompose the time series into causal factors and the local components. This decomposition is useful for demand analysis (promotional analysis and intervention analysis).

## Transformed Models

Except for the Winters method multiplicative model, the preceding forecasting models are linear; that is, the components must be added together to re-create the series. Because time series are not always linear with respect to these components, automatic forecasting must consider transformed versions of the preceding forecasting models. Some useful time series transformations are the following:

- logarithmic
- square-root
- logistic
- Box-Cox

For example, suppose the underlying process that generated the series has one of the following nonlinear forms:

| Form | Description |
|---|---|
| (series) = exp ( (local level) + (local trend) + (error) ) | exponential growth model |
| (series) = (local level) x (local season) x (error) | multiplicative error model |

Transforming the preceding nonlinear forms permits the use of a linear forecasting model:

| Model | Description |
|---|---|
| log(series) = (local level) + (local trend) + (error) | log local trend model |
| log(series) = log(local level) + log(local seasonal) + log(error) | log local seasonal model |

The preceding transformations can be applied only to positive-valued time series.

## Intermittent Demand Models

Intermittent demand models or interrupted time series models forecast intermittent time series data. Because intermittent series are mostly constant valued (usually 0) except on relatively few occasions, it is often easier to predict when the series departs and how much the series departs from this constant value rather than to predict the next value. An example of an intermittent demand model is Croston's method.

Intermittent demand models decompose the time series into two parts: the interval series and the size series. The interval series measures the number of time periods between departures. The size series measures the

magnitude of the departures. After this decomposition, each part is modeled and forecast independently. The interval forecast predicts when the next departure will occur. The size forecast predicts the magnitude of the next departure. After the interval and size predictions are computed, they are combined (predicted magnitude divided by predicted number of periods for the next departure) to produce a forecast for the average departure from the constant value for the next time period.

### External and User-Defined Models

In addition to the previously described general classes of exponential smoothing models, unobserved component models, autoregressive integrated moving average models, and intermittent demand models, SAS Forecast Server Procedures software allows for external models and user-defined models.

*External models* are used for forecasts that are provided external to the system. These external forecasts might have originated from an external statistical model from another software package, might have been provided by an outside organization (for example, marketing organization or government agency), or might be based on judgment. External models allow for the evaluation of external forecasts and for tests for unbiasedness.

*User-defined models* are external models that you implement by using the SAS programming language or the C programming language. For these models, you create your own computational algorithm to generate the forecasts. These models are considered external because they are not implemented in SAS Forecast Server Procedures software.

## Forecasts

Forecasts are time series predictions made for future periods. They are random variables and therefore have an associated probability distribution. For example, assuming a normal distribution, the forecasts for the next three months can be viewed as three "bell curves" that are progressively flatter (or wider). The mean or median of each forecast is called the *prediction*. The variance of each forecast is called the *prediction error variance*, and the square root of the variance is called the *prediction standard error*. The variance is computed from the forecast model parameter estimates and the model residual variance.

The forecast for the next future period is called the *one-step-ahead forecast*. The forecast for $h$ periods in the future is called the *h-step-ahead forecast*. The *forecast horizon* or *forecast lead* is the number of periods into the future for which predictions are made (one-step, two-step, . . . , $h$-step). The larger the forecast horizon, the larger the prediction error variance at the end of the horizon. For example, forecasting daily data four weeks into the future implies a forecast horizon of 28, whereas forecasting weekly data four weeks into the future implies a forecast horizon of only 4. The prediction standard error at the end of the horizon in the former case might be larger than the prediction standard error in the latter case.

The *confidence limits* are based on the prediction standard errors and a chosen confidence limit size. A confidence limit size of 0.05 results in 95% confidence limits. The confidence limits are often computed assuming a normal distribution, but other distributions could be used. As with the prediction standard errors, the width of the confidence limits increases with the forecast horizon. Once again, the forecast horizon of 28 will have wide confidence limits at the end of the horizon, representing greater uncertainty.

The *prediction error* is the difference between the actual value and the predicted value when the actual value is known. The prediction errors are used to calculate the statistics of fit that are described later. For transformed models, it is important to understand the difference between the model errors (residuals) and the prediction errors. The residuals measure the departure from the model in the transformed metric (log,

square root, and so on). The prediction errors measure the departure from the original series. You should not directly compare the model residuals of a transformed model and a nontransformed model when evaluating the model fit. You can compare the prediction errors between any two models because prediction errors are computed on the same metric.

Taken together, the predictions, prediction standard errors, and confidence limits at each period in the forecast horizon are the *forecasts*. Although many people use the term "forecast" to imply only prediction, a forecast is not one number for each future time period.

Using a transformed forecasting model requires the following steps:

1. The time series data are transformed.
2. The transformed time series data are fit using the forecasting model.
3. The forecasts are computed using the parameter estimates and the transformed time series data.
4. The forecasts (predictions, prediction standard errors, and confidence limits) are inverse transformed.

The naive inverse transformation results in *median forecasts*. Obtaining *mean forecasts* requires that the prediction and the prediction error variance both be adjusted based on the transformation. In addition, the model residuals will be different from the prediction errors because of this inverse transformation. If no transformation is used, the model residual and the prediction error will be the same, and likewise the mean and median forecast will be the same (assuming a symmetric disturbance distribution).

For causal models, the future values of the causal factors must be provided in order to forecast the time series. A causal factor is *deterministic* if its future values are known with certainty. A causal factor is *controllable* if its future values are under the control of the organization that produces the forecasts. A causal factor is *stochastic* if its future values are not known with certainty. If the causal factor is *stochastic*, it must also be forecast, and the uncertainty of its forecast (prediction standard errors) must be incorporated into the uncertainty of the time series forecast.

## Forecast Function (Scoring)

For causal models that include controllable causal factors, the predictions can be influenced by the future decisions made by the organization that produces the forecasts. Changing the future values of the controllable causal factors changes the forecasts. Organizations want to make decisions that benefit themselves. To help organizations make better decisions, the future values of the controllable causal factors can be varied to their benefit. The future values of the causal factors can be varied for scenario analysis (what-if analysis), stochastic optimization, or goal-seeking to aid proper decision-making.

In scenario analysis, the organization sets the future values of the causal factors to specific values and then evaluates the effect on the forecasts. In stochastic optimization, the organization algorithmically varies the future values of the causal factors to find the optimum of an objective function (profit, revenue, or cost function) based on the forecasts. In goal seeking, the organization algorithmically varies the future values of the causal factors in order to determine the values that achieve a certain goal (profit, revenue, or cost goal) based on the forecasts.

For example, suppose the following:

- An organization desires to predict the demand for a product or service.
- The demand is influenced by its sales price and by its advertising expenditures.
- These data are recorded over time.

The following types of analysis can be used to answer questions about the time series data:

- Scenario analysis can help answer the question "What happens to demand if the organization increases the sales price and decreases the advertising expenditures?"
- Stochastic optimization can help answer the question "What is the optimal sales price and advertising expenditure combination that maximizes profit?"
- Goal-seeking can help answer the question "What are the combinations of sales price and advertising expenditures that achieve a specified sales target?"

The sales price and advertising expenditures for a given time period can influence demand in future time periods. Static regression ignores these dynamic effects, which often leads to poor predictions, which in turn leads to poor decisions. Dynamic regression captures these dynamic effects and provides better predictions, which in turn facilitates better decisions.

*Forecast score files* (or forecast functions) summarize the time series model's parameter estimates and the final states (historical time series information). These files can be used to quickly generate the forecasts required for the iterative nature of scenario analysis, stochastic optimization, and goal-seeking computations. Since most of the computational effort associated with automatic forecasting is time series analysis, diagnostics, model selection, and parameter estimation, forecast scoring is relatively effortless. Therefore, forecast scoring makes the iterative nature of large scale decision-making more tractable.

The results of forecast scoring include the predictions, prediction standard errors, and the confidence limits. All of these results can be used in decision-making.

## Statistics of Fit

The *statistics of fit* evaluate how well a forecasting model performs by comparing the actual data to the predictions. For a given forecast model that has been fitted to the time series data, the model should be checked or evaluated to see how well it fits or forecasts the data. Commonly used statistics of fit are root mean square error (RMSE), mean absolute percentage error (MAPE), Akaike information criteria (AIC), and many others. The statistics of fit can be computed from the model residuals or the prediction errors.

When the full range of data is used to both fit and also evaluate the model, this is referred to as *in-sample evaluation*. When the most recent data are excluded for parameter estimation (holdout) and this *holdout sample* is used for evaluation, this is referred to as *holdout sample evaluation*. Holdout sample analysis is similar to *training* and *testing* of neural networks. A portion of the data is withheld from training (fit) and the withheld data (holdout) are used to test performance.

When a particular statistic of fit is used for forecast model selection, it is referred to as the *model selection criterion*. For example, if the MAPE (an often recommended choice) is used as a model selection criterion, the forecast model with smallest MAPE in the evaluation region (in-sample or holdout-sample) is chosen as the *best model*.

When a particular statistic of fit is used to judge how well the forecasting process is predicting the future, it is referred to as the *performance statistic*.

# Automatic Forecasting Process

*Automatic forecasting* is usually defined as forecasting without the aid of an analyst skilled in time series analysis techniques or as forecasting when the number of forecasts is too numerous for an analyst to investigate. Automatic forecasting is usually performed on each time series independently. For each time series and for each candidate model, the parameter estimates are optimized for best results. This means that several optimizations might be required for each time series.

## Accumulation Step

The *accumulation* of time-stamped data into time series data is based on a particular frequency. For example, time-stamped data can be accumulated to form hourly, daily, weekly, monthly, or yearly time series. Additionally, the method for accumulating the transactions within each time period is based on a particular statistic. For example, the sum, mean, median, minimum, maximum, standard deviation, and other statistics can be used to accumulate the transactions within a particular time period.

For automatic forecasting, accumulation is the most important decision because the software makes most of the remaining decisions. If weekly forecasts of the average of the transactions are needed, then the accumulation frequency should be weekly and the accumulation statistic should be the average.

Accumulating the transactional data on a relatively small time interval can require a long forecast horizon. For example, if the data are accumulated on an hourly basis and if it is desired to forecast one month into the future, the forecast horizon is very long and the width of the confidence limits will be very wide toward the end of the horizon. In this situation, the forecast content or usefulness of the forecast will be low.

## Interpretation Step

Once the time-stamped data has been accumulated, there might be no data recorded for certain time periods (resulting in missing values in the accumulated time series). These missing values can represent unknown values (and so they should remain missing) o r they can represent no activity (in which case they should be set to zero or some other appropriate value). Some transactional databases set missing data at the beginning or end of the time series to zero values. These zero values should be set to missin g values. Missing values and zero values need to be interpreted before analyzing the time series.

## Adjustment Step

Once the time-stamped data has been accumulated and interpreted, the time series to forecast might require adjustment prior to analysis or *pre-forecast adjustment*. By adjusting the time series for known systematic variations or deterministic comp onents, the underlying stochastic (unknown) time series process can be more readily identified and modeled.

Examples of systematic adjustments are currency-unit conversions, exchange rates, trading days, and other known systematic variations. Examples of deterministic adjustments are advanced bookings and reservations, contractual agreements, and other known co ntributions or deterministic components.

After analysis, the statistical forecast of the adjusted time series might require *post-forecast adjustment* to return forecasts in the original metric.

Typically the pre-forecast and post-forecast adjustments are operations that are inverses of each other. For example, to adjust a time series for exchange rates, it is often desirable to perform the following steps, in which division and multiplication ar e inverse operations of each other:

1. *Divide* the time series by the exchange rate.
2. Analyze and forecast the adjusted time series without regard to exchange rates.
3. Adjust the forecasts, *multiplying* by the exchange rate.

For another example, to adjust a time series for advanced bookings, it is often desirable to perform the following steps, in which subtraction and addition are inverse operations of each other:

1. *Subtract* the advanced bookings from the time series.
2. Analyze and forecast the adjusted time series without regard to advanced booking.
3. Adjust the forecasts, *adding* the advanced bookings.

Systematic variations or deterministic components are included in the time series data. Adjustments are data whose effect is *excluded* prior to statistical analysis. Causal factors are data whose effect is *included* with the statistical a nalysis.

## Diagnostic Step

Given the time series data, the time series diagnostics subset the potential list of candidate models to those that are judged appropriate to a particular time series. Time series that have trends (deterministic or stochastic) should be forecast with mode ls that have a trend component. Time series with seasonal trends (deterministic or stochastic) should be forecast with models that have a seasonal component. Time series that are nonlinear should be transformed for use with linear models. Time series that are intermittent should be forecast with intermittent models.

The importance of the diagnostics should not be underestimated. Applying a seasonal model to a nonseasonal time series, particularly one with a short history, can lead to over-parameterization or false seasonality.

Applying a linear model to a nonlinear t ime series can lead to underestimation of the growth (or decline). Applying a non-intermittent model to an intermittent series will result in predictions biased toward zero.

If it is known, *a priori*, that a time series has a particular characteristic, then the diagnostics should be overridden and the appropriate model should be used. For example, if the time series is known to be seasonal, the diagnostics should be o verridden to always choose a seasonal model.

There can be several causal factors that might or might not influence the dependent time series. The multivariate time series diagnostics determine which of the causal factors *significantly* influence the dependent time series. These diagnostics i nclude cross-correlation analysis and transfer function analysis.

Once again, if it is known, *a priori*, that a particular causal factor is known to influence the dependent time series, then the diagnostics should be overridden and the appropriate model should be used.

## Model Selection Step

After the candidate models have been subset by the diagnostics, each model is fit to the data (with the holdout sample excluded). After model fitting, the one-step-ahead forecasts are made in the fit region (in-sample) or the multistep-ahead forecasts are made in the holdout sample region (out-of-sample). The model selection criterion is used to select the best performing model from the appropriate subset of the candidate models. As described previously, the model selection criteria are statistics of fit.

If the length of the time series is short, holdout sample analysis might not be possible due to a lack of data. In this situation, the full range of the data should be used for fitting and evaluation; otherwise, holdout sample analysis is recommended.

## Parameter Estimation Step

After the best forecasting model is selected from the candidate models, the selected model is fit to the full range of the data to obtain the most accurate model parameter estimates. If you excluded the holdout sample in this step, you would be ignoring t he most recent and influential observations. Most univariate forecasting models are weighted averages of the past data, with the most recent having the greatest weight. After the model is selected, excluding the holdout sample can result in poor forecasts . Holdout sample analysis is used only for forecast model selection, not for forecasting.

## Forecasting Step

After the model parameters are estimated, forecasts (predictions, prediction standard errors, prediction errors, and confidence limits) are made using the model parameter estimates, the model residual variance, and the full range of data. If a model trans formation was used, the forecasts are inverse transformed on a mean or median basis.

When it comes to decision-making based on the forecasts, the analyst must decide whether to base the decision on the predictions, lower confidence limits, upper confidence limits, or the distribution (predictions and prediction standard errors). If there is a greater penalty for over-predicting, the lower confidence limit should be used. If there is a greater penalty for under-predicting, the upper confidence limit should be used. Often for inventory control decisions, the distribution (mean and variance) is important.

## Evaluation Step

After the forecasts are made, the in-sample statistics of fit are computed based on the one-step-ahead forecasts and the actual data. These statistics can be used to identify poorly fitting models prior to making business decisions based on these forecast s. If forecasts do not predict the actual data well, they can be flagged to signal the need for more detailed investigation by the analyst.

In addition to the statistics of fit, distribution and correlation analysis of the prediction errors can help evaluate the adequacy of the forecasting model.

## Performance Step

The previous steps are used to forecast the future. This ex-post forecast evaluation judges the performance of the forecasting model. After forecasting future periods, the actual data becomes available as time passes. For example, suppose that monthly for ecasts are computed for the next three months into the future. After three months pass, the actual data are available. The forecasts made three months ago can now be compared to the actual data of the last three months.

The availability of the new data begs the following questions:

- How well are you forecasting?
- Why are you forecasting poorly?
- If you were forecasting well before, what went wrong?

Some useful measures of forecast performance are the statistics of fit described in the section "Statistics of Fit" on page 551. When the statistics of fit are used for performance measures, the statistics are computed from the previous predictions and the newly available actual data in the forecast horizon. For example, the MAPE can be computed from the previous predictions and the newly available actual data in the three-month forecast horizon.

Another useful measure of forecast performance is determining whether the newly available data fall within the previous forecasts' confidence limits. For example, performance could be measured by whether or not the newly available actual data fall outside the previous forecasts' confidence limits in the three-month forecast horizon.

If the forecasts were judged to be accurate in the past, a poor performance measure (such as actual data outside the confidence limits) could also be indicative of a change in the underlying process. A change in behavior, an unusual event, or other depart ure from past patterns might have occurred since the forecasts were made.

Such departures from past trends might be normal and indicate the need to update the forecasting model selection for this variable, or they can be a warning of special circumstances that warrant further investigation.

Large departures from forecast can sometimes reflect data errors, changes in policies or data definitions (for example, what exactly is counted as sales), fraud, or a structural change in the market environment.

## Forecast Function (Score File) Generation Step

After the selected model is fit to the full range of the data, a summary of model parameter estimates and the final states (historical time series information) are stored in a forecast score file. Subsequent decision-making processes can the use the forecast score file for scenario (what-if) analysis, stochastic optimization, or goal-seeking.

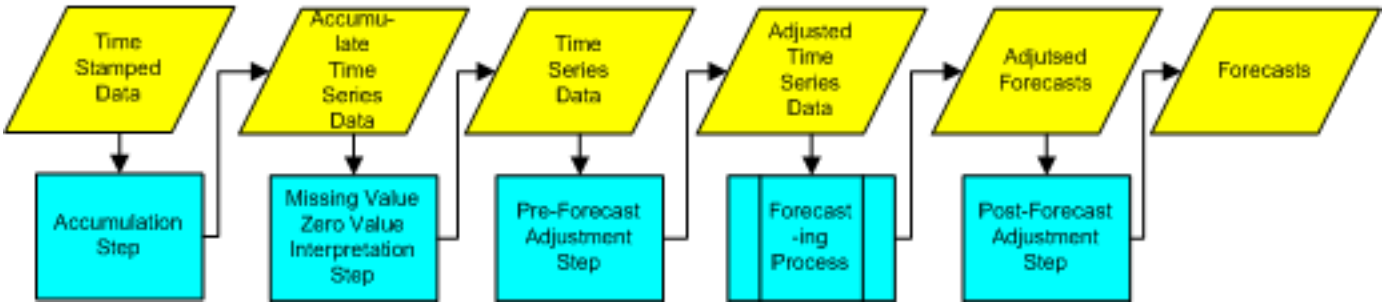
# Automatic Forecasting Data

For forecast scoring, you must specify the future values of the controllable causal factors (scenario analysis); otherwise, the future values must be iteratively generated by the decision process (stochastic optimization or goal-seeking).

## Automatic Forecasting Data Flow

The input and output of the automatic forecasting process are the time-stamped data set and the forecasts, respectively. Figure 18.1 depicts the automatic forecasting data flow.

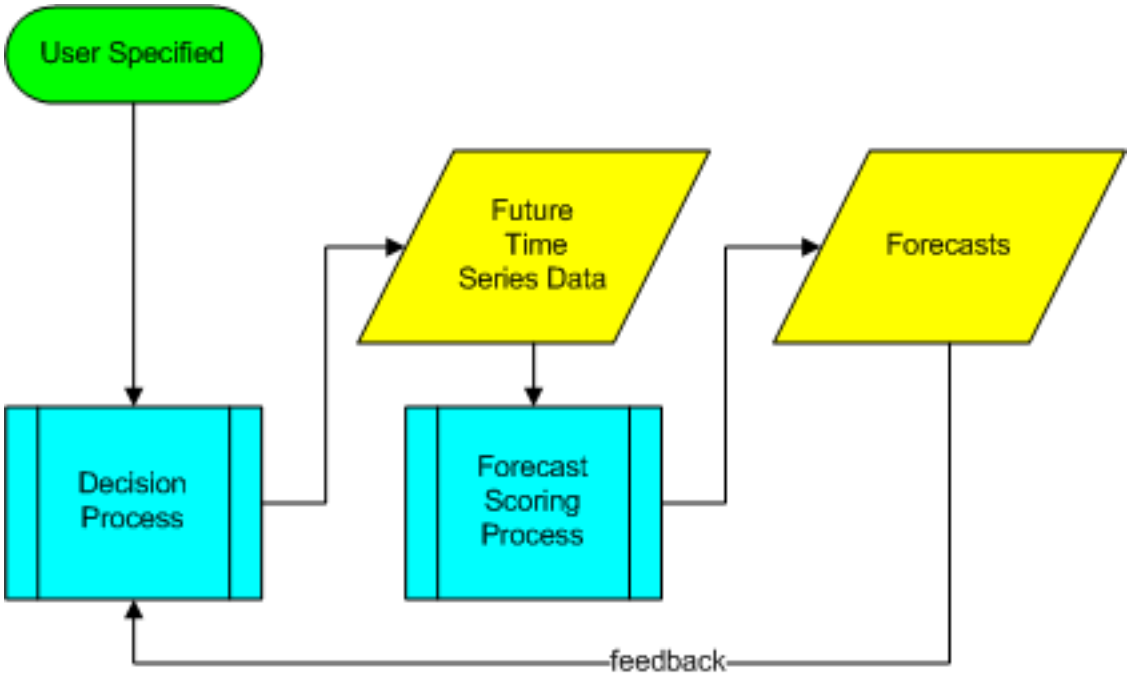
**Figure 18.1** Automatic Forecasting Data Flow




## Forecast Scoring Data Flow

The input and output of the forecast scoring process are the future values of the controllable causal factors and the forecasts, respectively. Figure 18.2 illustrates the forecast scoring data flow.

**Figure 18.2** Forecast Scoring Data Flow




# Automatic Forecasting Information

To automatically forecast a single time series, the time series must be diagnosed, selected, or specified to obtain a selected model used for forecasting. These abstract statistical concepts must be made concrete and persistent in a computer's storage. In addition to the time series data, the information described in the section "Model Specification" on page 557 is needed.

## Model Specification

A *model specification* indicates that a specific type of forecasting model be fit to the historical data and used to produce forecasts. Given a time series and a model specification, a forecast for the time series is generated by applying the abst ract statistical concepts associated with model specification. A model specification is not dependent on any specific time series data; a given specification can be used for many different series.

Associated with a model specification is a list of symbols that represent the time series to which the specification applies. These symbols must be mapped to actual time series variables in the input data set, or to event specifications, before the model specification can be used to created a fitted model for forecasting.

The following theoretical time series models are supported: ESM, IDM, ARIMAX, UCM, EXTERNAL, USER-DEFINED.

Except for the external and user-defined models, all of the models are implemented to allow nonlinear transformations (log, square root, logistic, Box-Cox) of the dependent and independent time series.

## Exponential Smoothing Models (the HPFESMSPEC Procedure)

Exponential smoothing models are extrapolation methods that predict future values based on exponentially weighted past values of the time series.

The following exponential smoothing models are supported:

- simple exponential smoothing (SIMPLE)
- double exponential smoothing (DOUBLE)
- linear exponential smoothing (LINEAR)
- damped-trend exponential smoothing (DAMPTREND)
- seasonal exponential smoothing (SEASONAL)
- multiplicative Winters method (WINTERS)
- additive Winters method (ADDWINTERS)

## Intermittent Demand Models (the HPFIDMSPEC Procedure)

Intermittent demand models are extrapolation methods that predict future values based on exponentially weighted past values of intermittent (interrupted) time series components. These methods use nonseasonal exponential smoothing models to forecast the in termittent time series components (interval, size, average demand) independently.

The following intermittent demand models are supported:

- Croston's method (CROSTON)
- average demand (AVERAGE)

## Autoregressive Moving Average with Exogenous Inputs (the HPFARIMASPEC Procedure)

ARIMAX models implement Box-Jenkins models with or without transfer function inputs.

The following ARIMA models are supported:

- simple and seasonal ARIMA
- factored and subset ARIMA
- preceding models with simple and seasonal transfer function inputs
- preceding models with factored and subset transfer function inputs

### Unobserved Component Models with Exogenous Inputs (the HPFUCMSPEC Procedure)

Unobserved component models (UCMs) implement structural time series models with or without input variables.

The following UCMs are supported:

- local level
- local slope (or trend)
- local seasons (up to three seasons)
- local cycles (no limit to the number of cycles)
- exogenous inputs
- combinations of the preceding components

### External Models (the HPFEXMSPEC Procedure)

External models are forecasts provided by methods external to the system. These methods might be judgmental inputs or forecasts provided by an external system such as another forecasting system. These forecasts must be recorded in the input time series da ta set.

When only the future predictions are provided, prediction standard errors and confidence limits are computed using the past prediction errors, if available. These additional forecasting components can be computed assuming nonlinear transformations (log, s quare root, logistic, Box-Cox) and autocorrelation (white noise, prediction error autocorrelation, series autocorrelation).

Because the system has no knowledge of how the forecasts were computed, there are no parameter estimates. However, the forecast bias can be computed, and a test for unbiasedness can be made.

### User-Defined Models

User-defined models are forecasting methods provided by the user of the system. These methods are implemented in the SAS language or the C language. Since the system has no knowledge of how the forecasts were computed, the forecasts are treated as if they were external forecasts.

There is usually more than one model specification associated with a model repository. A model specification does not depend on a particular time series, and a particular model specification can be assigned to different time series. However, a unique mode l specification must be assigned or selected for each time series in order to forecast the time series.

A model specification can also be referred to in one or more model selection lists.

The model specification is stored in an XML format, and this format follows the spirit of the predictive model markup language (PMML) specification. It is stored as a SAS catalog entry or as an external file.

See Chapter 23, “Using User-Defined Models,” for more information.

## Model Selection List

A model selection list specifies a list of candidate model specifications and how to choose which model specification is best suited to forecast a particular time series. Given a time series and an appropriate model selection list, a forecasting model can be automatically selected for the time series. Since the model selection process is applied to each series individually, the process might select a different model for different series and might select a different model for a given time series, with the passage of time as more data are collected. A model selection list is not dependent on any specific time series data.

A model selection list consists of the following:

| | |
|---|---|
| List of candidate model specifications | specifies the list of model specifications to consider when choosing the model for forecasting. |
| Selection diagnostics | specifies how to subset the list of model specifications models to those that are judged appropriate to a particular time series. |
| Holdout sample size | specifies the size of the holdout sample region. A holdout sample size of zero indicates that the full range of data is used to both fit and also evaluate the forecast. The holdout sample size can be an absolute size or a percentage of the length of the t ime series data. |
| Model selection criterion | specifies the statistic of fit to be used to select the best performing model from the subset list of the candidate models returned by the selection diagnostics. |
| Confidence limit size | specifies the confidence limit size for computing lower and upper confidence limits. |

There might be more than one model selection list associated with a model repository. A model selection list does not depend on a particular time series, and a particular model selection list can be assigned to different time series. However, a unique mod el selection list must be assigned to each time series. If desired, each time series to be forecast can have its own model selection list; typically, for time series with similar characteristics, the same model selection list is assigned.

The model selection list is stored in an XML format and this format follows the spirit of the PMML specification. It is stored as a SAS catalog entry or as an external file.

See Chapter 15, “The HPFSELECT Procedure,” for more information.

## Selected Model Specification

A selected model specification is a model specification that is the result of the diagnostic or model selection processes for a particular time series. The selected model specification is used to forecast this time series.

The file reference of the selected model specification is stored in a SAS data set.

## Fitted Model

A fitted model results from applying a model specification to specific time series data. Given a time series and a model specification (diagnosed, selected, or specified), the model parameter estimates can be optimized to fit the time series data. The f itted model is used to forecast this time series.

The parameter estimates associated with fitted models are stored in a SAS data set.

## Forecast Function (Score File)

A forecast model score file encodes the information that is needed to compute forecasts for a time series given the future values of the causal factors. Given a time series and a model specification (diagnosed, selected, or specified), a fitted time series model is estimated. Given a fitted model and a time series, a forecast model score file can be generated that efficiently encapsulates all information needed to forecast the series when future inputs are provided.

The forecast model score file is stored in an XML format that follows the spirit of the PMML score file. It is stored as a SAS catalog entry or as an external file.

Forecast model score files can be used for scenario analysis, goal seeking, or stochastic optimization. SAS functions are provided that can refer to the forecast model score files to calculate forecasts from the fitted model given alternative inputs. These functions can be used in user-written SAS data set programs or in SAS analytical procedures such as the MODEL procedure or the NLP procedure.

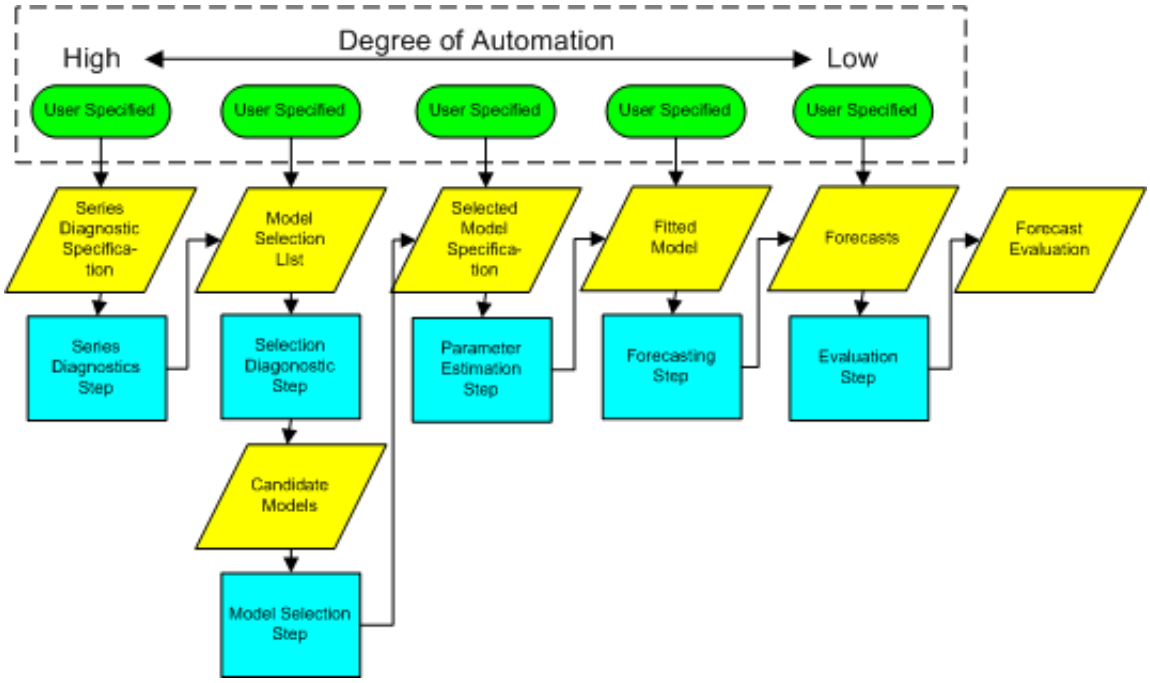
For examples and additional information, see Chapter 22, “Using Forecasting Model Score Files and DATA Step Functions.”

## Automatic Forecasting Information Flow

SAS Forecast Server Procedures software is designed to support fully automated forecasting. SAS Forecast Server Procedures software also provides you a great deal of control over the forecasting process when you want to override steps in the automatic process. You can control any or all of the forecasting steps, or you can allow the system to control all steps.

The degree of automation depends on how much information you specify. For each time series, the information flow of the automatic forecasting technique is described in Figure 18.3.

**Figure 18.3** Automatic Forecasting Information Flow



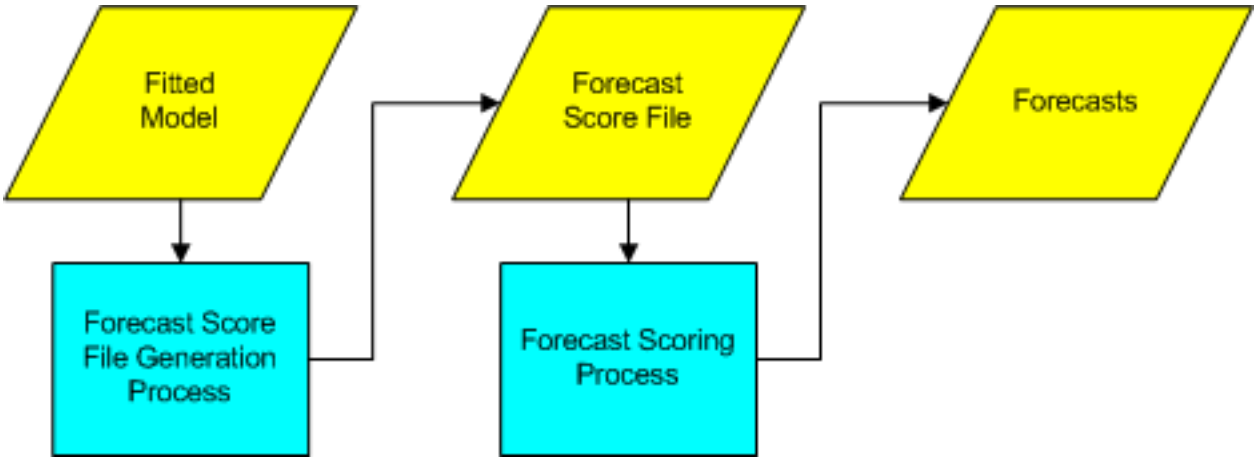

The more information you provide, the less automation is needed. If you specify the forecasts (external forecasts), nothing is required. If you specify the fitted model, only forecasting is required. If you specify the selected model specification, then parameter estimation and forecasting are required. If you specify a model selection list, then model selection, parameter estimation, and forecasting are required. If the diagnostic specification is specified, then diagnostics, model selection, parameter estimation, and forecasting are required. If you specify nothing, the default diagnostics or model selection list is used.

The more information you provide, the less computational effort is needed. Series diagnostics are the most expensive, followed by model selection, parameter estimation, and forecasting. Because the information persists in the computer's storage, differing degrees of automation can be used over time. For instance, it might be desirable to use the diagnostic step every six months, the model selection step every three months, the parameter estimation step every month, and the forecasting step every week. This staggering of the degree of automation reduces the processing time by allowing the most up-to-date information about the time series data to influence the automatic forecasting process over time.

## Forecast Scoring Information Flow

For each time series, the information flow of the forecast scoring technique presented here is described in Figure 18.4.

**Figure 18.4** Forecast Scoring Information Flow




A fitted model generates a forecast score file. Using the forecast score file and given the future values of the controllable causal factors, the forecast scoring process generates the forecasts.

# Automatic Forecasting Repositories

Since there are many time series to forecast, large-scale automatic forecasting requires the efficient management of large amounts of information about each time series. In addition to the time series data, the following information repositories are needed.

## Event Repository

An event repository stores information about calendar events with a brief description of each event. Calendar events can be represented by indicator variables that could be stored in the time series data. However, because the influential calendar events can vary from series to series, there might be too many to store efficiently and many calendar events will be redundant, making updates difficult. Therefore, it is better to store a brief description of the calendar event, to reproduce the indicator variable in the computer's memory when needed, and to store the calendar events independently of the time series data, to allow the reuse and update of the calendar events. Additionally, the event repository can be used by more than one time-stamped data set.

See Chapter 9, "The HPFEVENTS Procedure," for more information about creating event definitions and storing them in an event repository.

## Model Specification Repository

A *model specification repository* stores information about time series models (model specification) and how to select an appropriate time series model (model selection list) when given a particular time series. A model specification can be assigned to each time series. However, because the model specification can vary from series to series, there might be too many to store efficiently and many model specifications will be redundant, making updates difficult. Therefore, it is better to store model specifications independently of the time series data to allow the reuse and update of the model specification. Additionally, the model specification repository can be used by more than one time-stamped data set.

The model specification repository contains the following information:

| | |
|---|---|
| model specification file | SAS catalog entry or external file that *specifies a time series model to use* for forecasting |
| model selection list file | SAS catalog entry or external file that specifies *how to select a model specification* to use for a particular time series |

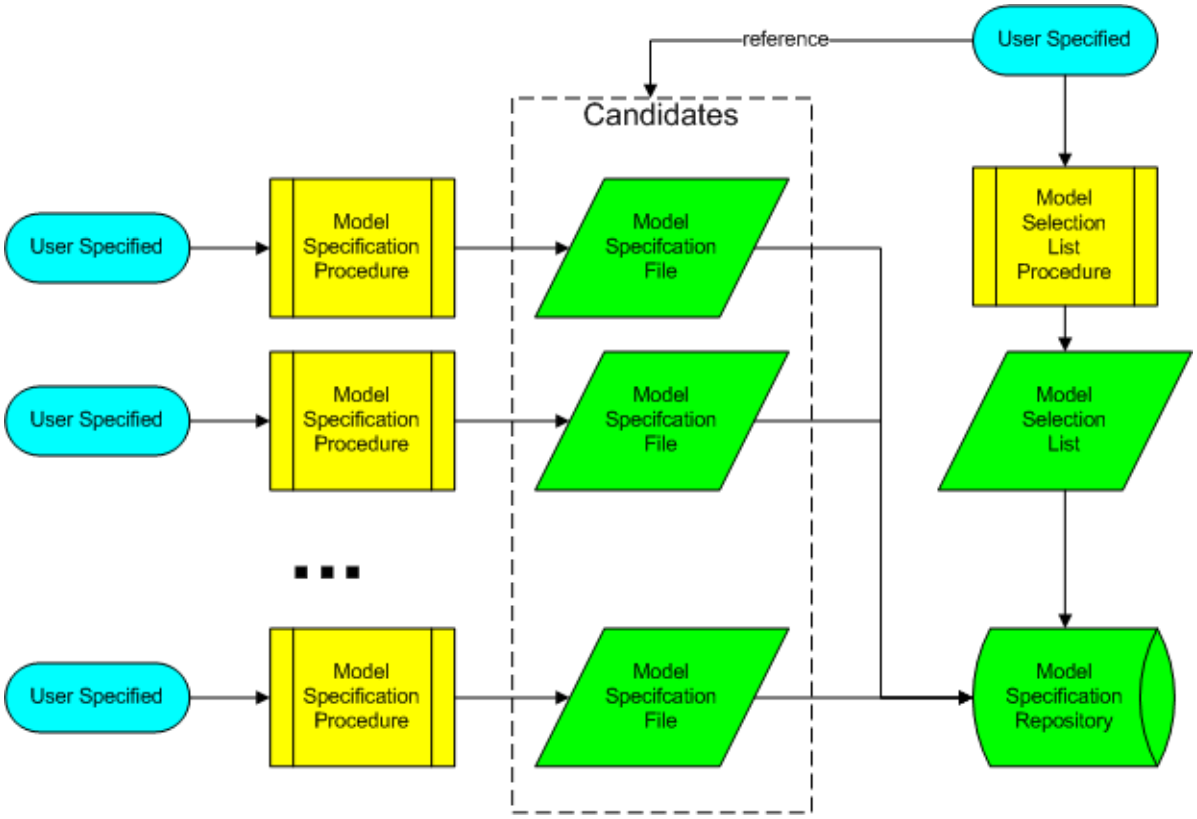
The repository consists of SAS catalogs or external directories (or folders). More than one catalog can be combined using the SAS Libname Engine.

