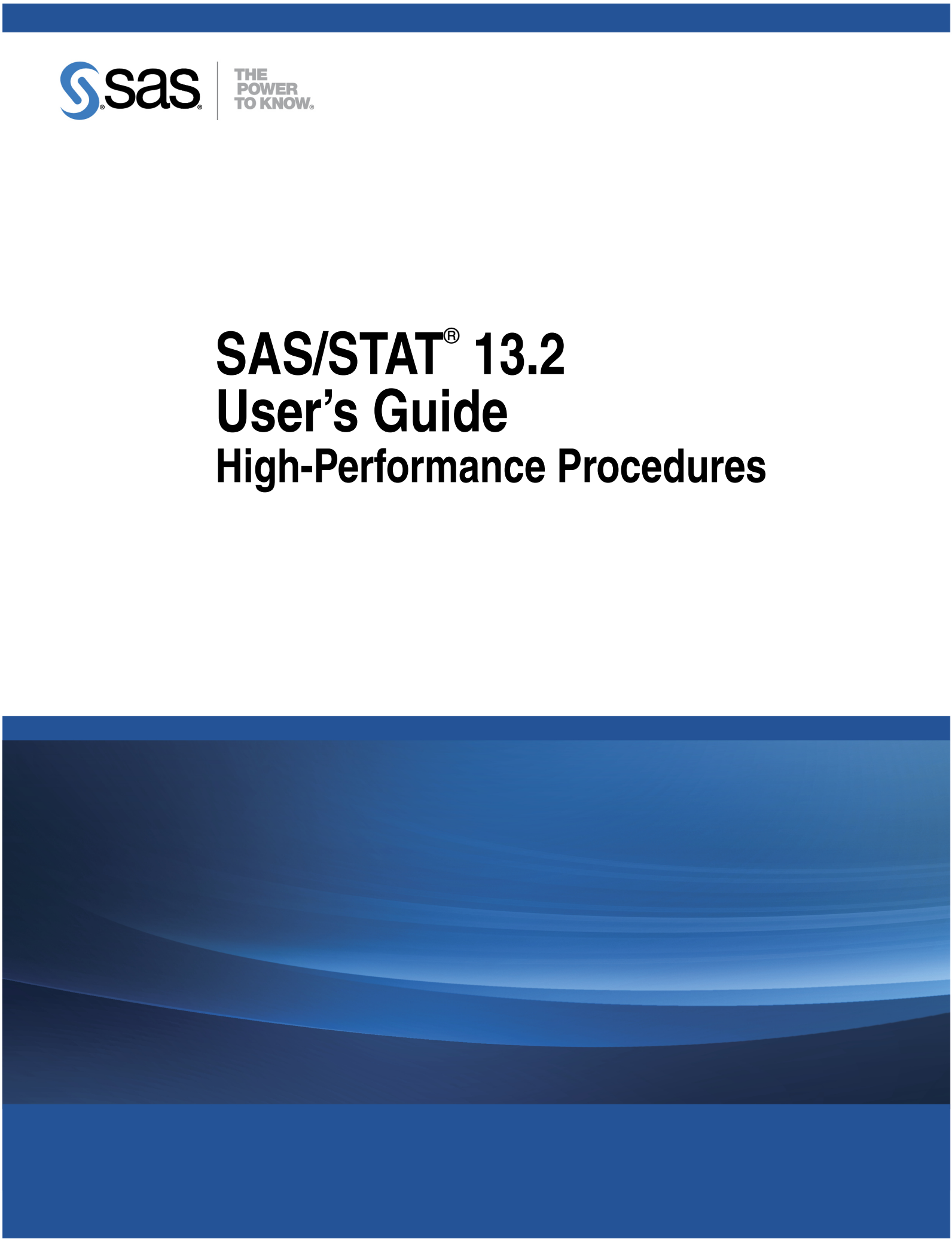


SAS/STAT® 13.2 User's Guide

High-Performance Procedures

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2014. *SAS/STAT® 13.2 User's Guide: High-Performance Procedures*. Cary, NC: SAS Institute Inc.

SAS/STAT® 13.2 User's Guide: High-Performance Procedures

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.

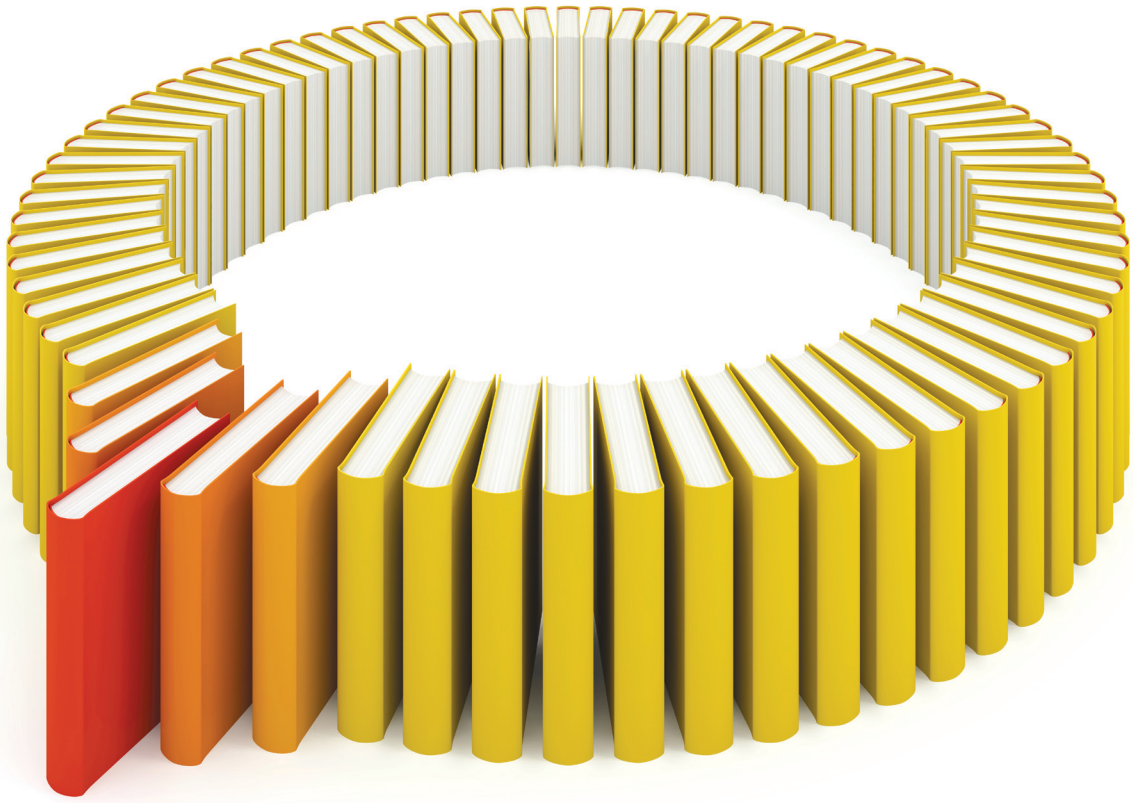
SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

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

Chapter 1. What's New in SAS High-Performance Statistics 13.2 1
Chapter 2. Introduction 5
Chapter 3. Shared Concepts and Topics 9
Chapter 4. Shared Statistical Concepts 41
Chapter 5. The HPCANDISC Procedure 73
Chapter 6. The HPFMM Procedure 105
Chapter 7. The HPGENSELECT Procedure 223
Chapter 8. The HPLMIXED Procedure 285
Chapter 9. The HPLOGISTIC Procedure 329
Chapter 10. The HPNLMOD Procedure 393
Chapter 11. The HPPLS Procedure 429
Chapter 12. The HPPRINCOMP Procedure 469
Chapter 13. The HPQUANTSELECT Procedure 493
Chapter 14. The HPREG Procedure 539
Chapter 15. The HPSPLIT Procedure 593

Subject Index **633**

Syntax Index **646**

Credits and Acknowledgments

Credits

Documentation

Editing	Anne Baxter, Ed Huddleston
Documentation Support	Tim Arnold

Software

The procedures in this book were implemented by the following members of the development staff. Program development includes design, programming, debugging, support, and documentation. In the following list, the names of the developers who currently provide primary support are listed first; other developers and previous developers are also listed.

HPCANDISC	Ning Kang
HPFMM	Biruk Gebremariam
HPGENSELECT	Gordon Johnston
HPLMIXED	Tianlin Wang, Biruk Gebremariam
HPLOGISTIC	Robert E. Derr, Oliver Schabenberger
HPNLMOD	Marc Kessler, Oliver Schabenberger
HPPLS	Ning Kang
HPPRINCOMP	Ning Kang
HPQUANTSELECT	Yonggang Yao
HPREG	Robert Cohen
HPSPLIT	Joseph Pingenot
High-performance computing foundation	Steve E. Krueger
High-performance analytics foundation	Robert Cohen, Georges H. Guirguis, Trevor Kearney, Richard Knight, Gang Meng, Scott Pope, Oliver Schabenberger, Charles Shorb, Tom P. Weber
Numerical routines	Georges H. Guirguis, Scott Pope

The following people contribute with their leadership and support: Chris Bailey, Robert Cohen, David Pope, Oliver Schabenberger, Renee Sciortino, Jonathan Wexler.

Testing

Shu An, Jack Berry, Tim Carter, Enzo D'Andreti, Girija Gavankar, Greg Goodwin, Dright Ho, Seungho Huh, Gerardo Hurtado, Cheryl LeSaint, Yu Liang, Jim McKenzie, Jim Metcalf, Huiping Miao, Bengt Pederson, Jaymie Shanahan, Fouad Younan

Internationalization Testing

Feng Gao, Alex (Wenqi) He, David Li, Frank (Jidong) Wang, Lina Xu

Technical Support

Phil Gibbs

Acknowledgments

Many people make significant and continuing contributions to the development of SAS software products.

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

Chapter 1

What's New in SAS High-Performance Statistics 13.2

Contents

Overview **1**
Infrastructure Enhancements **1**
BY Processing with Predistributed Data 1
GRIDDATASERVER= Option in the PERFORMANCE Statement No Longer Needed 2
GRIDMODE= Option in the PERFORMANCE Statement No Longer Needed 2
Multiple Data Access Modes Supported in a Single Procedure Step 2
Running Asymmetrically Alongside SAP HANA 2
New Procedures **2**
HPPLS Procedure 2
HPQUANTSELECT Procedure 2
Procedure Enhancements **3**
HPLOGISTIC Procedure 3
HPGENSELECT Procedure 3

Overview

SAS High-Performance Statistics 13.2 includes infrastructure enhancements, two new procedures, and enhancements to several procedures.

Infrastructure Enhancements

BY Processing with Predistributed Data

BY processing support that is available in selected high-performance analytical procedures is now also provided for data that are predistributed in databases, Hadoop, SAS LASR Analytic Servers, and SAP HANA.

GRIDDATASERVER= Option in the PERFORMANCE Statement No Longer Needed

When you run high-performance analytical procedures alongside databases, you no longer need to specify the GRIDDATASERVER= option in the PERFORMANCE statement or set the GRIDDATASERVER environment variable. Any values that you specify are ignored.

GRIDMODE= Option in the PERFORMANCE Statement No Longer Needed

With one exception, which is described in the GRIDMODE= option section of the PERFORMANCE statement documentation in Chapter 3, "Shared Concepts and Topics," you no longer need to specify the GRIDMODE= option in the PERFORMANCE statement or set the GRIDMODE environment variable. Any values that you specify are ignored.

Multiple Data Access Modes Supported in a Single Procedure Step

The modes in which data sets are accessed are determined automatically for each data set that is used in a single procedure step.

Running Asymmetrically Alongside SAP HANA

High-performance analytical procedures can now read data from and write data to SAP HANA asymmetrically in parallel.

New Procedures

HPPLS Procedure

The new HPPLS procedure fits models by using any of several linear predictive methods, including partial least squares (PLS), to optimally address one or both of these two goals: explaining response variation and explaining predictor variation.

HPQUANTSELECT Procedure

The new HPQUANTSELECT procedure performs high-performance quantile regression analysis. PROC HPQUANTSELECT not only fits quantile regression models but also offers extensive capabilities for quantile regression model selection, and it supports statistical inferences with or without the assumption of independently and identically distributed (iid) errors.

Procedure Enhancements

HPLOGISTIC Procedure

- The PARTITION statement divides the observations in the input data set into disjoint subsets for model training, validation, and testing. Various fit statistics are displayed in the new "Partition Fit Statistics" table.
- The new CUTPOINT= option in the MODEL statement enables you to control the classification of events and nonevents.
- The new CHOOSE=VALIDATE and STOP=VALIDATE options in the SELECTION statement use the validation data set during the selection process.
- The AIC, BIC, and AICC criteria are added to the SELECT= option in the SELECTION statement.
- The INEST=option in the PROC HPLOGISTIC statement enables you to input your own starting values for the optimization. The OUTEST option adds a column that contains the parameter names to the "Parameter Estimates" ODS OUTPUT data set.
- The CTABLE option in the MODEL statement creates data for receiver operating characteristic (ROC) curves. The PRIOR= option in the MODEL statement specifies population prevalences that are used to adjust statistics displayed by the CTABLE option and by the PARTITION statement.
- The POST keyword in the OUTPUT statement outputs the posterior probabilities, and the ROLE keyword outputs the partition to which the observation is assigned.

HPGENSELECT Procedure

The PARTITION statement specifies how observations in the input data set are to be logically partitioned into disjoint subsets for model training, validation, and testing. Models are fit and selected based on the training data. You can use the validation and test sets to assess how the selected model generalizes on data that played no role in selecting the model.

Chapter 2
Introduction

Contents

Overview of SAS/STAT High-Performance Procedures **5**
About This Book **5**
Chapter Organization 6
Typographical Conventions 6
Options Used in Examples 7
Online Documentation **7**
SAS Technical Support Services **7**

Overview of SAS/STAT High-Performance Procedures

SAS/STAT high-performance procedures provide predictive modeling tools that have been specially developed to take advantage of parallel processing in both multithreaded single-machine mode and distributed multiple-machine mode. Predictive modeling methods include regression, logistic regression, generalized linear models, linear mixed models, nonlinear models, and decision trees. The procedures provide model selection, dimension reduction, and identification of important variables whenever this is appropriate for the analysis.

In addition to the high-performance statistical procedures described in this book, SAS/STAT includes high-performance utility procedures, which are described in *Base SAS Procedures Guide: High-Performance Procedures*.

You can run the high-performance statistical procedures in single-machine mode without licensing SAS High-Performance Statistics. However, to run these procedures in distributed mode, you must license SAS High-Performance Statistics and you must install and configure the SAS High-Performance Analytics infrastructure. For more information, see the *SAS High-Performance Analytics Infrastructure: Installation and Configuration Guide*.

About This Book

This book assumes that you are familiar with Base SAS software and with the books *SAS Language Reference: Concepts* and *Base SAS Procedures Guide*. It also assumes that you are familiar with basic SAS System concepts, such as using the DATA step to create SAS data sets and using Base SAS procedures (such as the PRINT and SORT procedures) to manipulate SAS data sets.

Chapter Organization

This book is organized as follows:

Chapter 2, this chapter, provides an overview of SAS/STAT high-performance procedures.

Chapter 3, "Shared Concepts and Topics," describes the modes in which SAS/STAT high-performance procedures can execute.

Chapter 4, "Shared Statistical Concepts," describes common syntax elements that are supported by SAS/STAT high-performance procedures.

Subsequent chapters describe the individual procedures. These chapters appear in alphabetical order by procedure name. Each chapter is organized as follows:

- The "Overview" section provides a brief description of the analysis provided by the procedure.
- The "Getting Started" section provides a quick introduction to the procedure through a simple example.
- The "Syntax" section describes the SAS statements and options that control the procedure.
- The "Details" section discusses methodology and other topics, such as ODS tables.
- The "Examples" section contains examples that use the procedure.
- The "References" section contains references for the methodology.

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 in the syntax definitions and in text to represent arguments for which you supply a value.
VariableName	is used for the names of variables and data sets when they appear in the text.
bold	is used to for matrices and vectors.
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 code.

Options Used in Examples

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

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

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

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

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

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

Online Documentation

This documentation is available online with the SAS System. To access documentation for the SAS/STAT high-performance procedures from the SAS windowing environment, select **Help** from the main menu and then select **SAS Help and Documentation**. On the **Contents** tab, expand the **SAS Products**, **SAS/STAT**, and **SAS/STAT User's Guide: High-Performance Procedures** items. Then expand chapters and click on sections. You can search the documentation by using the **Search** tab.

You can also access the documentation by going to `http://support.sas.com/documentation`.

SAS Technical Support Services

The SAS Technical Support staff is available to respond to problems and answer technical questions regarding the use of high-performance procedures. Go to `http://support.sas.com/techsup` for more information.

Chapter 3
Shared Concepts and Topics

Contents

Overview **10**
Processing Modes **10**
Single-Machine Mode 10
Distributed Mode 10
Controlling the Execution Mode with Environment Variables and Performance Statement Options 11
Determining Single-Machine Mode or Distributed Mode 12
Data Access Modes **15**
Single-Machine Data Access Mode 15
Distributed Data Access Mode 15
Determining the Data Access Mode 16
Alongside-the-Database Execution **18**
Alongside-LASR Distributed Execution **20**
Running High-Performance Analytical Procedures Alongside a SAS LASR Analytic Server in Distributed Mode **20**
Starting a SAS LASR Analytic Server Instance 21
Associating a SAS Libref with the SAS LASR Analytic Server Instance 21
Running a High-Performance Analytical Procedure Alongside the SAS LASR Analytic Server Instance 22
Terminating a SAS LASR Analytic Server Instance 23
Alongside-LASR Distributed Execution on a Subset of the Appliance Nodes **23**
Running High-Performance Analytical Procedures in Asymmetric Mode **23**
Running in Symmetric Mode 24
Running in Asymmetric Mode on One Appliance 25
Running in Asymmetric Mode on Distinct Appliances 26
Alongside-HDFS Execution **29**
Alongside-HDFS Execution by Using the SASHDAT Engine 29
Alongside-HDFS Execution by Using the Hadoop Engine 31
Output Data Sets **34**
Working with Formats **35**
PERFORMANCE Statement **36**

Overview

This chapter describes the modes of execution in which SAS high-performance analytical procedures can execute. If you have SAS/STAT installed, you can run any procedure in this book on a single machine. However, to run procedures in this book in distributed mode, you must also have SAS High-Performance Statistics software installed. For more information about these modes, see the next section.

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

Processing Modes

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. In this book, single-machine mode refers to an application running multiple concurrent threads on a multicore machine in order to take advantage of parallel execution on multiple processing units. More simply, single-machine mode for high-performance analytical procedures means multithreading on the client machine.

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

Distributed Mode

Distributed mode is a computing model in which several nodes in a distributed computing environment participate in the calculations. In this book, the distributed mode of a high-performance analytical procedure refers to the procedure performing 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

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 in high-performance analytical procedures, 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 SAS High-Performance Statistics software connects to run in distributed mode.
- *installation location* identifies the directory where the SAS High-Performance Statistics software is installed on the appliance.

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

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

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

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

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.

The key variable that determines whether a high-performance analytical procedure executes in single-machine or distributed mode is the *grid host*. The installation location is 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 has been set by your system administrator.

The following sets of SAS statements are functionally equivalent:

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

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

Determining Single-Machine Mode or Distributed Mode

High-performance analytical procedures use the following rules to determine whether they run 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.

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

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

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

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

Figure 3.1 shows the results from the analysis.

Figure 3.1 Results from Logistic Regression in Single-Machine Mode

The HPLOGISTIC Procedure

Performance Information	
Execution Mode	Single-Machine
Number of Threads	4

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

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

Parameter Estimates

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

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

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

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

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

Figure 3.2 Results from Logistic Regression in Distributed Mode

The HPLOGISTIC Procedure

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

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

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

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

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

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

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

Figure 3.3 Single-Machine Mode Despite Host Specification

The HPLOGISTIC Procedure

Performance Information	
Execution Mode	Single-Machine
Number of Threads	4

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

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

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

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

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.

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

Data Access Modes

Single-Machine Data Access Mode

When high-performance analytical procedures run in single-machine mode, they access data in the same way as traditional SAS procedures. They use Base SAS to access input and output SAS data sets on the client machine, and they use the relevant SAS/ACCESS interface to bring data from other sources, such as third-party databases, Hadoop, and SAS LASR servers, to the client.

Distributed Data Access Mode

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

- Client-data (local-data) mode: The input and output data for the analytic task are stored on the client machine where the high-performance procedure is invoked. When the procedure runs, the SAS High-Performance Analytics infrastructure sends input data from the client to the distributed computing environment and sends output data from the distributed computing environment to the client.
- Parallel symmetric mode: Input and output data are stored on the same nodes that are used for the distributed computation, and the data move in parallel from the data store to the computational nodes without crossing node boundaries. Parallel symmetric mode is available with the following distributed data sources:
  - Data in Greenplum or Teradata databases that are collocated with the computational nodes. This access mode is also called alongside-the-database mode. For more information, see the section "Alongside-the-Database Execution" on page 18.

  - Data in SASHDAT format in the Hadoop Distributed File System (HDFS) that is collocated with the computational nodes. This access mode is also called alongside-HDFS mode. For more information, see the section "Alongside-HDFS Execution by Using the SASHDAT Engine" on page 29.
  - Data in a SAS LASR Analytic Server that is collocated with the computational nodes. This access mode is also called alongside-LASR mode. For more information, see the section "Running High-Performance Analytical Procedures Alongside a SAS LASR Analytic Server in Distributed Mode" on page 20.

- Parallel asymmetric mode: The primary reason for providing this mode is to enable you to manage and house data on appliances (the data appliances) and to run high-performance analytical procedures on a different appliance (the computing appliance). The high-performance analytical procedures run in a SAS process on the computing appliance. For each data source that is accessed in parallel asymmetric mode, a SAS Embedded Process must run on the associated 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 mode is called asymmetric mode because the number of nodes on the data appliance does not need to match the number of nodes on the computing appliance. Parallel asymmetric mode is supported for data in Teradata, Greenplum, and Oracle databases and for data in HDFS and SAP HANA. In these cases, the parallel asymmetric access is somewhat loosely described as being asymmetric alongside access, even though the data storage and computation can occur on different appliances. For more information, see the section "Running High-Performance Analytical Procedures in Asymmetric Mode" on page 23.

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

Determining the Data Access Mode

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

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

The second reason why parallel data access might not be possible for a particular data set is that the required driver software might not be installed on the compute nodes. In this case, the required data feeder that

moves the data from the compute nodes to the data source cannot be successfully loaded, and a note like the following is reported in the SAS log:

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

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

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

For data in Teradata databases, parallel asymmetric mode is used by default. If your data and computation are collocated on the same appliance and you want to use parallel symmetric mode, then you need to specify GRIDMODE=SYM in the PERFORMANCE statement or you need to set the GRIDMODE='SYM' environment variable by using an OPTION SET statement.

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

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

proc hpds2 data=hdatlib.neuralgia out=neuralgia;
    performance host='hpa.sas.com';
    data DS2GTF.out;
       method run();
      set DS2GTF.in;
    end;
   enddata;
 run;
```

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

Figure 3.4 Performance Information and Data Access Information Tables

The HPDS2 Procedure

Performance Information	
Host Node	hpa.sas.com
Execution Mode	Distributed
Number of Compute Nodes	13
Number of Threads per Node	24

Figure 3.4 *continued*

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

Alongside-the-Database Execution

High-performance analytical procedures interface 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, high-performance analytical procedures create a distributed computing environment in which an analytic process is collocated 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 high-performance analytical procedures if execution is alongside the database.

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

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


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

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

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

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

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

The subtle differences are as follows:

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

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

Figure 3.5 Alongside-the-Database Execution on Greenplum

The HPLOGISTIC Procedure

Performance Information	
Host Node	green.sas.com
Execution Mode	Distributed
Number of Compute Nodes	16
Number of Threads per Node	24

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

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

Figure 3.5 *continued*

Parameter Estimates

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

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

This section provides an example of steps that you can use to start and load data into a SAS LASR Analytic Server instance and then run high-performance analytical procedures 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 simData data set that is used in the preceding examples. 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.

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

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 DATA= option specifies the name of a data set that is loaded into this LASR Analytic Server instance. (You do not need to specify the DATA= option at this time because you can add tables to the LASR Analytic Server instance at any stage of its life.) For more information about starting and using a LASR Analytic Server, see the *SAS LASR Analytic Server: Reference Guide*.

The NODES=ALL option in the PERFORMANCE statement specifies 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 23.

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

Figure 3.6 Performance and Data Access Information

The LASR Procedure

Performance Information	
Host Node	hpa.sas.com
Execution Mode	Distributed
Number of Compute Nodes	13

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

Associating a SAS Libref with the SAS LASR Analytic Server Instance

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

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

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

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

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

```
proc hplogistic data=MyLasr.simData;
   class a b c;
   model y = a b c x1 x2 x3;
   output out=MyLasr.simulateScores pred=PredictedProbabliity;
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 HPLOGISTIC step runs in distributed mode alongside the LASR Analytic Server, as indicated in the "Performance Information" table shown in Figure 3.7.

Figure 3.7 Performance and Data Access Information

The HPLOGISTIC Procedure

Performance Information	
Host Node	hpa.sas.com
Execution Mode	Distributed
Number of Compute Nodes	13
Number of Threads per Node	24

Data Access Information

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

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

The preceding OUTPUT statement creates an output table that is added to the LASR Analytic Server instance. 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 set. However, in order for the output data to be created in parallel symmetric mode, all the nodes that are used by the server instance that holds the input data must also be used by the server 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=54545;
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 high-performance analytical procedure 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 a high-performance analytical 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 High-Performance Analytical Procedures in Asymmetric Mode

This section provides examples of how you can run high-performance analytical procedures in asymmetric mode. It also includes examples that run in symmetric mode to highlight differences between the modes. For a description of asymmetric mode, see the section "Distributed Data Access Mode" on page 15.

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

The following examples use a 24-node data appliance named "data_appliance.sas.com," which houses a Teradata DBMS and has a SAS Embedded Process installed. Because SAS High-Performance Statistics is also installed on this appliance, it can be used to run high-performance analytical procedures in both symmetric and asymmetric modes.

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

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

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

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

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

Running in Symmetric Mode

The following statements run the HPLOGISTIC procedure in symmetric mode on the data appliance:

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

Figure 3.8 shows the results of this analysis.

Figure 3.8 Alongside-the-Database Execution in Symmetric Mode on Teradata

The HPLOGISTIC Procedure

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

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

Figure 3.8 *continued*

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

Parameter Estimates

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

The "Performance Information" table shows that the execution occurs in symmetric mode on the 24 nodes of the data appliance. In this case, the NODES=10 option in the PERFORMANCE statement is ignored because the number of nodes that are used is determined by the number of nodes across which the data are distributed, as indicated in the following warning message in the SAS log:

```
WARNING: The NODES=10 option in the PERFORMANCE statement is ignored because
         you are running alongside the distributed data source
         DATALIB.simData.DATA. The number of compute nodes is determined by the
         configuration of the distributed DBMS.
```

Running in Asymmetric Mode on One Appliance

By default, the HPLOGISTIC procedure runs in asymmetric mode, as shown Figure 3.9, which is produced by the following statements:

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

Figure 3.9 Alongside-Teradata Execution in Asymmetric Mode

The HPLOGISTIC Procedure

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

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

The "Performance Information" table confirms that the NODES=10 option that you specified in the PERFORMANCE statement was honored, and the "Data Access Information" table shows that the data were accessed in parallel asymmetric mode. The data were moved in parallel from the 24 nodes on which the data were stored to the 10 nodes on which the execution occurred. The numeric results are not reproduced here, but they agree with the previous analyses.

Running in Asymmetric Mode on Distinct Appliances

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

The advantage of using different computing and data appliances is that the data appliance is not affected by the execution of high-performance analytical procedures 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. The following statements show how this is done:

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

Figure 3.10 shows the "Performance Information" and "Data Access Information" tables.

Figure 3.10 Asymmetric Mode with Distinct Data and Computing Appliances

The HPLOGISTIC Procedure

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

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

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

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

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

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

Figure 3.11 show the "Performance Information" and "Data Access Information" tables.

Figure 3.11 PROC LASR Running in Asymmetric Mode

The LASR Procedure

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

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

By default, all the nodes on the computing appliance would be used. However, because NODES=30 was specified in the PERFORMANCE statement, PROC LASR ran on only 30 nodes of the computing appliance.

The data were loaded asymmetrically in parallel from the 24 data appliance nodes to the 30 compute nodes on which PROC LASR ran.

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

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

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

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

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

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

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

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

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

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

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

If you configured Hadoop on the computing appliance, then you can create output data tables that are stored in the HDFS on the computing appliance. You can do this by using the SASHDAT engine as described in the section “Alongside-HDFS Execution” on page 29.

Alongside-HDFS Execution

Running high-performance analytical procedures alongside HDFS shares many features with running alongside the database. You can execute high-performance analytical procedures 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 enables you to access files in SASHDAT format without supplying any additional metadata. Additionally, you can also use the SASHDAT engine to read data in CSV (comma-separated value) format, but you need 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 collocated 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 to HDFS the simData data set that was used in the previous two sections:

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

libname hdatLib sashdat
        path="/hps";

 data hdatLib.simData (replace = yes) ;
     set simData;
 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 simData 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 HPLOGISTIC procedure statements perform the analysis in alongside-HDFS mode. These statements are almost identical to the PROC HPLOGISTIC example in the previous two sections, which executed in single-machine mode and alongside-the-database distributed mode, respectively.

Figure 3.12 shows the "Performance Information" and "Data Access Information" tables. You see that the procedure ran in distributed mode and that the input data were read in parallel symmetric mode. The numeric results shown in Figure 3.13 agree with the previous analyses shown in Figure 3.1, Figure 3.2, and Figure 3.5.

Figure 3.12 Alongside-HDFS Execution Performance Information

The HPLOGISTIC Procedure

Performance Information	
Host Node	hpa.sas.com
Execution Mode	Distributed
Number of Compute Nodes	13
Number of Threads per Node	24

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

Figure 3.13 Alongside-HDFS Execution Model Information

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

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

Alongside-HDFS Execution by Using the Hadoop Engine

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

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

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

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

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

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

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

Figure 3.14 shows the "Performance Information" and "Data Access Information" tables. You see that the procedure ran in distributed mode and that the input data were read in parallel asymmetric mode. The numeric results shown in Figure 3.15 agree with the previous analyses.

Figure 3.14 Alongside-HDFS Execution by Using the Hadoop Engine

The HPLOGISTIC Procedure

Performance Information	
Host Node	compute_appliance.sas.com
Execution Mode	Distributed
Number of Compute Nodes	142
Number of Threads per Node	32

Figure 3.14 *continued*

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

Figure 3.15 Alongside-HDFS Execution by Using the Hadoop Engine

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

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

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

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

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

The following example shows how you can use PROC HDMD to register metadata for CSV data independently from Hive and then analyze these data by using high-performance analytical procedures. The CSV data in the

table csvExample.csv is stored in HDFS in the directory /user/demo/data. Each record in this table consists of the following fields, in the order shown and separated by commas.

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

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

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

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

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

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

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

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

Figure 3.16 shows the results of this analysis. You see that the procedure ran in distributed mode and that the input data were read in parallel asymmetric mode. The metadata that you created by using the HDMD procedure have been used successfully in executing this analysis.

Figure 3.16 Alongside-HDFS Execution with CSV Data

The HPLOGISTIC Procedure

Performance Information	
Host Node	compute_appliance.sas.com
Execution Mode	Distributed
Number of Compute Nodes	142
Number of Threads per Node	32

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

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

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

Number of Observations Read	1000
Number of Observations Used	1000

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

Output Data Sets

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

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

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

```
   output out=applianc.simData_out pred=p;
   performance host="hpa.sas.com" nodes=8;
run;
```

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

When a high-performance analytical procedure 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.

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

- variables that are explicitly created by the statement
- variables that are listed in the ID statement, as described in Chapter 4, "Shared Statistical Concepts"
- distribution keys or hash keys that are transferred from the input data set

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

Working with Formats

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

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

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

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

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

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


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

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

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

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

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

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 a high-performance analytical procedure.

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

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

COMMIT=*n*

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

High-performance analytical procedures do 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).

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

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

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

High-performance analytical procedures send the 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.

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

The only time you need to specify this option is when you want to run in symmetric mode alongside a Teradata database. In all other cases, this option is not needed and the value that you specify is ignored.

GRIDTIMEOUT=*s*

TIMEOUT=*s*
specifies the time-out in seconds for a high-performance analytical 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 high-performance analytical procedure are installed on the appliance. Specifying the INSTALL= option overrides the GRIDINSTALLLOC environment variable.

LASRSERVER=*"path"*

LASR=*"path"*

specifies the fully qualified path to the description file of a SAS LASR Analytic Server instance. If the input data set is held in memory by this LASR Analytic Server instance, then the procedure runs alongside LASR. This option is not needed to run alongside LASR if the DATA= specification of the input data uses a libref that is associated with a LASR Analytic Server instance. For more information, see the section "Alongside-LASR Distributed Execution" on page 20 and the *SAS LASR Analytic Server: Reference Guide*.

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 high-performance analytical procedures then perform the analysis on the client. For example, the following sets of statements are equivalent:

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

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

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. For example, the following statements perform the analysis in distributed mode by using 10 units of work on the appliance that is identified in the HOST= option:

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

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

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

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

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. For example, if you are running alongside an appliance with 24 nodes, the NODES= option in the following statements is ignored:

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

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 algorithm by which a CPU count is converted to a thread count is specific to the high-performance analytical procedure. Most procedures create one thread per CPU for the analytic computations.

By default, high-performance analytical procedures run 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. Individual high-performance analytical procedures can impose more stringent limits if called for by algorithmic considerations.

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

Chapter 4
Shared Statistical Concepts

Contents

Common Features of SAS High-Performance Statistical Procedures **42**
Syntax Common to SAS High-Performance Statistical Procedures **42**
CLASS Statement 42
FREQ Statement 46
ID Statement 46
SELECTION Statement 47
VAR Statement 52
WEIGHT Statement 52
Levelization of Classification Variables **52**
Specification and Parameterization of Model Effects **55**
Effect Operators 56
Bar and At Sign Operators 56
Colon, Dash, and Double Dash Operators 57
GLM Parameterization of Classification Variables and Effects 58
Intercept 59
Regression Effects 59
Main Effects 59
Interaction Effects 59
Nested Effects 60
Continuous-Nesting-Class Effects 61
Continuous-by-Class Effects 61
General Effects 61
Reference Parameterization 62
Model Selection **63**
Methods 63
Full Model Fitted 63
Forward Selection 63
Backward Elimination 66
Stepwise Selection 67
Forward-Swap Selection 68
Least Angle Regression 69
Lasso Selection 70
Adaptive Lasso Selection 70
References **71**

Common Features of SAS High-Performance Statistical Procedures

SAS high-performance statistical procedures behave in many ways like other procedures in the SAS System. This chapter provides details about and describes common syntax elements that are supported by many high-performance statistical procedures. Any deviation by a high-performance statistical procedure from the common syntax is documented in the specific chapter for the procedure.

Syntax Common to SAS High-Performance Statistical Procedures

CLASS Statement

CLASS *variable* < **(***options***)** >. . . < *variable* < **(***options***)** > > < / *global-options* > **;**

The CLASS statement names the classification variables to be used as explanatory variables in the analysis. These variables enter the analysis not through their values, but through levels to which the unique values are mapped. For more information about these mappings, see the section "Levelization of Classification Variables" on page 52.

If a CLASS statement is specified, it must precede the MODEL statement in high-performance statistical procedures that support a MODEL statement.

If the procedure permits a classification variable as a response (dependent variable or target), the response does not need to be specified in the CLASS statement.

You can specify options either as individual variable *options* or as *global-options*. You can specify *options* for each variable by enclosing the options in parentheses after the variable name. You can also specify *global-options* for the CLASS statement by placing them after a slash (/). *Global-options* are applied to all the variables that are specified in the CLASS statement. If you specify more than one CLASS statement, the *global-options* that are specified in any one CLASS statement apply to all CLASS statements. However, individual CLASS variable *options* override the *global-options*.

You can specify the following values for either an *option* or a *global-option* (except for the HPLMIXED procedure, which does not support options in this statement):

DESCENDING

DESC

reverses the sort order of the classification variable. If both the DESCENDING and ORDER= options are specified, high-performance statistical procedures order the categories according to the ORDER= option and then reverse that order.

ORDER=DATA | FORMATTED | INTERNAL

ORDER=FREQ | FREQDATA | FREQFORMATTED | FREQINTERNAL

specifies the sort order for the levels of classification variables. This ordering determines which parameters in the model correspond to each level in the data. By default, ORDER=FORMATTED. For ORDER=FORMATTED and ORDER=INTERNAL, the sort order is machine-dependent. When ORDER=FORMATTED is in effect for numeric variables for which you have supplied no explicit format, the levels are ordered by their internal values.

The following table shows how high-performance statistical procedures interpret values of the ORDER= option.

Value of ORDER=	Levels Sorted By
DATA	Order of appearance in the input data set
FORMATTED	External formatted values, except for numeric variables that have no explicit format, which are sorted by their unformatted (internal) values
FREQ	Descending frequency count (levels that have more observations come earlier in the order)
FREQDATA	Order of descending frequency count, and within counts by order of appearance in the input data set when counts are tied
FREQFORMATTED	Order of descending frequency count, and within counts by formatted value when counts are tied
FREQINTERNAL	Order of descending frequency count, and within counts by unformatted (internal) value when counts are tied
INTERNAL	Unformatted value

For more information about sort order, see the chapter about the SORT procedure in *Base SAS Procedures Guide* and the discussion of BY-group processing in *SAS Language Reference: Concepts*.

REF=*'level'* | *keyword*

REFERENCE=*'level'* | *keyword*

specifies the reference level that is used when you specify PARAM=REFERENCE. For an individual (but not a global) variable REF= *option*, you can specify the *level* of the variable to use as the reference level. Specify the formatted value of the variable if a format is assigned. For a REF= *option* or *global-option*, you can use one of the following *keywords*. The default is REF=LAST.

FIRST designates the first ordered level as reference.

LAST designates the last ordered level as reference.

If you choose a reference level for any CLASS variable, all variables are parameterized in the reference parameterization for computational efficiency. In other words, high-performance statistical procedures apply a single parameterization method to all classification variables.

Suppose that the variable temp has three levels (`'hot'`, `'warm'`, and `'cold'`) and that the variable gender has two levels (`'M'` and `'F'`). The following statements fit a logistic regression model:

```
proc hplogistic;
   class gender(ref='F') temp;
   model y = gender gender*temp;
run;
```

Both CLASS variables are in reference parameterization in this model. The reference levels are `'F'` for the variable gender and `'warm'` for the variable temp, because the statements are equivalent to the following statements:

```
proc hplogistic;
   class gender(ref='F') temp(ref=last);
   model y = gender gender*temp;
run;
```

SPLIT

requests that the columns of the design matrix that correspond to any effect that contains a split classification variable can be selected to enter or leave a model independently of the other design columns of that effect. This option is specific to the HPREG procedure.

Suppose that the variable temp has three levels (`'hot'`, `'warm'`, and `'cold'`), that the variable gender has two levels (`'M'` and `'F'`), and that the variables are used in a PROC HPREG run as follows:

```
proc hpreg;
   class temp gender / split;
   model y = gender gender*temp;
run;
```

The two effects in the MODEL statement are split into eight independent effects. The effect “gender” is split into two effects that are labeled “gender_M” and “gender_F”. The effect “gender*temp” is split into six effects that are labeled “gender_M*temp_hot”, “gender_F*temp_hot”, “gender_M*temp_warm”, “gender_F*temp_warm”, “gender_M*temp_cold”, and “gender_F*temp_cold”. The previous PROC HPREG step is equivalent to the following:

```
proc hpreg;
   model y = gender_M gender_F
             gender_M*temp_hot  gender_F*temp_hot
             gender_M*temp_warm gender_F*temp_warm
             gender_M*temp_cold gender_F*temp_cold;
run;
```

The SPLIT option can be used on individual classification variables. For example, consider the following PROC HPREG step:

```
proc hpreg;
   class temp(split) gender;
   model y = gender gender*temp;
run;
```

In this case, the effect "gender" is not split and the effect "gender*temp" is split into three effects, which are labeled "gender*temp_hot", "gender*temp_warm", and "gender*temp_cold". Furthermore, each of these three split effects now has two parameters that correspond to the two levels of "gender." The PROC HPREG step is equivalent to the following:

```
proc hpreg;
   class gender;
   model y = gender gender*temp_hot gender*temp_warm gender*temp_cold;
run;
```

You can specify the following *global-options*:

MISSING
treats missing values (".", ".A", . . . , ".Z" for numeric variables and blanks for character variables) as valid values for the CLASS variable.

If you do not specify the MISSING option, observations that have missing values for CLASS variables are removed from the analysis, even if the CLASS variables are not used in the model formulation.

PARAM=*keyword*
specifies the parameterization method for the classification variable or variables. You can specify the following *keywords*:

GLM specifies a less-than-full-rank reference cell coding. This parameterization is used in, for example, the GLM, MIXED, and GLIMMIX procedures in SAS/STAT.

REFERENCE specifies a reference cell encoding. You can choose the reference value by specifying an option for a specific *variable* or set of *variables* in the CLASS statement, or designate the first or last ordered value by specifying a *global-option*. The default is REF=LAST.

For example, suppose that the variable temp has three levels (`'hot'`, `'warm'`, and `'cold'`), that the variable gender has two levels (`'M'` and `'F'`), and that the variables are used in a CLASS statement as follows:

```
class gender(ref='F') temp / param=ref;
```

Then `'F'` is used as the reference level for gender and `'warm'` is used as the reference level for temp.

The GLM parameterization is the default. For more information about how parameterization of classification variables affects the construction and interpretation of model effects, see the section "Specification and Parameterization of Model Effects" on page 55.

TRUNCATE<=*n*>
specifies the truncation width of formatted values of CLASS variables when the optional *n* is specified.

If *n* is not specified, the TRUNCATE option requests that classification levels be determined by using no more than the first 16 characters of the formatted values of CLASS variables.

FREQ Statement

FREQ *variable* **;**

The *variable* in the FREQ statement identifies a numeric variable in the data set that contains the frequency of occurrence for each observation. High-performance statistical procedures that support the FREQ statement treat each observation as if it appeared f times, where f is the value of the FREQ variable for the observation. If the frequency value is not an integer, it is truncated to an integer. If the frequency value is less than 1 or missing, the observation is not used in the analysis. When the FREQ statement is not specified, each observation is assigned a frequency of 1.

ID Statement

ID *variables* **;**

The ID statement lists one or more variables from the input data set that are transferred to output data sets that are created by high-performance statistical procedures, provided that the output data set contains one (or more) records per input observation. For example, when an OUTPUT statement is used to produce observationwise scores or prediction statistics, ID variables are added to the output data set.

By default, high-performance statistical procedures do not include all variables from the input data set in output data sets. In the following statements, a logistic regression model is fit and then scored. The input and output data are stored in the Greenplum database. The output data set contains three columns (p, account, trans_date) where p is computed during the scoring process and the account and transaction date are transferred from the input data set. (High-performance statistical procedures also transfer any distribution keys from the input to the output data.)

```
libname GPLib greenplm server=gpdca user=XXX password=YYY
              database=ZZZ;
proc hplogistic data=gplib.myData;
   class a b;
   model y = a b x1-x20;
   output out=gplib.scores pred=p;
   id account trans_date;
run;
```

SELECTION Statement

SELECTION < *options* > **;**

High-performance statistical procedures that support model selection use the SELECTION statement to control details about the model selection process. This statement is supported in different degrees by the HPGENSELECT, HPREG, and HPLOGISTIC procedures. The HPREG procedure supports the most complete set of options.

You can specify the following *options* in the SELECTION statement:

METHOD=NONE | *method*< *method-options* >

specifies the method used to select the model. You can also specify *method-options* that apply to the specified method by enclosing them in parentheses after the *method*. The default selection method (when the METHOD= option is not specified) is METHOD=STEPWISE.

The following *methods* are available and are explained in detail in the section "Methods" on page 63.

NONE specifies no model selection.

FORWARD specifies forward selection. This method starts with no effects in the model and adds effects.

BACKWARD specifies backward elimination. This method starts with all effects in the model and deletes effects.

STEPWISE specifies stepwise regression. This method is similar to the FORWARD method except that effects already in the model do not necessarily stay there.

FORWARDSWAP specifies forward-swap selection, which is an extension of the forward selection method. Before any addition step, all pairwise swaps of one effect in the model and one effect out of the current model that improve the selection criterion are made. When the selection criterion is R square, this method is the same as the MAXR method in the REG procedure in SAS/STAT software. The high-performance statistical procedure that supports this method is the HPREG procedure.

LAR specifies least angle regression. Like forward selection, this method starts by adding effects to an empty model. The parameter estimates at any step are "shrunk" when they are compared to the corresponding least squares estimates. If the model contains classification variables, then these classification variables are split. See the SPLIT option in the CLASS statement for details. The only high-performance statistical procedure that supports this method is the HPREG procedure.

LASSO adds and deletes parameters by using a version of ordinary least squares in which the sum of the absolute regression coefficients is constrained. If the model contains classification variables, then these classification variables are split. For more information, see the SPLIT option in the CLASS statement. The only high-performance statistical procedure that supports this method is the HPREG procedure.

Table 4.1 lists the applicable *method-options* for each of these methods.

Table 4.1 Applicable *method-options* by *method*

method-option	FORWARD	BACKWARD	STEPWISE	FORWARDSWAP	LAR	LASSO
ADAPTIVE						x
CHOOSE =	x	x	x		x	x
COMPETITIVE			x			
CRITERION =	x	x	x	x		
FAST		x				
LSCOEFFS					x	x
MAXEFFECTS =	x		x	x	x	x
MAXSTEPS =	x	x	x	x	x	x
MINEFFECTS =		x	x			
SELECT =	x	x	x	x		
SLENTRY =	x		x		x	x
SLSTAY =		x	x			x
STOP =	x	x	x	x	x	x

The syntax of the *method-options* that you can specify in parentheses after the SELECTION= option *method* follows. As described in Table 4.1, not all selection *method-options* are applicable to every SELECTION= *method*.

ADAPTIVE < (GAMMA=*nonnegative number***) >**

requests that adaptive weights be applied to each of the coefficients when METHOD=LASSO. Ordinary least squares estimates of the model parameters are used to form the adaptive weights. You use the GAMMA= option to specify the power transformation that is applied to the parameters in forming the adaptive weights. The default value is GAMMA=1.

CHOOSE=*criterion*

chooses from the list of models (at each step of the selection process) the model that yields the best value of the specified criterion. If the optimal value of the specified criterion occurs for models at more than one step, then the model that has the smallest number of parameters is chosen. If you do not specify the CHOOSE= option, then the selected model is the model at the final step in the selection process. The criteria that are supported depend on the type of model that is being fit. For the supported criteria, see the chapters for the relevant high-performance statistical procedures.

COMPETITIVE

is applicable only as a *method-option* when METHOD=STEPWISE and the SELECT criterion is not SL. If you specify the COMPETITIVE option, then the SELECT= criterion is evaluated for all models in which an effect currently in the model is dropped or an effect not yet in the model is added. The effect whose removal from or addition to the model yields the maximum improvement to the SELECT= criterion is dropped or added.

CRITERION=*criterion*

is an alias for the SELECT option.

FAST
implements the computational algorithm of Lawless and Singhal (1978) to compute a first-order approximation to the remaining slope estimates for each subsequent elimination of a variable from the model. When applied in backward selection, this option essentially leads to approximating the selection process as the selection process of a linear regression model in which the crossproducts matrix equals the Hessian matrix in the full model under consideration. The FAST option is available only when METHOD=BACKWARD in the HPLOGISTIC procedure. It is computationally efficient in logistic regression models because the model is not fit after removal of each effect.

LSCOEFFS
requests a hybrid version of the LAR and LASSO methods, in which the sequence of models is determined by the LAR or LASSO algorithm but the coefficients of the parameters for the model at any step are determined by using ordinary least squares.

MAXEFFECTS=*n*
specifies the maximum number of effects in any model that is considered during the selection process. This option is ignored with METHOD=BACKWARD. If at some step of the selection process the model contains the specified maximum number of effects, then no candidates for addition are considered.

MAXSTEPS=*n*
specifies the maximum number of selection steps that are performed. The default value of *n* is the number of effects in the MODEL statement when METHOD=FORWARD, METHOD=BACKWARD, or METHOD=LAR. The default is three times the number of effects when METHOD=STEPWISE or METHOD=LASSO.

MINEFFECTS=*n*
specifies the minimum number of effects in any model that is considered during backward selection. This option is ignored unless METHOD=BACKWARD is specified. The backward selection process terminates if, at some step of the selection process, the model contains the specified minimum number of effects.

SELECT=SL | *criterion*
specifies the criterion that the procedure uses to determine the order in which effects enter or leave at each step of the selection method. The criteria that are supported depend on type of model that is being fit. See the chapter for the relevant high-performance statistical procedure for the supported criteria.

The SELECT option is not valid when METHOD=LAR or METHOD=LASSO. You can use SELECT=SL to request the traditional approach, where effects enter and leave the model based on the significance level. When the value of the SELECT= option is not SL, the effect that is selected to enter or leave at any step of the selection process is the effect whose addition to or removal from the current model yields the maximum improvement in the specified criterion.

SLENTRY=*value*

SLE=*value*
specifies the significance level for entry when STOP=SL or SELECT=SL. The default is 0.05.

SLSTAY=*value*

SLS=*value*

specifies the significance level for staying in the model when STOP=SL or SELECT=SL. The default is 0.05.

STOP=SL | NONE | *criterion*

specifies a criterion that is used to stop the selection process. The criteria that are supported depend on the type of model that is being fit. For information about the supported criteria, see the chapter about the relevant high-performance statistical procedure.

If you do not specify the STOP= option but do specify the SELECT= option, then the criterion specified in the SELECT= option is also used as the STOP= criterion.

If you specify STOP=NONE, then the selection process stops if no suitable add or drop candidates can be found or if a size-based limit is reached. For example, if you specify STOP=NONE MAXEFFECTS=5, then the selection process stops at the first step that produces a model with five effects.

When STOP=SL, selection stops at the step where the significance level of the candidate for entry is greater than the SLENTRY= value for addition steps when METHOD=FORWARD or METHOD=STEPWISE and where the significance level of the candidate for removal is greater than the SLSTAY= value when METHOD=BACKWARD or METHOD=STEPWISE.

If you specify a criterion other than SL for the STOP= option, then the selection process stops if the selection process produces a local extremum of this criterion or if a size-based limit is reached. For example, if you specify STOP=AIC MAXSTEPS=5, then the selection process stops before step 5 if the sequence of models has a local minimum of the AIC criterion before step 5. The determination of whether a local minimum is reached is made on the basis of a stop horizon. The default stop horizon is 3, but you can change it by using the STOPHORIZON= option. If the stop horizon is *n* and the STOP= criterion at any step is better than the stop criterion at the next *n* steps, then the selection process terminates.

DETAILS=NONE | SUMMARY | ALL

DETAILS=STEPS< (CANDIDATES(ALL | *n*)) >

specifies the level of detail to be produced about the selection process. The default is DETAILS=SUMMARY.