### Creating a Model Specification Repository

You can create model specification files and populate the model specification repository by using the HPFESMSPEC, HPFIDMSPEC, HPFARIMASPEC, HPFUCMSPEC, and HPFEXMSPEC procedures. After creating the model specification files, you can create model selection list files by using the HPFSELECT procedure.

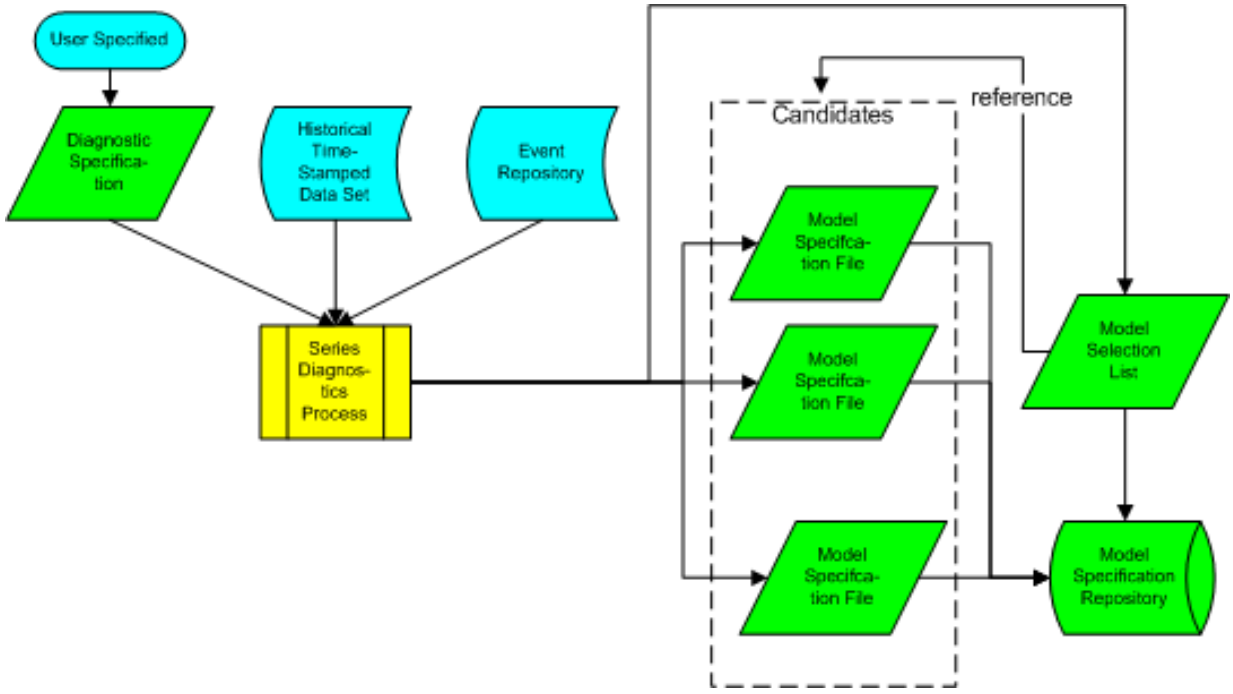
Figure 18.5 illustrates the process of adding user-created model specification files and model selection list files.

**Figure 18.5** Creating a Model Specification Repository



You can also create the model specification files and model selection files by using the HPFDIAGNOSE procedure. Given the historical time series data and the calendar events, the HPFDIAGNOSE procedure automatically creates model specification files and model selection files. Figure 18.6 illustrates the series diagnostic process of automatically creating model specification files and model selection list files.

**Figure 18.6** Series Diagnostic Process




## Fitted Model Repository

A fitted model repository stores information about the selected model specification and its parameter estimates for each time series. Because each time series has different parameter estimates, the fitted model repository will often be large. There is one fitted model repository for each time-stamped data set.

The repository consists of a single SAS data set and associated SAS catalogs or external files that are referenced in the rows of the data set. The forecasting model repository is generated using the series diagnostics or default model selection lists.

For each time series, the fitted model repository specifies the following:

model selection list name (reference) — SAS catalog entry name or external file name for the model selection list used to select this model. These lists are contained in a model specification repository.

| | |
|---|---|
| model specification name (reference) | SAS catalog entry name or external file name that specifies the current model being used for forecasting. These specifications are contained in the model specification repository. |
| variable mapping | data set rows that map the time series data specification variables and events to model specification symbols |
| forecasting model parameter estimates | data set rows that contain the model parameter estimates associated with the current model |
| forecast score name (reference) | SAS catalog entry name or external file name that specifies the forecast scores associated with the current model. These scores are stored in the forecast score repository. |

## Forecast Results Repository

A forecast results repository stores information about the forecasts, forecast evaluations, and forecast performance for each time series. The forecast results repository consists of several data sets. Because each time series has forecasts and statistics of fit associated with these forecasts, the forecast results repository will often be large. There is one forecast results repository for each time-stamped data set.

## Score Repository

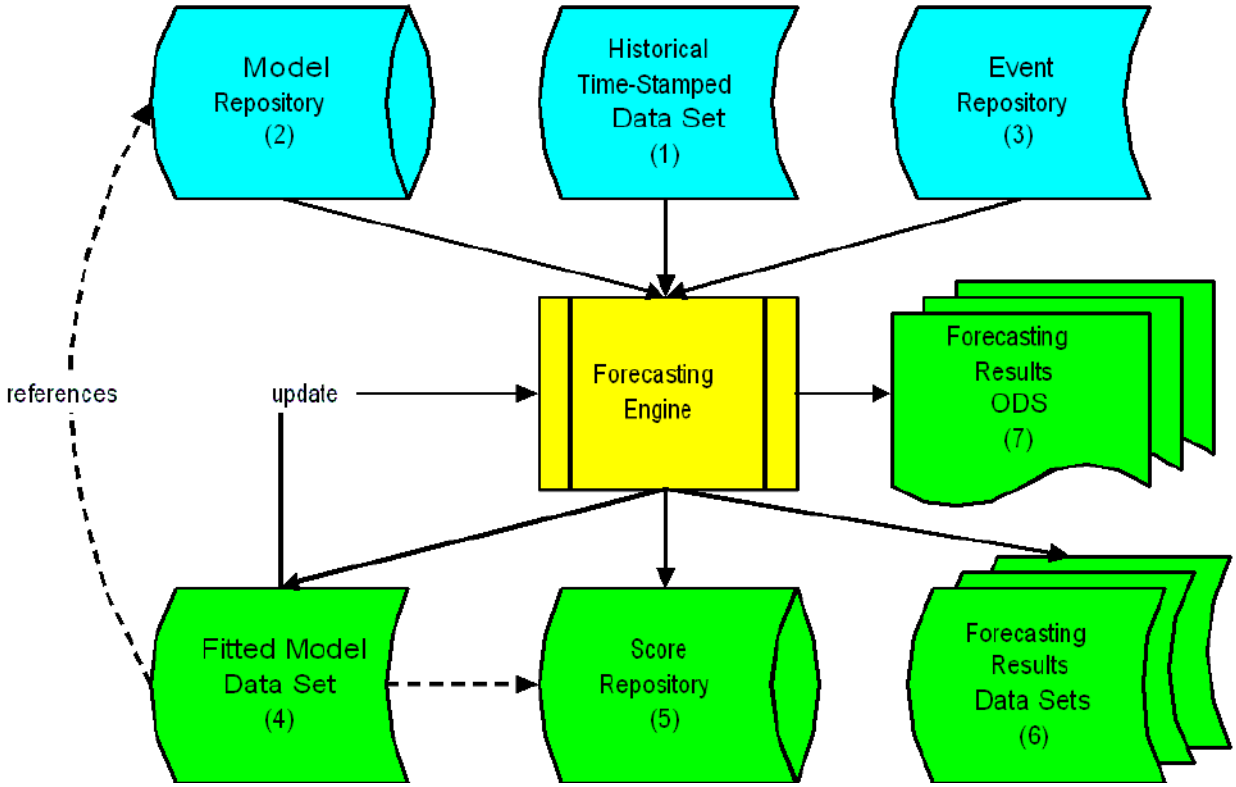
A score repository stores information about how to score each time series. Because each time series has a different score, the score repository will often be large because it summarizes information contained in the model specification repository, fitted model repository, and the final states (historical time series data). There is one score repository for each time-stamped data set.

## Automatic Forecasting System Flow

Along with the time series data, the preceding information repositories are needed for large-scale automatic forecasting. Figure 18.7 shows the system flow for the automatic forecasting technique when there are many time series.

following diagram:

**Figure 18.7** Automatic Forecasting System Flow



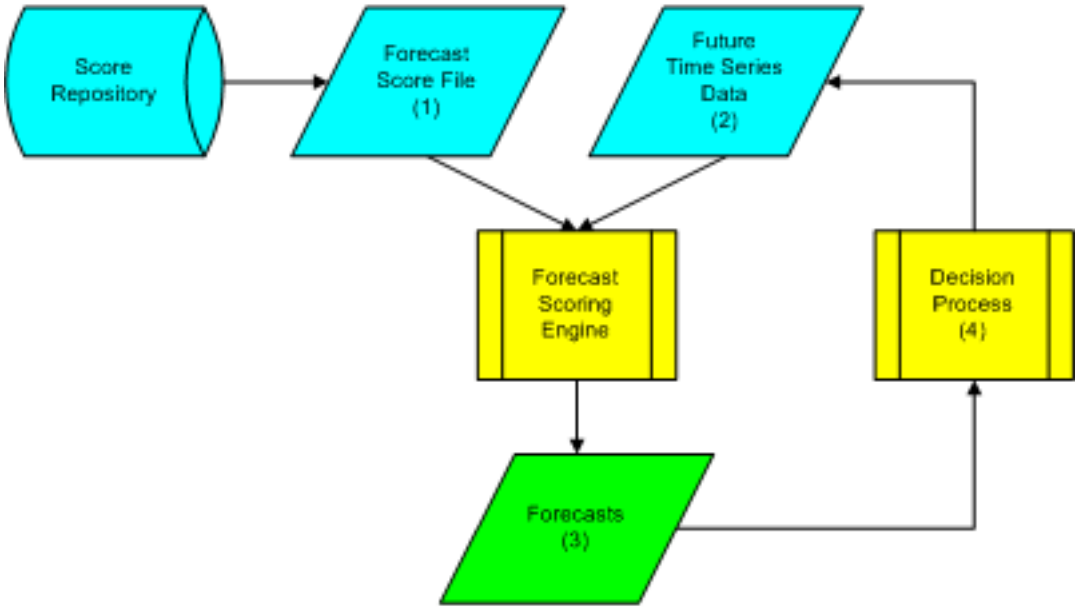

For each historical time series to forecast, the automatic forecasting system works as follows:

1. The time-stamped data are read from the time-stamped data set and accumulated, interpreted, and adjusted to form the time series to forecast.
2. The modeling information (model specifications and model selection list) associated with the time series is read from the model repository.
3. The calendar events associated with each model specification are read from the event repository.
4. Using the time series, modeling information, and calendar events, the forecasting engine creates or uses (updates) the fitted model.
5. From the fitted model, forecast score files are generated and stored in the score repository.
6. From the fitted model, forecast results data sets are created and stored.
7. From the fitted model, forecasting results ODS (printed tables and graphs) are created and rendered.

## Forecast Scoring System Flow

For each time series, the automatic forecasting system generates a forecast score file that can be used in subsequent decision-making processes. Figure 18.8 shows the system flow for each file that uses the forecast scoring technique.

**Figure 18.8** Forecast Scoring System Flow




For each time series to score, the forecast scoring process works as follows:

1. The forecast score file is read from the score repository.
2. The future values of the controllable causal factors are provided by the decision-making process.
3. Using the forecast score file and the future values, the forecast scoring process generates forecast results.
4. Steps 2 and 3 are repeated as needed by the decision-making process.

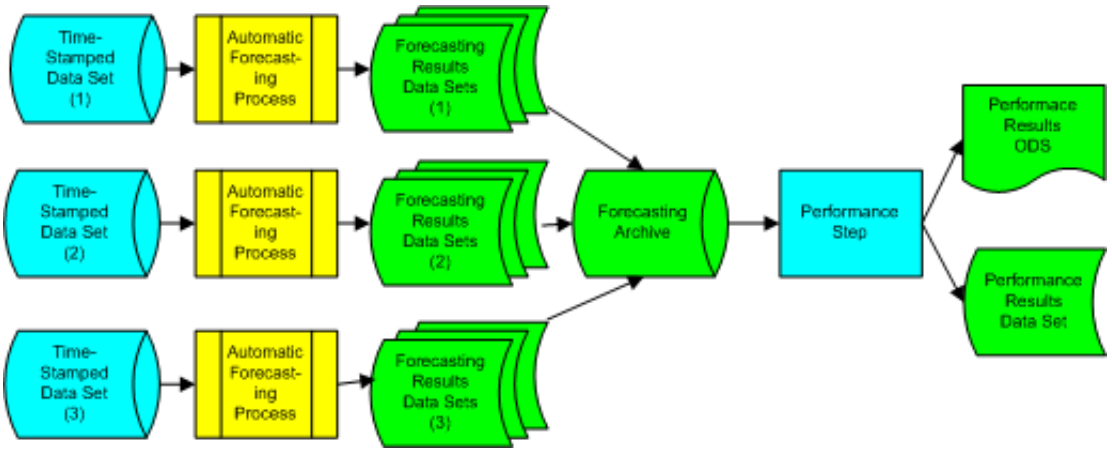
The automatic forecasting process that creates the forecast scoring file is significantly more computationally expensive than the forecast scoring process. The forecast score file needs to be created only once, whereas the iterative nature of decision-mak ing processes might require many scores.

## Automatic Forecasting Archives

Since automatic forecasting is used to forecast over time, the automatic forecasting process must be monitored for accuracy (quality control) or *forecast performance*. Therefore, the forecasts, generated over time, must be archived to measure fore cast performance. Likewise, the forecast performance must be archived over time.

The automatic forecasting archives are shown in Figure 18.9.

**Figure 18.9** Automatic Forecasting Archives




At each forecast origin (or time the forecast is created), forecasts are created from the time-stamped data observed up to the forecast origin. The forecasts from the forecast origin through the forecast horizon are recorded in the *forecasting arc hive*. The forecast archive contains the historical forecasts as well as their forecast origins. The forecast archive can be evaluated to measure the historical performance.

# References

Box, G. E. P., Jenkins, G. M., and Reinsel, G. C. (1994), *Time Series Analysis: Forecasting and Control*, 3rd Edition, Englewood Cliffs, NJ: Prentice-Hall.

Brockwell, P. J. and Davis, R. A. (1996), *Introduction to Time Series and Forecasting*, New York: Springer-Verlag.

Chatfield, C. (2000), *Time-Series Forecasting*, Boca Raton, FL: Chapman & Hall/CRC.

Fuller, W. A. (1995), *Introduction to Statistical Time Series*, New York: John Wiley & Sons.

Hamilton, J. D. (1994), *Time Series Analysis*, Princeton, NJ: Princeton University Press.

Harvey, A. C. (1994), *Time Series Models*, Cambridge, MA: MIT Press.

Makridakis, S. G., Wheelwright, S. C., and Hyndman, R. J. (1997), *Forecasting: Methods and Applications*, New York: John Wiley & Sons.

# Chapter 19
# Forecasting Process Details

## Contents


Forecasting Process Summary . . . . . **574**
- Parameter Estimation . . . . . 574
- Model Evaluation . . . . . 574
- Forecasting . . . . . 574

Smoothing Models . . . . . **575**
- Smoothing Model Calculations . . . . . 575
- Smoothing State and Smoothing Equations . . . . . 575
- Smoothing State Initialization . . . . . 576
- Missing Values . . . . . 576
- Predictions and Prediction Errors . . . . . 576
- Smoothing Weights . . . . . 577
- Specifying the Smoothing Weights . . . . . 577
- Optimizing the Smoothing Weights . . . . . 577
- Equations for the Smoothing Models . . . . . 578

ARIMA Models . . . . . **589**
- ARIMA Model Based Series Decomposition . . . . . 589

Unobserved Component Models . . . . . **590**

Intermittent Models . . . . . **591**
- Intermittent Time Series . . . . . 591
- Intermittent Series Decomposition and Analysis . . . . . 591
- Croston's Method . . . . . 592
- Average Demand Method . . . . . 593
- Time-Indexed versus Demand-Indexed Holdout Samples . . . . . 594
- Automatic Intermittent Demand Model Selection . . . . . 594

External Models . . . . . **595**
- External Forecasts . . . . . 595
- External Forecast Prediction Errors . . . . . 595
- External Forecast Prediction Bias . . . . . 595
- External Forecast Prediction Standard Errors . . . . . 596
- External Forecast Confidence Limits . . . . . 597

Series Transformations . . . . . **598**
- Predictions for Transformed Models . . . . . 598

Series Diagnostic Tests . . . . . **599**

Statistics of Fit . . . . . **600**

References . . . . . **603**

This chapter provides computational details on several aspects of the SAS Forecast Server Procedures software.

# Forecasting Process Summary

This section summarizes the forecasting process. You can use a variety of forecasting models to forecast a series with the SAS Forecast Server Procedures software. The final choice of model depends on the type of analysis needed and whether any predictor variables will be used in the forecasting. The available model types are ARIMA models, unobserved component models (UCMs), smoothing models, and intermittent models. The ARIMA model and UCM can use the predictor variables, whereas the smoothing and intermittent models do not involve predictor variables.

## Parameter Estimation

Computational details for the smoothing and intermittent models are provided in the sections "Smoothing Models" on page 575 and "Intermittent Models" on page 591. The details for ARIMA and UCM modeling are given in Chapter 7, "The ARIMA Procedure" (*SAS/ETS User's Guide*) and Chapter 34, "The UCM Procedure" (*SAS/ETS User's Guide*), respectively. The results of the parameter estimation process are printed in the "Parameter Estimates" table or stored in the OUTEST= data set.

## Model Evaluation

Model evaluation is based on the one-step-ahead prediction errors for observations within the period of evaluation. The one-step-ahead predictions are generated from the model specification and parameter estimates. The predictions are inverse transformed (median or mean) and adjustments are removed. The prediction errors (the difference of the dependent series and the predictions) are used to compute the statistics of fit, which are described in section "Statistics of Fit" on page 600. The results generated by the evaluation process are printed in the "Statistics of Fit" table or stored in the OUTSTAT= data set.

## Forecasting

The forecasting process is similar to the model evaluation process described in the preceding section, except that $k$-step-ahead predictions are made from the end of the data through the specified forecast horizon and prediction standard errors and confidence limits are calculated. The forecasts and confidence limits are printed in the "Forecast" table or stored in the OUTFOR= data set.

# Smoothing Models

This section details the computations performed for the exponential smoothing and Winters method forecasting models.

## Smoothing Model Calculations

For the descriptions and properties of various smoothing methods see: Gardner (1985); Chatfield (1978); Bowerman and O'Connell (1979). This section summarizes the smoothing model computations.

Given a time series $\{Y_t : 1 \leq t \leq n\}$, the underlying model assumed by the smoothing models has the following (additive seasonal) form:

$$Y_t = \mu_t + \beta_t t + s_p(t) + \epsilon_t$$

where

| | |
|---|---|
| $\mu_t$ | represents the time-varying mean term. |
| $\beta_t$ | represents the time-varying slope. |
| $s_p(t)$ | represents the time-varying seasonal contribution for one of the $p$ seasons. |
| $\epsilon_t$ | are disturbances. |

For smoothing models without trend terms, $\beta_t = 0$; and for smoothing models without seasonal terms, $s_p(t) = 0$. Each smoothing model is described in the following sections.

At each time $t$, the smoothing models estimate the time-varying components described above with the *smoothing state*. After initialization, the smoothing state is updated for each observation by using the *smoothing equations*. The smoothing state at the last nonmissing observation is used for predictions.

## Smoothing State and Smoothing Equations

Depending on the smoothing model, the *smoothing state* at time $t$ consists of the following:

$L_t$ is a smoothed level that estimates $\mu_t$.

$T_t$ is a smoothed trend that estimates $\beta_t$.

$S_{t-j}$, $j = 0, \ldots, p-1$, are seasonal factors that estimate $s_p(t)$.

The smoothing process starts with an initial estimate of the smoothing state, which is subsequently updated for each observation by using the *smoothing equations*.

The smoothing equations determine how the smoothing state changes as time progresses. Knowledge of the smoothing state at time $t-1$ and that of the time-series value at time $t$ uniquely determine the smoothing

state at time $t$. The *smoothing weights* determine the contribution of the previous smoothing state to the current smoothing state. The smoothing equations for each smoothing model are listed in section "Equations for the Smoothing Models" on page 578.

## Smoothing State Initialization

Given a time series $\{Y_t : 1 \leq t \leq n\}$, the smoothing process first computes the smoothing state for time $t = 1$. However, this computation requires an initial estimate of the smoothing state at time $t = 0$, even though no data exists at or before time $t = 0$.

An appropriate choice for the initial smoothing state is made by backcasting from time $t = n$ to $t = 1$ to obtain a prediction at $t = 0$. The initialization for the backcast is obtained by regression with constant and linear terms and seasonal dummy variables (additive or multiplicative) as appropriate for the smoothing model. For models with linear or seasonal terms, the estimates obtained by the regression are used for initial smoothed trend and seasonal factors; however, the initial smoothed level for backcasting is always set to the last observation, $Y_n$.

The smoothing state at time $t = 0$ obtained from the backcast is used to initialize the smoothing process from time $t = 1$ to $t = n$ (Chatfield and Yar 1988).

For models with seasonal terms, the smoothing state is normalized so that the seasonal factors $S_{t-j}$ for $j = 0, \ldots, p-1$ sum to zero for models that assume additive seasonality and average to one for models (such as Winters method) that assume multiplicative seasonality.

## Missing Values

When a missing value is encountered at time $t$, the smoothed values are updated using the *error-correction form* of the smoothing equations with the one-step-ahead prediction error $e_t$ set to zero. The missing value is estimated using the one-step-ahead prediction at time $t-1$, that is $\hat{Y}_{t-1}(1)$ (Aldrin and Damsleth 1989). The error-correction forms of each of the smoothing models are listed in the sections "ARIMA Models" on page 589, "Unobserved Component Models" on page 590, "Intermittent Models" on page 591, "External Models" on page 595, "Series Transformations" on page 598, "Series Diagnostic Tests" on page 599, and "Statistics of Fit" on page 600.

## Predictions and Prediction Errors

Predictions are made based on the last known smoothing state. Predictions made at time $t$ for $k$ steps ahead are denoted $\hat{Y}_t(k)$, and the associated prediction errors are denoted $e_t(k) = Y_{t+k} - \hat{Y}_t(k)$. The *prediction equation* for each smoothing model is listed in the sections "ARIMA Models" on page 589, "Unobserved Component Models" on page 590, "Intermittent Models" on page 591, "External Models" on page 595, "Series Transformations" on page 598, "Series Diagnostic Tests" on page 599, and "Statistics of Fit" on page 600.

The *one-step-ahead predictions* refer to predictions made at time $t-1$ for one time unit into the future—that is, $\hat{Y}_{t-1}(1)$—and the *one-step-ahead prediction errors* are more simply denoted $e_t = e_{t-1}(1) = Y_t - \hat{Y}_{t-1}(1)$. The one-step-ahead prediction errors are also the model residuals, and

the sum of squares of the one-step-ahead prediction errors is the objective function used in smoothing weight optimization.

The *variance of the prediction errors* are used to calculate the confidence limits (Sweet 1985; McKenzie 1986; Yar and Chatfield 1990; Chatfield and Yar 1991). The equations for the variance of the prediction errors for each smoothing model are listed in the sections "ARIMA Models" on page 589, "Unobserved Component Models" on page 590, "Intermittent Models" on page 591, "External Models" on page 595, "Series Transformations" on page 598, "Series Diagnostic Tests" on page 599, and "Statistics of Fit" on page 600.

Note: $var(\epsilon_t)$ is estimated by the mean square of the one-step-ahead prediction errors.

## Smoothing Weights

Depending on the smoothing model, the smoothing weights consist of the following:

| | |
|---|---|
| $\alpha$ | is a level smoothing weight. |
| $\gamma$ | is a trend smoothing weight. |
| $\delta$ | is a seasonal smoothing weight. |
| $\phi$ | is a trend damping weight. |

Larger smoothing weights (less damping) permit the more recent data to have a greater influence on the predictions. Smaller smoothing weights (more damping) give less weight to recent data.

## Specifying the Smoothing Weights

Typically the smoothing weights are chosen to be from zero to one. (This is intuitive because the weights associated with the past smoothing state and the value of current observation would normally sum to one.) However, each smoothing model (except Winters method—multiplicative version) has an ARIMA equivalent. Weights chosen to be within the ARIMA additive-invertible region guarantee stable predictions (Archibald 1990; Gardner 1985). The ARIMA equivalent and the additive-invertible region for each smoothing model are listed in the sections "ARIMA Models" on page 589, "Unobserved Component Models" on page 590, "Intermittent Models" on page 591, "External Models" on page 595, "Series Transformations" on page 598, "Series Diagnostic Tests" on page 599, and "Statistics of Fit" on page 600.

## Optimizing the Smoothing Weights

Smoothing weights are determined so as to minimize the sum of squared one-step-ahead prediction errors. The optimization is initialized by choosing from a predetermined grid the initial smoothing weights that result in the smallest sum of squared, one-step-ahead prediction errors. The optimization process is highly dependent on this initialization. It is possible that the optimization process will fail due to the inability to obtain stable initial values for the smoothing weights (Greene 1993), and it is possible for the optimization to result in a local minima.

The optimization process can result in weights being chosen outside both the zero-to-one range and the ARIMA additive-invertible region. By restricting weight optimization to additive-invertible region, you can obtain a local minimum with stable predictions. Likewise, weight optimization can be restricted to the zero-to-one range or other ranges.

#### Standard Errors

The standard errors associated with the smoothing weights are calculated from the Hessian matrix of the sum of squared, one-step-ahead prediction errors with respect to the smoothing weights used in the optimization process.

#### Weights near Zero or One

Sometimes the optimization process results in weights near zero or one.

For simple or double (Brown) exponential smoothing, a level weight near zero implies that simple differencing of the time series might be appropriate.

For linear (Holt) exponential smoothing, a level weight near zero implies that the smoothed trend is constant and that an ARIMA model with deterministic trend might be a more appropriate model.

For damped-trend linear exponential smoothing, a damping weight near one implies that linear (Holt) exponential smoothing might be a more appropriate model.

For Winters method and seasonal exponential smoothing, a seasonal weight near one implies that a nonseasonal model might be more appropriate and a seasonal weight near zero implies that deterministic seasonal factors might be present.

## Equations for the Smoothing Models

### Simple Exponential Smoothing

The model equation for simple exponential smoothing is

$$Y_t = \mu_t + \epsilon_t$$

The smoothing equation is

$$L_t = \alpha Y_t + (1-\alpha)L_{t-1}$$

The error-correction form of the smoothing equation is

$$L_t = L_{t-1} + \alpha e_t$$

**NOTE:** For missing values, $e_t = 0$.

The $k$-step prediction equation is

$$\hat{Y}_t(k) = L_t$$

The ARIMA model equivalency to simple exponential smoothing is the ARIMA(0,1,1) model

$$(1 - B)Y_t = (1 - \theta B)\epsilon_t$$

$$\theta = 1 - \alpha$$

The moving-average form of the equation is

$$Y_t = \epsilon_t + \sum_{j=1}^{\infty} \alpha\epsilon_{t-j}$$

For simple exponential smoothing, the additive-invertible region is

$$\{0 < \alpha < 2\}$$

The variance of the prediction errors is estimated as

$$var(e_t(k)) = var(\epsilon_t)\left[1 + \sum_{j=1}^{k-1} \alpha^2\right] = var(\epsilon_t)(1 + (k-1)\alpha^2)$$

## Double (Brown) Exponential Smoothing

The model equation for double exponential smoothing is

$$Y_t = \mu_t + \beta_t t + \epsilon_t$$

The smoothing equations are

$$L_t = \alpha Y_t + (1 - \alpha)L_{t-1}$$

$$T_t = \alpha(L_t - L_{t-1}) + (1 - \alpha)T_{t-1}$$

This method can be described equivalently in terms of two successive applications of simple exponential smoothing:

$$S_t^{[1]} = \alpha Y_t + (1-\alpha) S_{t-1}^{[1]}$$

$$S_t^{[2]} = \alpha S_t^{[1]} + (1-\alpha) S_{t-1}^{[2]}$$

where $S_t^{[1]}$ are the smoothed values of $Y_t$ and $S_t^{[2]}$ are the smoothed values of $S_t^{[1]}$. The prediction equation then takes the form:

$$\hat{Y}_t(k) = (2 + \alpha k/(1-\alpha)) S_t^{[1]} - (1 + \alpha k/(1-\alpha)) S_t^{[2]}$$

The error-correction form of the smoothing equations is

$$L_t = L_{t-1} + T_{t-1} + \alpha e_t$$

$$T_t = T_{t-1} + \alpha^2 e_t$$

**NOTE:** For missing values, $e_t = 0$.

The $k$-step prediction equation is

$$\hat{Y}_t(k) = L_t + ((k-1) + 1/\alpha) T_t$$

The ARIMA model equivalency to double exponential smoothing is the ARIMA(0,2,2) model

$$(1-B)^2 Y_t = (1-\theta B)^2 \epsilon_t$$

$$\theta = 1 - \alpha$$

The moving-average form of the equation is

$$Y_t = \epsilon_t + \sum_{j=1}^{\infty} (2\alpha + (j-1)\alpha^2) \epsilon_{t-j}$$

For double exponential smoothing, the additive-invertible region is

$$\{0 < \alpha < 2\}$$

The variance of the prediction errors is estimated as

$$var(e_t(k)) = var(\epsilon_t)\left[1 + \sum_{j=1}^{k-1}(2\alpha + (j-1)\alpha^2)^2\right]$$

## Linear (Holt) Exponential Smoothing

The model equation for linear exponential smoothing is

$$Y_t = \mu_t + \beta_t t + \epsilon_t$$

The smoothing equations are

$$L_t = \alpha Y_t + (1-\alpha)(L_{t-1} + T_{t-1})$$

$$T_t = \gamma(L_t - L_{t-1}) + (1-\gamma)T_{t-1}$$

The error-correction form of the smoothing equations is

$$L_t = L_{t-1} + T_{t-1} + \alpha e_t$$

$$T_t = T_{t-1} + \alpha\gamma e_t$$

**NOTE:** For missing values, $e_t = 0$.

The $k$-step prediction equation is

$$\hat{Y}_t(k) = L_t + kT_t$$

The ARIMA model equivalency to linear exponential smoothing is the ARIMA(0,2,2) model

$$(1-B)^2 Y_t = (1-\theta_1 B - \theta_2 B^2)\epsilon_t$$

$$\theta_1 = 2-\alpha-\alpha\gamma$$

$$\theta_2 = \alpha - 1$$

The moving-average form of the equation is

$$Y_t = \epsilon_t + \sum_{j=1}^{\infty}(\alpha + j\alpha\gamma)\epsilon_{t-j}$$

For linear exponential smoothing, the additive-invertible region is

$$\{0 < \alpha < 2\}$$

$$\{0 < \gamma < 4/\alpha - 2\}$$

The variance of the prediction errors is estimated as

$$var(e_t(k)) = var(\epsilon_t)\left[1 + \sum_{j=1}^{k-1}(\alpha + j\alpha\gamma)^2\right]$$

## Damped-Trend Linear Exponential Smoothing

The model equation for damped-trend linear exponential smoothing is

$$Y_t = \mu_t + \beta_t t + \epsilon_t$$

The smoothing equations are

$$L_t = \alpha Y_t + (1-\alpha)(L_{t-1} + \phi T_{t-1})$$

$$T_t = \gamma(L_t - L_{t-1}) + (1-\gamma)\phi T_{t-1}$$

The error-correction form of the smoothing equations is

$$L_t = L_{t-1} + \phi T_{t-1} + \alpha e_t$$

$$T_t = \phi T_{t-1} + \alpha\gamma e_t$$

**NOTE:** For missing values, $e_t = 0$.

The $k$-step prediction equation is

$$\hat{Y}_t(k) = L_t + \sum_{i=1}^{k} \phi^i T_t$$

The ARIMA model equivalency to damped-trend linear exponential smoothing is the ARIMA(1,1,2) model

$$(1 - \phi B)(1 - B)Y_t = (1 - \theta_1 B - \theta_2 B^2)\epsilon_t$$

$$\theta_1 = 1 + \phi - \alpha - \alpha\gamma\phi$$

$$\theta_2 = (\alpha - 1)\phi$$

The moving-average form of the equation (assuming $|\phi| < 1$) is

$$Y_t = \epsilon_t + \sum_{j=1}^{\infty} (\alpha + \alpha\gamma\phi(\phi^j - 1)/(\phi - 1))\epsilon_{t-j}$$

For damped-trend linear exponential smoothing, the additive-invertible region is

$$\{0 < \alpha < 2\}$$

$$\{0 < \phi\gamma < 4/\alpha - 2\}$$

The variance of the prediction errors is estimated as

$$var(e_t(k)) = var(\epsilon_t)\left[1 + \sum_{j=1}^{k-1} (\alpha + \alpha\gamma\phi(\phi^j - 1)/(\phi - 1))^2\right]$$

## Seasonal Exponential Smoothing

The model equation for seasonal exponential smoothing is

$$Y_t = \mu_t + s_p(t) + \epsilon_t$$

The smoothing equations are

$$L_t = \alpha(Y_t - S_{t-p}) + (1-\alpha)L_{t-1}$$

$$S_t = \delta(Y_t - L_t) + (1-\delta)S_{t-p}$$

The error-correction form of the smoothing equations is

$$L_t = L_{t-1} + \alpha e_t$$

$$S_t = S_{t-p} + \delta(1-\alpha)e_t$$

**NOTE:** For missing values, $e_t = 0$.

The $k$-step prediction equation is

$$\hat{Y}_t(k) = L_t + S_{t-p+k}$$

The ARIMA model equivalency to seasonal exponential smoothing is the ARIMA(0,1,p+1)(0,1,0)$_p$ model

$$(1-B)(1-B^p)Y_t = (1-\theta_1 B - \theta_2 B^p - \theta_3 B^{p+1})\epsilon_t$$

$$\theta_1 = 1-\alpha$$

$$\theta_2 = 1-\delta(1-\alpha)$$

$$\theta_3 = (1-\alpha)(\delta-1)$$

The moving-average form of the equation is

$$Y_t = \epsilon_t + \sum_{j=1}^{\infty} \psi_j \epsilon_{t-j}$$

$$\psi_j = \begin{cases} \alpha & \text{for } j \mod p \neq 0 \\ \alpha + \delta(1-\alpha) & \text{for } j \mod p = 0 \end{cases}$$

For seasonal exponential smoothing, the additive-invertible region is

$$\{\max(-p\alpha, 0) < \delta(1-\alpha) < (2-\alpha)\}$$

The variance of the prediction errors is estimated as

$$var(e_t(k)) = var(\epsilon_t)\left[1 + \sum_{j=1}^{k-1} \psi_j^2\right]$$

## Multiplicative Seasonal Smoothing

In order to use the multiplicative version of seasonal smoothing, the time series and all predictions must be strictly positive.

The model equation for the multiplicative version of seasonal smoothing is

$$Y_t = \mu_t s_p(t) + \epsilon_t$$

The smoothing equations are

$$L_t = \alpha(Y_t / S_{t-p}) + (1-\alpha)L_{t-1}$$

$$S_t = \delta(Y_t / L_t) + (1-\delta)S_{t-p}$$

The error-correction form of the smoothing equations is

$$L_t = L_{t-1} + \alpha e_t / S_{t-p}$$

$$S_t = S_{t-p} + \delta(1-\alpha)e_t / L_t$$

**NOTE:** For missing values, $e_t = 0$.

The $k$-step prediction equation is

$$\hat{Y}_t(k) = L_t S_{t-p+k}$$

The multiplicative version of seasonal smoothing does not have an ARIMA equivalent; however, when the seasonal variation is small, the ARIMA additive-invertible region of the additive version of seasonal described in the preceding section can approximate the stability region of the multiplicative version.

The variance of the prediction errors is estimated as

$$var(e_t(k)) = var(\epsilon_t)\left[\sum_{i=0}^{\infty}\sum_{j=0}^{p-1}(\psi_{j+ip}S_{t+k}/S_{t+k-j})^2\right]$$

where $\psi_j$ are as described for the additive version of seasonal method and $\psi_j = 0$ for $j \geq k$.

### Winters Method—Additive Version

The model equation for the additive version of Winters method is

$$Y_t = \mu_t + \beta_t t + s_p(t) + \epsilon_t$$

The smoothing equations are

$$L_t = \alpha(Y_t - S_{t-p}) + (1-\alpha)(L_{t-1} + T_{t-1})$$

$$T_t = \gamma(L_t - L_{t-1}) + (1-\gamma)T_{t-1}$$

$$S_t = \delta(Y_t - L_t) + (1-\delta)S_{t-p}$$

The error-correction form of the smoothing equations is

$$L_t = L_{t-1} + T_{t-1} + \alpha e_t$$

$$T_t = T_{t-1} + \alpha\gamma e_t$$

$$S_t = S_{t-p} + \delta(1-\alpha)e_t$$

**NOTE:** For missing values, $e_t = 0$.

The $k$-step prediction equation is

$$\hat{Y}_t(k) = L_t + kT_t + S_{t-p+k}$$

The ARIMA model equivalency to the additive version of Winters method is the ARIMA(0,1,p+1)(0,1,0)$_p$ model

$$(1-B)(1-B^p)Y_t = \left[1 - \sum_{i=1}^{p+1} \theta_i B^i\right]\epsilon_t$$

$$\theta_j = \begin{cases} 1-\alpha-\alpha\gamma & j = 1 \\ -\alpha\gamma & 2 \le j \le p-1 \\ 1-\alpha\gamma-\delta(1-\alpha) & j = p \\ (1-\alpha)(\delta-1) & j = p+1 \end{cases}$$

The moving-average form of the equation is

$$Y_t = \epsilon_t + \sum_{j=1}^{\infty} \psi_j \epsilon_{t-j}$$

$$\psi_j = \begin{cases} \alpha + j\alpha\gamma & \text{for} j \mod p \neq 0 \\ \alpha + j\alpha\gamma + \delta(1-\alpha), & \text{for} j \mod p = 0 \end{cases}$$

For the additive version of Winters method (Archibald 1990), the additive-invertible region is

$$\{\max(-p\alpha, 0) < \delta(1-\alpha) < (2-\alpha)\}$$
$$\{0 < \alpha\gamma < 2-\alpha-\delta(1-\alpha)(1-cos(\vartheta)\}$$

where $\vartheta$ is the smallest nonnegative solution to the equations listed in Archibald (1990).

The variance of the prediction errors is estimated as

$$var(e_t(k)) = var(\epsilon_t)\left[1 + \sum_{j=1}^{k-1} \psi_j^2\right]$$

## Winters Method—Multiplicative Version

In order to use the multiplicative version of Winters method, the time series and all predictions must be strictly positive.