The DETAILS=ALL and DETAILS=STEPS options produce the following output:

- tables that provide information about the model that is selected at each step of the selection process.
- entry and removal statistics for inclusion or exclusion candidates at each step. By default, only the top 10 candidates at each step are shown. If you specify STEPS(CANDIDATES(*n*)), then the best *n* candidates are shown. If you specify STEPS(CANDIDATES(ALL)), then all candidates are shown.
- a selection summary table that shows by step the effect that is added to or removed from the model in addition to the values of the SELECT, STOP, and CHOOSE criteria for the resulting model.
- a stop reason table that describes why the selection process stopped.

- a selection reason table that describes why the selected model was chosen.
- a selected effects table that lists the effects that are in the selected model.

The DETAILS=SUMMARY option produces only the selection summary, stop reason, selection reason, and selected effects tables.

HIERARCHY=NONE | SINGLE | SINGLECLASS
specifies whether and how the model hierarchy requirement is applied. This option also controls whether a single effect or multiple effects are allowed to enter or leave the model in one step. You can specify that only classification effects, or both classification and continuous effects, be subject to the hierarchy requirement. The HIERARCHY= option is ignored unless you also specify one of the following options: METHOD=FORWARD, METHOD=BACKWARD, or METHOD=STEPWISE.

Model hierarchy refers to the requirement that, for any term to be in the model, all model effects that are contained in the term must be present in the model. For example, in order for the interaction A*B to enter the model, the main effects A and B must be in the model. Likewise, neither effect A nor effect B can leave the model while the interaction A*B is in the model.

You can specify the following values:

NONE specifies that model hierarchy not be maintained. Any single effect can enter or leave the model at any given step of the selection process.

SINGLE specifies that only one effect enter or leave the model at one time, subject to the model hierarchy requirement. For example, suppose that the model contains the main effects A and B and the interaction A*B. In the first step of the selection process, either A or B can enter the model. In the second step, the other main effect can enter the model. The interaction effect can enter the model only when both main effects have already entered. Also, before A or B can be removed from the model, the A*B interaction must first be removed. All effects (CLASS and interval) are subject to the hierarchy requirement.

SINGLECLASS is the same as HIERARCHY=SINGLE except that only CLASS effects are subject to the hierarchy requirement.

The default value is HIERARCHY=NONE.

SCREEN < (*global-screen-options*) > < = *screen-options* >
requests that a subset of the effects specified in the MODEL statement be chosen as candidate effects for model selection. You use the *global-screen-options* and *screen-options* to specify how such a subset is chosen and to control the detail level of the associated output. The SCREEN option is fully documented in the section "SELECTION Statement" on page 556 in Chapter 14, "The HPREG Procedure," which is the only high-performance statistical procedure that supports the SCREEN option.

SELECTION=NONE | BACKWARD | FORWARD | FORWARDSWAP | STEPWISE | LAR | LASSO
is an alias for the METHOD= option.

STOPHORIZON=*n*
specifies the number of consecutive steps at which the STOP= criterion must worsen in order for a local extremum to be detected. The default value is STOPHORIZON=3. The stop horizon value is ignored if you also specify STOP=NONE or STOP=SL. For example, suppose that STOP=AIC and

the sequence of AIC values at steps 1 to 6 of a selection are 10, 7, 4, 6, 5, 2. If STOPHORIZON=2, then the AIC criterion is deemed to have a local minimum at step 3 because the AIC value at the next two steps are greater than the value 4 that occurs at step 3. However, if STOPHORIZON=3, then the value at step 3 is not deemed to be a local minimum because the AIC value at step 6 is lower than the AIC value at step 3.

VAR Statement

VAR *variable-list* **;**

Some high-performance statistical procedures (in particular procedures that do not support a MODEL statement) use a VAR statement to identify numerical variables for the analysis.

WEIGHT Statement

WEIGHT *variable* **;**

The *variable* in the WEIGHT statement is used as a weight to perform a weighted analysis of the data. Observations with nonpositive or missing weights are not included in the analysis. If a WEIGHT statement is not included, all observations that are used in the analysis are assigned a weight of 1.

Levelization of Classification Variables

A classification variable enters the statistical analysis or model not through its values but through its levels. The process of associating values of a variable with levels is termed *levelization*.

During the process of levelization, observations that share the same value are assigned to the same level. The manner in which values are grouped can be affected by the inclusion of formats. The sort order of the levels can be determined by specifying the ORDER= option in the procedure statement. In high-performance statistical procedures, you can also control the sorting order separately for each variable in the CLASS statement.

Consider the data on nine observations in Table 4.2. The variable A is integer-valued, and the variable X is a continuous variable that has a missing value for the fourth observation. The fourth and fifth columns of Table 4.2 apply two different formats to the variable X.

Table 4.2 Example Data for Levelization

Obs	A	x	FORMAT x 3.0	FORMAT x 3.1
1	2	1.09	1	1.1
2	2	1.13	1	1.1
3	2	1.27	1	1.3
4	3	.	.	.
5	3	2.26	2	2.3

Table 4.2 *continued*

Obs	A	x	FORMAT x 3.0	FORMAT x 3.1
6	3	2.48	2	2.5
7	4	3.34	3	3.3
8	4	3.34	3	3.3
9	4	3.14	3	3.1

By default, levelization of the variables groups the observations by the formatted value of the variable, except for numerical variables for which no explicit format is provided. Numerical variables for which no explicit format is provided are sorted by their internal value. The levelization of the four columns in Table 4.2 leads to the level assignment in Table 4.3.

Table 4.3 Values and Levels

	A		X		FORMAT x 3.0		FORMAT x 3.1	
Obs	Value	Level	Value	Level	Value	Level	Value	Level
1	2	1	1.09	1	1	1	1.1	1
2	2	1	1.13	2	1	1	1.1	1
3	2	1	1.27	3	1	1	1.3	2
4	3	2	.	.	.	.	.	.
5	3	2	2.26	4	2	2	2.3	3
6	3	2	2.48	5	2	2	2.5	4
7	4	3	3.34	7	3	3	3.3	6
8	4	3	3.34	7	3	3	3.3	6
9	4	3	3.14	6	3	3	3.1	5

The sort order for the levels of CLASS variables can be specified in the ORDER= option in the CLASS statement.

When ORDER=FORMATTED (which is the default) is in effect for numeric variables for which you have supplied no explicit format, the levels are ordered by their internal values. To order numeric class levels that have no explicit format by their BEST12. formatted values, you can specify the BEST12. format explicitly for the CLASS variables.

Table 4.4 shows how values of the ORDER= option are interpreted.

Table 4.4 Interpretation of Values of ORDER= Option

Value of ORDER=	Levels Sorted By
DATA	Order of appearance in the input data set
FORMATTED	External formatted value, except for numeric variables that have no explicit format, which are sorted by their unformatted (internal) value

Table 4.4 *continued*

Value of ORDER=	Levels Sorted By
FREQ	Descending frequency count (levels that have the most observations come first in the order)
INTERNAL	Unformatted value
FREQDATA	Order of descending frequency count, and within counts by order of appearance in the input data set when counts are tied
FREQFORMATTED	Order of descending frequency count, and within counts by formatted value when counts are tied
FREQINTERNAL	Order of descending frequency count, and within counts by unformatted (internal) value when counts are tied

For FORMATTED, FREQFORMATTED, FREQINTERNAL, and INTERNAL values, the sort order is machine-dependent. For more information about sort order, see the chapter about the SORT procedure in the *Base SAS Procedures Guide* and the discussion of BY-group processing in *SAS Language Reference: Concepts*.

When the MISSING option is specified in the CLASS statement, the missing values ('.' for a numeric variable and blanks for a character variable) are included in the levelization and are assigned a level. Table 4.5 displays the results of levelizing the values in Table 4.2 when the MISSING option is in effect.

Table 4.5 Values and Levels with the MISSING Option

	A		X		FORMAT x 3.0		FORMAT x 3.1	
Obs	Value	Level	Value	Level	Value	Level	Value	Level
1	2	1	1.09	2	1	2	1.1	2
2	2	1	1.13	3	1	2	1.1	2
3	2	1	1.27	4	1	2	1.3	3
4	3	2	.	1	.	1	.	1
5	3	2	2.26	5	2	3	2.3	4
6	3	2	2.48	6	2	3	2.5	5
7	4	3	3.34	8	3	4	3.3	7
8	4	3	3.34	8	3	4	3.3	7
9	4	3	3.14	7	3	4	3.1	6

When the MISSING option is not specified, it is important to understand the implications of missing values for your statistical analysis. When a high-performance statistical procedure levelizes the CLASS variables, an observation for which any CLASS variable has a missing value is excluded from the analysis. This is true regardless of whether the variable is used to form the statistical model. For example, consider the case in which some observations contain missing values for variable A but the records for these observations are otherwise complete with respect to all other variables in the statistical models. The analysis results from the following statements do not include any observations for which variable A contains missing values, even though A is not specified in the MODEL statement:

```
class A B;
model y = B x B*x;
```

High-performance statistical procedures print a "Number of Observations" table that shows the number of observations that are read from the data set and the number of observations that are used in the analysis. Pay careful attention to this table—especially when your data set contains missing values—to ensure that no observations are unintentionally excluded from the analysis.

Specification and Parameterization of Model Effects

High-performance statistical procedures that have a MODEL statement support the formation of effects. An *effect* is an element in a linear model structure that is formed from one or more variables. At some point the statistical representations of these models involve linear structures such as

$$\mathbf{X}\boldsymbol{\beta}$$

or

$$\mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\boldsymbol{\gamma}$$

The model matrices $\mathbf{X}$ and $\mathbf{Z}$ are formed according to effect construction rules.

Procedures that also have a CLASS statement support the rich set of effects that is discussed in this section. In order to correctly interpret the results from a statistical analysis, you need to understand how construction (*parameterization*) rules apply to regression-type models, whether these are linear models in the HPREG procedure or generalized linear models in the HPLOGISTIC and HPGENSELECT procedures.

Effects are specified by a special notation that uses variable names and operators. There are two types of variables: classification (or CLASS) variables and continuous variables. *Classification variables* can be either numeric or character and are specified in a CLASS statement. For more information, see the section "Levelization of Classification Variables" on page 52. An independent variable that is not declared in the CLASS statement is assumed to be *continuous*. For example, the heights and weights of subjects are continuous variables.

Two primary operators (crossing and nesting) are used for combining the variables, and several additional operators are used to simplify effect specification. Operators are discussed in the section "Effect Operators" on page 56.

High-performance statistical procedures that have a CLASS statement support a general linear model (GLM) parameterization and a reference parameterization for the classification variables. The GLM parameterization is the default for all high-performance statistical procedures. For more information, see the sections "GLM Parameterization of Classification Variables and Effects" on page 58 and "Reference Parameterization" on page 62.

Effect Operators

Table 4.6 summarizes the operators that are available for selecting and constructing effects. These operators are discussed in the following sections.

Table 4.6 Available Effect Operators

Operator	Example	Description
Interaction	A*B	Crosses the levels of the effects
Nesting	A(B)	Nests A levels within B levels
Bar operator	A | B | C	Specifies all interactions
At sign operator	A | B | C@2	Reduces interactions in bar effects
Dash operator	A1-A10	Specifies sequentially numbered variables
Colon operator	A:	Specifies variables with common prefix
Double dash operator	A--C	Specifies sequential variables in data set order

Bar and At Sign Operators

You can shorten the specification of a large factorial model by using the bar operator. For example, two ways of writing the model for a full three-way factorial model follow:

```
model Y = A B C   A*B A*C B*C   A*B*C;

model Y = A|B|C;
```

When the bar (|) is used, the right and left sides become effects, and the cross of them becomes an effect. Multiple bars are permitted. The expressions are expanded from left to right, using rules 2–4 given in Searle (1971, p. 390).

- Multiple bars are evaluated from left to right. For example, A | B | C is evaluated as follows:

$$\begin{aligned} \text{A | B | C} &\rightarrow \{\text{ A | B }\}\text{ | C} \\ &\rightarrow \{\text{ A B A*B }\}\text{ | C} \\ &\rightarrow \text{A B A*B C A*C B*C A*B*C} \end{aligned}$$

- Crossed and nested groups of variables are combined. For example, A(B) | C(D) generates A*C(B D), among other terms.
- Duplicate variables are removed. For example, A(C) | B(C) generates A*B(C C), among other terms, and the extra C is removed.
- Effects are discarded if a variable occurs on both the crossed and nested parts of an effect. For example, A(B) | B(D E) generates A*B(B D E), but this effect is eliminated immediately.

You can also specify the maximum number of variables involved in any effect that results from bar evaluation by specifying that maximum number, preceded by an at sign (@), at the end of the bar effect. For example, the following specification selects only those effects that contain two or fewer variables:

```
model Y = A|B|C@2;
```

The preceding example is equivalent to specifying the following MODEL statement:

```
model Y = A B C   A*B A*C B*C;
```

More examples of using the bar and at operators follow:

A | C(B)	is equivalent to	A C(B) A*C(B)
A(B) | C(B)	is equivalent to	A(B) C(B) A*C(B)
A(B) | B(D E)	is equivalent to	A(B) B(D E)
A | B(A) | C	is equivalent to	A B(A) C A*C B*C(A)
A | B(A) | C@2	is equivalent to	A B(A) C A*C
A | B | C | D@2	is equivalent to	A B A*B C A*C B*C D A*D B*D C*D
A*B(C*D)	is equivalent to	A*B(C D)

Colon, Dash, and Double Dash Operators

You can simplify the specification of a large model when some of your variables have a common prefix by using the colon (:) operator and the dash (-) operator. The dash operator enables you to list variables that are numbered sequentially, and the colon operator selects all variables with a given prefix. For example, if your data set contains the variables X1 through X9, the following MODEL statements are equivalent:

```
model Y = X1 X2 X3 X4 X5 X6 X7 X8 X9;

model Y = X1-X9;

model Y = X:;
```

If your data set contains only the three covariates X1, X2, and X9, then the colon operator selects all three variables:

```
model Y = X:;
```

However, the following specification returns an error because X3 through X8 are not in the data set:

```
model Y = X1-X9;
```

The double dash (- -) operator enables you to select variables that are stored sequentially in the SAS data set, whether or not they have a common prefix. You can use the CONTENTS procedure (see *Base SAS Procedures Guide*) to determine your variable ordering. For example, if you replace the dash in the preceding MODEL statement with a double dash, as follows, then all three variables are selected:

```
model Y = X1--X9;
```

If your data set contains the variables A, B, and C, then you can use the double dash operator to select these variables by specifying the following:

```
model Y = A--C;
```

GLM Parameterization of Classification Variables and Effects

Table 4.7 shows the types of effects that are available in high-performance statistical procedures; they are discussed in more detail in the following sections. Let A, B, and C represent classification variables, and let X and Z represent continuous variables.

Table 4.7 Available Types of Effects

Effect	Example	Description
Intercept	Default	Intercept (unless NOINT)
Regression	X Z	Continuous variables
Polynomial	X*Z	Interaction of continuous variables
Main	A B	CLASS variables
Interaction	A*B	Crossing of CLASS variables
Nested	A(B)	Main effect A nested within CLASS effect B
Continuous-by-class	X*A	Crossing of continuous and CLASS variables
Continuous-nesting-class	X(A)	Continuous variable X1 nested within CLASS variable A
General	X*Z*A(B)	Combinations of different types of effects

Table 4.8 shows some examples of MODEL statements that use various types of effects.

Table 4.8 Model Statement Effect Examples

Specification	Type of Model
`model Y=X;`	Simple regression
`model Y=X Z;`	Multiple regression
`model Y=X X*X;`	Polynomial regression
`model Y=A;`	One-way analysis of variance (ANOVA)
`model Y=A B C;`	Main-effects ANOVA
`model Y=A B A*B;`	Factorial ANOVA with interaction
`model y=A B(A) C(B A);`	Nested ANOVA
`model Y=A X;`	Analysis of covariance (ANCOVA)
`model Y=A X(A);`	Separate-slopes regression
`model Y=A X X*A;`	Homogeneity-of-slopes regression

Intercept

By default, high-performance statistical linear models automatically include a column of 1s in **X**. This column corresponds to an intercept parameter. In many procedures, you can use the NOINT option in the MODEL statement to suppress this intercept. For example, the NOINT option is useful when the MODEL statement contains a classification effect and you want the parameter estimates to be in terms of the mean response for each level of that effect.

Regression Effects

Numeric variables or polynomial terms that involve them can be included in the model as regression effects (covariates). The actual values of such terms are included as columns of the relevant model matrices. You can use the bar operator along with a regression effect to generate polynomial effects. For example, X | X | X expands to X X*X X*X*X, which is a cubic model.

Main Effects

If a classification variable has m levels, the GLM parameterization generates m columns for its main effect in the model matrix. Each column is an indicator variable for a given level. The order of the columns is the sort order of the values of their levels and can be controlled by the ORDER= option in the CLASS statement.

Table 4.9 is an example where β_0 denotes the intercept and A and B are classification variables that have two and three levels, respectively.

Table 4.9 Example of Main Effects

Data		**I**	A		B		
A	B	β_0	A1	A2	B1	B2	B3
1	1	1	1	0	1	0	0
1	2	1	1	0	0	1	0
1	3	1	1	0	0	0	1
2	1	1	0	1	1	0	0
2	2	1	0	1	0	1	0
2	3	1	0	1	0	0	1

There are usually more columns for these effects than there are degrees of freedom to estimate them. In other words, the GLM parameterization of main effects is *singular*.

Interaction Effects

Often a model includes interaction (crossed) effects to account for how the effect of a variable changes along with the values of other variables. With an interaction, the terms are first reordered to correspond to the order of the variables in the CLASS statement. Thus, B*A becomes A*B if A precedes B in the CLASS statement. Then, the GLM parameterization generates columns for all combinations of levels that occur in the data. The order of the columns is such that the rightmost variables in the interaction change faster than the leftmost variables (Table 4.10).

In the HPLMIXED procedure, which supports both fixed- and random-effects models, empty columns (that is, columns that would contain all 0s) are not generated for fixed effects, but they are generated for random effects.

Table 4.10 Example of Interaction Effects

Data		I	A		B			A*B					
A	B	β_0	A1	A2	B1	B2	B3	A1B1	A1B2	A1B3	A2B1	A2B2	A2B3
1	1	1	1	0	1	0	0	1	0	0	0	0	0
1	2	1	1	0	0	1	0	0	1	0	0	0	0
1	3	1	1	0	0	0	1	0	0	1	0	0	0
2	1	1	0	1	1	0	0	0	0	0	1	0	0
2	2	1	0	1	0	1	0	0	0	0	0	1	0
2	3	1	0	1	0	0	1	0	0	0	0	0	1

In the preceding matrix, main-effects columns are not linearly independent of crossed-effects columns. In fact, the column space for the crossed effects contains the space of the main effect.

When your model contains many interaction effects, you might be able to code them more parsimoniously by using the bar operator (|). The bar operator generates all possible interaction effects. For example, A | B | C expands to A B A*B C A*C B*C A*B*C. To eliminate higher-order interaction effects, use the at sign (@) in conjunction with the bar operator. For example, A | B | C | D@2 expands to A B A*B C A*C B*C D A*D B*D C*D.

Nested Effects

Nested effects are generated in the same manner as crossed effects. Hence, the design columns that are generated by the following two statements are the same (but the ordering of the columns is different):

```
model Y=A B(A);

model Y=A A*B;
```

The nesting operator in high-performance statistical procedures is more of a notational convenience than an operation that is distinct from crossing. Nested effects are typically characterized by the property that the nested variables do not appear as main effects. The order of the variables within nesting parentheses is made to correspond to the order of these variables in the CLASS statement. The order of the columns is such that variables outside the parentheses index faster than those inside the parentheses, and the rightmost nested variables index faster than the leftmost variables (Table 4.11).

Table 4.11 Example of Nested Effects

Data		I	A		B(A)					
A	B	β_0	A1	A2	B1A1	B2A1	B3A1	B1A2	B2A2	B3A2
1	1	1	1	0	1	0	0	0	0	0
1	2	1	1	0	0	1	0	0	0	0
1	3	1	1	0	0	0	1	0	0	0
2	1	1	0	1	0	0	0	1	0	0
2	2	1	0	1	0	0	0	0	1	0
2	3	1	0	1	0	0	0	0	0	1

Continuous-Nesting-Class Effects

When a continuous variable nests or crosses with a classification variable, the design columns are constructed by multiplying the continuous values into the design columns for the classification effect (Table 4.12).

Table 4.12 Example of Continuous-Nesting-Class Effects

Data		I	A		X(A)	
X	A	β_0	A1	A2	X(A1)	X(A2)
21	1	1	1	0	21	0
24	1	1	1	0	24	0
22	1	1	1	0	22	0
28	2	1	0	1	0	28
19	2	1	0	1	0	19
23	2	1	0	1	0	23

This model estimates a separate intercept and a separate slope for X within each level of A.

Continuous-by-Class Effects

Continuous-by-class effects generate the same design columns as continuous-nesting-class effects. Table 4.13 shows the construction of the X*A effect. The two columns for this effect are the same as the columns for the X(A) effect in Table 4.12.

Table 4.13 Example of Continuous-by-Class Effects

Data		I	X	A		X*A	
X	A	β_0	X	A1	A2	X*A1	X*A2
21	1	1	21	1	0	21	0
24	1	1	24	1	0	24	0
22	1	1	22	1	0	22	0
28	2	1	28	0	1	0	28
19	2	1	19	0	1	0	19
23	2	1	23	0	1	0	23

You can use continuous-by-class effects together with pure continuous effects to test for homogeneity of slopes.

General Effects

An example that combines all the effects is X1*X2*A*B*C(D E). The continuous list comes first, followed by the crossed list, followed by the nested list in parentheses. You should be aware of the sequencing of parameters when you use statements that depend on the ordering of parameters. Such statements include CONTRAST and ESTIMATE statements, which are used in a number of procedures to estimate and test functions of the parameters.

Effects might be renamed by the procedure to correspond to ordering rules. For example, B*A(E D) might be renamed A*B(D E) to satisfy the following:

- Classification variables that occur outside parentheses (crossed effects) are sorted in the order in which they appear in the CLASS statement.
- Variables within parentheses (nested effects) are sorted in the order in which they appear in the CLASS statement.

The sequencing of the parameters that are generated by an effect is determined by the variables whose levels are indexed faster:

- Variables in the crossed list index faster than variables in the nested list.
- Within a crossed or nested list, variables to the right index faster than variables to the left.

For example, suppose a model includes four effects—A, B, C, and D—each having two levels, 1 and 2. If the CLASS statement is

```
class A B C D;
```

then the order of the parameters for the effect B*A(C D), which is renamed A*B(C D), is

$$\begin{array}{llll} A_1B_1C_1D_1 \rightarrow & A_1B_2C_1D_1 \rightarrow & A_2B_1C_1D_1 \rightarrow & A_2B_2C_1D_1 \rightarrow \\ A_1B_1C_1D_2 \rightarrow & A_1B_2C_1D_2 \rightarrow & A_2B_1C_1D_2 \rightarrow & A_2B_2C_1D_2 \rightarrow \\ A_1B_1C_2D_1 \rightarrow & A_1B_2C_2D_1 \rightarrow & A_2B_1C_2D_1 \rightarrow & A_2B_2C_2D_1 \rightarrow \\ A_1B_1C_2D_2 \rightarrow & A_1B_2C_2D_2 \rightarrow & A_2B_1C_2D_2 \rightarrow & A_2B_2C_2D_2 \end{array}$$

Note that first the crossed effects B and A are sorted in the order in which they appear in the CLASS statement so that A precedes B in the parameter list. Then, for each combination of the nested effects in turn, combinations of A and B appear. The B effect changes fastest because it is rightmost in the cross list. Then A changes next fastest, and D changes next fastest. The C effect changes most slowly because it is leftmost in the nested list.

Reference Parameterization

Classification variables can be represented in the reference parameterization in high-performance statistical procedures. Only one parameterization applies to the variables in the CLASS statement.

To understand the reference representation, consider the classification variable A that has four values, 1, 2, 5, and 7. The reference parameterization generates three columns (one less than the number of variable levels). The columns indicate group membership of the nonreference levels. For the reference level, the three dummy variables have a value of 0. If the reference level is 7 (REF='7'), the design columns for variable A are as shown in Table 4.14.

Table 4.14 Reference Coding

	Design Matrix		
A	**A1**	**A2**	**A5**
1	1	0	0
2	0	1	0
5	0	0	1
7	0	0	0

Parameter estimates of CLASS main effects that use the reference coding scheme estimate the difference in the effect of each nonreference level compared to the effect of the reference level.

Model Selection

Methods

The model selection methods implemented in high-performance statistical procedures are specified in the METHOD= option in the SELECTION statement. The following methods are available, although specific procedures might support only a subset of these methods. Furthermore, the examples in this section refer to fit criteria that might not be supported by a specific procedure.

Full Model Fitted

When METHOD=NONE, the complete model that is specified in the MODEL statement is used to fit the model, and no effect selection is done.

Forward Selection

METHOD=FORWARD specifies the forward selection technique, which begins with just the intercept and then sequentially adds the effect that most improves the fit. The process terminates when no significant improvement can be obtained by adding any effect.

In the traditional implementation of forward selection, the statistic that is used to determine whether to add an effect is the significance level of a hypothesis test that reflects an effect's contribution to the model if it is included. At each step, the effect that is most significant is added. The process stops when the significance level for adding any effect is greater than some specified entry significance level.

An alternative approach to address the critical problem of when to stop the selection process is to assess the quality of the models that are produced by the forward selection method and choose the model from this sequence that "best" balances goodness of fit against model complexity. You can use several criteria for this purpose. These criteria fall into two groups—information criteria and criteria that are based on out-of-sample prediction performance.

You use the CHOOSE= option to specify the criterion for selecting one model from the sequence of models produced. If you do not specify a CHOOSE= criterion, then the model at the final step is the selected model.

For example, if you specify the following statement, then forward selection terminates at the step where no effect can be added at the 0.2 significance level:

```
selection method=forward(select=SL choose=AIC SLE=0.2);
```

However, the selected model is the first one that has the minimum value of Akaike's information criterion. In some cases, this minimum value might occur at a step much earlier than the final step. In other cases, the AIC might start increasing only if more steps are performed—that is, a larger value is used for the significance level for entry. If you want to minimize AIC, then too many steps are performed in the former case and too few in the latter case. To address this issue, high-performance statistical procedures enable you to specify a stopping criterion by using the STOP= option. When you specify a stopping criterion, forward selection continues until a local extremum of the stopping criterion in the sequence of models generated is reached. To be deemed a local extremum, a criterion value at a given step must be better than its value at the next *n* steps, where *n* is known as the "stop horizon." By default, the stop horizon is three steps, but you can change this by specifying the STOPHORIZON= option.

For example, if you specify the following statement, then forward selection terminates at the step where the effect to be added at the next step would produce a model that has an AIC statistic larger than the AIC statistic of the current model:

```
selection method=forward(select=SL stop=AIC) stophorizon=1;
```

In most cases, provided that the entry significance level is large enough that the local extremum of the named criterion occurs before the final step, specifying either of the following statements selects the same model, but more steps are done in the first case:

```
selection method=forward(select=SL choose=CRITERION);

selection method=forward(select=SL stop=CRITERION);
```

In some cases, there might be a better local extremum that cannot be reached if you specify the STOP= option but can be found if you use the CHOOSE= option. Also, you can use the CHOOSE= option in preference to the STOP= option if you want to examine how the named criterion behaves as you move beyond the step where the first local minimum of this criterion occurs.

You can specify both the CHOOSE= and STOP= options. You can also use these options together with options that specify size-based limits on the selected model. You might want to consider models that are generated by forward selection and have at most some fixed number of effects, but select from within this set based on a criterion that you specify. For example, specifying the following statements requests that forward selection continue until there are 20 effects in the final model and chooses among the sequence of models the one that has the largest value of the adjusted R-square statistic:

```
selection method=forward(stop=none maxeffects=20 choose=ADJRSQ);
```

You can also combine these options to select a model where one of two conditions is met. For example, the following statement chooses whatever occurs first between a local minimum of the sum of squares on validation data and a local minimum of the corrected Akaike's information criterion (AICC):

```
selection method=forward(stop=AICC choose=VALIDATE);
```

It is important to keep in mind that forward selection bases the decision about what effect to add at any step by considering models that differ by one effect from the current model. This search paradigm cannot guarantee reaching a "best" subset model. Furthermore, the add decision is greedy in the sense that the effect that is deemed most significant is the effect that is added. However, if your goal is to find a model that is best in terms of some selection criterion other than the significance level of the entering effect, then even this one step choice might not be optimal. For example, the effect that you would add to get a model that has the smallest value of the Mallows' $C(p)$ statistic at the next step is not necessarily the same effect that is most significant based on a hypothesis test. High-performance statistical procedures enable you to specify the criterion to optimize at each step by using the SELECT= option. For example, the following statement requests that at each step the effect that is added be the one that produces a model that has the smallest value of the Mallows' $C(p)$ statistic:

```
selection method=forward(select=CP);
```

In the case where all effects are variables (that is, effects with one degree of freedom and no hierarchy), using ADJRSQ, AIC, AICC, BIC, CP, RSQUARE, or SBC as the selection criterion for forward selection produces the same sequence of additions. However, if the degrees of freedom contributed by different effects are not constant or if an out-of-sample prediction-based criterion is used, then different sequences of additions might be obtained.

You can use the SELECT= option together with the CHOOSE= and STOP= options. If you specify only the SELECT= criterion, then this criterion is also used as the stopping criterion. In the previous example where only the selection criterion is specified, not only do effects enter based on the Mallows' $C(p)$ statistic, but the selection terminates when the $C(p)$ statistic has a local minimum.

You can find discussion and references to studies about criteria for variable selection in Burnham and Anderson (2002), along with some cautions and recommendations.

Examples of Forward Selection Specifications

The following statement adds effects that at each step produce the lowest value of the SBC statistic and stops at the step where adding any effect would increase the SBC statistic:

```
selection method=forward stophorizon=1;
```

The following statement adds effects based on significance level and stops when all candidate effects for entry at a step have a significance level greater than the default entry significance level of 0.05:

```
selection=forward(select=SL);
```

The following statement adds effects based on significance level and stops at a step where adding any effect increases the error sum of squares computed on the validation data:

```
selection=forward(select=SL stop=validation) stophorizon=1;
```

The following statement adds effects that at each step produce the lowest value of the AIC statistic and stops at the first step whose AIC value is smaller than the AIC value at the next three steps:

```
selection=forward(select=AIC);
```

The following statement adds effects that at each step produce the largest value of the adjusted R-square statistic and stops at the step where the significance level that corresponds to the addition of this effect is greater than 0.2:

```
selection=forward(select=ADJRSQ stop=SL SLE=0.2);
```

Backward Elimination

METHOD=BACKWARD specifies the backward elimination technique. This technique starts from the full model, which includes all independent effects. Then effects are deleted one by one until a stopping condition is satisfied. At each step, the effect that shows the smallest contribution to the model is deleted.

In the traditional implementation of backward selection, the statistic that is used to determine whether to drop an effect is significance level. At any step, the least significant predictor is dropped and the process continues until all effects that remain in the model are significant at a specified stay significance level (SLS).

Just as with forward selection, you can use the SELECT= option to change the criterion that is used to assess effect contributions. You can also specify a stopping criterion in the STOP= option and use a CHOOSE= option to provide a criterion for selecting among the sequence of models produced. For more information, see the discussion in the section "Forward Selection" on page 63.

Examples of Backward Selection Specifications

The following statement removes effects that at each step produce the largest value of the Schwarz Bayesian information criterion (SBC) statistic and stops at the step where removing any effect increases the SBC statistic:

```
selection method=backward stophorizon=1;
```

The following statement bases removal of effects on significance level and stops when all candidate effects for removal at a step are significant at the default stay significance level of 0.05:

```
selection method=backward(select=SL);
```

The following statement bases removal of effects on significance level and stops when all effects in the model are significant at the 0.1 level. Finally, from the sequence of models generated, the chosen model is the one that produces the smallest average square error when scored on the validation data:

```
selection method=backward(select=SL choose=validate SLS=0.1);
```

The following statement applies in logistic regression models the fast backward technique of Lawless and Singhal (1978), a first-order approximation that has greater numerical efficiency than full backward selection:

```
selection method=backward(fast);
```

The fast technique fits an initial full logistic model and a reduced model after the candidate effects have been dropped. On the other hand, full backward selection fits a logistic regression model each time an effect is removed from the model.

Stepwise Selection

METHOD=STEPWISE specifies the stepwise method, which is a modification of the forward selection technique that differs in that effects already in the model do not necessarily stay there.

In the traditional implementation of stepwise selection method, the same entry and removal significance levels for the forward selection and backward elimination methods are used to assess contributions of effects as they are added to or removed from a model. If, at a step of the stepwise method, any effect in the model is not significant at the SLSTAY= level, then the least significant of these effects is removed from the model and the algorithm proceeds to the next step. This ensures that no effect can be added to a model while some effect currently in the model is not deemed significant. Only after all necessary deletions have been accomplished can another effect be added to the model. In this case the effect whose addition is the most significant is added to the model and the algorithm proceeds to the next step. The stepwise process ends when none of the effects outside the model is significant at the SLENTRY= level and every effect in the model is significant at the SLSTAY= level. In some cases, neither of these two conditions for stopping is met and the sequence of models cycles. In this case, the stepwise method terminates at the end of the cycle.

Just as you can in forward selection and backward elimination, you can use the SELECT= option to change the criterion that is used to assess effect contributions. You can also use the STOP= option to specify a stopping criterion and use a CHOOSE= option to provide a criterion for selecting among the sequence of models produced. For more information, see the section "Forward Selection" on page 63.

For selection criteria other than significance level, high-performance statistical procedures optionally support a further modification in the stepwise method. In the standard stepwise method, no effect can enter the model if removing any effect currently in the model would yield an improved value of the selection criterion. In the modification, you can use the COMPETITIVE option to specify that addition and deletion of effects should be treated competitively. The selection criterion is evaluated for all models that are produced by deleting an effect from the current model or by adding an effect to this model. The action that most improves the selection criterion is the action taken.

Examples of Stepwise Selection Specifications

The following statement requests stepwise selection based on the SBC criterion:

```
selection method=stepwise;
```

First, if removing any effect yields a model that has a lower SBC statistic than the current model, then the effect that produces the smallest SBC statistic is removed. If removing any effect increases the SBC statistic, then provided that adding some effect lowers the SBC statistic, the effect that produces the model that has the lowest SBC is added.

The following statement requests the traditional stepwise method:

```
selection=stepwise(select=SL)
```

First, if the removal of any effect in the model is not significant at the default stay level of 0.05, then the least significant effect is removed and the algorithm proceeds to the next step. Otherwise, the effect whose addition is the most significant is added, provided that it is significant at the default entry level of 0.05.

The following statement requests the traditional stepwise method, where effects enter and leave based on significance levels, but with the following extra check: if any effect to be added or removed yields a model whose SBC statistic is greater than the SBC statistic of the current model, then the stepwise method terminates at the current model.

```
selection method=stepwise(select=SL stop=SBC) stophorizon=1;
```

In this case, the entry and stay significance levels still play a role because they determine whether an effect is deleted from or added to the model. This extra check might result in the selection terminating before a local minimum of the SBC criterion is found.

The following statement selects effects to enter or drop as in the previous example except that the significance level for entry is now 0.1 and the significance level to stay is 0.08. From the sequence of models produced, the selected model is chosen to yield the minimum AIC statistic:

```
selection method=stepwise(select=SL SLE=0.1 SLS=0.08 choose=AIC);
```

The following statement requests stepwise selection that is based on the AICC criterion and treats additions and deletions competitively:

```
selection method=stepwise(select=AICC competitive);
```

Each step evaluates the AICC statistics that correspond to the removal of any effect in the current model or the addition of any effect to the current model and chooses the addition or removal that produced the minimum value, provided that this minimum is lower than the AICC statistic of the current model.

The following statement requests stepwise selection that is based on the SBC criterion, treats additions and deletions competitively, and stops based on the average square error over the validation data:

```
selection=stepwise(select=SBC competitive stop=VALIDATE);
```

At any step, SBC statistics that correspond to the removal of any effect from the current model or the addition of any effect to the current model are evaluated. The addition or removal that produces the minimum SBC value is made. The average square error on the validation data for the model with this addition or removal is evaluated. The selection stops when the average square error so produced increases for three consecutive steps.

Forward-Swap Selection

METHOD=FORWARDSWAP specifies the forward-swap selection method, which is an extension of the forward selection method. The forward-swap selection method incorporates steps that improve a model by replacing an effect in the model with an effect that is not in the model. When the model selection criterion is R square, this method is the same as the maximum R-square improvement (MAXR) method that is implemented in the REG procedure in SAS/STAT software. You cannot use the effect significance level as the selection criterion for the forward-swap method.

The forward-swap selection method begins by finding the one-effect model that produces the best value of the selection criterion. Then another effect (the one that yields the greatest improvement in the selection criterion) is added. After the two-effect model is obtained, each of the effects in the model is compared to each effect that is not in the model. For each comparison, the forward-swap method determines whether removing one effect and replacing it with the other effect improves the selection criterion. After comparing all possible swaps, the forward-swap method makes the swap that produces the greatest improvement in the selection criterion. Comparisons begin again, and the process continues until the forward-swap method finds that no other swap could improve the selection criterion. Thus, the two-variable model that is produced is considered the "best" two-variable model that the technique can find. Another variable is then added to the model, and the comparing-and-swapping process is repeated to find the "best" three-variable model, and so on.

The difference between the stepwise selection method and the forward-swap selection method is that all swaps are evaluated before any addition is made in the forward-swap method. In the stepwise selection method, the "worst" effect might be removed without considering what adding the "best" remaining effects might accomplish. Because the forward-swap method needs to examine all possible pairwise effect swaps at each step of the selection process, the forward-swap method is much more computationally expensive than the stepwise selection method; it might not be appropriate for models that contain a large number of effects.

Least Angle Regression

METHOD=LAR specifies least angle regression (LAR), which is supported in the HPREG procedure. LAR was introduced by Efron et al. (2004). Not only does this algorithm provide a selection method in its own right, but with one additional modification, it can be used to efficiently produce LASSO solutions. Just like the forward selection method, the LAR algorithm produces a sequence of regression models in which one parameter is added at each step, terminating at the full least squares solution when all parameters have entered the model.

The algorithm starts by centering the covariates and response and scaling the covariates so that they all have the same corrected sum of squares. Initially all coefficients are zero, as is the predicted response. The predictor that is most correlated with the current residual is determined, and a step is taken in the direction of this predictor. The length of this step determines the coefficient of this predictor and is chosen so that some other predictor and the current predicted response have the same correlation with the current residual. At this point, the predicted response moves in the direction that is equiangular between these two predictors. Moving in this direction ensures that these two predictors continue to have a common correlation with the current residual. The predicted response moves in this direction until a third predictor has the same correlation with the current residual as the two predictors already in the model. A new direction is determined that is equiangular among these three predictors, and the predicted response moves in this direction until a fourth predictor, which has the same correlation with the current residual, joins the set. This process continues until all predictors are in the model.

As in other selection methods, the issue of when to stop the selection process is crucial. You can use the CHOOSE= option to specify a criterion for choosing among the models at each step. You can also use the STOP= option to specify a stopping criterion. These formulas use the approximation that at step k of the LAR algorithm, the model has k degrees of freedom. See Efron et al. (2004) for a detailed discussion of this so-called simple approximation.

A modification of LAR selection that is suggested in Efron et al. (2004) uses the LAR algorithm to select the set of covariates in the model at any step, but it uses ordinary least squares regression with just these covariates

to obtain the regression coefficients. You can request this hybrid method by specifying the LSCOEFFS suboption of METHOD=LAR.

Lasso Selection

Method=LASSO specifies the least absolute shrinkage and selection operator (LASSO) method, which is supported in the HPREG procedure. LASSO arises from a constrained form of ordinary least squares regression where the sum of the absolute values of the regression coefficients is constrained to be smaller than a specified parameter. More precisely let $X = (x_1, x_2, \ldots, x_m)$ denote the matrix of covariates and let y denote the response, where the x_is have been centered and scaled to have unit standard deviation and mean zero and y has mean zero. Then for a given parameter t, the LASSO regression coefficients $\beta = (\beta_1, \beta_2, \ldots, \beta_m)$ are the solution to the following constrained optimization problem:

$$\text{minimize} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||^2 \qquad \text{subject to} \quad \sum_{j=1}^{m} |\beta_j| \leq t$$

Provided that the LASSO parameter t is small enough, some of the regression coefficients are exactly 0. Hence, you can view the LASSO as selecting a subset of the regression coefficients for each LASSO parameter. By increasing the LASSO parameter in discrete steps, you obtain a sequence of regression coefficients in which the nonzero coefficients at each step correspond to selected parameters.

Early implementations (Tibshirani 1996) of LASSO selection used quadratic programming techniques to solve the constrained least squares problem for each LASSO parameter of interest. Later Osborne, Presnell, and Turlach (2000) developed a "homotopy method" that generates the LASSO solutions for all values of t. Efron et al. (2004) derived a variant of their algorithm for least angle regression that can be used to obtain a sequence of LASSO solutions from which all other LASSO solutions can be obtained by linear interpolation. This algorithm for METHOD=LASSO is used in PROC HPREG. It can be viewed as a stepwise procedure with a single addition to or deletion from the set of nonzero regression coefficients at any step.

As in the other selection methods that are supported by high-performance statistical procedures, you can use the CHOOSE= option to specify a criterion to choose among the models at each step of the LASSO algorithm. You can also use the STOP= option to specify a stopping criterion. For more information, see the discussion in the section "Forward Selection" on page 63. The model degrees of freedom that PROC GLMSELECT uses at any step of the LASSO are simply the number of nonzero regression coefficients in the model at that step. Efron et al. (2004) cite empirical evidence for doing this but do not give any mathematical justification for this choice.

A modification of LASSO selection suggested in Efron et al. (2004) uses the LASSO algorithm to select the set of covariates in the model at any step, but it uses ordinary least squares regression and just these covariates to obtain the regression coefficients. You can request this hybrid method by specifying the LSCOEFFS suboption of SELECTION=LASSO.

Adaptive Lasso Selection

Adaptive lasso selection is a modification of lasso selection; in adaptive lasso selection, weights are applied to each of the parameters in forming the lasso constraint (Zou 2006). More precisely, suppose that the response y has mean 0 and the regressors x are scaled to have mean 0 and common standard deviation. Furthermore, suppose that you can find a suitable estimator $\hat{\beta}$ of the parameters in the true model and you define a weight

vector by $w = 1/|\hat{\beta}|^{\gamma}$, where $\gamma \geq 0$. Then the adaptive lasso regression coefficients $\beta = (\beta_1, \beta_2, \ldots, \beta_m)$ are the solution to the following constrained optimization problem:

$$\text{minimize}||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||^2 \qquad \text{subject to} \quad \sum_{j=1}^{m} |w_j \beta_j| \leq t$$

PROC HPREG uses the solution to the unconstrained least squares problem as the estimator $\hat{\beta}$. This is appropriate unless collinearity is a concern. If the regressors are collinear or nearly collinear, then Zou (2006) suggests using a ridge regression estimate to form the adaptive weights.

References

Burnham, K. P. and Anderson, D. R. (2002), *Model Selection and Multimodel Inference*, 2nd Edition, New York: Springer-Verlag.

Efron, B., Hastie, T. J., Johnstone, I. M., and Tibshirani, R. (2004), "Least Angle Regression (with Discussion)," *Annals of Statistics*, 32, 407–499.

Lawless, J. F. and Singhal, K. (1978), "Efficient Screening of Nonnormal Regression Models," *Biometrics*, 34, 318–327.

Osborne, M. R., Presnell, B., and Turlach, B. A. (2000), "A New Approach to Variable Selection in Least Squares Problems," *IMA Journal of Numerical Analysis*, 20, 389–404.

Searle, S. R. (1971), *Linear Models*, New York: John Wiley & Sons.

Tibshirani, R. (1996), "Regression Shrinkage and Selection via the Lasso," *Journal of the Royal Statistical Society, Series B*, 58, 267–288.

Zou, H. (2006), "The Adaptive Lasso and Its Oracle Properties," *Journal of the American Statistical Association*, 101, 1418–1429.

Chapter 5
The HPCANDISC Procedure

Contents

Overview: HPCANDISC Procedure **73**
PROC HPCANDISC Features 75
PROC HPCANDISC Compared with PROC CANDISC 75
Getting Started: HPCANDISC Procedure **76**
Syntax: HPCANDISC Procedure **80**
PROC HPCANDISC Statement 81
BY Statement 84
CLASS Statement 85
FREQ Statement 85
ID Statement 85
PERFORMANCE Statement 85
VAR Statement 86
WEIGHT Statement 86
Details: HPCANDISC Procedure **86**
Missing Values 86
Computational Method 86
General Formulas 86
Multithreading 87
Output Data Sets 88
OUT= Data Set 88
OUTSTAT= Data Set 89
Displayed Output 91
ODS Table Names 93
Examples: HPCANDISC Procedure **94**
Example 5.1: Analyzing Iris Data with PROC HPCANDISC 94
Example 5.2: Performing Canonical Discriminant Analysis in Single-Machine and Distributed Modes 100
References **103**

Overview: HPCANDISC Procedure

The HPCANDISC procedure is a high-performance procedure that performs canonical discriminant analysis. It is a high-performance version of the CANDISC procedure in SAS/STAT software. PROC HPCANDISC runs in either single-machine mode or distributed mode.

NOTE: Distributed mode requires SAS High-Performance Statistics.

Canonical discriminant analysis is a dimension-reduction technique related to principal component analysis and canonical correlation. The methodology that is used in deriving the canonical coefficients parallels that of a one-way multivariate analysis of variance (MANOVA). MANOVA tests for equality of the mean vector across class levels. Canonical discriminant analysis finds linear combinations of the quantitative variables that provide maximal separation between classes or groups. Given a classification variable and several quantitative variables, the HPCANDISC procedure derives *canonical variables*, which are linear combinations of the quantitative variables that summarize between-class variation in much the same way that principal components summarize total variation.

The HPCANDISC procedure performs a canonical discriminant analysis, computes squared Mahalanobis distances between class means, and performs both univariate and one-way multivariate analyses of variance. Two output data sets can be produced: one that contains the canonical coefficients and another that contains, among other things, scored canonical variables. You can rotate the canonical coefficients output data set by using the FACTOR procedure. It is customary to standardize the canonical coefficients so that the canonical variables have means that are equal to 0 and pooled within-class variances that are equal to 1. PROC HPCANDISC displays both standardized and unstandardized canonical coefficients. Correlations between the canonical variables and the original variables in addition to the class means for the canonical variables are also displayed; these correlations, sometimes known as loadings, are called *canonical structures*.

When you have two or more groups of observations that have measurements on several quantitative variables, canonical discriminant analysis derives a linear combination of the variables that has the highest possible multiple correlation with the groups. This maximal multiple correlation is called the *first canonical correlation*. The coefficients of the linear combination are the *canonical coefficients* or *canonical weights*. The variable that is defined by the linear combination is the *first canonical variable* or *canonical component*. The second canonical correlation is obtained by finding the linear combination uncorrelated with the first canonical variable that has the highest possible multiple correlation with the groups. The process of extracting canonical variables can be repeated until the number of canonical variables equals the number of original variables or the number of classes minus one, whichever is smaller.

The first canonical correlation is at least as large as the multiple correlation between the groups and any of the original variables. If the original variables have high within-group correlations, the first canonical correlation can be large even if all the multiple correlations are small. In other words, the first canonical variable can show substantial differences between the classes, even if none of the original variables do. Canonical variables are sometimes called *discriminant functions*, but this usage is ambiguous because the DISCRIM procedure produces very different functions for classification that are also called discriminant functions.

For each canonical correlation, PROC HPCANDISC tests the hypothesis that it and all smaller canonical correlations are zero in the population. An F approximation (Rao 1973; Kshirsagar 1972) is used that gives better small-sample results than the usual chi-square approximation. The variables should have an approximate multivariate normal distribution within each class, with a common covariance matrix in order for the probability levels to be valid.

Canonical discriminant analysis is equivalent to canonical correlation analysis between the quantitative variables and a set of dummy variables coded from the class variable. Performing canonical discriminant analysis is also equivalent to performing the following steps:

1. Transform the variables so that the pooled within-class covariance matrix is an identity matrix.

2. Compute class means on the transformed variables.
3. Perform a principal component analysis on the means, weighting each mean by the number of observations in the class. The eigenvalues are equal to the ratio of between-class variation to within-class variation in the direction of each principal component.
4. Back-transform the principal components into the space of the original variables to obtain the canonical variables.

An interesting property of the canonical variables is that they are uncorrelated whether the correlation is calculated from the total sample or from the pooled within-class correlations. However, the canonical coefficients are not orthogonal, so the canonical variables do not represent perpendicular directions through the space of the original variables.

PROC HPCANDISC Features

The main features of the HPCANDISC procedure are as follows:

- performs a canonical discriminant analysis, computes squared Mahalanobis distances between class means, and performs both univariate and multivariate one-way analyses of variance
- can perform analysis on a massively parallel SAS high-performance appliance
- reads input data in parallel and writes output data in parallel when the data source is the appliance database
- is highly multithreaded during calculations of the within-class sum-of-squares-and-crossproducts (SSCP) matrix and the canonical variable scores
- supports a FREQ statement for grouped analysis
- supports a WEIGHT statement for weighted analysis
- displays both standardized and unstandardized canonical coefficients
- displays correlations between the canonical variables and the original variables
- displays class means for the canonical variables
- produces two output data sets: one that contains the canonical coefficients and another that contains scored canonical variables

PROC HPCANDISC Compared with PROC CANDISC

The HPCANDISC procedure and the CANDISC procedure in SAS/STAT have the following similarities and differences:

- All the statements that are available in PROC CANDISC are supported in the HPCANDISC procedure.
- As input, PROC CANDISC can accept ordinary SAS data set and other types of special SAS data sets. In the HPCANDISC procedure, only the ordinary SAS data set (raw data) can be used as input.
- The HPCANDISC procedure supports an ID statement that is not available in PROC CANDISC.
- The HPCANDISC procedure is specifically designed to operate in the high-performance distributed environment. By default, PROC HPCANDISC performs computations on multiple threads. The CANDISC procedure executes on a single thread.

Getting Started: HPCANDISC Procedure

The data in this example are measurements of 159 fish caught in Finland's Lake Laengelmaevesi; this data set is available from the Puranen. For each of the seven species (bream, roach, whitefish, parkki, perch, pike, and smelt), the weight, length, height, and width of each fish are tallied. Three different length measurements are recorded: from the nose of the fish to the beginning of its tail, from the nose to the notch of its tail, and from the nose to the end of its tail. The height and width are recorded as percentages of the third length variable. The fish data set is available from the Sashelp library.