The model equation for the multiplicative version of Winters method is

$$Y_t = (\mu_t + \beta_t t)s_p(t) + \epsilon_t$$

The smoothing equations are

$$L_t = \alpha(Y_t/S_{t-p}) + (1-\alpha)(L_{t-1} + T_{t-1})$$

$$T_t = \gamma(L_t - L_{t-1}) + (1-\gamma)T_{t-1}$$

$$S_t = \delta(Y_t/L_t) + (1-\delta)S_{t-p}$$

The error-correction form of the smoothing equations is

$$L_t = L_{t-1} + T_{t-1} + \alpha e_t/S_{t-p}$$

$$T_t = T_{t-1} + \alpha\gamma e_t/S_{t-p}$$

$$S_t = S_{t-p} + \delta(1-\alpha)e_t/L_t$$

**NOTE:** For missing values, $e_t = 0$.

The $k$-step prediction equation is

$$\hat{Y}_t(k) = (L_t + kT_t)S_{t-p+k}$$

The multiplicative version of Winters method does not have an ARIMA equivalent; however, when the seasonal variation is small, the ARIMA additive-invertible region of the additive version of Winters method described in the preceding section can approximate the stability region of the multiplicative version.

The variance of the prediction errors is estimated as

$$var(e_t(k)) = var(\epsilon_t)\left[\sum_{i=0}^{\infty}\sum_{j=0}^{p-1}(\psi_{j+ip}S_{t+k}/S_{t+k-j})^2\right]$$

where $\psi_j$ are as described for the additive version of Winters method and $\psi_j = 0$ for $j \geq k$.

# ARIMA Models

The SAS Forecast Server Procedures software uses the same statistical model technology to identify, fit and forecast ARIMA models as does SAS/ETS Software. For information about the methods SAS Forecast Server Procedures software uses for ARIMA models, see Chapter 7, "The ARIMA Procedure" (*SAS/ETS User's Guide*). All the SAS/ETS ARIMA output (such as the parameter estimates, forecasts, diagnostic measures, and so on) is available in SAS Forecast Server Procedures software. Moreover, you can obtain additional output in SAS Forecast Server Procedures software. This additional output includes a wider variety of fit statistics and a model-based decomposition of the response series forecasts into subcomponents such as transfer functions effects and the estimated stationary noise component. These subcomponents can be useful for interpreting the model being used.

## ARIMA Model Based Series Decomposition

Consider a general ARIMA model that can be fit to a response series $Y_t$:

$$D(B)Y_t = \mu + \sum_i \frac{\omega_i(B)}{\delta_i(B)} B^{k_i} X_{i,t} + \frac{\theta(B)}{\phi(B)} a_t$$

where

| | |
|---|---|
| $t$ | indexes time. |
| $B$ | is the backshift operator; that is, $BX_t = X_{t-1}$. |
| $D(B)$ | is the difference operator operating on the response series $Y_t$. |
| $\mu$ | is the constant term. |
| $\phi(B)$ | is the autoregressive operator, represented as a polynomial in the backshift operator: $\phi(B) = 1 - \phi_1 B - \ldots - \phi_p B^p$. |
| $\theta(B)$ | is the moving-average operator, represented as a polynomial in the back shift operator: $\theta(B) = 1 - \theta_1 B - \ldots - \theta_q B^q$. |
| $a_t$ | is the independent disturbance, also called the random error. |
| $X_{i,t}$ | is the $i$th input time series or a difference of the $i$th input series at time $t$. |
| $k_i$ | is the pure time delay for the effect of the $i$th input series. |
| $\omega_i(B)$ | is the numerator polynomial of the transfer function for the $i$th input series. |
| $\delta_i(B)$ | is the denominator polynomial of the transfer function for the $i$th input series. |

The model expresses the response series, possibly differenced, as a sum of a constant term, various transfer function effects, and the effect of a stationary ARMA disturbance term. Of course, in a given situation many of these effects might be absent. Denoting the individual transfer function effects by $\gamma_i$ and the ARMA disturbance term by $n$, you can decompose the response series in a few different ways. Consider two alternate decompositions here. In the first case $Y_t$ is decomposed as

$$Y_t = L_t + \mu + \sum_i \gamma_{it} + n_t$$

where the $L_t$ term includes the contribution from the lagged response values that correspond to the differencing operator $D(B)$. For example, if $D(B) = 1 - B$, corresponding to the differencing of order 1, then $L_t = Y_{t-1}$. An alternate decomposition of $Y_t$ can be as follows:

$$Y_t = \frac{\mu}{D(B)} + \sum_i \frac{\gamma_{it}}{D(B)} + \frac{n_t}{D(B)}$$

Note that if the differencing operator $D(B)$ is identity, then the $L_t$ term is zero and these two alternate decompositions are identical. Otherwise the terms in the second decomposition are the "integrated," with respect to $D(B)$, versions of the corresponding terms in the first decomposition. In practice many terms in these decompositions are not observed but are estimated from the data, which results in similar decompositions of $\hat{Y}_t$, the *forecasts* of $Y_t$. In the HPFENGINE procedure you can obtain these decompositions of $\hat{Y}_t$ by choosing various options in the PROC HPFENGINE statement: you can output them as a data set by using the OUTCOMPONENT= data set option, print them by using the PRINT=COMPONENT option, or plot them by using the PLOT=COMPONENTS option. The type of decomposition is controlled by the COMPONENTS= option in the HPFENGINE statement. If COMPONENTS=INTEGRATE is specified, then the calculated decomposition is of the second type, and the default decomposition is of the first type. Consider the following points about these decompositions:

- If the response series being modeled is actually log transformed, then the resulting decompositions are *multiplicative* rather than *additive*. In this case, similar to the series forecasts, the decomposition terms are also inverse transformed. If the response series is transformed using a transformation other than log (such as Box-Cox or square root), then these decompositions are difficult to interpret and they do not have such additive or multiplicative properties.
- In the first type of decomposition the components in the decomposition always add up, or multiply in the log transformation case, to the series forecasts. In the integrated version of the decomposition this additive property might not always hold because there are no natural choices of starting values that can be used during the integration of these components that guarantee the additivity of the resulting decomposition.

## Unobserved Component Models

The SAS Forecast Server Procedures software uses the same statistical model technology to identify, fit and forecast unobserved component models (UCMs) as does SAS/ETS software. For information about the methods the SAS Forecast Server Procedures software uses for UCMs, see Chapter 34, "The UCM Procedure" (*SAS/ETS User's Guide*).

# Intermittent Models

This section details the computations performed for intermittent forecasting models.

## Intermittent Time Series

Intermittent time series have a large number of values that are zero. These types of series commonly occur in Internet, inventory, sales, and other data where the demand for a particular item is intermittent. Typically, when the value of the series associated with a particular time period is nonzero, *demand* occurs; and when the value is zero (or missing), *no demand* occurs. Since it is entirely possible that the number of time periods for which *no demand* occurs is large, many of the series values are zero. Typical time series models (for example, smoothing models) are inadequate in the case of intermittent time series because many of the series values are zero. Since these models are based on weighted-summations of past values, they bias forecasts away from zero. Unlike the smoothing models that provide forecasts for future time periods, intermittent forecasting models provide recommended *stocking levels* or *estimated demand per period* that are used to satisfy future demand.

## Intermittent Series Decomposition and Analysis

An intermittent time series (demand series) can be decomposed into two components: a demand interval series and a demand size series. Both of these component series are indexed based on when a *demand* occurred (demand index) rather than each time period (time index). The demand interval series is constructed based on the number of time periods between *demands*. The demand size series is constructed based on the size (or value) of the *demands* excluding zero (or base) demand values. Using these two component series, the average demand series is computed by dividing the size component values by the interval component values.

When a *demand* occurs typically depends on a *base* value. Typically, the base value is zero (default), but it can be any constant value and can be automatically determined based on the characteristics of the demand series.

Given a time series $y_t$, for $t = 1$ to $T$, where $t$ is the time index, suppose that there are $N$ nonzero demands occurring at times $t = t_i$, where $t_{i-1} < t_i$, for $i = 1$ to $N$. The time series is dissected into the demand interval series and the demand size series as follows:

| | | |
|---|---|---|
| (demand interval series) | $q_i = t_i - t_{i-1}$ | for $i = 2$ to $N$ |
| (demand size series) | $d_i = y_{t_i} - base$ | for $i = 1$ to $N$ |
| (average demand series) | $a_i = d_i / q_i$ | for $i = 2$ to $N$ |

For the beginning of the demand series, $q_1$ is assigned to $t_1$, which assumes that a demand just occurred prior to the first recorded observation. For the end of the demand series, $q_{N+1}$ is assigned to $(T + 1 - t_N)$, which assumes that demand will occur just after the last recorded observation. The next future demand size, $d_{N+1}$, is always set to missing.

After decomposition, descriptive (summary) statistics can be computed to gain a greater understanding of the demand series including those statistics based on the season index.

For statistical analysis and model fitting, $q_i$ and $a_i$ for $i = 2$ to $N$ and $d_i$ for $i = 1$ to $N$ are used. For forecasting, $q_i$ for $i = 1$ to $N + 1$, $a_i$ for $i = 1$ to $N$, and $d_i$ for $i = 1$ to $N$ are used.

## Croston's Method

Croston's method models and forecasts each component independently, then combines the two forecasts. The following provides a description of how Croston's method is used in SAS Forecast Server Procedures software. For more detailed information about this method see: Croston (1972); Willemain, Smart, and Shocker (1994). The following description of Croston's method is based on the perspective of a person familiar with typical time series modeling techniques such as smoothing models.

By treating each component of the demand series as a time series based on the demand index, optimal smoothing parameters can be estimated and predictions for each component can be computed using nonseasonal exponential smoothing methods (simple, double, linear, and damped-trend) as well as their transformed versions (log, square-root, logistic, and Box-Cox).

For example, the following simple smoothing equations are used to generate predictions for demand size and interval components:

$$\text{(smoothed demand interval series)} \quad L_i^q = L_{i-1}^q + \alpha_q(q_{i-1} - L_{i-1}^q)$$
$$\text{(smoothed demand size series)} \quad L_i^d = L_{i-1}^d + \alpha_d(d_{i-1} - L_{i-1}^d)$$

The demand interval parameter $\alpha_q$ and demand size parameter $\alpha_d$ and the starting, intermediate, and final smoothing level states, $L_i^q$ and $L_i^d$, are estimated from the data using simple exponential smoothing parameter estimation. For the starting state at $i = 1$, $L_1^q = \max(q_1, L_0^q)$ where $L_0^q$ is the final backcast level state. For $i > 1$, the one-step-ahead prediction for demand interval $q_i$ is $\hat{q}_i = L_{i-1}^q$. For $i > 0$, the one-step-ahead prediction for demand size $d_i$ is $\hat{d}_i = L_{i-1}^d$.

Other (transformed) nonseasonal exponential smoothing methods can be used in a similar fashion. For linear smoothing, $L_1^q = \max(q_1 - T_0^q, L_0^q)$ where $T_0^q$ is the final backcast trend state. For damp-trend smoothing, $L_1^q = \max(q_1 - \phi_q T_0^q, L_0^q)$ where $\phi_q$ is the damping parameter and $T_0^q$ is the final backcast trend state. For double smoothing, $L_1^q = \max(q_1 - T_0^q/\alpha_q, L_0^q)$ where $\alpha_q$ is the weight parameter and $T_0^q$ is the final backcast trend state.

Using these predictions based on the demand index, predictions of the average demand per period can be estimated. Predicted demand per period is also known as "stocking level," assuming that disturbances affecting $q_i$ are independent of $d_i$. (This assumption is quite significant.)

$$\text{(estimated demand per period)} \quad y_i^* = \hat{d}_i/\hat{q}_i$$
$$\mathrm{E}[y_i^*] = E[d_i]/E[q_i] = E[z_{i-1}]/E[p_{i-1}] = \mu_d/\mu_q$$
$$\text{(variance)} \quad Var(y_i^*) = (\hat{d}_i/\hat{q}_i)^2(Var(d_i)/\hat{d}_i^2 + Var(q_i)/\hat{q}_i^2)$$

where $\mu_d$, $\bar{d}$, and $s_d$ are the mean, sample average, and standard deviation, respectively, of the nonzero demands, and $\mu_q$, $\bar{q}$, and $s_q$ are the mean, sample average, and standard deviation, respectively, of the number of time periods between demands.

For the beginning of the series, the denominator of $y_1^*$ is assigned $q_i$ or the starting smoothing state $p_0$, whichever is greater. For the end of the series, the denominator of $y_{N+1}^*$ is assigned $q_{N+1} = (T + 1 - t_N)$ or the final smoothing state $p_N$, whichever is greater.

After the average demand per period has been estimated, a stocking level can be recommended:

| | | |
|---|---|---|
| (recommended stocking level) | $\hat{y}_t = y_i^*$ | when $t_i =< t < t_{i+1}$ |
| (variance) | $Var(\hat{y}_t) = Var(y_i^*)$ | when $t_i =< t < t_{i+1}$ |

Since the predicted demand per period is different than typical time series forecasts, the usual way of computing statistics of fit should also be different. The statistics of fit are based on the difference between the recommended stocking levels between demands and the demands:

| | | |
|---|---|---|
| (accumulated recommended stocks) | $s_t = \sum_{i=0}^{t}(\hat{y}_t - y_i^*)$ | |
| (estimate—demand) | $e_{t_i} = \hat{d}_i q_i - d_i$ | when time $t_{i+1}$ has demand |

Croston's method produces the same forecasts as simple exponential smoothing when demand occurs in every time period, $q_i = 1$ for all $i$, but different (lower) prediction error variances. Croston's method is recommended for intermittent time series only.

## Average Demand Method

Similar to Croston's method, the average demand method is used to forecast intermittent time series; however, the average demand method forecasts the average demand series directly, whereas Croston's method forecasts average demand series indirectly using the inverse decomposition of the demand interval and size series forecasts.

By treating the average demand series as a time series based on the demand index, optimal smoothing parameters can be estimated and predictions for average demand can be computed using nonseasonal exponential smoothing methods (simple, double, linear, and damped-trend) as well as their transformed versions (log, square-root, logistic, and Box-Cox).

For example, the following simple smoothing equations are used to generate predictions for the average demand series:

(smoothed average demand series) $\quad L_i^a = L_{i-1}^a + \alpha_a(a_{i-1} - L_{i-1}^a)$

The average demand level smoothing parameter $\alpha_a$ and the starting, intermediate, and final smoothing level states $L_i^a$ are estimated from the data by using simple exponential smoothing parameter estimation. For the starting state at $i = 1$, $L_1^a = \max(a_1, L_0^a)$ where $L_0^a$ is the final backcast level state. For $i > 1$, the one-step-ahead prediction for $a_i$ is $\hat{a}_i = L_{i-1}^a$.

Other nonseasonal exponential smoothing methods can be used in a similar fashion. For linear smoothing, $L_1^a = \max(a_1 - T_0^a, L_0^a)$ where $T_0^a$ is the final backcast trend state. For damp-trend smoothing, $L_1^a = \max(a_1 - \phi_a T_0^a, L_0^a)$ where $\phi_a$ is the damping parameter and $T_0^a$ is the final backcast trend state. For double smoothing, $L_1^a = \max(a_1 - T_0^a/\alpha_a, L_0^a)$ where $\alpha_a$ is the weight parameter and $T_0^a$ is the final backcast trend state.

Using these predictions based on the demand index, predictions of the average demand per period are provided directly, unlike Croston's method where the average demand is predicted using a ratio of predictions of the demand interval and size components.

| | |
|---|---|
| (estimated demand per period) | $y_i^* = \hat{a}_i + base$ |
| | $\mathrm{E}[y_i^*] = E[a_i]$ |
| (variance) | see the exponential smoothing models |

For the beginning of the series, $\hat{a}_1$ is derived from the starting level smoothing state and starting trend smoothing state (if applicable).

After the average demand per period has been estimated, a stocking level can be recommended similar to Croston's method.

The average demand method produces the same forecasts as exponential smoothing when demand occurs in every time period, $q_i = 1$ for all $i$, but different (lower) prediction error variances. The average demand method is recommended for intermittent time series only.

## Time-Indexed versus Demand-Indexed Holdout Samples

Holdout samples are typically specified based on the time index. But for intermittent demand model selection, demand-indexed-based holdouts are used for model selection.

For example, consider "holdout the last six months data." For a demand series, the demand indexed holdout refers to the "demands that have occurred in the last six months." If there are four demands in the last six months, the demand indexed holdout is four for a time-indexed holdout of six. If there are no demands in the time-indexed holdout, the demand-indexed holdout is zero and in-sample analysis is used.

## Automatic Intermittent Demand Model Selection

The exponential smoothing method to be used to forecast the intermittent demand series can be specified, or it can be selected automatically using a model selection criterion and either in-sample or holdout sample analysis. The exponential smoothing method for each demand series component (interval, size, and average) and the choice between Croston's method and the average demand method can be automatically selected.

For Croston's method, the exponential smoothing methods used to forecast the demand interval and size components are automatically selected independently. For the average demand method, the exponential smoothing methods used to forecast the average demand component are automatically selected, again independently. Based on the model selection criterion, the selection is based on how well the method fits (in sample) or predicts (holdout sample) the demand series component by treating the demand index as a time index. The following equations describe the component prediction errors associated with each of the demand series components that are used in component model selection:

$$\begin{array}{lll} \text{(demand interval series)} & e_i^q = q_i - \hat{q}_i & \text{for } i = 2 \text{ to } N \\ \text{(demand size series)} & e_i^d = d_i - \hat{d}_i & \text{for } i = 1 \text{ to } N \\ \text{(average demand series)} & e_i^a = a_i - \hat{a}_i & \text{for } i = 2 \text{ to } N \end{array}$$

After the exponential smoothing methods are selected for each demand series component, the predictions for either Croston's method ($\hat{d}_i / \hat{q}_i$), the average demand method $\hat{a}_i$, or both are computed based on the selected method for each component.

When choosing between Croston's method and the average demand method, the model is selected by considering how well the model predicts average demand with respect to the time. The following equations describe the average prediction errors associated with the predicted average demand that are used in model selection:

$$\begin{array}{lll} \text{(Croston's method)} & e_i^c = (d_i / q_i) - (\hat{d}_i / \hat{q}_i) & \text{for } i = 2 \text{ to } N \\ \text{(average demand method)} & e_i^a = a_i - \hat{a}_i & \text{for } i = 2 \text{ to } N \end{array}$$

# External Models

*External forecasts* are forecasts provided by an *external source*. External forecasts might originate from a statistical model, from another software package, might have been provided by an outside organization (for example, marketing organization or government agency), or might be given based solely judgment.

## External Forecasts

Given a time series, $\{y_t\}_{t=1}^{T}$, where $t = 1, \ldots, T$ is the time index and $T$ is the length of the time series, the external model data source must provide predictions for the future time periods, $\{\hat{y}_t\}_{t=T+1}^{T+H}$, where $H$ represents the forecast horizon. The external data source might or might not provide in-sample predictions for past time periods, $\{\hat{y}_t\}_{t=1}^{T}$. Additionally, the external source might or might not provide the prediction standard errors $\{Std(\hat{y}_t)\}_{t=1}^{T+H}$, lower confidence limits $\{Lower(\hat{y}_t)\}_{t=1}^{T+H}$, and the upper confidence limits $\{Upper(\hat{y}_t\}_{t=1}^{T+H}$ for the past or future time periods.

## External Forecast Prediction Errors

If the external forecast predictions are provided for past time periods, the external forecast prediction errors $\{\hat{e}_t\}_{t=1}^{T}$ can be computed, $\hat{e}_t = y_t - \hat{y}_t$. If any of these predictions are not provided by the external source, the prediction errors are set to missing.

For judgmental forecast with no historical judgments, all prediction errors are missing. For this situation, a judgment about the prediction standards errors should also be provided.

## External Forecast Prediction Bias

You often want forecasts to be unbiased. If available, the external forecast prediction errors are used to compute prediction bias from the external source.

When the external forecast predictions are provided for past time periods, SAS Forecast Server Procedures software uses the in-sample prediction errors $\{\hat{e}_t\}_{t=1}^{T}$ to estimate the bias of the forecasts that are provided by the external source as follows:

$$Bias = \sum_{t=1}^{T} (y_t - \hat{y}_t)/T = \sum_{t=1}^{T} \hat{e}_t/T$$

The prediction mean square error can be used to help judge the significance of this bias.

$$Variance = \sum_{t=1}^{T} (\hat{e}_t - Bias)^2/(T-1)$$

Missing values are ignored in the above computations, and the denominator term is reduced for each missing value.

## External Forecast Prediction Standard Errors

The external model data might include estimates of the prediction standard errors and confidence limits. If these estimates are not supplied along with the data for the external model (which is usually the case for judgmental forecasts), the SAS Forecast Server Procedures software can approximate the prediction standard errors and the confidence limits by using the in-sample prediction errors.

When the external forecasts do not contain the prediction standard errors, they are approximated using the external forecast prediction errors. In order to approximate the external forecast prediction standard errors, the external model residuals, variance, and autocorrelations must be approximated. (If the prediction standard errors are provided by the external source, approximation is not required.)

In order to approximate the external model residuals $\{\hat{\epsilon}_t\}_{t=1}^{T}$, the transformation used to generate the external forecast must be provided as part of the external model specification. Let $z_t = f(y_t)$ be the specified transformation; if there is no specified functional transformation, $z_t = y_t$. The transformation can be used to approximate the external model residuals, $\hat{\epsilon}_t = f(y_t) - f(\hat{y}_t) = z_t - \hat{z}_t$.

After they are approximated, the external model residuals can be used to approximate the external model variance $\hat{\sigma}_\epsilon^2 = \frac{1}{T}\sum_{t=1}^{T} \hat{\epsilon}_t^2$ and the external model residual autocorrelation.

The external model residual autocorrelations can be approximated given assumptions about the autocorrelation structure $\rho(j)$ where $j \geq 0$ represents the time lags.

The following autocorrelation structures can be specified using the METHOD= option. These options are listed in order of increasing assumed autocorrelation. (In the following formulas $0 < v < 1$ represents the value of the NLAGPCT= option.)

no autocorrelation (METHOD=NONE)

$$\rho(j) = 1 \qquad \text{for} \qquad j = 0$$
$$\rho(j) = 0 \qquad \text{for} \qquad j > 0$$

This option generates constant prediction error variances.

white noise autocorrelation (METHOD=WN)

This option generates white noise prediction error variances.

error autocorrelation (METHOD=ERRORACF)

$$\rho(j) = 1 \qquad \text{for} \qquad j = 0$$
$$\hat{\rho}(j) = \frac{1}{T}\sum_{t=j+1}^{T} \hat{\epsilon}_t \hat{\epsilon}_{t-j} / \hat{\sigma}_\epsilon^2 \qquad \text{for} \qquad 0 < j < vT$$
$$\rho(j) = 0 \qquad \text{for} \qquad j > vT$$

series autocorrelation (METHOD=ACF)

$$\rho(j) = 1 \qquad \text{for} \qquad j = 0$$

$$\hat{\rho}(j) = \frac{1}{T}\sum_{t=j+1}^{T} z_t z_{t-j}/\hat{\sigma}_z^2 \qquad \text{for} \qquad 0 < j < vT$$

$$\rho(j) = 0 \qquad \text{for} \qquad j > vT$$

where $\hat{\sigma}_z^2 = \frac{1}{T}\sum_{t=1}^{T}(z_t - \bar{z})^2$ and $0 < v < 1$.

This option typically generates linear prediction error variances larger than ERRORACF.

perfect autocorrelation (METHOD=PERFECT)

$$\rho(j) = 1 \qquad \text{for} \qquad j \geq 0$$

This option generates linear prediction error variances.

The external model residual variance and the autocorrelations can approximate the external (transformed) prediction standard errors $\{Std(\hat{z}_t)\}_{t=1}^{T+H}$ where $Std(\hat{z}_t) = \sigma_\epsilon$ for $t = 1, \ldots, T$ are the one-step-ahead prediction standards errors and $Std(\hat{z}_{T+h}) = \sqrt{\sigma_\epsilon^2 \sum_{j=0}^{h-1} \hat{\rho}(j)^2}$ for $h = 1, \ldots, H$ are the multistep-ahead prediction standard errors.

If there was no transformation used to generate the external forecasts, $Std(\hat{y}_t) = Std(\hat{z}_t)$. Otherwise, the external transformed predictions $\{\hat{z}\}_{t=1}^{T}$ and the approximate external transformed prediction standard errors $\{Std(\hat{z}_t)\}_{t=1}^{T+H}$ are used to approximate the external prediction standard errors $\{Std(\hat{y}_t)\}_{t=1}^{T+H}$ by computing the conditional expectation of the inverse transformation (mean) or computing the inverse transformation (median).

To summarize, the external forecast prediction standard errors $\{Std(\hat{y}_t)\}_{t=1}^{T+H}$ can be approximated from the external forecast prediction errors $\{\hat{e}_t\}_{t=1}^{T}$, given the following assumptions about the external forecasts provided by the external source:

| | | |
|---|---|---|
| functional transformation | (TRANSFORM= option) | $z_t = f(y_t)$ |
| autocorrelation structure | (METHOD= option) | $\rho(j)$ |
| autocorrelation cutoff | (NLAGPCT= option) | $v$ |

## External Forecast Confidence Limits

Because the external forecasts might not contain confidence limits, they must be approximated using the forecast prediction standard errors (which might be provided by the external model data source or approximated from the in-sample prediction errors) as follows:

$$Lower(\hat{y}_t) = \hat{y}_t - Std(\hat{y}_t)Z_{\frac{\alpha}{2}}$$

$$Upper(\hat{y}_t) = \hat{y}_t + Std(\hat{y}_t)Z_{\frac{\alpha}{2}}$$

where $\alpha$ is the confidence limit width, $(1-\alpha)$ is the confidence level, and $Z_{\frac{\alpha}{2}}$ is the $\frac{\alpha}{2}$ quantile of the standard normal distribution.

To summarize, the external forecast confidence limits can be approximated given only the actual time series $\{y_t\}_{t=1}^{T}$ and the external forecast predictions $\{\hat{y}_t\}_{t=T+1}^{T+H}$. More information provided about the external forecast prediction standard errors $\{Std(y_t)\}_{t=1}^{T+H}$, such as the functional transform $f()$ and the autocorrelation structure $\rho(j)$ and $v$ improves the accuracy of these approximations.

# Series Transformations

For forecasting models, transforming the time series can aid in improving forecasting accuracy.

There are four transformations available, for strictly positive series only. Let $y_t > 0$ be the original time series, and let $w_t$ be the transformed series. The transformations are defined as follows:

log — is the logarithmic transformation

$$w_t = \ln(y_t)$$

logistic — is the logistic transformation

$$w_t = \ln(c y_t / (1 - c y_t))$$

where the scaling factor $c$ is

$$c = (1 - e^{-6}) 10^{-\text{ceil}(\log_{10}(max(y_t)))}$$

and $\text{ceil}(x)$ is the smallest integer greater than or equal to $x$ .

square root — is the square root transformation

$$w_t = \sqrt{y_t}$$

Box Cox — is the Box-Cox transformation

$$w_t = \begin{cases} \frac{y_t^{\lambda} - 1}{\lambda}, & \lambda \neq 0 \\ \ln(y_t), & \lambda = 0 \end{cases}$$

Parameter estimation is performed using the transformed series. The transformed model predictions and confidence limits are then obtained from the transformed time-series and these parameter estimates.

The transformed model predictions $\hat{w}_t$ are used to obtain either the minimum mean absolute error (MMAE) or minimum mean squared error (MMSE) predictions $\hat{y}_t$, depending on the setting of the forecast options. The model is then evaluated based on the residuals of the original time series and these predictions. The transformed model confidence limits are inverse-transformed to obtain the forecast confidence limits.

## Predictions for Transformed Models

Since the transformations described in the previous section are monotonic, applying the inverse-transformation to the transformed model predictions results in the *median* of the conditional probability density function at each point in time. This is the minimum mean absolute error (MMAE) prediction.

If $w_t = \text{F}(y_t)$ is the transform with inverse-transform $y_t = \text{F}^{-1}(w_t)$, then

$$\text{median}(\hat{y}_t) = \text{F}^{-1}(E\,[w_t]) = \text{F}^{-1}(\hat{w}_t)$$

The minimum mean squared error (MMSE) predictions are the *mean* of the conditional probability density function at each point in time. Assuming that the prediction errors are normally distributed with variance $\sigma_t^2$, the MMSE predictions for each of the transformations are as follows:

log — is the conditional expectation of inverse-logarithmic transformation

$$\hat{y}_t = E\left[e^{w_t}\right] = \exp\left(\hat{w}_t + \sigma_t^2/2\right)$$

logistic — is the conditional expectation of inverse-logistic transformation

$$\hat{y}_t = E\left[\frac{1}{c(1+exp(-w_t))}\right]$$

where the scaling factor $c = (1-10^{-6})10^{-\text{ceil}(\log_{10}(\max(y_t)))}$.

square root — is the conditional expectation of the inverse-square root transformation

$$\hat{y}_t = E\left[w_t^2\right] = \hat{w}_t^2 + \sigma_t^2$$

Box Cox — is the conditional expectation of the inverse Box-Cox transformation

$$\hat{y}_t = \begin{cases} E\left[(\lambda w_t + 1)^{1/\lambda}\right], & \lambda \neq 0 \\ E\left[e^{w_t}\right] = \exp(\hat{w}_t + \frac{1}{2}\sigma_t^2), & \lambda = 0 \end{cases}$$

The expectations of the inverse logistic and Box-Cox ( $\lambda \neq 0$ ) transformations do not generally have explicit solutions and are computed using numerical integration.

## Series Diagnostic Tests

This section describes the diagnostic tests that are used to determine the kinds of forecasting models appropriate for a series.

The series diagnostics are a set of heuristics that provide recommendations about whether or not the forecasting model should contain a log transform, trend terms, and seasonal terms or whether or not the time series is intermittent. These recommendations are used by the automatic model selection process to restrict the model search to a subset of the model selection list.

The tests that are used by the series diagnostics do not always produce the correct classification of the series. They are intended to accelerate the process of searching for a good forecasting model for the series, but you should not rely on them if finding the very best model is important to you.

The series diagnostics tests are intended as a heuristic tool only, and no statistical validity is claimed for them. These tests might be modified and enhanced in future releases of the SAS Forecast Server Procedures software. The testing strategy is as follows:

1. **Intermittent test.** Compute the average time interval between demands. If the average time interval is greater than a preset limit, an intermittent forecasting model is used.
2. **Seasonality test.** The resultant series is tested for seasonality. A seasonal dummy model with AR(1) errors is fit, and the joint significance of the seasonal dummy estimates is tested. If the seasonal dummies are significant, the AIC statistic for this model is compared to the AIC for and AR(1) model without seasonal dummies.

## Statistics of Fit

This section explains the goodness-of-fit statistics reported to measure how well different models fit the data. The statistics of fit for the various forecasting models can be printed or stored in a data set.

The various statistics of fit reported are as follows. In these formulae below, the following conventions apply:

1. $n$ is the number of nonmissing observations,
2. $k$ is the number of fitted parameters in the model,
3. Unless otherwise noted, the errors are set to zero in the statistical computations when $|y_t - \hat{y}_t| < \epsilon$ where $\epsilon$ is the largest double precision number satisfying $1 = 1 + \epsilon$. This happens without regard to whether the denominator is zero.
4. Unless otherwise noted, the errors are ignored in the statistical computations when the denominator is zero.

$APE = |100 * (y_t - \hat{y}_t)/y_t|$ is the absolute percent error.

$ASPE = |100 * (y_t - \hat{y}_t)/0.5(y_t + \hat{y}_t)|$ is the absolute symmetric percent error.

$APPE = |100 * (y_t - \hat{y}_t)/\hat{y}_t|$ is the absolute predictive percent error.

$RAE = |(y_t - \hat{y}_t)/(y_t - y_{t-1})|$ is the relative absolute error.

$APES = 100 * |y_t - \hat{y}_t| / \sqrt{\sum_{t=1}^{n}(y_t - \bar{y})^2/(n-1)}$ is the absolute error as a percentage of the standard deviation.

$ASE = |y_t - \hat{y}_t| / \frac{1}{n-1} \sum_{t=2}^{n} |y_t - y_{t-1}|$ is the absolute scaled error.

*Number of nonmissing observations.*
The number of nonmissing observations used to fit the model.

*Number of observations.*
The total number of observations used to fit the model, including both missing and nonmissing observations.

*Number of missing actuals.*
The number of missing actual values.

*Number of missing predicted values.*
The number of missing predicted values.

*Number of model parameters.*
The number of parameters fit to the data. For combined forecast, this is the number of forecast components.

*Total sum of squares (uncorrected).*
The total sum of squares for the series, SST, uncorrected for the mean: $\sum_{t=1}^{n} y_t^2$.

*Total sum of squares (corrected).*
The total sum of squares for the series, SST, corrected for the mean: $\sum_{t=1}^{n} (y_t - \overline{y})^2$, where $\overline{y}$ is the series mean.

*Sum of square errors.*
The sum of the squared prediction errors, SSE. $SSE = \sum_{t=1}^{n} (y_t - \hat{y}_t)^2$, where $\hat{y}_t$ is the one-step predicted value.

*Mean square error.*
The mean squared prediction error, MSE, calculated from the one-step-ahead forecasts. $MSE = \frac{1}{n} SSE$. This formula enables you to evaluate small holdout samples.

*Root mean square error.*
The root mean square error (RMSE), $\sqrt{MSE}$.

*Mean absolute error.*
The mean absolute prediction error, $\frac{1}{n} \sum_{t=1}^{n} |y_t - \hat{y}_t|$.

*Minimum absolute percent error.*
The minimum of the absolute percent errors (MINAPE).

*Maximum absolute percent error.*
The maximum of the absolute percent errors (MAXAPE).

*Mean absolute percent error.*
The mean of the absolute percent errors (MAPE).

*Median absolute percent error.*
The median of the absolute percent errors (MdAPE).

*Geometric mean absolute percent error.*
The geometric mean of the absolute percent errors (GMAPE).

*Minimum absolute error as percentage of SD.*
The minimum of the absolute error as a percentage of the standard deviation (MINAPES).

*Maximum absolute error as percentage of SD.*
The maximum of the absolute error as a percentage of the standard deviation (MAXAPES).

*Mean absolute error as percentage of SD.*
The mean of the absolute error as a percentage of the standard deviation (MAPES).

*Median absolute error as percentage of SD.*
The median of the absolute error as a percentage of the standard deviation (MdAPES).

*Geometric mean error as percentage of SD.*
The geometric mean of the absolute error as a percentage of the standard deviation (GMAPES).

*Minimum absolute symmetric percent error.*
The minimum of the absolute symmetric percent errors (MINASPE).

*Maximum absolute symmetric percent error.*
The maximum of the absolute symmetric percent errors (MAXASPE).

*Mean absolute symmetric percent error.*
The mean of the absolute symmetric percent errors (MASPE).

*Median absolute symmetric percent error.*
The median of absolute symmetric percent errors (MdASPE).

*Geometric mean symmetric percent error.*
The geometric mean of the absolute symmetric percent errors (GMASPE).

*Minimum absolutepredictive percent error.*
The minimum of the absolute predictive percent errors (MINAPPE).

*Maximum absolute predictive percent error.*
The maximum of the absolute predictive percent errors (MAXAPPE).

*Mean absolute predictive percent error.*
The mean of the absolute predictive percent errors (MAPPE).

*Median absolute predictive percent error.*
The median absolute predictive percent prediction error (MdAPPE).

*Geometric mean absolute predictive percent error.*
The geometric mean absolute predictive percent prediction error (GMAPPE).

*Minimum relative absolute error.*
The minimum of the relative absolute errors (MINRAE).

*Maximum relative absolute error.*
The maximum of the relative absolute errors (MAXRAE).

*Mean relative absolute error.*
The mean of the relative absolute errors (MRAE).

*Median relative absolute error.*
The median of the relative absolute errors (MdRAE).

*Geometric relative absolute error.*
The geometric mean of the relative absolute errors (GMRAE).

*Mean absolute scaled error.*
The mean of the absolute scaled errors (MASE).

*R-square.*
The $R^2$ statistic, $R^2 = 1 - SSE/SST$. If the model fits the series badly, the model error sum of squares, *SSE* , might be larger than *SST* and the $R^2$ statistic will be negative.

*Adjusted R-square.*
The adjusted $R^2$ statistic, $1 - (\frac{n-1}{n-k})(1 - R^2)$.

*Amemiya's adjusted R-square.*
Amemiya's adjusted $R^2$, $1 - (\frac{n+k}{n-k})(1 - R^2)$.

*Random walk R-square.*
The random walk $R^2$ statistic (Harvey's $R^2$ statistic using the random walk model for comparison), $1 - (\frac{n-1}{n})SSE/RWSSE$, where $RWSSE = \sum_{t=2}^{n}(y_t - y_{t-1} - \mu)^2$ and $\mu = \frac{1}{n-1}\sum_{t=2}^{n}(y_t - y_{t-1})$.

*Akaike's information criterion.*
Akaike's information criterion (AIC), $n \ln(SSE/n) + 2k$.

*Akaike's information criterion, finite sample size corrected.*
Akaike's information criterion with an empirical correction for small sample sizes (AICC), $AIC + \frac{2k(k+1)}{n-k-1}$.

*Schwarz Bayesian information criterion.*
Schwarz Bayesian information criterion (SBC or BIC),
$n \ln(SSE/n) + k \ln(n)$.

*Amemiya's prediction criterion.*
Amemiya's prediction criterion, $\frac{1}{n}SST(\frac{n+k}{n-k})(1 - R^2) = (\frac{n+k}{n-k})\frac{1}{n}SSE$.

*Maximum error.*
The largest prediction error.

*Minimum error.*
The smallest prediction error.

*Maximum percent error.*
The largest percent prediction error, $100 \max((y_t - \hat{y}_t)/y_t)$. The summation ignores observations where $y_t = 0$.

*Minimum percent error.*
The smallest percent prediction error, $100 \min((y_t - \hat{y}_t)/y_t)$. The summation ignores observations where $y_t = 0$.

*Mean error.*
The mean prediction error, $\frac{1}{n} \sum_{t=1}^{n} (y_t - \hat{y}_t)$.

*Mean percent error.*
The mean percent prediction error, $\frac{100}{n} \sum_{t=1}^{n} \frac{(y_t - \hat{y}_t)}{y_t}$. The summation ignores observations where $y_t = 0$.

# References


Aldrin, M. and Damsleth, E. (1989), "Forecasting Non-seasonal Time Series with Missing Observations," *Journal of Forecasting*, 8, 97–116.

Anderson, T. W. (1971), *The Statistical Analysis of Time Series*, New York: John Wiley & Sons.

Archibald, B. C. (1990), "Parameter Space of the Holt-Winters' Model," *International Journal of Forecasting*, 6, 199–209.

Bartolomei, S. M. and Sweet, A. L. (1989), "A Note on the Comparison of Exponential Smoothing Methods for Forecasting Seasonal Series," *International Journal of Forecasting*, 5, 111–116.

Bowerman, B. L. and O'Connell, R. T. (1979), *Time Series and Forecasting: An Applied Approach*, North Scituate, MA: Duxbury Press.

Box, G. E. P. and Cox, D. R. (1964), "An Analysis of Transformations," *Journal of the Royal Statistical Society, Series B*, 26, 211–234.

Box, G. E. P. and Jenkins, G. M. (1976), *Time Series Analysis: Forecasting and Control*, Rev. Edition, San Francisco: Holden-Day.

Brocklebank, J. C. and Dickey, D. A. (1986), *SAS System for Forecasting Time Series*, 1986 Edition, Cary, NC: SAS Institute Inc.

Brown, R. G. (1962), *Smoothing, Forecasting, and Prediction of Discrete Time Series*, New York: Prentice-Hall.

Brown, R. G. and Meyer, R. F. (1961), "The Fundamental Theorem of Exponential Smoothing," *Operations Research*, 9, 673–685.

Chatfield, C. (1978), "The Holt-Winters Forecasting Procedure," *Applied Statistics*, 27, 264–279.

Chatfield, C. and Yar, M. (1988), "Holt-Winters Forecasting: Some Practical Issues," *The Statistician*, 37, 129–140.

Chatfield, C. and Yar, M. (1991), "Prediction Intervals for Multiplicative Holt-Winters," *International Journal of Forecasting*, 7, 31–37.

Cogger, K. O. (1974), "The Optimality of General-Order Exponential Smoothing," *Operations Research*, 22, 858–867.

Cox, D. R. (1961), "Prediction by Exponentially Weighted Moving Averages and Related Methods," *Journal of the Royal Statistical Society, Series B*, 23, 414–422.

Croston, J. D. (1972), "Forecasting and Stock Control for Intermittent Demands," *Operational Research Quarterly*, 23, 289–303.

Davidson, J. (1981), "Problems with the Estimation of Moving Average Models," *Journal of Econometrics*, 16, 295.

Fair, R. C. (1986), "Evaluating the Predictive Accuracy of Models," in Z. Griliches and M. D. Intriligator, eds., *Handbook of Econometrics*, volume 3, New York: North-Holland.

Fildes, R. (1979), "Quantitative Forecasting—the State of the Art: Extrapolative Models," *Journal of Operational Research Society*, 30, 691–710.

Fuller, W. A. (1976), *Introduction to Statistical Time Series*, New York: John Wiley & Sons.

Gardner, E. S., Jr. (1984), "The Strange Case of the Lagging Forecasts," *Interfaces*, 14, 47–50.

Gardner, E. S., Jr. (1985), "Exponential Smoothing: The State of the Art," *Journal of Forecasting*, 4, 1–38.

Granger, C. W. J. and Newbold, P. (1977), *Forecasting Economic Time Series*, New York: Academic Press.

Greene, W. H. (1993), *Econometric Analysis*, 2nd Edition, New York: Macmillan.

Hamilton, J. D. (1994), *Time Series Analysis*, Princeton, NJ: Princeton University Press.

Harvey, A. C. (1981), *Time Series Models*, New York: John Wiley & Sons.

Harvey, A. C. (1984), "A Unified View of Statistical Forecasting Procedures," *Journal of Forecasting*, 3, 245–275.

Hopewood, W. S., McKeown, J. C., and Newbold, P. (1984), "Time Series Forecasting Models Involving Power Transformations," *Journal of Forecasting*, 3, 57–61.

Hyndman, R. J. (2006), "Another Look at Forecast-Accuracy Metrics for Intermittent Demand," *Foresight*, 43–46.

Ledolter, J. and Abraham, B. (1984), "Some Comments on the Initialization of Exponential Smoothing," *Journal of Forecasting*, 3, 79–84.

Ljung, G. M. and Box, G. E. P. (1978), "On a Measure of Lack of Fit in Time Series Models," *Biometrika*, 65, 297–303.

Makridakis, S. G., Wheelwright, S. C., and McGee, V. E. (1983), *Forecasting: Methods and Applications*, 2nd Edition, New York: John Wiley & Sons.

McKenzie, E. (1984), "General Exponential Smoothing and the Equivalent ARMA Process," *Journal of Forecasting*, 3, 333–344.

McKenzie, E. (1986), "Error Analysis for Winters' Additive Seasonal Forecasting System," *International Journal of Forecasting*, 2, 373–382.

Montgomery, D. C. and Johnson, L. A. (1976), *Forecasting and Time Series Analysis*, New York: McGraw-Hill.

Morf, M., Sidhu, G. S., and Kailath, T. (1974), "Some New Algorithms for Recursive Estimation on Constant Linear Discrete Time Systems," *IEEE Transactions on Automatic Control*, 19, 315–323.

Nelson, C. R. (1973), *Applied Time Series for Managerial Forecasting*, San Francisco: Holden-Day.

Newbold, P. (1981), "Some Recent Developments in Time Series Analysis," *International Statistical Review*, 49, 53–66.

Pankratz, A. and Dudley, U. (1987), "Forecast of Power-Transformed Series," *Journal of Forecasting*, 6, 239–248.

Priestley, M. B. (1981), *Spectral Analysis and Time Series*, London: Academic Press.

Roberts, S. A. (1982), "A General Class of Holt-Winters Type Forecasting Models," *Management Science*, 28, 808–820.

Sweet, A. L. (1985), "Computing the Variance of the Forecast Error for the Holt-Winters Seasonal Models," *Journal of Forecasting*, 4, 235–243.

Willemain, T. R., Smart, C. N., and Shocker, J. H. (1994), "Forecasting Intermittent Demand in Manufacturing: Comparative Evaluation of Croston's Method," *International Journal of Forecasting*, 10, 529–538.

Winters, P. R. (1960), "Forecasting Sales by Exponentially Weighted Moving Averages," *Management Science*, 6, 324–342.

Yar, M. and Chatfield, C. (1990), "Prediction Intervals for the Holt-Winters Forecasting Procedure," *International Journal of Forecasting*, 6, 127–137.

# Chapter 20
# Forecast Combination Computational Details

**Contents**


Forecast Combination Model . . . . . **607**
Forecast Combination Process . . . . . **609**
Encompassing Tests . . . . . **611**
OLS and HLN Encompassing Tests . . . . . 611
Methods for Determining Forecast Combination Weights . . . . . **614**
Simple Average . . . . . 614
User-Specified Weights . . . . . 614
Rank Weights . . . . . 615
Ranked User Weights . . . . . 615
RMSE Weights . . . . . 615
AICC Weights . . . . . 616
Ordinary Least Squares (OLS) Weights . . . . . 616
Restricted Least Squares Weights . . . . . 617
Least Absolute Deviation (LAD) Weights . . . . . 617
Weighted Combination of Selected Forecasts . . . . . **619**
Combined Forecast Error Variance Estimation . . . . . **620**
References . . . . . **621**


## Forecast Combination Model

Given a time series $Y_t$ and set of $M$ forecasting models denoted by $F_i(), i = 1, \ldots, M$ with possible inputs (not shown), the following relationship is presumed for each

$$y_t = F_i(Y_{t-1}) + \epsilon_{i,t}$$

where $Y_{t-1}$ denotes a vector of time series observations up to time $t-1$ and $\epsilon_{i,t}$ are IID disturbances with respect to $t$ assumed $E[\epsilon_{i,t}] = 0$. Forecasting results from the set of models $F_i()$ are presumed to be available at the outset of the combination processing. They are generated on demand from running the models and consuming their results to generate the forecast combination.

Independently, each model $F_i()$ is fit to produce a fitted forecast model that is denoted $\hat{F}_i()$. Define the following notation for the results of $\hat{F}_i()$:

$\hat{y}_{i,t}$ denotes the prediction for fitted model $\hat{F}_i()$ at time $t$.

$e_{i,t} = y_t - \hat{y}_{i,t}$ denotes the prediction error for fitted model $\hat{F}_i()$ at time $t$.

$\hat{\sigma}^2_{i,t} = \mathrm{Var}(e_{i,t})$ denotes the prediction error variance for fitted model $\hat{F}_i()$ at time $t$.

$\hat{\sigma}_{i,t} = \sqrt{\mathrm{Var}(e_{i,t})}$ denotes the prediction standard error for fitted model $\hat{F}_i()$ at time $t$.

The combined forecast, denoted $y_c$, uses combination weights, $w_i$, to combine the fitted model forecasts as a weighted average of the predicted values with possible restrictions on the combination weights. More precisely,

$\hat{y}_{c,t} = \sum_{i=1}^{M} w_i \hat{y}_{i,t}$ denotes the combined forecast at time $t$.

The combination weights can correspond to a simple average, they can be user-specified, or they can be estimated using a variety of methods from the fitted forecast results. For the present discussion, the estimated weights are denoted $\hat{w}_i$ regardless of how they are obtained. This leads to the following expressions for the combined forecast and its prediction error:

$\hat{y}_{c,t} = \sum_{i=1}^{M} \hat{w}_i \hat{y}_{i,t}$ denotes the combined forecast prediction at time $t$.

$e_{c,t} = y_t - \hat{y}_{c,t}$ denotes the combined forecast prediction error at time $t$.

The prediction error variances are most generally defined by:

$$\begin{aligned}
\mathrm{Var}(e_{c,t}) &= \mathrm{Var}(y_t - \hat{y}_{c,t}) \\
&= \mathrm{Var}(y_t - \sum_{i=1}^{M} \hat{w}_i \hat{y}_{i,t}) \\
&= \mathrm{Var}(y_t - \sum_{i=1}^{M} \hat{w}_i (y_t - e_{i,t}) \\
&= \mathrm{Var}(y_t (1 - \sum_{i=1}^{M} \hat{w}_i) + \sum_{i=1}^{M} \hat{w}_i e_{i,t})
\end{aligned}$$

When you assume or impose the constraint $\sum_{i=1}^{M} \hat{w}_i = 1$, simplification results in

$$\begin{aligned}
\mathrm{Var}(e_{c,t}) &= \mathrm{Var}(\sum_{i=1}^{M} \hat{w}_i e_{i,t}) \\
&= \hat{W}_t^T \Sigma_t \hat{W}_t
\end{aligned}$$

where $\hat{W}_t$ denotes the fitted weight vector $M \times 1$ and $\Sigma_t = \mathrm{Var}(E_t)$ with $E_t = (e_{1,t}, e_{2,t}, \ldots, e_{M,t})^T$ denotes the ensemble prediction error vector at time $t$.

Alternatively, this can be expressed as:

$$\begin{aligned}\mathrm{Var}(e_{c,t}) &= \sum_{i=1}^{M} \hat{w}_i^2 \mathrm{Var}(e_{i,t}) + 2\sum_{j<i} \hat{w}_i \hat{w}_j \mathrm{Cov}(e_{i,t}, e_{j,t}) \\ &= \sum_{i=1}^{M} \hat{w}_i^2 \sigma_{i,t}^2 + 2\sum_{j<i} \hat{w}_i \hat{w}_j \rho_{i,j,t} \sigma_{i,t} \sigma_{j,t}\end{aligned}$$

If the prediction errors for the individual forecasts are not correlated at time $t$, then

$$\Sigma_t = \mathrm{diag}(\sigma_{i,t}^2)$$

and the combined forecast prediction error variance and standard error series are simply

$$\begin{aligned}\mathrm{Var}(e_{c,t}) &= \textstyle\sum_{i=1}^{M} \hat{w}_i^2 \sigma_{i,t}^2 \\ \hat{\sigma}_t &= \sqrt{\textstyle\sum_{i=1}^{M} \hat{w}_i^2 \sigma_{i,t}^2}\end{aligned}$$

Otherwise, computation of prediction error variance estimates is more complicated. See the section "Combined Forecast Error Variance Estimation" on page 620 for more discussion about this topic.

## Forecast Combination Process

As described in the previous section, forecast combination operates in the context of a set of forecast models, $\{F_i()\}_{i=1}^{M}$, and uses results of the set of the corresponding fitted models $\{\hat{F}_i()\}_{i=1}^{M}$. That discussion of the mathematical details of forecast combination presumes all of the forecasts in the set $F_i()$ are used in the combination. The set of candidate forecasts considered during a forecast combination can in fact be a superset of the active set of forecasts that are finally combined. Denote the candidate set of forecasts for consideration as $\mathcal{F}(0)$. Elements of this set are some collection of forecasts $F_i(), i = 1, \ldots, M$. This set of candidate forecasts is defined through the XML that is generated from the HPFSELECT procedure when you create a combined model list.

A concept used in the remainder of this chapter is that of a time interval for weight estimation. Denote the time index set for an estimation interval region by

$$\mathcal{T}_W = \{t_b^W, \ldots, t_e^W\}$$

where $t_b^W$ and $t_e^W$ denote the begin and end time indices for the weight estimation region.

The forecast combination process follows a sequence of steps that operates on and within a set of forecasts. In the abstract, step $i$ in this process takes a set $\mathcal{F}(i-1)$ as input and produces a set $\mathcal{F}(i)$ as output to the next step. As discussed, some steps function to potentially reduce the candidate set $\mathcal{F}(i-1)$ by the application of some form of test. The following sequence reduces the initial candidate set, $\mathcal{F}(0)$, to a final set of forecasts that are combined in the final step of the process:

1. Forecast candidate tests:
   Tests for seasonality and intermittency can be specified in the combined model list. As for the model selection list, these tests are determined by diagnostics on the actual time series $y_t$. The tests have the same behavior as those specified in model selection list context. As a result of these tests, a possibly reduced set of candidate forecasts $\mathcal{F}(1)$ is generated. If $\mathcal{F}(1)$ is empty, no further processing is performed and the combined forecast produces no result.

2. Forecast missing value tests:
   For each $F_i() \in \mathcal{F}(1)$, two forms of missing value tests can be performed on its predicted series $\hat{y}_{i,t}, t = 1, \ldots, T$. In the following, define

$$\text{MISSING}(x) = \begin{cases} 1 & \text{if } x \text{ is a missing value} \\ 0 & \text{otherwise} \end{cases}$$

   Forecast fit region percentage missing: A test for missing forecast values over the fit region of the $\hat{y}_{i,t}$ series can be specified as a threshold, $P_{miss}$. The percentage missing from $\hat{y}_{i,t}$ is defined by

$$p_i = 100[\frac{\sum_{t=1}^{T} \text{MISSING}(\hat{y}_{i,t})}{|\mathcal{T}_W|}]$$

   where $|\mathcal{T}_W|$ denotes the number of time indices in the estimation region. If $p_i \geq P_{miss}$, then the forecast for candidate $F_i()$ is omitted from $\mathcal{F}(1)$.

   Forecast horizon percentage missing: A forecast horizon missing test can be specified as a threshold, $H_{miss}$. The forecast horizon is the range of time indices that correspond to multistep forecasts denoted by $[t_b^H, t_e^H]$ with span $H = t_e^H - t_b^H$. The percentage of missing values from the predicted series $\hat{y}_{i,t}$ is then defined by

$$p_i = 100[\frac{\sum_{t=t_b^H}^{t_e^H} \text{MISSING}(\hat{y}_{i,t})}{H}]$$

   If $p_i \geq H_{miss}$, then the forecast for candidate $F_i()$ is omitted from $\mathcal{F}(2)$.

   The end result of this step is the set of candidates $\mathcal{F}(2) \subseteq \mathcal{F}(1)$.

3. Encompassing tests:
   For the set $\mathcal{F}(2)$, a test can be performed to determine which of the forecasts $F_i() \in \mathcal{F}(1)$ adds information to the final combined forecast. By default, no encompassing tests are performed on the combination candidates. The goal of this step is to produce a new set of candidates $\mathcal{F}(3)$ in which each forecast contributes information to the final combination. If $\mathcal{F}(2)$ contains only one element, no test is performed. The details of encompassing tests are further discussed the section "Encompassing Tests" on page 611.

4. Weight estimation:
   This step assigns weights to the forecasts that are present in the set $\mathcal{F}(3)$ as directed by the combined model list. A detailed discussion of weight methods is provided in the section "Methods for Determining Forecast Combination Weights" on page 614. Provision is made for weight assignment to reduce

the set of forecasts based on method-specific criteria. For some estimation methods, a weight might be non-estimable, for example, and the corresponding forecast is eliminated from use in the combined forecast. For others, a user-defined criterion allows for exclusion of weights that fall below a given threshold; hence the corresponding forecasts are eliminated from use in the combined forecast. The result is that a set of weights for the forecasts in $\mathcal{F}(4)$ is produced by this step.

5. Forecast combination:
   This step performs the weighted combination of the forecasts in the set $\mathcal{F}(4)$. For each $t, t = 1, \ldots, T$, forecast combination produces a value for $\hat{y}_{c,t}$ according to the expression

$$\hat{y}_{c,t} = \sum_{i=1}^{M} \hat{w}_{i,t}\, \hat{y}_{i,t}$$

   where $\hat{y}_{i,t}$ denotes the prediction for fitted model $\hat{F}_i()$ at time $t$ and $\hat{w}_{i,t}$ denotes the weight assigned to fitted model $\hat{F}_i()$ at time $t$.

   This expression provides for the possibility that the weight estimates assigned in the previous step can be adjusted for each time slice $t$ in the panel of forecasts being combined. This permits treatment of missing values in one or more of the $\hat{y}_{i,t}$ series at a given time index $t$. The combined model list specifies a treatment mode for missing forecast values in the combination step. The default treatment mode is dependent on the weight estimation method. For more details about treatment modes, see "Weighted Combination of Selected Forecasts" on page 619.

# Encompassing Tests

Methods for testing forecast encompassing are somewhat varied in the literature, ranging from relatively simple pairwise comparisons to more complicated schemes based on causal inference and D-separation which model the causal relationships in a directed acyclic graph (DAG). The current set of methods supported includes the following:

- Ordinary least squares (OLS) test
- Harvey, Leybourne, and Newbold (HLN) test

Both OLS and HLN tests are very similar in their elimination process and are discussed together in the next section.

## OLS and HLN Encompassing Tests

The elimination process begins by ranking all of the forecasts in $\mathcal{F}(2)$ by a designated statistic-of-fit criterion value from the best to worst performing over the range of time indices $\mathcal{T}_W$. Option settings in effect during the PROC HPFENGINE run and the actual time series $y_t$ determine $\mathcal{T}_W$.

The elimination process is iterative. Each pass begins with the best-ranked forecast of those not previously certified and compares it to each of the lower-ranked ones in the rank sequence. Suppose $F_i()$ from $\mathcal{F}(2)$

is the best-ranked of those not already certified. At this point, $F_i()$ becomes certified for inclusion. What remains then is to determine whether any lower-ranked uncertified forecast $F_j()$ is encompassed by $F_i()$. Define that forecast $F_i()$ encompasses forecast $F_j()$ if the encompassing test on the pair $(F_i(), F_j())$ fails to reject the null hypothesis of the encompassing test. Both OLS and HLN encompassing tests produce a statistic that has a $t$ distribution. The OLS test uses a one-sided alternative in the right tail, while the HLN test uses a two-tailed test. The details of the $t$ statistic computation and its associated degrees of freedom are provided later, but are not needed to define the process. The prediction error values $\hat{e}_{i,t}$ and $\hat{e}_{j,t}$ are used to compute this $t$ statistic for both of these tests. Call it $T_{i,j}$ and denote its associated degrees of freedom as $v_{i,j}$. Note that $T_{i,j}$ estimates the degree of orthogonality between $F_i()$ and $F_j()$. The null is then

$$H_0: T_{i,j} = 0$$

When $H_0$ is rejected, encompassing is not conclusive and the forecast $F_j()$ remains in $\mathcal{F}(2)$. When $H_0$ is not rejected, encompassing is concluded at the specified confidence level and the forecast $F_j()$ is removed from $\mathcal{F}(2)$. This pairwise comparison process proceeds through all of the models ranked lower than $F_i()$. At the end of the pass on $F_i()$, one of two conditions exists: there are no lower-ranked elements that remain in $\mathcal{F}(2)$ or else there are more. If no more lower-ranked forecasts remain, then the set of certified forecast candidates in $\mathcal{F}(2)$ are passed along to the next step. If there are more, another pass is started with the best of those forecasts that remain in $\mathcal{F}(2)$ and are not already certified (that is, tested against those of lower rank than itself). This process terminates with some collection of certified forecasts to be combined being those left in the set $\mathcal{F}(2)$.

## OLS Test Statistic

This test regresses a convex combination of two prediction error series to test the hypothesis that the combination error is best explained by the use of both forecasts in combination. The mathematical details of this regression are summarized here. Take two forecasts, $F_{1,t}$ and $F_{2,t}$ over time $t = 1, \ldots T$, and postulate the model:

$$F_{c,t} = (1 - \lambda)F_{1,t} + \lambda F_{2,t} + \epsilon_t, \text{ where } 0 \leq \lambda \leq 1$$

Define $e_{i,t} = y_t - F_{i,t}, i = 1, 2$, as the respective prediction errors with $\epsilon_t$ as the prediction error of the combined forecast. Then observe that the proposed regression is equivalent to:

$$e_{1,t} = \lambda(e_{1,t} - e_{2,t}) + \epsilon_t$$

The combined forecast has a smaller expected error variance than $F_{1,t}$ unless the covariance between $e_{1,t}$ and $(e_{1,t} - e_{2,t})$ is 0. Rejecting the hypothesis $\lambda = 0$ means there is statistical evidence to conclude that this covariance is not 0. Failure to reject means that $\lambda$ is not significantly different from 0. Respectively, these amount to rejecting the hypothesis that $F_{1,t}$ encompasses $F_{2,t}$, or in failing to do so, that $F_{1,t}$ encompasses $F_{2,t}$ and that $F_{2,t}$ should not be included in the combined forecast. A great deal has been written about the use of OLS to estimate the parameter $\lambda$ in this setting and the Type I errors that result from tests that use such estimates. Harvey, Leybourne, and Newbold (1998) discuss this at some length and argue for more robust encompassing tests including their HLN test.

## HLN Test Statistic

The HLN test statistic proposed and studied by Harvey, Leybourne, and Newbold (1998) is a modified form of the Diebold-Mariano (DM) test. The mathematical details of the HLN test are summarized here. Diebold and Mariano (1995) proposed a test statistic for the equality of prediction mean squared errors that can be adapted to test for forecast encompassing. Given two forecast error series $e_{i,t}$, define a loss differential sequence $d_t$ by

$$d_t = (e_{1,t} - e_{2,t})e_{1,t}, t = 1, \ldots, T$$

To test the null hypothesis that $E[d_t] = 0$, the DM test statistic is defined as the ratio

$$\mathrm{DM} = \frac{\overline{d}}{\sqrt{\mathrm{Var}(\overline{d})}}$$

Assuming $k$-step ahead forecasts and that $k$-step ahead forecasts exhibit dependence up to lag $k-1$, a consistent estimator of variance in the DM statistic is

$$\hat{\mathrm{Var}}(\overline{d}) = \frac{1}{T}[\gamma_{\overline{d}}(0) + 2\sum_{h=1}^{k-1}\gamma_{\overline{d}}(h)]$$

where $\gamma_{\overline{d}}(h)$ is the autocovariance of $\overline{d}$ at lag $h$. $\gamma_{\overline{d}}(h)$ is estimated in the usual way via

$$\hat{\gamma}_{\overline{d}}(h) = \frac{1}{T}\sum_{t=h+1}^{T}(d_t - \overline{d})(d_{t-h} - \overline{d})$$

Harvey, Leybourne, and Newbold (1998) further refine this by creating a modified DM test statistic denoted MDM:

$$\mathrm{MDM} = T^{-1/2}[T + 1 - 2k + T^{-1}k(k-1)]^{1/2}\mathrm{DM}$$

They compare the resulting statistic against critical values from a $t_{T-1}$ distribution. These two modifications to the DM test result in the HLN test for forecast encompassing. The test for encompassing then tests the null:

$$H_0: MDM = 0$$

Rejecting $H_0$ is inconclusive for $F_i()$ encompassing $F_j()$, and $F_j()$ remains in the forecast set. Failure to reject $H_0$ concludes that $F_i()$ encompasses $F_j()$ at the specified confidence level, and $F_j()$ is removed from the forecast set.

# Methods for Determining Forecast Combination Weights

Forecast combination weights $\hat{w}_i, i = 1, \ldots, M$, are determined via one of the following methods:

- simple average
- user-defined weights
- rank weights
- ranked user weights
- root mean squared error (RMSE) weights
- corrected Akaike's information criterion (AICC) weights
- ordinary least squares (OLS) weights
- restricted least squares weights,
- least absolute deviation (LAD) weights

In all of these methods, the original set of candidate forecasts, previously called $\mathcal{F}(0)$, can be reduced through steps that lead to the estimation of combination weights. For purposes of the following discussion, denote the possibly reduced set of forecasts as $\mathcal{F}$ and denote its index set by

$$\mathcal{I}(\mathcal{F}) = \{i : F_i() \in \mathcal{F}\}$$

Subsequent sections describe each of these methods.

## Simple Average

The combination weights are determined solely by the cardinality of the set $\mathcal{F}$. Define the combination weight estimates for the forecasts in $\mathcal{F}$ by

$$\hat{w}_i = \frac{1}{M}, \forall i \in \mathcal{I}(\mathcal{F})$$

where $M = |\mathcal{F}|$

## User-Specified Weights

The combined model list from PROC HPFSELECT that specifies this method includes a set of user-defined weights, $w_i, i = 1, \ldots, M$, which correspond to the forecasts $F_i(), i = 1, \ldots, M$ specified in the combined model list.

Define the combination weight estimates for the forecasts in $\mathcal{F}$ by

$$\hat{w}_i = \frac{w_i}{W}, \forall i \in \mathcal{I}(\mathcal{F})$$

where $W = \sum_{i \in \mathcal{I}(\mathcal{F})} w_i$.

## Rank Weights

This method uses a user-designated statistic-of-fit criterion to rank the performance of the forecasts that comprise the set $\mathcal{F}$ over a fit region $\mathcal{T}_W$. Define $\mathcal{R}$ to be the ordered set of ranked forecasts $F_i()$ in $\mathcal{F}$, omitting any forecast with a noncomputable value, and let

$$R_i = \mathrm{RANK}(F_i), \forall i \in \mathcal{I}(\mathcal{F})$$

where $\mathrm{RANK}(F_i)$ denotes the position of $F_i$ in the ordered set $\mathcal{R}$.

Define the combination weight estimates for the forecasts in $\mathcal{F}$ by

$$\hat{w}_i = \frac{1/R_i}{W}, \forall i \in \mathcal{I}(\mathcal{F})$$

where $W = \sum_{i \in \mathcal{I}(\mathcal{F})} 1/R_i$.

For discussion of the use of this method, see Kişinbay (2007); Aiolfi and Timmerman (2006).

## Ranked User Weights

This weight estimation method is a combination of the preceding rank weights and user-defined weights methods. Given the set of forecasts $\mathcal{F}$, the forecasts are ranked by a designated statistic-of-fit criterion over the estimation region $\mathcal{T}_W$. Then the first user-defined weight is associated with the best forecast, the second user-defined weight is associated with the second best forecast, and so on. The resulting weight assignments are then normalized by the same procedure as for user-defined weights.

## RMSE Weights

This weight estimation method uses root mean squared error (RMSE) values computed from the prediction error series associated with each forecast $F_i() \in \mathcal{F}$. RMSE values are computed over a fit region $\mathcal{T}_W$. This method can result in a reduction of the set $\mathcal{F}$ if the computation of the RMSE value for the prediction error series fails for a given forecast.

Define

$$\mathrm{RMSE}_i = \mathrm{RMSE}(\hat{\mathbf{e}}_i), \forall i \in \mathcal{I}(\mathcal{F})$$

where $\hat{\mathbf{e}}_i$ is the prediction error series vector for $F_i() \in \mathcal{F}$.

Remove any $F_i()$ with a noncomputable RMSE from $\mathcal{F}$ and its corresponding index $i$ from $\mathcal{I}(\mathcal{F})$.

Define the combination weight estimates for the forecasts in $\mathcal{F}$ by

$$\hat{w}_i = \frac{1/\mathrm{RMSE}_i}{W}, \forall i \in \mathcal{I}(\mathcal{F})$$

where $W = \sum_{i \in \mathcal{I}(\mathcal{F})} 1/\mathrm{RMSE}_i$.

A reference that discusses the use of this method is Kişinbay (2007).

## AICC Weights

This weight estimation method uses corrected Akaike's information criterion (AICC) values computed from the prediction error series that are associated with each forecast $F_i() \in \mathcal{F}$. This method can result in a reduction of the set $\mathcal{F}$ if the computation of the AICC value fails for the prediction error series for a given forecast. Let $P_i$ denote the number of parameters in the forecast model $F_i \in \mathcal{F}$ and define

$$\mathrm{AICC}_i = \mathrm{AICC}(\hat{\mathbf{e}}_i, P_i), \forall i \in \mathcal{I}(\mathcal{F})$$

where $\hat{\mathbf{e}}_i$ is the prediction error series vector for $F_i() \in \mathcal{F}$.

Remove any $F_i()$ with a noncomputable AICC from $\mathcal{F}$ and its corresponding index from $\mathcal{I}(\mathcal{F})$. Denote the least $\mathrm{AICC}_i$ value computed as $\mathrm{AICC}_{\min}$ and define

$$\Delta_i = \mathrm{AICC}_i - \mathrm{AICC}_{\min}, \forall i \in \mathcal{I}(\mathcal{F})$$

Define the combination weight estimates for the forecasts in $\mathcal{F}$ by

$$\hat{w}_i = \frac{\exp(-\frac{1}{2}\lambda\Delta_i)}{W}$$

where $W = \sum_{i \in \mathcal{I}(\mathcal{F})} \exp(-\frac{1}{2}\lambda\Delta_i)$ and $0 \leq \lambda$. When $0 \leq \lambda \leq 1$, the resulting method is a compromise between simple forecast averaging (obtained when $\lambda = 0$) and the usual Akaike weights (obtained when $\lambda = 1$). When $\lambda$ is set to large values, say $\lambda > 10$, the resulting method tends to select the best single model based on its AICC performance. Frequently, the tendency to effectively pick the best forecast based on its AICC value is observed even when $\lambda = 1$ unless the other candidate forecasts have AICC values that are reasonably close to the smallest.

## Ordinary Least Squares (OLS) Weights

OLS weights are computed by solving an unconstrained least squares problem. As usual, the weights are computed via the normal equations to minimize

$$z = \sum_{t=1}^{T}(y_t - \sum_{i=1}^{M} w_i \hat{y}_{i,t})^2$$

The resulting weight estimates $\hat{w}_i$ are used to combine the forecasts in $\mathcal{F}$. This estimation does not impose a constraint that the weights to sum to one. Therefore, results in section "Forecast Combination Model" on page 607 related to that assumption do not hold for OLS weights.

## Restricted Least Squares Weights

Three weight methods, NRLS, ERLS, and NERLS, fall into this category. All are formulated as a constrained least squares problem and solved through a nonlinear optimization solver. All forms minimize the objective function:

$$z = \sum_{t \in \mathcal{T}_W} (y_t - \sum_{i=1}^{M} w_i \hat{y}_{i,t})^2$$

Constraints imposed by methods cast into this form include:

- NRLS (nonnegative restricted least squares) imposes the constraints:

$$w_i \geq 0, i = 1, \ldots, M$$

- ERLS (equality restricted least squares) imposes the constraint:

$$\sum_{i=1}^{M} w_i = 1$$

- NERLS (nonnegative equality restricted least squares) imposes the constraints:

$$\sum_{i=1}^{M} w_i = 1,$$
$$w_i \geq 0, i = 1, \ldots, M$$

The resulting weight estimates $\hat{w}_i$ are used to combine the forecasts in $\mathcal{F}$.

## Least Absolute Deviation (LAD) Weights

Many time series statistic-of-fit criteria measure some form of absolute deviation of the forecast series from the actual series, so it is natural to cast the combination weights problem into some form of least absolute deviations (LAD) setting. LAD weights are determined by formulating a linear program (LP) to minimize an objective function expressed in terms of absolute losses at each time index $t$.

A loss series, denoted $u_t$, is constructed for $t \in \mathcal{T}_W$, depending on the specification of the type of loss to minimize in the objective. Define $u_t$ as follows for these different loss types:

- Simple error

$$u_t = y_t - \sum_{i=1}^{M} w_i \hat{y}_{i,t}$$

  This corresponds to ERRTYPE=ABS in the COMBINE statement.

- Percentage error

$$u_t = \frac{y_t - \sum_{i=1}^{M} w_i \hat{y}_{i,t}}{y_t}$$

  This corresponds to ERRTYPE=APE in the COMBINE statement.

- Relative error

$$u_t = \frac{y_t - \sum_{i=1}^{M} w_i \hat{y}_{i,t}}{y_t - \eta_t}$$

where $\eta_t$ denotes the naive forecast of $y_t$. Under normal circumstances $\eta_t = y_{t-1}$ except when $y_{t-1}$ is missing. In those cases $\eta_t$ holds the last nonmissing $y_{t-1}$ value until a nonmissing $y_t$ is encountered to replace in lagged context. Evidently, $u_t$ can be defined only when both $y_t$ and $\eta_t$ are contemporaneously nonmissing. This corresponds to ERRTYPE=RAE in the COMBINE statement.

The optimization problem (LAD) is posed in terms of this loss series. LAD weights can be determined to minimize one of these objectives:

- $\ell_1$ norm of the loss series produces an objective in the form:

$$\min z = \sum_{t \in \mathcal{T}_W} |u_t|$$

This corresponds to OBJTYPE=L1 in the COMBINE statement.

- $\ell_\infty$ norm of the loss series produces an objective of the form:

$$\min z = \max \{|u_t|\}_{t \in \mathcal{T}_W}$$

This corresponds to OBJTYPE=LINF in the COMBINE statement.

For details about the COMBINE statement syntax, see the section "COMBINE Statement" on page 464.

So the problem (LAD) takes this abstract form:

$$\begin{aligned} &\min z = f(\mathbf{y}, \hat{\mathbf{Y}}, \mathbf{w}) \\ &\text{subject to:} \\ &\mathbf{1}^T \mathbf{w} = 1, \\ &\mathbf{w} \geq 0 \end{aligned}$$

where $\mathbf{w}$ is the $M \times 1$ matrix of combination weights, $\mathbf{y}$ is the $T \times 1$ matrix of actual values, $\hat{\mathbf{Y}}$ is the $T \times M$ matrix of forecasts, and $\mathbf{1}$ is an $M \times 1$ matrix of ones.

The abstractly posed form of the LAD problem is inherently nonlinear due to the absolute values in the objective. It can be linearized by introducing a different set of artificial decision variables and expressing the objective in terms of those. Constraints that involve those artificial decision variables must be added to insure that the objective evaluates in terms of absolute values. See Chvátal (1983) for details.

The essence of this translation into an LP uses the fact that

$$\xi = |u| \Leftrightarrow \xi \geq u \text{ and } \xi \geq -u$$

Using this, each $|u_t|$ in the $\ell_1$ objective is replaced by $\xi_t$

$$\min z = \sum_{t=1}^{T} \xi_t$$

and two constraints are added to the LP for each $u_t$:

$$\xi_t \geq u_t \text{ and } \xi_t \geq -u_t$$

For the $\ell_\infty$ objective, only one artificial variable $\xi_0$ is needed. The objective becomes

$$\min z = \xi_0$$

and two constraints are added to the LP for each $u_t$:

$$\xi_0 \geq u_t \text{ and } \xi_0 \geq -u_t$$

Three complications arise in the construction of the LP formulation:

- Missing values in forecasts:
  When $\hat{y}_{i,t}$ is missing, the row for time index $t$ is omitted from the $\hat{\mathbf{Y}}$ matrix.
- Missing values in the actual series:
  When $y_t$ is missing, the row that corresponds to $t$ is omitted from the $\hat{\mathbf{Y}}$ matrix
- Zero values in actual series:
  When the loss series is expressed in terms of percentage deviation, the $u_t$ value is undefined for any time index $t$ where $y_t$ is zero. For percentage deviation, when $y_t$ is zero, the row for time index $t$ is omitted from the $\hat{\mathbf{Y}}$ matrix.

In terms of the abstract LAD problem, all of these omissions affect the objective function. The same is true in the LP formulation, but the impact on the simplex of feasible solutions by the omission of some constraints becomes apparent. Omitting time indices from the formulation decreases the number of artificial decision variables added; hence fewer LP constraints incorporates less loss series information into the weight estimation.

Solving the LP for the LAD problem results in a set of weights $\hat{w}_i$ that optimize the selected objective function subject to the normalization and nonnegativity constraints on the weights.

## Weighted Combination of Selected Forecasts

Denote the set of forecasts $F_i()$ to combine as $\mathcal{F}$. As noted previously, the combined forecast $\hat{y}_{c,t}$ is defined by

$$\hat{y}_{c,t} = \sum_{i \in \mathcal{F}} \hat{w}_{i,t} \hat{y}_{i,t}$$

where $\hat{y}_{i,t}$ denotes the prediction for fitted model $\hat{F}_i()$ at time $t$ and $\hat{w}_{i,t}$ denotes the weight assigned to fitted model $\hat{F}_i()$ at time $t$.

The presence of missing values in the forecasts at a given time $t$ affects the generation of the combined forecast $\hat{y}_{c,t}$. The resulting value is determined by the forecast combination missing value treatment mode.

This behavior can be specified in the combined model list for the forecast combination and if specified there takes precedence over any default. The default value for missing value treatment mode is dependent on the weight method selected.

The following discussion presumes that at a given time $t$ one or more of the $\hat{y}_{i,t}$ values is missing. Given that premise, the behaviors defined for missing value treatment are:

- Rescale (MISSMODE=RESCALE):
  $\hat{w}_i$ values associated with the nonmissing $\hat{y}_{i,t}$ values are dynamically normalized to satisfy the requirement for the sum of weights equal one. Define

  $$\mathcal{N}_t = \{i : \hat{y}_{i,t} \text{ is nonmissing}\}$$

  and define $W$ as

  $$W = \sum_{i \in \mathcal{N}_t} \hat{w}_i$$

  Then the combined forecast $y_{c,t}$ is given by

  $$\hat{y}_{c,t} = \frac{1}{W} \sum_{i \in \mathcal{N}_t} \hat{w}_{i,t} \hat{y}_{i,t}$$

- Missing (MISSMODE=MISSING):
  The combined forecast $\hat{y}_{c,t}$ is set to missing.

The default missing value treatment is defined by weight method as follows:

- MISSMODE=RESCALE is the default for simple average, user-specified weights, ranked user weights, rank weights, and RMSE weights.
- MISSMODE=MISSING is the default for AICC weights, OLS weights, restricted least squares weights, and LAD weights. For OLS and NRLS, you are not permitted to specify MISSMODE=RESCALE since the estimated weights are not constrained to sum to one.

## Combined Forecast Error Variance Estimation

The combination weight estimates are treated as if they are known parameters in all the prediction error variance expressions shown in the section "Forecast Combination Model" on page 607.

To control the method of estimating prediction error variance, you can specify an option in the combined model list that defines the forecast combination. In all cases, the estimated prediction error variance of the combined forecast makes use of the estimates of prediction error variance from the forecasts that are finally combined and the exact weights used in that combination at each time $t$. See the section "Weighted Combination of Selected Forecasts" on page 619 for details that impact weights over the combination time span.