The following step uses PROC HPCANDISC to find the three canonical variables that best separate the species of fish in the Sashelp.Fish data and create the output data set outcan. When the NCAN=3 option is specified, only the first three canonical variables are displayed. The ID statement adds the variable Species from the input data set to the output data set. The ODS EXCLUDE statement excludes the canonical structure tables and most of the canonical coefficient tables in order to obtain a more compact set of results. The TEMPLATE and SGRENDER procedures create a plot of the first two canonical variables. The following statements produce Figure 5.1 through Figure 5.6:

```
title 'Fish Measurement Data';

proc hpcandisc data=sashelp.fish ncan=3 out=outcan;
   ods exclude tstruc bstruc pstruc tcoef pcoef;
   id Species;
   class Species;
   var Weight Length1 Length2 Length3 Height Width;
run;

proc template;
   define statgraph scatter;
      begingraph;
         entrytitle 'Fish Measurement Data';
         layout overlayequated / equatetype=fit
            xaxisopts=(label='Canonical Variable 1')
            yaxisopts=(label='Canonical Variable 2');
            scatterplot x=Can1 y=Can2 / group=species name='fish';
            layout gridded / autoalign=(topright);
               discretelegend 'fish' / border=false opaque=false;
            endlayout;
         endlayout;
```

```
      endgraph;
   end;
run;

proc sgrender data=outcan template=scatter;
run;
```

PROC HPCANDISC begins by displaying performance information, data access information, and summary information about the variables in the analysis, as shown in Figure 5.1.

The "Performance Information" table shows the procedure executes in single-machine mode; that is, the data reside and the computation is conducted on the machine where the SAS session executes. This run of the HPCANDISC procedure took place on a multicore machine that had four CPUs; one computational thread was spawned per CPU.

The "Data Access Information" table shows that the input data set and the output data set are both accessed with the V9 (base) engine on the client machine where the MVA SAS session executes.

The summary information includes the number of observations, the number of quantitative variables in the analysis (specified using the VAR statement), and the number of class levels in the classification variable (specified using the CLASS statement). The value and frequency of each class level are also displayed.

Figure 5.1 Fish Data: Performance, Data Access, and Summary Information

Fish Measurement Data

The HPCANDISC Procedure

Performance Information	
Execution Mode	Single-Machine
Number of Threads	4

Data Access Information			
Data	**Engine**	**Role**	**Path**
SASHELP.FISH	V9	Input	On Client
WORK.OUTCAN	V9	Output	On Client

Total Sample Size	158	**DF Total**	157
Variables	6	**DF Within Classes**	151
Class Levels	7	**DF Between Classes**	6

Number of Observations Read	159
Number of Observations Used	158

Class Level Information			
Species	**Frequency**	**Weight**	**Proportion**
Bream	34	34.00000	0.21519
Parkki	11	11.00000	0.06962
Perch	56	56.00000	0.35443
Pike	17	17.00000	0.10759
Roach	20	20.00000	0.12658
Smelt	14	14.00000	0.08861
Whitefish	6	6.00000	0.03797

Figure 5.2 displays the "Multivariate Statistics and F Approximations" table. PROC HPCANDISC performs a one-way multivariate analysis of variance (one-way MANOVA) and provides four multivariate tests of the hypothesis that the class mean vectors are equal. These tests indicate that not all the mean vectors are equal ($p < 0.0001$).

Figure 5.2 Fish Data: MANOVA and Multivariate Tests

Fish Measurement Data

The HPCANDISC Procedure

Multivariate Statistics and F Approximations

S=6 M=-0.5 N=72

Statistic	Value	F Value	Num DF	Den DF	Pr > F
Wilks' Lambda	0.000363	90.71	36	643.89	<.0001
Pillai's Trace	3.104651	26.99	36	906	<.0001
Hotelling-Lawley Trace	52.057997	209.24	36	413.64	<.0001
Roy's Greatest Root	39.134998	984.90	6	151	<.0001

NOTE: F Statistic for Roy's Greatest Root is an upper bound.

Figure 5.3 displays the "Canonical Correlations" table. The first canonical correlation is the greatest possible multiple correlation with the classes that you can achieve by using a linear combination of the quantitative variables. The first canonical correlation, displayed in the table, is 0.987463. The figure shows a likelihood ratio test of the hypothesis that the current canonical correlation and all smaller ones are zero. The first line is equivalent to Wilks' lambda multivariate test.

Figure 5.3 Fish Data: Canonical Correlations

Fish Measurement Data

The HPCANDISC Procedure

	Canonical Correlation	Adjusted Canonical Correlation	Approximate Standard Error	Squared Canonical Correlation	Eigenvalues of Inv(E)*H = CanRsq/(1-CanRsq) Eigenvalue	Difference	Proportion	Cumulative
1	0.987463	0.986671	0.001989	0.975084	39.1350	29.3859	0.7518	0.7518
2	0.952349	0.950095	0.007425	0.906969	9.7491	7.3786	0.1873	0.9390
3	0.838637	0.832518	0.023678	0.703313	2.3706	1.7016	0.0455	0.9846
4	0.633094	0.623649	0.047821	0.400809	0.6689	0.5346	0.0128	0.9974
5	0.344157	0.334170	0.070356	0.118444	0.1344	0.1343	0.0026	1.0000
6	0.005701	.	0.079806	0.000033	0.0000		0.0000	1.0000

Test of H0: The canonical correlations in the current row and all that follow are zero

	Likelihood Ratio	Approximate F Value	Num DF	Den DF	Pr > F
1	0.00036325	90.71	36	643.89	<.0001
2	0.01457896	46.46	25	547.58	<.0001
3	0.15671134	23.61	16	452.79	<.0001
4	0.52820347	12.09	9	362.78	<.0001
5	0.88152702	4.88	4	300	0.0008
6	0.99996749	0.00	1	151	0.9442

Figure 5.4 displays the "Raw Canonical Coefficients" table. The first canonical variable, Can1, shows that the linear combination of the centered variables Can1 = –0.0006 × Weight – 0.33 × Length1 2.49 × Length2 + 2.60 × Length3 + 1.12 × Height – 1.45 × Width separates the species most effectively.

Figure 5.4 Fish Data: Raw Canonical Coefficients

Fish Measurement Data

The HPCANDISC Procedure

Raw Canonical Coefficients

Variable	Can1	Can2	Can3
Weight	-0.00064851	-0.00523	-0.00560
Length1	-0.32944	-0.62660	-2.93432
Length2	-2.48613	-0.69025	4.04504
Length3	2.59565	1.80318	-1.13926
Height	1.12198	-0.71475	0.28320
Width	-1.44639	-0.90703	0.74149

Figure 5.5 displays the "Class Means on Canonical Variables" table. PROC HPCANDISC computes the means of the canonical variables for each class. The first canonical variable is the linear combination of the variables Weight, Length1, Length2, Length3, Height, and Width that provides the greatest difference (in terms of a univariate F test) between the class means. The second canonical variable provides the greatest difference between class means while being uncorrelated with the first canonical variable.

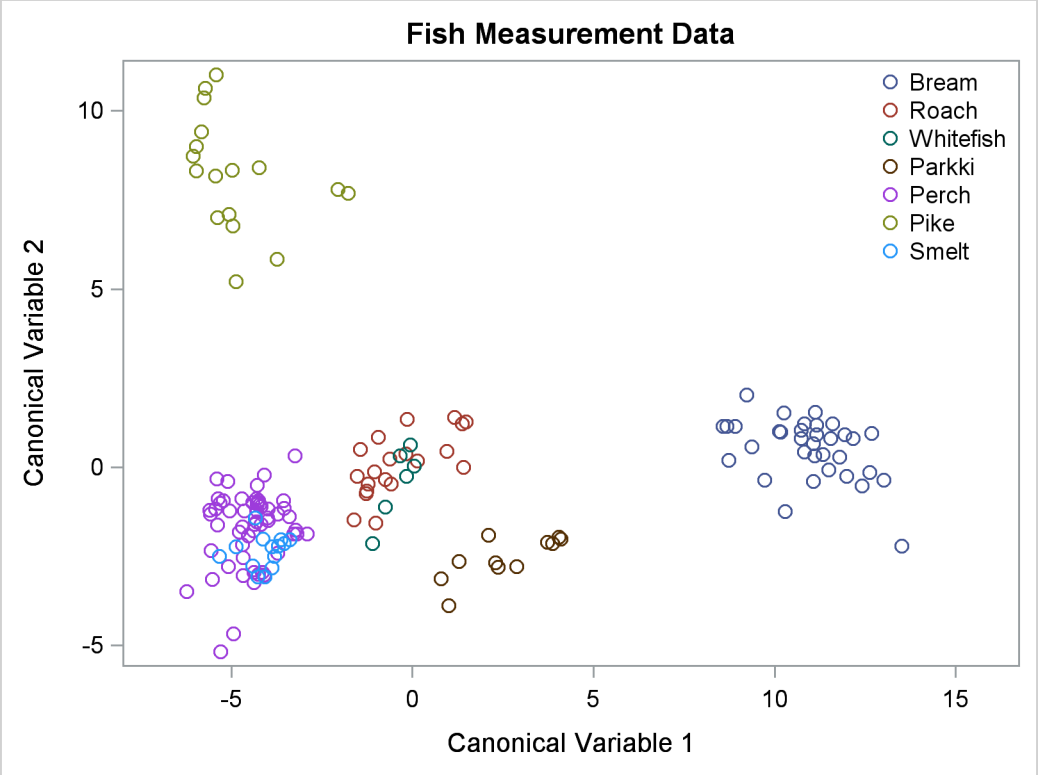
Figure 5.5 Fish Data: Class Means for Canonical Variables

Class Means on Canonical Variables

Species	Can1	Can2	Can3
Bream	10.94142	0.52078	0.23497
Parkki	2.58904	-2.54722	-0.49326
Perch	-4.47181	-1.70823	1.29281
Pike	-4.89689	8.22141	-0.16469
Roach	-0.35837	0.08734	-1.10056
Smelt	-4.09137	-2.35806	-4.03836
Whitefish	-0.39542	-0.42072	1.06459

Figure 5.6 displays a plot of the first two canonical variables, which shows that Can1 discriminates among three groups: (1) bream; (2) whitefish, roach, and parkki; and (3) smelt, pike, and perch. Can2 best discriminates between pike and the other species.

Figure 5.6 Fish Data: Plot of First Two Canonical Variables



Syntax: HPCANDISC Procedure

The following statements are available in the HPCANDISC procedure:

PROC HPCANDISC < *options* > ;
BY *variables* ;
CLASS *variable* ;
FREQ *variable* ;
ID *variables* ;
PERFORMANCE *performance-options* ;
VAR *variables* ;
WEIGHT *variable* ;

The PROC HPCANDISC statement and a single CLASS statement are required. All other statements are optional.

PROC HPCANDISC Statement

PROC HPCANDISC < *options* > **;**

The PROC HPCANDISC statement invokes the HPCANDISC procedure. Optionally, it also identifies input and output data sets, specifies the analyses performed, and controls displayed output. Table 5.1 summarizes the options available in the PROC HPCANDISC statement.

Table 5.1 PROC HPCANDISC Statement Options

Option	Description
Specify Data Sets	
DATA=	Specifies the input data set
OUT=	Specifies the output data set that contains canonical scores
OUTSTAT=	Specifies the output statistics data set
Specify Details of Analysis	
NCAN=	Specifies the number of canonical variables
PREFIX=	Specifies a prefix for naming the canonical variables
SINGULAR=	Specifies the singularity criterion
Control Displayed Output	
ALL	Displays all output
ANOVA	Displays univariate statistics
BCORR	Displays between correlations
BCOV	Displays between covariances
BSSCP	Displays between SSCPs
DISTANCE	Displays squared Mahalanobis distances
NOPRINT	Suppresses all displayed output
PCORR	Displays pooled correlations
PCOV	Displays pooled covariances
PSSCP	Displays pooled SSCPs
SHORT	Suppresses some displayed output
SIMPLE	Displays simple descriptive statistics
STDMEAN	Displays standardized class means
TCORR	Displays total correlations
TCOV	Displays total covariances
TSSCP	Displays total SSCPs
WCORR	Displays within correlations
WCOV	Displays within covariances
WSSCP	Displays within SSCPs

The following list provides details about these *options*.

ALL
activates all the display options.

ANOVA
displays univariate statistics for testing the hypothesis that the class means are equal in the population for each variable.

BCORR
displays between-class correlations.

BCOV
displays between-class covariances. The between-class covariance matrix equals the between-class SSCP matrix divided by $n(c-1)/c$, where n is the number of observations and c is the number of classes. The between-class covariances should be interpreted in comparison with the total-sample and within-class covariances, not as formal estimates of population parameters.

BSSCP
displays the between-class SSCP matrix.

DATA=*SAS-data-set*
specifies the data set to be analyzed. The data set can only be an ordinary SAS data set (raw data). If you omit the DATA= option, PROC HPCANDISC uses the most recently created SAS data set.

If PROC HPCANDISC executes in distributed mode, the input data are distributed to memory on the appliance nodes and analyzed in parallel, unless the data are already distributed in the appliance database. In that case the procedure reads the data alongside the distributed database. For more information, see the section "Processing Modes" on page 10 about the various execution modes and the section "Alongside-the-Database Execution" on page 18 about the alongside-the-database model in Chapter 3, "Shared Concepts and Topics."

DISTANCE
MAHALANOBIS
displays squared Mahalanobis distances between the group means, the F statistics, and the corresponding probabilities of greater squared Mahalanobis distances between the group means.

NCAN=*n*
specifies the number of canonical variables to be computed. The value of *n* must be less than or equal to the number of variables. If you specify NCAN=0, PROC HPCANDISC displays the canonical correlations but not the canonical coefficients, structures, or means. A negative value suppresses the canonical analysis entirely. Let v be the number of variables in the VAR statement, and let c be the number of classes. If you omit the NCAN= option, only $\min(v, c-1)$ canonical variables are generated; if you also specify an OUT= output data set, v canonical variables are generated, and the last $v-(c-1)$ canonical variables have missing values.

NOPRINT
suppresses the normal display of results. This option temporarily disables the Output Delivery System (ODS). For more information about ODS, see Chapter 20, "Using the Output Delivery System" (*SAS/STAT User's Guide*).

OUT=*SAS-data-set*
creates an output SAS data set to contain observationwise canonical variable scores. The variables in the input data set are *not* included in the output data set to avoid data duplication for large data sets; however, variables that are specified in the ID statement are included.

If the input data are in distributed form, in which access of data in a particular order cannot be guaranteed, the HPCANDISC procedure copies the distribution or partition key to the output data set so that its contents can be joined with the input data.

If you want to create a SAS data set in a permanent library, you must specify a two-level name. For more information about permanent libraries and SAS data sets, see *SAS Language Reference: Concepts*. For more information about OUT= data sets, see the section "Output Data Sets" on page 88.

OUTSTAT=*SAS-data-set*
creates a TYPE=CORR output SAS data set to contain various statistics, including class means, standard deviations, correlations, canonical correlations, canonical structures, canonical coefficients, and means of canonical variables for each class level.

If you want to create a SAS data set in a permanent library, you must specify a two-level name. For more information about permanent libraries and SAS data sets, see *SAS Language Reference: Concepts*.

PCORR
displays pooled within-class correlations (partial correlations based on the pooled within-class covariances).

PCOV
displays pooled within-class covariances.

PREFIX=*name*
specifies a prefix for naming the canonical variables. By default, the names are Can1, Can2, Can3, and so on. If you specify PREFIX=Abc, the components are named Abc1, Abc2, and so on. The number of characters in the prefix plus the number of digits required to designate the canonical variables should not exceed 32. The prefix is truncated if the combined length exceeds 32.

PSSCP
displays the pooled within-class corrected SSCP matrix.

SHORT
suppresses the display of canonical structures, canonical coefficients, and class means on canonical variables; only tables of canonical correlations and multivariate test statistics are displayed.

SIMPLE
displays simple descriptive statistics for the total sample and within each class.

SINGULAR=*p*
specifies the criterion for determining the singularity of the total-sample correlation matrix and the pooled within-class covariance matrix, where $0 < p < 1$. The default is SINGULAR=1E–8.

Let S be the total-sample correlation matrix. If the R square for predicting a quantitative variable in the VAR statement from the variables that precede it exceeds $1 - p$, then S is considered singular. If S is singular, the probability levels for the multivariate test statistics and canonical correlations are adjusted for the number of variables whose R square exceeds $1 - p$.

If **S** is considered singular and the inverse of **S** (squared Mahalanobis distances) is required, a quasi inverse is used instead. For more information, see the section "Quasi-inverse" (Chapter 35, *SAS/STAT User's Guide*).

STDMEAN
displays total-sample and pooled within-class standardized class means.

TCORR
displays total-sample correlations.

TCOV
displays total-sample covariances.

TSSCP
displays the total-sample corrected SSCP matrix.

WCORR
displays within-class correlations for each class level.

WCOV
displays within-class covariances for each class level.

WSSCP
displays the within-class corrected SSCP matrix for each class level.

BY Statement

BY *variables* **;**

You can specify a BY statement with PROC HPCANDISC to obtain separate analyses of observations in groups that are 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 you specify more than one BY statement, only the last one specified is used.

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 HPCANDISC 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 (in Base SAS software).

BY-statement processing is not supported when the HPCANDISC procedure runs alongside the database or alongside the Hadoop Distributed File System (HDFS). Along-the-database mode or alongside-the HDFS mode is used if the input data are stored in a database or HDFS and the grid host is the appliance that houses the data.

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

CLASS Statement

CLASS *variable* **;**

The values of the CLASS variable define the groups for analysis. Class levels are determined by the formatted values of the CLASS variable. The CLASS variable can be numeric or character. A CLASS statement is required.

FREQ Statement

FREQ *variable* **;**

The *variable* in the FREQ statement identifies a numeric variable in the data set that contains the frequency of occurrence of each observation. SAS high-performance analytics procedures that support the FREQ statement treat each observation as if it appeared f times, where f is the value of the FREQ variable for the observation. If the frequency value is not an integer, it is truncated to an integer. If the frequency value is less than 1 or missing, the observation is not used in the analysis. When the FREQ statement is not specified, each observation is assigned a frequency of 1.

The total number of observations is considered to be equal to the sum of the FREQ variable when the procedure determines degrees of freedom for significance probabilities.

ID Statement

ID *variables* **;**

The ID statement lists one or more variables from the input data set that are transferred to output data sets created by SAS high-performance analytics procedures, provided that the output data set produces one (or more) records per input observation.

For information about the common ID statement in SAS high-performance analytics procedures, see the section "ID Statement" on page 46 in Chapter 4, "Shared Statistical Concepts."

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 the HPCANDISC procedure.

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

The PERFORMANCE statement is documented further in the section "PERFORMANCE Statement" on page 36 of Chapter 3, "Shared Concepts and Topics."

VAR Statement

VAR *variables* **;**

You specify the quantitative variables to include in the analysis by using a VAR statement. If you do not use a VAR statement, the analysis includes all numeric variables that are not listed in other statements.

WEIGHT Statement

WEIGHT *variable* **;**

The *variable* in the WEIGHT statement is used as a weight to perform a weighted analysis of the data. Observations that have nonpositive or missing weights are not included in the analysis. If a WEIGHT statement is not included, all observations that are used in the analysis are assigned a weight of 1.

The WEIGHT statement does not alter the degrees of freedom.

Details: HPCANDISC Procedure

Missing Values

If an observation has a missing value for any of the quantitative variables, it is omitted from the analysis. If an observation has a missing CLASS value but is otherwise complete, PROC HPCANDISC does not use it in computing the canonical correlations and coefficients; however, canonical variable scores are computed for that observation for the OUT= data set.

Computational Method

General Formulas

Canonical discriminant analysis is equivalent to canonical correlation analysis between the quantitative variables and a set of dummy variables coded from the class variable. In the following notation, the dummy variables are denoted by $\mathbf{y}$ and the quantitative variables are denoted by $\mathbf{x}$. The total sample covariance matrix for the $\mathbf{x}$ and $\mathbf{y}$ variables is

$$\mathrm{S} = \begin{bmatrix} \mathrm{S}_{xx} & \mathrm{S}_{xy} \\ \mathrm{S}_{yx} & \mathrm{S}_{yy} \end{bmatrix}$$

When c is the number of groups, n_t is the number of observations in group t, and S_t is the sample covariance matrix for the $\mathbf{x}$ variables in group t, the within-class pooled covariance matrix for the $\mathbf{x}$ variables is

$$\mathrm{S}_p = \frac{1}{\sum n_t - c} \sum (n_t - 1)\mathrm{S}_t$$

The canonical correlations, ρ_i, are the square roots of the eigenvalues, λ_i, of the following matrix. The corresponding eigenvectors are $\mathbf{v}_i$.

$$\mathbf{S}_p{}^{-1/2}\mathbf{S}_{xy}\mathbf{S}_{yy}{}^{-1}\mathbf{S}_{yx}\mathbf{S}_p{}^{-1/2}$$

Let $\mathbf{V}$ be the matrix that contains the eigenvectors $\mathbf{v}_i$ that correspond to nonzero eigenvalues as columns. The raw canonical coefficients are calculated as follows:

$$\mathbf{R} = \mathbf{S}_p{}^{-1/2}\mathbf{V}$$

The pooled within-class standardized canonical coefficients are

$$\mathbf{P} = \mathrm{diag}(\mathbf{S}_p)^{1/2}\mathbf{R}$$

The total sample standardized canonical coefficients are

$$\mathbf{T} = \mathrm{diag}(\mathbf{S}_{xx})^{1/2}\mathbf{R}$$

Let $\mathbf{X}_c$ be the matrix that contains the centered $\mathbf{x}$ variables as columns. The canonical scores can be calculated by any of the following:

$$\mathbf{X}_c\,\mathbf{R}$$

$$\mathbf{X}_c\,\mathrm{diag}(\mathbf{S}_p)^{-1/2}\mathbf{P}$$

$$\mathbf{X}_c\,\mathrm{diag}(\mathbf{S}_{xx})^{-1/2}\mathbf{T}$$

For the multivariate tests based on $\mathbf{E}^{-1}\mathbf{H}$,

$$\mathbf{E} = (n-1)(\mathbf{S}_{yy} - \mathbf{S}_{yx}\mathbf{S}_{xx}^{-1}\mathbf{S}_{xy})$$

$$\mathbf{H} = (n-1)\mathbf{S}_{yx}\mathbf{S}_{xx}^{-1}\mathbf{S}_{xy}$$

where n is the total number of observations.

Multithreading

Threading is the organization of computational work into multiple tasks (processing units that can be scheduled by the operating system). A task is associated with a thread. Multithreading is the concurrent execution of threads. When multithreading is possible, you can realize substantial performance gains compared to the performance that you get from sequential (single-threaded) execution.

The number of threads that the HPCANDISC procedure spawns is determined by the number of CPUs on a machine and can be controlled in the following ways:

- You can specify the CPU count by using the CPUCOUNT= SAS system option. For example, if you specify the following statement, PROC HPCANDISC schedules threads as if it were executing on a system that had four CPUs, regardless of the actual CPU count:

```
options cpucount=4;
```

- You can specify the NTHREADS= option in the PERFORMANCE statement to determine the number of threads. This specification overrides the system option. Specify NTHREADS=1 to force single-threaded execution.

The number of threads per machine is displayed in the "Performance Information" table, which is part of the default output. The HPCANDISC procedure allocates one thread per CPU.

The tasks that are multithreaded by the HPCANDISC procedure are primarily defined by dividing the data processed on a single machine among the threads; that is, PROC HPCANDISC implements multithreading through a data-parallel model. For example, if the input data set has 1,000 observations and you are running on four threads, then 250 observations are associated with each thread. All operations that require access to the data are then multithreaded. These operations include the following:

- variable levelization
- formation of the crossproducts matrix
- canonical variable scoring of observations

Output Data Sets

OUT= Data Set

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

- new variables that are explicitly created for the OUT= data set
- variables that are listed in the ID statement
- distribution keys or hash keys that are transferred from the input data set

Having these variables and keys in the OUT= data set enables you to add output data set information that is necessary for subsequent SQL joins without copying the entire input data set to the output data set. For more information about output data sets that are produced when PROC HPCANDISC is run in distributed mode, see the section "Output Data Sets" on page 34 in Chapter 3, "Shared Concepts and Topics."

The new variables that are created for the OUT= data set contain the canonical variable scores. You determine the number of new variables by using the NCAN= option. The names of the new variables are formed as they are for the PREFIX= option. The new variables have means equal to 0 and pooled within-class variances equal to 1.

OUTSTAT= Data Set

The OUTSTAT= data set is similar to the TYPE=CORR data set that the CORR procedure produces but contains many results in addition to those produced by PROC CORR.

The OUTSTAT= data set is TYPE=CORR, and it contains the following variables:

- the BY variables, if any
- the CLASS variable
- _TYPE_, a character variable of length 8 that identifies the type of statistic
- _NAME_, a character variable of length 32 that identifies the row of the matrix or the name of the canonical variable
- the quantitative variables (those in the VAR statement, or if there is no VAR statement, all numeric variables not listed in any other statement)

The observations, as identified by the variable _TYPE_, have the following _TYPE_ values:

TYPE	**Contents**
N	number of observations for the total sample (CLASS variable missing) and within each class (CLASS variable present)
SUMWGT	sum of weights for the total sample (CLASS variable missing) and within each class (CLASS variable present) if a WEIGHT statement is specified
MEAN	means for the total sample (CLASS variable missing) and within each class (CLASS variable present)
STDMEAN	total-standardized class means
PSTDMEAN	pooled within-class standardized class means
STD	standard deviations for the total sample (CLASS variable missing) and within each class (CLASS variable present)
PSTD	pooled within-class standard deviations
BSTD	between-class standard deviations
RSQUARED	univariate R squares

The following kinds of observations are identified by the combination of the variables _TYPE_ and _NAME_. When the _TYPE_ variable has one of the following values, the _NAME_ variable identifies the row of the matrix:

TYPE	**Contents**
CSSCP	corrected SSCP matrix for the total sample (CLASS variable missing) and within each class (CLASS variable present)

PSSCP	pooled within-class corrected SSCP matrix
BSSCP	between-class SSCP matrix
COV	covariance matrix for the total sample (CLASS variable missing) and within each class (CLASS variable present)
PCOV	pooled within-class covariance matrix
BCOV	between-class covariance matrix
CORR	correlation matrix for the total sample (CLASS variable missing) and within each class (CLASS variable present)
PCORR	pooled within-class correlation matrix
BCORR	between-class correlation matrix

When the _TYPE_ variable has one of the following values, the _NAME_ variable identifies the canonical variable:

TYPE	**Contents**
CANCORR	canonical correlations
STRUCTUR	canonical structure
BSTRUCT	between canonical structure
PSTRUCT	pooled within-class canonical structure
SCORE	total-sample standardized canonical coefficients
PSCORE	pooled within-class standardized canonical coefficients
RAWSCORE	raw canonical coefficients
CANMEAN	means of the canonical variables for each class

You can use this data set in PROC SCORE to get scores on the canonical variables for new data by using one of the following forms:

```
* The CLASS variable C is numeric;
proc score data=NewData score=Coef(where=(c = .  )) out=Scores;
run;

* The CLASS variable C is character;
proc score data=NewData score=Coef(where=(c = ' ')) out=Scores;
run;
```

The WHERE clause excludes the within-class means and standard deviations. PROC SCORE standardizes the new data by subtracting the original variable means that are stored in the _TYPE_='MEAN' observations and dividing by the original variable standard deviations from the _TYPE_='STD' observations. Then PROC SCORE multiplies the standardized variables by the coefficients from the _TYPE_='SCORE' observations to get the canonical scores.

Displayed Output

By default, the HPCANDISC procedure begins by displaying the output along with the following:

- The "Performance Information" table, which is produced by default. It displays information about the execution mode. For single-machine mode, the table displays the number of threads used. For distributed mode, the table displays the grid mode (symmetric or asymmetric), the number of compute nodes, and the number of threads per node.
- The "Data Access Information" table, which is produced by default. For the input and output data sets, it displays the libref and data set name, the engine used to access the data, the role (input or output) of the data set, and the path that data followed to reach the computation.
- Summary information about the variables in the analysis that displays the total sample size, the number of quantitative variables, the number of class levels, and the number of degrees of freedom
- The "Number of Observations" table, which displays the number of observations read from the input data set and the number of observations used in the analysis. If you specify a FREQ statement, the table also displays the sum of frequencies read and used.
- The "Class Level Information" table, which displays, for each level of the classification variable, the frequency sum, weight sum, and proportion of the total sample

The optional output from PROC HPCANDISC includes the following:

- Within-class SSCP matrices for each group
- Pooled within-class SSCP matrix
- Between-class SSCP matrix
- Total-sample SSCP matrix
- Within-class covariance matrices for each group
- Pooled within-class covariance matrix
- Between-class covariance matrix, equal to the between-class SSCP matrix divided by $n(c-1)/c$, where n is the number of observations and c is the number of classes
- Total-sample covariance matrix
- Within-class correlation coefficients and $\Pr > |r|$ to test the hypothesis that the within-class population correlation coefficients are zero
- Pooled within-class correlation coefficients and $\Pr > |r|$ to test the hypothesis that the partial population correlation coefficients are zero
- Between-class correlation coefficients and $\Pr > |r|$ to test the hypothesis that the between-class population correlation coefficients are zero

- Total-sample correlation coefficients and $\Pr > |r|$ to test the hypothesis that the total population correlation coefficients are zero
- Simple statistics, including N (the number of observations), sum, mean, variance, and standard deviation for the total sample and within each class
- Total-sample standardized class means, obtained by subtracting the grand mean from each class mean and dividing by the total sample standard deviation
- Pooled within-class standardized class means, obtained by subtracting the grand mean from each class mean and dividing by the pooled within-class standard deviation
- Pairwise squared distances between groups
- Univariate test statistics, including total-sample standard deviations, pooled within-class standard deviations, between-class standard deviations, R square, $R^2/(1-R^2)$, F, and $\Pr > F$ (univariate F values and probability levels for one-way analyses of variance)
- The "Timing" table, which displays the elapsed time for each main task of the procedure, if you specify the DETAILS option in the PERFORMANCE statement

By default, PROC HPCANDISC displays these statistics:

- Multivariate statistics and F approximations, including Wilks' lambda, Pillai's trace, Hotelling-Lawley trace, and Roy's greatest root with F approximations, numerator and denominator degrees of freedom (Num DF and Den DF), and probability values ($\Pr > F$). Each of these four multivariate statistics tests the hypothesis that the class means are equal in the population. For more information, see the section "Multivariate Tests" (Chapter 4, *SAS/STAT User's Guide*).
- Canonical correlations
- Adjusted canonical correlations (Lawley 1959). These are asymptotically less biased than the raw correlations and can be negative. The adjusted canonical correlations might not be computable and are displayed as missing values if two canonical correlations are nearly equal or if some are close to zero. A missing value is also displayed if an adjusted canonical correlation is larger than a previous adjusted canonical correlation.
- Approximate standard error of the canonical correlations
- Squared canonical correlations
- Eigenvalues of $\mathbf{E}^{-1}\mathbf{H}$. Each eigenvalue is equal to $\rho^2/(1-\rho^2)$, where ρ^2 is the corresponding squared canonical correlation and can be interpreted as the ratio of between-class variation to pooled within-class variation for the corresponding canonical variable. The table includes eigenvalues, differences between successive eigenvalues, the proportion of the sum of the eigenvalues, and the cumulative proportion.
- Likelihood ratio for the hypothesis that the current canonical correlation and all smaller ones are zero in the population. The likelihood ratio for the hypothesis that all canonical correlations equal zero is Wilks' lambda.

- Approximate F statistic based on Rao's approximation to the distribution of the likelihood ratio (Rao 1973, p. 556; Kshirsagar 1972, p. 326)
- Numerator degrees of freedom (Num DF), denominator degrees of freedom (Den DF), and $\Pr > F$, the probability level associated with the F statistic

You can suppress the following statistics by specifying the SHORT option:

- Total canonical structure, giving total-sample correlations between the canonical variables and the original variables
- Between canonical structure, giving between-class correlations between the canonical variables and the original variables
- Pooled within canonical structure, giving pooled within-class correlations between the canonical variables and the original variables
- Total-sample standardized canonical coefficients, standardized to give canonical variables that have zero mean and unit pooled within-class variance when applied to the total-sample standardized variables
- Pooled within-class standardized canonical coefficients, standardized to give canonical variables that have zero mean and unit pooled within-class variance when applied to the pooled within-class standardized variables
- Raw canonical coefficients, standardized to give canonical variables that have zero mean and unit pooled within-class variance when applied to the centered variables
- Class means on the canonical variables

ODS Table Names

PROC HPCANDISC assigns a name to each table that it creates. You can use these names to reference the ODS table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in Table 5.2. For more information about ODS, see Chapter 20, "Using the Output Delivery System" (*SAS/STAT User's Guide*).

Table 5.2 ODS Tables Produced by PROC HPCANDISC

Table Name	Description	Required Statement / Option
ANOVA	Univariate statistics	ANOVA
AveRSquare	Average R square	ANOVA
BCorr	Between-class correlations	BCORR
BCov	Between-class covariances	BCOV
BSSCP	Between-class SSCP matrix	BSSCP
BStruc	Between canonical structure	Default
CanCorr	Canonical correlations	Default
CanonicalMeans	Class means on canonical variables	Default

Table 5.2 *continued*

Table Name	Description	Required Statement / Option
ClassLevels	Class level information	Default
Counts	Number of observations, variables, class levels, *df*	Default
DataAccessInfo	Information about modes of data access	Default
Dist	Squared distances	DISTANCE
DistFValues	*F* statistics based on squared distances	DISTANCE
DistProb	Probabilities for *F* statistics from squared distances	DISTANCE
MultStat	MANOVA	Default
NObs	Number of observations	Default
PCoef	Pooled standard canonical coefficients	Default
PCorr	Pooled within-class correlations	PCORR
PCov	Pooled within-class covariances	PCOV
PerformanceInfo	Information about the high-performance computing environment	Default
PSSCP	Pooled within-class SSCP matrix	PSSCP
PStdMeans	Pooled standardized class means	STDMEAN
PStruc	Pooled within canonical structure	Default
RCoef	Raw canonical coefficients	Default
SimpleStatistics	Simple statistics	SIMPLE
TCoef	Total-sample standard canonical coefficients	Default
TCorr	Total-sample correlations	TCORR
TCov	Total-sample covariances	TCOV
Timing	Absolute and relative times for tasks performed by the procedure	PERFORMANCE DETAILS
TSSCP	Total-sample SSCP matrix	TSSCP
TStdMeans	Total standardized class means	STDMEAN
TStruc	Total canonical structure	Default
WCorr	Within-class correlations	WCORR
WCov	Within-class covariances	WCOV
WSSCP	Within-class SSCP matrices	WSSCP

Examples: HPCANDISC Procedure

Example 5.1: Analyzing Iris Data with PROC HPCANDISC

The iris data that were published by Fisher (1936) have been widely used for examples in discriminant analysis and cluster analysis. The sepal length, sepal width, petal length, and petal width are measured in millimeters in 50 iris specimens from each of three species: *Iris setosa*, *I. versicolor*, and *I. virginica*. The iris data set is available from the Sashelp library.

This example is a canonical discriminant analysis that creates an output data set that contains scores on the canonical variables and plots the canonical variables. The ID statement is specified to add the variable Species from the input data set to the output data set.

The following statements produce Output 5.1.1 through Output 5.1.6:

```
title 'Fisher (1936) Iris Data';

proc hpcandisc data=sashelp.iris out=outcan distance anova;
   id Species;
   class Species;
   var SepalLength SepalWidth PetalLength PetalWidth;
run;
```

Output 5.1.1 displays performance information, data access information, and summary information about the observations and the classes in the data set.

Output 5.1.1 Iris Data: Performance, Data Access, and Summary Information

Fisher (1936) Iris Data

The HPCANDISC Procedure

Performance Information	
Execution Mode	Single-Machine
Number of Threads	4

Data Access Information			
Data	**Engine**	**Role**	**Path**
SASHELP.IRIS	V9	Input	On Client
WORK.OUTCAN	V9	Output	On Client

Total Sample Size	150	**DF Total**	149
Variables	4	**DF Within Classes**	147
Class Levels	3	**DF Between Classes**	2

Number of Observations Read	150
Number of Observations Used	150

Class Level Information			
Species	**Frequency**	**Weight**	**Proportion**
Setosa	50	50.00000	0.33333
Versicolor	50	50.00000	0.33333
Virginica	50	50.00000	0.33333

Output 5.1.2 shows results from the DISTANCE option in the PROC HPCANDISC statement, which display squared Mahalanobis distances between class means.

Output 5.1.2 Iris Data: Squared Mahalanobis Distances and Distance Statistics

Fisher (1936) Iris Data

The HPCANDISC Procedure

Squared Distance to Species

From Species	Setosa	Versicolor	Virginica
Setosa	0	89.86419	179.38471
Versicolor	89.86419	0	17.20107
Virginica	179.38471	17.20107	0

F Statistics, Num DF=4, Den DF=144 for Squared Distance to Species

From Species	Setosa	Versicolor	Virginica
Setosa	0	550.18889	1098.27375
Versicolor	550.18889	0	105.31265
Virginica	1098.27375	105.31265	0

Prob > Mahalanobis Distance for Squared Distance to Species

From Species	Setosa	Versicolor	Virginica
Setosa	1.0000	<.0001	<.0001
Versicolor	<.0001	1.0000	<.0001
Virginica	<.0001	<.0001	1.0000

Output 5.1.3 displays univariate and multivariate statistics. The ANOVA option uses univariate statistics to test the hypothesis that the class means are equal. The resulting R-square values range from 0.4008 for SepalWidth to 0.9414 for PetalLength, and each variable is significant at the 0.0001 level. The multivariate test for differences between the class levels (which is displayed by default) is also significant at the 0.0001 level; you would expect this from the highly significant univariate test results.

Output 5.1.3 Iris Data: Univariate and Multivariate Statistics

Fisher (1936) Iris Data

The HPCANDISC Procedure

Univariate Test Statistics

F Statistics, Num DF=2, Den DF=147

Variable	Label	Total Standard Deviation	Pooled Standard Deviation	Between Standard Deviation	R-Square	R-Square / (1-Rsq)	F Value	Pr > F
SepalLength	Sepal Length (mm)	8.28066	5.14789	7.95061	0.6187	1.6226	119.26	<.0001
SepalWidth	Sepal Width (mm)	4.35866	3.39688	3.36822	0.4008	0.6688	49.16	<.0001
PetalLength	Petal Length (mm)	17.65298	4.30334	20.90700	0.9414	16.0566	1180.16	<.0001
PetalWidth	Petal Width (mm)	7.62238	2.04650	8.96735	0.9289	13.0613	960.01	<.0001

Output 5.1.3 *continued*

Average R-Square	
Unweighted	0.7224358
Weighted by Variance	0.8689444

Multivariate Statistics and F Approximations

S=2 M=0.5 N=71

Statistic	Value	F Value	Num DF	Den DF	Pr > F
Wilks' Lambda	0.023439	199.15	8	288	<.0001
Pillai's Trace	1.191899	53.47	8	290	<.0001
Hotelling-Lawley Trace	32.477320	582.20	8	203.4	<.0001
Roy's Greatest Root	32.191929	1166.96	4	145	<.0001

NOTE: F Statistic for Roy's Greatest Root is an upper bound.

NOTE: F Statistic for Wilks' Lambda is exact.

Output 5.1.4 displays canonical correlations and eigenvalues. The R square between Can1 and the class variable, 0.969872, is much larger than the corresponding R square for Can2, 0.222027.

Output 5.1.4 Iris Data: Canonical Correlations and Eigenvalues

Fisher (1936) Iris Data

The HPCANDISC Procedure

	Canonical Correlation	Adjusted Canonical Correlation	Approximate Standard Error	Squared Canonical Correlation	Eigenvalues of Inv(E)*H = CanRsq/(1-CanRsq) Eigenvalue	Difference	Proportion	Cumulative
1	0.984821	0.984508	0.002468	0.969872	32.1919	31.9065	0.9912	0.9912
2	0.471197	0.461445	0.063734	0.222027	0.2854		0.0088	1.0000

Test of H0: The canonical correlations in the current row and all that follow are zero

	Likelihood Ratio	Approximate F Value	Num DF	Den DF	Pr > F
1	0.02343863	199.15	8	288	<.0001
2	0.77797337	13.79	3	145	<.0001

Output 5.1.5 displays correlations between canonical and original variables.

Output 5.1.5 Iris Data: Correlations between Canonical and Original Variables

Fisher (1936) Iris Data

The HPCANDISC Procedure

Total Canonical Structure

Variable	Label	Can1	Can2
SepalLength	Sepal Length (mm)	0.79189	0.21759
SepalWidth	Sepal Width (mm)	-0.53076	0.75799
PetalLength	Petal Length (mm)	0.98495	0.04604
PetalWidth	Petal Width (mm)	0.97281	0.22290

Between Canonical Structure

Variable	Label	Can1	Can2
SepalLength	Sepal Length (mm)	0.99147	0.13035
SepalWidth	Sepal Width (mm)	-0.82566	0.56417
PetalLength	Petal Length (mm)	0.99975	0.02236
PetalWidth	Petal Width (mm)	0.99404	0.10898

Pooled Within Canonical Structure

Variable	Label	Can1	Can2
SepalLength	Sepal Length (mm)	0.22260	0.31081
SepalWidth	Sepal Width (mm)	-0.11901	0.86368
PetalLength	Petal Length (mm)	0.70607	0.16770
PetalWidth	Petal Width (mm)	0.63318	0.73724

Output 5.1.6 displays canonical coefficients. The raw canonical coefficients for the first canonical variable, Can1, show that the class levels differ most widely on the linear combination of the centered variables: $-0.0829378 \times \text{SepalLength} - 0.153447 \times \text{SepalWidth} + 0.220121 \times \text{PetalLength} + 0.281046 \times \text{PetalWidth}$.

Output 5.1.6 Iris Data: Canonical Coefficients

Fisher (1936) Iris Data

The HPCANDISC Procedure

Total-Sample Standardized Canonical Coefficients

Variable	Label	Can1	Can2
SepalLength	Sepal Length (mm)	-0.68678	0.01996
SepalWidth	Sepal Width (mm)	-0.66883	0.94344
PetalLength	Petal Length (mm)	3.88580	-1.64512
PetalWidth	Petal Width (mm)	2.14224	2.16414

Pooled Within-Class Standardized Canonical Coefficients

Variable	Label	Can1	Can2
SepalLength	Sepal Length (mm)	-0.42695	0.01241
SepalWidth	Sepal Width (mm)	-0.52124	0.73526
PetalLength	Petal Length (mm)	0.94726	-0.40104
PetalWidth	Petal Width (mm)	0.57516	0.58104

Output 5.1.6 *continued*

Raw Canonical Coefficients			
Variable	**Label**	**Can1**	**Can2**
SepalLength	Sepal Length (mm)	-0.08294	0.00241
SepalWidth	Sepal Width (mm)	-0.15345	0.21645
PetalLength	Petal Length (mm)	0.22012	-0.09319
PetalWidth	Petal Width (mm)	0.28105	0.28392

Output 5.1.7 displays class means on canonical variables.

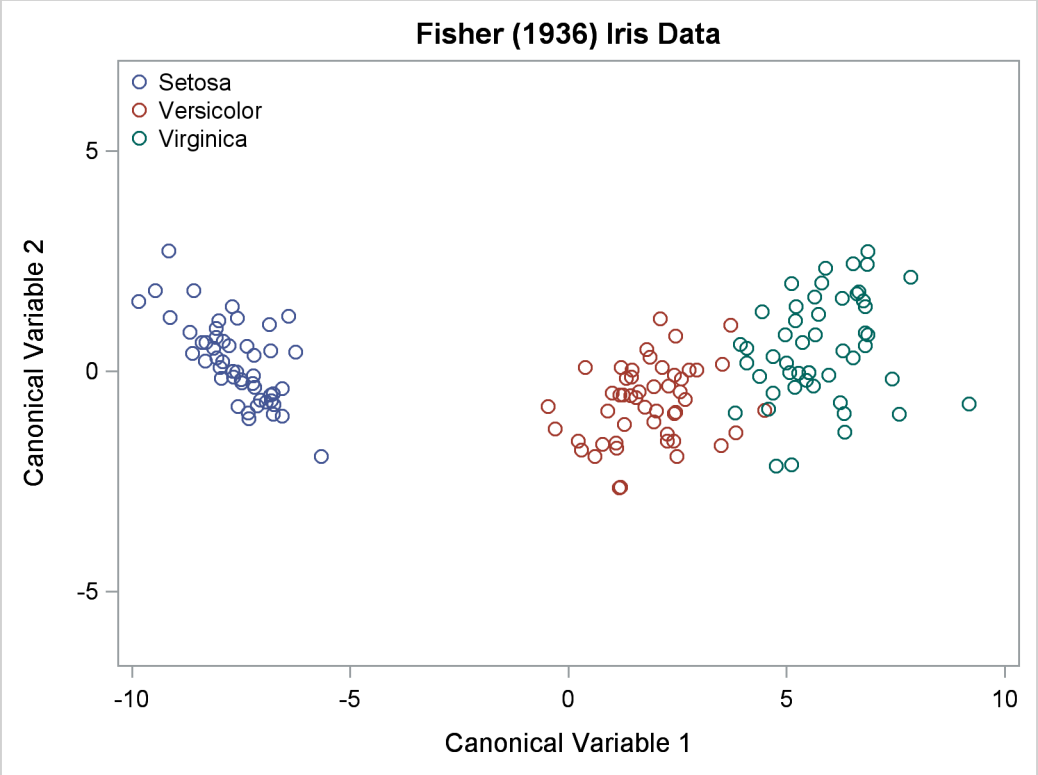
Output 5.1.7 Iris Data: Canonical Means

Class Means on Canonical Variables		
Species	**Can1**	**Can2**
Setosa	-7.60760	0.21513
Versicolor	1.82505	-0.72790
Virginica	5.78255	0.51277

The TEMPLATE and SGRENDER procedures are used to create a plot of the first two canonical variables. The following statements produce Output 5.1.8:

```
proc template;
   define statgraph scatter;
      begingraph;
         entrytitle 'Fisher (1936) Iris Data';
         layout overlayequated / equatetype=fit
            xaxisopts=(label='Canonical Variable 1')
            yaxisopts=(label='Canonical Variable 2');
            scatterplot x=Can1 y=Can2 / group=species name='iris';
            layout gridded / autoalign=(topleft);
               discretelegend 'iris' / border=false opaque=false;
            endlayout;
         endlayout;
      endgraph;
   end;
run;

proc sgrender data=outcan template=scatter;
run;
```

Output 5.1.8 Iris Data: Plot of First Two Canonical Variables



The plot of canonical variables in Output 5.1.8 shows that of the two canonical variables, Can1 has more discriminatory power.

Example 5.2: Performing Canonical Discriminant Analysis in Single-Machine and Distributed Modes

PROC HPCANDISC shows its real power when the computation is conducted using multiple threads or in a distributed environment.

This example shows how you can run PROC HPCANDISC in single-machine and distributed modes. For more information about the execution modes of SAS high-performance analytics procedures, see the section "Processing Modes" on page 10 in Chapter 3, "Shared Concepts and Topics." The focus of this example is to show how you can switch the modes of execution in PROC HPCANDISC. The following DATA step generates the data:

```
data ex2Data;
   drop i j n n1 n2 n3 n4;

   n  = 5000000;
   n1 = n*0.1;
   n2 = n*0.25;
   n3 = n*0.45;
   n4 = n*0.7;
```

```
   array x{100};

   do i=1 to n;
      do j=1 to dim(x);
         x{j} = ranuni(1);
      end;

      if      i <= n1 then z='small';
      else if i <= n2 then z='medium';
      else if i <= n3 then z='big';
      else if i <= n4 then z='verybig';
      else                 z='huge';

      output;
   end;
run;
```

The following statements use PROC HPCANDISC to perform a canonical discriminant analysis and to output various statistics to the stats data set (OUTSTAT= stats).

```
proc hpcandisc data=ex2Data outstat=stats;
   var x:;
   class z;
   performance details;
run;
```

Output 5.2.1 shows the "Performance Information" table. This table shows that the HPCANDISC procedure executes in single-machine mode on four threads, because the client machine has four CPUs. You can force a certain number of threads on any machine to be involved in the computations by specifying the NTHREADS= option in the PERFORMANCE statement.

Output 5.2.1 Performance Information in Single-Machine Mode

The HPCANDISC Procedure

Performance Information	
Execution Mode	Single-Machine
Number of Threads	4

Output 5.2.2 shows timing information for the PROC HPCANDISC run. This table is produced when you specify the DETAILS option in the PERFORMANCE statement. You can see that, in this case, the majority of time is spent reading, levelizing, and processing the data.

Output 5.2.2 Timing in Single-Machine Mode

Procedure Task Timing		
Task	**Seconds**	**Percent**
Reading, Levelizing, and Processing Data	128.61	99.85%
Computing SSCP and Covariance Matrices	0.00	0.00%
Performing Canonical Analysis	0.16	0.12%
Producing Output Statistics Data Set	0.03	0.02%

To switch to running PROC HPCANDISC in distributed mode, specify valid values for the NODES=, INSTALL=, and HOST= options in the PERFORMANCE statement. An alternative to specifying the INSTALL= and HOST= options in the PERFORMANCE statement is to use the OPTIONS SET commands to set appropriate values for the GRIDHOST and GRIDINSTALLLOC environment variables. For information about setting these options or environment variables, see the section "Processing Modes" on page 10 in Chapter 3, "Shared Concepts and Topics."

The following statements provide an example. To run these statements successfully, you need to set the macro variables GRIDHOST and GRIDINSTALLLOC to resolve to appropriate values, or you can replace the references to macro variables with appropriate values.

```
proc hpcandisc data=ex2Data outstat=stats;
   var x:;
   class z;
   performance details nodes = 4
               host="&GRIDHOST" install="&GRIDINSTALLLOC";
run;
```

The execution mode in the "Performance Information" table shown in Output 5.2.3 indicates that the calculations were performed in a distributed environment that uses four nodes, each of which uses 32 threads.

Output 5.2.3 Performance Information in Distributed Mode

Performance Information	
Host Node	<< your grid host >>
Install Location	<< your grid install location >>
Execution Mode	Distributed
Number of Compute Nodes	4
Number of Threads per Node	32

Another indication of distributed execution is the following message issued by all high-performance analytics procedures in the SAS log:

```
NOTE: The HPCANDISC procedure is executing in the distributed
      computing environment with 4 worker nodes.
```

Output 5.2.4 shows timing information for this distributed run of the HPCANDISC procedure. In contrast to the single-machine mode (where reading, levelizing, and processing the data dominated the time spent), the majority of time in the distributed mode run is spent distributing the data.

Output 5.2.4 Timing in Distributed Mode

Procedure Task Timing		
Task	**Seconds**	**Percent**
Obtaining Settings	0.00	0.00%
Distributing Data	63.97	95.27%
Reading, Levelizing, and Processing Data	2.73	4.07%
Computing SSCP and Covariance Matrices	0.00	0.00%
Performing Canonical Analysis	0.00	0.01%
Producing Output Statistics Data Set	0.17	0.25%
Waiting on Client	0.27	0.40%

References

Fisher, R. A. (1936), "The Use of Multiple Measurements in Taxonomic Problems," *Annals of Eugenics*, 7, 179–188.

Kshirsagar, A. M. (1972), *Multivariate Analysis*, New York: Marcel Dekker.

Lawley, D. N. (1959), "Tests of Significance in Canonical Analysis," *Biometrika*, 46, 59–66.