Prediction error variance estimation modes available for the combined forecast include:

STDERR=DIAG specifies that the variance computation assumes the forecast errors at time $t$ are uncorrelated so that the simple diagonal form of $\Sigma_t$ is used.

STDERR=ESTCORR specifies that the variance computation estimates correlations $\rho_{i,j,t}$ via the sample cross-correlation between $e_{i,t}$ and $e_{j,t}$ over the time span $t = 1, \ldots, T$, where $t$ denotes the last time index of the actual series $y_t$. Of course, using STDERR=ESTCORR implies that the error series $e_{i,t}$ and $e_{j,t}$ are assumed to be jointly stationary.

STDERR=ESTCORR(TAU=$\tau$) is similar to STDERR=ESTCORR except that the cross-correlation estimates are localized to a time window of $n$ steps. The time span $t = 1, \ldots, T$ is quantized into segments of $\tau$ steps working from $t$ backwards for in-sample cross-correlation estimates. The cross-correlation estimates from the interval $[T - \tau, T]$ are used for the period of multistep forecasts that extend beyond time $t$.

# References


Aiolfi, M. and Timmerman, A. (2006), "Persistence of Forecasting Performance and Combination Strategies," *Econometrics*, 135, 31–53.

Burnham, K. P. and Anderson, D. R. (1998), *Model Selection and Inference: A Practical Information-Theoretic Approach*, New York: Springer-Verlag.

Chvátal, V. (1983), *Linear Programming*, New York: W. H. Freeman.

Diebold, F. and Mariano, R. (1995), "Comparing Predictive Accuracy," *Journal of Business and Economic Statistics*, 13, 253–265.

Diebold, F. X. and Lopez, J. A. (1996), "Forecast Evaluation and Combination," in G. S. Maddala and C. R. Rao, eds., *Handbook of Statistics*, 241–268, Amsterdam: North-Holland.

Harvey, D., Leybourne, S., and Newbold, P. (1998), "Tests for Forecast Encompassing," *Journal of Business and Economic Statistics*, 16, 254–259.

Kişinbay, T. (2007), *The Use of Encompassing Tests for Forecast Combination*, Working paper WP/07/264, International Monetary Fund.

Little, M. (2009), "An Information Criterion-Based Method for Constructing Combining Weights for Multimodel Forecasting and Prediction," U.S. Patent Number 20090319310.

Winkler, R. L. and Makridakis, S. G. (1983), "The Combination of Forecasts," *Journal of the Royal Statistical Society, Series A*, 146, 150–157.

# Chapter 21
# Forecast Model Selection Graph Details

**Contents**


Introduction . . . . . **623**
Forecast Model Selection Graphs by Example . . . . . **624**
Example 21.1: Selection between Model Forecasts and a Combined Forecast . . . . . 624
Example 21.2: Combining Selected Forecasts . . . . . 627
Forecast Model Selection Graph Operation . . . . . **633**
OUTSTATSELECT Behavior . . . . . 636
OUTEST Behavior . . . . . 637
OUTSTAT Behavior . . . . . 637
OUTMODELINFO Behavior . . . . . 637
OUTSUM Behavior . . . . . 637
OUTCOMPONENT Behavior . . . . . 637
OUTFOR Behavior . . . . . 638
OUT Behavior . . . . . 638
ODS Print Behavior . . . . . 638
ODS Plot Behavior . . . . . 638


## Introduction

This chapter provides details on several aspects of the forecast model selection graph used in the SAS Forecast Server Procedures software.

The forecast model selection graph is a directed acyclic graph (DAG) defined through the use of the HPFSELECT procedure and the various model specification procedures. Nodes in the forecast model selection graph are the XML specifications, and the arcs are defined by the references in list specifications that are created via PROC HPFSELECT.

The forecast model selection graph supports a wide range of model selection and forecasting options and is an evolution of previous mechanisms. Earlier versions of SAS Forecast Server Procedures software permitted a one-level structure known as a model selection list. It held a set of alternative time series models evaluated to pick the best model for each time series realization where it was applied. With the addition of combined forecasts, the extension of the model selection list concept to the forecast model selection graph is a natural one. You can easily see the motivation with these two scenarios:

- selecting between the best performing individual model forecast and a combined forecast
- combining the best performing forecast from a model selection list with other forecasts

Given these scenarios, the need for the forecast model selection graph generalization becomes apparent. Indeed, supporting these two scenarios basically provides the framework for the generality of the forecast model selection graph. Subsequent sections of this chapter demonstrate by example use of these concepts and provide insight into any rules on usage imposed by the software.

# Forecast Model Selection Graphs by Example

## Example 21.1: Selection between Model Forecasts and a Combined Forecast

This example demonstrates the use of the model selection list to select between a reference model for a time series forecast and a combined forecast. The candidate models that make up the combined model list are forecast, combined using the simple average of the forecasts, and the final forecast selected based on best performance between the reference model's forecast and the combined forecast.

Begin by defining an exponential smoothing model (ESM) as the reference to use as the gauge for better performance. The ESM generated is denoted by the name T0.

```
proc hpfesmspec rep=work.rep specname=t0;
   esm method=best;
run;
```

Define three forecast candidates to be considered for combination. Each of these models is forecast independently, and the results are used in a combined forecast definition.

```
proc hpfesmspec rep=work.rep specname=t1;
   esm method=bestn;
run;

proc hpfesmspec rep=work.rep specname=t2;
   esm method=bests;
run;

proc hpfarimaspec rep=work.rep specname=t3;
   forecast symbol=y dif=(1,s) q=(1)(1)s noint;
   estimate method=ml converge=.0001 delta=.0001 maxiter=150;
run;
```

The following statements define a combined model list. Note that the combined model list has a name, COMB3, and it specifies the three models, T1, T2, and T3, which are identified by their names in the SPEC statement. Recognizing this connection through XML specification names is the key to building forecast model selection graphs. In this example, the nonseasonal ESM BESTN defined in specification T1 is excluded from the combination because of the seasonality test performed by the combined model list.

```
proc hpfselect rep=work.rep name=comb3 label="Average(T1,T2,T3)";
   combine method=average;
   spec t1 t2 t3;
run;
```

The next group of statements defines the model selection list to select between the better of the reference ESM forecast and the combined forecast. Note again that this declaration relies on the names used in the SPEC statement in PROC HPFSELECT to define the set of candidates that comprise the list. The essential difference is the semantic intent for the model selection list, which is to choose the best forecast from the set of candidates as opposed to combining them as in the previous group of statements. The default behavior of the PROC HPFSELECT invocation is to define a model selection list; when you add the COMBINE statement to the PROC HPFSELECT statement block, you define a combined model list for the candidates identified via the SPEC statements. This is discussed in greater detail in section "Forecast Model Selection Graph Operation" on page 633.

```
proc hpfselect rep=work.rep name=select;
   spec comb3 t0;
run;
```

Run the model selection list to select between the better of the combined forecast and the reference ESM forecast.

```
proc hpfengine data=sashelp.air
               rep=work.rep
               globalselection=select
               out=out1
               lead=24
               print=(all)
               plot=(components);
   id date interval=month;
   forecast air;
run;
```

It can be seen in Output 21.1.1 that the combined forecast achieves the better performance between the two competing forecasts in the model selection list.

**Output 21.1.1** Model Selection Results

**The HPFENGINE Procedure**

| Model Selection Criterion = MAPE | | | |
|---|---|---|---|
| **Model** | **Statistic** | **Selected** | **Label** |
| **COMB3** | 2.9607784 | Yes | Average(T1,T2,T3) |
| **T0** | 3.0845016 | No | Best Smoothing Method |

Output 21.1.2 and Output 21.1.3 show parameter estimates for the forecasts that contribute to the combination. Output 21.1.4 shows the parameter estimates for the combined forecast.

**Output 21.1.2** Parameter Estimates for the T2 Model

**Parameter Estimates for T2 Model**

| Component | Parameter | Estimate | Standard Error | t Value | Approx Pr > \|t\| |
|---|---|---|---|---|---|
| AIR | Level Weight | 0.30728 | 0.03153 | 9.74 | <.0001 |
| AIR | Trend Weight | 0.0010000 | 0.0030237 | 0.33 | 0.7413 |
| AIR | Seasonal Weight | 0.87493 | 0.07769 | 11.26 | <.0001 |

**Output 21.1.3** Parameter Estimates for the T3 Model

**Parameter Estimates for T3 Model**

| Component | Parameter | Estimate | Standard Error | t Value | Approx Pr > \|t\| |
|---|---|---|---|---|---|
| AIR | MA1_1 | 0.30868 | 0.08433 | 3.66 | 0.0003 |
| AIR | MA2_12 | 0.10744 | 0.10190 | 1.05 | 0.2917 |

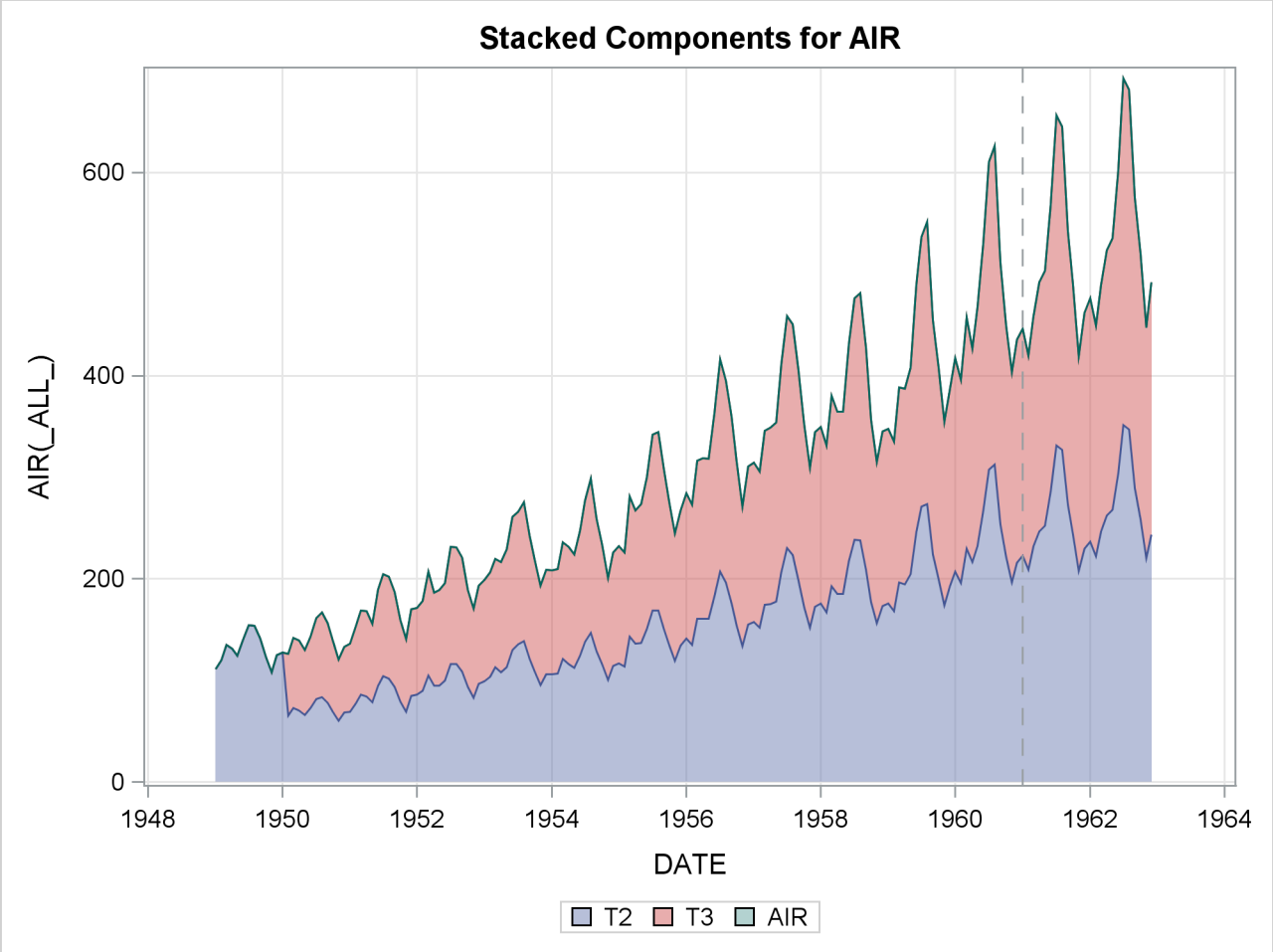
**Output 21.1.4** Parameter Estimates for COMB3 Combined Forecast

**Parameter Estimates for COMB3 Model**

| Component | Parameter | Estimate | Standard Error | t Value | Approx Pr > \|t\| |
|---|---|---|---|---|---|
| T2 | WEIGHT | 0.50000 | . | . | . |
| T3 | WEIGHT | 0.50000 | . | . | . |

The weighted forecast components that contribute to the final combined forecast are plotted in a stack on the same set of axes. The weighted forecast components are plotted from the bottom to the top in the order of their combination, yielding the final combined forecast.

**Output 21.1.5** Combined Forecast Components




## Example 21.2: Combining Selected Forecasts

This example demonstrates combining the forecast from a time series model with the best forecast selected from a model selection list. The candidates in the model selection list, named TBEST, are forecast, and the best is selected. That forecast is combined with the forecast from the ARIMA model named AIRLINE. Combination weights for the final forecast are generated based on corrected Akaike's information criterion (AICC) weights.

The following statements define two forecast candidates, T1 and T2, to be considered for model selection. Each of these models is forecast independently, and the best forecast is selected by the model selection list named TBEST. TBEST inherits its holdout and criterion from the root model selection list, SELECT, which is defined in the next group of statements. See "Forecast Model Selection Graph Operation" on page 633 for the definition of the root model selection list.

```
proc hpfesmspec rep=work.rep specname=t1;
   esm method=bests;
run;

proc hpfucmspec rep=work.rep specname=t2;
   level;
   slope;
   irregular;
   season type=dummy length=s;
run;

proc hpfselect rep=work.rep name=tbest;
   spec t1 t2;
run;
```

The following statements define the ARIMA AIRLINE model to generate a forecast that is combined with the forecast selected by TBEST. AICC weights are used for this combination. You must define a model selection list as the basis for running the combination, as described in more detail in the section "Forecast Model Selection Graph Operation" on page 633. The model selection list, named SELECT, defines a holdout of six samples and root mean squared error (RMSE) as the selection criterion.

```
proc hpfarimaspec rep=work.rep specname=airline;
   forecast symbol=y dif=(1,s) q=(1)(1)s noint;
   estimate method=ml converge=.0001 delta=.0001 maxiter=150;
run;

proc hpfselect rep=work.rep name=comb2 label="AICC(TBEST,T3)";
   combine method=aicc;
   spec tbest airline;
run;

proc hpfselect rep=work.rep name=select;
   select holdout=6 criterion=rmse;
   spec comb2;
run;

proc hpfengine data=sashelp.air
               rep=work.rep
               globalselection=select
               out=out1
               lead=24
               print=(all)
               plot=(components);
   id date interval=month;
   forecast air;
run;
```

**Output 21.2.1** Model Selection Results

**The HPFENGINE Procedure**

| Model Selection Criterion = RMSE | | | |
|---|---|---|---|
| **Model** | **Statistic** | **Selected** | **Label** |
| **COMB2** | 20.059523 | Yes | AICC(TBEST,T3) |

Output 21.2.2 and Output 21.2.3 show the parameter estimates the forecasts that contribute to the combination. Output 21.2.4 shows the parameter estimates for the combined forecast.

**Output 21.2.2** Parameter Estimates for T1 Model

| Parameter Estimates for T1 Model | | | | | |
|---|---|---|---|---|---|
| **Component** | **Parameter** | **Estimate** | **Standard Error** | **t Value** | **Approx Pr > \|t\|** |
| **AIR** | **Level Weight** | 0.30728 | 0.03153 | 9.74 | <.0001 |
| **AIR** | **Trend Weight** | 0.0010000 | 0.0030237 | 0.33 | 0.7413 |
| **AIR** | **Seasonal Weight** | 0.87493 | 0.07769 | 11.26 | <.0001 |

**Output 21.2.3** Parameter Estimates for AIRLINE Model

| Parameter Estimates for AIRLINE Model | | | | | |
|---|---|---|---|---|---|
| **Component** | **Parameter** | **Estimate** | **Standard Error** | **t Value** | **Approx Pr > \|t\|** |
| **AIR** | **MA1_1** | 0.30868 | 0.08433 | 3.66 | 0.0003 |
| **AIR** | **MA2_12** | 0.10744 | 0.10190 | 1.05 | 0.2917 |

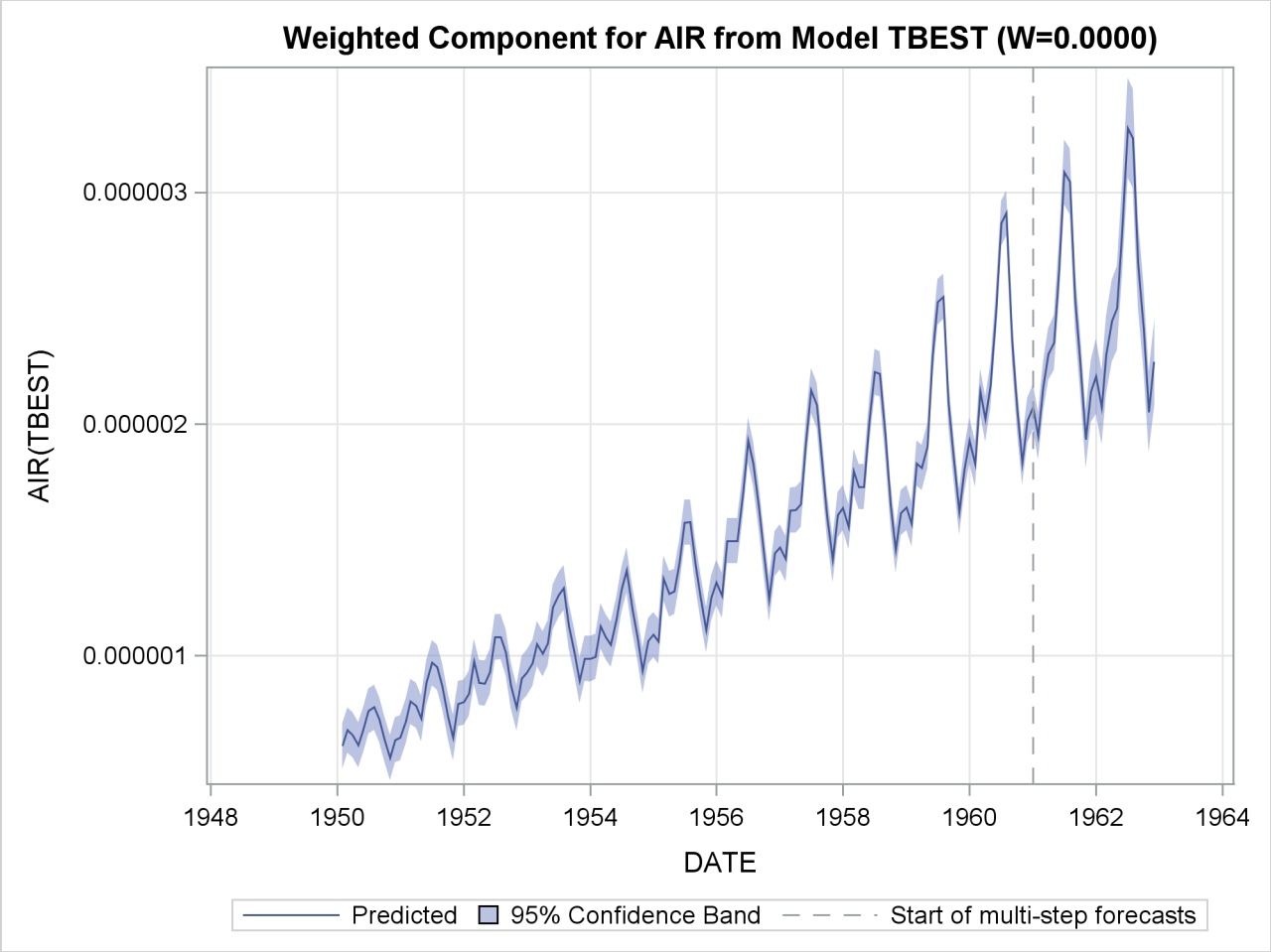
**Output 21.2.4** Parameter Estimates for Combined Forecast

| Parameter Estimates for COMB2 Model | | | | | |
|---|---|---|---|---|---|
| **Component** | **Parameter** | **Estimate** | **Standard Error** | **t Value** | **Approx Pr > \|t\|** |
| **TBEST** | **WEIGHT** | 4.65996E-9 | . | . | . |
| **AIRLINE** | **WEIGHT** | 1.00000 | . | . | . |

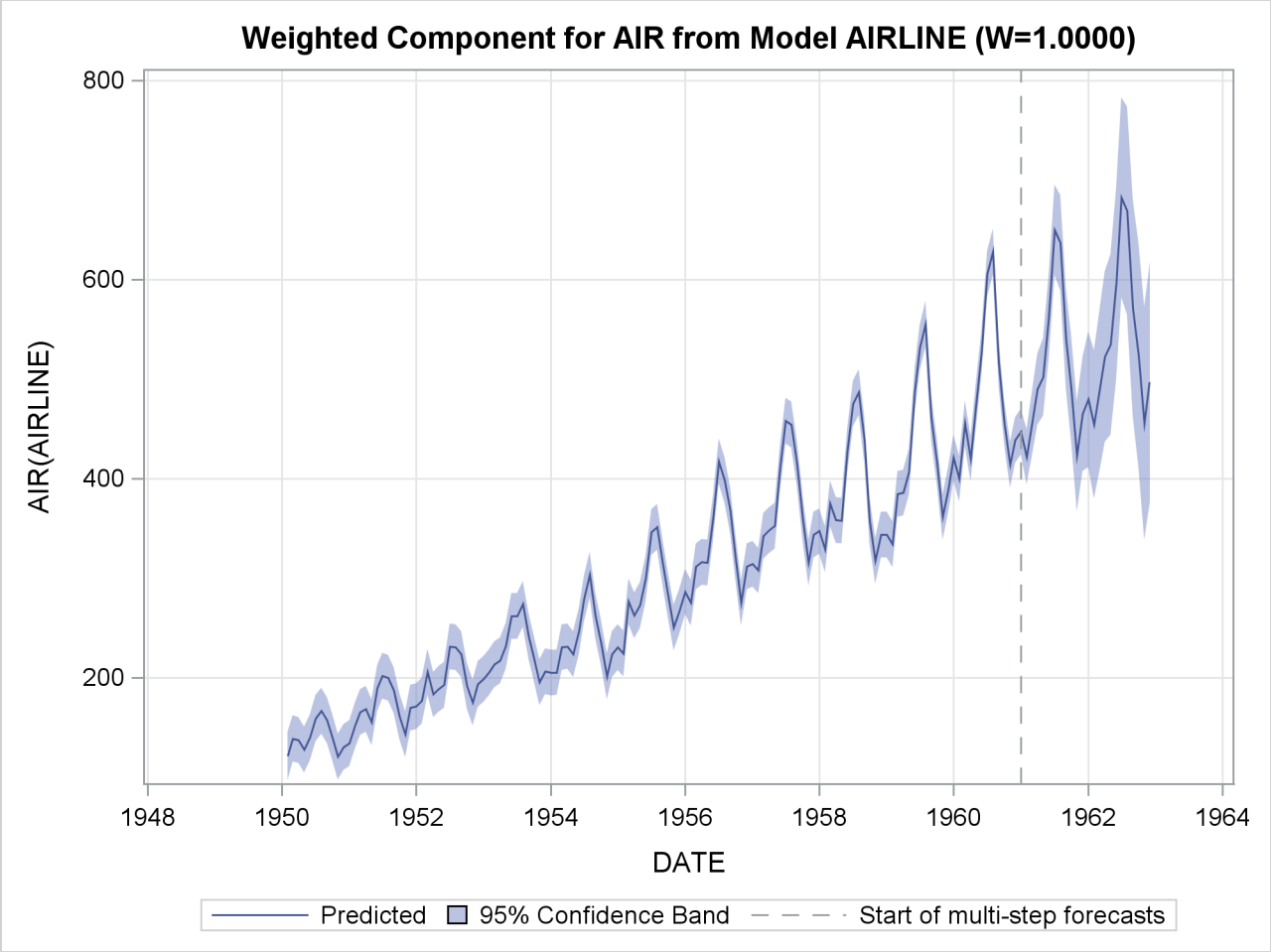
From the weights used in COMB2, it is obvious that the AICC weights are heavily slanted towards the ARIMA AIRLINE model. This is not surprising since the ARIMA AIRLINE model is well-tuned from its history to model this particular data set.

The weighted components are plotted separately followed by the stack of weighted forecast components as in the previous example. Pay careful attention to the Y-axis scale of the weighted component plot for the TBEST forecast in contrast to the Y-axis scale for the weighted component plot for the AIRLINE model forecast. The stacked component plot clearly reflects the dominance of the AIRLINE model forecast in the combined forecast.

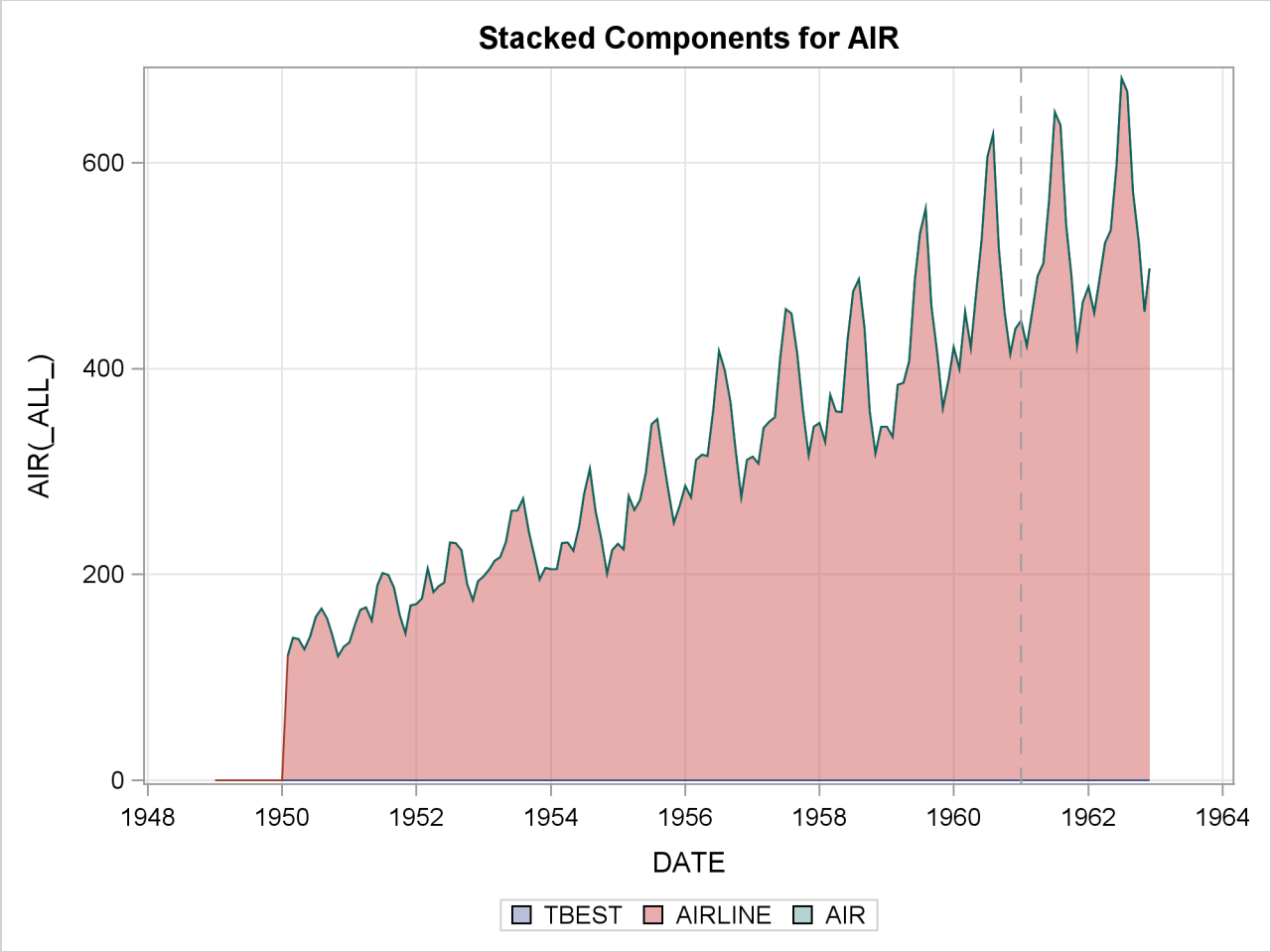
**Output 21.2.5** Combined Forecast Components



**Output 21.2.6** Combined Forecast Components



**Output 21.2.7** Combined Forecast Components

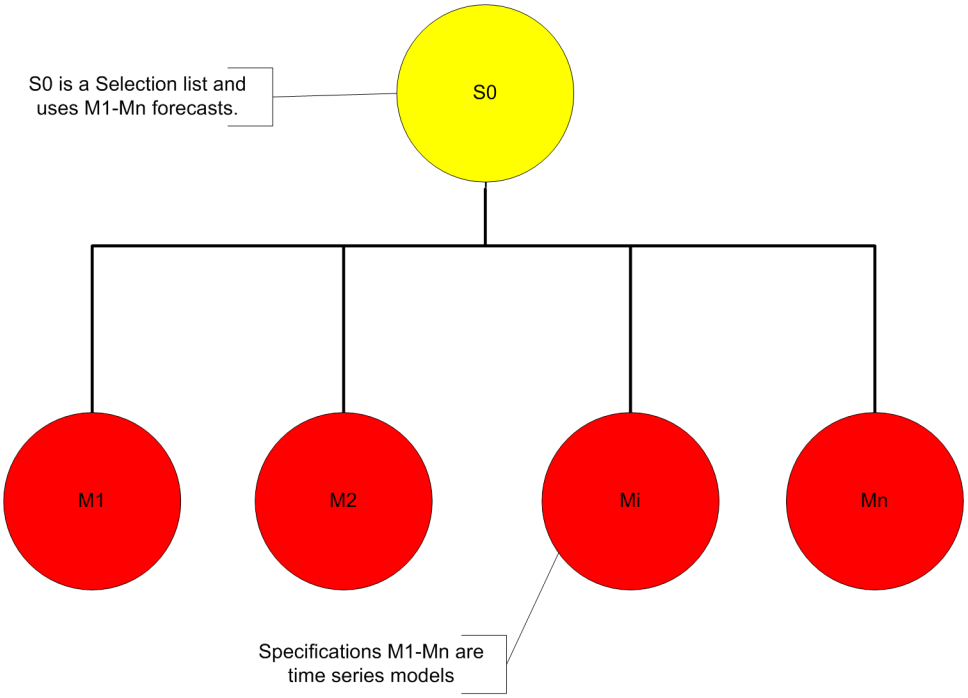


# Forecast Model Selection Graph Operation

This section provides some insight into the operation of the forecast model selection graph as it works in the context of the HPFENGINE procedure. This insight can help you understand the behavior of forecast model selection graphs with respect to output data sets and ODS-related output.

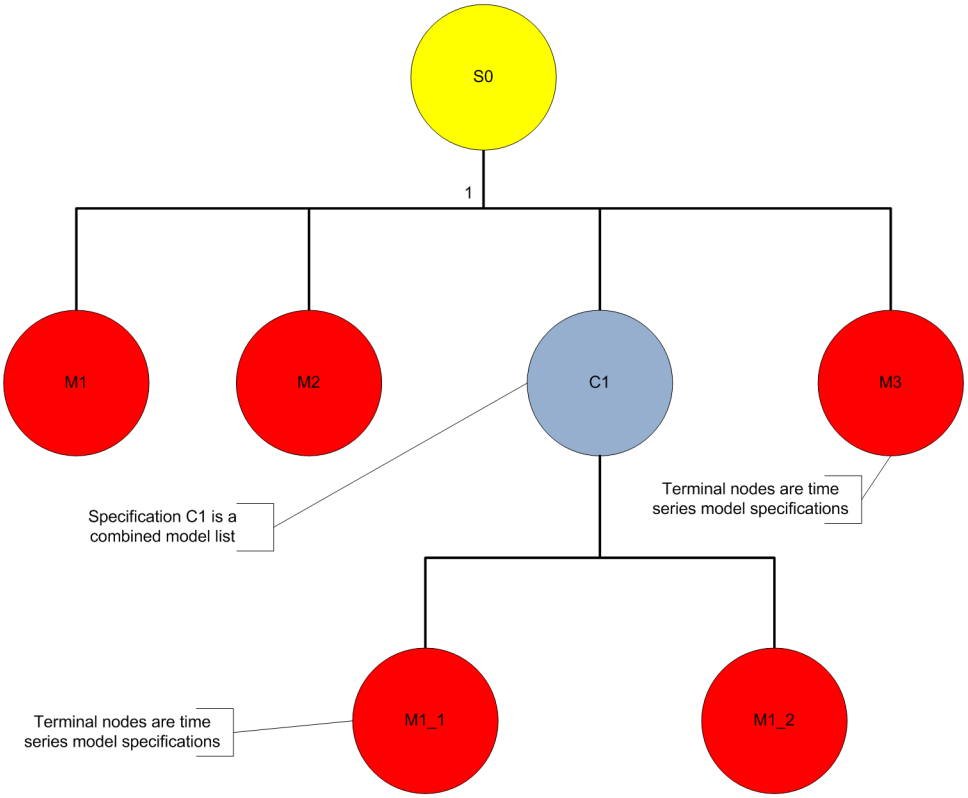
Every time series processed by PROC HPFENGINE during one of its runs uses some form of forecast model selection graph to produce its forecast. Previously these were limited to a one-level model selection list. The model selection list held the names of the time series models in its list. In the context of the forecast model selection graph, more complex forecasting scenarios are now possible. Figure 21.2.8 depicts the forecast model selection graph equivalent for the one-level model selection list.

**Output 21.2.8** Simple Model Selection List



Every forecast model selection graph starts with a single root node which must be a model selection list node. The forecast model selection graph generated from the root model selection list must contain at least one time series forecasting model to use in forecasting the time series realization for which it is used. Forecast model selection graphs that have no time series model specifications can produce no forecast by themselves. Obviously, only the last-chance forecasts in PROC HPFENGINE can handle the failed forecast from any such forecast model selection graph; without any time series models in the forecast model selection graph, there is nothing to produce any forecast for any list to either combine or select. Figure 21.1 depicts the forecast model selection graph for a two-level forecasting process.

**Figure 21.1** Simple Model Selection Graph




The root model selection list in the forecast model selection graph defines some properties that apply to all of the nodes in the forecast model selection graph. These properties include:

- SELECT statement HOLDOUT= option
- SELECT statement HOLDOUTPCT= option
- DIAGNOSE statement
- FORECASTOPTIONS statement

The stages of PROC HPFENGINE's execution, its so-called "tasks," perform as usual in the context of the more general forecast model selection graph with the natural extension from the one-level model selection list. When a TASK=SELECT invocation is used, the best forecast from the forecast model selection graph is selected. Then the best path through the forecast model selection graph is fitted and forecast to produce a final forecast for any time series realization where it is used. When a TASK=FIT, UPDATE, or FORECAST invocation is used, the saved model and parameter information from a previous OUTEST data set is used to restore the context needed to run the best path through the forecast model selection graph as directed by the specified TASK= option.

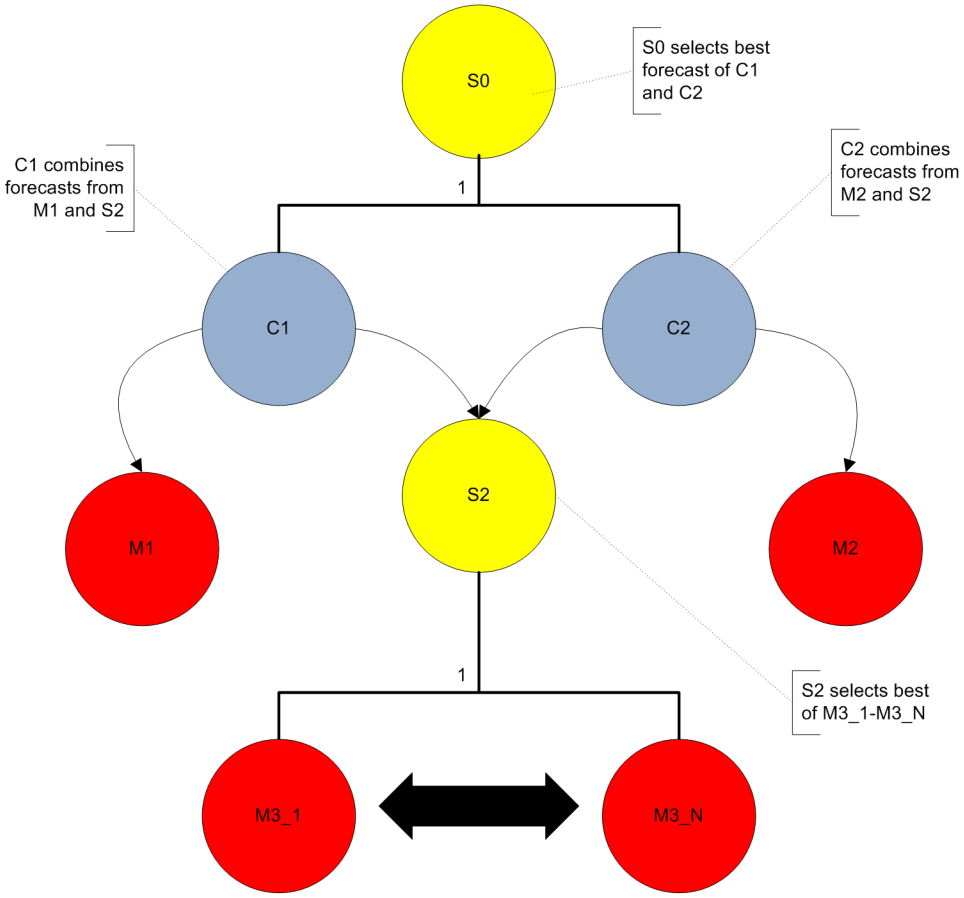
As noted from the examples, the specification name forms the basis for constructing the forecast model selection graph and connecting it together. Nodes in the forecast model selection graph generally fall into two distinct groups: lists and leaf nodes. Leaf nodes, as the name implies, are terminal in the forecast model selection graph. They represent instances of one of the time series model families: ESM, IDM, ARIMA, UCM, or EXM. List nodes, as the name implies, are lists of specifications, or more correctly, specification names. In general, the specification name that appears in a list can be the name of a leaf node or the name of another list. For obvious reasons, cycles are not permitted in the forecast model selection graph (the forecast model selection graph is a rooted-DAG).

Lists fall into two categories:

- the model selection list selects the best of its alternative forecasts according to a selection criterion that is evaluated over some specified region of the forecast. The model selection list's forecast is the forecast of the best of its list of contributors.
- the combined model list combines its contributing forecasts as a weighted average according to a specified method for generating the weights. The combined model list has its own forecast apart from its contributors.

Each node in the forecast model selection graph is viewed by any consumer of its results as the source of a time series forecast. Lists define context that is directly applied to the evaluation of time series models that belong to the list. Each list is uniquely identified in the context of the forecast model selection graph construction by its name, and the topology of the forecast model selection graph is determined by the arcs that connect a list to its specifications. The same time series model specifications can belong to different lists in the forecast model selection graph, and they are evaluated independently in the context of their respective lists. The results of a list can be consumed by many traversal paths through the forecast model selection graph, but the list's results are computed only once when first demanded and then consumed by all traversal paths that reference the list. Apart from the previous discussion of shared root node properties, a list node does not inherit properties accumulated along the traversal path that demands its results. So properties such as the maps are autonomously defined in the scope of each list to satisfy the needs of the time series models that are referenced from the list. Each list is endowed with those properties from its definition via PROC HPFSELECT and subjected overrides from the invocation of PROC HPFENGINE. The isolation of the computation of a list node's forecast from any traversal path context enables all consumers to share its forecast. Figure 21.2 depicts the forecast model selection graph of a forecasting process with multiple nodes that consume the results of a common list node.

**Figure 21.2** Complex Model Selection Graph




## OUTSTATSELECT Behavior

The OUTSTATSELECT data set receives a record for each forecast that is evaluated during a TASK=SELECT invocation of PROC HPFENGINE. As in the one-level model selection list case, this happens regardless of whether the forecast is used in the final forecast or not. Leaf nodes and combined model list's processed during the TASK=SELECT forecast model selection graph traversal add records to OUTSTATSELECT. Model selection list nodes in the forecast model selection graph traversal produce no records of their own. The following values are generated in the _SELECTED_ column of the OUTSTATSELECT records:

YES — to indicate that the _MODEL_ specification in the OUTSTATSELECT record is selected by the root model selection list in the forecast model selection graph

| | |
|---|---|
| USED | to indicate that the _MODEL_ specification in the OUTSTATSELECT record is used to produce the selected forecast in the forecast model selection graph |
| NO | to indicate that the _MODEL_ specification in the OUTSTATSELECT record is neither selected nor used to produced the selected forecast in the forecast model selection graph |

## OUTEST Behavior

The OUTEST data set receives a set of records for each model that is used during the final forecast traversal path through the forecast model selection graph. Parameter estimate records are added from the lowest level contributor nodes in the traversal path up to the highest level. Each node in the traversal can produce one or more OUTEST data set records.

## OUTSTAT Behavior

The OUTSTAT data set receives output for the selected forecast from the root model selection list in the forecast model selection graph. OUTSTAT always receives a record for the parameter estimation or fit region of the forecast. If a forecast performance region is specified via the BACK= option in the PROC HPFENGINE statement, a record is also generated for the selected forecast over the performance region.

## OUTMODELINFO Behavior

The OUTMODELINFO data set receives output for the selected model from the root model selection list in the forecast model selection graph. When the selected model is a combined model, the information recorded in the OUTMODELINFO data set reflects characteristics of the models used in the combination. See the section "OUTMODELINFO= Data Set" on page 224 for more details.

## OUTSUM Behavior

The OUTSUM data set receives output for the selected forecast from the root model selection list in the forecast model selection graph. When the selected model is a combined model, the forecasts contained in the OUTSUM data set are the combined forecasts.

## OUTCOMPONENT Behavior

The OUTCOMPONENT data set receives output for the selected forecast from the root model selection list in the forecast model selection graph. When the selected model is a combined model, the components are the weighted forecasts used to generate the combined model's forecast.

## OUTFOR Behavior

The OUTFOR data set receives the forecast that results from the final forecast traversal path through the forecast model selection graph. One collection of OUTFOR records is produced for the requested time span for the time series being forecast.

## OUT Behavior

The OUT data set receives the forecast that results from the final forecast traversal path through the forecast model selection graph. One collection of OUTFOR records is produced for the time span for the time series being forecast.

## ODS Print Behavior

Requested ODS tables are printed for the selected forecast from the root model selection list in the forecast model selection graph with the following exceptions:

- The ODS ModelSelection table displays information for each forecast considered in the root model selection list.
- The ODS Estimates table is repeated as needed to display parameter estimates for the models used during the final forecast traversal path through the forecast model selection graph. Tables are printed from the lowest level contributor nodes up to the highest level contributor nodes.

## ODS Plot Behavior

Requested ODS plots are generated for the selected forecast from the root model selection list in the forecast model selection graph.

# Chapter 22
# Using Forecasting Model Score Files and DATA Step Functions

**Contents**


Overview . . . . . **639**
HPFSCSIG Function . . . . . **639**
Example 22.1: HPFSGSIG Use . . . . . 640
HPFSCSUB Function . . . . . **641**
Example 22.2: HPFSCSUB Use . . . . . 642


## Overview

The HPFENGINE procedure can generate forecast score files. The HPFSCSIG and HPFSCSUB functions are used in conjunction with these score files to produce forecasts.

## HPFSCSIG Function

The HPFSCSIG function creates a text string that represents a template for the appropriate usage of the HPFSCSUB function given a forecast scoring file.

### Syntax

**HPFSCSIG** *(scoreFile, horizon, outType)* **;**

*scoreFile* is a SAS file reference that contains the forecast scoring information.

*horizon* is the forecast horizon or lead. This must be a positive integer value.

*outType* is one of the following strings: PREDICT, STDERR, LOWER, or UPPER. *OutType* determines whether the score function computes and returns the forecast value, standard error, lower confidence limit, or upper confidence limit, respectively. By default a 95% confidence interval($\alpha = 0.05$) is assumed for computing critical values for upper and lower confidence limits. You can change this by specifying the alternate syntax Lxx for LOWER or Uxx for UPPER where xx is the confidence interval percentage desired($\alpha = (100 - xx)/100$). The legal range for xx is $0 < xx \leq 99$.

### Details

The syntax of the forecast scoring function is variable and depends on the horizon and certain details found in the score file. HPFSCSIG aids the user in constructing subroutine calls with the correct syntax.

## Example 22.1: HPFSGSIG Use

Consider the following case of an ARIMAX model with three inputs. Two of them are predefined trend curves, and the third is specified as controllable in the call to PROC HPFENGINE. A score file is produced, and HPFSCSIG called to provide a template for the call of HPFSCSUB.

```
   data air;
      set sashelp.air;
      controlinput = log(air);
   run;

proc hpfarimaspec modelrepository=work.repo
                  specname=ar;
   dependent symbol=Y q=1 dif=12;
   input predefined=LINEAR;
   input symbol=controlinput;
   input predefined=INVERSE;
run;

proc hpfselect modelrepository=work.repo
               selectname=select;
   spec ar;
run;

proc hpfengine data=air
               print=(select estimates)
               modelrepository=work.repo
               globalselection=select
               outest=engineoutest
               scorerepository=work.scores;
   id date interval=month;
   forecast air;
   controllable controlinput / extend=avg;
   score;
run;

filename score catalog "work.scores.scor0.xml";

data a;
   sig = hpfscsig( 'score', 3, 'predict' );
   put sig=;
run;

proc print data=a;
run;
```

The HPFSCSIG function call produces the result shown in Figure 22.1.1.

**Output 22.1.1** Result of the HPFSCSIG Function Call

| Obs | sig |
|---|---|
| 1 | HPFSCSUB('XML',3,'controlinput',?,?,?,'PREDICT',!,!,!) |

Provide the pre-assigned SAS fileref in place of the XML, replace the question marks (?) with the desired inputs, and the output is written to variables placed where there are exclamation marks (!).

# HPFSCSUB Function

The HPFSCSUB function returns the forecasts, standard errors, or confidence limits given a forecast scoring file and future values of all controllable inputs.

## Syntax

**HPFSCSUB** *( scoreFile, horizon,* $X_1, X_{1,1}, \ldots, X_{1,horizon}, \ldots, X_j, X_{j,1}, \ldots, X_{j,horizon}, \ldots,$ *outType,* $O_1, \ldots, O_{horizon},$ *<,name-value-pair, ... >)* **;**

*scoreFile* is a SAS file reference that contains the forecast scoring information.

*horizon* is the forecast horizon or lead. This must be a positive integer value.

$X_j$ specifies the name of a controllable variable needed to produce forecasts. The next *horizon* arguments are the future inputs for controllable variable $X_j$.

$X_{j,k}$ is the value for the controllable variable $X_j$ at horizon $k$. For each $X_j$ you must specify numeric arguments $X_{j,1}$ through $X_{j,horizon}$.

*outType* is one of the following strings: PREDICT, STDERR, LOWER, or UPPER. *outType* determines whether the score function computes and returns the forecast value, standard error, lower confidence limit, or upper confidence limit, respectively. By default a 95% confidence interval($\alpha = 0.05$) is assumed for computing critical values for upper and lower confidence limits. You can change this by specifying the alternate syntax Lxx for LOWER or Uxx for UPPER where xx is the confidence interval percentage desired($\alpha = (100 - xx)/100$). The legal range for xx is $0 < xx \leq 99$.

$O_k$ is the subroutine output at horizon $k$. You must specify arguments $O_1$ through $O_{horizon}$ as numeric variables.

You can specify optional name-value pairs (*name-value-pair* arguments) the HPFSCSUB call. These optional arguments allow you to specify settings that alter the behavior of the HPFSCSUB call. These *name-value-pair* arguments must be specified in groups of two. The first argument in each pair must be a character string (literal or character variable) that identifies the name of a property you wish to set. Property names begin and end with a "." character. The type of the second argument depends on the on the property name that you specify, as do the values that are accepted for it.

You can specify the following property name values:

**.HORIZONSTART.** specifies the time stamp value that is used to determine the starting time index that corresponds to $X_{j,1}$ and $O_1$. The value argument must be numeric. It is interpreted as a SAS DATE or DATETIME value, and it must be consistent with the units for the time ID variable that was specified in the PROC HPFENGINE step that created the score file. The time stamp value that you specify must fall into a time period in the region of multi-step forecasts that are produced from the PROC HPFENGINE step that created the score file; otherwise, the HPFSCSUB call generates an error.

### Details

The HPFSCSUB function returns the forecasts, standard errors, or confidence limits given a forecast scoring file and future values of all controllable inputs. Because the syntax is variable and depends on the input score file, HPFSCSIG is normally used to determine the layout of the subroutine call.

The score might have been computed using ARIMAX, UCM, or another model. HPFSCSUB automatically detects this and computes the requested return values appropriately.

## Example 22.2: HPFSCSUB Use

This example demonstrates how to use the HPFSCSUB function for a simple case. It uses the score file from PROC HPFENGINE to compute a forecast. The score file is stored in the scor0 entry within the catalog work.score. Note that in the ARIMAX model specification there are two transfer functions with PREDEFINED= behavior and one based on the symbol Controlinput. Note that even though the model includes three inputs, it is only necessary to designate the variable Controlinput as controllable in the PROC HPFENGINE invocation. Therefore, only future values of the variable Controlinput are required to generate forecasts by using the score file. Future values of the other inputs are determined automatically by their respective predefined behavior in the model specification.

In the call to PROC HPFENGINE, the controllable input was extended with the mean of the Controlinput series. Therefore the mean is used as input to the forecast score function so that a valid comparison can be made between the forecast results from PROC HPFENGINE and HPFSCSUB.

```
proc hpfarimaspec modelrepository=work.repo
                  specname=ar;
   dependent symbol=Y q=1 dif=12;
   input predefined=LINEAR;
   input symbol=controlinput;
   input predefined=INVERSE;
run;

proc hpfselect modelrepository=work.repo
               selectname=select;
   spec ar;
run;

* generate score;
proc hpfengine data=air
               modelrepository=work.repo
               out=engineout
               globalselection=select
```

```
                scorerepository=work.scores;
   id date interval=month;
   forecast air;
   controllable controlinput / extend=avg;
   score;
run;

filename score catalog "work.scores.scor0.xml";

proc means data=air mean noprint;
   var controlinput;
   output out=controlmean mean=mean;
run;

data _null_;
   set controlmean;
   call symput( "mean", mean );
run;


data forecasts;
   array p{3} p1-p3;
   keep date forecast;

   format date monyy.;
   date = '01jan1961'd;

   call HPFSCSUB( 'score', 3, 'CONTROLINPUT', &mean, &mean, &mean,
                  'PREDICT', p1, p2, p3 );
   do i=1 to 3;
     forecast = p{i};
     output;
     date = intnx('month', date, 1);
   end;
run;

data compare;
   merge engineout forecasts;
   by date;
run;

proc print data=compare(where=(forecast ne .)) noobs;
run;
```

Figure 22.2.1 shows the output generated by the preceding statements.

**Output 22.2.1** Output From Example

| DATE | AIR | forecast |
|---|---|---|
| JAN1961 | 416.408 | 416.408 |
| FEB1961 | 391.715 | 391.715 |
| MAR1961 | 419.312 | 419.312 |

Continuing the previous example, you can also specify the confidence interval percentage points for HPFSCSUB as the following sample code demonstrates. Observe that you can specify different percentage points for the upper and lower critical values to obtain asymmetric limit bands.

You should note that even though you specify the 'Uxx' or 'Lxx' selectors independently of each other, the critical Z-value computed is that for a symmetric two-sided confidence interval. In order to affect a true 1-sided confidence limit, you can make the appropriate adjustment in the 2-sided percentage point you provide in the selector string or perform a custom calculation in DATA step code using the desired percentage point value in the PROBIT function in conjunction with the HPSCSUB function returns for 'PREDICT' and 'STDERR'. The following DATA step demonstrates using this technique to directly compute the corresponding upper and lower confidence interval limits as a comparison to the values returned from HPFSCSUB.

```
data myoutfor;
   array p{3} p1-p3;
   array se{3} se1-se3;
   array ucl{3} ucl1-ucl3;
   array lcl{3} lcl1-lcl3;
   keep date scucl sclcl myucl mylcl;

   format date monyy.;
   date = '01jan1961'd;

   call HPFSCSUB( 'score', 3, 'CONTROLINPUT', &mean, &mean, &mean,
                  'PREDICT', p1, p2, p3 );
   call HPFSCSUB( 'score', 3, 'CONTROLINPUT', &mean, &mean, &mean,
                  'STDERR', se1, se2, se3 );
   call HPFSCSUB( 'score', 3, 'CONTROLINPUT', &mean, &mean, &mean,
                  'L99', lcl1, lcl2, lcl3 );
   call HPFSCSUB( 'score', 3, 'CONTROLINPUT', &mean, &mean, &mean,
                  'U90', ucl1, ucl2, ucl3 );
   do i=1 to 3;
       scucl = ucl{i};
       sclcl = lcl{i};
       myucl=p{i} + probit(1-0.1/2) * se{i};
       mylcl=p{i} - probit(1-0.01/2) * se{i};
       output;
       date = intnx('month', date, 1);
   end;
run;

proc print data=myoutfor noobs;
run;
```

Figure 22.2.2 shows the output generated by the preceding statements.

**Output 22.2.2** Output From Example

| date | scucl | sclcl | myucl | mylcl |
|---|---|---|---|---|
| JAN61 | 435.744 | 386.128 | 435.744 | 386.128 |
| FEB61 | 413.103 | 358.221 | 413.103 | 358.221 |
| MAR61 | 440.700 | 385.817 | 440.700 | 385.817 |

The previous example can be extended to demonstrate the use of optional property-value pairs in the HPFSCSUB call. Suppose that you only want to control the forecast response for the last three periods of the 12 month horizon in the score file that was generated by the PROC HPFENGINE step. You can use the *.HORIZONSTART.* property to set the start of the forecast horizon to 01OCT1961 with a *horizon* argument of 3. HPFSCSUB then uses the values for the controllable variables prior to 01OCT1961 from the PROC HPFENGINE step so that you only specify future values for the periods 01OCT1961 through 01DEC1961.

```
data last3;
   array p{3} p1-p3;
   array se{3} se1-se3;
   array ucl{3} ucl1-ucl3;
   array lcl{3} lcl1-lcl3;
   keep date scucl sclcl myucl mylcl;

   format date monyy.;
   date = '01oct1961'd;

   call HPFSCSUB( 'score', 3, 'CONTROLINPUT', &mean, &mean, &mean,
                  'PREDICT', p1, p2, p3, '.horizonstart.', date);
   call HPFSCSUB( 'score', 3, 'CONTROLINPUT', &mean, &mean, &mean,
                  'STDERR', se1, se2, se3, '.horizonstart.', date);
   call HPFSCSUB( 'score', 3, 'CONTROLINPUT', &mean, &mean, &mean,
                  'L99', lcl1, lcl2, lcl3, '.horizonstart.', date);
   call HPFSCSUB( 'score', 3, 'CONTROLINPUT', &mean, &mean, &mean,
                  'U90', ucl1, ucl2, ucl3, '.horizonstart.', date);
   do i=1 to 3;
       scucl = ucl{i};
       sclcl = lcl{i};
       myucl=p{i} + probit(1-0.1/2) * se{i};
       mylcl=p{i} - probit(1-0.01/2) * se{i};
       output;
       date = intnx('month', date, 1);
   end;
run;

proc print data=last3 noobs;
run;
```

Figure 22.2.3 shows the output generated by the preceding statements.

**Output 22.2.3** Output From Example

| date | scucl | sclcl | myucl | mylcl |
|---|---|---|---|---|
| OCT61 | 479.878 | 424.995 | 479.878 | 424.995 |
| NOV61 | 408.475 | 353.592 | 408.475 | 353.592 |
| DEC61 | 450.072 | 395.189 | 450.072 | 395.189 |

The use of the *.HORIZONSTART.* property is particularly useful in conjunction with the HORIZONSTART= option from the PROC HPFENGINE ID statement. The *.HORIZONSTART.* property allows you to set the time index where you expect to logically begin the specification of future values for the controllable variables without regard to the exact point in time where multi-step forecasting begins (multi-step forecasting can begin prior to the HORIZONSTART= time if the dependent series ends in missing values). You should refer to the Examples section for PROC HPFENGINE for additional information about using HPFSCSUB. There you will find a more substantial example demonstrating the integration of optimization and forecasting.

# Chapter 23
# Using User-Defined Models

**Contents**


Introduction . . . . . 647
Defining and Using a SAS Language Function or Subroutine . . . . . 648
Defining and Using a C Language External Function . . . . . 651
Input Time Series Keywords . . . . . 655
Returned Forecast Component Keywords . . . . . 656


## Introduction

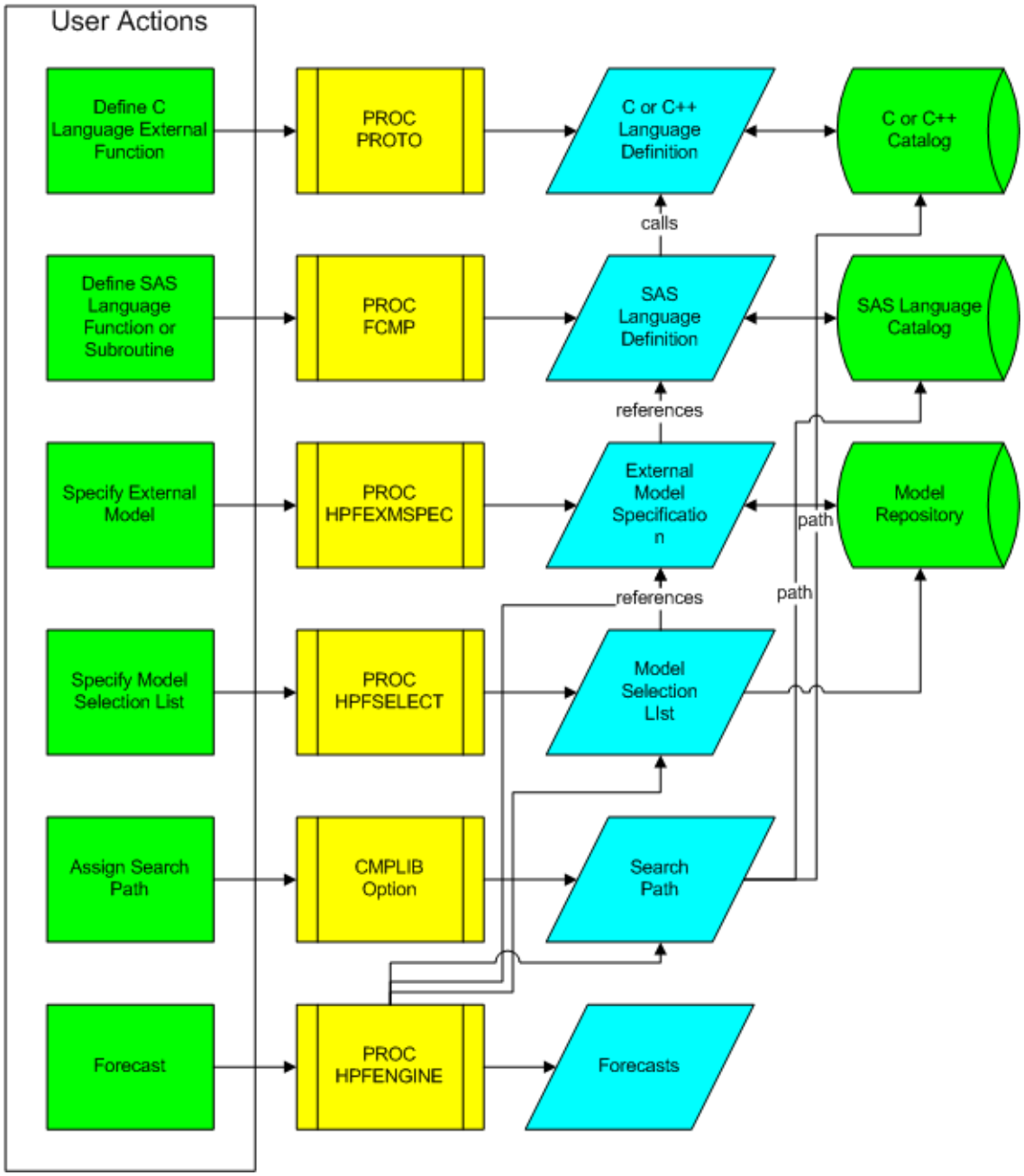
You can use user-defined forecasting models in the SAS Forecast Server Procedures software. The forecasts that are produced by these models are considered external forecasts because they are forecasts that originate from an external source.

A user-defined forecasting model can be written in the SAS language or in the C language by using the FCMP procedure or the PROTO procedure, respectively. The HPFENGINE procedure cannot use C language routines directly. The procedure can use only SAS language routines that might or might not call C language routines. Creating user-defined routines is more completely described in the FCMP procedure and the PROTO procedure documentation. For more information about the FCMP and PROTO procedures, see the *Base SAS Procedures Guide*.

The SAS language provides integrated memory management and exception handling such as operations on missing values. The C language provides flexibility and enables the integration of existing C language libraries. However, proper memory management and exception handling are solely the responsibility of the user. Additionally, the support for standard C libraries is restricted. If given a choice, it is highly recommended that user-defined functions and subroutines and functions be written in the SAS language using the FCMP procedure.

In order to use a SAS language function or routine, an external model specification must also be specified. In order to use a C or C++ language external function, it must be called by a SAS language function or subroutine, and an external model specification must also be specified. External model specifications are specified by the HPFEXMSPEC procedure. The following diagram describes an example of the ways user-defined forecasting models can be defined, specified, and used to forecast.

**Figure 23.1** User-Defined Forecasting Models




The SAS language function or routine can call other SAS language functions or routines as long as the search path for these functions and routines are provided.

## Defining and Using a SAS Language Function or Subroutine

The FCMP procedure provides a simple interface for compiling functions and subroutines for use by the SAS Forecast Server Procedures software. The FCMP procedure accepts a slight variant of the SAS DATA

step language. Most features of the SAS programming language can be used in functions and subroutines processed by the FCMP procedure. For more information about the FCMP procedure, see the *Base SAS Procedures Guide*.

For example, the following SAS statement creates a user-defined forecasting model for a simple linear trend line called LINEARTREND that is written in the SAS language and stores this subroutine in the catalog WORK.HPFUSER. The user-defined forecasting model has the following subroutine signature:

```
subroutine <subroutine-name> ( <array-name>[*], <array-name>[*],
                               <array-name>[*], <array-name>[*],
                               <array-name>[*] );
```

where the first array, ACTUAL, contains the time series to be forecast, the second array, PREDICT, contains the returned predictions, the third array, STD, contains the returned prediction standard errors, the fourth array, LOWER, contains the returned lower confidence limits, and the fifth array, UPPER, contains the returned upper confidence limits.

```
proc fcmp outlib=work.hpfuser.funcs;
   subroutine lineartrend( actual[*], predict[*], std[*],
                           lower[*], upper[*] );
      outargs predict, std, lower, upper;
      nobs = dim(actual);
      n      = 0;
      sumx   = 0;
      sumx2  = 0;
      sumxy  = 0;
      sumy   = 0;
      sumy2  = 0;

      do t = 1 to nobs;
         value = actual[t];
         if nmiss(value) = 0 then do;
            n     = n     + 1;
            sumx  = sumx  + t;
            sumx2 = sumx2 + t*t;
            sumxy = sumxy + t*value;
            sumy  = sumy  + value;
            sumy2 = sumy2 + value*value;
         end;
      end;

      det = (n*sumx2 - sumx*sumx);
      constant = (sumx2 * sumy - sumx * sumxy) / det;
      slope    = (-sumx  * sumy + n    * sumxy) / det;
      length = DIM(predict);
      do t = 1 to length;
         predict[t] = constant + slope*t;
      end;

      sume2 = 0;
      do t = 1 to nobs;
         value = actual[t];
```

```
         if nmiss(value) = 0 then do;
            error = value - predict[t];
            sume2  = sume2 + error*error;
         end;
      end;

      stderr   = sqrt(sume2 / (n-2));
      zvalue   = probit(1-0.05/2);
      width    = zvalue*stderr;
      length = DIM(predict);
      do t = 1 to length;
         std[t]     = stderr;
         lower[t]   = predict[t] - width;
         upper[t]   = predict[t] + width;
      end;

   endsub;
quit;
```

In order to use a user-defined forecasting model, an external forecast model specification must be specified using the HPFEXMSPEC procedure. For example, the following SAS statements create an external model specification called LINEARTREND and store this model specification in the catalog WORK.MYREPOSITORY. Since the user-defined forecasting model uses two parameters, CONSTANT and SLOPE, the NPARMS=2 option is specified in the EXM statement. The number specified in this option is used for computing statistics of fit (for example, RMSE, AIC, and BIC).

```
proc hpfexmspec modelrepository=work.myrepository
                specname=lineartrend
                speclabel="User defined linear trend";
   exm nparms=2;
run;
```

The HPFSELECT procedure can be used to create a model selection list that contains an external model specification as a possible candidate model. For example, the following SAS statements create a model selection list called MYSELECT and store this model selection list in the catalog WORK.MYREPOSITORY.

```
proc hpfselect modelrepository=work.myrepository
               selectname=myselect
               selectlabel="My Select List";
   spec lineartrend /
      exmfunc(
      "lineartrend(_actual_ _predict_ _stderr_ _lower_ _upper_)"
      );
run;
```

To use the user-defined forecasting model defined by the FCMP procedure in the HPFENGINE procedure, the CMPLIB option must list the catalogs that contain the SAS language functions and routines. For more information about the FCMP procedure, see the *Base SAS Procedures Guide*.

For example, the following SAS statement specifies the SAS catalogs that are required to use the user-defined forecasting model.

```
options cmplib= work.hpfuser;
```

At this point:

- A SAS language subroutine has been defined, LINEARTREND, and stored in the SAS catalog, WORK.HPFUSER.
- An external model specification, LINEARTREND, has been stored in the model repository, WORK.MYREPOSITORY.
- A model selection list, MYSELECT, has been stored in the model repository, WORK.MYREPOSITORY.
- The search path for the SAS language functions and subroutines has been set to WORK.HPFUSER.

The HPFENGINE procedure can now use the user-defined forecasting routine.

For example, the following SAS statements forecast the monthly time series contained in the SASHELP.AIR data set. This data set contains two variables DATE and AIR. The MODELREPOSITORY= WORK.MYREPOSITORY option of the PROC HPFENGINE statement specifies the model repository, and the GLOBALSELECTION=MYSELECT options specifies the model selection list.

```
proc hpfengine data=sashelp.air
               out=cmpout
               outfor=cmpfor
               outstat=cmpstat
               modelrepository=myrepository
               globalselection=myselect;
   id date interval=month;
   forecast air;
run;
```

The OUT= data set contains the original data extrapolated by the simple linear trend model (values returned in the _PREDICT_ array), and the OUTFOR= data set contains the forecasts (values returned in the _PREDICT_, _STDERR_, _LOWER_, and _UPPER_ array) and the prediction errors. The OUTSTAT= data set contains the statistics of fit based on the prediction errors and the NPARMS=2 option of the external model specification.

## Defining and Using a C Language External Function

The PROTO procedure enables you to register, in batch, external functions written in the C or C++ programming languages for use in SAS. In order to use an external function, it must be called from a SAS language function or subroutine. For more information about the PROTO procedure, see the *Base SAS Procedures Guide*.

For example, the following SAS statements create a user-defined forecasting model for a simple linear trend line called LINEARTREND_C that is written in the C language and store this external function in the catalog WORK.CHPFUSER.

```
proc proto package=work.chpfuser.cfuncs;
   double lineartrend_c( double   zvalue,
                         double * actual,
                         int      actualLength,
                         double * predict,
                         double * std,
                         double * lower,
                         double * upper,
                         int      predictLength );
   externc lineartrend_c;
      double lineartrend_c( double   zvalue,
                            double * actual,
                            int      actualLength,
                            double * predict,
                            double * std,
                            double * lower,
                            double * upper,
                            int      predictLength ) {

      int      t, n;
      double   value, sumxy, sumx, sumx2, sumy, sumy2;
      double   det, constant, slope, stderr, width;
      double   error, sume2;
      n      = 0;
      sumx   = 0;
      sumx2  = 0.;
      sumxy  = 0.;
      sumy   = 0.;
      sumy2  = 0.;
      for ( t = 0; t < actualLength; t ++ ) {
         value = actual[t];
         n     = n     + 1;
         sumx  = sumx  + t;
         sumx2 = sumx2 + t*t;
         sumxy = sumxy + t*value;
         sumy  = sumy  + value;
         sumy2 = sumy2 + value*value;
      }

      det = (n*sumx2 - sumx*sumx);
      constant = (sumx2 * sumy - sumx * sumxy) / det;
      slope    = (-sumx  * sumy + n    * sumxy) / det;
      for ( t = 0; t < predictLength; t ++ ) {
         predict[t] = constant + slope*t;
      }

      sume2 = 0;
      for ( t = 0; t < actualLength; t ++ ) {
```

```
         value = actual[t];
         error = value - predict[t];
         sume2  = sume2 + error*error;
      }

      stderr   = sqrt(sume2 / (n-2.));
      width    = zvalue*stderr;
      for ( t = 0; t < predictLength; t ++ ) {
         std[t]     = stderr;
         lower[t]   = predict[t] - width;
         upper[t]   = predict[t] + width;
      }
   return(0);
   }
externcend;
run;
```

The following SAS statements create a user-defined forecasting model for a simple linear trend called LINEARTREND and store this subroutine in the catalog WORK.HPFUSER. The catalog WORK.CHPFUSER contains functions or subroutines that are used in LINEARTREND. The user-defined forecasting model has the following subroutine signature:

```
SUBROUTINE <SUBROUTINE-NAME> ( <ARRAY-NAME>[*], <ARRAY-NAME>[*],
                               <ARRAY-NAME>[*], <ARRAY-NAME>[*],
                               <ARRAY-NAME>[*] );
```

where the first array, ACTUAL, contains the time series to be forecast, the second array, PREDICT, contains the return predictions, the third array, STD, contains the returned prediction standard errors, the fourth array, LOWER, contains the return lower confidence limits, and the fifth array, UPPER, contains the return upper confidence limits. The LINEARTREND subroutine calls the external function LINEARTREND_C. The DIM function returns the length of the array. For example, the DIM(ACTUAL) function returns the length of the time series array; and the DIM(PREDICT) returns the length of the prediction array. DIM(PREDICT) DIM(ACTUAL) represents the forecast horizon or lead.

```
proc fcmp outlib=work.hpfuser.funcs
          inlib=work.chpfuser;
   subroutine lineartrend( actual[*], predict[*],
                           std[*], lower[*], upper[*] );
      outargs actual, predict, std, lower, upper;
      zvalue = probit(1-0.05/2);
      ret = lineartrend_c( zvalue, actual, DIM(actual),
                           predict, std, lower, upper, DIM(predict));
   endsub;
quit;
```

In order to use a user-defined forecasting model, an external forecast model specification must be specified using the HPFEXMSPEC procedure. For example, the following SAS statements create an external model specification called LINEARTREND and store this model specification in the catalog WORK.MYREPOSITORY. Since the user-defined forecasting model uses two parameters, CONSTANT and SLOPE, the NPARMS=2 option is specified in the EXM statement. The number specified in this option is used in computing statistics of fit.

```
proc hpfexmspec modelrepository=work.myrepository
                specname=lineartrend
                speclabel="User defined linear trend";
   exm nparms=2;
run;
```

The HPFSELECT procedure can be used to create a model selection list that contains an external model specification as a possible candidate model. For example, the following SAS statements create a model selection list called MYSELECT and store this model selection list in the catalog WORK.MYREPOSITORY. The keyword _PREDICT_ indicates the returned predictions, the keyword _STDERR_ indicates the returned prediction standard errors, the keyword _LOWER_ indicates the returned lower confidence limits, and the keyword _UPPER_ indicates the returned upper confidence limits.

```
proc hpfselect modelrepository=work.myrepository
               selectname=myselect
               selectlabel="My Select List";
   spec lineartrend /
      exmfunc(
      "lineartrend(_actual_ _predict_ _stderr_ _lower_ _upper_)"
      );
run;
```

To use the user-defined forecasting model defined by the FCMP procedure in the HPFENGINE procedure, the CMPLIB option must list the catalogs that contains the SAS language functions and routines and C language external functions. For more information about the FCMP procedure, see the *Base SAS Procedures Guide*.

For example, the following SAS statement specifies the SAS catalogs that are required to use a user-defined forecasting model.

```
options cmplib=( work.hpfuser work.chpfuser);
```

At this point:

- A C language external function has been defined, LINEARTREND_C, and stored in the SAS catalog, WORK.CHPFUSER.
- A SAS language subroutine has been defined, LINEARTREND, which calls the external function, LINEARTREND_C, and stored in the SAS catalog, WORK.HPFUSER.
- An external model specification, LINEARTREND, has been stored in the model repository, WORK.MYREPOSITORY.
- A model selection list, MYSELECT, has been stored in the model repository, WORK.MYREPOSITORY.
- The search path for the SAS language functions and subroutines has been set to WORK.HPFUSER and WORK.CHPFUSER.

The HPFENGINE procedure can now use the user-defined forecasting routine.

For example, the following SAS statements forecast the monthly time series contained in the SASHELP.AIR data set. This data set contains two variables DATE and AIR. The MODELREPOSITORY= WORK.MYREPOSITORY option of the PROC HPFENGINE statement specifies the model repository, and the GLOBALSELECTION=MYSELECT options specifies the model selection list.

```
proc hpfengine data=sashelp.air
               out=proout
               outfor=profor
               outstat=prostat
               modelrepository=myrepository
               globalselection=myselect;
   id date interval=month;
   forecast air;
run;
```

The OUT= data set contains the original data extrapolated by the simple linear trend model (values returned in the _PREDICT_ array), and the OUTFOR= data set contains the forecasts (values returned in the _PREDICT_, _STDERR_, _LOWER_, and _UPPER_ array) and the prediction errors. The OUTSTAT= data set contains the statistics of fit based on the prediction errors and the NPARMS=2 option of the external model specification.

Output 23.2 shows the results of a PROC COMPARE run that compares the forecasts from the OUTFOR= data sets from these two implementations of the same algorithm.

**Figure 23.2** Comparison of OUTFOR Data Sets

```
                    Observation Summary

                Observation      Base  Compare

                First Obs           1        1
                Last  Obs         156      156

Number of Observations in Common: 156.
Total Number of Observations Read from WORK.CMPFOR: 156.
Total Number of Observations Read from WORK.PROFOR: 156.

Number of Observations with Some Compared Variables Unequal: 0.
Number of Observations with All Compared Variables Equal: 156.


                  Values Comparison Summary

Number of Variables Compared with All Observations Equal: 8.
Number of Variables Compared with Some Observations Unequal: 0.
Total Number of Values which Compare Unequal: 0.
Total Number of Values not EXACTLY Equal: 99.
Maximum Difference: 1.1369E-13.
```

## Input Time Series Keywords

The user can specify keywords related to the input time series in the EXMFUNC option of the SPECIFICATION statement in the HPFSELECT procedure. The _TIMEID_ keyword specifies that the time ID values are passed as input arrays to the user-defined forecasting model. The _SEASON_ keyword specifies that the season index values are passed as input arrays to the user-defined forecasting model.

## Returned Forecast Component Keywords

A user-defined forecasting function or subroutine must return the predictions, specified by the keyword _PREDICT_ in the signature description found in the EXMFUNC option of the SPECIFICATION statement in the HPFSELECT procedure. The prediction standard errors and the lower and upper confidence limits are optional and are specified by the keywords, _STDERR_, _LOWER_, and _UPPER_, respectively. The HPFENGINE procedure computes the forecast components that are not returned by the user-defined forecasting function based on the external model specification.