Puranen, J. (1917), "Fish Catch data set (1917)," Journal of Statistics Education Data Archive, accessed May 22, 2009.
URL http://www.amstat.org/publications/jse/datasets/fishcatch.txt

Rao, C. R. (1973), *Linear Statistical Inference and Its Applications*, 2nd Edition, New York: John Wiley & Sons.

Chapter 6
The HPFMM Procedure

Contents

Overview: HPFMM Procedure **106**
Basic Features 107
PROC HPFMM Contrasted with PROC FMM 108
Assumptions 109
Notation for the Finite Mixture Model 109
Homogeneous Mixtures 109
Special Mixtures 110
Getting Started: HPFMM Procedure **110**
Mixture Modeling for Binomial Overdispersion: "Student," Pearson, Beer, and Yeast . 110
Modeling Zero-Inflation: Is it Better to Fish Poorly or Not to Have Fished At All? . . 116
Looking for Multiple Modes: Are Galaxies Clustered? 124
Comparison with Roeder's Method 131
Syntax: HPFMM Procedure **134**
PROC HPFMM Statement 134
BAYES Statement 146
BY Statement 156
CLASS Statement 156
FREQ Statement 157
ID Statement 157
MODEL Statement 157
Response Variable Options 159
Model Options 160
OUTPUT Statement 165
PERFORMANCE Statement 168
PROBMODEL Statement 168
RESTRICT Statement 170
WEIGHT Statement 172
Details: HPFMM Procedure **172**
A Gentle Introduction to Finite Mixture Models 172
The Form of the Finite Mixture Model 172
Mixture Models Contrasted with Mixing and Mixed Models: Untangling the Terminology Web 173
Overdispersion 175
Log-Likelihood Functions for Response Distributions 175
Bayesian Analysis 182
Conjugate Sampling 182

Metropolis-Hastings Algorithm . . . 183
Latent Variables via Data Augmentation . . . 184
Prior Distributions . . . 184
Parameterization of Model Effects . . . 185
Computational Method . . . 186
Multithreading . . . 186
Choosing an Optimization Algorithm . . . 187
First- or Second-Order Algorithms . . . 187
Algorithm Descriptions . . . 188
Output Data Set . . . 190
Default Output . . . 191
Performance Information . . . 191
Model Information . . . 191
Class Level Information . . . 191
Number of Observations . . . 191
Response Profile . . . 192
Default Output for Maximum Likelihood . . . 192
Default Output for Bayes Estimation . . . 194
ODS Table Names . . . 195
ODS Graphics . . . 197
Examples: HPFMM Procedure . . . 197
Example 6.1: Modeling Mixing Probabilities: All Mice Are Created Equal, but Some Are More Equal . . . 197
Example 6.2: The Usefulness of Custom Starting Values: When Do Cows Eat? . . . 205
Example 6.3: Enforcing Homogeneity Constraints: Count and Dispersion—It Is All Over! . . . 214
References . . . 219

Overview: HPFMM Procedure

The HPFMM procedure is a high-performance counterpart of the FMM procedure that fits statistical models to data for which the distribution of the response is a finite mixture of univariate distributions—that is, each response comes from one of several random univariate distributions that have unknown probabilities. You can use PROC HPFMM to model the component distributions in addition to the mixing probabilities. For more precise definitions and a discussion of similar but distinct modeling methodologies, see the section "A Gentle Introduction to Finite Mixture Models" on page 172.

The HPFMM procedure is designed to fit finite mixtures of regression models or finite mixtures of generalized linear models in which the covariates and regression structure can be the same across components or can be different. You can fit finite mixture models by maximum likelihood or Bayesian methods. Note that classical statistical models are a special case of the finite mixture models in which the distribution of the data has only a single component.

PROC HPFMM runs in either single-machine mode or distributed mode.

NOTE: Distributed mode requires SAS High-Performance Statistics.

Basic Features

The HPFMM procedure estimates the parameters in univariate finite mixture models and produces various statistics to evaluate parameters and model fit. The following list summarizes some basic features of the HPFMM procedure:

- maximum likelihood estimation for all models
- Markov chain Monte Carlo estimation for many models, including zero-inflated Poisson models
- many built-in link and distribution functions for modeling, including the beta, shifted *t*, Weibull, beta-binomial, and generalized Poisson distributions, in addition to many standard members of the exponential family of distributions
- specialized built-in mixture models such as the binomial cluster model (Morel and Nagaraj 1993; Morel and Neerchal 1997; Neerchal and Morel 1998)
- acceptance of multiple MODEL statements to build mixture models in which the model effects, distributions, or link functions vary across mixture components
- model-building syntax using CLASS and effect-based MODEL statements familiar from many other SAS/STAT procedures (for example, the GLM, GLIMMIX, and MIXED procedures)
- evaluation of sequences of mixture models when you specify ranges for the number of components
- simple syntax to impose linear equality and inequality constraints among parameters
- ability to model regression and classification effects in the mixing probabilities through the PROBMODEL statement
- ability to incorporate full or partially known component membership into the analysis through the PARTIAL= option in the PROC HPFMM statement
- OUTPUT statement that produces a SAS data set with important statistics for interpreting mixture models, such as component log likelihoods and prior and posterior probabilities
- ability to add zero-inflation to any model
- output data set with posterior parameter values for the Markov chain
- multithreading and distributed computing for high-performance optimization and Monte Carlo sampling

The HPFMM procedure uses ODS Graphics to create graphs as part of its output. For general information about ODS Graphics, see Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*). For specific information about the statistical graphics available with the HPFMM procedure, see the PLOTS options in the PROC HPFMM statement.

Because the HPFMM procedure is a high-performance analytical procedure, it also does the following:

- enables you to run in distributed mode on a cluster of machines that distribute the data and the computations

- enables you to run in single-machine mode on the server where SAS is installed
- exploits all the available cores and concurrent threads, regardless of execution mode

For more information, see the section "Processing Modes" on page 10 in Chapter 3, "Shared Concepts and Topics."

PROC HPFMM Contrasted with PROC FMM

For general contrasts between SAS high-performance analytical procedures and other SAS procedures, see the section "Common Features of SAS High-Performance Statistical Procedures" on page 42 in Chapter 4, "Shared Statistical Concepts." The HPFMM procedure is somewhat distinct from other high-performance analytical procedures in being very nearly a twin of its counterpart, PROC FMM. You can fit the same kinds of models and get the same kinds of tabular, graphical, and data set results from PROC HPFMM as from PROC FMM. The main difference is that PROC HPFMM was developed primarily to work in a distributed environment, and PROC FMM primarily for a single (potentially multithreaded) host.

PROC HPFMM and PROC FMM have several differences because of their respective underlying technology:

- The ORDER option that specifies the sort order for the levels of CLASS variables is not available in the PROC statement of the HPFMM procedure. Instead the HPFMM procedure makes this option available in the CLASS statement.
- The CLASS statement in the HPFMM procedure provides many more options than the CLASS statement in the FMM procedure.
- The PERFORMANCE statement in the HPFMM procedure includes a superset of the options that are available in the PERFORMANCE statement in the FMM procedure.
- The NOVAR option in the OUTPUT statement in the FMM procedure is not available in the OUTPUT statement of the HPFMM procedure.

The OUTPUT statement in PROC HPFMM produces observationwise statistics. However, as is customary for SAS high-performance analytical procedures, PROC HPFMM's OUTPUT statement does not by default include the input and BY variables in the output data set. This is to avoid data duplication for large data sets. In order to include any input or BY variables in the output data set, you must list these variables in the ID statement. Furthermore, PROC HPFMM's OUTPUT statement includes the predicted values of the response variable if you do not specify any output statistics.

In contrast, when you request that the posterior sample be saved to a SAS data by specifying the OUTPOST= option in the BAYES statement, PROC HPFMM includes the BY variables in the data set.

Assumptions

The HPFMM procedure makes the following assumptions in fitting statistical models:

- The number of components k in the finite mixture is known a priori and is not a parameter to be estimated.
- The parameters of the components are distinct a priori.
- The observations are uncorrelated.

Notation for the Finite Mixture Model

The general expression for the finite mixture model fitted with the HPFMM procedure is as follows:

$$f(y) = \sum_{j=1}^{k} \pi_j(\mathbf{z}, \boldsymbol{\alpha}_j)\, p_j(y; \mathbf{x}_j'\boldsymbol{\beta}_j, \phi_j)$$

The number of components in the mixture is denoted as k. The mixture probabilities π_j can depend on regressor variables $\mathbf{z}$ and parameters $\boldsymbol{\alpha}_j$. By default, the HPFMM procedure models these probabilities using a logit transform if $k = 2$ and as a generalized logit model if $k > 2$. The component distributions p_j can also depend on regressor variables in $\mathbf{x}_j$, regression parameters $\boldsymbol{\beta}_j$, and possibly scale parameters ϕ_j. Notice that the component distributions p_j are indexed by j since the distributions might belong to different families. For example, in a two-component model, you might model one component as a normal (Gaussian) variable and the second component as a variable with a t distribution with low degrees of freedom to manage overdispersion.

The mixture probabilities π_j satisfy $\pi_j \geq 0$, for all j, and

$$\sum_{j=1}^{k} \pi_j(\mathbf{z}, \boldsymbol{\alpha}_j) = 1$$

Homogeneous Mixtures

If the component distributions are of the same distributional form, the mixture is called homogeneous. In most applications of homogeneous mixtures, the mixing probabilities do not depend on regression parameters. The general model then simplifies to

$$f(y) = \sum_{j=1}^{k} \pi_j\, p(y; \mathbf{x}'\boldsymbol{\beta}_j, \phi_j)$$

Since the component distributions depend on regression parameters $\boldsymbol{\beta}_j$, this model is known as a homogeneous regression mixture. A homogeneous regression mixture assumes that the regression effects are the same across the components, although the HPFMM procedure does not impose such a restriction. If the component distributions do not contain regression effects, the model

$$f(y) = \sum_{j=1}^{k} \pi_j \, p(y; \mu_j, \phi_j)$$

is *the* homogeneous mixture model. A classical case is the estimation of a continuous density as a k-component mixture of normal distributions.

Special Mixtures

The HPFMM procedure enables you to fit several special mixture models. The Morel-Neerchal binomial cluster model (Morel and Nagaraj 1993; Morel and Neerchal 1997; Neerchal and Morel 1998) is a mixture of binomial distributions in which the success probabilities depend on the mixing probabilities.

Zero-inflated count models are obtained as two-component mixtures where one component is a classical count model—such as the Poisson or negative binomial model—and the other component is a distribution that is concentrated at zero. If the nondegenerate part of this special mixture is a zero-truncated model, the resulting two-component mixture is known as a hurdle model (Cameron and Trivedi 1998).

Getting Started: HPFMM Procedure

Mixture Modeling for Binomial Overdispersion: "Student," Pearson, Beer, and Yeast

The following example demonstrates how you can model a complicated, two-component binomial mixture distribution, either with maximum likelihood or with Bayesian methods, with a few simple PROC HPFMM statements.

William Sealy Gosset, a chemist at the Arthur Guinness Son and Company brewery in Dublin, joined the statistical laboratory of Karl Pearson in 1906–1907 to study statistics. At first Gosset—who published all but one paper under the pseudonym "Student" because his employer forbade publications by employees after a co-worker had disclosed trade secrets—worked on the Poisson limit to the binomial distribution, using haemacytometer yeast cell counts. Gosset's interest in studying small-sample (and limit) problems was motivated by the small sample sizes he typically saw in his work at the brewery.

Subsequently, Gosset's yeast count data have been examined and revisited by many authors. In 1915, Karl Pearson undertook his own examination and realized that the variability in "Student's" data exceeded that consistent with a Poisson distribution. Pearson (1915) bemoans the fact that if this were so, "it is certainly most unfortunate that such material should have been selected to illustrate Poisson's limit to the binomial."

Using a count of Gosset's yeast cell counts on the 400 squares of a haemacytometer (Table 6.1), Pearson argues that a mixture process would explain the heterogeneity (beyond the Poisson).

Table 6.1 "Student's" Yeast Cell Counts

Number of Cells	0	1	2	3	4	5
Frequency	213	128	37	18	3	1

Pearson fits various models to these data, chief among them a mixture of two binomial series

$$\nu_1(p_1 + q_1)^\theta + \nu_2(p_2 + q_2)^\theta$$

where θ is real-valued and thus the binomial series expands to

$$(p+q)^\theta = \sum_{k=0}^{\infty} \frac{\Gamma(\theta+1)}{\Gamma(k+1)\Gamma(\theta-k+1)} p^k q^{\theta-k}$$

Pearson's fitted model has $\theta = 4.89997$, $\nu_1 = 356.986$, $\nu_2 = 43.014$ (corresponding to a mixing proportion of $356.986/(43.014 + 356.986) = 0.892$), and estimated success probabilities in the binomial components of 0.1017 and 0.4514, respectively. The success probabilities indicate that although the data have about a 90% chance of coming from a distribution with small success probability of about 0.1, there is a 10% chance of coming from a distribution with a much larger success probability of about 0.45.

If θ is an integer, the binomial series is the cumulative mass function of a binomial random variable. The value of θ suggests that a suitable model for these data could also be constructed as a two-component mixture of binomial random variables as follows:

$$f(y) = \pi \,\text{binomial}(5, \mu_1) + (1-\pi)\,\text{binomial}(5, \mu_2)$$

The binomial sample size n=5 is suggested by Pearson's estimate of $\theta = 4.89997$ and the fact that the largest cell count in Table 6.1 is 5.

The following DATA step creates a SAS data set from the data in Table 6.1.

```
data yeast;
   input count f;
   n = 5;
   datalines;
   0     213
   1     128
   2      37
   3      18
   4       3
   5       1
;
```

The two-component binomial model is fit with the HPFMM procedure with the following statements:

```
proc hpfmm data=yeast;
   model count/n =  / k=2;
   freq f;
run;
```

Because the events/trials syntax is used in the MODEL statement, PROC HPFMM defaults to the binomial distribution. The K=2 option specifies that the number of components is fixed and known to be two. The FREQ statement indicates that the data are grouped; for example, the first observation represents 213 squares on the haemacytometer where no yeast cells were found.

The "Model Information" and "Number of Observations" tables in Figure 6.1 convey that the fitted model is a two-component homogeneous binomial mixture with a logit link function. The mixture is *homogeneous* because there are no model effects in the MODEL statement and because both component distributions belong to the same distributional family. By default, PROC HPFMM estimates the model parameters by maximum likelihood.

Although only six observations are read from the data set, the data represent 400 observations (squares on the haemacytometer). Since a constant binomial sample size of 5 is assumed, the data represent 273 successes (finding a yeast cell) out of 2,000 Bernoulli trials.

Figure 6.1 Model Information for Yeast Cell Model

The HPFMM Procedure

Model Information	
Data Set	YEAST
Response Variable (Events)	count
Response Variable (Trials)	n
Frequency Variable	f
Type of Model	Homogeneous Mixture
Distribution	Binomial
Components	2
Link Function	Logit
Estimation Method	Maximum Likelihood

Number of Observations Read	6
Number of Observations Used	6
Sum of Frequencies Read	400
Sum of Frequencies Used	400
Number of Events	273
Number of Trials	2000

The estimated intercepts (on the logit scale) for the two binomial means are –2.2316 and –0.2974, respectively. These values correspond to binomial success probabilities of 0.09695 and 0.4262, respectively (Figure 6.2). The two components mix with probabilities 0.8799 and $1 - 0.8799 = 0.1201$. These values are generally close to the values found by Pearson (1915) using infinite binomial series instead of binomial mass functions.

Figure 6.2 Maximum Likelihood Estimates

Parameter Estimates for Binomial Model

Component	Parameter	Estimate	Standard Error	z Value	Pr > |z|	Inverse Linked Estimate
1	Intercept	-2.2316	0.1522	-14.66	<.0001	0.09695
2	Intercept	-0.2974	0.3655	-0.81	0.4158	0.4262

Figure 6.2 *continued*

Parameter Estimates for Mixing Probabilities

		Linked Scale			
Component	Mixing Probability	Logit(Prob)	Standard Error	z Value	Pr > |z|
1	0.8799	1.9913	0.5725	3.48	0.0005
2	0.1201	-1.9913			

To obtain fitted values and other observationwise statistics under the stipulated two-component model, you can add the OUTPUT statement to the previous PROC HPFMM run. The following statements request componentwise predicted values and the posterior probabilities:

```
proc hpfmm data=yeast;
   model count/n =  / k=2;
   freq f;
   id f n;
   output out=hpfmmout pred(components) posterior;
run;
data hpfmmout;
   set hpfmmout;
   PredCount_1 = post_1 * f;
   PredCount_2 = post_2 * f;
run;
proc print data=hpfmmout;
run;
```

The DATA step following the PROC HPFMM step computes the predicted cell counts in each component (Figure 6.3). Note that the The predicted means in the components, 0.48476 and 2.13099, are close to the values determined by Pearson (0.4983 and 2.2118), as are the predicted cell counts.

Figure 6.3 Predicted Cell Counts

Obs	f	n	Pred_1	Pred_2	Post_1	Post_2	PredCount_1	PredCount_2
1	213	5	0.096951	0.42620	0.98606	0.01394	210.030	2.9698
2	128	5	0.096951	0.42620	0.91089	0.08911	116.594	11.4058
3	37	5	0.096951	0.42620	0.59638	0.40362	22.066	14.9341
4	18	5	0.096951	0.42620	0.17598	0.82402	3.168	14.8323
5	3	5	0.096951	0.42620	0.02994	0.97006	0.090	2.9102
6	1	5	0.096951	0.42620	0.00444	0.99556	0.004	0.9956

Gosset, who was interested in small-sample statistical problems, investigated the use of prior knowledge in mathematical-statistical analysis—for example, deriving the sampling distribution of the correlation coefficient after having assumed a uniform prior distribution for the coefficient in the population (Aldrich 1997). Pearson also was not opposed to using prior information, especially uniform priors that reflect "equal distribution of ignorance." Fisher, on the other hand, would not have any of it: the best estimator in his opinion is obtained by a criterion that is absolutely independent of prior assumptions about probabilities of particular values. He objected to the insinuation that his derivations in the work on the correlation were deduced from Bayes theorem (Fisher 1921).

The preceding analysis of the yeast cell count data uses maximum likelihood methods that are free of prior assumptions. The following analysis takes instead a Bayesian approach, assuming a beta prior distribution for the binomial success probabilities and a uniform prior distribution for the mixing probabilities. The changes from the previous run of PROC HPFMM are the addition of the ODS GRAPHICS, PERFORMANCE, and BAYES statements and the SEED=12345 option.

```
ods graphics on;
proc hpfmm data=yeast seed=12345;
   model count/n = / k=2;
   freq f;
   performance nthreads=2;
   bayes;
run;
ods graphics off;
```

When ODS Graphics is enabled, PROC HPFMM produces diagnostic trace plots for the posterior samples. Bayesian analyses are sensitive to the random number seed and thread count; the SEED= and NTHREADS= options in the PERFORMANCE statement ensure consistent results for the purposes of this example. The SEED=12345 option in the PROC HPFMM statement determines the random number seed for the random number generator that the analysis used. The NTHREADS=2 option in the PERFORMANCE statement sets the number of threads to be used by the procedure to two. The BAYES statement requests a Bayesian analysis.

The "Bayes Information" table in Figure 6.4 provides basic information about the Markov chain Monte Carlo sampler. Because the model is a homogeneous mixture, the HPFMM procedure applies an efficient conjugate sampling algorithm with a posterior sample size of 10,000 samples after a burn-in size of 2,000 samples. The "Prior Distributions" table displays the prior distribution for each parameter along with its mean and variance and the initial value in the chain. Notice that in this situation all three prior distributions reduce to a uniform distribution on (0, 1).

Figure 6.4 Basic Information about MCMC Sampler

The HPFMM Procedure

Bayes Information	
Sampling Algorithm	Conjugate
Data Augmentation	Latent Variable
Initial Values of Chain	Data Based
Burn-In Size	2000
MC Sample Size	10000
MC Thinning	1
Parameters in Sampling	3
Mean Function Parameters	2
Scale Parameters	0
Mixing Prob Parameters	1

Prior Distributions

Component	Parameter	Distribution	Mean	Variance	Initial Value
1	**Success Probability**	Beta(1, 1)	0.5000	0.08333	0.1365
2	**Success Probability**	Beta(1, 1)	0.5000	0.08333	0.1365
1	**Probability**	Dirichlet(1, 1)	0.5000	0.08333	0.6180

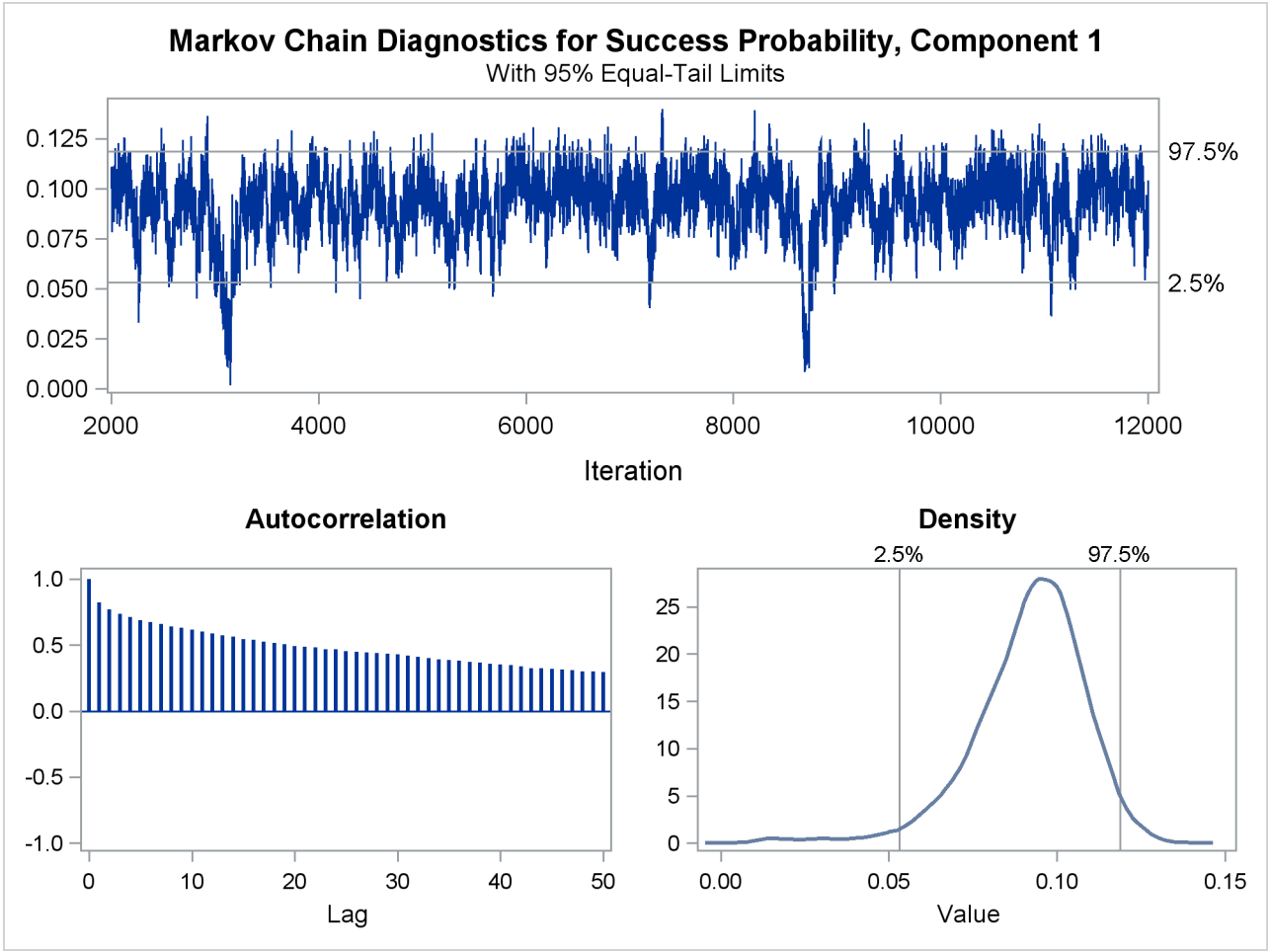
The HPFMM procedure produces a log note for this model, indicating that the sampled quantities are not the linear predictors on the logit scale, but are the actual population parameters (on the data scale):

```
NOTE: Bayesian results for this model (no regressor variables,
      non-identity link) are displayed on the data scale, not the
      linked scale. You can obtain results on the linked (=linear)
      scale by requesting a Metropolis-Hastings sampling algorithm.
```

The trace panel for the success probability in the first binomial component is shown in Figure 6.5. Note that the first component in this Bayesian analysis corresponds to the second component in the MLE analysis. The graphics in this panel can be used to diagnose the convergence of the Markov chain. If the chain has not converged, inferences cannot be made based on quantities derived from the chain. You generally look for the following:

- a smooth unimodal distribution of the posterior estimates in the density plot displayed on the lower right
- good mixing of the posterior samples in the trace plot at the top of the panel (good mixing is indicated when the trace traverses the support of the distribution and appears to have reached a stationary distribution)

Figure 6.5 Trace Panel for Success Probability in First Component



The autocorrelation plot in Figure 6.5 shows fairly high and sustained autocorrelation among the posterior estimates. While this is generally not a problem, you can affect the degree of autocorrelation among the posterior estimates by running a longer chain and thinning the posterior estimates; see the NMC= and THIN= options in the BAYES statement.

Both the trace plot and the density plot in Figure 6.5 are indications of successful convergence.

Figure 6.6 reports selected results that summarize the 10,000 posterior samples. The arithmetic means of the success probabilities in the two components are 0.0917 and 0.3974, respectively. The posterior mean of the mixing probability is 0.8312. These values are similar to the maximum likelihood parameter estimates in Figure 6.2 (after swapping components).

Figure 6.6 Summaries for Posterior Estimates

Posterior Summaries

Component	Parameter	N	Mean	Standard Deviation	Percentiles 25	50	75
1	Success Probability	10000	0.0917	0.0168	0.0830	0.0938	0.1027
2	Success Probability	10000	0.3974	0.0871	0.3379	0.3957	0.4556
1	Probability	10000	0.8312	0.1045	0.7986	0.8578	0.8984

Posterior Intervals

Component	Parameter	Alpha	Equal-Tail Interval		HPD Interval	
1	Success Probability	0.050	0.0530	0.1187	0.0585	0.1212
2	Success Probability	0.050	0.2272	0.5722	0.2343	0.5780
1	Probability	0.050	0.5464	0.9454	0.6325	0.9642

Note that the standard errors in Figure 6.2 are not comparable to those in Figure 6.6, since the standard errors for the MLEs are expressed on the logit scale and the Bayes estimates are expressed on the data scale. You can add the METROPOLIS option in the BAYES statement to sample the quantities on the logit scale.

The "Posterior Intervals" table in Figure 6.6 displays 95% credible intervals (equal-tail intervals and intervals of highest posterior density). It can be concluded that the component with the higher success probability contributes less than 40% to the process.

Modeling Zero-Inflation: Is it Better to Fish Poorly or Not to Have Fished At All?

The following example shows how you can use PROC HPFMM to model data with more zero values than expected.

Many count data show an excess of zeros relative to the frequency of zeros expected under a reference model. An excess of zeros leads to overdispersion since the process is more variable than a standard count data model. Different mechanisms can lead to excess zeros. For example, suppose that the data are generated from two processes with different distribution functions—one process generates the zero counts, and the other process generates nonzero counts. In the vernacular of Cameron and Trivedi (1998), such a model is called a *hurdle* model. With a certain probability—the probability of a nonzero count—a hurdle is crossed, and events are being generated. Hurdle models are useful, for example, to model the number of doctor visits

per year. Once the decision to see a doctor has been made—the hurdle has been overcome—a certain number of visits follow.

Hurdle models are closely related to zero-inflated models. Both can be expressed as two-component mixtures in which one component has a degenerate distribution at zero and the other component is a count model. In a hurdle model, the count model follows a zero-truncated distribution. In a zero-inflated model, the count model has a nonzero probability of generating zeros. Formally, a zero-inflated model can be written as

$$\Pr(Y = y) = \pi p_1 + (1 - \pi) p_2(y, \mu)$$
$$p_1 = \begin{cases} 1 & y = 0 \\ 0 & \text{otherwise} \end{cases}$$

where $p_2(y, \mu)$ is a standard count model with mean μ and support $y \in \{0, 1, 2, \cdots\}$.

The following data illustrates the use of a zero-inflated model. In a survey of park attendees, randomly selected individuals were asked about the number of fish they caught in the last six months. Along with that count, the gender and age of each sampled individual was recorded. The following DATA step displays the data for the analysis:

```
data catch;
   input gender $ age count @@;
   datalines;
  F   54  18   M   37   0   F   48  12   M   27   0
  M   55   0   M   32   0   F   49  12   F   45  11
  M   39   0   F   34   1   F   50   0   M   52   4
  M   33   0   M   32   0   F   23   1   F   17   0
  F   44   5   M   44   0   F   26   0   F   30   0
  F   38   0   F   38   0   F   52  18   M   23   1
  F   23   0   M   32   0   F   33   3   M   26   0
  F   46   8   M   45   5   M   51  10   F   48   5
  F   31   2   F   25   1   M   22   0   M   41   0
  M   19   0   M   23   0   M   31   1   M   17   0
  F   21   0   F   44   7   M   28   0   M   47   3
  M   23   0   F   29   3   F   24   0   M   34   1
  F   19   0   F   35   2   M   39   0   M   43   6
;
```

At first glance, the prevalence of zeros in the DATA set is apparent. Many park attendees did not catch any fish. These zero counts are made up of two populations: attendees who do not fish and attendees who fish poorly. A zero-inflation mechanism thus appears reasonable for this application since a zero count can be produced by two separate distributions.

The following statements fit a standard Poisson regression model to these data. A common intercept is assumed for men and women, and the regression slope varies with gender.

```
proc hpfmm data=catch;
   class gender;
   model count = gender*age / dist=Poisson;
run;
```

Figure 6.7 displays information about the model and data set. The "Model Information" table conveys that the model is a single-component Poisson model (a Poisson GLM) and that parameters are estimated by maximum likelihood. There are two levels in the CLASS variable gender, with females preceding males.

Figure 6.7 Model Information and Class Levels in Poisson Regression

The HPFMM Procedure

Model Information	
Data Set	CATCH
Response Variable	count
Type of Model	Generalized Linear (GLM)
Distribution	Poisson
Components	1
Link Function	Log
Estimation Method	Maximum Likelihood

Class Level Information		
Class	**Levels**	**Values**
gender	2	F M

Number of Observations Read	52
Number of Observations Used	52

The "Fit Statistics" and "Parameter Estimates" tables from the maximum likelihood estimation of the Poisson GLM are shown in Figure 6.8. If the model is not overdispersed, the Pearson statistic should roughly equal the number of observations in the data set minus the number of parameters. With n=52, there is evidence of overdispersion in these data.

Figure 6.8 Fit Results in Poisson Regression

Fit Statistics	
-2 Log Likelihood	182.7
AIC (Smaller is Better)	188.7
AICC (Smaller is Better)	189.2
BIC (Smaller is Better)	194.6
Pearson Statistic	85.9573

Parameter Estimates for Poisson Model					
Effect	**gender**	**Estimate**	**Standard Error**	**z Value**	**Pr > |z|**
Intercept		-3.9811	0.5439	-7.32	<.0001
age*gender	F	0.1278	0.01149	11.12	<.0001
age*gender	M	0.1044	0.01224	8.53	<.0001

Suppose that the cause of overdispersion is zero-inflation of the count data. The following statements fit a zero-inflated Poisson model.

```
proc hpfmm data=catch;
   class gender;
   model count = gender*age / dist=Poisson ;
   model       +            / dist=Constant;
run;
```

There are two MODEL statements, one for each component of the mixture. Because the distributions are different for the components, you cannot specify the mixture model with a single MODEL statement. The first MODEL statement identifies the response variable for the model (count) and defines a Poisson model with intercept and gender-specific slopes. The second MODEL statement uses the continuation operator ("+") and adds a model with a degenerate distribution by using DIST=CONSTANT. Because the mass of the constant is placed by default at zero, the second MODEL statement adds a zero-inflation component to the model. It is sufficient to specify the response variable in one of the MODEL statements; you use the "=" sign in that statement to separate the response variable from the model effects.

Figure 6.9 displays the "Model Information" and "Optimization Information" tables for this run of the HPFMM procedure. The model is now identified as a zero-inflated Poisson (ZIP) model with two components, and the parameters continue to be estimated by maximum likelihood. The "Optimization Information" table shows that there are four parameters in the optimization (compared to three parameters in the Poisson GLM model). The four parameters correspond to three parameters in the mean function (intercept and two gender-specific slopes) and the mixing probability.

Figure 6.9 Model and Optimization Information in the ZIP Model

The HPFMM Procedure

Model Information	
Data Set	CATCH
Response Variable	count
Type of Model	Zero-inflated Poisson
Components	2
Estimation Method	Maximum Likelihood

Optimization Information	
Optimization Technique	Dual Quasi-Newton
Parameters in Optimization	4
Mean Function Parameters	3
Scale Parameters	0
Mixing Prob Parameters	1

Results from fitting the ZIP model by maximum likelihood are shown in Figure 6.10. The –2 log likelihood and the information criteria suggest a much-improved fit over the single-component Poisson model (compare Figure 6.10 to Figure 6.8). The Pearson statistic is reduced by factor 2 compared to the Poisson model and suggests a better fit than the standard Poisson model.

Figure 6.10 Maximum Likelihood Results for the ZIP model

Fit Statistics	
-2 Log Likelihood	145.6
AIC (Smaller is Better)	153.6
AICC (Smaller is Better)	154.5
BIC (Smaller is Better)	161.4
Pearson Statistic	43.4467
Effective Parameters	4
Effective Components	2

Figure 6.10 *continued*

Parameter Estimates for Poisson Model

Component	Effect	gender	Estimate	Standard Error	z Value	Pr > |z|
1	Intercept		-3.5215	0.6448	-5.46	<.0001
1	age*gender	F	0.1216	0.01344	9.04	<.0001
1	age*gender	M	0.1056	0.01394	7.58	<.0001

Parameter Estimates for Mixing Probabilities

		Linked Scale			
Component	Mixing Probability	Logit(Prob)	Standard Error	z Value	Pr > |z|
1	0.6972	0.8342	0.4768	1.75	0.0802
2	0.3028	-0.8342			

The number of effective parameters and components shown in Figure 6.8 equals the values from Figure 6.9. This is not always the case because components can collapse (for example, when the mixing probability approaches zero or when two components have identical parameter estimates). In this example, both components and all four parameters are identifiable. The Poisson regression and the zero process mix, with a probability of approximately 0.6972 attributed to the Poisson component.

The HPFMM procedure enables you to fit some mixture models by Bayesian techniques. The following statements add the BAYES statement to the previous PROC HPFMM statements:

```
proc hpfmm data=catch seed=12345;
   class gender;
   model count = gender*age / dist=Poisson;
   model       +            / dist=constant;
   performance nthreads=2;
   bayes;
run;
```

The "Model Information" table indicates that the model parameters are estimated by Markov chain Monte Carlo techniques, and it displays the random number seed (Figure 6.11). This is useful if you did not specify a seed to identify the seed value that reproduces the current analysis. The "Bayes Information" table provides basic information about the Monte Carlo sampling scheme. The sampling method uses a data augmentation scheme to impute component membership and then the Gamerman (1997) algorithm to sample the component-specific parameters. The 2,000 burn-in samples are followed by 10,000 Monte Carlo samples without thinning.

Figure 6.11 Model, Bayes, and Prior Information in the ZIP Model

The HPFMM Procedure

Model Information	
Data Set	CATCH
Response Variable	count
Type of Model	Zero-inflated Poisson
Components	2
Estimation Method	Markov Chain Monte Carlo
Random Number Seed	12345

Bayes Information	
Sampling Algorithm	Gamerman
Data Augmentation	Latent Variable
Initial Values of Chain	ML Estimates
Burn-In Size	2000
MC Sample Size	10000
MC Thinning	1
Parameters in Sampling	4
Mean Function Parameters	3
Scale Parameters	0
Mixing Prob Parameters	1

Prior Distributions

Component	Effect	gender	Distribution	Mean	Variance	Initial Value
1	**Intercept**		Normal(0, 1000)	0	1000.00	-3.5215
1	**age*gender**	F	Normal(0, 1000)	0	1000.00	0.1216
1	**age*gender**	M	Normal(0, 1000)	0	1000.00	0.1056
1	**Probability**		Dirichlet(1, 1)	0.5000	0.08333	0.6972

The "Prior Distributions" table identifies the prior distributions, their parameters for the sampled quantities, and their initial values. The prior distribution of parameters associated with model effects is a normal distribution with mean 0 and variance 1,000. The prior distribution for the mixing probability is a Dirichlet(1,1), which is identical to a uniform distribution (Figure 6.11). Since the second mixture component is a degeneracy at zero with no associated parameters, it does not appear in the "Prior Distributions" table in Figure 6.11.

Figure 6.12 displays descriptive statistics about the 10,000 posterior samples. Recall from Figure 6.10 that the maximum likelihood estimates were –3.5215, 0.1216, 0.1056, and 0.6972, respectively. With this choice of prior, the means of the posterior samples are generally close to the MLEs in this example. The "Posterior Intervals" table displays 95% intervals of equal-tail probability and 95% intervals of highest posterior density (HPD) intervals.

Figure 6.12 Posterior Summaries and Intervals in the ZIP Model

Posterior Summaries

Component	Effect	gender	N	Mean	Standard Deviation	Percentiles 25	50	75
1	Intercept		10000	-3.5456	0.6463	-3.9822	-3.5286	-3.1082
1	age*gender	F	10000	0.1219	0.0135	0.1129	0.1216	0.1310
1	age*gender	M	10000	0.1057	0.0140	0.0962	0.1053	0.1148
1	Probability		10000	0.6923	0.0950	0.6280	0.6960	0.7589

Posterior Intervals

Component	Effect	gender	Alpha	Equal-Tail Interval		HPD Interval	
1	Intercept		0.050	-4.8615	-2.3246	-4.8979	-2.3808
1	age*gender	F	0.050	0.0955	0.1490	0.0963	0.1495
1	age*gender	M	0.050	0.0784	0.1338	0.0786	0.1339
1	Probability		0.050	0.4999	0.8683	0.5139	0.8799

You can generate trace plots for the posterior parameter estimates by enabling ODS Graphics:

```
ods graphics on;
ods select TADPanel;
proc hpfmm data=catch seed=12345;
   class gender;
   model count = gender*age / dist=Poisson;
   model       +            / dist=constant;
   performance nthreads=2;
   bayes;
run;
ods graphics off;
```

A separate trace panel is produced for each sampled parameter, and the panels for the gender-specific slopes are shown in Figure 6.13. There is good mixing in the chains: the modest autocorrelation that diminishes after about 10 successive samples. By default, the HPFMM procedure transfers the credible intervals for each parameter from the "Posterior Intervals" table to the trace plot and the density plot in the trace panel.

Figure 6.13 Trace Panels for Gender-Specific Slopes

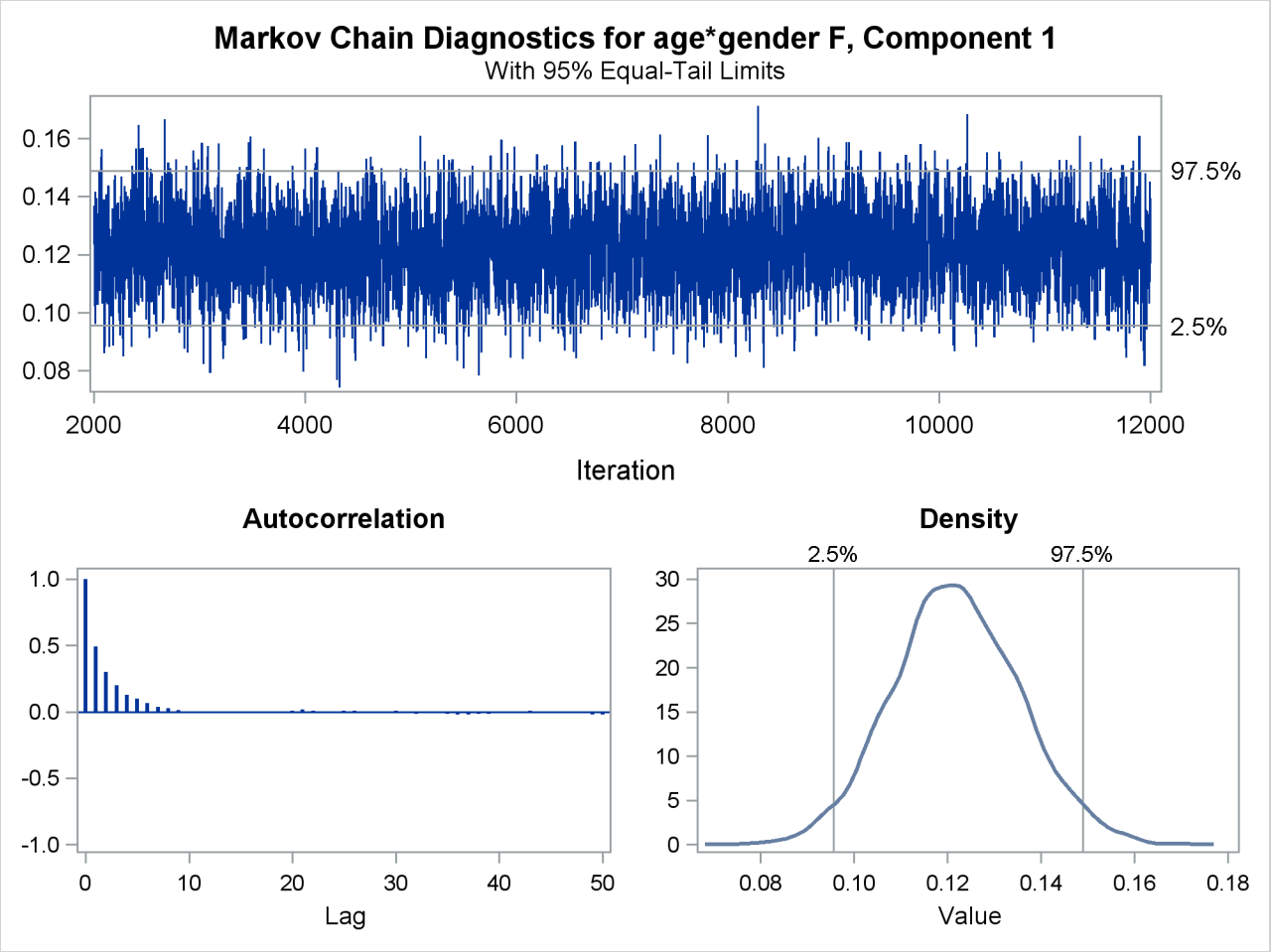
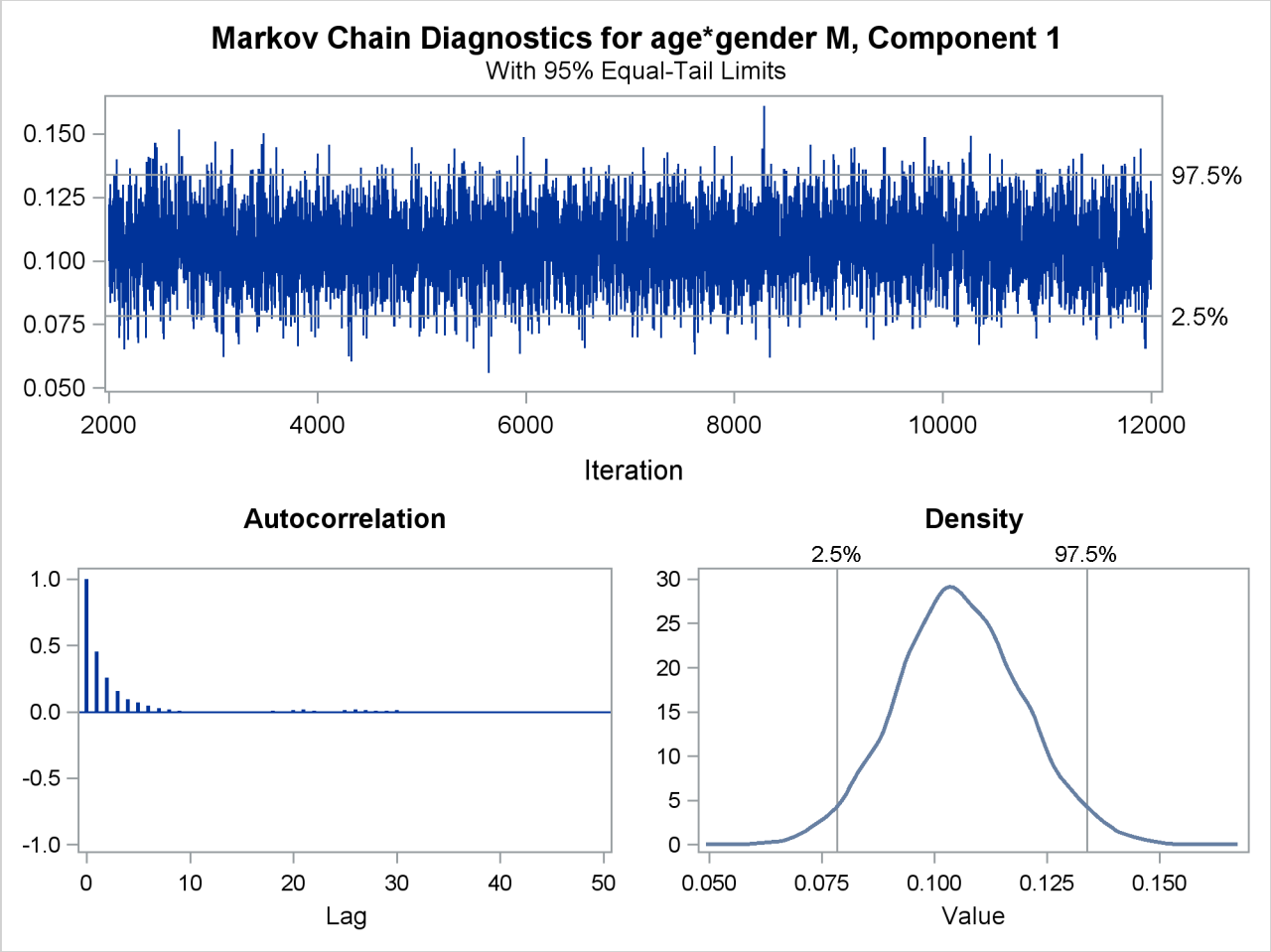


Figure 6.13 *continued*



Looking for Multiple Modes: Are Galaxies Clustered?

Mixture modeling is essentially a generalized form of one-dimensional cluster analysis. The following example shows how you can use PROC HPFMM to explore the number and nature of Gaussian clusters in univariate data.

Roeder (1990) presents data from the Corona Borealis sky survey with the velocities of 82 galaxies in a narrow slice of the sky. Cosmological theory suggests that the observed velocity of each galaxy is proportional to its distance from the observer. Thus, the presence of multiple modes in the density of these velocities could indicate a clustering of the galaxies at different distances.

The following DATA step recreates the data set in Roeder (1990). The computed variable v represents the measured velocity in thousands of kilometers per second.

```
title "HPFMM Analysis of Galaxies Data";
data galaxies;
   input velocity @@;
   v = velocity / 1000;
   datalines;
9172  9350  9483  9558  9775  10227 10406 16084 16170 18419
18552 18600 18927 19052 19070 19330 19343 19349 19440 19473
19529 19541 19547 19663 19846 19856 19863 19914 19918 19973
19989 20166 20175 20179 20196 20215 20221 20415 20629 20795
20821 20846 20875 20986 21137 21492 21701 21814 21921 21960
22185 22209 22242 22249 22314 22374 22495 22746 22747 22888
22914 23206 23241 23263 23484 23538 23542 23666 23706 23711
24129 24285 24289 24366 24717 24990 25633 26960 26995 32065
32789 34279
;
```

Analysis of potentially multimodal data is a natural application of finite mixture models. In this case, the modeling is complicated by the question of the variance for each of the components. Using identical variances for each component could obscure underlying structure, but the additional flexibility granted by component-specific variances might introduce spurious features.

You can use PROC HPFMM to prepare analyses for equal and unequal variances and use one of the available fit statistics to compare the resulting models. You can use the model selection facility to explore models with varying numbers of mixture components—say, from three to seven as investigated in Roeder (1990). The following statements select the best unequal-variance model using Akaike's information criterion (AIC), which has a built-in penalty for model complexity:

```
title2 "Three to Seven Components, Unequal Variances";
ods graphics on;
proc hpfmm data=galaxies criterion=AIC;
   model v = / kmin=3 kmax=7;
   ods exclude IterHistory OptInfo ComponentInfo;
run;
```

The KMIN= and KMAX= options indicate the smallest and largest number of components to consider. The ODS GRAPHICS and ODS SELECT statements request a density plot. The output for unequal variances is shown in Figure 6.14 and Figure 6.15.

Figure 6.14 Model Selection for Galaxy Data Assuming Unequal Variances

HPFMM Analysis of Galaxies Data
Three to Seven Components, Unequal Variances

The HPFMM Procedure

Model Information	
Data Set	GALAXIES
Response Variable	v
Type of Model	Homogeneous Mixture
Distribution	Normal
Min Components	3
Max Components	7
Link Function	Identity
Estimation Method	Maximum Likelihood

Component Evaluation for Mixture Models

Model ID	Number of Components Total	Number of Components Eff.	Number of Parameters Total	Number of Parameters Eff.	-2 Log L	AIC	AICC	BIC	Pearson	Max Gradient
1	**3**	**3**	**8**	**8**	406.96	422.96	424.94	442.22	82.00	0.000027
2	**4**	**4**	**11**	**11**	406.96	428.96	432.74	455.44	82.00	0.00012
3	**5**	**5**	**14**	**14**	406.96	434.96	441.23	468.66	82.00	0.000040
4	**6**	**6**	**17**	**17**	406.96	440.96	450.53	481.88	82.00	0.000029
5	**7**	**7**	**20**	**20**	406.96	446.96	460.73	495.10	82.00	0.000076

The model with 3 components (ID=1) was selected as 'best' based on the AIC statistic.