# Chapter 24
# Using External Forecasts

**Contents**

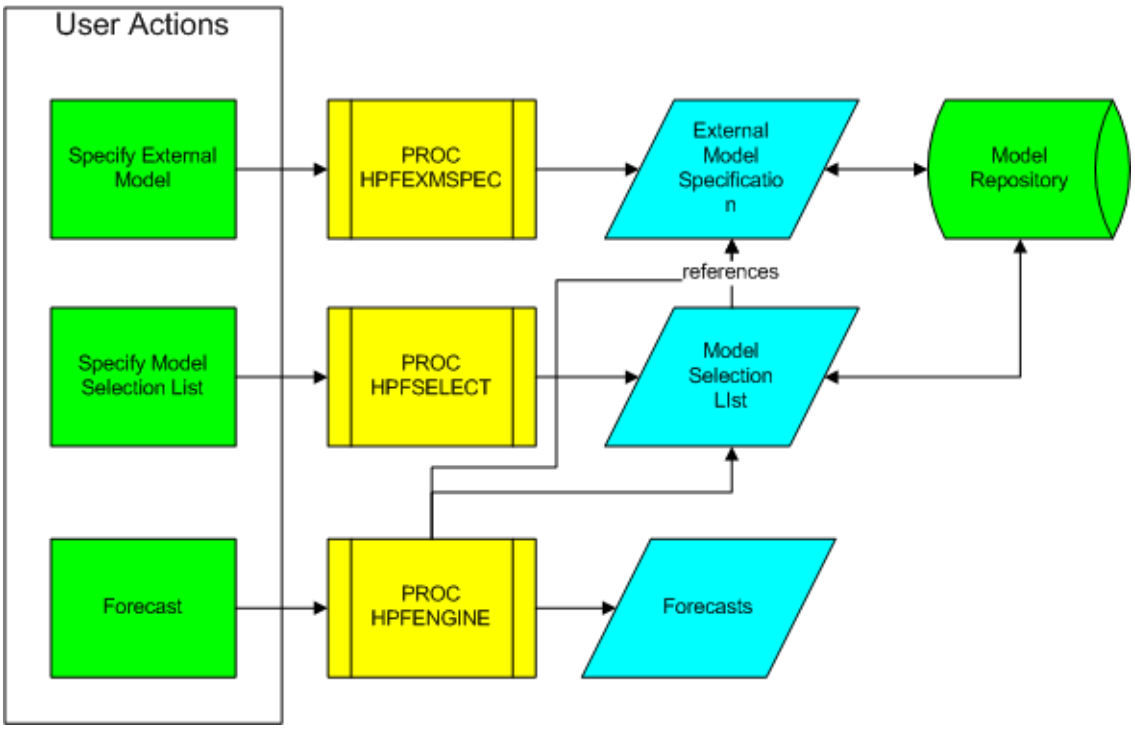

Introduction . . . . . 657
Specifying and Using an External Model Specification . . . . . 658


## Introduction

External forecasts are forecasts provided by an external source. External forecasts can originate from a statistical model or from another software package, and can have been provided by an outside organization (for example, marketing organization or government agency), or might be given based solely on judgment.

To use an external forecast in SAS Forecast Server Procedures software, an external model specification must be specified using the HPFEXMSPEC procedure. The model specification describes statistical properties of how the forecast was derived. Figure 24.1 describes an example of the ways external forecasts can be specified and used to forecast.

**Figure 24.1** Using External Forecasts



## Specifying and Using an External Model Specification

The HPFEXMSPEC procedure specifies external models for use by the SAS Forecast Server Procedures software. The HPFEXMSPEC procedure enables the user to specify information about how the forecasts were derived. This information is used to compute forecast components that are not provided by the user.

For example, the data set Sashelp.Air contains a monthly time series represented by two variables: DATE and AIR. The following SAS statements create an external forecast in the log transformed metric and store the external forecasts in the data set Work.Externalforecast. In this example, the HPF procedure is used as the source of the forecasts; but one could imagine that the origin could be any external source.

```
proc timeseries data=sashelp.air
                out=logair(rename=air=logair);
   id date interval=month;
   var air / transform=log;
run;

proc hpf data=logair out=hpfout
         outfor=hpfforecast;
   id date interval=month;
   forecast logair / model=addwinters;
run;

data externalforecast;
   drop _NAME_ ACTUAL ERROR;
   merge sashelp.air hpfforecast; by date;
run;
```

The data set Work.Externalforecast contains six variables: DATA, AIR, PREDICT, STD, LOWER and UPPER.

The HPFEXMSPEC procedure can be used to create an external model specification. Continuing the example, the following SAS statements create an external model specification called MYEXTERNAL and store this model specification in the model repository Sasuser.Mymodels. The TRANSFORM=LOG option specifies that the external forecasts were generated in the log transformed metric and these forecasts need to be inverse-transformed back to the original metric. The NPARMS=3 option specifies the number of parameters used to generate the external forecasts.

```
proc hpfexmspec modelrepository=sasuser.myrepository
                specname=myexternal;
    exm transform=log nparms=3;
run;
```

The HPFSELECT procedure can be used to create a model selection list that contains an external model specification as a possible candidate model. Continuing the example, the following SAS statements create a model selection list called MYSELECT and store this model selection list in the catalog Sasuser.Myrepository. The EXMMAP option in the following SPEC statement provides the mapping for the external model's symbols to the data set variables. The PREDICT= option specifies the variable that contains the predictions, the STDERR= option specifies the variable that contains the prediction standard errors, the LOWER= option specifies the variable that contains the lower confidence limits, and the UPPER= option specifies the variable that contains the upper confidence limits.

```
proc hpfselect modelrepository=sasuser.myrepository
               selectname=myselect
               selectlabel="My Select List";
   spec myexternal /
        exmmap(predict=predict lower=lower upper=upper stderr=std);
run;
```

At this point:

- External forecasts are contained in the data set Work.Externalforecast.
- An external model specification, MYEXTERNAL, has been stored in the model repository, Sasuser.Myrepository.
- A model selection list, MYSELECT, has been stored in the model repository, Sasuser.Myrepository.

The HPFENGINE procedure can now use both the external model specification and the external forecasts.

Continuing the example, the following SAS statements forecast the monthly time series contained in the Work.Externalforecast data set. The MODELREPOSITORY option of the PROC HPFENGINE statement specifies the model repository as Sasuser.Myrepository, the GLOBALSELECTION=MYSELECT option specifies the model selection list, and the EXTERNAL statement specifies the data set variables required to satisfy the mappings defined in the SPEC statement's EXMMAP option for the external model, MYEXTERNAL.

```
proc hpfengine data=externalforecast
               out=engout
               outfor=engforecast
               outstat=outstat
               modelrepository=sasuser.myrepository
               globalselection=myselect;
   id date interval=month;
   forecast air;
   external predict lower upper std;
run;
```

The OUT=ENGOUT data set contains the original data extrapolated by the external forecasts and the OUTFOR=ENGFORECAST data set contains the forecasts (values contained in the PREDICT=, STDERR=, LOWER=, and UPPER= data set variables) and the prediction errors. The OUTSTAT=ENGSTAT data set contains the statistics of fit based on the prediction errors and the NPARMS=3 option of the external model specification.

# Chapter 25
# Using Auxiliary Data Sets

**Contents**


Auxiliary Data Summary . . . . . **661**
AUXDATA Functionality . . . . . **661**
AUXDATA Alignment across BY Groups . . . . . 663
AUXDATA Alignment over the Time Dimension . . . . . 663
AUXDATA Examples . . . . . **666**
Example 25.1: Simple AUXDATA Demonstration . . . . . 666
Example 25.2: AUXDATA with BY Groups . . . . . 670


## Auxiliary Data Summary

Auxiliary data set support enables the HPFENGINE and HPFDIAGNOSE procedures to use auxiliary data sets to contribute input variables to the run of the procedure step. This functionality creates a virtual data source that enables some of the input variables to physically reside in different data sets; in previous versions, all variables were required to be physically present in a single DATA= data set. For example, this functionality enables sharing of common explanatory time series data across multiple forecasting projects.

Furthermore, auxiliary data set support enables more than the simple separation of shared data. It also facilitates the elimination of redundancy in these auxiliary data sources by performing partial matching on BY-group qualification. Duplication of explanatory time series for the full BY-group hierarchy is no longer required for the auxiliary data sets.

Finally, this functionality permits more than one auxiliary data source to be used concurrently to materialize the virtual time series vectors across a given BY-group hierarchy. So explanatory variables that have naturally different levels of BY-group qualification can be isolated into separate data sets and supplied with separate AUXDATA= options to optimize data management and performance.

## AUXDATA Functionality

Both the HPFDIAGNOSE and HPFENGINE procedures now support the AUXDATA=*DataSet* option. When used, this option declares the presence of an auxiliary data set to optionally provide input variables to satisfy various declaration statements in the respective procedure steps.

There are two classes of time series data set sources:

- a primary data set from the DATA=*DataSet* option
- auxiliary data sources from AUXDATA=*DataSet* options

You can specify zero or more AUXDATA= options in the PROC statement. Each AUXDATA= option establishes an auxiliary data set source to supply variables declared in subsequent statements that comprise the procedure step.

Variables referenced in the PROC invocation fall into two classes:

- those that must be physically present in the primary data set
- those that can reside in either the primary or an auxiliary data set

For those that can be resolved from an auxiliary data set, variable resolution proceeds in reverse order from the last AUXDATA= option in the PROC statement to the first. If the variable in question is not found in any of those, the variable must be present in the primary data set for the procedure step to be successful.

Currently, for the HPFENGINE and HPFDIAGNOSE procedures, dependent variables and the variable identified in the ID statement must be physically present in the primary data set. Dependent variables are those that are identified in FORECAST statements in the respective HPFENGINE or HPFDIAGNOSE procedure invocations. The ID variable, if specified, must also be present in each of the auxiliary data sets with the same variable name and units as in the primary data set.

For variables that are candidates for placement in auxiliary data sets, the procedure performs a lookup step. The lookup proceeds in reverse order across the auxiliary data sets specified in the PROC statement.

For PROC HPFDIAGNOSE, variables in the following statements are candidates for AUXDATA placement:

- ADJUST (right-hand-side variables)
- INPUT

For PROC HPFENGINE, variables in the following statements are candidates for AUXDATA placement:

- ADJUST (right-hand-side variables)
- CONTROL
- EXTERNAL
- INPUT
- STOCHASTIC

## AUXDATA Alignment across BY Groups

All BY statement variables must be physically present in the primary data set. However, it is not necessary to have the BY variables present in any of the auxiliary data sets. All, some, or none of the BY variables can be present in any auxiliary data set, as your requirements dictate. Partial BY-group matching is performed between the primary data set and the auxiliary data sets based on the number of BY statement variables that are present in the respective auxiliary data sets.

For example, suppose you have a hierarchy of (REGION, PRODUCT) in the primary data set, which holds the time series variables for monthly sales metrics. Suppose you have an auxiliary data set with time series qualified by REGION for pertinent explanatory variables and another with time series for other explanatory variables to be applied across all (REGION, PRODUCT) groupings of the primary data set. In this scenario, each (REGION, PRODUCT) group in the primary data set seeks a match with a corresponding REGION from the first auxiliary data set to materialize the time series for its variables, but no matching is performed on the second auxiliary data set to materialize the time series for its variables. So if ('SOUTH','EDSEL') is a BY group from the primary data set, the 'SOUTH' BY group series from the first auxiliary data set are used, and the series from the second auxiliary data set are supplied without qualification. If the next primary BY group is ('SOUTH', 'HUDSON'), then the 'SOUTH' BY group is again used to supply the time series from the first auxiliary data set, and the unqualified series are supplied from the second auxiliary data set. So on it goes, each auxiliary data set performing a partial match on the BY variables it holds within the BY group from the primary data set.

## AUXDATA Alignment over the Time Dimension

The series from each BY group of the primary data set defines a reference time span for the auxiliary data sets. Only the intersection of the time interval for each auxiliary series with the reference span is materialized. Head or tail missing values are inserted into the auxiliary series for start or stop times that lie inside the reference span. More generally, missing value semantics apply to the head and tail regions that require filling to materialize the full reference time span.

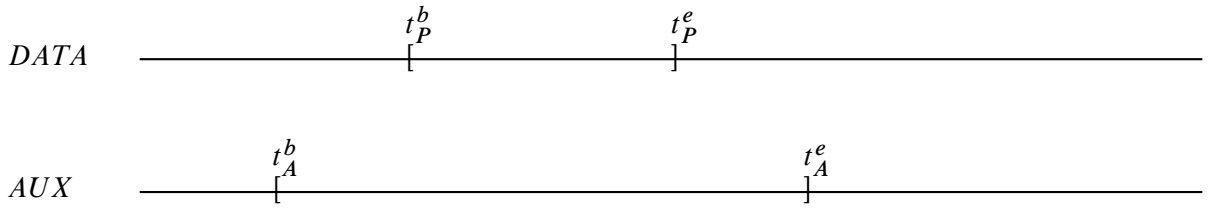
With time series materialized from a single primary data set, there is no latitude for different time ID ranges between the different variables because each observation read contains not only the time ID but also the associated values for all of the variables. With some series materialized from the primary data set and some materialized from auxiliary data sets, the possibility exists for the reference time span to have an arbitrary intersection with the time span of the corresponding series from the auxiliary sources. The intent is to materialize the portion of the auxiliary series time span that intersects with the reference time span and to handle head and tail shortages via missing value semantics as needed.

For the previous usage scenario with a primary data set and two auxiliary data sets, when data is read over a sequence of primary BY groups it might be necessary to materialize various spans of the auxiliary series with appropriate missing value semantics applied as needed to resolve head and tail shortages even though the actual time series contributed from the auxiliary data sets does not physically change. The following discussion breaks this down into several cases depending on intersection possibilities between the reference time span and the auxiliary time span.

Legend:

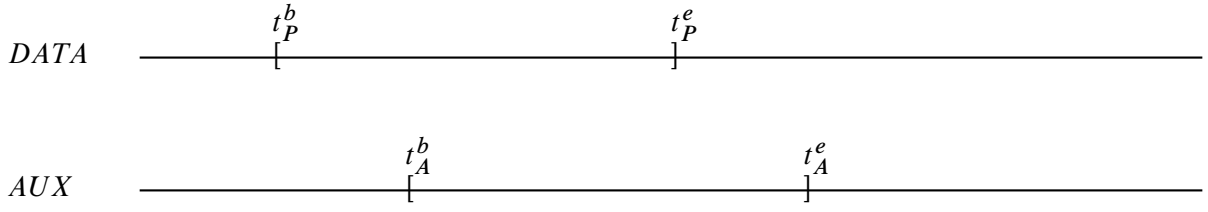
- $t_P^b$ denotes the begin time ID of the primary (DATA=) series.
- $t_P^e$ denotes the end time ID of the primary (DATA=) series.
- $t_A^b$ denotes the begin time ID of the AUXDATA series.
- $t_A^e$ denotes the end time ID of the AUXDATA series.
- $[t_P^b, t_P^e]$ denotes the time span for the primary (DATA=) series (also known as the reference time span).
- $[t_A^b, t_A^e]$ denotes the time span for the AUXDATA series.

### Case 1:



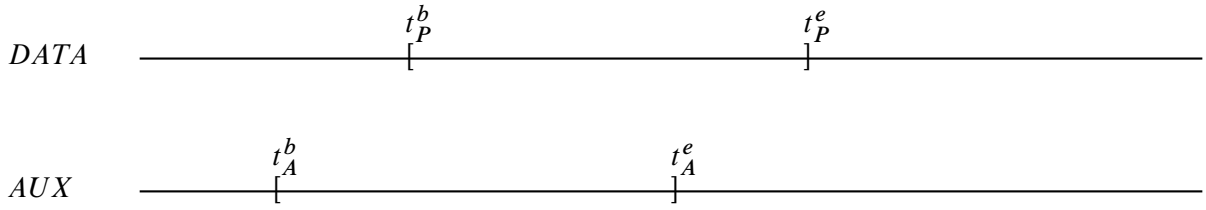

Here $[t_P^b, t_P^e] \subseteq [t_A^b, t_A^e]$. The auxiliary time span includes the reference span as a subset. Values in the AUXDATA series to the left of $t_P^b$ and values to the right of $t_P^e$ are truncated from the AUXDATA series that is materialized in connection with the primary series.

### Case 2:



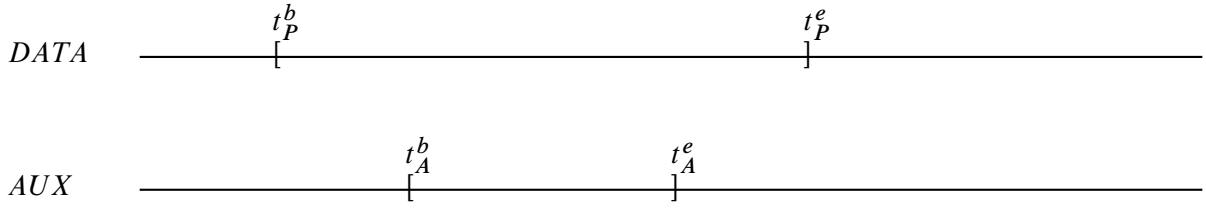

Here $[t_P^b, t_P^e] = [t_P^b, t_A^b) \cup [t_A^b, t_P^e]$. The reference time span leads the auxiliary time span with a non-empty intersection. AUXDATA series values in $[t_P^b, t_A^b)$ are materialized with missing value semantics. AUXDATA series values in $[t_A^b, t_P^e]$ are materialized as actual subject to missing value semantics.

**Case 3:**



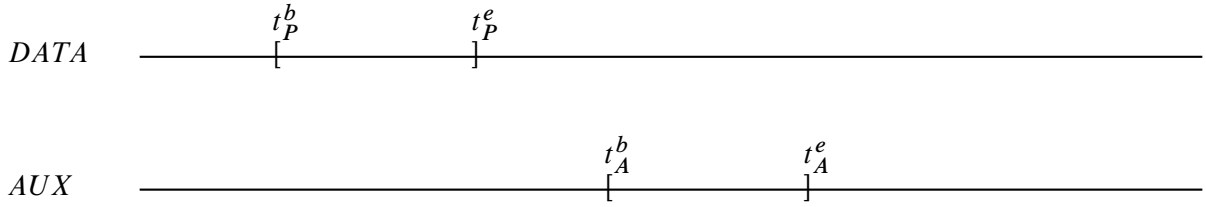

Here $[t_P^b, t_P^e] = [t_P^b, t_A^e] \cup (t_A^e, t_P^e]$. The reference time span lags the auxiliary time span with a non-empty intersection. AUXDATA series values in $[t_P^b, t_A^e]$ are materialized as actual subject to missing value semantics. AUXDATA series values in $(t_A^e, t_P^e]$ are materialized with missing value semantics.

**Case 4:**




Here $[t_A^b, t_A^e] \subset [t_P^b, t_P^e]$. The auxiliary time span is a subset of the reference time span. AUXDATA series values in $[t_P^b, t_A^b)$ and values in $(t_A^e, t_P^e]$ are materialized with missing value semantics. AUXDATA series values in $[t_A^b, t_A^e]$ are materialized as actual subject to missing value semantics.

**Case 5:**




Here $[t_P^b, t_P^e] \cap [t_A^b, t_A^e] = \oslash$. The auxiliary time span does not intersect the reference time span at all. In this case all AUXDATA series values are materialized with missing value semantics.

# AUXDATA Examples

## Example 25.1: Simple AUXDATA Demonstration

This is a simple demonstration of the use of AUXDATA in the context of PROC HPFDIAGNOSE and PROC HPFENGINE that shows the separation of explanatory variables from forecast variables.

The following statements create an unobserved components model (UCM) specification, MYUCM, with an input X1, and a model selection list, MYSELECT, to map the model's Y symbol to data set variable MASONRY and its X1 symbol to the data set variable ELECTRIC:

```
proc hpfucmspec repository=work.mymodels name=myucm;
   dependent symbol=Y;
   irregular;
   level;
   slope;
   season length=12;
   input symbol=X1;
run;

proc hpfselect repository=work.mymodels name=myselect;
   spec myucm / inputmap(symbol=Y var=MASONRY)
                inputmap(symbol=X1 var=ELECTRIC);
run;
```

The following statements run PROC HPFENGINE with a data set Workers as its single data source.

```
proc hpfengine data=workers
               repository=work.mymodels
               globalselection=myselect
               out=wout
               outcomponent=wcomp
               outest=west
               outfor=wfor
               outstat=wstat
               outmodelinfo=wmodel
               outindep=wind
               print=select;
   id date interval=month;
   forecast masonry;
   stochastic electric;
run;
```

The following statements show how you can use that Workers data set as both DATA= and AUXDATA= sources by judicious use of the KEEP= data set option. There is no practical advantage to this. It simply demonstrates the scoping of the AUXDATA feature to resolve variables from the proper data set and materialize the same result regardless of how PROC HPFENGINE acquires the time series it processes.

```
proc hpfengine data=workers(keep=date masonry)
               auxdata=workers
               repository=work.mymodels
               globalselection=myselect
               out=woutx
               outcomponent=wcompx
               outest=westx
               outfor=wforx
               outstat=wstatx
               outmodelinfo=wmodelx
               outindep=windx
               print=select;
   id date interval=month;
   forecast masonry;
   stochastic electric;
run;
```

But perhaps you are skeptical. Is this is really working? After all, Workers is the input data set and it does have all of the columns.

So split up Workers into Workdep and Workaux data sets as follows:

```
   data workdep;
      set workers(keep=date masonry);
   run;

   data workaux;
      set workers(keep=date electric);
   run;
```

Then run PROC HPFENGINE again on the separated data sets Workdep and Workaux:

```
proc hpfengine data=workdep
               auxdata=workaux
               repository=work.mymodels
               globalselection=myselect
               out=woutx2
               outcomponent=wcompx2
               outest=westx2
               outfor=wforx2
               outstat=wstatx2
               outmodelinfo=wmodelx2
               outindep=windx2
               print=select;
   id date interval=month;
   forecast masonry;
   stochastic electric;
run;
```

You can use the COMPARE procedure to compare the output data sets between this and either of the previous steps. You will see some differences in the data set labels for some of them; everything else should be the same.

```
proc compare data=wfor(label='x')
             compare=wforx2(label='x');
run;
```

**Output 25.1.1** Compare AUXDATA OUTFOR Results

```
                        The COMPARE Procedure
                 Comparison of WORK.WFOR with WORK.WFORX2
                              (Method=EXACT)

                           Data Set Summary

Dataset               Created          Modified  NVar    NObs  Label

WORK.WFOR    26MAR14:08:10:34  26MAR14:08:10:34     8      79  x
WORK.WFORX2  26MAR14:08:10:35  26MAR14:08:10:35     8      79  x


                          Variables Summary

                  Number of Variables in Common: 8.


                         Observation Summary

                     Observation      Base  Compare

                     First Obs           1        1
                     Last  Obs          79       79

  Number of Observations in Common: 79.
  Total Number of Observations Read from WORK.WFOR: 79.
  Total Number of Observations Read from WORK.WFORX2: 79.

  Number of Observations with Some Compared Variables Unequal: 0.
  Number of Observations with All Compared Variables Equal: 79.

  NOTE: No unequal values were found. All values compared are exactly
        equal.
```

You can also run this example on a combination of the HPFDIAGNOSE and HPFENGINE procedures as follows. First, run PROC HPFDIAGNOSE and PROC HPFENGINE on the full Workers data set:

```
proc hpfdiagnose data=workers
                 repository=work.wdiagnose
                 outest=wdiagest;
   id date interval=month;
   forecast masonry;
   input electric;
   arimax identify=both;
   ucm component=(all);
run;

proc hpfengine data=workers
               repository=work.wdiagnose
               inest=wdiagest
```

```
                out=wout
                outcomponent=wcomp
                outest=west
                outfor=wfor
                outstat=wstat
                outmodelinfo=wmodel
                outindep=wind
                print=select;
   id date interval=month;
   forecast masonry;
   stochastic electric;
run;
```

Next, do the same thing using the Workdep and Workaux data sets from above:

```
proc hpfdiagnose data=workdep
                 auxdata=workaux
                 repository=work.wdiagnosex
                 outest=wdiagestx;
   id date interval=month;
   forecast masonry;
   input electric;
   arimax identify=both;
   ucm component=(all);
run;

proc hpfengine data=workdep
               auxdata=workaux
               repository=work.wdiagnosex
               inest=wdiagestx
               out=woutx
               outcomponent=wcompx
               outest=westx
               outfor=wforx
               outstat=wstatx
               outmodelinfo=wmodelx
               outindep=windx
               print=select;
   id date interval=month;
   forecast masonry;
   stochastic electric;
run;
```

To finish up this example, run PROC COMPARE on the OUTFOR data sets between the two runs:

```
proc compare data=wfor(label='x')
             compare=wforx(label='x');
run;
```

**Output 25.1.2** Compare AUXDATA OUTFOR Results

```
                    The COMPARE Procedure
              Comparison of WORK.WFOR with WORK.WFORX
                        (Method=EXACT)

                      Data Set Summary

Dataset              Created          Modified  NVar    NObs  Label

WORK.WFOR   26MAR14:08:10:39  26MAR14:08:10:39     8      79  x
WORK.WFORX  26MAR14:08:10:40  26MAR14:08:10:40     8      79  x


                      Variables Summary

                Number of Variables in Common: 8.


                     Observation Summary

                  Observation      Base  Compare

                  First Obs           1        1
                  Last  Obs          79       79

  Number of Observations in Common: 79.
  Total Number of Observations Read from WORK.WFOR: 79.
  Total Number of Observations Read from WORK.WFORX: 79.

  Number of Observations with Some Compared Variables Unequal: 0.
  Number of Observations with All Compared Variables Equal: 79.

  NOTE: No unequal values were found. All values compared are exactly
        equal.
```

## Example 25.2: AUXDATA with BY Groups

This example is a more complicated case that demonstrates the use of partial BY-group matching between the primary source specified in the DATA= option and multiple AUXDATA= data sets. The example is contrived, but it demonstrates the utility of AUXDATA for consolidating explanatory data and eliminating redundancy. Suppose you have collection of time series that represent demand for a service qualified by the variables REGION and CLASS, which represent a region of the country and the service class. Suppose you want to include regional variables in the model as regressors and also some global (unqualified) variables as regressors. With the single-source DATA= mode of the HPFDIAGNOSE and HPFENGINE procedures, you would have to replicate each of those regional variables and each of those global variables in the SAS data set you provide as input to those respective procedures in order to perform time series diagnosis and subsequent model selection and forecasting. If there are hundreds of thousands (or more) distinct groups of REGION and CLASS, then the storage cost and I/O penalty for storing and moving all that redundant information can become significant, especially if that common data can be used across a variety of forecasting projects in the organization. So how would the SAS code look to use these kinds of data?

Some data are fabricated to fit this structure. The same series is replicated in each BY group regardless of BY-group level. The following details for the data sets constructed for this example are pertinent:

- REGION and CLASS each have two distinct values for a total of four primary BY groups.
- The Master data set contains the fully qualified, single-source version of the data that you would use as the data source if you did not have AUXDATA support. It has lots of redundant information for the explanatory variables included in it as a result.
- The Depdata data set contains the dependent variables to be forecast, the time ID, and the BY variables.
- The Regdata data set contains the regional independent variables, the time ID, and the REGION BY variable.
- The Natdata data set contains the global independent variables and time ID.

The following statements set up PROC HPFDIAGNOSE and PROC HPFENGINE runs to compare the results. First, the single-source run is just a matter of running PROC HPFDIAGNOSE over the Master data set and passing the generated model repository and OUTEST data set into PROC HPFENGINE to forecast the Master data set. This would have been the only option available to you for this analysis prior to AUXDATA support.

```
proc hpfdiagnose data=master
                 repository=work.diagrep
                 outest=diagest;
   by region class;
   id date interval=day;
   forecast y;
   input x z;
   arimax identify=both;
   ucm component=(all);
run;

proc hpfengine data=master
               repository=work.diagrep
               inest=diagest
               out=ex2out
               outcomponent=ex2comp
               outest=ex2est
               outfor=ex2for
               outstat=ex2stat
               outmodelinfo=ex2model
               outindep=ex2indep
               print=select;
   by region class;
   id date interval=day;
   forecast y;
   input x z;
run;
```

The following statements use the AUXDATA approach. The fully qualified dependent variables of interest to analyze and forecast are in the Depdata data set. The regional independent variables are in Regdata and the global (unqualified) independent variables are in Natdata. The following HPFDIAGNOSE and HPFENGINE procedure steps perform the same analysis, model selection, and forecast, but they use the normalized data sources as input instead of the single-source Master data set:

```
proc hpfdiagnose data=depdata
                 auxdata=natdata
                 auxdata=regdata
                 repository=work.diagrepx
                 outest=diagestx;
   by region class;
   id date interval=day;
   forecast y;
   input x z;
   arimax identify=both;
   ucm component=(all);
run;

proc hpfengine data=depdata
               auxdata=natdata
               auxdata=regdata
               repository=work.diagrepx
               inest=diagestx
               out=ex2outx
               outcomponent=ex2compx
               outest=ex2estx
               outfor=ex2forx
               outstat=ex2statx
               outmodelinfo=ex2modelx
               outindep=ex2indepx
               print=select;
   by region class;
   id date interval=day;
   forecast y;
   input x z;
run;
```

To finish up, the following statements compare the statistics of fit to see that indeed the same results are produced regardless of how the data are obtained. While the other output data sets are omitted here for brevity, they should also compare cleanly.

```
proc compare data=ex2stat(label='x')
             compare=ex2statx(label='x');
run;
```

**Output 25.2.1** Compare AUXDATA Statistics of Fit Results

```
                        The COMPARE Procedure
                Comparison of WORK.EX2STAT with WORK.EX2STATX
                              (Method=EXACT)

                           Data Set Summary

Dataset                 Created          Modified  NVar    NObs  Label

WORK.EX2STAT   26MAR14:08:10:43  26MAR14:08:10:43    59       4  x
WORK.EX2STATX  26MAR14:08:10:45  26MAR14:08:10:45    59       4  x


                           Variables Summary

                  Number of Variables in Common: 59.


                          Observation Summary

                      Observation      Base  Compare

                      First Obs           1        1
                      Last  Obs           4        4

   Number of Observations in Common: 4.
   Total Number of Observations Read from WORK.EX2STAT: 4.
   Total Number of Observations Read from WORK.EX2STATX: 4.

   Number of Observations with Some Compared Variables Unequal: 0.
   Number of Observations with All Compared Variables Equal: 4.

   NOTE: No unequal values were found. All values compared are exactly
         equal.
```

# Subject Index


additive-invertible region
    smoothing weights, 577
adjusted R-square
    statistics of fit, 602
AIC, *see* Akaike's information criterion
AICC, *see* Akaike's information criterion, finite sample size corrected
AICC weights, 616
Akaike's information criterion
    AIC, 602
    AICC, 602
    statistics of fit, 602
Akaike's information criterion, finite sample size corrected
    statistics of fit, 602
Amemiya's prediction criterion
    statistics of fit, 602
Amemiya's R-square
    statistics of fit, 602
ARIMA models
    forecasting models, 589
arima models
    ARIMA model, 589
ARMA Order Selection
    HPFDIAGNOSE procedure, 130
average weights, 614

Base SAS software, 14
BIC, *see* Schwarz Bayesian information criterion
boundaries
    smoothing weights, 577
Box Cox
    transformations, 598
Box Cox transformation, *see* transformations
Brown smoothing model, *see* double exponential smoothing
BY groups
    HPF procedure, 35
    HPFENGINE procedure, 201
    HPFEVENTS procedure, 281, 285
    HPFRECONCILE procedure, 426
    HPFTEMPRECON procedure, 495

calculations
    smoothing models, 575
character functions, 16
combination model, 607
combination process, 609
combined forecast error variance, 620
corrected sum of squares
    statistics of fit, 600
Croston's, *see* intermittent models

damped-trend exponential smoothing, 582
    smoothing models, 582
data preparation
    HPFORECAST procedure, 374
DATA step, 14
    SAS data sets, 14
diagnostic tests, 599
Differencing variables in a UCM
    HPFDIAGNOSE procedure, 135
double exponential smoothing, 579
    Brown smoothing model, 579
    smoothing models, 579

error sum of squares
    statistics of fit, 601
ESTIMATE groups
    HPFARIMASPEC procedure, 82
EVENTBY= Data Set
    HPFDIAGNOSE procedure, 144
EVENTS
    HPFDIAGNOSE procedure, 139
Events in a UCM
    HPFDIAGNOSE procedure, 139
Events in an ARIMAX Model
    HPFDIAGNOSE procedure, 139
exponential smoothing, *see* smoothing models
Exponential Smoothing Model
    HPFDIAGNOSE procedure, 135

FORECAST groups
    HPFARIMASPEC procedure, 79
    HPFUCMSPEC procedure, 529
forecast model selection graph examples, 624
forecast model selection graph ODS plot behavior, 638
forecast model selection graph ODS print behavior, 638
forecast model selection graph operation, 633
forecast model selection graph OUTEST, 637
forecast model selection graph OUTFOR, 638
forecast model selection graph OUTMODELINFO, 637
forecast model selection graph OUTMSUM, 637
forecast model selection graph OUTSTAT, 637
forecast model selection graph OUTSTATSELECT, 636

forecasting, 574
forecasting models
    ARIMA models, 589
    intermittent models, 591
    smoothing models, 575
    UCM models, 590
Functional Transformation
    HPFDIAGNOSE procedure, 127

geometric mean absolute percent error
    statistics of fit, 601
geometric mean absolute predictive percent error
    statistics of fit, 602
geometric mean absolute symmetric percent error
    statistics of fit, 601
geometric mean relative absolute error
    statistics of fit, 602
goodness-of-fit statistics, *see* statistics of fit

help system, 13
Holt smoothing model, *see* linear exponential smoothing
Holt-Winters Method, *see* Winters Method
HPF procedure
    BY groups, 35
    ODS graph names, 60
HPFARIMASPEC procedure
    ESTIMATE groups, 82
    FORECAST groups, 79
    INPUT groups, 80
HPFDIAGNOSE procedure
    ARMA Order Selection, 130
    Differencing variables in a UCM, 135
    EVENTBY= Data Set, 144
    EVENTS, 139
    Events in a UCM, 139
    Events in an ARIMAX Model, 139
    Exponential Smoothing Model, 135
    Functional Transformation, 127
    HPFENGINE, 142
    INEVENT= Data Set, 145
    Input/Output Data Sets, 144
    Intermittent Demand Model, 134
    Joint Differencing, 129
    Missing Values, 125
    OUTEST= Data Set, 145
    Outliers, 133
    OUTOUTLIER= Data Set, 147
    Season Dummy Test, 130
    Stationarity Test, 128
    Threading, 141
    Transfer Function in a UCM, 136
    Transfer Function in an ARIMAX Model, 131
    Unobserved Components Model, 135
HPFENGINE
    HPFDIAGNOSE procedure, 142
HPFENGINE procedure
    BY groups, 201
    ODS graph names, 232
    Threading, 218
HPFEVENTS procedure
    BY groups, 281, 285
HPFORECAST procedure
    data preparation, 374
    LEVEL groups, 354
    multithreading, 356
    output, 380
    parallel processing, 363
HPFRECONCILE procedure
    BY groups, 426
HPFTEMPRECON procedure
    BY groups, 495
HPFUCMSPEC procedure
    FORECAST groups, 529
    INPUT groups, 530
    parameters, 526–529, 532–535

INEVENT= Data Set
    HPFDIAGNOSE procedure, 145
initializations
    smoothing models, 576
INPUT groups
    HPFARIMASPEC procedure, 80
    HPFUCMSPEC procedure, 530
Input/Output Data Sets
    HPFDIAGNOSE procedure, 144
Intermittent Demand Model
    HPFDIAGNOSE procedure, 134
intermittent models
    Croston's Method, 591
    forecasting models, 591

Joint Differencing
    HPFDIAGNOSE procedure, 129

LAD weights, 617
LEVEL groups
    HPFORECAST procedure, 354
linear exponential smoothing, 581
    Holt smoothing model, 581
    smoothing models, 581
log
    transformations, 598
log test, 599
log transformation, *see* transformations
logistic
    transformations, 598

maximum absolute percent error

statistics of fit, 601
maximum absolute predictive percent error
statistics of fit, 602
maximum absolute symmetric percent error
statistics of fit, 601
maximum relative absolute error
statistics of fit, 602
mean absolute error
statistics of fit, 601
mean absolute percent error
statistics of fit, 601
mean absolute predictive percent error
statistics of fit, 602
mean absolute scaled error
statistics of fit, 602
mean absolute symmetric percent error
statistics of fit, 601
mean percent error
statistics of fit, 603
mean prediction error
statistics of fit, 603
mean relative absolute error
statistics of fit, 602
mean square error
statistics of fit, 601
median absolute percent error
statistics of fit, 601
median absolute predictive percent error
statistics of fit, 602
median absolute symmetric percent error
statistics of fit, 601
median relative absolute error
statistics of fit, 602
minimum absolute percent error
statistics of fit, 601
minimum absolute predictive percent error
statistics of fit, 601
minimum absolute symmetric percent error
statistics of fit, 601
minimum relative absolute error
statistics of fit, 602
Missing Values
HPFDIAGNOSE procedure, 125
missing values
smoothing models, 576
MMAE, 598
MMSE, 598
model evaluation, 574
multiplicative seasonal smoothing, 585
smoothing models, 585
multithreading
HPFORECAST procedure, 356

nonmissing observations
statistics of fit, 600
number of observations
statistics of fit, 600

ODS graph names
HPF procedure, 60
HPFENGINE procedure, 232
OLS weights, 616
optimizations
smoothing weights, 577
OUTEST= Data Set
HPFDIAGNOSE procedure, 145
Outliers
HPFDIAGNOSE procedure, 133
OUTOUTLIER= Data Set
HPFDIAGNOSE procedure, 147
output
HPFORECAST procedure, 380

parallel processing
HPFORECAST procedure, 363
parameter estimation, 574
parameters
HPFUCMSPEC procedure, 526–529, 532–535
predictions
smoothing models, 576

R-square statistic
statistics of fit, 602
random walk R-square
statistics of fit, 602
rank weights, 615
ranked user weights, 615
Restricted LSQ weights, 617
RMSE weights, 615
root mean square error
statistics of fit, 601

SAS data sets
DATA step, 14
SBC, *see* Schwarz Bayesian information criterion
Schwarz Bayesian information criterion
BIC, 602
SBC, 602
statistics of fit, 602
Season Dummy Test
HPFDIAGNOSE procedure, 130
seasonal exponential smoothing, 584
smoothing models, 584
seasonality test, 599
series diagnostics, 599
simple exponential smoothing, 578
smoothing models, 578
smoothing equations, 575
smoothing models, 575

smoothing models
  calculations, 575
  damped-trend exponential smoothing, 582
  double exponential smoothing, 579
  exponential smoothing, 575
  forecasting models, 575
  initializations, 576
  linear exponential smoothing, 581
  missing values, 576
  multiplicative seasonal smoothing, 585
  predictions, 576
  seasonal exponential smoothing, 584
  simple exponential smoothing, 578
  smoothing equations, 575
  smoothing state, 575
  smoothing weights, 577
  standard errors, 578
  underlying model, 575
  Winters Method, 586, 588
smoothing state, 575
  smoothing models, 575
smoothing weights, 577
  additive-invertible region, 577
  boundaries, 577
  optimizations, 577
  smoothing models, 577
  specifications, 577
  weights, 577
specifications
  smoothing weights, 577
square root
  transformations, 598
square root transformation, *see* transformations
standard errors
  smoothing models, 578
Stationarity Test
  HPFDIAGNOSE procedure, 128
statistics of fit, 600
  adjusted R-square, 602
  Akaike's information criterion, 602
  Akaike's information criterion, finite sample size corrected, 602
  Amemiya's prediction criterion, 602
  Amemiya's R-square, 602
  corrected sum of squares, 600
  error sum of squares, 601
  geometric mean absolute percent error, 601
  geometric mean absolute predictive percent error, 602
  geometric mean absolute symmetric percent error, 601
  geometric mean relative absolute error, 602
  goodness-of-fit statistics, 600
  maximum absolute percent error, 601
  maximum absolute predictive percent error, 602
  maximum absolute symmetric percent error, 601
  maximum relative absolute error, 602
  mean absolute error, 601
  mean absolute percent error, 601
  mean absolute predictive percent error, 602
  mean absolute scaled error, 602
  mean absolute symmetric percent error, 601
  mean percent error, 603
  mean prediction error, 603
  mean relative absolute error, 602
  mean square error, 601
  median absolute percent error, 601
  median absolute predictive percent error, 602
  median absolute symmetric percent error, 601
  median relative absolute error, 602
  minimum absolute percent error, 601
  minimum absolute predictive percent error, 601
  minimum absolute symmetric percent error, 601
  minimum relative absolute error, 602
  nonmissing observations, 600
  number of observations, 600
  R-square statistic, 602
  random walk R-square, 602
  root mean square error, 601
  Schwarz Bayesian information criterion, 602
  uncorrected sum of squares, 600

Threading
  HPFDIAGNOSE procedure, 141
  HPFENGINE procedure, 218
Time Series Reconciliation, *see also*
    HPFRECONCILE procedure, *see also*
    HPFRECONCILE procedure
Time Series Temporal Benchmarking, *see also*
    HPFTEMPRECON procedure
Time Series Temporal Reconciliation, *see also*
    HPFTEMPRECON procedure
Transfer Function in a UCM
  HPFDIAGNOSE procedure, 136
Transfer Function in an ARIMAX Model
  HPFDIAGNOSE procedure, 131
transformations
  Box Cox, 598
  Box Cox transformation, 598
  log, 598
  log transformation, 598
  logistic, 598
  square root, 598
  square root transformation, 598

UCM models
  forecasting models, 590
uncorrected sum of squares

statistics of fit, 600
underlying model
smoothing models, 575
Unobserved Components Model
HPFDIAGNOSE procedure, 135
used-defined weights, 614

weight methods, 614, 619
weights, *see* smoothing weights
Winters Method, 586, 588
Holt-Winters Method, 586
smoothing models, 586, 588

# Syntax Index


/POSTORDER option
PROC HPFREPOSITORY statement, 452
/PREORDER option
PROC HPFREPOSITORY statement, 452

ABEL= option
PROC HPFIDMSPEC statement, 325
ACCDATA=option
PROC HPFTEMPRECON statement, 493
ACCUMULATE= option
ADJUST statement, 108
ADJUST statement (ENG), 201
CONTROL statement (ENG), 202
FORECAST statement, 118
FORECAST statement (ENG), 204
FORECAST statement (HPF), 36
FORECAST statement (HPFORECAST), 351
ID statement, 118
ID statement (ENG), 205
ID statement (HPF), 40
INPUT statement, 121
INPUT statement (ENG), 208
INPUT statement (HPFORECAST), 352
STOCHASTIC statement (ENG), 210
TIMEID statement (HPFORECAST), 360
ADJUST statement
HPFDIAGNOSE procedure, 107
HPFENGINE procedure, 200
AFTER= option
EVENTDEF statement (HPFEVENTS), 283
AGGBY statement
HPFRECONCILE procedure, 425
AGGDATA Statement
HPFRECONCILE procedure, 426
AGGDATA=option
PROC HPFRECONCILE statement, 422
AGGREGATE= option
PROC HPFRECONCILE statement, 423
ALIGN= option
BENCHID statement (HPFTEMPRECON ), 496
ID statement, 118
ID statement (ENG), 206
ID statement (HPF), 41
ID statement (HPFEVENTS), 288
ID statement (HPFRECONCILE), 428
TIMEID statement (HPFORECAST), 361
ALIGN=option
ID statement (HPFTEMPRECON ), 498
ALPHA= option
FORECAST statement (HPF), 36
FORECASTOPTIONS statement (HPFSELECT), 470
PROC HPFDIAGNOSE statement, 100
PROC HPFRECONCILE statement, 423
AR= option
FORECAST statement (HPFARIMASPEC), 80
INPUT statement (HPFARIMASPEC), 82
ARIMAX statement
HPFDIAGNOSE procedure, 108
HPFDIAGSPEC procedure, 162
AUTOREG statement
HPFUCMSPEC procedure, 525
AUXDATA= option
PROC HPFDIAGNOSE statement, 101
PROC HPFENGINE statement, 188
AVERAGE= option
IDM statement (HPF), 44
IDM statement (HPFIDMSPEC), 326

BACK= option
PROC HPF statement, 32
PROC HPFDIAGNOSE statement, 101
PROC HPFENGINE statement, 188
PROC HPFORECAST statement, 344
BASE= option
IDM statement, 120, 171
IDM statement (HPF), 44
IDM statement (HPFIDMSPEC), 326
BASENAME= option
PROC HPFDIAGNOSE statement, 101
SCORE statement, 209
BEFORE= option
EVENTDEF statement (HPFEVENTS), 283
BENCHACCUMULATION=option
PROC HPFTEMPRECON statement, 494
BENCHCONS=option
PROC HPFTEMPRECON statement, 494
BENCHDATA=option
PROC HPFTEMPRECON statement, 493
BENCHDIFF option
PROC HPFTEMPRECON statement, 493
BENCHID statement
HPFTEMPRECON procedure, 496
BIASCORRECTION=option
PROCHPFTEMPRECON statement, 494
BLOCKSEASON statement

HPFUCMSPEC procedure, 526
BLOCKSIZE= option
BLOCKSEASON statement (HPFUCMSPEC), 527
Bottom-up Reconciliation
HPFRECONCILE procedure, 432
BOUNDS= option
IDM statement (HPF), 45
IDM statement (HPFIDMSPEC), 327
BY statement
HPF procedure, 35
HPFDIAGNOSE procedure, 111
HPFENGINE procedure, 201
HPFEVENTS procedure, 281
HPFRECONCILE procedure, 426
HPFTEMPRECON procedure, 495
BYVARSSORTED option
PROC HPFRECONCILE statement, 427

CHOOSE= specification
SELECT statement (HPFSELECT), 470
CLMETHOD= option
PROC HPFRECONCILE statement, 423
COMBINE statement
HPFDIAGNOSE procedure, 112
HPFDIAGSPEC procedure, 164
HPFSELECT procedure, 464
COMBINEBASE= option
PROC HPFDIAGNOSE statement, 101
COMMIT= option
PERFORMANCE statement, 357
COMPONENT= option
UCM statement, 124
UCM statement (PROC HPFDIAGSPEC), 174
COMPONENTS= option
PROC HPFENGINE statement, 188
CONDENSE option
EVENTDATA statement, 282
Confidence Limits
Bottom-up Reconciliation (HPFRECONCILE), 434
Top-Down Reconciliation (HPFRECONCILE), 432
CONSTRAINT= option
PROC HPFRECONCILE statement, 422
CONTROL statement
HPFENGINE procedure, 202
CONVERGE= option
ESTIMATE statement (HPFARIMASPEC), 83
CRITERION= option
ARIMAX statement, 109, 163
COMBINE statement (DIAG), 113
COMBINE statement (HPFDIAGSPEC procedure), 165
COMBINE statement (HPFSELECT), 465
DIAGNOSE statement (PROC HPFDIAGSPEC), 168
FORECAST statement (HPF), 36
IDM statement (HPF), 45
IDM statement (HPFIDMSPEC), 327
PROC HPFDIAGNOSE statement, 101
SELECT statement (HPFSELECT), 470
CRITERION=option
ESM statement (HPFESMSPEC), 263
CYCLE statement
HPFUCMSPEC procedure, 527

DAMPPARM= option
ESM statement (HPFESMSPEC), 265
IDM statement (HPF), 45
IDM statement (HPFIDMSPEC), 328
DAMPREST= option
ESM statement (HPFESMSPEC), 265
IDM statement (HPF), 45
IDM statement (HPFIDMSPEC), 328
DATA= option
PROC HPF statement, 32
PROC HPFDIAGNOSE statement, 102
PROC HPFENGINE statement, 189
PROC HPFEVENTS statement, 281
PROC HPFORECAST statement, 346
DATA=option
PROC HPFTEMPRECON statement, 493
DATASERVER= option
PERFORMANCE statement, 357
DELAY= option
INPUT statement (HPFARIMASPEC), 81
INPUT statement (HPFUCMSPEC), 530
DELAYEVENT= option
PROC HPFDIAGNOSE statement, 103
DELAYINPUT= option
DIAGNOSE statement (PROC HPFDIAGSPEC), 168
PROC HPFDIAGNOSE statement, 103
DELETE statement
HPFSELECT procedure, 468
DELTA= option
ESTIMATE statement (HPFARIMASPEC), 83
DEN= option
ARIMAX statement, 109, 163
INPUT statement (HPFARIMASPEC), 82
DEPLAG statement
HPFUCMSPEC procedure, 528
DETAILS option
PERFORMANCE statement, 357
DIAGNOSE statement
HPFSELECT procedure, 469

DIAGNOSE statement (HPFDIAGSPEC procedure), 168
DIAGNOSE= option
  FORECAST statement (HPFORECAST), 351
  PROC FORECAST statement, 347
DIF= option
  FORECAST statement (HPFARIMASPEC), 79
  INPUT statement (HPFARIMASPEC), 81
  INPUT statement (HPFUCMSPEC), 530
DIFF= option
  TREND statement, 123, 173
DIRECTION= option
  PROC HPFRECONCILE statement, 423
DISAGGDATA Statement
  HPFRECONCILE procedure, 427
DISAGGDATA=option
  PROC HPFRECONCILE statement, 422
DISAGGREGATION= option
  PROC HPFRECONCILE statement, 424
DISTBY statement
  HPFORECAST procedure, 350
DURATION= option
  EVENTDEF statement (HPFEVENTS), 283

ENCOMPASS= option
  COMBINE statement (DIAG), 113
  COMBINE statement (HPFDIAGSPEC procedure), 165
  COMBINE statement (HPFSELECT), 465
END LEVEL statement
  HPFORECAST procedure, 351
END= option
  BENCHID statement (HPFTEMPRECON ), 496
  ID statement, 118
  ID statement (ENG), 206
  ID statement (HPF), 41
  ID statement (HPFEVENTS), 288
  ID statement (HPFRECONCILE), 428
  ID statement (HPFTEMPRECON ), 498
  TIMEID statement (HPFORECAST), 361
ENTRYPCT= option
  DIAGNOSE statement (PROC HPFDIAGSPEC), 168
  PROC HPFDIAGNOSE statement, 103
ERRORCONTROL= option
  PROC HPFDIAGNOSE statement, 103
  PROC HPFENGINE statement, 189
ERRORTRACE= option
  PROC HPFRECONCILE statement, 423
ESM statement
  HPFDIAGNOSE procedure, 116
  HPFDIAGSPEC procedure, 170
  HPFESMSPEC procedure, 263
ESTIMATE statement
  HPFARIMASPEC procedure, 82
ESTMETHOD= option
  ARIMAX statement, 109, 163
EVENT statement
  HPFDIAGNOSE procedure, 117
EVENT= option
  PROC HPFENGINE statement, 191
EVENTBY= option
  PROC HPFDIAGNOSE statement, 104
EVENTCOMB statement
  HPFEVENTS procedure, 282
EVENTDATA statement
  HPFEVENTS procedure, 282
EVENTDEF statement
  HPFEVENTS procedure, 283
EVENTDUMMY statement
  HPFEVENTS procedure, 285
EVENTGROUP statement
  HPFEVENTS procedure, 286
EVENTKEY statement
  HPFEVENTS procedure, 287
EVENTMAP option
  SPECIFICATIONS statement (HPFSELECT), 471
EVENTS statement
  HPFREPOSITORY procedure, 451
EXCEPTIONS= option
  PROC HPFDIAGNOSE statement, 104
  PROC HPFENGINE statement, 190
EXM statement
  HPFEXMSPEC procedure, 319
EXMFUNC option
  SPECIFICATIONS statement (HPFSELECT), 472
EXMMAP option
  SPECIFICATIONS statement (HPFSELECT), 471
EXTEND= option
  CONTROL statement (ENG), 202
  INPUT statement (HPFORECAST), 352
EXTERNAL statement
  HPFENGINE procedure, 203

FORCEBACK option
  PROC HPFENGINE statement, 190
FORCECONSTRAINT option
  PROC HPFRECONCILE statement, 422
FORECAST statement
  HPF procedure, 36
  HPFARIMASPEC procedure, 79
  HPFDIAGNOSE procedure, 117
  HPFENGINE procedure, 204
  HPFORECAST procedure, 351
  HPFUCMSPEC procedure, 529

FORECASTOPTIONS statement
HPFSELECT procedure, 470
FORMAT= option
BENCHID statement (HPFTEMPRECON ), 496
ID statement (ENG), 206
ID statement (HPF), 41
ID statement (HPFEVENTS), 288
ID statement (HPFRECONCILE), 428
ID statement (HPFTEMPRECON ), 498
TIMEID statement (HPFORECAST), 361

GLOBALDIAGNOSE= option
PROC FORECAST statement, 347
GLOBALSELECTION= option
PROC FORECAST statement, 347
PROC HPFENGINE statement, 191
GRIDHOST= option
PERFORMANCE statement, 357
GRIDMODE= option
PERFORMANCE statement, 358
GRIDTIMEOUT= option
PERFORMANCE statement, 358

high-performance analytical procedures, PERFORMANCE statement
DETAILS option, 357
HOLDOUT= option
DIAGNOSE statement (PROC HPFDIAGSPEC), 169
FORECAST statement (HPF), 36
PROC FORECAST statement, 347
PROC HPFDIAGNOSE statement, 104
SELECT statement (HPFSELECT), 470
HOLDOUTPCT= option
DIAGNOSE statement (PROC HPFDIAGSPEC), 169
FORECAST statement (HPF), 37
PROC HPFDIAGNOSE statement, 104
PROC HPFORECAST statement , 348
SELECT statement (HPFSELECT), 470
HORIZONSTART= option
ID statement (ENG), 206
ID statement (HPFORECAST), 361
HORMISSPERCENT= option
COMBINE statement (DIAG), 113
COMBINE statement (HPFDIAGSPEC procedure), 165
COMBINE statement (HPFSELECT), 465
HOST= option
PERFORMANCE statement, 357
HPF, 26
HPF procedure, 30
syntax, 30
HPFDIAGNOSE procedure, PROC HPFDIAGNOSE statement
OUTPROCINFO= option, 105
HPFENGINE procedure, 185
syntax, 185
HPFESMSPEC procedure, 262
syntax, 262
HPFEVENTS procedure, 279
syntax, 279
HPFEXMSPEC, 317
HPFEXMSPEC procedure, 318
syntax, 318
HPFIDMSPEC, 261, 323
HPFIDMSPEC procedure, 324
syntax, 324
HPFORECAST procedure, 336, 342
DISTBY statement, 350
PERFORMANCE statement, 356
syntax, 342
HPFRECONCILE procedure
syntax, 419
HPFRECONCILE procedure, PROC HPFRECONCILE statement
AGGDATA= option, 422
AGGREGATE= option, 423
ALPHA= option, 423
BYVARSSORTED option, 427
CLMETHOD= option, 423
CONSTRAINT= option, 422
DIRECTION= option, 423
DISAGGDATA= option, 422
DISAGGREGATION= option, 424
ERRORTRACE= option, 423
FORCECONSTRAINT option, 422
IGNOREMISSF option, 424
LOCKZERO option, 424
MAXITER option, 424
OUTFOR= option, 422
OUTINFEASIBLE= option, 422
OUTPROCINFO= option, 422
PREDICTONLY option, 424
RECDIFF option, 423
SIGN= option, 424
STDDIFBD= option, 425
STDMETHOD= option, 425
WEIGHTED option, 425
HPFREPO, 447
HPFREPOSITORY procedure, 449
syntax, 449
HPFSELECT, 459
HPFSELECT procedure, 462
syntax, 462
HPFTEMPRECON procedure
syntax, 491
HPFTEMPRECON procedure, PROC HPFTEMPRECON statement

ACCDATA= option, 493
BENCHCONS= option, 494
BENCHDATA= option, 493
BENCHDIFF option, 493
BIASCORRECTION=option, 494
DATA= option, 493
MAXITER option, 494
OUTFOR= option, 493
OUTPROCINFO= option, 493
OUTSTAT= option, 494
PREDICTONLY option, 494
SCALEEXP= option, 495
SIGN= option, 495
SMOOTH= option, 495

ID statement
HPF procedure, 39
HPFDIAGNOSE procedure, 118
HPFENGINE procedure, 205
HPFEVENTS procedure, 287
HPFRECONCILE procedure, 427
HPFTEMPRECON procedure, 498
IDENTIFYORDER= option
ARIMAX statement, 110, 163
IDM statement
HPF procedure, 42
HPFDIAGNOSE procedure, 120
HPFDIAGSPEC procedure, 171
HPFIDMSPEC procedure, 326
IDMBASE= option
DIAGNOSE statement (HPFSELECT), 469
IGNORECHOOSE option
PROC HPFENGINE statement, 191
IGNOREMISSF option
PROC HPFRECONCILE statement, 424
IN= option
EVENTDATA statement, 282
INDEPENDENT statement
HPFORECAST procedure, 352
INEST= option
PROC HPFENGINE statement, 191
INEVENT= option
PROC HPFDIAGNOSE statement, 104
INPUT statement
HPFARIMASPEC procedure, 80
HPFDIAGNOSE procedure, 121
HPFENGINE procedure, 208
HPFORECAST procedure, 352
HPFUCMSPEC procedure, 530
INPUTMAP option
SPECIFICATIONS statement (HPFSELECT), 473
INPUTMISSINGPCT= option
DIAGNOSE statement (PROC HPFDIAGSPEC), 169
PROC HPFDIAGNOSE statement, 104
INSELECTNAME= option
PROC HPFDIAGNOSE statement, 105
PROC HPFSELECT statement, 463
INSTALL= option
PERFORMANCE statement, 358
INSTALLLOC= option
PERFORMANCE statement, 358
INTERMITTENT= option
DIAGNOSE statement (HPFSELECT), 469
FORECAST statement (HPF), 37
IDM statement, 120, 171
PROC HPFORECAST statement, 348
INTERVAL= option
BENCHID statement (HPFTEMPRECON ), 496
ID statement, 119
ID statement (HPF), 41
ID statement (HPFENGINE), 206
ID statement (HPFEVENTS), 288
ID statement (HPFRECONCILE), 428
ID statement (HPFTEMPRECON ), 498
IDM statement (HPF), 44
IDM statement (HPFIDMSPEC), 326
TIMEID statement (HPFORECAST), 361
IRREGULAR option
ID statement (HPFRECONCILE), 428
IRREGULAR statement
HPFUCMSPEC procedure, 531

KEEP= option
INPUT statement (HPFORECAST), 353

LABEL= option
EVENTCOMB statement (HPFEVENTS), 282, 286
EVENTDEF statement (HPFEVENTS), 283
PROC HPFEXMSPEC statement, 319
PROC HPFSELECT statement, 463
SPECIFICATIONS statement (HPFSELECT), 473
LABEL=option
PROC HPFESMSPEC statement, 263
LAGS= option
DEPLAG statement (HPFUCMSPEC), 529
LEAD= option
PROC HPF statement, 32
PROC HPFENGINE statement, 191
PROC HPFEVENTS statement, 281
PROC HPFORECAST statement, 348
LENGTH= option
SEASON statement (HPFUCMSPEC), 534
LEVEL statement

HPFORECAST procedure, 354
HPFUCMSPEC procedure, 532
LEVELPARM= option
ESM statement (HPFESMSPEC), 265
IDM statement (HPF), 45
IDM statement (HPFIDMSPEC), 328
LEVELREST= option
ESM statement (HPFESMSPEC), 265
IDM statement (HPF), 45
IDM statement (HPFIDMSPEC), 328
LOCKZERO option
PROC HPFRECONCILE statement, 424

MA= option
FORECAST statement (HPFARIMASPEC), 80
MAXERROR= option
PROC HPF statement, 32
PROC HPFEVENTS statement, 281
MAXIT= option
ESTIMATE statement (HPFARIMASPEC), 83
MAXITER option
PROC HPFRECONCILE statement, 424
PROC HPFTEMPRECON statement, 494
MAXITER= option
ESTIMATE statement (HPFARIMASPEC), 83
MEDIAN option
ESM statement (HPFESMSPEC), 265
EXM statement (HPFEXMSPEC), 320
FORECAST statement (HPF), 37
IDM statement (HPF), 45
IDM statement (HPFIDMSPEC), 328
METHOD= option
ARIMAX statement, 109, 163
COMBINE statement (DIAG), 113, 165
COMBINE statement (HPFSELECT), 465
ESM statement, 116, 170
ESM statement (HPFESMSPEC), 265
EXM statement (HPFEXMSPEC), 320
IDM statement, 120, 171
IDM statement (HPF), 45
IDM statement (HPFIDMSPEC), 329
METHOD=CLS option
ESTIMATE statement (HPFARIMASPEC), 82
METHOD=ML option
ESTIMATE statement (HPFARIMASPEC), 82
METHOD=ULS option
ESTIMATE statement (HPFARIMASPEC), 82
MINOBS=(SEASON= ) option
PROC HPFDIAGNOSE statement, 105
MINOBS=(TREND= ) option
PROC HPFDIAGNOSE statement, 105
Missing Values
Bottom-up Reconciliation (HPFRECONCILE), 433
Top-Down Reconciliation (HPFRECONCILE), 431
MISSING= option
BENCHID statement (HPFTEMPRECON ), 497
MISSMODE= option
COMBINE statement (DIAG), 115, 167
COMBINE statement (HPFSELECT), 467
MISSPERCENT= option
COMBINE statement (DIAG), 116, 167
COMBINE statement (HPFSELECT), 468
MODE= option
PERFORMANCE statement, 358
PROC HPFORECAST statement, 348
MODEL= option
FORECAST statement (HPF), 37
MODELREP= option
PROC HPFDIAGSPEC statement, 162
MODELREP=option
PROC HPFESMSPEC statement, 263
MODELREPOSITORY= option
PROC HPFDIAGSPEC statement, 162
PROC HPFORECAST statement, 348
MODELREPOSITORY=option
PROC HPFESMSPEC statement, 263
MU= option
FORECAST statement (HPFARIMASPEC), 80

NAME= option
PROC HPFEXMSPEC statement, 319
PROC HPFIDMSPEC statement, 325
PROC HPFSELECT statement, 463
NAME=option
PROC HPFESMSPEC statement, 263
NBACKCAST= option
FORECAST statement (HPF), 38
NBLOCKS= option
BLOCKSEASON statement (HPFUCMSPEC), 527
NLAGPCT= option
EXM statement (HPFEXMSPEC), 320
NNODES= option
PERFORMANCE statement, 358
NOCONSTANT option
FORECAST statement (HPFARIMASPEC), 80
NODES= option
PERFORMANCE statement, 358
NODIAGNOSE option
PROC HPFDIAGNOSE statement, 105
PROC HPFORECAST statement, 349
NOEST option
AUTOREG statement (HPFUCMSPEC), 526
BLOCKSEASON statement (HPFUCMSPEC), 527
CYCLE statement (HPFUCMSPEC), 528

DEPLAG statement (HPFUCMSPEC), 529
ESM statement (HPFESMSPEC), 266
ESTIMATE statement (HPFARIMASPEC), 83
EXM statement (HPFEXMSPEC), 320
IDM statement (HPF), 46
IDM statement (HPFIDMSPEC), 329
IRREGULAR statement (HPFUCMSPEC), 532
LEVEL statement (HPFUCMSPEC), 532
SEASON statement (HPFUCMSPEC), 534
SLOPE statement (HPFUCMSPEC), 535
NOFORECAST option
LEVEL statement (HPFORECAST), 355
NOINESTOPTS option
PROC HPFDIAGNOSE statement, 105
NOINT
ARIMAX statement, 111, 164
NOINT option
FORECAST statement (HPFARIMASPEC), 80
NOISEVAR= option
FORECAST statement (HPFARIMASPEC), 80
NOLS option
ESTIMATE statement (HPFARIMASPEC), 83
NOOUTALL option
OUTPUT statement (HPFORECAST), 356
PROC HPF statement, 32
NOREPLACE option
FORECAST statement (ENG), 204
NOSTABLE option
ESM statement (HPFESMSPEC), 266
ESTIMATE statement (HPFARIMASPEC), 83
IDM statement (HPF), 46
IDM statement (HPFIDMSPEC), 329
NOTSORTED option
ID statement (HPF), 41
NPARMS= option
EXM statement (HPFEXMSPEC), 320
NTHREADS= option
PERFORMANCE statement, 358
NUM= option
ARIMAX statement, 109, 164
INPUT statement (HPFARIMASPEC), 81
NZ= option
INPUT statement (HPFARIMASPEC), 82

OFFSET= option
BLOCKSEASON statement (HPFUCMSPEC), 527
OPERATION= option
ADJUST statement, 108
ADJUST statement (ENG), 200
OUT= option
EVENTDATA statement, 282
PROC HPF statement, 32
PROC HPFENGINE statement, 191
PROC HPFEVENTS statement, 286
OUTACCDATA= option
PROC HPFENGINE statement, 191
OUTCOMPONENT= option
PROC HPFENGINE statement, 191
OUTEST= option
OUTPUT statement (HPFORECAST), 355
PROC HPF statement, 33
PROC HPFDIAGNOSE statement, 105
PROC HPFENGINE statement, 191
OUTFOR= option
OUTPUT statement (HPFORECAST), 355
PROC HPF statement, 33
PROC HPFENGINE statement, 192
PROC HPFRECONCILE statement, 422
PROC HPFTEMPRECON statement, 493
OUTINDEP= option
OUTPUT statement (HPFORECAST), 356
PROC HPFENGINE statement, 192
OUTINFEASIBLE= option
PROC HPFRECONCILE statement, 422
OUTLIER= option
ARIMAX statement, 109
OUTLOG= option
OUTPUT statement (HPFORECAST), 355
OUTMODELINFO= option
OUTPUT statement (HPFORECAST), 356
PROC HPFENGINE statement, 192
OUTOUTLIER= option
PROC HPFDIAGNOSE statement, 105
OUTPROCINFO= option
PROC HPF statement, 33
PROC HPFENGINE statement, 192
PROC HPFDIAGNOSE statement, 105
PROC HPFRECONCILE statement, 422
PROC HPFTEMPRECON statement, 493
OUTPUT statement
HPFORECAST procedure, 355
OUTSEASON= option
PROC HPF statement, 33
OUTSTAT= option
OUTPUT statement (HPFORECAST), 355
PROC HPF statement, 33
PROC HPFENGINE statement, 192
PROC HPFTEMPRECON statement, 494
OUTSTATSELECT= option
OUTPUT statement (HPFORECAST), 356
PROC HPFENGINE statement, 192
OUTSUM= option
OUTPUT statement (HPFORECAST), 356
PROC HPF statement, 33
PROC HPFENGINE statement, 192
OUTTREND= option
PROC HPF statement, 33

OVERRIDE= option
PROC HPFORECAST statement, 349

P= option
ARIMAX statement, 109, 164
FORECAST statement (HPFARIMASPEC), 79
TRANSFORM statement, 122, 172
TREND statement, 123, 173
PERFORMANCE statement
HPFORECAST procedure, 356
PERIOD= option
CYCLE statement (HPFUCMSPEC), 528
EVENTDEF statement (HPFEVENTS), 284
PERROR= option
ARIMAX statement, 109, 164
PHI= option
DEPLAG statement (HPFUCMSPEC), 529
PLOT= option
PROC HPF statement, 33
PROC HPFENGINE statement, 192
PREDEFINED= option
INPUT statement (HPFARIMASPEC), 81
INPUT statement (HPFUCMSPEC), 530
PREDICTONLY option
PROC HPFRECONCILE statement, 424
PROC HPFTEMPRECON statement, 494
PREDICTOR statement
HPFORECAST procedure, 352
PREFILTER= option
PROC HPFDIAGNOSE statement, 105
PROC HPFDIAGSPEC statement, 169
PRINT= option
PROC HPF statement, 34
PROC HPFDIAGNOSE statement, 106
PROC HPFENGINE statement, 194
PROC HPFORECAST statement, 349
PROC HPFREPOSITORY statement, 450
PRINTDETAILS option
PROC HPF statement, 35
PROC HPFENGINE statement, 195
PROC HPFORECAST statement, 350
PROC HPFREPOSITORY statement, 450
PROC HPF statement, 32
PROC HPFARIMASPEC statement, 78
PROC HPFDIAG statement, 100
PROC HPFDIAGNOSE statement, 96, 100
PROC HPFDIAGSPEC statement, 159, 162
PROC HPFENGINE statement, 188
PROC HPFESMSPEC statement, 263
PROC HPFEVENTS statement, 281
PROC HPFEXMSPEC statement, 319
PROC HPFIDMSPEC statement, 325
PROC HPFORECAST statement, 344
PROC HPFRECONCILE statement, 422
PROC HPFREPOSITORY statement, 450
PROC HPFSELECT statement, 463
PROC HPFTEMPRECON statement, 493
PROC HPFUCMSPEC statement, 525
PULSE= option
EVENTDEF statement (HPFEVENTS), 284

Q= option
ARIMAX statement, 109, 164
FORECAST statement (HPFARIMASPEC), 79

RECDIFF option
PROC HPFRECONCILE statement, 423
RECONCILE statement
HPFTEMPRECON procedure, 500
REFINEPARMS= option
ARIMAX statement, 111, 164
UCM statement, 124
UCM statement (PROC HPFDIAGSPEC), 174
REP= option
PROC HPFDIAGSPEC statement, 162
REPLACEBACK option
FORECAST statement (HPF), 38
REPLACEMISSING option
CONTROL statement (ENG), 203
FORECAST statement (ENG), 204
FORECAST statement (HPF), 38
STOCHASTIC statement (ENG), 211
REPLACEMISSING= option
INPUT statement (HPFORECAST), 353
REPOSITORY= option
PROC HPFDIAGNOSE statement, 106
PROC HPFDIAGSPEC statement, 162
PROC HPFENGINE statement, 195
PROC HPFEXMSPEC statement, 319
PROC HPFIDMSPEC statement, 325
PROC HPFREPOSITORY statement, 450
PROC HPFSELECT statement, 464
REPOSITORY=option
PROC HPFESMSPEC statement, 263
REQUIRED=
INPUT statement (ENG), 208
STOCHASTIC statement (ENG), 210
REQUIRED= option
EVENT statement, 117
INPUT statement, 121
INPUT statement (HPFORECAST), 353
RETAINCHOOSE= option
PROC HPFDIAGNOSE statement, 106
RHO= option
CYCLE statement (HPFUCMSPEC), 528
RHO=option
AUTOREG statement (HPFUCMSPEC), 526
RULE option

EVENTCOMB statement (HPFEVENTS), 282
EVENTDEF statement (HPFEVENTS), 284

SCALEEXP=option
PROC HPFTEMPRECON statement, 495
SCORE statement
HPFENGINE procedure, 209
SCOREREPOSITORY= option
PROC HPFENGINE statement, 195
SDIFF= option
TREND statement, 123, 173
SEASON statement
HPFUCMSPEC procedure, 533
SEASONALITY= number
PROC HPFENGINE statement, 195
SEASONALITY= option
PROC HPF statement, 35
PROC HPFDIAGNOSE statement, 106
PROC HPFORECAST statement, 350
SEASONPARM= option
ESM statement (HPFESMSPEC), 266
SEASONREST= option
ESM statement (HPFESMSPEC), 266
SEASONTEST= option
DIAGNOSE statement (HPFSELECT), 469
FORECAST statement (HPF), 38
SELECT statement
HPFSELECT procedure, 470
SELECT= option
FORECAST statement (HPF), 36
IDM statement (HPF), 45
IDM statement (HPFIDMSPEC), 327
SELECT=option
ESM statement (HPFESMSPEC), 263
SELECTBASE= option
PROC HPFDIAGNOSE statement, 107
SELECTEVENT= option
PROC HPFDIAGNOSE statement, 106
SELECTINPUT= option
DIAGNOSE statement (PROC HPFDIAGSPEC), 169
PROC HPFDIAGNOSE statement, 106
SELECTION=
STOCHASTIC statement (ENG), 210
SELECTION= option
FORECAST statement (HPFORECAST), 351
INPUT statement (HPFORECAST), 354
PROC FORECAST statement, 347
SEMODE= option
COMBINE statement (HPFDIAGNOSE), 116
COMBINE statement (HPFDIAGSPEC), 168
COMBINE statement (HPFSELECT), 468
SET statement
HPFORECAST procedure, 359
SETMISSINF= option
INPUT statement (ENG), 209
SETMISSING= option
ADJUST statement, 108
ADJUST statement (ENG), 201
BENCHID statement (HPFTEMPRECON ), 496
CONTROL statement (ENG), 203
EXTERNAL statement (ENG), 203
FORECAST statement, 118
FORECAST statement (ENG), 204
FORECAST statement (HPF), 39
FORECAST statement (HPFORECAST), 351
ID statement, 119
ID statement (HPF), 41
ID statement (HPFENGINE), 207
ID statement (HPFEVENTS), 288
ID statement (HPFTEMPRECON ), 498
INPUT statement, 121
INPUT statement (HPFORECAST), 354
STOCHASTIC statement (ENG), 211
TIMEID statement (HPFORECAST), 362
SHIFT= option
EVENTDEF statement (HPFEVENTS), 285
SIGLEVEL= option
ARIMAX statement, 109, 164
DIAGNOSE statement (PROC HPFDIAGSPEC), 170
PROC HPFDIAGNOSE statement, 107
TRANSFORM statement, 122, 172
TREND statement, 123, 174
UCM statement, 124
UCM statement (PROC HPFDIAGSPEC), 175
SIGMA= option
EXM statement (HPFEXMSPEC), 320
SIGN= option
PROC HPFRECONCILE statement, 424
PROC HPFTEMPRECON statement, 495
SINGULAR= option
ESTIMATE statement (HPFARIMASPEC), 83
SIZE= option
IDM statement (HPF), 44
IDM statement (HPFIDMSPEC), 326
SLOPE statement
HPFUCMSPEC procedure, 534
SLOPE= option
EVENTDEF statement (HPFEVENTS), 283
SMOOTH=option
PROC HPFTEMPRECON statement, 495
SORTNAMES option
PROC HPF statement, 35
PROC HPFENGINE statement, 195
PROC HPFEVENTS statement, 281
SPECBASE= option
PROC HPFDIAGNOSE statement, 107

SPECIFICATION statement
HPFSELECT procedure, 471
SPECIFICATIONS statement
HPFREPOSITORY procedure, 451
SPECLABEL=option
PROC HPFESMSPEC statement, 263
SPECNAME=option
PROC HPFESMSPEC statement, 263
Standard Errors
Bottom-up Reconciliation (HPFRECONCILE), 433
Top-Down Reconciliation (HPFRECONCILE), 432
START= option
ID statement, 119
ID statement (ENG, 207
ID statement (HPF), 42
ID statement (HPFEVENTS), 289
ID statement (HPFRECONCILE), 428
ID statement (HPFTEMPRECON , 499
TIMEID statement (HPFORECAST), 362
START=option
BENCHID statement (HPFTEMPRECON , 497
STARTSUM= option
PROC HPF statement, 35
STDDIFBD= option
PROC HPFRECONCILE statement, 425
STDERR= option
COMBINE statement (HPFDIAGNOSE), 116
COMBINE statement (HPFDIAGSPEC), 168
COMBINE statement (HPFSELECT), 468
STDMETHOD= option
PROC HPFRECONCILE statement, 425
STOCHASTIC statement
HPFENGINE procedure, 210

TASK= option
PROC HPFENGINE statement (ENG), 195
TCPARM= option
EVENTDEF statement (HPFEVENTS), 285
TESTINPUT= option
DIAGNOSE statement (PROC HPFDIAGSPEC), 170
PROC HPFDIAGNOSE statement, 107
THREADS= option
PERFORMANCE statement, 358
TIMEOUT= option
PERFORMANCE statement, 358
TIMETIMEID statement
HPFORECAST procedure, 360
Top-Down Reconciliation
HPFRECONCILE procedure, 429
TRANSFORM statement
HPFDIAGNOSE procedure, 122
HPFDIAGSPEC procedure, 172
TRANSFORM= option
ESM statement (HPFESMSPEC), 266
EXM statement (HPFEXMSPEC), 320
FORECAST statement (HPF), 39
FORECAST statement (HPFARIMASPEC), 80
IDM statement, 120, 171
IDM statement (HPF), 46
IDM statement (HPFIDMSPEC), 329
INPUT statement (HPFARIMASPEC), 82
INPUT statement (HPFUCMSPEC), 530, 531
TRANSOPT= option
FORECAST statement (HPFARIMASPEC), 80
TRAVERSE statement
HPFREPOSITORY procedure, 451
TREND statement
HPFDIAGNOSE procedure, 122
HPFDIAGSPEC procedure, 173
TRENDPARM= option
ESM statement (HPFESMSPEC), 267
IDM statement (HPF), 46
IDM statement (HPFIDMSPEC), 329
TRENDREST= option
ESM statement (HPFESMSPEC), 267
IDM statement (HPF), 46
IDM statement (HPFIDMSPEC), 330
TRIMMISS= option
ADJUST statement, 108
ADJUST statement (ENG), 201
CONTROL statement (ENG), 203
EXTERNAL statement (ENG), 203
FORECAST statement, 118
FORECAST statement (ENG), 204, 207
ID statement, 119
INPUT statement, 121
INPUT statement (ENG), 209
STOCHASTIC statement (ENG), 211
VAR statement (HPFTEMPRECON ), 499
TRIMMISS=option
BENCHID statement (HPFTEMPRECON ), 497
TYPE= option
BLOCKSEASON statement (HPFUCMSPEC), 527
EVENTDEF statement (HPFEVENTS), 285
SEASON statement (HPFUCMSPEC), 534
TRANSFORM statement, 122, 172

UCM statement
HPFDIAGNOSE procedure, 123
HPFDIAGSPEC procedure, 174
USE= option
FORECAST statement (HPF), 39

VALIDATE option

PROC HPFSELECT statement, 464
VALUE= option
EVENTDEF statement (HPFEVENTS), 285
VAR statement
HPFEVENTS procedure, 289
VARIABLES statement
HPFREPOSITORY procedure, 452
VARIANCE= option
AUTOREG statement (HPFUCMSPEC), 526
BLOCKSEASON statement (HPFUCMSPEC), 527
CYCLE statement (HPFUCMSPEC), 528
IRREGULAR statement (HPFUCMSPEC), 532
LEVEL statement (HPFUCMSPEC), 533
SEASON statement (HPFUCMSPEC), 534
SLOPE statement (HPFUCMSPEC), 535

WEIGHTED option
PROC HPFRECONCILE statement, 425

ZEROMISS= option
ADJUST statement, 108
ADJUST statement (ENG), 201
CONTROL statement (ENG), 203
EXTERNAL statement (ENG), 204
FORECAST statement, 118
FORECAST statement (ENG), 205
FORECAST statement (HPF), 39
FORECAST statement (HPFORECAST), 351
ID statement, 119
INPUT statement, 121
INPUT statement (ENG), 209
INPUT statement (HPFORECAST), 354
STOCHASTIC statement (ENG), 211
ZEROMISSING= option
ID statement (ENG), 207
ID statement (PROC HPF), 42
TIMEID statement (HPFORECAST), 362
ZEROMISSING=option
ID statement (HPFTEMPRECON ), 499