Fit Statistics	
-2 Log Likelihood	407.0
AIC (Smaller is Better)	423.0
AICC (Smaller is Better)	424.9
BIC (Smaller is Better)	442.2
Pearson Statistic	82.0001
Effective Parameters	8
Effective Components	3

Parameter Estimates for Normal Model

Component	Parameter	Estimate	Standard Error	z Value	Pr > |z|
1	**Intercept**	9.7101	0.1597	60.80	<.0001
2	**Intercept**	33.0444	0.5322	62.09	<.0001
3	**Intercept**	21.4039	0.2597	82.41	<.0001
1	**Variance**	0.1785	0.09542		
2	**Variance**	0.8496	0.6937		
3	**Variance**	4.8567	0.8098		

Figure 6.14 *continued*

Parameter Estimates for Mixing Probabilities					
		Linked Scale			
Component	Mixing Probability	GLogit(Prob)	Standard Error	z Value	Pr > |z|
1	0.0854	-2.3308	0.3959	-5.89	<.0001
2	0.0366	-3.1781	0.5893	-5.39	<.0001
3	0.8781	0			

Figure 6.15 Density Plot for Best (Three-Component) Model Assuming Unequal Variances

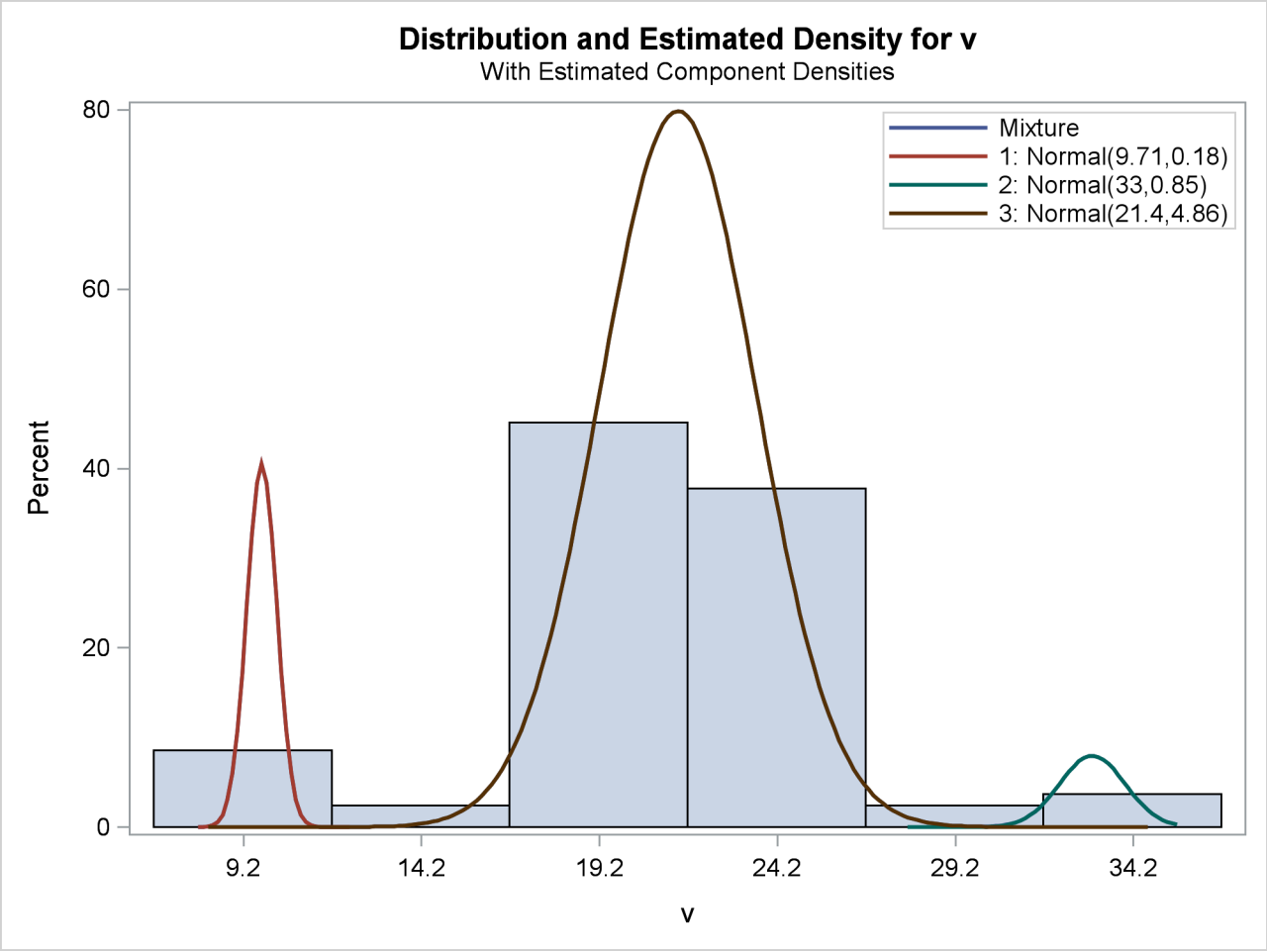
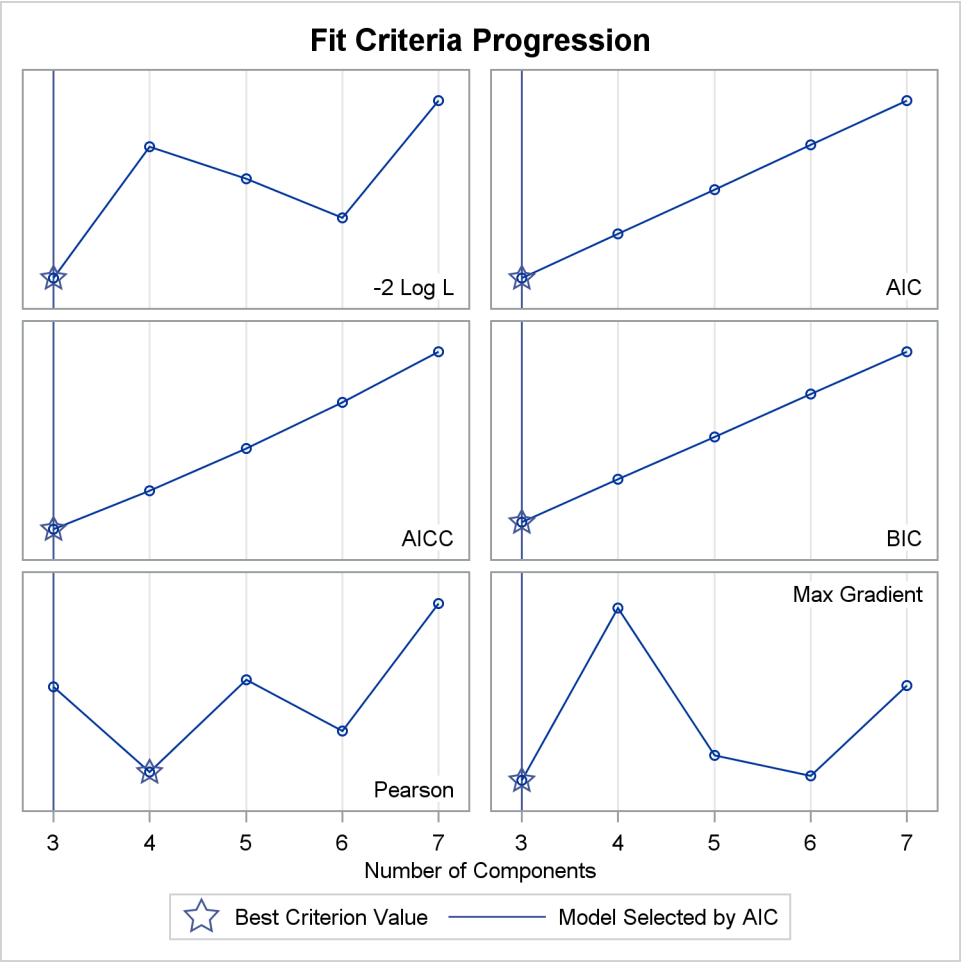


Figure 6.16 Criterion Panel Plot for Model Selection Assuming Unequal Variances



This example uses the AIC for model selection. Figure 6.16 shows the AIC and other model fit criteria for each of the fitted models.

To require that the separate components have identical variances, add the EQUATE=SCALE option in the MODEL statement:

```
title2 "Three to Seven Components, Equal Variances";
proc hpfmm data=galaxies criterion=AIC gconv=0;
   model v = / kmin=3 kmax=7  equate=scale;
run;
```

The GCONV= convergence criterion is turned off in this PROC HPFMM run to avoid the early stoppage of the iterations when the relative gradient changes little between iterations. Turning the criterion off usually ensures that convergence is achieved with a small absolute gradient of the objective function.

The output for equal variances is shown in Figure 6.17 and Figure 6.18.

Figure 6.17 Model Selection for Galaxy Data Assuming Equal Variances

HPFMM Analysis of Galaxies Data
Three to Seven Components, Equal Variances

The HPFMM Procedure

Model Information	
Data Set	GALAXIES
Response Variable	v
Type of Model	Homogeneous Mixture
Distribution	Normal
Min Components	3
Max Components	7
Link Function	Identity
Estimation Method	Maximum Likelihood

Component Evaluation for Mixture Models

Model ID	Number of Components Total	Components Eff.	Parameters Total	Parameters Eff.	-2 Log L	AIC	AICC	BIC	Pearson	Max Gradient
1	3	3	6	6	478.74	490.74	491.86	505.18	82.00	1.197E-6
2	4	4	8	8	416.49	432.49	434.47	451.75	82.00	3.913E-6
3	5	5	10	10	416.49	436.49	439.59	460.56	82.00	4.319E-6
4	6	6	12	12	416.49	440.49	445.02	469.37	82.00	6.294E-6
5	7	7	14	14	416.49	444.49	450.76	478.19	82.00	4.885E-6

The model with 4 components (ID=2) was selected as 'best' based on the AIC statistic.

Fit Statistics	
-2 Log Likelihood	416.5
AIC (Smaller is Better)	432.5
AICC (Smaller is Better)	434.5
BIC (Smaller is Better)	451.7
Pearson Statistic	82.0000
Effective Parameters	8
Effective Components	4

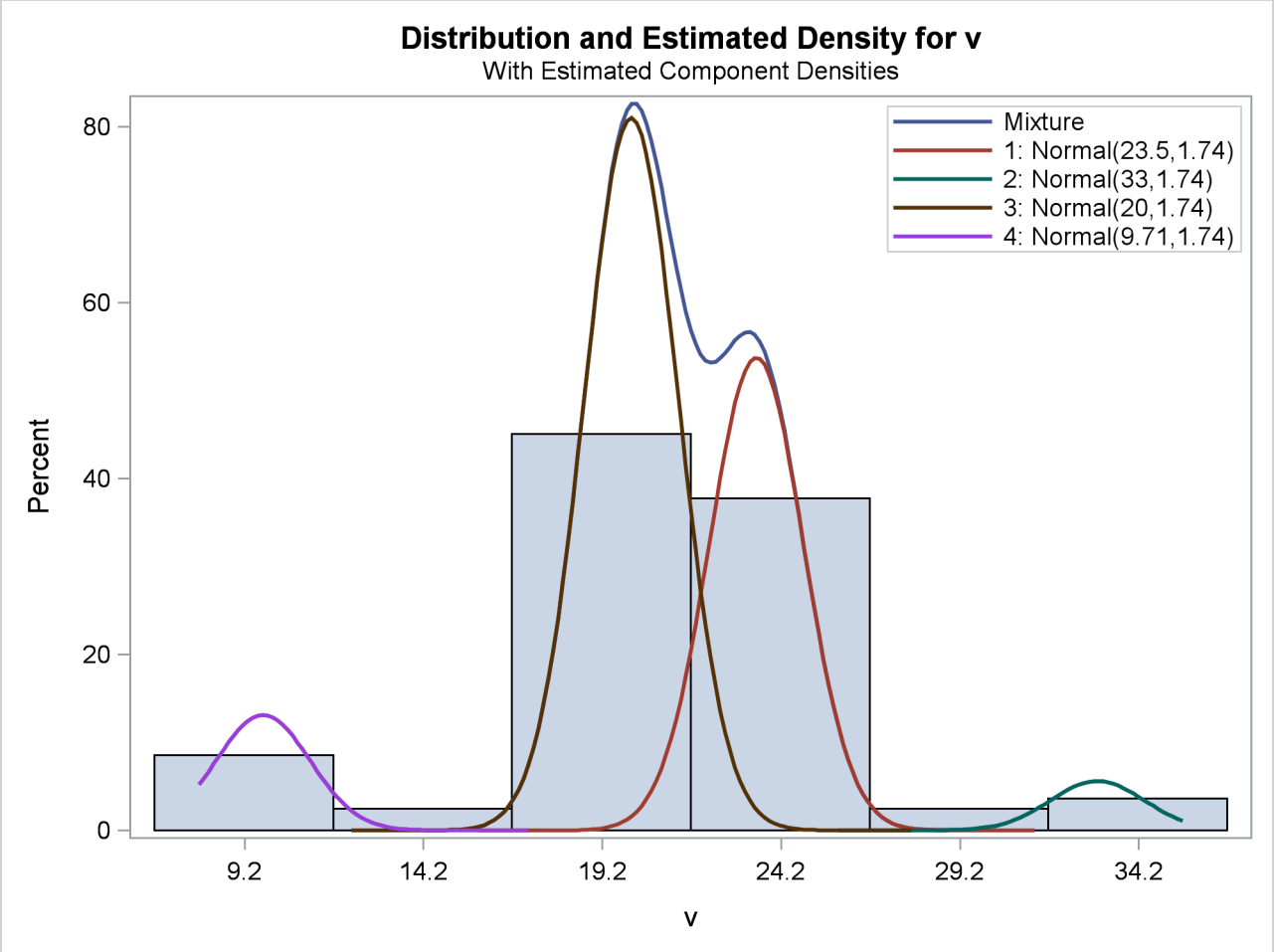
Parameter Estimates for Normal Model

Component	Parameter	Estimate	Standard Error	z Value	Pr > |z|
1	Intercept	23.5058	0.3460	67.93	<.0001
2	Intercept	33.0440	0.7610	43.42	<.0001
3	Intercept	20.0086	0.3029	66.06	<.0001
4	Intercept	9.7103	0.4981	19.50	<.0001
1	Variance	1.7354	0.3905		
2	Variance	1.7354	0.3905		
3	Variance	1.7354	0.3905		
4	Variance	1.7354	0.3905		

Figure 6.17 *continued*

Parameter Estimates for Mixing Probabilities					
		Linked Scale			
Component	Mixing Probability	GLogit(Prob)	Standard Error	z Value	Pr > |z|
1	0.3503	1.4118	0.4497	3.14	0.0017
2	0.0366	-0.8473	0.6901	-1.23	0.2195
3	0.5277	1.8216	0.4205	4.33	<.0001
4	0.0854	0			

Figure 6.18 Density Plot for Best (Six-Component) Model Assuming Equal Variances



Not surprisingly, the two variance specifications produce different optimal models. The unequal variance specification favors a three-component model while the equal variance specification favors a four-component model. Comparison of the AIC fit statistics, 423.0 and 432.5, indicates that the three-component, unequal variance model provides the best overall fit.

Comparison with Roeder's Method

It is important to note that Roeder's original analysis proceeds in a different manner than the finite mixture modeling presented here. The technique presented by Roeder first develops a "best" range of scale parameters based on a specific criterion. Roeder then uses fixed scale parameters taken from this range to develop optimal equal-scale Gaussian mixture models.

You can reproduce Roeder's point estimate for the density by specifying a five-component Gaussian mixture. In addition, use the EQUATE=SCALE option in the MODEL statement and a RESTRICT statement fixing the first component's scale parameter at 0.9025 (Roeder's $h = 0.95$, scale$= h^2$). The combination of these options produces a mixture of five Gaussian components, each with variance 0.9025. The following statements conduct this analysis:

```
title2 "Five Components, Equal Variances = 0.9025";
proc hpfmm data=galaxies;
   model v = / K=5 equate=scale;
   restrict int 0 (scale 1) = 0.9025;
run;
ods graphics off;
```

The output is shown in Figure 6.19 and Figure 6.20.

Figure 6.19 Reproduction of Roeder's Five-Component Analysis of Galaxy Data

HPFMM Analysis of Galaxies Data
Five Components, Equal Variances = 0.9025

The HPFMM Procedure

Model Information	
Data Set	GALAXIES
Response Variable	v
Type of Model	Homogeneous Mixture
Distribution	Normal
Components	5
Link Function	Identity
Estimation Method	Maximum Likelihood

Fit Statistics	
-2 Log Likelihood	412.2
AIC (Smaller is Better)	430.2
AICC (Smaller is Better)	432.7
BIC (Smaller is Better)	451.9
Pearson Statistic	82.5549
Effective Parameters	9
Effective Components	5

Linear Constraints at Solution

k = 1			Constraint Active
Variance	=	0.90	Yes

Figure 6.19 *continued*

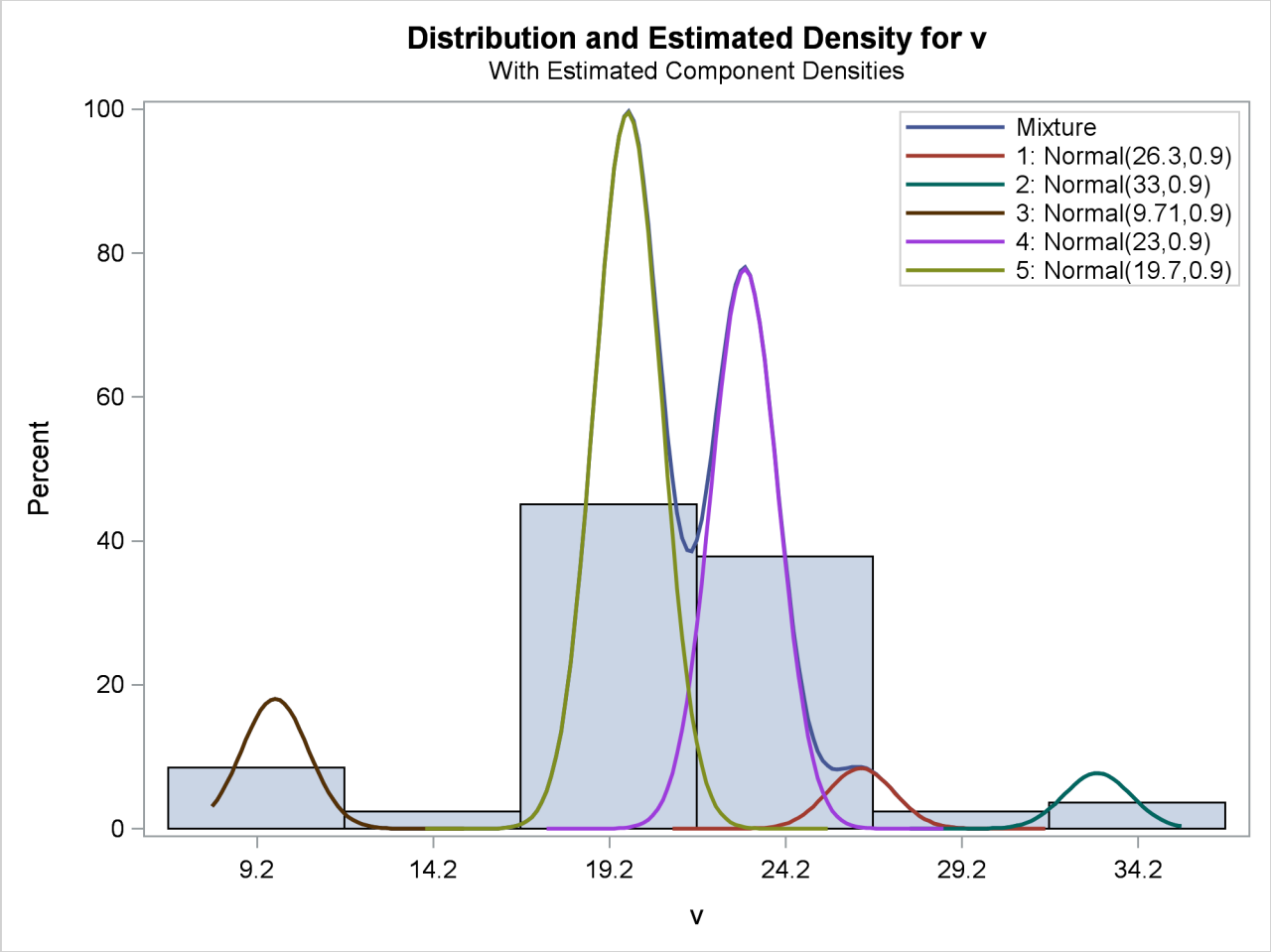
Parameter Estimates for Normal Model

Component	Parameter	Estimate	Standard Error	z Value	Pr > |z|
1	Intercept	26.3266	0.7778	33.85	<.0001
2	Intercept	33.0443	0.5485	60.25	<.0001
3	Intercept	9.7101	0.3591	27.04	<.0001
4	Intercept	23.0295	0.2294	100.38	<.0001
5	Intercept	19.7187	0.1784	110.55	<.0001
1	Variance	0.9025	0		
2	Variance	0.9025	0		
3	Variance	0.9025	0		
4	Variance	0.9025	0		
5	Variance	0.9025	0		

Parameter Estimates for Mixing Probabilities

		Linked Scale			
Component	Mixing Probability	GLogit(Prob)	Standard Error	z Value	Pr > |z|
1	0.0397	-2.4739	0.7084	-3.49	0.0005
2	0.0366	-2.5544	0.6016	-4.25	<.0001
3	0.0854	-1.7071	0.4141	-4.12	<.0001
4	0.3678	-0.2466	0.2699	-0.91	0.3609
5	0.4706	0			

Figure 6.20 Density Plot for Roeder's Analysis



Syntax: HPFMM Procedure

The following statements are available in the HPFMM procedure:

```
PROC HPFMM < options > ;
   BAYES bayes-options ;
   BY variables ;
   CLASS variables ;
   FREQ variable ;
   ID variables ;
   MODEL response< (response-options) > = < effects > < / model-options > ;
   MODEL events/trials = < effects > < / model-options > ;
   MODEL                 + < effects > < / model-options > ;
   OUTPUT < OUT=SAS-data-set >
            < keyword< (keyword-options) > < =name > > . . .
            < keyword< (keyword-options) > < =name > > < / options > ;
   PERFORMANCE performance-options ;
   PROBMODEL < effects > < / probmodel-options > ;
   RESTRICT < 'label' > constraint-specification < , . . . , constraint-specification >
              < operator < value > > < / option > ;
   WEIGHT variable ;
```

The PROC HPFMM statement and at least one MODEL statement is required. The CLASS, RESTRICT and MODEL statements can appear multiple times. If a CLASS statement is specified, it must precede the MODEL statements. The RESTRICT statements must appear after the MODEL statements.

PROC HPFMM Statement

```
PROC HPFMM < options > ;
```

The PROC HPFMM statement invokes the HPFMM procedure. Table 6.2 summarizes the *options* available in the PROC HPFMM statement. These and other *options* in the PROC HPFMM statement are then described fully in alphabetical order.

Table 6.2 PROC HPFMM Statement Options

Option	Description
Basic Options	
DATA=	Specifies the input data set
EXCLUSION=	Specifies how the procedure responds to support violations in the data
NAMELEN=	Specifies the length of effect names
SEED=	Specifies the random number seed for analyses that require random number draws

Table 6.2 *continued*

Option	Description
Displayed Output	
COMPONENTINFO	Displays information about the mixture components
CORR	Displays the asymptotic correlation matrix of the maximum likelihood parameter estimates or the empirical correlation matrix of the Bayesian posterior estimates
COV	Displays the asymptotic covariance matrix of the maximum likelihood parameter estimates or the empirical covariance matrix of the Bayesian posterior estimates
COVI	Displays the inverse of the covariance matrix of the parameter estimates
FITDETAILS	Displays fit information for all examined models
ITDETAILS	Adds estimates and gradients to the "Iteration History" table
NOCLPRINT	Suppresses the "Class Level Information" table completely or partially
NOITPRINT	Suppresses the "Iteration History Information" table
NOPRINT	Suppresses tabular and graphical output
PARMSTYLE=	Specifies how parameters are displayed in ODS tables
PLOTS	Produces ODS statistical graphics
Computational Options	
CRITERION=	Specifies the criterion used in model selection
NOCENTER	Prevents centering and scaling of the regressor variables
PARTIAL=	Specifies a variable that defines a partial classification
Options Related to Optimization	
ABSCONV=	Tunes an absolute function convergence criterion
ABSFCONV=	Tunes an absolute function difference convergence criterion
ABSGCONV=	Tunes the absolute gradient convergence criterion
FCONV=	Specifies a relative function convergence criterion that is based on a relative change of the function value
FCONV2=	Specifies a relative function convergence criterion that is based on a predicted reduction of the objective function
GCONV=	Tunes the relative gradient convergence criterion
MAXITER=	Specifies the maximum number of iterations in any optimization
MAXFUNC=	Specifies the maximum number of function evaluations in any optimization
MAXTIME=	Specifies the upper limit of CPU time in seconds for any optimization
MINITER=	Specifies the minimum number of iterations in any optimization
TECHNIQUE=	Selects the optimization technique

Table 6.2 *continued*

Option	Description
Singularity Tolerances	
INVALIDLOGL=	Tunes the value assigned to an invalid component log likelihood
SINGCHOL=	Tunes singularity for Cholesky decompositions
SINGRES=	Tunes singularity for the residual variance
SINGULAR=	Tunes general singularity criterion

You can specify the following *options* in the PROC HPFMM statement.

ABSCONV=*r*

ABSTOL=*r*

specifies an absolute function convergence criterion. For minimization, the termination criterion is $f(\boldsymbol{\psi}^{(k)}) \leq r$, where $\boldsymbol{\psi}$ is the vector of parameters in the optimization and $f(\cdot)$ is the objective function. The default value of *r* is the negative square root of the largest double-precision value, which serves only as a protection against overflows.

ABSFCONV=*r* **<** *n* **>**

ABSFTOL=*r***<** *n* **>**

specifies an absolute function difference convergence criterion. For all techniques except NMSIMP, the termination criterion is a small change of the function value in successive iterations:

$$|f(\boldsymbol{\psi}^{(k-1)}) - f(\boldsymbol{\psi}^{(k)})| \leq r$$

Here, $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization, and $f(\cdot)$ is the objective function. The same formula is used for the NMSIMP technique, but $\boldsymbol{\psi}(k)$ is defined as the vertex with the lowest function value, and $\boldsymbol{\psi}^{(k-1)}$ is defined as the vertex with the highest function value in the simplex. The default value is *r*=0. The optional integer value *n* specifies the number of successive iterations for which the criterion must be satisfied before the process can be terminated.

ABSGCONV=*r* **<** *n* **>**

ABSGTOL=*r***<** *n* **>**

specifies an absolute gradient convergence criterion. The termination criterion is a small maximum absolute gradient element:

$$\max_j |g_j(\boldsymbol{\psi}^{(k)})| \leq r$$

Here, $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization, and $g_j(\cdot)$ is the gradient of the objective function with respect to the *j*th parameter. This criterion is not used by the NMSIMP technique. The default value is *r*=1E–5. The optional integer value *n* specifies the number of successive iterations for which the criterion must be satisfied before the process can be terminated.

COMPONENTINFO

COMPINFO

CINFO

produces a table with additional details about the fitted model components.

COV

produces the covariance matrix of the parameter estimates. For maximum likelihood estimation, this matrix is based on the inverse (projected) Hessian matrix. For Bayesian estimation, it is the empirical covariance matrix of the posterior estimates. The covariance matrix is shown for all parameters, even if they did not participate in the optimization or sampling.

COVI

produces the inverse of the covariance matrix of the parameter estimates. For maximum likelihood estimation, the covariance matrix is based on the inverse (projected) Hessian matrix. For Bayesian estimation, it is the empirical covariance matrix of the posterior estimates. This matrix is then inverted by sweeping, and rows and columns that correspond to linear dependencies or singularities are zeroed.

CORR

produces the correlation matrix of the parameter estimates. For maximum likelihood estimation this matrix is based on the inverse (projected) Hessian matrix. For Bayesian estimation, it is based on the empirical covariance matrix of the posterior estimates.

CRITERION=*keyword*

CRIT=*keyword*

specifies the criterion by which the HPFMM procedure ranks models when multiple models are evaluated during maximum likelihood estimation. You can choose from the following *keywords* to rank models:

AIC	based on Akaike's information criterion
AICC	based on the bias-corrected AIC criterion
BIC	based on the Bayesian information criterion
GRADIENT	based on the largest element of the gradient (in absolute value)
LOGL | LL	based on the mixture log likelihood
PEARSON	based on the Pearson statistic

The default is CRITERION=BIC.

DATA=*SAS-data-set*

names the SAS data set to be used by PROC HPFMM. The default is the most recently created data set.

EXCLUSION=NONE | ANY | ALL

EXCLUDE=NONE | ANY | ALL

specifies how the HPFMM procedure handles support violations of observations. For example, in a mixture of two Poisson variables, negative response values are not possible. However, in a mixture of a Poisson and a normal variable, negative values are possible, and their likelihood contribution to the Poisson component is zero. An observation that violates the support of one component distribution of the model might be a valid response with respect to one or more other component distributions. This requires some nuanced handling of support violations in mixture models.

The default exclusion technique, EXCLUSION=ALL, removes an observation from the analysis only if it violates the support of all component distributions. The other extreme, EXCLUSION=NONE, permits an observation into the analysis regardless of support violations. EXCLUSION=ANY removes observations from the analysis if the response violates the support of any component distributions. In the single-component case, EXCLUSION=ALL and EXCLUSION=ANY are identical.

FCONV=*r*< *n* >

FTOL=*r*< *n* >

specifies a relative function convergence criterion that is based on the relative change of the function value. For all techniques except NMSIMP, PROC HPFMM terminates when there is a small relative change of the function value in successive iterations:

$$\frac{|f(\boldsymbol{\psi}^{(k)}) - f(\boldsymbol{\psi}^{(k-1)})|}{|f(\boldsymbol{\psi}^{(k-1)})|} \leq r$$

Here, $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization, and $f(\cdot)$ is the objective function. The same formula is used for the NMSIMP technique, but $\boldsymbol{\psi}^{(k)}$ is defined as the vertex with the lowest function value, and $\boldsymbol{\psi}^{(k-1)}$ is defined as the vertex with the highest function value in the simplex.

The default is $r = 10^{-\mathrm{FDIGITS}}$, where FDIGITS is by default $-\log_{10}\{\epsilon\}$, and ϵ is the machine precision. The optional integer value *n* specifies the number of successive iterations for which the criterion must be satisfied before the process terminates.

FCONV2=*r*< *n* >

FTOL2=*r*< *n* >

specifies a relative function convergence criterion that is based on the predicted reduction of the objective function. For all techniques except NMSIMP, the termination criterion is a small predicted reduction

$$df^{(k)} \approx f(\boldsymbol{\theta}^{(k)}) - f(\boldsymbol{\theta}^{(k)} + \mathrm{s}^{(k)})$$

of the objective function. The predicted reduction

$$\begin{aligned} df^{(k)} &= -\mathbf{g}^{(k)\prime}\mathrm{s}^{(k)} - \frac{1}{2}\mathrm{s}^{(k)\prime}\mathbf{H}^{(k)}\mathrm{s}^{(k)} \\ &= -\frac{1}{2}\mathrm{s}^{(k)\prime}\mathbf{g}^{(k)} \\ &\leq r \end{aligned}$$

is computed by approximating the objective function f by the first two terms of the Taylor series and substituting the Newton step:

$$\mathrm{s}^{(k)} = -[\mathbf{H}^{(k)}]^{-1}\mathbf{g}^{(k)}$$

For the NMSIMP technique, the termination criterion is a small standard deviation of the function values of the $n+1$ simplex vertices $\boldsymbol{\theta}_l^{(k)}$, $l = 0, \ldots, n$,

$$\sqrt{\frac{1}{n+1}\sum_l \left[f(\boldsymbol{\theta}_l^{(k)}) - \overline{f}(\boldsymbol{\theta}^{(k)})\right]^2} \leq r$$

where $\overline{f}(\boldsymbol{\theta}^{(k)}) = \frac{1}{n+1}\sum_l f(\boldsymbol{\theta}_l^{(k)})$. If there are n_{act} boundary constraints active at $\boldsymbol{\theta}^{(k)}$, the mean and standard deviation are computed only for the $n + 1 - n_{act}$ unconstrained vertices.

The default value is *r* = 1E–6 for the NMSIMP technique and *r* = 0 otherwise. The optional integer value *n* specifies the number of successive iterations for which the criterion must be satisfied before the process terminates.

FITDETAILS

requests that the "Optimization Information," "Iteration History," and "Fit Statistics" tables be produced for all optimizations when models with different number of components are evaluated. For example, the following statements fit a binomial regression model with up to three components and produces fit and optimization information for all three:

```
proc hpfmm fitdetails;
   model y/n  = x / kmax=3;
run;
```

Without the FITDETAILS option, only the "Fit Statistics" table for the selected model is displayed.

In Bayesian estimation, the FITDETAILS option displays the following tables for each model that the procedure fits: "Bayes Information," "Iteration History," "Prior Information," "Fit Statistics," "Posterior Summaries," "Posterior Intervals," and any requested diagnostics tables. The "Iteration History" table appears only if the BAYES statement includes the INITIAL=MLE option.

Without the FITDETAILS option, these tables are listed only for the selected model.

GCONV=*r*<*n*>

GTOL=*r*<*n*>

specifies a relative gradient convergence criterion. For all techniques except CONGRA and NMSIMP, the termination criterion is a small normalized predicted function reduction:

$$\frac{\mathbf{g}(\boldsymbol{\psi}^{(k)})'[\mathbf{H}^{(k)}]^{-1}\mathbf{g}(\boldsymbol{\psi}^{(k)})}{|f(\boldsymbol{\psi}^{(k)})|} \leq r$$

Here, $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization, $f(\cdot)$ is the objective function, and $\mathbf{g}(\cdot)$ is the gradient. For the CONGRA technique (where a reliable Hessian estimate $\mathbf{H}$ is not available), the following criterion is used:

$$\frac{\| \mathbf{g}(\boldsymbol{\psi}^{(k)}) \|_2^2 \quad \| \mathbf{s}(\boldsymbol{\psi}^{(k)}) \|_2}{\| \mathbf{g}(\boldsymbol{\psi}^{(k)}) - \mathbf{g}(\boldsymbol{\psi}^{(k-1)}) \|_2 \; |f(\boldsymbol{\psi}^{(k)})|} \leq r$$

This criterion is not used by the NMSIMP technique. The default value is *r*=1E–8. The optional integer value *n* specifies the number of successive iterations for which the criterion must be satisfied before the process can terminate.

HESSIAN

displays the Hessian matrix of the model. This option is not available for Bayesian estimation.

INVALIDLOGL=*r*
specifies the value assumed by the HPFMM procedure if a log likelihood cannot be computed (for example, because the value of the response variable falls outside of the response distribution's support). The default value is –1E20.

ITDETAILS
adds parameter estimates and gradients to the "Iteration History" table. If the HPFMM procedure centers or scales the model variables (or both), the parameter estimates and gradients reported during the iteration refer to that scale. You can suppress centering and scaling with the NOCENTER option.

MAXFUNC=*n*
MAXFU=*n*
specifies the maximum number of function calls in the optimization process. The default values are as follows, depending on the optimization technique:

- TRUREG, NRRIDG, and NEWRAP: 125
- QUANEW and DBLDOG: 500
- CONGRA: 1000
- NMSIMP: 3000

The optimization can terminate only after completing a full iteration. Therefore, the number of function calls that are actually performed can exceed the number that is specified by the MAXFUNC= option. You can choose the optimization technique with the TECHNIQUE= option.

MAXITER=*n*
MAXIT=*n*
specifies the maximum number of iterations in the optimization process. The default values are as follows, depending on the optimization technique:

- TRUREG, NRRIDG, and NEWRAP: 50
- QUANEW and DBLDOG: 200
- CONGRA: 400
- NMSIMP: 1000

These default values also apply when *n* is specified as a missing value. You can choose the optimization technique with the TECHNIQUE= option.

MAXTIME=*r*
specifies an upper limit of *r* seconds of CPU time for the optimization process. The time is checked only at the end of each iteration. Therefore, the actual run time might be longer than the specified time. By default, CPU time is not limited.

MINITER=*n*
MINIT=*n*
specifies the minimum number of iterations. The default value is 0. If you request more iterations than are actually needed for convergence to a stationary point, the optimization algorithms can behave strangely. For example, the effect of rounding errors can prevent the algorithm from continuing for the required number of iterations.

NAMELEN=*number*
specifies the length to which long effect names are shortened. The default and minimum value is 20.

NOCENTER
requests that regressor variables not be centered or scaled. By default the HPFMM procedure centers and scales columns of the **X** matrix if the models contain intercepts. If NOINT options in MODEL statements are in effect, the columns of **X** are scaled but not centered. Centering and scaling can help with the stability of estimation and sampling algorithms. The HPFMM procedure does not produce a table of the centered and scaled coefficients and provides no user control over the type of centering and scaling that is applied. The NOCENTER option turns any centering and scaling off and processes the raw values of the continuous variables.

NOCLPRINT< =*number* **>**
suppresses the display of the "Class Level Information" table if you do not specify *number*. If you specify *number*, the values of the classification variables are displayed for only those variables whose number of levels is less than *number*. Specifying a *number* helps to reduce the size of the "Class Level Information" table if some classification variables have a large number of levels.

NOITPRINT
suppresses the display of the "Iteration History Information" table.

NOPRINT
suppresses the normal display of tabular and graphical results. The NOPRINT option is useful when you want to create only one or more output data sets with the procedure. This option temporarily disables the Output Delivery System (ODS); see Chapter 20, "Using the Output Delivery System" (*SAS/STAT User's Guide*), for more information.

PARMSTYLE=EFFECT | LABEL
specifies the display style for parameters and effects. The HPFMM procedure can display parameters in two styles:

- The EFFECT style (which is used by the MIXED and GLIMMIX procedure, for example) identifies a parameter with an "Effect" column and adds separate columns for the CLASS variables in the model.
- The LABEL style creates one column, named Parameter, that combines the relevant information about a parameter into a single column. If your model contains multiple CLASS variables, the LABEL style might use space more economically.

The EFFECT style is the default for models that contain effects; otherwise the LABEL style is used (for example, in homogeneous mixtures). You can change the display style with the PARMSTYLE= option. Regardless of the display style, ODS output data sets that contain information about parameter estimates contain columns for both styles.

PARTIAL=*variable*

MEMBERSHIP=*variable*
specifies a variable in the input data set that identifies component membership. You can specify missing values for observations whose component membership is undetermined; this is known as a partial classification (McLachlan and Peel 2000, p. 75). For observations with known membership, the

likelihood contribution is no longer a mixture. If observation i is known to be a member of component m, then its log likelihood contribution is

$$\log\left\{\pi_m(\mathbf{z},\boldsymbol{\alpha}_m)\ \ p_m(y;\mathbf{x}_m'\boldsymbol{\beta}_m,\phi_m)\right\}$$

Otherwise, if membership is undetermined, it is

$$\log\left\{\sum_{j=1}^{k}\ \pi_j(\mathbf{z},\boldsymbol{\alpha}_j)\,p_j(y;\mathbf{x}_j'\boldsymbol{\beta}_j,\phi_j)\right\}$$

The variable specified in the PARTIAL= option can be numeric or character. In case of a character variable, the variable must appear in the CLASS statement. If the PARTIAL= variable appears in the CLASS statement, the membership assignment is made based on the levelized values of the variable, as shown in the "Class Level Information" table. Invalid values of the PARTIAL= variable are ignored.

In a model in which label switching is a problem, the switching can sometimes be avoided by assigning just a few observations to categories. For example, in a three-component model, switches might be prevented by assigning the observation with the smallest response value to the first component and the observation with the largest response value to the last component.

PLOTS <(*global-plot-options*)> <=*plot-request* <(*options*)>>

PLOTS <(*global-plot-options*)> <=(*plot-request* <(*options*)> <... *plot-request* <(*options*)>>)>
controls the plots produced through ODS Graphics.

ODS Graphics must be enabled before plots can be requested. For example:

```
ods graphics on;
proc hpfmm data=yeast seed=12345;
   model count/n = / k=2;
   freq f;
   performance nthreads=2;
   bayes;
run;
ods graphics off;
```

Global Plot Options

The *global-plot-options* apply to all relevant plots generated by the HPFMM procedure. The *global-plot-options* supported by the HPFMM procedure are as follows:

UNPACKPANEL

UNPACK
displays each graph separately. (By default, some graphs can appear together in a single panel.)

ONLY
produces only the specified plots. This option is useful if you do not want the procedure to generate all default graphics, but only the ones specified.

Specific Plot Options

The following listing describes the specific plots and their *options*.

ALL
requests that all plots appropriate for the analysis be produced.

NONE
requests that no ODS graphics be produced.

DENSITY < (*density-options*) >
requests a plot of the data histogram and mixture density function. This graphic is a default graphic in models without effects in the MODEL statements and is available only in these models. Furthermore, all distributions involved in the mixture must be continuous. You can specify the following *density-options* to modify the plot:

CUMULATIVE
CDF
displays the histogram and densities in cumulative form.

NBINS=*n*
BINS=*n*
specifies the number of bins in the histogram; *n* is greater than or equal to 0. By default, the HPFMM procedure computes a suitable bin width and number of bins, based on the range of the response and the number of usable observations. The option has no effect for binary data.

NOCOMPONENTS
NOCOMP
suppresses the component densities from the plot. If the component densities are displayed, they are scaled so that their sum equals the mixture density at any point on the graph. In single-component models, this option has no effect.

NODENSITY
NODENS
suppresses the computation of the mixture density (and the component densities if the COMPONENTS suboption is specified). If you specify the NOHISTOGRAM and the NODENSITY option, no graphic is produced.

NOLABEL
suppresses the component identification with labels. By default, the HPFMM procedure labels component densities in the legend of the plot. If you do not specify a model label with the LABEL= option in the MODEL statement, an identifying label is constructed from the parameter estimates that are associated with the component. In this case the parameter values are not necessarily the mean and variance of the distribution; the values used to identify the densities on the plot are chosen to simplify linking between graphical and tabular results.

NOHISTOGRAM

NOHIST

suppresses the computation of the histogram of the raw values. If you specify the NOHISTOGRAM and the NODENSITY option, no graphic is produced.

NPOINTS=*n*

N=*n*

specifies the number of values used to compute the density functions; *n* is greater than or equal to 0. The default is N=200.

WIDTH=*value*

BINWIDTH=*value*

specifies the bin width for the histogram. The *value* is specified in units of the response variable and must be positive. The option has no effect for binary data.

TRACE < (*tadpanel-options***) >**

requests a trace panel with posterior diagnostics for a Bayesian analysis. If a BAYES statement is present, the trace panel plots are generated by default, one for each sampled parameter. You can specify the following *tadpanel-options* to modify the graphic:

BOX

BOXPLOT

replaces the autocorrelation plot with a box plot of the posterior sample.

SMOOTH=NONE | MEAN | SPLINE

adds a reference estimate to the trace plot. By default, SMOOTH=NONE. SMOOTH=MEAN uses the arithmetic mean of the trace as the reference. SMOOTH=SPLINE adds a penalized B-spline.

REFERENCE= *reference-style*

adds vertical reference lines to the density plot, trace plot, and box plot. The available options for the *reference-style* are:

NONE	suppresses the reference lines
EQT	requests equal-tail intervals
HPD	requests intervals of highest posterior density. The level for the credible or HPD intervals is chosen based on the "Posterior Interval Statistics" table.
PERCENTILES	(or PERC) for percentiles. Up to three percentiles can be displayed, as based on the "Posterior Summary Statistics" table.

The default is REFERENCE=EQT.

UNPACK

unpacks the panel graphic and displays its elements as separate plots.

CRITERIONPANEL <(*critpanel-options*)>
requests a plot for comparing the model fit criteria for different numbers of components. This plot is available only if you also specify the KMAX option in at least one MODEL statement. The plot includes different criteria, depending on whether you are using maximum likelihood or Bayesian estimation. You can specify the following *critpanel-option* to modify the plot:

UNPACK
unpacks the panel plot and displays its elements as separate plots, one for each fit criterion.

SEED=*n*
determines the random number seed for analyses that depend on a random number stream. If you do not specify a seed or if you specify a value less than or equal to zero, the seed is generated from reading the time of day from the computer clock. The largest possible value for the seed is $2^{31} - 1$. The seed value is reported in the "Model Information" table.

You can use the SYSRANDOM and SYSRANEND macro variables after a PROC HPFMM run to query the initial and final seed values. However, using the final seed value as the starting seed for a subsequent analysis does not continue the random number stream where the previous analysis left off. The SYSRANEND macro variable provides a mechanism to pass on seed values to ensure that the sequence of random numbers is the same every time you run an entire program.

Analyses that use the same (nonzero) seed are not completely reproducible if they are executed with a different number of threads since the random number streams in separate threads are independent. You can control the number of threads used by the HPFMM procedure with system options or through the PERFORMANCE statement in the HPFMM procedure.

SINGCHOL=*number*
tunes the singularity criterion in Cholesky decompositions. The default is 1E4 times the machine epsilon; this product is approximately 1E–12 on most computers.

SINGRES=*number*
sets the tolerance for which the residual variance or scale parameter is considered to be zero. The default is 1E4 times the machine epsilon; this product is approximately 1E–12 on most computers.

SINGULAR=*number*
tunes the general singularity criterion applied by the HPFMM procedure in sweeps and inversions. The default is 1E4 times the machine epsilon; this product is approximately 1E–12 on most computers.

TECHNIQUE=*keyword*

TECH=*keyword*
specifies the optimization technique to obtain maximum likelihood estimates. You can choose from the following techniques by specifying the appropriate *keyword*:

CONGRA	performs a conjugate-gradient optimization.
DBLDOG	performs a version of double-dogleg optimization.
NEWRAP	performs a Newton-Raphson optimization combining a line-search algorithm with ridging.
NMSIMP	performs a Nelder-Mead simplex optimization.
NONE	performs no optimization.

NRRIDG performs a Newton-Raphson optimization with ridging.

QUANEW performs a dual quasi-Newton optimization.

TRUREG performs a trust-region optimization.

The default is TECH=QUANEW.

For more details about these optimization methods, see the section "Choosing an Optimization Algorithm" on page 187.

ZEROPROB=*number*

tunes the threshold (a value between 0 and 1) below which the HPFMM procedure considers a component mixing probability to be zero. This affects the calculation of the number of effective components. The default is the square root of the machine epsilon; this is approximately 1E–8 on most computers.

BAYES Statement

BAYES *bayes-options* **;**

The BAYES statement requests that the parameters of the model be estimated by Markov chain Monte Carlo sampling techniques. The HPFMM procedure can estimate by maximum likelihood the parameters of all models supported by the procedure. Bayes estimation, on the other hand, is available for only a subset of these models.

In Bayesian analysis, it is essential to examine the convergence of the Markov chains before you proceed with posterior inference. With ODS Graphics turned on, the HPFMM procedure produces graphs at the end of the procedure output; these graphs enable you to visually examine the convergence of the chain. Inferences cannot be made if the Markov chain has not converged.

The output produced for a Bayesian analysis is markedly different from that for a frequentist (maximum likelihood) analysis for the following reasons:

- Parameter estimates do not have the same interpretation in the two analyses. Parameters are fixed unknown constants in the frequentist context and random variables in a Bayesian analysis.
- The results of a Bayesian analysis are summarized through chain diagnostics and posterior summary statistics and intervals.
- The HPFMM procedure samples the mixing probabilities in Bayesian models directly, rather than mapping them onto a logistic (or other) scale.

The HPFMM procedure applies highly specialized sampling algorithms in Bayesian models. For single-component models without effects, a conjugate sampling algorithm is used where possible. For models in the exponential family that contain effects, the sampling algorithm is based on Gamerman (1997). For the normal and *t* distributions, a conjugate sampler is the default sampling algorithm for models with and without effects. In multi-component models, the sampling algorithm is based on latent variable sampling through data augmentation (Frühwirth-Schnatter 2006) and the Gamerman or conjugate sampler. Because of this specialization, the options for controlling the prior distributions of the parameters are limited.

Table 6.3 summarizes the *bayes-options* available in the BAYES statement. The full assortment of options is then described in alphabetical order.

Table 6.3 BAYES Statement Options

Option	Description
Options Related to Sampling	
INITIAL=	Specifies how to construct initial values
NBI=	Specifies the number of burn-in samples
NMC=	Specifies the number of samples after burn-in
METROPOLIS	Forces a Metropolis-Hastings sampling algorithm even if conjugate sampling is possible
OUTPOST=	Generates a data set that contains the posterior estimates
THIN=	Controls the thinning of the Markov chain
Specification of Prior Information	
MIXPRIORPARMS	Specifies the prior parameters for the Dirichlet distribution of the mixing probabilities
BETAPRIORPARMS=	Specifies the parameters of the normal prior distribution for individual parameters in the $\boldsymbol{\beta}$ vector
MUPRIORPARMS=	Specifies the parameters of the prior distribution for the means in homogeneous mixtures without effects
PHIPRIORPARMS=	Specifies the parameters of the inverse gamma prior distribution for the scale parameters in homogeneous mixtures
PRIOROPTIONS	Specifies additional options used in the determination of the prior distribution
Posterior Summary Statistics and Convergence Diagnostics	
DIAGNOSTICS=	Displays convergence diagnostics for the Markov chain
STATISTICS	Displays posterior summary information for the Markov chain
Other Options	
ESTIMATE=	Specifies which estimate is used for the computation of OUTPUT statistics and graphics
TIMEINC=	Specifies the time interval to report on sampling progress (in seconds)

You can specify the following *bayes-options* in the BAYES statement.

BETAPRIORPARMS=*pair-specification*

BETAPRIORPARMS(*pair-specification* **...** *pair-specification***)**

specifies the parameters for the normal prior distribution of the parameters that are associated with model effects (βs). The *pair-specification* is of the form (a, b), and the values a and b are the mean and variance of the normal distribution, respectively. This option overrides the PRIOROPTIONS option.

The form of the BETAPRIORPARMS with an equal sign and a single pair is used to specify one pair of prior parameters that applies to all components in the mixture. In the following example, the two intercepts and the two regression coefficients all have a $N(0, 100)$ prior distribution:

```
proc hpfmm;
   model y = x / k=2;
   bayes betapriorparms=(0,100);
run;
```

You can also provide a list of pairs to specify different sets of prior parameters for the various regression parameters and components. For example:

```
proc hpfmm;
   model y = x/ k=2;
   bayes betapriorparms( (0,10) (0,20) (.,.) (3,100) );
run;
```

The simple linear regression in the first component has a $N(0, 10)$ prior for the intercept and a $N(0, 20)$ prior for the slope. The prior for the intercept in the second component uses the HPFMM default, whereas the prior for the slope is $N(3, 100)$.

DIAGNOSTICS=ALL | NONE | (*keyword-list*)

DIAG=ALL | NONE | (*keyword-list*)

controls the computation of diagnostics for the posterior chain. You can request all posterior diagnostics by specifying DIAGNOSTICS=ALL or suppress the computation of posterior diagnostics by specifying DIAGNOSTICS=NONE. The following *keywords* enable you to select subsets of posterior diagnostics; the default is DIAGNOSTICS=(AUTOCORR).

AUTOCORR < (LAGS= *numeric-list*) >

computes for each sampled parameter the autocorrelations of lags specified in the LAGS= list. Elements in the list are truncated to integers, and repeated values are removed. If the LAGS= option is not specified, autocorrelations are computed by default for lags 1, 5, 10, and 50. See the section "Autocorrelations" (Chapter 7, *SAS/STAT User's Guide*) in Chapter 7, "Introduction to Bayesian Analysis Procedures" (*SAS/STAT User's Guide*), for details.

ESS

computes an estimate of the effective sample size (Kass et al. 1998), the correlation time, and the efficiency of the chain for each parameter. See the section "Effective Sample Size" (Chapter 7, *SAS/STAT User's Guide*) in Chapter 7, "Introduction to Bayesian Analysis Procedures" (*SAS/STAT User's Guide*), for details.

GEWEKE < (*geweke-options***) >**

computes the Geweke spectral density diagnostics (Geweke 1992), which are essentially a two-sample t test between the first f_1 portion and the last f_2 portion of the chain. The default is $f_1 = 0.1$ and $f_2 = 0.5$, but you can choose other fractions by using the following *geweke-options*:

FRAC1=*value*

specifies the fraction f_1 for the first window.

FRAC2=*value*
specifies the fraction f_2 for the second window.

See the section "Geweke Diagnostics" (Chapter 7, *SAS/STAT User's Guide*) in Chapter 7, "Introduction to Bayesian Analysis Procedures" (*SAS/STAT User's Guide*), for details.

HEIDELBERGER < (*Heidel-options***) >**

HEIDEL < (*Heidel-options***) >**
computes the Heidelberger and Welch diagnostic (which consists of a stationarity test and a half-width test) for each variable. The stationary diagnostic test tests the null hypothesis that the posterior samples are generated from a stationary process. If the stationarity test is passed, a half-width test is then carried out. See the section "Heidelberger and Welch Diagnostics" (Chapter 7, *SAS/STAT User's Guide*) in Chapter 7, "Introduction to Bayesian Analysis Procedures" (*SAS/STAT User's Guide*), for more details.

These diagnostics are not performed by default. You can specify the DIAGNOSTICS=HEIDELBERGER option to request these diagnostics, and you can also specify suboptions, such as DIAGNOSTICS=HEIDELBERGER(EPS=0.05), as follows:

SALPHA=*value*
specifies the α level ($0 < \alpha < 1$) for the stationarity test. By default, SALPHA=0.05.

HALPHA=*value*
specifies the α level ($0 < \alpha < 1$) for the half-width test. By default, HALPHA=0.05.

EPS=*value*
specifies a small positive number ϵ such that if the half-width is less than ϵ times the sample mean of the retaining iterates, the half-width test is passed. By default, EPS=0.1.

MCERROR

MCSE
computes an estimate of the Monte Carlo standard error for each sampled parameter. See the section "Standard Error of the Mean Estimate" (Chapter 7, *SAS/STAT User's Guide*) in Chapter 7, "Introduction to Bayesian Analysis Procedures" (*SAS/STAT User's Guide*), for details.

MAXLAG=*n*
specifies the largest lag used in computing the effective sample size and the Monte Carlo standard error. Specifying this option implies the ESS and MCERROR options. The default is MAXLAG=250.

RAFTERY < (*Raftery-options***) >**

RL < (*Raftery-options***) >**
computes the Raftery and Lewis diagnostics, which evaluate the accuracy of the estimated quantile ($\hat{\theta}_Q$ for a given $\mathrm{Q} \in (0, 1)$) of a chain. $\hat{\theta}_Q$ can achieve any degree of accuracy when the chain is allowed to run for a long time. The algorithm stops when the estimated probability $\hat{P}_Q = \Pr(\theta \leq \hat{\theta}_Q)$ reaches within $\pm$R of the value Q with probability S; that is, $\Pr(\mathrm{Q} - \mathrm{R} \leq \hat{\mathrm{P}}_Q \leq \mathrm{Q} + \mathrm{R}) = \mathrm{S}$. See the section "Raftery and Lewis Diagnostics" (Chapter 7, *SAS/STAT*

User's Guide) in Chapter 7, "Introduction to Bayesian Analysis Procedures" (*SAS/STAT User's Guide*), for more details. The *Raftery-options* enable you to specify Q, R, S, and a precision level ϵ for a stationary test.

These diagnostics are not performed by default. You can specify the DIAGNOSTICS=RAFERTY option to request these diagnostics, and you can also specify suboptions, such as DIAGNOSTICS=RAFERTY(QUANTILE=0.05), as follows:

QUANTILE=*value*
Q=*value*

specifies the order (a value between 0 and 1) of the quantile of interest. By default, QUANTILE=0.025.

ACCURACY=*value*
R=*value*

specifies a small positive number as the margin of error for measuring the accuracy of estimation of the quantile. By default, ACCURACY=0.005.

PROB=*value*
S=*value*

specifies the probability of attaining the accuracy of the estimation of the quantile. By default, PROB=0.95.

EPS=*value*

specifies the tolerance level (a small positive number between 0 and 1) for the stationary test. By default, EPS=0.001.

MIXPRIORPARMS=K
MIXPRIORPARMS(*value-list***)**

specifies the parameters used in constructing the Dirichlet prior distribution for the mixing parameters. If you specify MIXPRIORPARMS=K, the parameters of the k-dimensional Dirichlet distribution are a vector that contains the number of components in the model (k), whatever that might be. You can specify an explicit list of parameters in *value-list*. If the MIXPRIORPARMS option is not specified, the default Dirichlet parameter vector is a vector of length k of ones. This results in a uniform prior over the unit simplex; for k=2, this is the uniform distribution. See the section "Prior Distributions" on page 184 for the distribution function of the Dirichlet as used by the HPFMM procedure.

ESTIMATE=MEAN | MAP

determines which overall estimate is used, based on the posterior sample, in the computation of OUTPUT statistics and certain ODS graphics. By default, the arithmetic average of the (thinned) posterior sample is used. If you specify ESTIMATE=MAP, the parameter vector is used that corresponds to the maximum log posterior density in the posterior sample. In any event, a message is written to the SAS log if postprocessing results depend on a summary estimate of the posterior sample.

INITIAL=DATA | MLE | MODE | RANDOM

determines how initial values for the Markov chain are obtained. The default when a conjugate sampler is used is INITIAL=DATA, in which case the HPFMM procedure uses the same algorithm to obtain data-dependent starting values as it uses for maximum likelihood estimation. If no conjugate sampler is available or if you use the METROPOLIS option to explicitly request that it not be used, then the default is INITIAL=MLE, in which case the maximum likelihood estimates are used as the initial values. If the maximum likelihood optimization fails, the HPFMM procedure switches to the default INITIAL=DATA.

The options INITIAL=MODE and INITIAL=RANDOM use the mode and random draws from the prior distribution, respectively, to obtain initial values. If the mode does not exist or if it falls on the boundary of the parameter space, the prior mean is used instead.

METROPOLIS

requests that the HPFMM procedure use the Metropolis-Hastings sampling algorithm based on Gamerman (1997), even in situations where a conjugate sampler is available.

MUPRIORPARMS=*pair-specification*

MUPRIORPARMS(*pair-specification* **...** *pair-specification***)**

specifies the parameters for the means in homogeneous mixtures without regression coefficients. The *pair-specification* is of the form (a, b), where a and b are the two parameters of the prior distribution, optionally delimited with a comma. The actual distribution of the parameter is implied by the distribution selected in the MODEL statement. For example, it is a normal distribution for a mixture of normals, a gamma distribution for a mixture of Poisson variables, a beta distribution for a mixture of binary variables, and an inverse gamma distribution for a mixture of exponential variables. This option overrides the PRIOROPTIONS option.

The parameters correspond as follows:

Beta:	The parameters correspond to the α and β parameters of the beta prior distribution such that its mean is $\mu = \alpha/(\alpha+\beta)$ and its variance is $\mu(1-\mu)/(\alpha+\beta+1)$.
Normal:	The parameters correspond to the mean and variance of the normal prior distribution.
Gamma:	The parameters correspond to the α and β parameters of the gamma prior distribution such that its mean is α/β and its variance is α/β^2.
Inverse gamma:	The parameters correspond to the α and β parameters of the inverse gamma prior distribution such that its mean is $\mu = \beta/(\alpha-1)$ and its variance is $\mu^2/(\alpha-2)$.

The two techniques for specifying the prior parameters with the MUPRIORPARMS option are as follows:

- Specify an equal sign and a single pair of values:

```
proc hpfmm seed=12345;
   model y = / k=2;
   bayes mupriorparms=(0,50);
run;
```

- Specify a list of parameter pairs within parentheses:

```
proc hpfmm seed=12345;
   model y = / k=2;
   bayes mupriorparms( (.,.) (1.4,10.5));
run;
```

If you specify an invalid value (outside of the parameter space for the prior distribution), the HPFMM procedure chooses the default value and writes a message to the SAS log. If you want to use the default values for a particular parameter, you can also specify missing values in the *pair-specification*. For example, the preceding list specification assigns default values for the first component and uses the values 1.4 and 10.5 for the mean and variance of the normal prior distribution in the second component. The first example assigns a $N(0, 50)$ prior distribution to the means in both components.

NBI=*n*

specifies the number of burn-in samples. During the burn-in phase, chains are not saved. The default is NBI=2000.

NMC=*n*

SAMPLE=*n*

specifies the number of Monte Carlo samples after the burn-in. Samples after the burn-in phase are saved unless they are thinned with the THIN= option. The default is NMC=10000.

OUTPOST< (*outpost-options***) >=***data-set*

requests that the posterior sample be saved to a SAS data set. In addition to variables that contain log likelihood and log posterior values, the OUTPOST data set contains variables for the parameters. The variable names for the parameters are generic (Parm_1, Parm_2, $\cdots$, Parm_p). The labels of the parameters are descriptive and correspond to the "Parameter Mapping" table that is produced when the OUTPOST= option is in effect.

You can specify the following *outpost-options* in parentheses:

LOGPRIOR

adds the value of the log prior distribution to the data set.

NONSINGULAR | NONSING | COMPRESS

eliminates parameters that correspond to singular columns in the design matrix (and were not sampled) from the posterior data set. This is the default.

SINGULAR | SING

adds columns of zeros to the data set in positions that correspond to singularities in the model or to parameters that were not sampled for other reasons. By default, these columns of zeros are not written to the posterior data set.

PHIPRIORPARMS=*pair-specification*

PHIPRIORPARMS(*pair-specification* **...** *pair-specification***)**

specifies the parameters for the inverse gamma prior distribution of the scale parameters (ϕ's) in the model. The *pair-specification* is of the form (a, b), and the values are chosen such that the prior distribution has mean $\mu = b/(a-1)$ and variance $\mu^2/(a-2)$.

The form of the PHIPRIORPARMS with an equal sign and a single pair is used to specify one pair of prior parameters that applies to all components in the mixture. For example:

```
proc hpfmm seed=12345;
   model y = / k=2;
   bayes phipriorparms=(2.001,1.001);
run;
```

The form with a list of pairs is used to specify different prior parameters for the scale parameters in different components. For example:

```
proc hpfmm seed=12345;
   model y = / k=2;
   bayes phipriorparms( (.,1.001)  (3.001,2.001) );
run;
```

If you specify an invalid value (outside of the parameter space for the prior distribution), the HPFMM procedure chooses the default value and writes a message to the SAS log. If you want to use the default values for a particular parameter, you can also specify missing values in the *pair-specification*. For example, the preceding list specification assigns default values for the first component a prior parameter and uses the value 1.001 for the b prior parameter. The second pair assigns 3.001 and 2.001 for the a and b prior parameters, respectively.

PRIOROPTIONS < = >(*prior-options***)**

PRIOROPTS < = >(*prior-options***)**

specifies options related to the construction of the prior distribution and the choice of their parameters. Some *prior-options* apply only in particular models. The BETAPRIORPARMS= and MUPRIORPARMS= options override this option.

You can specify the following *prior-options*:

CONDITIONAL | COND

chooses a conditional prior specification for the homogeneous normal and t distribution response components. The default prior specification in these models is an independence prior where the mean of the hth component has prior $\mu_h \sim N(a, b)$. The conditional prior is characterized by $\mu_h \sim N(a, \sigma_h^2/b)$.

DEPENDENT | DEP

chooses a data-dependent prior for the homogeneous models without effects. The prior parameters a and b are chosen as follows, based on the distribution in the MODEL statement:

Binary and binomial: $a = \bar{y}/(1-\bar{y})$, b=1, and the prior distribution for the success probability is $\mathrm{beta}(a, b)$.

Poisson: $a = 1$, $b = 1/\bar{y}$, and the prior distribution for μ is $\mathrm{gamma}(a, b)$. See Frühwirth-Schnatter (2006, p. 280) and Viallefont, Richardson, and Greene (2002).

Exponential: $a = 3$, $b = 2\bar{y}$, and the prior distribution for μ is inverse gamma with parameters a and b.

Normal and *t*: Under the default independence prior, the prior distribution for μ is $N(\bar{y}, fs^2)$ where f is the variance factor from the VAR= option and

$$s^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2$$

Under the default conditional prior specification, the prior for μ_h is $N(a, \sigma_h^2/b)$ where $a = \bar{y}$ and $b = 2.6/(\max\{y\} - \min\{y\})$. The prior for the scale parameter is inverse gamma with parameters 1.28 and $0.36s^2$. For further details, see Raftery (1996) and Frühwirth-Schnatter (2006, p. 179).

VAR=*f*

specifies the variance for normal prior distributions. The default is VAR=1000. This factor is used, for example, in determining the prior variance of regression coefficients or in determining the prior variance of means in homogeneous mixtures of *t* or normal distributions (unless a data-dependent prior is used).

MLE< =*r* **>**

specifies that the prior distribution for regression variables be based on a multivariate normal distribution centered at the MLEs and whose dispersion is a multiple *r* of the asymptotic MLE covariance matrix. The default is MLE=10. In other words, if you specify PRIOROPTIONS(MLE), the HPFMM procedure chooses the prior distribution for the regression variables as $N(\widehat{\beta}, 10\mathrm{Var}[\widehat{\beta}])$ where $\widehat{\beta}$ is the vector of maximum likelihood estimates. The prior for the scale parameter is inverse gamma with parameters 1.28 and $0.36s^2$ where

$$s^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2$$

For further details, see Raftery (1996) and Frühwirth-Schnatter (2006, p. 179). If you specify PRIOROPTIONS(MLE) for the regression parameters, then the data-dependent prior is used for the scale parameter; see the PRIOROPTIONS(DEPENDENT) option above.

The MLE option is not available for mixture models in which the parameters are estimated directly on the data scale, such as homogeneous mixture models or mixtures of distributions without model effects for which a conjugate sampler is available. By using the METROPOLIS option, you can always force the HPFMM procedure to abandon a conjugate sampler in favor of a Metropolis-Hastings sampling algorithm to which the MLE option applies.

STATISTICS < (*global-options*) > = ALL | NONE | *keyword* | (*keyword-list*)

SUMMARIES < (*global-options*) > = ALL | NONE | *keyword* | (*keyword-list*)

controls the number of posterior statistics produced. Specifying STATISTICS=ALL is equivalent to specifying STATISTICS=(SUMMARY INTERVAL). To suppress the computation of posterior statistics, specify STATISTICS=NONE. The default is STATISTICS=(SUMMARY INTERVAL). See the section "Summary Statistics" (Chapter 7, *SAS/STAT User's Guide*) in Chapter 7, "Introduction to Bayesian Analysis Procedures" (*SAS/STAT User's Guide*), for more details.

The *global-options* include the following:

ALPHA=*numeric-list*
controls the coverage levels of the equal-tail credible intervals and the credible intervals of highest posterior density (HPD) credible intervals. The ALPHA= values must be between 0 and 1. Each ALPHA= value produces a pair of $100(1-\alpha)\%$ equal-tail and HPD credible intervals for each sampled parameter. The default is ALPHA=0.05, which results in 95% credible intervals for the parameters.

PERCENT=*numeric-list*
requests the percentile points of the posterior samples. The values in *numeric-list* must be between 0 and 100. The default is PERCENT=(25 50 75), which yields for each parameter the 25th, 50th, and 75th percentiles, respectively.

The list of *keywords* includes the following:

SUMMARY
produces the means, standard deviations, and percentile points for the posterior samples. The default is to produce the 25th, 50th, and 75th percentiles; you can modify this list with the global PERCENT= option.

INTERVAL
produces equal-tail and HPD credible intervals. The default is to produce the 95% equal-tail credible intervals and 95% HPD credible intervals, but you can use the ALPHA= *global-option* to request credible intervals for any probabilities.

THIN=*n*

THINNING=*n*
controls the thinning of the Markov chain after the burn-in. Only one in every k samples is used when THIN=k, and if NBI=n_0 and NMC=n, the number of samples kept is

$$\left[\frac{n_0+n}{k}\right]-\left[\frac{n_0}{k}\right]$$

where $[a]$ represents the integer part of the number a. The default is THIN=1—that is, all samples are kept after the burn-in phase.

TIMEINC=*n*
specifies a time interval in seconds to report progress during the burn-in and sampling phase. The time interval is approximate, since the minimum time interval in which the HPFMM procedure can respond depends on the multithreading configuration.

BY Statement

BY *variables* **;**

You can specify a BY statement with PROC HPFMM to obtain separate analyses of observations in groups that are 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 you specify more than one BY statement, only the last one specified is used.

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 HPFMM 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 (in Base SAS software).

BY statement processing is not supported when the HPFMM procedure runs alongside the database or alongside the Hadoop Distributed File System (HDFS). These modes are used if the input data are stored in a database or HDFS and the grid host is the appliance that houses the data.

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

CLASS Statement

CLASS *variable* < *(options)* >. . . < *variable* < *(options)* > > < / *global-options* > **;**

The CLASS statement names the classification variables to be used as explanatory variables in the analysis. The CLASS statement must precede the MODEL statement.

The CLASS statement for SAS high-performance analytical procedures is documented in the section "CLASS Statement" on page 42 of Chapter 4, "Shared Statistical Concepts." The HPFMM procedure also supports the following *global-option* in the CLASS statement:

UPCASE

uppercases the values of character-valued CLASS variables before levelizing them. For example, if the UPCASE option is in effect and a CLASS variable can take the values 'a,' 'A,' and 'b,' then 'a' and 'A' represent the same level and the CLASS variable is treated as having only two values: 'A' and 'B.'

FREQ Statement

FREQ *variable* **;**

The *variable* in the FREQ statement identifies a numeric variable in the data set that contains the frequency of occurrence for each observation. SAS high-performance analytical procedures that support the FREQ statement treat each observation as if it appeared f times, where f is the value of the FREQ variable for the observation. If the frequency value is not an integer, it is truncated to an integer. If the frequency value is less than 1 or missing, the observation is not used in the analysis. When the FREQ statement is not specified, each observation is assigned a frequency of 1.

ID Statement

ID *variables* **;**

The ID statement lists one or more variables from the input data set that are transferred to output data sets created by SAS high-performance analytical procedures, provided that the output data set produces one or more records per input observation.

For more information about the common ID statement in SAS high-performance analytical procedures, see the section "ID Statement" on page 46 in Chapter 4, "Shared Statistical Concepts."

MODEL Statement

MODEL *response* < **(***response-options***)** > **=** < *effects* > < / *model-options* > **;**

MODEL *events/trials* **=** < *effects* > < / *model-options* > **;**

MODEL **+** < *effects* > < / *model-options* > **;**

The MODEL statement defines elements of the mixture model, such as the model effects, the distribution, and the link function. At least one MODEL statement is required. You can specify more than one MODEL statement. Each MODEL statement identifies one or more components of a mixture. For example, if components differ in their distributions, link functions, or regressor variables, then you can use separate MODEL statements to define the components. If the finite mixture model is homogeneous—in the sense that all components share the same regressors, distribution, and link function—then you can specify the mixture model with a single MODEL statement by using the K= option.

An intercept is included in each model by default. It can be removed with the NOINT option.

The dependent variable can be specified by using either the *response* syntax or the *events/trials* syntax. The *events/trials* syntax is specific to models for binomial-type data. A binomial(n, π) variable is the sum of n independent Bernoulli trials with event probability π. Each Bernoulli trial results in either an event or a nonevent (with probability $1 - \pi$). The value of the second variable, *trials*, gives the number n of Bernoulli trials. The value of the first variable, *events*, is the number of events out of n. The values of both *events* and (*trials*–*events*) must be nonnegative, and the value of *trials* must be positive. Other distributions that allow the *events/trials* syntax are the beta-binomial distribution and the binomial cluster model.

If the *events/trials* syntax is used, the HPFMM procedure defaults to the binomial distribution. If you use the *response* syntax, the procedure defaults to the normal distribution unless the response variable is a character variable or listed in the CLASS statement.

The HPFMM procedure supports a continuation-style syntax in MODEL statements. Since a mixture has only one response variable, it is sufficient to specify the response variable in one MODEL statement. Other MODEL statements can use the continuation symbol "+" before the specification of effects. For example, the following statements fit a three-component binomial mixture model:

```
class A;
model y/n = x / k=2;
model     + A;
```

The first MODEL statement uses the "=" sign to separate response from effect information and specifies the response variable by using the *events/trials* syntax. This determines the distribution as binomial. This MODEL statement adds two components to the mixture models with different intercepts and regression slopes. The second MODEL statement adds another component to the mixture where the mean is a function of the classification main effect for variable A. The response is also binomial; it is a continuation from the previous MODEL statement.

There are two sets of options in the MODEL statement. The *response-options* determine how the HPFMM procedure models probabilities for binary data. The *model-options* control other aspects of model formation and inference. Table 6.4 summarizes the *response-options* and *model-options* available in the MODEL statement. These are subsequently discussed in detail in alphabetical order by option category.

Table 6.4 Summary of MODEL Statement Options

Option	Description
Response Variable Options	
DESCENDING	Reverses the order of response categories
EVENT=	Specifies the event category in binary models
ORDER=	Specifies the sort order for the response variable
REFERENCE=	Specifies the reference category in categorical models
Model Building	
DIST=	Specifies the response distribution
LINK=	Specifies the link function
K=	Specifies the number of mixture components
KMAX=	Specifies the maximum number of mixture components
KMIN=	Specifies the minimum number of mixture components
KRESTART	Requests that the starting values for each analysis be determined separately instead of sequentially
NOINT	Excludes fixed-effect intercept from model
OFFSET=	Specifies the offset variable for linear predictor
Statistical Computations and Output	
ALPHA=α	Determines the confidence level ($1 - \alpha$)
CL	Displays confidence limits for fixed-effects parameter estimates
EQUATE=	Imposes simple equality constraints on parameters in this model

Table 6.4 *continued*

Option	Description
LABEL=	Identifies the model
PARMS	Provides starting values for the parameters in this model

Response Variable Options

Response variable options determine how the HPFMM procedure models probabilities for binary data.

You can specify the following *response-options* by enclosing them in parentheses after the *response* variable. The default is ORDER=FORMATTED.

DESCENDING

DESC

reverses the order of the response categories. If both the DESCENDING and ORDER= options are specified, PROC HPFMM orders the response categories according to the ORDER= option and then reverses that order.

EVENT='*category*' | *keyword*

specifies the event category for the binary response model. PROC HPFMM models the probability of the event category. You can specify the value (formatted, if a format is applied) of the event category in quotes, or you can specify one of the following *keywords*:

FIRST

designates the first ordered category as the event. This is the default.

LAST

designates the last ordered category as the event.

ORDER=*order-type*

specifies the sort order for the levels of the response variable. You can specify the following values for *order-type*:

DATA

sorts the levels by order of appearance in the input data set.

FORMATTED

sorts the levels by external formatted value, except for numeric variables with no explicit format, which are sorted by their unformatted (internal) value.

FREQ

sorts the levels by descending frequency count; levels with the most observations come first in the order.

INTERNAL

sorts the levels by unformatted value.

FREQDATA
sorts the levels by order of descending frequency count, and within counts by order of appearance in the input data set when counts are tied.

FREQFORMATTED
sorts the levels by order of descending frequency count, and within counts by formatted value (as above) when counts are tied.

FREQINTERNAL
sorts the levels by order of descending frequency count, and within counts by unformatted value when counts are tied.

When ORDER=FORMATTED (the default) for numeric variables for which you have supplied no explicit format (that is, for which there is no corresponding FORMAT statement in the current PROC HPFMM run or in the DATA step that created the data set), the levels are ordered by their internal (numeric) value. If you specify the ORDER= option in the MODEL statement and the ORDER= option in the CLASS statement, the former takes precedence.

By default, ORDER=FORMATTED. For the FORMATTED and INTERNAL values, the sort order is machine-dependent.

For more information about sort order, see the chapter on the SORT procedure in the *Base SAS Procedures Guide* and the discussion of BY-group processing in *SAS Language Reference: Concepts*.

REFERENCE='*category*' | *keyword*
REF='*category*' | *keyword*
specifies the reference category for categorical models. For the binary response model, specifying one response category as the reference is the same as specifying the other response category as the event category. You can specify the value (formatted if a format is applied) of the reference category in quotes, or you can specify one of the following *keywords*:

FIRST
designates the first ordered category as the reference category.

LAST
designates the last ordered category as the reference category. This is the default.

Model Options

ALPHA=*number*
requests that confidence intervals be constructed for each of the parameters with confidence level $1 - number$. The value of *number* must be between 0 and 1; the default is 0.05.

CL
requests that confidence limits be constructed for each of the parameter estimates. The confidence level is 0.95 by default; this can be changed with the ALPHA= option.

DISTRIBUTION=*keyword*

DIST=*keyword*

specifies the probability distribution for a mixture component.

If you specify the DIST= option and you do not specify a link function with the LINK= option, a default link function is chosen according to Table 6.5. If you do not specify a distribution, the HPFMM procedure defaults to the normal distribution for continuous response variables and to the binary distribution for classification or character variables, unless the *events/trial* syntax is used in the MODEL statement. If you choose the *events/trial* syntax, the HPFMM procedure defaults to the binomial distribution.

Table 6.5 lists *keywords* that you can specify for the DISTRIBUTION= option and the corresponding default link functions. For generalized linear models with these distributions, you can find expressions for the log-likelihood functions in the section "Log-Likelihood Functions for Response Distributions" on page 175.

Table 6.5 Keyword Values of the DIST= Option

keyword	Alias	Distribution	Default Link Function
BETA		Beta	Logit
BETABINOMIAL	**BETABIN**	Beta-binomial	Logit
BINARY	**BERNOULLI**	Binary	Logit
BINOMIAL	**BIN**	Binomial	Logit
BINOMCLUSTER	**BINOMCLUS**	Binomial cluster	Logit
CONSTANT <(*c*)>	**DEGENERATE** <(*c*)>	Degenerate	N/A
EXPONENTIAL	**EXPO**	Exponential	Log
FOLDEDNORMAL	**FNORMAL**	Folded normal	Identity
GAMMA	**GAM**	Gamma	Log
GAUSSIAN	**NORMAL**	Normal	Identity
GENPOISSON	**GPOISSON**	Generalized Poisson	Log
GEOMETRIC	**GEOM**	Geometric	Log
INVGAUSS	**IGAUSSIAN**, **IG**	Inverse Gaussian	Inverse squared (power(–2))
LOGNORMAL	**LOGN**	Lognormal	Identity
NEGBINOMIAL	**NEGBIN**, **NB**	Negative binomial	Log
POISSON	**POI**	Poisson	Log
T <(ν)>	**STUDENT** <(ν)>	*t*	Identity
TRUNCEXPO <(*a*,*b*)>	**TEXPO** <(*a*,*b*)>	Truncated exponential	Log
TRUNCLOGN <(*a*,*b*)>	**TLOGN** <(*a*,*b*)>	Lognormal	Identity
TRUNCNEGBIN	**TNEGBIN**, **TNB**	Negative binomial	Log
TRUNCNORMAL <(*a*,*b*)>	**TNORMAL** <(*a*,*b*)>	Truncated normal	Identity
TRUNCPOISSON	**TPOISSON**, **TPOI**	Truncated Poisson	Log
UNIFORM <(*a*,*b*)>	**UNIF** <(*a*,*b*)>	Uniform	N/A
WEIBULL		Weibull	Log

Note that the PROC HPFMM default link for the gamma or exponential distribution is not the canonical link (the reciprocal link).

The binomial cluster model is a two-component model described in Morel and Nagaraj (1993); Morel and Neerchal (1997); Neerchal and Morel (1998). See Example 6.1 for an application of the binomial cluster model in a teratological experiment.

If the *events/trials* syntax is used, the default distribution is the binomial and only the following choices are available: DIST=BINOMIAL, DIST=BETABINOMIAL, and DIST=BINOMCLUSTER. The *trials* variable is ignored for all other distributions. This enables you to fit models in which some components have a binomial or binomial-like distribution. For example, suppose that variable n is a binomial denominator and variable logn is its logarithm. Then the following statements model a two-component mixture of a binomial and Poisson count model:

```
model y/n  = ;
model      +  / dist=Poisson offset=logn;
```

The OFFSET= option is used in the second MODEL statement to specify that the Poisson counts refer to different base counts, since the trial variable n is ignored in the second model.

If DIST=BINOMIAL is specified without the *events/trials* syntax, then n=1 is used for the default number of trials.

DIST=TRUNCNEGBIN and DIST=TRUNCPOISSON are zero-truncated versions of DIST=NEGBINOMIAL and DIST=POISSON, respectively—that is, only the value of 0 is excluded from the support.

For DIST=TRUNCEXPO, DIST=TRUNCLOGN, and DIST=TRUNCNORMAL, you must specify the lower (*a*) and upper (*b*) truncation points of the distribution. For example:

DIST=TRUNCEXPO<(*a*,*b*)>

DIST=TRUNCLOGN<(*a*,*b*)>

DIST=TRUNCNORMAL<(*a*,*b*)>

Each of these distributions is the conditional version of its corresponding nontruncated distribution that is confined to the support $[a, b]$ (inclusive). You can specify a missing value (.) for either *a* or *b* to truncate only on the other side; that is, *a*=. indicates a right-truncated distribution, and *b*=. indicates a left-truncated distribution.

For several distribution specifications you can provide additional optional parameters to further define the distribution. These optional parameters are listed in the following:

CONSTANT<(*c*)> The number *c* specifies the value where the mass is concentrated. The default is DIST=CONSTANT(0), so you can add zero-inflation to any model by adding a MODEL statement with DIST=CONSTANT.

T<(ν)> The number ν specifies the degrees of freedom for the (shifted) t distribution. The default is DIST=T(3); this leads to a heavy-tailed distribution for which the variance is defined. See the section "Log-Likelihood Functions for Response Distributions" on page 175 for the density function of the shifted t_ν distribution.

UNIFORM<(*a*,*b*)> The values *a* and *b* define the support of the uniform distribution, $a < b$. By default, $a = 0$ and $b = 1$.

EQUATE=MEAN | SCALE | NONE | EFFECTS(*effect-list*)

specifies simple sets of parameter constraints across the components in a MODEL statement; the default is EQUATE=NONE. This option is available only for maximum likelihood estimation. If you specify EQUATE=MEAN, the parameters that determine the mean are reduced to a single set that is applicable to all components in the MODEL statement. If you specify EQUATE=SCALE, a single parameter represents the common scale for all components in the MODEL statement. The EFFECTS option enables you to force the parameters for the chosen model effects to be equal across components; however, the number of parameters is unaffected.

For example, the following statements fit a two-component multiple regression model in which the coefficients for variable logd vary by component and the intercepts and coefficients for variable dose are the same for the two components:

```
proc hpfmm;
   model num = dose logd / equate=effects(int dose) k=2;
run;
```

To fix all coefficients across the two components, you can write the MODEL statement as

```
model num = dose logd / equate=effects(int dose logd) k=2;
```

or

```
model num = dose logd / equate=mean k=2;
```

If you restrict all parameters in a *k*-component MODEL statement to be equal, the HPFMM procedure reduces the model to *k*=1.

K=*n*

NUMBER=*n*

specifies the number of components the MODEL statement contributes to the overall mixture. For the binomial cluster model, this option is not available, since this model is a two-component model by definition.

KMAX=*n*

specifies the maximum number of components the MODEL statement contributes to the overall mixture.

If the maximum number of components in the mixture, as determined by all KMAX= options, is larger than the minimum number of components, the HPFMM procedure fits all possible models and displays summary fit information for the sequence of evaluated models. The "best" model according to the CRITERION= option in the PROC HPFMM statement is then chosen, and the remaining output and analyses performed by PROC HPFMM pertain to this "best" model.

When you use MCMC methods to estimate the parameters of a mixture, you need to ensure that the chain for a given value of *k* has converged; otherwise, comparisons among models that have varying numbers of components might not be meaningful. You can use the FITDETAILS option to display summary and diagnostic information for the MCMC chains from each model.

If you specify the KMIN= option but not the KMAX= option, then the default value for the KMAX= option is the value of the KMIN= option (unless KMIN=0, in which case the KMAX= option is set to 1).

KMIN=*n*

specifies the minimum number of components that the MODEL statement contributes to the overall mixture. When you use MCMC methods to estimate the parameters of a mixture, you need to ensure that the chain for a given value of k has converged; otherwise, comparisons among models that have varying numbers of components might not be meaningful.

KRESTART

requests that the starting values for each analysis (that is, for each unique number of components as determined by the KMIN= and KMAX= options) be determined separately, in the same way as if no other analyses were performed. If you do not specify the KRESTART option, then the starting values for each analysis are based on results from the previous analysis with one less component.

LABEL='*label*'

specifies an optional label for the model that is used to identify the model in printed output, on graphics, and in data sets created from ODS tables.

LINK=*keyword*

specifies the link function in the model. The *keywords* and expressions for the associated link functions are shown in Table 6.6.

Table 6.6 Link Functions in MODEL Statement of the HPFMM Procedure

LINK=	Alias	Link Function	$g(\mu) = \eta =$
CLOGLOG	**CLL**	Complementary log-log	$\log(-\log(1-\mu))$
IDENTITY	**ID**	Identity	μ
LOG		Log	$\log(\mu)$
LOGIT		Logit	$\log(\mu/(1-\mu))$
LOGLOG		Log-log	$-\log(-\log(\mu))$
PROBIT	**NORMIT**	Probit	$\Phi^{-1}(\mu)$
POWER(λ)	**POW**(λ)	Power with exponent λ= *number*	$\begin{cases} \mu^{\lambda} & \text{if } \lambda \neq 0 \\ \log(\mu) & \text{if } \lambda = 0 \end{cases}$
POWERMINUS2		Power with exponent -2	$1/\mu^2$
RECIPROCAL	**INVERSE**	Reciprocal	$1/\mu$

The default link functions for the various distributions are shown in Table 6.5.

NOINT

requests that no intercept be included in the model. An intercept is included by default, unless the distribution is DIST=CONSTANT or DIST=UNIFORM.

OFFSET=*variable*

specifies the offset variable function for the linear predictor in the model. An offset variable can be thought of as a regressor variable whose regression coefficient is known to be 1. For example, you can use an offset in a Poisson model when counts have been obtained in time intervals of different lengths. With a log link function, you can model the counts as Poisson variables with the logarithm of the time interval as the offset variable.

PARAMETERS(*parameter-specification***)**
PARMS(*parameter-specification***)**

specifies starting values for the model parameters. If no PARMS option is given, the HPFMM procedure determines starting values by a data-dependent algorithm. To determine initial values for the Markov chain with Bayes estimation, see also the INITIAL= option in the BAYES statement. The specification of the parameters takes the following form: parameters in the mean function precede the scale parameters, and parameters for different components are separated by commas.

The following statements specify starting parameters for a two-component normal model. The initial values for the intercepts are 1 and –3; the initial values for the variances are 0.5 and 4.

```
proc hpfmm;
   model y = / k=2 parms(1 0.5, -3 4);
run;
```

You can specify missing values for parameters whose starting values are to be determined by the default method. Only values for parameters that participate in the optimization are specified. The values for model effects are specified on the linear (linked) scale.

OUTPUT Statement

OUTPUT < **OUT=***SAS-data-set* >
< *keyword*< **(***keyword-options***)** > < **=***name* > > . . .
< *keyword*< **(***keyword-options***)** > < **=***name* > > < */ options* > **;**

The OUTPUT statement creates a data set that contains observationwise statistics that are computed after fitting the model. The variables in the input data set are *not* included in the output data set to avoid data duplication for large data sets; however, variables specified in the ID statement are included.

The output statistics are computed based on the parameter estimates of the converged model if the parameters are estimated by maximum likelihood. If a Bayesian analysis is performed, the output statistics are computed based on the arithmetic mean in the posterior sample. You can change to the maximum posterior estimate with the ESTIMATE=MAP option in the BAYES statement.

You can specify the following syntax elements in the OUTPUT statement before the slash (/).

OUT=*SAS-data-set*

specifies the name of the output data set. If the OUT= option is omitted, the procedure uses the DATA*n* convention to name the output data set.

*keyword***<** **(***keyword-options***)** **>** **<** **=***name* **>**

specifies a statistic to include in the output data set and optionally assigns the variable the name *name*. If you do not provide a name, the HPFMM procedure assigns a default name based on the type of

statistic requested. If you provide a name for a statistic that leads to multiple output statistics, the name is modified to index the associated component number.

You can use the *keyword-options* to control which type of a particular statistic is computed. The following are valid values for *keyword* and *keyword-options*:

PREDICTED< (COMPONENT | OVERALL) >

PRED< (COMPONENT | OVERALL) >

MEAN< (COMPONENT | OVERALL) >

requests predicted values (predicted means) for the response variable. The predictions in the output data set are mapped onto the data scale in all cases except for a binomial or binary response with *events/trials* syntax and when PREDTYPE=COUNT has not been specified. In that case the predictions are predicted success probabilities.

The default is to compute the predicted value for the mixture (OVERALL). You can request predictions for the means of the component distributions by adding the COMPONENT suboption in parentheses. The predicted values for some distributions are not identical to the parameter modeled as μ. For example, in the lognormal distribution the predicted mean is $\exp\{\mu + 0.5\phi\}$ where μ and ϕ are the parameters of an underlying normal process; see the section "Log-Likelihood Functions for Response Distributions" on page 175 for details.

RESIDUAL< (COMPONENT | OVERALL) >

RESID< (COMPONENT | OVERALL) >

requests residuals for the response or residuals in the component distributions. Only "raw" residuals on the data scale are computed (observed minus predicted).

VARIANCE< (COMPONENT | OVERALL) >

VAR< (COMPONENT | OVERALL) >

requests variances for the mixture or the component distributions.

LOGLIKE< (COMPONENT | OVERALL) >

LOGL< (COMPONENT | OVERALL) >

requests values of the log-likelihood function for the mixture or the components. For observations used in the analysis, the overall computed value is the observations' contribution to the log likelihood; if a FREQ statement is present, the frequency is accounted for in the computed value. In other words, if all observations in the input data set have been used in the analysis, adding the value of the log-likelihood contributions in the OUTPUT data set produces the negative of the final objective function value in the "Iteration History" table. By default, the log-likelihood contribution to the mixture is computed. You can request the individual mixture component contributions with the COMPONENT suboption.

MIXPROBS< (COMPONENT | MAX) >

MIXPROB< (COMPONENT | MAX) >

PRIOR< (COMPONENT | MAX) >

MIXWEIGHTS< (COMPONENT | MAX) >

requests that the prior weights $\pi_j(\mathbf{z}, \boldsymbol{\alpha}_j)$ be added to the OUTPUT data set. By default, the probabilities are output for all components. You can limit the output to a single statistic, the largest mixing probability, with the MAX suboption.

NOTE: The keyword "prior" is used here because of long-standing practice to refer to the mixing probabilities as prior weights. This must not be confused with the prior distribution and its parameters in a Bayesian analysis.

POSTERIOR< (COMPONENT | MAX) >

POST< (COMPONENT | MAX) >

PROB< (COMPONENT | MAX) >

requests that the posterior weights

$$\frac{\pi_j(\mathbf{z}, \boldsymbol{\alpha}_j)\, p_j(y; \mathbf{x}'_j \boldsymbol{\beta}_j, \phi_j)}{\sum_{j=1}^{k} \pi_j(\mathbf{z}, \boldsymbol{\alpha}_j)\, p_j(y; \mathbf{x}'_j \boldsymbol{\beta}_j, \phi_j)}$$

be added to the OUTPUT data set. By default, the probabilities are output for all components. You can limit the output to a single statistic, the largest posterior probability, with the MAX suboption.

NOTE: The keyword "posterior" is used here because of long-standing practice to refer to these probabilities as posterior probabilities. This must not be confused with the posterior distribution in a Bayesian analysis.

LINP

XBETA

requests that the linear predictors for the models be added to the OUTPUT data set.

CLASS | CATEGORY | GROUP

adds the estimated component membership to the OUTPUT data set. An observation is associated with the component that has the highest posterior probability.

MAXPOST | MAXPROB

adds the highest posterior probability to the OUTPUT data set.

A *keyword* can appear multiple times. For example, the following OUTPUT statement requests predicted values for the mixture in addition to the predicted means in the individual components:

```
output out=hpfmmout pred=MixtureMean pred(component)=CompMean;
```

In a three-component model, this produces four variables in the hpfmmout data set: MixtureMean, CompMean_1, CompMean_2, and CompMean_3.

You can specify the following *options* in the OUTPUT statement after a slash (/).

ALLSTATS

requests that all statistics are computed. If you do not use a *keyword* to assign a name, the HPFMM procedure uses the default name.

PREDTYPE=PROB | COUNT

specifies the type of predicted values that are produced for a binomial or binary response with *events/trials* syntax. If PREDTYPE=PROB, the predicted values are success probabilities. If PREDTYPE=COUNT, the predicted values are success counts. The default is PREDTYPE=PROB.

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 the HPFMM procedure.

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

The PERFORMANCE statement is documented further in the section "PERFORMANCE Statement" on page 36.

PROBMODEL Statement

PROBMODEL < *effects* > < / *probmodel-options* > **;**

The PROBMODEL statement defines the model effects for the mixing probabilities and their link function and starting values. Model effects (other than the implied intercept) are not supported with Bayesian estimation. By default, the HPFMM procedure models mixing probabilities on the logit scale for two-component models and as generalized logit models in situations with more than two components. The PROBMODEL statement is not required.

The generalized logit model with k categories has a common vector of regressor or design variables, $\mathbf{z}$, $k-1$ parameter vectors that vary with category, and one linear predictor whose value is constant. The constant linear predictor is assigned by the HPFMM procedure to the last component in the model, and its value is zero ($\boldsymbol{\alpha}_k = \mathbf{0}$). The probability of observing category $1 \leq j \leq k$ is then

$$\pi_j(\mathbf{z}, \boldsymbol{\alpha}_j) = \frac{\exp\{\mathbf{z}'\boldsymbol{\alpha}_j\}}{\sum_{i=1}^{k} \exp\{\mathbf{z}'\boldsymbol{\alpha}_i\}}$$

For k=2, the generalized logit model reduces to a model with the logit link (a logistic model); hence the attribute *generalized* logit.

By default, an intercept is included in the model for the mixing probabilities. If you suppress the intercept with the NOINT option, you must specify at least one effect in the statement.

You can specify the following *probmodel-options* in the PROBMODEL statement after the slash (/):

ALPHA=*number*
: requests that confidence intervals that have the confidence level $1 - $ *number* be constructed for the parameters in the probability model. The value of *number* must be between 0 and 1; the default is 0.05. If the probability model is simple—that is, it does not contain any effects—the confidence intervals are produced for the estimated parameters (on the logit scale) and for the mixing probabilities. This option has no effect when you perform Bayesian estimation. You can modify credible interval settings by specifying the STATISTICS(ALPHA=) option in the BAYES statement.

CL

requests that confidence limits be constructed for each of the parameter estimates. The confidence level is 0.95 by default; this can be changed with the ALPHA= option.

LINK=*keyword*

specifies the link function in the model for the mixing probabilities. The default is a logit link for models with two components. For models with more than two components, only the generalized logit link is available. The *keywords* and expressions for the associated link functions for two-component models are shown in Table 6.7.

Table 6.7 Link Functions in the PROBMODEL Statement

LINK=	Link Function	$g(\mu) = \eta =$
CLOGLOG | CLL	Complementary log-log	$\log(-\log(1-\mu))$
LOGIT	Logit	$\log(\mu/(1-\mu))$
LOGLOG	Log-log	$-\log(-\log(\mu))$
PROBIT | NORMIT	Probit	$\Phi^{-1}(\mu)$

NOINT

requests that no intercept be included in the model for the mixing probabilities. An intercept is included by default. If you suppress the intercept with the NOINT option, you must specify at least one other effect for the mixing probabilities—since an empty probability model is not meaningful.

PARAMETERS(*parameter-specification***)**

PARMS(*parameter-specification***)**

specifies starting values for the parameters. The specification of the parameters takes the following form: parameters in the mean function appear in a list, and parameters for different components are separated by commas. Starting values are given on the linked scale, not in terms of probabilities. Also, you need to specify starting values for each of the first *k–1* components in a *k*-component model. The linear predictor for the last component is always assumed to be zero.

The following statements specify a three-component mixture of multiple regression models. The PROBMODEL statement does not list any effects, a standard "intercept-only" generalized logit model is used to model the mixing probabilities.

```
proc hpfmm;
   model y = x1 x2 / k=3;
   probmodel  / parms(2, 1);
run;
```

There are three linear predictors in the model for the mixing probabilities, α_1, α_2, and α_3. With starting values of $\alpha_1 = 2$, $\alpha_2 = 1$, and $\alpha_3 = 0$, this leads to initial mixing probabilities of

$$\pi_1 = \frac{e^2}{e^2 + e^1 + e^0} = 0.24$$

$$\pi_2 = \frac{e^1}{e^2 + e^1 + e^0} = 0.66$$

$$\pi_3 = \frac{e^0}{e^2 + e^1 + e^0} = 0.1$$

You can specify missing values for parameters whose starting values are to be determined by the default method.

RESTRICT Statement

RESTRICT < *'label'* > *constraint-specification* < **,** . . . **,** *constraint-specification* >
< *operator* < *value* > > < / *option* > **;**

The RESTRICT statement enables you to specify linear equality or inequality constraints among the parameters of a mixture model. These restrictions are incorporated into the maximum likelihood analysis. The RESTRICT statement is not available for a Bayesian analysis with the HPFMM procedure.

Following are reasons why you might want to place constraints and restrictions on the model parameters:

- to fix a parameter at a particular value
- to equate parameters in different components in a mixture
- to impose order conditions on the parameters in a model
- to specify contrasts among the parameters that the fitted model should honor

A restriction is composed of a left-hand side and a right-hand side, separated by an operator. If the operator and right-hand side are not specified, the restriction is assumed to be an equality constraint against zero. If the right-hand side is not specified, the value is assumed to be zero.

An individual *constraint-specification* is written in (nearly) the same form as estimable linear functions are specified in the ESTIMATE statement of the GLM, MIXED, or GLIMMIX procedure. The *constraint-specification* takes the form

model-effect value-list < . . . *model-effect value-list* > < **(SCALE=** *value***)** >

At least one *model-effect* must be specified followed by one or more values in the *value-list*. The values in the list correspond to the multipliers of the corresponding parameter that is associated with the position in the model effect. If you specify more values in the *value-list* than the *model-effect* occupies in the model design matrix, the extra coefficients are ignored.

To specify restrictions for effects in specific components in the model, separate the *constraint-specification* by commas. The following statements provide an example:

```
proc hpfmm;
   class A;
   model y/n = A x /  k = 2;
   restrict A 1 0 -1;
   restrict x 2, x -1 >= 0.5;
run;
```

The linear predictors for this two-component model can be written as

$$\eta_1 = \beta_{10} + \alpha_{11} A_1 + \cdots + \alpha_{1a} A_a + x\beta_{11}$$
$$\eta_2 = \beta_{20} + \alpha_{21} A_1 + \cdots + \alpha_{2a} A_a + x\beta_{21}$$

where A_k is the binary variable associated with the kth level of A.

The first RESTRICT statement applies only to the first component and specifies that the parameter estimates that are associated with the first and third level of the A effect are identical. In terms of the linear predictor, the restriction can be written as

$$\alpha_{11} - \alpha_{13} = 0$$

Now suppose that A has only two levels. Then the HPFMM procedure ignores the value –1 in the first RESTRICT statement and imposes the restriction

$$\alpha_{11} = 0$$

on the fitted model.

The second RESTRICT statement involves parameters in two different components of the model. In terms of the linear predictors, the restriction can be written as

$$2\beta_{11} - \beta_{21} \geq \frac{1}{2}$$

When restrictions are specified explicitly through the RESTRICT statement or implied through the EQUATE=EFFECTS option in the MODEL statement, the HPFMM procedure lists all restrictions after the model fit in a table of linear constraints and indicates whether a particular constraint is active at the converged solution.

The following operators can be specified to separate the left- and right-hand sides of the restriction: **=**, **>**, **<**, **>=**, **<=**.

Some distributions involve scale parameters (the parameter ϕ in the expressions of the log likelihood) and you can also use the *constraint-specification* to involve a component's scale parameter in a constraint. To this end, assign a value to the keyword SCALE, separated from the model effects and value lists with parentheses. The following statements fit a two-component normal model and restrict the component variances to be equal:

```
proc hpfmm;
   model y = / k=2;
   restrict int 0 (scale  1),
            int 0 (scale -1);
run;
```

The intercept specification is necessary because each *constraint-specification* requires at least one model effect. The zero coefficient ensures that the intercepts are not involved in the restriction. Instead, the RESTRICT statement leads to $\phi_1 - \phi_2 = 0$.

You can specify the following *option* in the RESTRICT statement after a slash (/).

DIVISOR=*value*
specifies a *value* by which all coefficients on the right-hand side and left-hand side of the restriction are divided.

WEIGHT Statement

WEIGHT *variable* **;**

The WEIGHT statement is used to perform a weighted analysis. Consult the section "Log-Likelihood Functions for Response Distributions" on page 175 for expressions on how weight variables are included in the log-likelihood functions. Because the probability structure of a mixture model is different from that of a classical statistical model, the presence of a weight variable in a mixture model *cannot* be interpreted as altering the variance of an observation.

Observations with nonpositive or missing weights are not included in the PROC HPFMM analysis. If a WEIGHT statement is not included, all observations used in the analysis are assigned a weight of 1.

Details: HPFMM Procedure

A Gentle Introduction to Finite Mixture Models

The Form of the Finite Mixture Model

Suppose that you observe realizations of a random variable Y, the distribution of which depends on an unobservable random variable S that has a discrete distribution. S can occupy one of k states, the number of which might be unknown but is at least known to be finite. Since S is not observable, it is frequently referred to as a latent variable.

Let π_j denote the probability that S takes on state j. Conditional on $S = j$, the distribution of the response Y is assumed to be $f_j(y; \alpha_j, \boldsymbol{\beta}_j | S = j)$. In other words, each distinct state j of the random variable S leads to a particular distributional form f_j and set of parameters $\{\alpha_j, \boldsymbol{\beta}_j\}$ for Y.

Let $\{\boldsymbol{\alpha}, \boldsymbol{\beta}\}$ denote the collection of α_j and $\boldsymbol{\beta}_j$ parameters across all $j = 1$ to k. The marginal distribution of Y is obtained by summing the joint distribution of Y and S over the states in the support of S:

$$
\begin{aligned}
f(y; \boldsymbol{\alpha}, \boldsymbol{\beta}) &= \sum_{j=1}^{k} \Pr(S = j)\, f(y; \alpha_j, \boldsymbol{\beta}_j | S = j) \\
&= \sum_{j=1}^{k} \pi_j f(y; \alpha_j, \boldsymbol{\beta}_j | S = j)
\end{aligned}
$$

This is a mixture of distributions, and the π_j are called the mixture (or prior) probabilities. Because the number of states k of the latent variable S is finite, the entire model is termed a finite mixture (of distributions) model.

The finite mixture model can be expressed in a more general form by representing $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ in terms of regressor variables and parameters with optional additional scale parameters for $\boldsymbol{\beta}$. The section "Notation for the Finite Mixture Model" on page 109 develops this in detail.

Mixture Models Contrasted with Mixing and Mixed Models: Untangling the Terminology Web

Statistical terminology can have its limitations. The terms mixture, mixing, and mixed models are sometimes used interchangeably, causing confusion. Even worse, the terms arise in related situations. One application needs to be eliminated from the discussion in this documentation: mixture experiments, where design factors are the proportions with which components contribute to a blend, are not mixture models and do not fall under the purview of the HPFMM procedure. However, the data from a mixture experiment might be analyzed with a mixture model, a mixing model, or a mixed model, besides other types of statistical models.

Suppose that you observe realizations of random variable Y and assume that Y follows some distribution $f(y;\boldsymbol{\alpha},\boldsymbol{\beta})$ that depends on parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. Furthermore, suppose that the model is found to be deficient in the sense that the variability implied by the fitted model is less than the observed variability in the data, a condition known as overdispersion (see the section "Overdispersion" on page 175). To tackle the problem the statistical model needs to be modified to allow for more variability. Clearly, one way of doing this is to introduce additional random variables into the process. Mixture, mixing, and mixed models are simply different ways of adding such random variables. The section "The Form of the Finite Mixture Model" on page 172 explains how mixture models add a discrete state variable S. The following two subsections explain how mixing and mixed models instead assume variation for a natural parameter or in the mean function.

Mixing Models

Suppose that the model is modified to allow for some random quantity U, which might be one of the parameters of the model or a quantity related to the parameters. Now there are two distributions to cope with: the conditional distribution of the response given the random effect U,

$$f(y;\boldsymbol{\alpha},\boldsymbol{\beta}|u)$$

and the marginal distribution of the data. If U is continuous, the marginal distribution is obtained by integration:

$$f(y;\boldsymbol{\alpha},\boldsymbol{\beta}) = \int f(y;\boldsymbol{\alpha},\boldsymbol{\beta}|u)\, f(u)\, du$$

Otherwise, it is obtained by summation over the support of U:

$$f(y;\boldsymbol{\alpha},\boldsymbol{\beta}) = \sum_u \Pr(U = u)\, f(y;\boldsymbol{\alpha},\boldsymbol{\beta}|u)$$

The important entity for statistical estimation is the marginal distribution $f(y;\boldsymbol{\alpha},\boldsymbol{\beta})$; the conditional distribution is often important for model description, genesis, and interpretation.

In a mixing model the marginal distribution is known and is typically of a well-known form. For example, if $Y|n$ has a binomial(n,μ) distribution and n follows a Poisson distribution, then the marginal distribution of Y is Poisson. The preceding operation is called *mixing* a binomial distribution with a Poisson distribution. Similarly, when mixing a Poisson(λ) distribution with a gamma(a,b) distribution for λ, a negative

binomial distribution results as the marginal distribution. Other important mixing models involve mixing a binomial(n, μ) random variable with a beta(a, b) distribution for the binomial success probability μ. This results in a distribution known as the beta-binomial.

The finite mixtures have in common with the mixing models the introduction of random effects into the model to vary some or all of the parameters at random.

Mixed Models

The difference between a mixing and a mixed model is that the conditional distribution is not that important in the mixing model. It matters to motivate the overdispersed reference model and to arrive at the marginal distribution. Inferences with respect to the conditional distribution, such as predicting the random variable U, are not performed in mixing models. In a mixed model the random variable U typically follows a continuous distribution—almost always a normal distribution. The random effects usually do not model the natural parameters of the distribution; instead, they are involved in linear predictors that relate to the conditional mean. For example, a linear mixed model is a model in which the response and the random effects are normally distributed, and the random effects enter the conditional mean function linearly:

$$\begin{aligned} \mathbf{Y} &= \mathbf{X}\boldsymbol{\beta} + \mathbf{ZU} + \boldsymbol{\epsilon} \\ \mathbf{U} &\sim N(\mathbf{0}, \mathbf{G}) \\ \boldsymbol{\epsilon} &\sim N(\mathbf{0}, \mathbf{R}) \\ \mathrm{Cov}[\mathbf{U}, \boldsymbol{\epsilon}] &= \mathbf{0} \end{aligned}$$

The conditional and marginal distributions are then

$$\begin{aligned} \mathbf{Y}|\mathbf{U} &\sim N(\mathbf{X}\boldsymbol{\beta} + \mathbf{ZU} + \boldsymbol{\epsilon}, \mathbf{R}) \\ \mathbf{Y} &\sim N(\mathbf{X}\boldsymbol{\beta}, \mathbf{ZGZ}' + \mathbf{R}) \end{aligned}$$

For this model, because of the linearity in the mean and the normality of the random effects, you could also refer to mixing the normal vector $\mathbf{Y}$ with the normal vector $\mathbf{U}$, since the marginal distribution is known. The linear mixed model can be fit with the MIXED procedure. When the conditional distribution is not normal and the random effects are normal, the marginal distribution does not have a closed form. In this class of mixed models, called generalized linear mixed models, model approximations and numerical integration methods are commonly used in model fitting; see for example, those models fit by the GLIMMIX and NLMIXED procedures. Chapter 6, "Introduction to Mixed Modeling Procedures" (*SAS/STAT User's Guide*), contains details about the various classes of mixed models and about the relevant SAS/STAT procedures.

The previous expression for the marginal variance in the linear mixed model, $\mathrm{var}[\mathbf{Y}] = \mathbf{ZGZ}' + \mathbf{R}$, emphasizes again that the variability in the marginal distribution of a model that contains random effects exceeds the variability in a model without the random effects ($\mathbf{R}$).

The finite mixtures have in common with the mixed models that the marginal distribution is not necessarily a well-known model, but is expressed through a formal integration over the random-effects distribution. In contrast to the mixed models, in particular those involving nonnormal distributions or nonlinear elements, this integration is rather trivial; it reduces to a weighted and finite sum of densities or mass functions.

Overdispersion

Overdispersion is the condition by which the data are more dispersed than is permissible under a reference model. Overdispersion arises only if the variability a model can capture is limited (for example, because of a functional relationship between mean and variance). For example, a model for normal data can never be overdispersed in this sense, although the reasons that lead to overdispersion also negatively affect a misspecified model for normal data. For example, omitted variables increase the residual variance estimate because variability that should have been modeled through changes in the mean is now "picked up" as error variability.

Overdispersion is important because an overdispersed model can lead to misleading inferences and conclusions. However, diagnosing and remedying overdispersion is complicated. In order to handle it appropriately, the source of overdispersion must be identified. For example, overdispersion can arise from any of the following conditions alone or in combination:

- omitted variables and model effects
- omitted random effects (a source of random variation is not being modeled or is modeled as a fixed effect)
- correlation among the observations
- incorrect distributional assumptions
- incorrectly specified mean-variance relationships
- outliers in the data

As discussed in the previous section, introducing randomness into a system increases its variability. Mixture, mixed, and mixing models have thus been popular in modeling data that appear overdispersed. Finite mixture models are particularly powerful in this regard, because even low-order mixtures of basic, symmetric distributions (such as two- or three-component mixtures of normal or t distributions) enable you to model data with multiple modes, heavy tails, and skewness. In addition, the latent variable S provides a natural way to accommodate omitted, unobservable variables into the model.

One approach to remedy overdispersion is to apply simple modifications of the variance function of the reference model. For example, with binomial-type data this approach replaces the variance of the binomial count variable $Y \sim \text{Binomial}(n, \mu)$, $\text{Var}[Y] = n \times \mu(1-\mu)$ with a scaled version, $\phi n \times \mu(1-\mu)$, where ϕ is called an overdispersion parameter, $\phi > 0$.

In addressing overdispersion problems, it is important to tackle the problem at its root. A missing scale factor on the variance function is hardly ever the root cause of overdispersion; it is only the easiest remedy.

Log-Likelihood Functions for Response Distributions

The HPFMM procedure calculates the log likelihood that corresponds to a particular response distribution according to the following formulas. The response distribution is the distribution specified (or chosen by default) through the DIST= option in the MODEL statement. The parameterizations used for log-likelihood functions of these distributions were chosen to facilitate expressions in terms of mean parameters that are modeled through an (inverse) link functions and in terms of scale parameters. These are not

necessarily the parameterizations in which parameters of prior distributions are specified in a Bayesian analysis of homogeneous mixtures. See the section "Prior Distributions" on page 184 for details about the parameterizations of prior distributions.

The HPFMM procedure includes all constant terms in the computation of densities or mass functions. In the expressions that follow, l denotes the log-likelihood function, ϕ denotes a general scale parameter, μ_i is the "mean", and w_i is a weight from the use of a WEIGHT statement.

For some distributions (for example, the Weibull distribution) μ_i is not the mean of the distribution. The parameter μ_i is the quantity that is modeled as $g^{-1}(\mathbf{x}'\boldsymbol{\beta})$, where $g^{-1}(\cdot)$ is the inverse link function and the $\mathbf{x}$ vector is constructed based on the effects in the MODEL statement. Situations in which the parameter μ does not represent the mean of the distribution are explicitly mentioned in the list that follows.

The parameter ϕ is frequently labeled as a "Scale" parameter in output from the HPFMM procedure. It is not necessarily the scale parameter of the particular distribution.

Beta(μ, ϕ)

$$\begin{aligned} l(\mu_i, \phi; y_i, w_i) &= \log\left\{\frac{\Gamma(\phi/w_i)}{\Gamma(\mu_i\phi/w_i)\Gamma((1-\mu_i)\phi/w_i)}\right\} \\ &+ (\mu_i\phi/w_i - 1)\log\{y_i\} \\ &+ ((1-\mu_i)\phi/w_i - 1)\log\{1-y_i\} \end{aligned}$$

This parameterization of the beta distribution is due to Ferrari and Cribari-Neto (2004) and has properties $\mathrm{E}[Y] = \mu$, $\mathrm{Var}[Y] = \mu(1-\mu)/(1+\phi)$, $\phi > 0$.

Beta-binomial$(n; \mu, \phi)$

$$\begin{aligned} \phi &= (1-\rho^2)/\rho^2 \\ l(\mu_i, \rho; y_i) &= \log\{\Gamma(n_i+1)\} - \log\{\Gamma(y_i+1)\} \\ &- \log\{\Gamma(n_i - y_i + 1)\} \\ &+ \log\{\Gamma(\phi)\} - \log\{\Gamma(n_i+\phi)\} + \log\{\Gamma(y_i+\phi\mu_i)\} \\ &+ \log\{\Gamma(n_i - y_i + \phi(1-\mu_i))\} - \log\{\Gamma(\phi\mu_i)\} \\ &- \log\{\Gamma(\phi(1-\mu_i))\} \\ l(\mu_i, \rho; y_i, w_i) &= w_i\, l(\mu_i, \rho; y_i) \end{aligned}$$

where y_i and n_i are the *events* and *trials* in the *events/trials* syntax and $0 < \mu < 1$. This parameterization of the beta-binomial model presents the distribution as a special case of the Dirichlet-Multinomial distribution—see, for example, Neerchal and Morel (1998). In this parameterization, $\mathrm{E}[Y] = n\mu$ and $\mathrm{Var}[Y] = n\mu(1-\mu)(1+(n-1)/(\phi+1))$, $0 \leq \rho \leq 1$. The HPFMM procedure models the parameter ϕ and labels it "Scale" on the procedure output. For other parameterizations of the beta-binomial model, see Griffiths (1973) or Williams (1975).

Binomial$(n; \mu)$

$$\begin{aligned} l(\mu_i; y_i) &= y_i \log\{\mu_i\} + (n_i - y_i)\log\{1-\mu_i\} \\ &+ \log\{\Gamma(n_i+1)\} - \log\{\Gamma(y_i+1)\} \\ &- \log\{\Gamma(n_i - y_i + 1)\} \\ l(\mu_i; y_i, w_i) &= w_i\, l(\mu_i; y_i) \end{aligned}$$

where y_i and n_i are the *events* and *trials* in the *events/trials* syntax and $0 < \mu < 1$. In this parameterization $\mathrm{E}[Y] = n\mu$, $\mathrm{Var}[Y] = n\mu(1-\mu)$.

Binomial cluster$(n; \mu, \pi)$

$$
\begin{aligned}
z &= \log\{\Gamma(n_i+1)\} - \log\{\Gamma(y_i+1)\} - \log\{\Gamma(n_i-y_i+1)\} \\
\mu_i^* &= (1-\mu_i)\pi \\
l(\mu_i, \pi; y_i) &= \log\{\pi\} + z + y_i \log\{\mu_i^* + \mu_i\} \\
&\quad + (n_i - y_i)\log\{1-\mu_i^* - \mu_i\} \\
&\quad + \log\{1-\pi\} + z + y_i \log\{\mu_i^*\} \\
&\quad + (n_i - y_i)\log\{1-\mu_i^*\} \\
l(\mu_i, \pi; y_i, w_i) &= w_i\, l(\mu_i, \pi; y_i)
\end{aligned}
$$

In this parameterization, $\mathrm{E}[Y] = n\pi$ and $\mathrm{Var}[Y] = n\pi(1-\pi)\left\{1 + \mu^2(n-1)\right\}$. The binomial cluster model is a two-component mixture of a binomial$(n, \mu^* + \mu)$ and a binomial(n, μ^*) random variable. This mixture is unusual in that it fixes the number of components and because the mixing probability π appears in the moments of the mixture components. For further details, see Morel and Nagaraj (1993); Morel and Neerchal (1997); Neerchal and Morel (1998) and Example 6.1 in this chapter. The expressions for the mean and variance in the binomial cluster model are identical to those of the beta-binomial model shown previously, with $\pi_{bc} = \mu_{bb}$, $\mu_{bc} = \rho_{bb}$.

The HPFMM procedure models the parameter μ through the MODEL statement and the parameter π through the PROBMODEL statement.

Constant(c)

$$
l(y_i) = \begin{cases} 0 & y_i = c \\ -1\mathrm{E}20 & y_i \neq c \end{cases}
$$

The extreme value when $y_i \neq c$ is chosen so that $\exp\{l(y_i)\}$ yields a likelihood of zero. You can change this value with the INVALIDLOGL= option in the PROC HPFMM statement. The constant distribution is useful for modeling overdispersion due to zero-inflation (or inflation of the process at support c).

The DIST=CONSTANT distribution is useful for modeling an inflated probability of observing a particular value (zero, by default) in data from other discrete distributions, as demonstrated in "Modeling Zero-Inflation: Is it Better to Fish Poorly or Not to Have Fished At All?" on page 116. While it is syntactically valid to mix a constant distribution with a continuous distribution, such as DIST=LOGNORMAL, such a mixture is not mathematically appropriate, because the constant log-likelihood is the log of a probability, while a continuous log-likelihood is the log of a probability density function. If you want to mix a constant distribution with a continuous distribution, you could model the constant as a very narrow continuous distribution, such as DIST=UNIFORM$(c-\delta, c+\delta)$ for a small value δ. However, using PROC HPFMM to analyze such mixtures is sensitive to numerical inaccuracy and ultimately unnecessary. Instead, the following approach is mathematically equivalent and more numerically stable:

1. Estimate the mixing probability $\mathrm{P}(Y = c)$ as the proportion of observations in the data set such that $|y_i - c| < \epsilon$.

2. Estimate the parameters of the continuous distribution from the observations for which $|y_i - c| \geq \epsilon$.

Exponential(μ)

$$l(\mu_i; y_i, w_i) = \begin{cases} -\log\{\mu_i\} - y_i/\mu_i & w_i = 1 \\ w_i \log\left\{\frac{w_i y_i}{\mu_i}\right\} - \frac{w_i y_i}{\mu_i} - \log\{y_i \Gamma(w_i)\} & w_i \neq 1 \end{cases}$$

In this parameterization, $\mathrm{E}[Y] = \mu$ and $\mathrm{Var}[Y] = \mu^2$.

Folded normal(μ, ϕ)

$$\begin{aligned} l(\mu_i, \phi; y_i, w_i) = &-\frac{1}{2}\log\{2\pi\} - \frac{1}{2}\log\{\phi/w_i\} \\ &+ \log\left\{\exp\left\{\frac{-w_i(y_i - \mu_i)^2}{2\phi}\right\} + \exp\left\{\frac{-w_i(y_i + \mu_i)^2}{2\phi}\right\}\right\} \end{aligned}$$

If X has a normal distribution with mean μ and variance ϕ, then $Y = |X|$ has a folded normal distribution and log-likelihood function $l(\mu, \phi; y, w)$ for $y \geq 0$. The folded normal distribution arises, for example, when normally distributed measurements are observed, but their signs are not observed. The mean and variance of the folded normal in terms of the underlying $\mathrm{N}(\mu, \phi)$ distribution are

$$\begin{aligned} \mathrm{E}[Y] =& \frac{1}{\sqrt{2\pi\phi}} \exp\left\{-\frac{\mu^2}{2/\phi}\right\} + \mu\left(1 - 2\Phi\left(-\mu/\sqrt{\phi}\right)\right) \\ \mathrm{Var}[Y] =& \phi + \mu^2 - \mathrm{E}[Y]^2 \end{aligned}$$

The HPFMM procedure models the folded normal distribution through the mean μ and variance ϕ of the underlying normal distribution. When the HPFMM procedure computes output statistics for the response variable (for example when you use the OUTPUT statement), the mean and variance of the response Y are reported. Similarly, the fit statistics apply to the distribution of $Y = |X|$, not the distribution of X. When you model a folded normal variable, the response input variable should be positive; the HPFMM procedure treats negative values of Y as a support violation.

Gamma(μ, ϕ)

$$l(\mu_i, \phi; y_i, w_i) = w_i\phi \log\left\{\frac{w_i y_i \phi}{\mu_i}\right\} - \frac{w_i y_i \phi}{\mu_i} - \log\{y_i\} - \log\{\Gamma(w_i\phi)\}$$

In this parameterization, $\mathrm{E}[Y] = \mu$ and $\mathrm{Var}[Y] = \mu^2/\phi$, $\phi > 0$. This parameterization of the gamma distribution differs from that in the GLIMMIX procedure, which expresses the log-likelihood function in terms of $1/\phi$ in order to achieve a variance function suitable for mixed model analysis.

Geometric(μ)

$$\begin{aligned} l(\mu_i; y_i, w_i) = &\, y_i \log\left\{\frac{\mu_i}{w_i}\right\} - (y_i + w_i)\log\left\{1 + \frac{\mu_i}{w_i}\right\} \\ &+ \log\left\{\frac{\Gamma(y_i + w_i)}{\Gamma(w_i)\Gamma(y_i + 1)}\right\} \end{aligned}$$

In this parameterization, $\mathrm{E}[Y] = \mu$ and $\mathrm{Var}[Y] = \mu + \mu^2$. The geometric distribution is a special case of the negative binomial distribution with $\phi = 1$.

Generalized Poisson(μ, ϕ)

$$\begin{aligned} \xi_i &= (1 - \exp\{-\phi\})/w_i \\ \mu_i^* &= \mu_i - \xi(\mu_i - y_i) \\ l(\mu_i^*, \xi_i; y_i, w_i) &= \log\{\mu_i^* - \xi_i y_i\} + (y_i - 1)\log\{\mu_i^*\} \\ &\quad -\mu_i^* - \log\{\Gamma(y_i + 1)\} \end{aligned}$$

In this parameterization, $\mathrm{E}[Y] = \mu$, $\mathrm{Var}[Y] = \mu/(1-\xi)^2$, and $\phi \geq 0$. The HPFMM procedure models the mean μ through the effects in the MODEL statement and applies a log link by default. The generalized Poisson distribution provides an overdispersed alternative to the Poisson distribution; $\phi = \xi_i = 0$ produces the mass function of a regular Poisson random variable. For details about the generalized Poisson distribution and a comparison with the negative binomial distribution, see Joe and Zhu (2005).

Inverse Gaussian(μ, ϕ)

$$l(\mu_i, \phi; y_i, w_i) = -\frac{1}{2}\left[\frac{w_i(y_i - \mu_i)^2}{y_i \phi \mu_i^2} + \log\left\{\frac{\phi y_i^3}{w_i}\right\} + \log\{2\pi\}\right]$$

The variance is $\mathrm{Var}[Y] = \phi\mu^3$, $\phi > 0$.

Lognormal(μ, ϕ)

$$\begin{aligned} z_i &= \log\{y_i\} - \mu_i \\ l(\mu_i, \phi; y_i, w_i) &= -\frac{1}{2}\left(2\log\{y_i\} + \log\left\{\frac{\phi}{w_i}\right\} + \log\{2\pi\} + \frac{w_i z_i^2}{\phi}\right) \end{aligned}$$

If $X = \log\{Y\}$ has a normal distribution with mean μ and variance ϕ, then Y has the log-likelihood function $l(\mu_i, \phi; y_i, w_i)$. The HPFMM procedure models the lognormal distribution and not the "shortcut" version you can obtain by taking the logarithm of a random variable and modeling that as normally distributed. The two approaches are not equivalent, and the approach taken by PROC HPFMM is the actual lognormal distribution. Although the lognormal model is a member of the exponential family of distributions, it is not in the "natural" exponential family because it cannot be written in canonical form.

In terms of the parameters μ and ϕ of the underlying normal process for X, the mean and variance of Y are $\mathrm{E}[Y] = \exp\{\mu\}\sqrt{\omega}$ and $\mathrm{Var}[Y] = \exp\{2\mu\}\omega(\omega - 1)$, respectively, where $\omega = \exp\{\phi\}$. When you request predicted values with the OUTPUT statement, the HPFMM procedure computes $\mathrm{E}[Y]$ and not μ.

Negative binomial(μ, ϕ)

$$\begin{aligned} l(\mu_i, \phi; y_i, w_i) &= y_i \log\left\{\frac{\phi\mu_i}{w_i}\right\} - (y_i + w_i/\phi)\log\left\{1 + \frac{\phi\mu_i}{w_i}\right\} \\ &\quad + \log\left\{\frac{\Gamma(y_i + w_i/\phi)}{\Gamma(w_i/\phi)\Gamma(y_i + 1)}\right\} \end{aligned}$$

The variance is $\mathrm{Var}[Y] = \mu + \phi\mu^2$, $\phi > 0$.

For a given ϕ, the negative binomial distribution is a member of the exponential family. The parameter ϕ is related to the scale of the data because it is part of the variance function. However, it cannot be factored from the variance, as is the case with the ϕ parameter in many other distributions.

Normal(μ, ϕ)

$$l(\mu_i, \phi; y_i, w_i) = -\frac{1}{2}\left[\frac{w_i(y_i - \mu_i)^2}{\phi} + \log\left\{\frac{\phi}{w_i}\right\} + \log\{2\pi\}\right]$$

The mean and variance are $\mathrm{E}[Y] = \mu$ and $\mathrm{Var}[Y] = \phi$, respectively, $\phi > 0$

Poisson(μ)

$$l(\mu_i; y_i, w_i) = w_i(y_i \log\{\mu_i\} - \mu_i - \log\{\Gamma(y_i + 1)\})$$

The mean and variance are $\mathrm{E}[Y] = \mu$ and $\mathrm{Var}[Y] = \mu$.

(Shifted) T$(\nu; \mu, \phi)$

$$\begin{aligned} z_i &= -0.5\log\{\phi/\sqrt{w_i}\} + \log\{\Gamma(0.5(\nu+1)\} \\ &\quad - \log\{\Gamma(0.5\nu)\} - 0.5 \times \log\{\pi\nu\} \\ l(\mu_i, \phi; y_i, w_i) &= -\left(\frac{\nu+1}{2}\right)\log\left\{1 + \frac{w_i}{\nu}\frac{(y_i - \mu_i)^2}{\phi}\right\} + z_i \end{aligned}$$

In this parameterization $\mathrm{E}[Y] = \mu$ and $\mathrm{Var}[Y] = \phi\nu/(\nu - 2)$, $\phi > 0, \nu > 0$. Note that this form of the *t* distribution is not a non-central distribution, but that of a shifted central *t* random variable.

Truncated Exponential$(\mu; a, b)$

$$\begin{aligned} l(\mu_i; a, b, y_i, w_i) &= w_i \log\left\{\frac{w_i y_i}{\mu_i}\right\} - \frac{w_i y_i}{\mu_i} - \log\{y_i\Gamma(w_i)\} \\ &\quad - \log\left[\frac{\gamma\left(w_i, \frac{w_i b}{\mu_i}\right)}{\Gamma(w_i)} - \frac{\gamma\left(w_i, \frac{w_i a}{\mu_i}\right)}{\Gamma(w_i)}\right] \end{aligned}$$

where

$$\gamma(c_1, c_2) = \int_0^{c_2} t^{c_1 - 1}\exp(-t)\mathrm{d}t$$

is the lower incomplete gamma function. The mean and variance are

$$\begin{aligned} \mathrm{E}[Y] &= \frac{(a + \mu_i)\exp(-a/\mu_i) - (b + \mu_i)\exp(-b/\mu_i)}{\exp(-a/\mu_i) - \exp(-b/\mu_i)} \\ \mathrm{Var}[Y] &= \frac{(a^2 + 2a\mu_i + 2\mu_i^2)\exp(-a/\mu_i) - (b^2 + 2b\mu_i + 2\mu_i^2)\exp(-b/\mu_i)}{\exp(-a/\mu_i) - \exp(-b/\mu_i)} \\ &\quad - (\mathrm{E}[Y])^2 \end{aligned}$$

Truncated Lognormal$(\mu, \phi; a, b)$

$$\begin{aligned} z_i &= \log\{y_i\} - \mu_i \\ l(\mu_i, \phi; a, b, y_i, w_i) &= -\frac{1}{2}\left(2\log\{y_i\} + \log\left\{\frac{\phi}{w_i}\right\} + \log\{2\pi\} + \frac{w_i z_i^2}{\phi}\right) \\ &\quad - \log\left\{\Phi\left[\sqrt{w_i/\phi}(\log b - \mu_i)\right] - \Phi\left[\sqrt{w_i/\phi}(\log a - \mu_i)\right]\right\} \end{aligned}$$

where $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution. The mean and variance are

$$\mathrm{E}[Y] = \exp(\mu_i + 0.5\phi)\frac{\Phi\left(\sqrt{\phi} - \frac{\log a - \mu_i}{\sqrt{\phi}}\right) - \Phi\left(\sqrt{\phi} - \frac{\log b - \mu_i}{\sqrt{\phi}}\right)}{\Phi\left(\frac{\log b - \mu_i}{\sqrt{\phi}}\right) - \Phi\left(\frac{\log a - \mu_i}{\sqrt{\phi}}\right)}$$

$$\mathrm{Var}[Y] = \exp(2\mu_i + 2\phi)\frac{\Phi\left(2\sqrt{\phi} - \frac{\log a - \mu_i}{\sqrt{\phi}}\right) - \Phi\left(2\sqrt{\phi} - \frac{\log b - \mu_i}{\sqrt{\phi}}\right)}{\Phi\left(\frac{\log b - \mu_i}{\sqrt{\phi}}\right) - \Phi\left(\frac{\log a - \mu_i}{\sqrt{\phi}}\right)} - (\mathrm{E}[Y])^2$$

Truncated Negative binomial(μ, ϕ)

$$\begin{aligned} l(\mu_i, \phi; y_i, w_i) = {} & y_i \log\left\{\frac{\phi\mu_i}{w_i}\right\} - (y_i + w_i/\phi)\log\left\{1 + \frac{\phi\mu_i}{w_i}\right\} \\ & + \log\left\{\frac{\Gamma(y_i + w_i/\phi)}{\Gamma(w_i/\phi)\Gamma(y_i + 1)}\right\} \\ & - \log\left\{1 - \left(\frac{\phi\mu_i}{w_i} + 1\right)^{-w_i/\phi}\right\} \end{aligned}$$

The mean and variance are

$$\begin{aligned} \mathrm{E}[Y] &= \mu_i\left\{1 - (\phi\mu_i + 1)^{-1/\phi}\right\}^{-1} \\ \mathrm{Var}[Y] &= (1 + \phi\mu_i + \mu_i)\mathrm{E}[Y] - (\mathrm{E}[Y])^2 \end{aligned}$$

Truncated Normal$(\mu, \phi; a, b)$

$$\begin{aligned} l(\mu_i, \phi; a, b, y_i, w_i) = {} & -\frac{1}{2}\left[\frac{w_i(y_i - \mu_i)^2}{\phi} + \log\left\{\frac{\phi}{w_i}\right\} + \log\{2\pi\}\right] \\ & - \log\left\{\Phi\left[\sqrt{w_i/\phi}(b - \mu_i)\right] - \Phi\left[\sqrt{w_i/\phi}(a - \mu_i)\right]\right\} \end{aligned}$$

where $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution. The mean and variance are

$$\mathrm{E}[Y] = \mu_i + \sqrt{\phi}\frac{\mathrm{phi}\left(\frac{a - \mu_i}{\sqrt{\phi}}\right) - \mathrm{phi}\left(\frac{b - \mu_i}{\sqrt{\phi}}\right)}{\Phi\left(\frac{b - \mu_i}{\sqrt{\phi}}\right) - \Phi\left(\frac{a - \mu_i}{\sqrt{\phi}}\right)}$$

$$\begin{aligned} \mathrm{Var}[Y] = \phi\Bigg[1 + {} & \frac{\frac{a - \mu_i}{\sqrt{\phi}}\mathrm{phi}\left(\frac{a - \mu_i}{\sqrt{\phi}}\right) - \frac{b - \mu_i}{\sqrt{\phi}}\mathrm{phi}\left(\frac{b - \mu_i}{\sqrt{\phi}}\right)}{\Phi\left(\frac{b - \mu_i}{\sqrt{\phi}}\right) - \Phi\left(\frac{a - \mu_i}{\sqrt{\phi}}\right)} \\ & - \left\{\frac{\mathrm{phi}\left(\frac{a - \mu_i}{\sqrt{\phi}}\right) - \mathrm{phi}\left(\frac{b - \mu_i}{\sqrt{\phi}}\right)}{\Phi\left(\frac{b - \mu_i}{\sqrt{\phi}}\right) - \Phi\left(\frac{a - \mu_i}{\sqrt{\phi}}\right)}\right\}^2\Bigg] \end{aligned}$$

where $\mathrm{phi}(\cdot)$ is the probability density function of the standard normal distribution.

Truncated Poisson(μ)

$$l(\mu_i; y_i, w_i) = w_i(y_i \log\{\mu_i\} - \log\{\exp(\mu_i) - 1\} - \log\{\Gamma(y_i + 1)\})$$

The mean and variance are

$$\mathrm{E}[Y] = \frac{\mu}{1 - \exp(-\mu_i)}$$
$$\mathrm{Var}[Y] = \frac{\mu_i\,[1 - \exp(-\mu_i) - \mu_i \exp(-\mu_i)]}{[1 - \exp(-\mu_i)]^2}$$

Uniform(a, b)

$$l(\mu_i; y_i, w_i) = -\log\{b - a\}$$

The mean and variance are $\mathrm{E}[Y] = 0.5(a + b)$ and $\mathrm{Var}[Y] = (b - a)^2/12$.

Weibull(μ, ϕ)

$$l(\mu_i, \phi; y_i) = -\frac{\phi - 1}{\phi} \log\left\{\frac{y_i}{\mu_i}\right\} - \log\{\mu_i \phi\} - \exp\left\{\log\left\{\frac{y_i}{\mu_i}\right\}/\phi\right\}$$

In this particular parameterization of the two-parameter Weibull distribution, the mean and variance of the random variable Y are $\mathrm{E}[Y] = \mu\Gamma(1 + \phi)$ and $\mathrm{Var}[Y] = \mu^2\left\{\Gamma(1 + 2\phi) - \Gamma^2(1 + \phi)\right\}$.

Bayesian Analysis

Conjugate Sampling

The HPFMM procedure uses Bayesian analysis via a conjugate Gibbs sampler if the model belongs to a small class of mixture models for which a conjugate sampler is available. See the section "Gibbs Sampler" (Chapter 7, *SAS/STAT User's Guide*) in Chapter 7, "Introduction to Bayesian Analysis Procedures" (*SAS/STAT User's Guide*), for a general discussion of Gibbs sampling. Table 6.8 summarizes the models for which conjugate and Metropolis-Hastings samplers are available.

Table 6.8 Availability of Conjugate and Metropolis Samplers in the HPFMM Procedure

Effects (exclusive of intercept)	Distributions	Available Samplers
No	Normal or T	Conjugate or Metropolis-Hastings
Yes	Normal or T	Conjugate or Metropolis-Hastings
No	Binomial, binary, Poisson, exponential	Conjugate or Metropolis-Hastings
Yes	Binomial, binary, Poisson, exponential	Metropolis-Hastings only

The conjugate sampler enjoys greater efficiency than the Metropolis-Hastings sampler and has the advantage of sampling in terms of the natural parameters of the distribution.

You can always switch to the Metropolis-Hastings sampling algorithm in any model by adding the METROPOLIS option in the BAYES statement.

Metropolis-Hastings Algorithm

If Metropolis-Hastings is the only sampler available for the specified model (see Table 6.8) or if the METROPOLIS option is specified in the BAYES statement, PROC HPFMM uses the Metropolis-Hastings approach of Gamerman (1997). See the section "Metropolis and Metropolis-Hastings Algorithms" (Chapter 7, *SAS/STAT User's Guide*) in Chapter 7, "Introduction to Bayesian Analysis Procedures" (*SAS/STAT User's Guide*), for a general discussion of the Metropolis-Hastings algorithm.

The Gamerman (1997) algorithm derives a specific density that is used to generate proposals for the component-specific parameters $\boldsymbol{\beta}_j$. The form of this proposal density is multivariate normal, with mean $\mathbf{m}_j$ and covariance matrix $\mathbf{C}_j$ derived as follows.

Suppose $\boldsymbol{\beta}_j$ is the vector of model coefficients in the jth component and suppose that $\boldsymbol{\beta}_j$ has prior distribution $N(\mathbf{a}, \mathbf{R})$. Consider a generalized linear model (GLM) with link function $g(\mu) = \eta = \mathbf{x}'\boldsymbol{\beta}$ and variance function $a(\mu)$. The pseudo-response and weight in the GLM for a weighted least squares step are

$$
\begin{aligned}
y^* &= \eta + (y - \mu)/\frac{\partial \mu}{\partial \eta} \\
w &= \frac{\partial \mu}{\partial \eta}/a(\mu)
\end{aligned}
$$

If the model contains offsets or FREQ or WEIGHT statements, or if a *trials* variable is involved, suitable adjustments are made to these quantities.

In each component, $j = 1, \cdots, k$, form an adjusted cross-product matrix with a "pseudo" border

$$
\begin{bmatrix}
\mathbf{X}_j'\mathbf{W}_j\mathbf{X}_j + \mathbf{R}^{-1} & \mathbf{X}_j'\mathbf{W}_j\mathbf{y}_j^* + \mathbf{R}^{-1}\mathbf{a} \\
\mathbf{y}_j^{*\prime}\mathbf{W}_j\mathbf{X}_j + \mathbf{a}'\mathbf{R}^{-1} & c
\end{bmatrix}
$$

where $\mathbf{W}_j$ is a diagonal matrix formed from the pseudo-weights w, $\mathbf{y}^*$ is a vector of pseudo-responses, and c is arbitrary. This is basically a system of normal equations with ridging, and the degree of ridging is governed by the precision and mean of the normal prior distribution of the coefficients. Sweeping on the leading partition leads to

$$
\begin{aligned}
\mathbf{C}_j &= \left(\mathbf{X}_j'\mathbf{W}_j\mathbf{X}_j + \mathbf{R}^{-1}\right)^- \\
\mathbf{m}_j &= \mathbf{C}_j\left(\mathbf{X}_j'\mathbf{W}_j\mathbf{y}_j^* + \mathbf{R}^{-1}\mathbf{a}\right)
\end{aligned}
$$

where the generalized inverse is a reflexive, g_2-inverse (see the section "Linear Model Theory" (Chapter 3, *SAS/STAT User's Guide*) in Chapter 3, "Introduction to Statistical Modeling with SAS/STAT Software" (*SAS/STAT User's Guide*), for details).

PROC HPFMM then generates a proposed parameter vector from the resulting multivariate normal distribution, and then accepts or rejects this proposal according to the appropriate Metropolis-Hastings thresholds.

Latent Variables via Data Augmentation

In order to fit finite Bayesian mixture models, the HPFMM procedure treats the mixture model as a missing data problem and introduces an assignment variable **S** as in Dempster, Laird, and Rubin (1977). Since **S** is not observable, it is frequently referred to as a latent variable. The unobservable variable **S** assigns an observation to a component in the mixture model. The number of states, k, might be unknown, but it is known to be finite. Conditioning on the latent variable **S**, the component memberships of each observation is assumed to be known, and Bayesian estimation is straightforward for each component in the finite mixture model. That is, conditional on $S = j$, the distribution of the response is now assumed to be $f(y; \alpha_j, \beta_j | S = j)$. In other words, each distinct state of the random variable **S** leads to a distinct set of parameters. The parameters in each component individually are then updated using a conjugate Gibbs sampler (where available) or a Metropolis-Hastings sampling algorithm.

The HPFMM procedure assumes that the random variable **S** has a discrete multinomial distribution with probability π_j of belonging to a component j; it can occupy one of k states. The distribution for the latent variable **S** is

$$f(S_i = j | \pi_1, \ldots, \pi_k) = \text{multinomial}(1, \pi_1, \ldots, \pi_k)$$

where $f(\cdot|\cdot)$ denotes a conditional probability density. The parameters in the density π_j denote the probability that S takes on state j.

The HPFMM procedure assumes a conjugate Dirichlet prior distribution on the mixture proportions π_j written as:

$$p(\boldsymbol{\pi}) \quad = \quad \text{Dirichlet}(a_1, \ldots, a_k)$$

where $p(\cdot)$ indicates a prior distribution.

Using Bayes' theorem, the likelihood function and prior distributions determine a conditionally conjugate posterior distribution of **S** and $\boldsymbol{\pi}$ from the multinomial distribution and Dirichlet distribution, respectively.

Prior Distributions

The following list displays the parameterization of prior distributions for situations in which the HPFMM procedure uses a conjugate sampler in mixture models without model effects and certain basic distributions (binary, binomial, exponential, Poisson, normal, and t). You specify the parameters a and b in the formulas below in the MUPRIORPARMS= and PHIPRIORPARMS= options in the BAYES statement in these models.

Beta(a, b)

$$f(y) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} y^{a-1} (1-y)^{b-1}$$

where $a > 0$, $b > 0$. In this parameterization, the mean and variance of the distribution are $\mu = a/(a+b)$ and $\mu(1-\mu)/(a+b+1)$, respectively. The beta distribution is the prior distribution for the success probability in binary and binomial distributions when conjugate sampling is used.

Dirichlet$(a_1, \cdots, a_k)$

$$f(\mathbf{y}) = \frac{\Gamma\left(\sum_{i=1}^{k} a_i\right)}{\prod_{i=1}^{k} \Gamma(a_i)} y_1^{a_1-1} \cdots y_k^{a_k-1}$$

where $\sum_{i=1}^{k} y_i = 1$ and the parameters $a_i > 0$. If any a_i were zero, an improper density would result. The Dirichlet density is the prior distribution for the mixture probabilities. You can affect the choice of the a_i through the MIXPRIORPARMS option in the BAYES statement. If k=2, the Dirichlet is the same as the beta(a, b) distribution.

Gamma(a, b)

$$f(y) = \frac{b^a}{\Gamma(a)} y^{a-1} \exp\{-by\}$$

where $a > 0$, $b > 0$. In this parameterization, the mean and variance of the distribution are $\mu = a/b$ and μ/b, respectively. The gamma distribution is the prior distribution for the mean parameter of the Poisson distribution when conjugate sampling is used.

Inverse gamma(a, b)

$$f(y) = \frac{b^a}{\Gamma(a)} y^{-a-1} \exp\{-b/y\}$$

where $a > 0$, $b > 0$. In this parameterization, the mean and variance of the distribution are $\mu = b/(a-1)$ if $a > 1$ and $\mu^2/(a-2)$ if $a > 2$, respectively. The inverse gamma distribution is the prior distribution for the mean parameter of the exponential distribution when conjugate sampling is used. It is also the prior distribution for the scale parameter ϕ in all models.

Multinomial$(1, \pi_1, \cdots, \pi_k)$

$$f(\mathbf{y}) = \frac{1}{y_1! \cdots y_k!} \pi_1^{y_1} \cdots \pi_k^{y_k}$$

where $\sum_{j=1}^{k} y_j = n$, $y_j \geq 0$, $\sum_{j=1}^{k} \pi_j = 1$, and n is the number of observations included in the analysis. The multinomial density is the prior distribution for the mixture proportions. The mean and variance of Y_j are $\mu_j = \pi_j$ and $\mu_j(1-\mu_j)$, respectively.

Normal(a, b)

$$f(y) = \frac{a}{\sqrt{2\pi b}} \exp\left\{-\frac{1}{2}\frac{(y-a)^2}{b}\right\}$$

where $b > 0$. The mean and variance of the distribution are $\mu = a$ and b, respectively. The normal distribution is the prior distribution for the mean parameter of the normal and t distribution when conjugate sampling is used.

When a MODEL statement contains effects or if you specify the METROPOLIS option, the prior distribution for the regression parameters is multivariate normal, and you can specify the means and variances of the parameters in the BETAPRIORPARMS= option in the BAYES statement.

Parameterization of Model Effects

PROC HPFMM constructs a finite mixture model according to the specifications in the CLASS, MODEL, and PROBMODEL statements. Each effect in the MODEL statement generates one or more columns in the matrix **X** for that model. The same **X** matrix applies to all components that are associated with the MODEL

statement. Each effect in the PROBMODEL statement generates one or more columns in the matrix **Z** from which the linear predictors in the model for the mixture probability models is formed. The same **Z** matrix applies to all components.

The formation of effects from continuous and classification variables in the HPFMM procedure follows the same general rules and techniques as for other linear modeling procedures. For information about constructing the model effects, see the section "Specification and Parameterization of Model Effects" on page 55 of Chapter 4, "Shared Statistical Concepts."

Computational Method

Multithreading

Threading refers to the organization of computational work into multiple tasks (processing units that can be scheduled by the operating system). A task is associated with a thread. Multithreading refers to the concurrent execution of threads. When multithreading is possible, substantial performance gains can be realized compared to sequential (single-threaded) execution.

The number of threads spawned by the HPFMM procedure is determined by the number of CPUs on a machine and can be controlled in the following ways:

- You can specify the CPU count with the CPUCOUNT= SAS system option. For example, if you specify the following statements, the HPFMM procedure schedules threads as if it were executing on a system that had four CPUs, regardless of the actual CPU count:

```
options cpucount=4;
```

- You can specify the NTHREADS= option in the PERFORMANCE statement to determine the number of threads. This specification overrides the system option. Specify NTHREADS=1 to force single-threaded execution.

The number of threads per machine is displayed in the "Performance Information" table, which is part of the default output. The HPFMM procedure allocates one thread per CPU.

The tasks that are multithreaded by the HPFMM procedure are primarily defined by dividing the data processed on a single machine among the threads—that is, the HPFMM procedure implements multithreading through a data-parallel model. For example, if the input data set has 1,000 observations and you are running with four threads, then 250 observations are associated with each thread. All operations that require access to the data are then multithreaded. This operations include the following:

- variable levelization
- effect levelization
- formation of the crossproducts matrix
- objective function, gradient, and Hessian evaluations
- scoring of observations

In addition, operations on matrices such as sweeps might be multithreaded if the matrices are of sufficient size to realize performance benefits from managing multiple threads for the particular matrix operation.

Choosing an Optimization Algorithm

First- or Second-Order Algorithms

The factors that affect how you choose a particular optimization technique for a particular problem are complex. Occasionally, you might benefit from trying several algorithms.

For many optimization problems, computing the gradient takes more computer time than computing the function value. Computing the Hessian sometimes takes *much* more computer time and memory than computing the gradient, especially when there are many decision variables. Unfortunately, optimization techniques that do not use some kind of Hessian approximation usually require many more iterations than techniques that do use a Hessian matrix; as a result, the total run time of these techniques is often longer. Techniques that do not use the Hessian also tend to be less reliable. For example, they can terminate more easily at stationary points than at global optima.

Table 6.9 shows which derivatives are required for each optimization technique.

Table 6.9 Derivatives Required

Algorithm	First-Order	Second-Order
TRUREG	x	x
NEWRAP	x	x
NRRIDG	x	x
QUANEW	x	-
DBLDOG	x	-
CONGRA	x	-
LEVMAR	x	-
NMSIMP	-	-

The second-derivative methods (TRUREG, NEWRAP, and NRRIDG) are best for small problems for which the Hessian matrix is not expensive to compute. Sometimes the NRRIDG algorithm can be faster than the TRUREG algorithm, but TRUREG can be more stable. The NRRIDG algorithm requires only one matrix with $p(p+1)/2$ double words; TRUREG and NEWRAP require two such matrices. Here, p denotes the number of parameters in the optimization.

The first-derivative methods QUANEW and DBLDOG are best for medium-sized problems for which the objective function and the gradient are much faster to evaluate than the Hessian. In general, the QUANEW and DBLDOG algorithms require more iterations than TRUREG, NRRIDG, and NEWRAP, but each iteration can be much faster. The QUANEW and DBLDOG algorithms require only the gradient to update an approximate Hessian, and they require slightly less memory than TRUREG or NEWRAP.

The first-derivative method CONGRA is best for large problems for which the objective function and the gradient can be computed much faster than the Hessian and for which too much memory is required to store the (approximate) Hessian. In general, the CONGRA algorithm requires more iterations than QUANEW or

DBLDOG, but each iteration can be much faster. Because CONGRA requires only a factor of p double-word memory, many large applications can be solved only by CONGRA.

The no-derivative method NMSIMP is best for small problems for which derivatives are not continuous or are very difficult to compute.

The LEVMAR method is appropriate only for least squares optimization problems.

Each optimization method uses one or more convergence criteria that determine when it has converged. An algorithm is considered to have converged when any one of the convergence criteria is satisfied. For example, under the default settings, the QUANEW algorithm converges if ABSGCONV <1E−5, FCONV $< 2 \times \epsilon$, or GCONV <1E−8.

By default, the HPFMM procedure applies the NRRIDG algorithm because it can take advantage of multi-threading in Hessian computations and inversions. If the number of parameters becomes large, specifying TECHNIQUE=QUANEW (which is a first-order method with good overall properties) is recommended.

Algorithm Descriptions

The following subsections provide details about each optimization technique and follow the same order as Table 6.9.

Trust Region Optimization (TRUREG)

The trust region method uses the gradient $\mathbf{g}(\boldsymbol{\psi}^{(k)})$ and the Hessian matrix $\mathbf{H}(\boldsymbol{\psi}^{(k)})$; thus, it requires that the objective function $f(\boldsymbol{\psi})$ have continuous first- and second-order derivatives inside the feasible region.

The trust region method iteratively optimizes a quadratic approximation to the nonlinear objective function within a hyperelliptic trust region that has radius Δ. Th radius constrains the step size that corresponds to the quality of the quadratic approximation. The trust region method is implemented based on Dennis, Gay, and Welsch (1981); Gay (1983); Moré and Sorensen (1983).

The trust region method performs well for small- to medium-sized problems, and it does not need many function, gradient, and Hessian calls. However, if the computation of the Hessian matrix is computationally expensive, one of the quasi-Newton or conjugate gradient algorithms might be more efficient.

Newton-Raphson Optimization with Line Search (NEWRAP)

The NEWRAP technique uses the gradient $\mathbf{g}(\boldsymbol{\psi}^{(k)})$ and the Hessian matrix $\mathbf{H}(\boldsymbol{\psi}^{(k)})$; thus, it requires that the objective function have continuous first- and second-order derivatives inside the feasible region.

If second-order derivatives are computed efficiently and precisely, the NEWRAP method can perform well for medium-sized to large problems, and it does not need many function, gradient, and Hessian calls.

This algorithm uses a pure Newton step when the Hessian is positive-definite and when the Newton step reduces the value of the objective function successfully. Otherwise, a combination of ridging and line search is performed to compute successful steps. If the Hessian is not positive-definite, a multiple of the identity matrix is added to the Hessian matrix to make it positive-definite (Eskow and Schnabel 1991).

In each iteration, a line search is performed along the search direction to find an approximate optimum of the objective function. The default line-search method uses quadratic interpolation and cubic extrapolation (LIS=2).

Newton-Raphson Ridge Optimization (NRRIDG)

The NRRIDG technique uses the gradient $\mathbf{g}(\boldsymbol{\psi}^{(k)})$ and the Hessian matrix $\mathbf{H}(\boldsymbol{\psi}^{(k)})$; thus, it requires that the objective function have continuous first- and second-order derivatives inside the feasible region.

This algorithm uses a pure Newton step when the Hessian is positive-definite and when the Newton step reduces the value of the objective function successfully. If at least one of these two conditions is not satisfied, a multiple of the identity matrix is added to the Hessian matrix.

The NRRIDG method performs well for small- to medium-sized problems, and it does not require many function, gradient, and Hessian calls. However, if the computation of the Hessian matrix is computationally expensive, one of the quasi-Newton or conjugate gradient algorithms might be more efficient.

Because the NRRIDG technique uses an orthogonal decomposition of the approximate Hessian, each iteration of NRRIDG can be slower than an iteration of the NEWRAP technique, which works with a Cholesky decomposition. However, NRRIDG usually requires fewer iterations than NEWRAP.

Quasi-Newton Optimization (QUANEW)

The (dual) quasi-Newton method uses the gradient $\mathbf{g}(\boldsymbol{\psi}^{(k)})$, and it does not need to compute second-order derivatives because they are approximated. It works well for medium-sized to moderately large optimization problems, where the objective function and the gradient can be computed much faster than the Hessian. However, in general it requires more iterations than the TRUREG, NEWRAP, and NRRIDG techniques, which compute second-order derivatives. QUANEW is the default optimization algorithm because it provides an appropriate balance between the speed and stability that are required for most nonlinear mixed model applications.

The QUANEW technique that is implemented by the HPFMM procedure is the dual quasi-Newton algorithm, which updates the Cholesky factor of an approximate Hessian.

In each iteration, a line search is performed along the search direction to find an approximate optimum. The line-search method uses quadratic interpolation and cubic extrapolation to obtain a step size α that satisfies the Goldstein conditions (Fletcher 1987). One of the Goldstein conditions can be violated if the feasible region defines an upper limit of the step size. Violating the left-side Goldstein condition can affect the positive-definiteness of the quasi-Newton update. In that case, either the update is skipped or the iterations are restarted by using an identity matrix, resulting in the steepest descent or ascent search direction.

The QUANEW algorithm uses its own line-search technique.

Double-Dogleg Optimization (DBLDOG)

The double-dogleg optimization method combines the ideas of the quasi-Newton and trust region methods. In each iteration, the double-dogleg algorithm computes the step $\mathbf{s}^{(k)}$ as the linear combination of the steepest descent or ascent search direction $\mathbf{s}_1^{(k)}$ and a quasi-Newton search direction $\mathbf{s}_2^{(k)}$:

$$\mathbf{s}^{(k)} = \alpha_1 \mathbf{s}_1^{(k)} + \alpha_2 \mathbf{s}_2^{(k)}$$

The step is requested to remain within a prespecified trust region radius (Fletcher 1987, p. 107). Thus, the DBLDOG subroutine uses the dual quasi-Newton update but does not perform a line search.

The double-dogleg optimization technique works well for medium-sized to moderately large optimization problems, where the objective function and the gradient are much faster to compute than the Hessian. The implementation is based on Dennis and Mei (1979); Gay (1983), but it is extended for dealing with boundary and linear constraints. The DBLDOG technique generally requires more iterations than the TRUREG,

NEWRAP, and NRRIDG techniques, which require second-order derivatives; however, each of the DBLDOG iterations is computationally cheap. Furthermore, the DBLDOG technique requires only gradient calls for the update of the Cholesky factor of an approximate Hessian.

Conjugate Gradient Optimization (CONGRA)

Second-order derivatives are not required by the CONGRA algorithm and are not even approximated. The CONGRA algorithm can be expensive in function and gradient calls, but it requires only $O(p)$ memory for unconstrained optimization. In general, the algorithm must perform many iterations to obtain a precise solution, but each of the CONGRA iterations is computationally cheap.

The CONGRA algorithm should be used for optimization problems that have large p. For the unconstrained or boundary-constrained case, the CONGRA algorithm requires only $O(p)$ bytes of working memory, whereas all other optimization methods require order $O(p^2)$ bytes of working memory. During p successive iterations, uninterrupted by restarts or changes in the working set, the CONGRA algorithm computes a cycle of p conjugate search directions. In each iteration, a line search is performed along the search direction to find an approximate optimum of the objective function. The default line-search method uses quadratic interpolation and cubic extrapolation to obtain a step size α that satisfies the Goldstein conditions. One of the Goldstein conditions can be violated if the feasible region defines an upper limit for the step size. Other line-search algorithms can be specified with the LIS= option.

Levenberg-Marquardt Optimization (LEVMAR)

The LEVMAR algorithm performs a highly stable optimization; however, for large problems, it consumes more memory and takes longer than the other techniques. The Levenberg-Marquardt optimization technique is a slightly improved variant of the Moré (1978) implementation.

Nelder-Mead Simplex Optimization (NMSIMP)

The Nelder-Mead simplex method does not use any derivatives and does not assume that the objective function has continuous derivatives. The objective function itself needs to be continuous. This technique is quite expensive in the number of function calls, and it might be unable to generate precise results for $p \gg 40$.

The original Nelder-Mead simplex algorithm is implemented and extended to boundary constraints. This algorithm does not compute the objective for infeasible points, but it changes the shape of the simplex by adapting to the nonlinearities of the objective function. This adaptation contributes to an increased speed of convergence. NMSIMP uses a special termination criterion.

Output Data Set

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

- those variables explicitly created by the statement
- variables listed in the ID statement

This enables you to add output data set information that is necessary for subsequent SQL joins without copying the entire input data set to the output data set. For more information about output data sets that are

produced when PROC HPFMM is run in distributed mode, see the section "Output Data Sets" on page 34 in Chapter 3, "Shared Concepts and Topics."

Default Output

The following sections describe the output that PROC HPFMM produces by default. The output is organized into various tables, which are discussed in the order of appearance for maximum likelihood and Bayes estimation, respectively.

Performance Information

The "Performance Information" table is produced by default. It displays information about the execution mode. For single-machine mode, the table displays the number of threads used. For distributed mode, the table displays the grid mode (symmetric or asymmetric), the number of compute nodes, and the number of threads per node.

Model Information

The "Model Information" table displays basic information about the model, such as the response variable, frequency variable, link function, and the model category that the HPFMM procedure determined based on your input and options. The "Model Information" table is one of a few tables that are produced irrespective of estimation technique. Most other tables are specific to Bayes or maximum likelihood estimation.

If the analysis depends on generated random numbers, the "Model Information" table also displays the random number seed used to initialize the random number generators. If you repeat the analysis and pass this seed value in the SEED= option in the PROC HPFMM statement, an identical stream of random numbers results.

Class Level Information

The "Class Level Information" table lists the levels of every variable specified in the CLASS statement. You should check this information to make sure that the data are correct. You can adjust the order of the CLASS variable levels with the ORDER= option in the CLASS statement. You can suppress the "Class Level Information" table completely or partially with the NOCLPRINT= option in the PROC HPFMM statement.

Number of Observations

The "Number of Observations" table displays the number of observations read from the input data set and the number of observations used in the analysis. If you specify a FREQ statement, the table also displays the sum of frequencies read and used. If the *events/trials* syntax is used for the response, the table also displays the number of events and trials used in the analysis.

Note that the number of observations "used" in the analysis is not unambiguous in a mixture model. An observation that is "unusable" for one component distribution (because the response value is outside of the support of the distribution) might still be usable in the mixture model when the response value is in the support of another component distribution. You can affect the way in which PROC HPFMM handles exclusion of observations due to support violations with the EXCLUSION= option in the PROC HPFMM statement.

Response Profile

For binary data, the "Response Profile" table displays the ordered value from which the HPFMM procedure determines the probability being modeled as an event for binary data. For each response category level, the frequency used in the analysis is reported.

Default Output for Maximum Likelihood

Optimization Information

The "Optimization Information" table displays basic information about the optimization setup to determine the maximum likelihood estimates, such as the optimization technique, the parameters that participate in the optimization, and the number of threads used for the calculations. This table is not produced during model selection—that is, if the KMAX= option is specified in the MODEL statement.

Iteration History

The "Iteration History" table displays for each iteration of the optimization the number of function evaluations (including gradient and Hessian evaluations), the value of the objective function, the change in the objective function from the previous iteration, and the absolute value of the largest (projected) gradient element. The objective function used in the optimization in the HPFMM procedure is the negative of the mixture log likelihood; consequently, PROC HPFMM performs a minimization. This table is not produced if you specify the KMAX= option in the MODEL statement. If you wish to see the "Iteration History" table in this setting, you must also specify the FITDETAILS option in the PROC HPFMM statement.

Convergence Status

The convergence status table is a small ODS table that follows the "Iteration History" table in the default output. In the listing, it appears as a message that identifies whether the optimization succeeded and which convergence criterion was met. If the optimization fails, the message indicates the reason for the failure. If you save the "Convergence Status" table to an output data set, a numeric Status variable is added that allows you to assess convergence programmatically. The values of the Status variable encode the following:

0	Convergence was achieved or an optimization was not performed (because of TECHNIQUE=NONE).
1	The objective function could not be improved.
2	Convergence was not achieved because of a user interrupt or because a limit was exceeded, such as the maximum number of iterations or the maximum number of function evaluations. To modify these limits, see the MAXITER=, MAXFUNC=, and MAXTIME= options in the PROC HPFMM statement.
3	Optimization failed to converge because function or derivative evaluations failed at the starting values or during the iterations or because a feasible point that satisfies the parameter constraints could not be found in the parameter space.

Fit Statistics

The "Fit Statistics" table displays a variety of fit measures based on the mixture log likelihood in addition to the Pearson statistic. All statistics are presented in "smaller is better" form. If you are fitting a single-component normal, gamma, or inverse gaussian model, the table also contains the unscaled Pearson statistic. If you are fitting a mixture model or the model has been fitted under restrictions, the table also contains the number of effective components and the number of effective parameters.

The calculation of the information criteria uses the following formulas, where p denotes the number of effective parameters, n denotes the number of observations used (or the sum of the frequencies used if a FREQ statement is present), and l is the log likelihood of the mixture evaluated at the converged estimates:

$$\begin{aligned}
\text{AIC} &= -2l + 2p \\
\text{AICC} &= \begin{cases} -2l + 2pn/(n-p-1) & n > p+2 \\ -2l + 2p(p+2) & \text{otherwise} \end{cases} \\
\text{BIC} &= -2l + p\log(n)
\end{aligned}$$

The Pearson statistic is computed simply as

$$\text{Pearson statistic} = \sum_{i=1}^{n} f_i \frac{(y_i - \widehat{\mu}_i)^2}{\widehat{\text{Var}}[Y_i]}$$

where n denotes the number of observations used in the analysis, f_i is the frequency associated with the ith observation (or 1 if no frequency is specified), μ_i is the mean of the mixture, and the denominator is the variance of the ith observation in the mixture. Note that the mean and variance in this expression are not those of the component distributions, but the mean and variance of the mixture:

$$\begin{aligned}
\mu_i &= \text{E}[Y_i] = \sum_{j=1}^{k} \pi_{ij}\mu_{ij} \\
\text{Var}[Y_i] &= -\mu_i^2 + \sum_{j=1}^{k} \pi_{ij}\left(\sigma_{ij}^2 + \mu_{ij}^2\right)
\end{aligned}$$

where μ_{ij} and σ_{ij}^2 are the mean and variance, respectively, for observation i in the jth component distribution and π_{ij} is the mixing probability for observation i in component j.

The unscaled Pearson statistic is computed with the same expression as the Pearson statistic with n, f_i, and μ_i as previously defined, but the scale parameter ϕ is set to 1 in the $\widehat{\text{Var}}[Y_i]$ expression.

The number of effective components and the number of effective parameters are determined by examining the converged solution for the parameters that are associated with model effects and the mixing probabilities. For example, if a component has an estimated mixing probability of zero, the values of its parameter estimates are immaterial. You might argue that all parameters should be counted towards the penalty in the information criteria. But a component with zero mixing probability in a k-component model effectively reduces the model to a $(k-1)$-component model. A situation of an overfit model, for which a parameter penalty needs to be taken when calculating the information criteria, is a different situation; here the mixing probability might be small, possibly close to zero.

Parameter Estimates

The parameter estimates, their estimated (asymptotic) standard errors, and p-values for the hypothesis that the parameter is zero are presented in the "Parameter Estimates" table. A separate table is produced for each MODEL statement, and the components that are associated with a MODEL statement are identified with an overall component count variable that counts across MODEL statements. If you assign a label to a model with the LABEL= option in the MODEL statement, the label appears in the title of the "Parameter Estimates" table. Otherwise, the internal label generated by the HPFMM procedure is used.

If the MODEL statement does not contain effects and the link function is not the identity, the inversely linked estimate is also displayed in the table. For many distributions, the inverse linked estimate is the estimated mean on the data scale. For example, in a binomial or binary model, it represents the estimated probability of an event. For some distributions (for example, the Weibull distribution), the inverse linked estimate is not the component distribution mean.

If you request confidence intervals with the CL or ALPHA= option in the MODEL statement, confidence limits are produced for the estimate on the linear scale. If the inverse linked estimate is displayed, confidence intervals for that estimate are also produced by inversely linking the confidence bounds on the linear scale.

Mixing Probabilities

If you fit a model with more than one component, the table of mixing probabilities is produced. If there are no effects in the PROBMODEL statement or if there is no PROBMODEL statement, the parameters are reported on the linear scale and as mixing probabilities. If model effects are present, only the linear parameters (on the scale of the logit, generalized logit, probit, and so on) are displayed.

Default Output for Bayes Estimation

Bayes Information

This table provides basic information about the sampling algorithm. The HPFMM procedure uses either a conjugate sampler or a Metropolis-Hastings sampling algorithm based on Gamerman (1997). The table reveals, for example, how many model parameters are sampled, how many parameters associated with mixing probabilities are sampled, and how many threads are used to perform multithreaded analysis.

Prior Distributions

The "Prior Distributions" table lists for each sampled parameter the prior distribution and its parameters. The mean and variance (if they exist) for those values of the parameters are also displayed, along with the initial value for the parameter in the Markov chain. The Component column in this table identifies the mixture component to which a particular parameter belongs. You can control how the HPFMM procedure determines initial values with the INITIAL= option in the BAYES statement.

Bayesian Fit Statistics

The "Bayesian Fit Statistics" table shows three measures based on the posterior sample. The "Average -2 Log Likelihood" is derived from the average mixture log-likelihood for the data, where the average is taken over the posterior sample. The deviance information criterion (DIC) is a Bayesian measure of model fit and the effective number of parameters (p_D) is a penalization term used in the computation of the DIC. See the section "Summary Statistics" (Chapter 7, *SAS/STAT User's Guide*) in Chapter 7, "Introduction to Bayesian Analysis Procedures" (*SAS/STAT User's Guide*), for a detailed discussion of the DIC and p_D.

Posterior Summaries

The arithmetic mean, standard deviation, and percentiles of the posterior distribution of the parameter estimates are displayed in the "Posterior Summaries" table. By default, the HPFMM procedure computes the 25th, 50th (median), and 75th percentiles of the sampling distribution. You can modify the percentiles through suboptions of the STATISTICS option in the BAYES statement. If a parameter corresponds to a singularity in the design and was removed from sampling for that purpose, it is also displayed in the table of posterior summaries (and in other tables that relate to output from the BAYES statement). The posterior sample size for such a parameter is shown as N = 0.

Posterior Intervals

The table of "Posterior Intervals" displays equal-tail intervals and intervals of highest posterior density for each parameter. By default, intervals are computed for an α-level of 0.05, which corresponds to 95% intervals. You can modify this confidence level by providing one or more α values in the ALPHA= suboption of the STATISTICS option in the BAYES statement. The computation of these intervals is detailed in section "Summary Statistics" (Chapter 7, *SAS/STAT User's Guide*) in Chapter 7, "Introduction to Bayesian Analysis Procedures" (*SAS/STAT User's Guide*).

Posterior Autocorrelations

Autocorrelations for the posterior estimates are computed by default for autocorrelation lags 1, 5, 10, and 50, provided that a sufficient number of posterior samples is available. See the section "Assessing Markov Chain Convergence" (Chapter 7, *SAS/STAT User's Guide*) in Chapter 7, "Introduction to Bayesian Analysis Procedures" (*SAS/STAT User's Guide*), for the computation of posterior autocorrelations and their utility in diagnosing convergence of Markov chains. You can modify the list of lags for which posterior autocorrelations are calculated with the AUTOCORR suboption of the DIAGNOSTICS= option in the BAYES statement.

ODS Table Names

Each table created by PROC HPFMM has a name associated with it, and you must use this name to reference the table when you use ODS statements. These names are listed in Table 6.10.

Table 6.10 ODS Tables Produced by PROC HPFMM

Table Name	Description	Required Statement / Option
Autocorr	Autocorrelation among posterior estimates	BAYES
BayesInfo	Basic information about Bayesian estimation	BAYES
ClassLevels	Level information from the CLASS statement	CLASS
CompDescription	Component description in models with varying number of components	KMAX= in MODEL with ML estimation
CompEvaluation	Comparison of mixture models with varying number of components	KMAX= in MODEL with ML estimation
CompInfo	Component information	COMPONENTINFO option in PROC HPFMM statement
ConvergenceStatus	Status of optimization at conclusion of optimization	Default output
Constraints	Linear equality and inequality constraints	RESTRICT statement or EQUATE=EFFECTS option in MODEL statement

Table 6.10 *continued*

Table Name	Description	Required Statement / Option
Corr	Asymptotic correlation matrix of parameter estimates (ML) or empirical correlation matrix of the Bayesian posterior estimates	CORR option in PROC HPFMM statement
Cov	Asymptotic covariance matrix of parameter estimates (ML) or empirical covariance matrix of the Bayesian posterior estimates	COV option in PROC HPFMM statement
CovI	Inverse of the covariance matrix of the parameter estimates	COVI option in PROC HPFMM statement
ESS	Effective sample size	DIAG=ESS option in BAYES statement
FitStatistics	Fit statistics	Default output
Geweke	Geweke diagnostics (Geweke 1992) for Markov chain	DIAG=GEWEKE option in BAYES statement
Hessian	Hessian matrix from the maximum likelihood optimization, evaluated at the converged estimates	HESSIAN
IterHistory	Iteration history	Default output for ML estimation
MCSE	Monte Carlo standard errors	DIAG=MCERROR in BAYES statement
MixingProbs	Solutions for the parameter estimates associated with effects in PROBMODEL statements	Default output for ML estimation if number of components is greater than 1
ModelInfo	Model information	Default output
NObs	Number of observations read and used, number of trials and events	Default output
OptInfo	Optimization information	Default output for ML estimation
ParameterEstimates	Solutions for the parameter estimates associated with effects in MODEL statements	Default output for ML estimation
ParameterMap	Mapping of parameter names to OUTPOST= data set	OUTPOST= option in BAYES statement
PriorInfo	Prior distributions and initial value of Markov chain	BAYES
PostSummaries	Summary statistics for posterior estimates	BAYES
PostIntervals	Equal-tail and highest posterior density intervals for posterior estimates	BAYES
ResponseProfile	Response categories and category modeled	Default output in models with binary response

ODS Graphics

You can reference every graph produced through ODS Graphics with a name. The names of the graphs that PROC HPFMM generates are listed in Table 6.11, along with the required statements and options.

Table 6.11 Graphs Produced by PROC HPFMM

ODS Graph Name	Plot Description	Option
TADPanel	Panel of diagnostic graphics to assess convergence of Markov chains	BAYES
DensityPlot	Histogram and density with component distributions	Default plot for homogeneous mixtures
CriterionPanel	Panel of plots showing progression of model fit criteria for mixtures with different numbers of components	KMIN= and KMAX= options in MODEL statement

Examples: HPFMM Procedure

Example 6.1: Modeling Mixing Probabilities: All Mice Are Created Equal, but Some Are More Equal

This example demonstrates how you can model the means and mixture proportions separately in a binomial cluster model. It also compares the binomial cluster model to the beta-binomial model.

In a typical teratological experiment, the offspring of animals that were exposed to a toxin during pregnancy are studied for malformation. If you count the number of malformed offspring in a litter of size n, then this count is typically not binomially distributed. The responses of the offspring from the same litter are not independent; hence their sum does not constitute a binomial random variable. Relative to a binomial model, data from teratological experiments exhibit overdispersion because ignoring positive correlation among the responses tends to overstate the precision of the parameter estimates. Overdispersion mechanisms are briefly discussed in the section "Overdispersion" on page 175.

In this application, the focus is on mixtures and models that involve a mixing mechanism. The mixing approach (Williams 1975; Haseman and Kupper 1979) supposes that the binomial success probability is a random variable that follows a $\text{beta}(\alpha, \beta)$ distribution:

$$\begin{aligned} Y|\mu &\sim \text{Binomial}(n, \mu) \\ \mu &\sim \text{Beta}(\alpha, \beta) \\ Y &\sim \text{Beta-binomial}(n, \mu, \phi) \\ \text{E}[Y] &= n\pi \\ \text{Var}[Y] &= n\pi(1-\pi)\left\{1 + \mu^2(n-1)\right\} \end{aligned}$$

If $\mu = 0$, then the beta-binomial distribution reduces to a standard binomial model with success probability π. The parameterization of the beta-binomial distribution used by the HPFMM procedure is based on Neerchal and Morel (1998), see the section "Log-Likelihood Functions for Response Distributions" on page 175 for details.

Morel and Nagaraj (1993); Morel and Neerchal (1997); Neerchal and Morel (1998) propose a different model to capture dependency within binomial clusters. Their model is a two-component mixture that gives rise to the same mean and variance function as the beta-binomial model. The genesis is different, however. In the binomial cluster model of Morel and Neerchal, suppose there is a cluster of n Bernoulli outcomes with success probability π. The number of responses in the cluster decomposes into $N \leq n$ outcomes that all respond with either "success" or "failure"; the important aspect is that they all respond identically. The remaining $n - N$ Bernoulli outcomes respond independently, so the sum of successes in this group is a binomial$(n - N, \pi)$ random variable. Denote the probability with which cluster members fall into the group of identical respondents as μ. Then $1 - \mu$ is the probability that a response belongs to the group of independent Bernoulli outcomes.

It is easy to see how this process of dividing the individual Bernoulli outcomes creates clustering. The binomial cluster model can be written as the two-component mixture

$$\Pr(Y = y) = \pi \Pr(U = y) + (1 - \pi) \Pr(V = y)$$

where $U \sim \text{Binomial}(n, \mu^* + \mu)$, $V \sim \text{Binomial}(n, \mu^*)$, and $\mu^* = (1 - \mu)\pi$. This mixture model is somewhat unusual because the mixing probability π appears as a parameter in the component distributions. The two probabilities involved, π and μ, have the following interpretation: π is the unconditional probability of success for any observation, and μ is the probability with which the Bernoulli observations respond identically. The complement of this probability, $1 - \mu$, is the probability with which the Bernoulli outcomes respond independently. If $\mu = 0$, then the two-component mixture reduces to a standard Binomial model with success probability π. Since both π and μ are involved in the success probabilities of the two Binomial variables in the mixture, you can affect these binomial means by specifying effects in the PROBMODEL statement (for the πs) or the MODEL statement (for the μs). In a "straight" two-component Binomial mixture,

$$\pi\text{Binomial}(n, \mu_1) + (1 - \pi)\text{Binomial}(n, \mu_2)$$

you would vary the success probabilities μ_1 and μ_2 through the MODEL statement.

With the HPFMM procedure, you can fit the beta-binomial model by specifying DIST=BETABIN and the binomial cluster model by specifying DIST=BINOMCLUS in the MODEL statement.

Morel and Neerchal (1997) report data from a completely randomized design that studies the teratogenicity of phenytoin in 81 pregnant mice. The treatment structure of the experiment is an augmented factorial. In

addition to an untreated control, mice received 60 mg/kg of phenytoin (PHT), 100 mg/kg of trichloropropene oxide (TCPO), and their combination. The design was augmented with a control group that was treated with water. As in Morel and Neerchal (1997), the two control groups are combined here into a single group.

The following DATA step creates the data for this analysis as displayed in Table 1 of Morel and Neerchal (1997). The second DATA step creates continuous variables x1–x3 to match the parameterization of these authors.

```
data ossi;
   length tx $8;
   input tx$ n @@;
   do i=1 to n;
      input y m @@;
      output;
   end;
   drop i;
   datalines;
Control  18 8 8 9  9  7  9 0  5 3  3 5 8 9 10 5 8 5 8 1 6 0 5
           8 8 9 10  5  5 4  7 9 10 6 6 3  5
Control  17 8 9 7 10 10 10 1  6 6  6 1 9 8  9 6 7 5 5 7 9
           2 5 5  6  2  8 1  8 0  2 7 8 5  7
PHT      19 1 9 4  9  3  7 4  7 0  7 0 4 1  8 1 7 2 7 2 8 1 7
           0 2 3 10  3  7 2  7 0  8 0 8 1 10 1 1
TCPO     16 0 5 7 10  4  4 8 11 6 10 6 9 3  4 2 8 0 6 0 9
           3 6 2  9  7  9 1 10 8  8 6 9
PHT+TCPO 11 2 2 0  7  1  8 7  8 0 10 0 4 0  6 0 7 6 6 1 6 1 7
;


data ossi;
   set ossi;
   array xx{3} x1-x3;
   do i=1 to 3; xx{i}=0; end;
   pht  = 0;
   tcpo = 0;
   if (tx='TCPO') then do;
      xx{1} = 1;
      tcpo  = 100;
   end; else if (tx='PHT') then do;
      xx{2} = 1;
      pht   = 60;
   end; else if (tx='PHT+TCPO') then do;
      pht  = 60;
      tcpo = 100;
      xx{1} = 1; xx{2} = 1; xx{3}=1;
   end;
run;
```

The HPFMM procedure models the mean parameters μ through the MODEL statement and the mixing proportions π through the PROBMODEL statement. In the binomial cluster model, you can place a regression structure on either set of probabilities, and the regression structure does not need to be the same. In the following statements, the unconditional probability of ossification is modeled as a two-way factorial, whereas the intralitter effect—the propensity to group within a cluster—is assumed to be constant:

```
proc hpfmm data=ossi;
   class pht tcpo;
   model y/m = / dist=binomcluster;
   probmodel pht tcpo pht*tcpo;
run;
```

The CLASS statement declares the PHT and TCPO variables as classification variables. They affect the analysis through their levels, not through their numeric values. The MODEL statement declares the distribution of the data to follow a binomial cluster model. The HPFMM procedure then automatically assumes that the model is a two-component mixture. An intercept is included by default. The PROBMODEL statement declares the effect structure for the mixing probabilities. The unconditional probability of ossification of a fetus depends on the main effects and the interaction in the factorial.

The "Model Information" table displays important details about the model fit with the HPFMM procedure (Output 6.1.1). Although no K= option was specified in the MODEL statement, the HPFMM procedure recognizes the model as a two-component model. The "Class Level Information" table displays the levels and values of the PHT and TCPO variables. Eighty-one observations are read from the data and are used in the analysis. These observations comprise 287 events and 585 total outcomes.

Output 6.1.1 Model Information in Binomial Cluster Model with Constant Clustering Probability

The HPFMM Procedure

Model Information	
Data Set	OSSI
Response Variable (Events)	y
Response Variable (Trials)	m
Type of Model	Binomial Cluster
Distribution	Binomial Cluster
Components	2
Link Function	Logit
Estimation Method	Maximum Likelihood

Class Level Information		
Class	**Levels**	**Values**
pht	2	0 60
tcpo	2	0 100

Number of Observations Read	81
Number of Observations Used	81
Number of Events	287
Number of Trials	585

The "Optimization Information" table in Output 6.1.2 gives details about the maximum likelihood optimization. By default, the HPFMM procedure uses a quasi-Newton algorithm. The model contains five parameters, four of which are part of the model for the mixing probabilities. The fifth parameter is the intercept in the model for μ.

Output 6.1.2 Optimization in Binomial Cluster Model with Constant Clustering Probability

Optimization Information	
Optimization Technique	Dual Quasi-Newton
Parameters in Optimization	5
Mean Function Parameters	1
Scale Parameters	0
Mixing Prob Parameters	4

Iteration History

Iteration	Evaluations	Objective Function	Change	Max Gradient
0	5	174.92723892	.	43.78769
1	2	154.13180744	20.79543149	11.2346
2	3	153.26693611	0.86487133	6.888215
3	2	152.84974281	0.41719329	3.541977
4	3	152.61756033	0.23218248	2.783556
5	3	152.54795303	0.06960730	1.146807
6	3	152.52684929	0.02110374	0.034367
7	3	152.52671214	0.00013715	0.011511
8	3	152.52670799	0.00000415	0.000202
9	3	152.52670799	0.00000000	4.001E-6

Convergence criterion (GCONV=1E-8) satisfied.

Fit Statistics	
-2 Log Likelihood	305.1
AIC (Smaller is Better)	315.1
AICC (Smaller is Better)	315.9
BIC (Smaller is Better)	327.0
Pearson Statistic	89.2077
Effective Parameters	5
Effective Components	2

After nine iterations, the iterative optimization converges. The –2 log likelihood at the converged solution is 305.1, and the Pearson statistic is 89.2077. The HPFMM procedure computes the Pearson statistic as a general goodness-of-fit measure that expresses the closeness of the fitted model to the data.

The estimates of the parameters in the conditional probability μ and in the unconditional probability π are given in Output 6.1.3. The intercept estimate in the model for μ is 0.3356. Since the default link in the binomial cluster model is the logit link, the estimate of the conditional probability is

$$\widehat{\mu} = \frac{1}{1 + \exp\{-0.3356\}} = 0.5831$$

This value is displayed in the "Inverse Linked Estimate" column. There is greater than a 50% chance that the individual fetuses in a litter provide the same response. The clustering tendency is substantial.

Output 6.1.3 Parameter Estimates in Binomial Cluster Model with Constant Clustering Probability

Parameter Estimates for Binomial Cluster Model

Component	Effect	Estimate	Standard Error	z Value	Pr > |z|	Inverse Linked Estimate
1	Intercept	0.3356	0.1714	1.96	0.0503	0.5831

Parameter Estimates for Mixing Probabilities

Component	Effect	pht	tcpo	Estimate	Standard Error	z Value	Pr > |z|
1	Intercept			-1.2194	0.4690	-2.60	0.0093
1	pht	0		0.9129	0.5608	1.63	0.1036
1	pht	60		0	.	.	.
1	tcpo		0	0.3295	0.5534	0.60	0.5516
1	tcpo		100	0	.	.	.
1	pht*tcpo	0	0	0.6162	0.6678	0.92	0.3561
1	pht*tcpo	0	100	0	.	.	.
1	pht*tcpo	60	0	0	.	.	.
1	pht*tcpo	60	100	0	.	.	.

The "Mixing Probabilities" table displays the estimates of the parameters in the model for π on the logit scale (Output 6.1.3). Table 6.12 constructs the estimates of the unconditional probabilities of ossification.

Table 6.12 Estimates of Ossification Probabilities

PHT	TCPO	$\widehat{\eta}$	$\widehat{\pi}$
0	0	–1.2194 + 0.9129 + 0.3295 + 0.6162 = 0.6392	0.6546
60	0	–1.2194 + 0.3295 = –0.8899	0.2911
0	100	–1.2194 + 0.9129 = –0.3065	0.4240
60	100	–1.2194	0.2280

Morel and Neerchal (1997) considered a model in which the intralitter effects also depend on the treatments. This model is fit with the HPFMM procedure with the following statements:

```
proc hpfmm data=ossi;
   class pht tcpo;
   model y/m = pht tcpo pht*tcpo / dist=binomcluster;
   probmodel   pht tcpo pht*tcpo;
run;
```

The –2 log likelihood of this model is much reduced compared to the previous model with constant conditional probability (compare 287.8 in Output 6.1.4 with 305.1 in Output 6.1.2). The likelihood-ratio statistic of 17.3 is significant, $\Pr(\chi^2_3 > 17.3 = 0.0006)$. Varying the conditional probabilities by treatment improved the model fit significantly.

Output 6.1.4 Fit Statistics and Parameter Estimates in Binomial Cluster Model

The HPFMM Procedure

Fit Statistics	
-2 Log Likelihood	287.8
AIC (Smaller is Better)	303.8
AICC (Smaller is Better)	305.8
BIC (Smaller is Better)	323.0
Pearson Statistic	85.5998
Effective Parameters	8
Effective Components	2

Parameter Estimates for Binomial Cluster Model

Component	Effect	pht	tcpo	Estimate	Standard Error	z Value	Pr > |z|
1	Intercept			1.8213	0.5889	3.09	0.0020
1	pht	0		-1.4962	0.6630	-2.26	0.0240
1	pht	60		0	.	.	.
1	tcpo		0	-3.1828	1.1261	-2.83	0.0047
1	tcpo		100	0	.	.	.
1	pht*tcpo	0	0	3.3736	1.1953	2.82	0.0048
1	pht*tcpo	0	100	0	.	.	.
1	pht*tcpo	60	0	0	.	.	.
1	pht*tcpo	60	100	0	.	.	.

Parameter Estimates for Mixing Probabilities

Component	Effect	pht	tcpo	Estimate	Standard Error	z Value	Pr > |z|
1	Intercept			-0.7394	0.5395	-1.37	0.1705
1	pht	0		0.4351	0.6203	0.70	0.4830
1	pht	60		0	.	.	.
1	tcpo		0	-0.5342	0.5893	-0.91	0.3646
1	tcpo		100	0	.	.	.
1	pht*tcpo	0	0	1.4055	0.7080	1.99	0.0471
1	pht*tcpo	0	100	0	.	.	.
1	pht*tcpo	60	0	0	.	.	.
1	pht*tcpo	60	100	0	.	.	.

Table 6.13 computes the conditional probabilities in the four treatment groups. Recall that the previous model estimated a constant clustering probability of 0.5831.

Table 6.13 Estimates of Clustering Probabilities

PHT	TCPO	$\widehat{\eta}$	$\widehat{\mu}$
0	0	1.8213 – 1.4962 – 3.1828 + 3.3736 = 0.5159	0.6262
60	0	1.8213 – 3.1828 = –1.3615	0.2040
0	100	1.8213 – 1.4962 = 0.3251	0.5806
60	100	1.8213	0.8607

The presence of phenytoin alone reduces the probability of response clustering within the litter. The presence of trichloropropene oxide alone does not have a strong effect on the clustering. The simultaneous presence of both agents substantially increases the probability of clustering.

The following statements fit the binomial cluster model in the parameterization of Morel and Neerchal (1997).

```
proc hpfmm data=ossi;
   model y/m = x1-x3 / dist=binomcluster;
   probmodel   x1-x3;
run;
```

The model fit is the same as in the previous model (compare the "Fit Statistics" tables in Output 6.1.5 and Output 6.1.4). The parameter estimates change due to the reparameterization of the treatment effects and match the results in Table III of Morel and Neerchal (1997).

Output 6.1.5 Fit Statistics and Estimates (Morel and Neerchal Parameterization)

The HPFMM Procedure

Fit Statistics	
-2 Log Likelihood	287.8
AIC (Smaller is Better)	303.8
AICC (Smaller is Better)	305.8
BIC (Smaller is Better)	323.0
Pearson Statistic	85.5999
Effective Parameters	8
Effective Components	2

Parameter Estimates for Binomial Cluster Model

Component	Effect	Estimate	Standard Error	z Value	Pr > |z|
1	Intercept	0.5159	0.2603	1.98	0.0475
1	x1	-0.1908	0.4006	-0.48	0.6339
1	x2	-1.8774	0.9946	-1.89	0.0591
1	x3	3.3736	1.1953	2.82	0.0048

Parameter Estimates for Mixing Probabilities

Component	Effect	Estimate	Standard Error	z Value	Pr > |z|
1	Intercept	0.5669	0.2455	2.31	0.0209
1	x1	-0.8712	0.3924	-2.22	0.0264
1	x2	-1.8405	0.3413	-5.39	<.0001
1	x3	1.4055	0.7080	1.99	0.0471

The following sets of statements fit the binomial and beta-binomial models, respectively, as single-component mixtures in the parameterization akin to the first binomial cluster model. Note that the model effects that affect the underlying Bernoulli success probabilities are specified in the MODEL statement, in contrast to the binomial cluster model.

```
proc hpfmm data=ossi;
   model y/m = x1-x3 / dist=binomial;
run;
```

```
proc hpfmm data=ossi;
   model y/m = x1-x3 / dist=betabinomial;
run;
```

The Pearson statistic for the beta-binomial model (Output 6.1.6) indicates a much better fit compared to the single-component binomial model (Output 6.1.7). This is not surprising since these data are obviously overdispersed relative to a binomial model because the Bernoulli outcomes are not independent. The difference between the binomial cluster and the beta-binomial model lies in the mechanism by which the correlations are induced:

- a mixing mechanism in the beta-binomial model that leads to a common shared random effect among all offspring in a cluster
- a mixture specification in the binomial cluster model that divides the offspring in a litter into identical and independent responders

Output 6.1.6 Fit Statistics in Binomial Model

The HPFMM Procedure

Fit Statistics	
-2 Log Likelihood	401.8
AIC (Smaller is Better)	409.8
AICC (Smaller is Better)	410.3
BIC (Smaller is Better)	419.4
Pearson Statistic	252.1

Output 6.1.7 Fit Statistics in Beta-Binomial Model

The HPFMM Procedure

Fit Statistics	
-2 Log Likelihood	306.6
AIC (Smaller is Better)	316.6
AICC (Smaller is Better)	317.4
BIC (Smaller is Better)	328.5
Pearson Statistic	87.5379

Example 6.2: The Usefulness of Custom Starting Values: When Do Cows Eat?

This example with a mixture of normal and Weibull distributions illustrates the benefits of specifying starting values for some of the components.

The data for this example were generously provided by Dr. Luciano A. Gonzalez of the Lethbridge Research Center of Agriculture and Agri-Food Canada and his collaborator, Dr. Bert Tolkamp, from the Scottish Agricultural College.

The outcome variable of interest is the logarithm of a time interval between consecutive visits by cattle to feeders. The intervals fall into three categories:

- short breaks within meals—such as when an animal stops eating for a moment and resumes shortly thereafter
- somewhat longer breaks when eating is interrupted to go have a drink of water
- long breaks between meals

Modeling such time interval data is important to understand the feeding behavior and biology of the animals and to derive other biological parameters such as the probability of an animal to stop eating after it has consumed a certain amount of a given food. Because there are three distinct biological categories, data of this nature are frequently modeled as three-component mixtures. The point at which the second and third components cross over is used to separate feeding events into meals.

The original data set comprises 141,414 observations of log feeding intervals. For the purpose of presentation in this document, where space is limited, the data have been rounded to precision 0.05 and grouped by frequency. The following DATA step displays the modified data used in this example. A comparison with the raw data and the results obtained in a full analysis of the original data show that the grouping does not alter the presentation or conclusions in a way that matters for the purpose of this example.

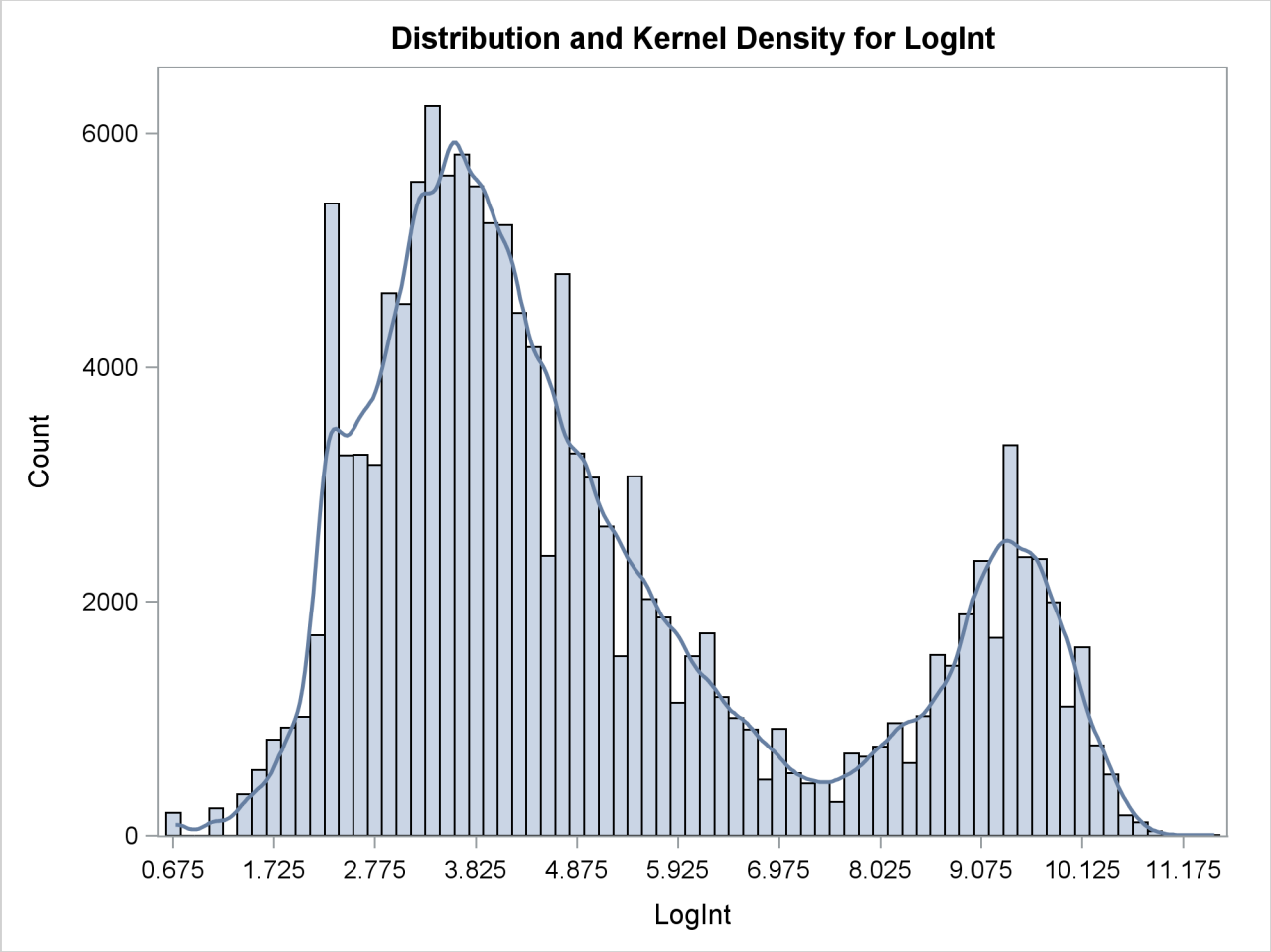
```
data cattle;
   input LogInt Count @@;
   datalines;
 0.70   195  1.10   233  1.40   355  1.60   563
 1.80   822  1.95   926  2.10  1018  2.20  1712
 2.30  3190  2.40  2212  2.50  1692  2.55  1558
 2.65  1622  2.70  1637  2.75  1568  2.85  1599
 2.90  1575  2.95  1526  3.00  1537  3.05  1561
 3.10  1555  3.15  1427  3.20  2852  3.25  1396
 3.30  1343  3.35  2473  3.40  1310  3.45  2453
 3.50  1168  3.55  2300  3.60  2174  3.65  2050
 3.70  1926  3.75  1849  3.80  1687  3.85  2416
 3.90  1449  3.95  2095  4.00  1278  4.05  1864
 4.10  1672  4.15  2104  4.20  1443  4.25  1341
 4.30  1685  4.35  1445  4.40  1369  4.45  1284
 4.50  1523  4.55  1367  4.60  1027  4.65  1491
 4.70  1057  4.75  1155  4.80  1095  4.85  1019
 4.90  1158  4.95  1088  5.00  1075  5.05   912
 5.10  1073  5.15   803  5.20   924  5.25   916
 5.30   784  5.35   751  5.40   766  5.45   833
 5.50   748  5.55   725  5.60   674  5.65   690
 5.70   659  5.75   695  5.80   529  5.85   639
 5.90   580  5.95   557  6.00   524  6.05   473
 6.10   538  6.15   444  6.20   456  6.25   453
 6.30   374  6.35   406  6.40   409  6.45   371
 6.50   320  6.55   334  6.60   353  6.65   305
 6.70   302  6.75   301  6.80   263  6.85   218
 6.90   255  6.95   240  7.00   219  7.05   202
 7.10   192  7.15   180  7.20   162  7.25   126
 7.30   148  7.35   173  7.40   142  7.45   163
```

```
 7.50   152  7.55   149  7.60   139  7.65   161
 7.70   174  7.75   179  7.80   188  7.85   239
 7.90   225  7.95   213  8.00   235  8.05   256
 8.10   272  8.15   290  8.20   320  8.25   355
 8.30   307  8.35   311  8.40   317  8.45   335
 8.50   369  8.55   365  8.60   365  8.65   396
 8.70   419  8.75   467  8.80   468  8.85   515
 8.90   558  8.95   623  9.00   712  9.05   716
 9.10   829  9.15   803  9.20   834  9.25   856
 9.30   838  9.35   842  9.40   826  9.45   834
 9.50   798  9.55   801  9.60   780  9.65   849
 9.70   779  9.75   737  9.80   683  9.85   686
 9.90   626  9.95   582 10.00   522 10.05   450
10.10   443 10.15   375 10.20   342 10.25   285
10.30   254 10.35   231 10.40   195 10.45   186
10.50   143 10.55   100 10.60    73 10.65    49
10.70    28 10.75    36 10.80    16 10.85     9
10.90     5 10.95     6 11.00     4 11.05     1
11.15     1 11.25     4 11.30     2 11.35     5
11.40     4 11.45     3 11.50     1
;
```

If you scan the columns for the Count variable in the DATA step, the prevalence of values between 2 and 5 units of LogInt is apparent, as is a long right tail. To explore these data graphically, the following statements produce a histogram of the data and a kernel density estimate of the density of the LogInt variable.

```
ods graphics on;
proc kde data=cattle;
   univar LogInt / bwm=4;
   freq count;
run;
```

Output 6.2.1 Histogram and Kernel Density for LogInt



Two modes are clearly visible in Output 6.2.1. Given the biological background, one would expect that three components contribute to the mixture. The histogram would suggest either a two-component mixture with modes near 4 and 9, or a three-component mixture with modes near 3, 5, and 9.

Following Dr. Gonzalez' suggestion, the process is modeled as a three-component mixture of two normal distributions and a Weibull distribution. The Weibull distribution is chosen because it can have long left and right tails and it is popular in modeling data that relate to time intervals.

```
proc hpfmm data=cattle gconv=0;
   model LogInt = / dist=normal k=2 parms(3 1, 5 1);
   model        + / dist=weibull;
   freq count;
run;
```

The GCONV= convergence criterion is turned off in this PROC HPFMM run to avoid the early stoppage of the iterations when the relative gradient changes little between iterations. Turning the criterion off usually ensures that convergence is achieved with a small absolute gradient of the objective function. The PARMS option in the first MODEL statement provides starting values for the means and variances for the parameters of the normal distributions. The means for the two components are started at $\mu = 3$ and $\mu = 5$, respectively. Specifying starting values is generally not necessary. However, the choice of starting values can play an

important role in modeling finite mixture models; the importance of the choice of starting values in this example is discussed further below.

The "Model Information" table shows that the model is a three-component mixture and that the HPFMM procedure considers the estimation of a density to be the purpose of modeling. The procedure draws this conclusion from the absence of effects in the MODEL statements. There are 187 observations in the data set, but these actually represent 141,414 measurements (Output 6.2.2).

Output 6.2.2 Model Information and Number of Observations

The HPFMM Procedure

Model Information	
Data Set	CATTLE
Response Variable	LogInt
Frequency Variable	Count
Type of Model	Density Estimation
Components	3
Estimation Method	Maximum Likelihood

Number of Observations Read	187
Number of Observations Used	187
Sum of Frequencies Read	141414
Sum of Frequencies Used	141414

There are eight parameters in the optimization: the means and variances of the two normal distributions, the μ and ϕ parameter of the Weibull distribution, and the two mixing probabilities (Output 6.2.3). At the converged solution, the –2 log likelihood is 563,153 and all parameters and components are effective—that is, the model is not overspecified in the sense that components have collapsed during the model fitting. The Pearson statistic is close to the number of observations in the data set, indicating a good fit.

Output 6.2.3 Optimization Information and Fit Statistics

Optimization Information	
Optimization Technique	Dual Quasi-Newton
Parameters in Optimization	8
Mean Function Parameters	3
Scale Parameters	3
Mixing Prob Parameters	2
Lower Boundaries	3
Upper Boundaries	0

Fit Statistics	
-2 Log Likelihood	563153
AIC (Smaller is Better)	563169
AICC (Smaller is Better)	563169
BIC (Smaller is Better)	563248
Pearson Statistic	141458
Effective Parameters	8
Effective Components	3

Output 6.2.4 displays the parameter estimates for the three models and for the mixing probabilities. The order in which the "Parameter Estimates" tables appear in the output corresponds to the order in which the MODEL statements were specified.

Output 6.2.4 Optimization Information and Fit Statistics

Parameter Estimates for Normal Model

Component	Parameter	Estimate	Standard Error	z Value	Pr > |z|
1	Intercept	3.3415	0.01260	265.16	<.0001
2	Intercept	4.8940	0.05447	89.84	<.0001
1	Variance	0.6718	0.01287		
2	Variance	1.4497	0.05247		

Parameter Estimates for Weibull Model

Component	Parameter	Estimate	Standard Error	z Value	Pr > |z|	Inverse Linked Estimate
3	Intercept	2.2531	0.000506	4452.11	<.0001	9.5174
3	Scale	0.06848	0.000427			

Parameter Estimates for Mixing Probabilities

Component	Mixing Probability	Linked Scale GLogit(Prob)	Standard Error	z Value	Pr > |z|
1	0.4545	0.8106	0.03409	23.78	<.0001
2	0.3435	0.5305	0.04640	11.43	<.0001
3	0.2021	0			

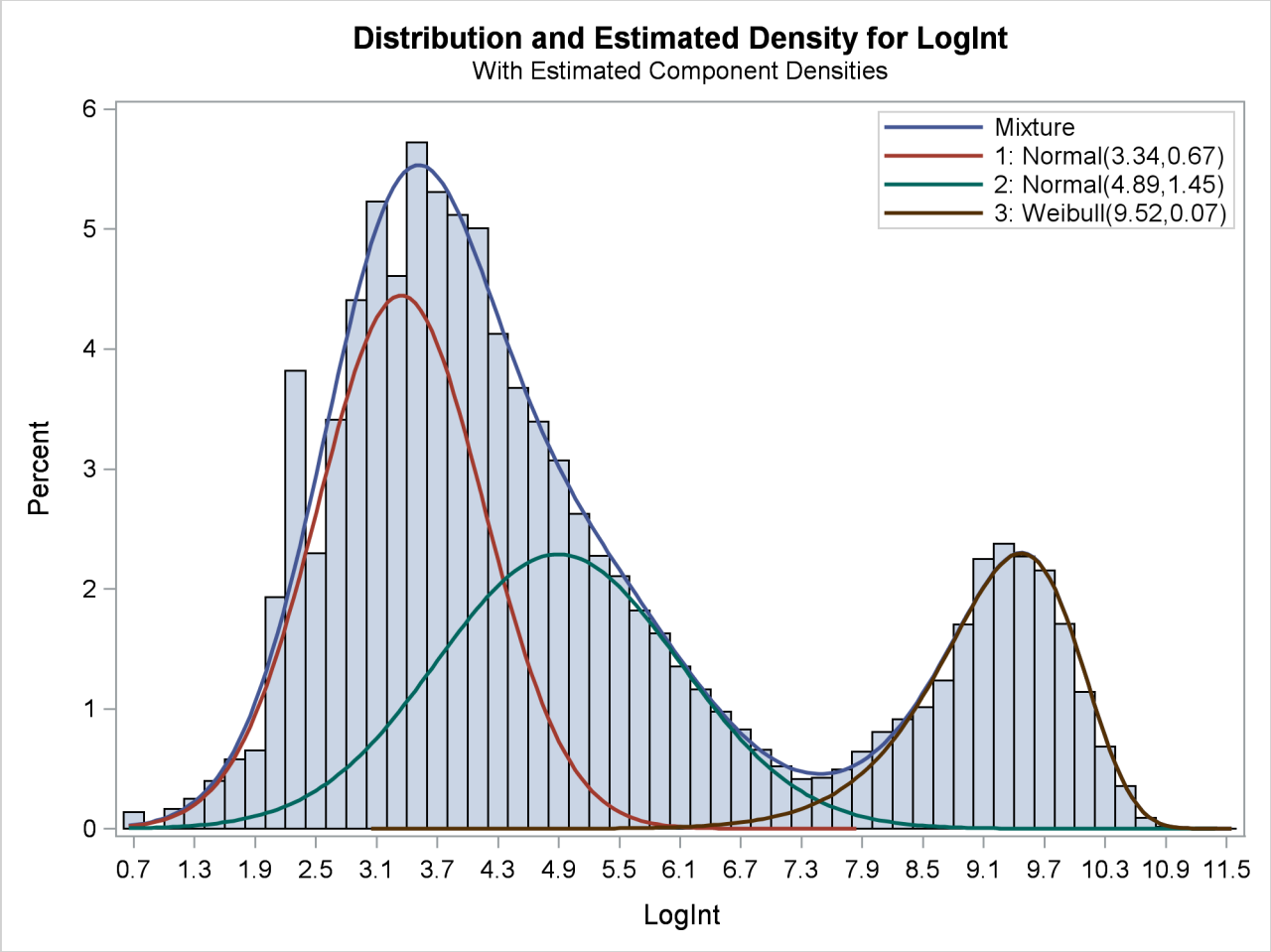
The estimated means of the two normal components are 3.3415 and 4.8940, respectively. Note that the means are displayed here as Intercept. The inverse linked estimate is not produced because the default link for the normal distribution is the identity link; hence the Estimate column represents the means of the component distributions. The parameter estimates in the Weibull model are $\widehat{\beta}_0 = 2.2531$, $\widehat{\phi} = 0.06848$, and $\widehat{\mu} = \exp\{\widehat{\beta}_0\} = 9.5174$. In the Weibull distribution, the μ parameter does not estimate the mean of the distribution, the maximum likelihood estimate of the distribution's mean is $\widehat{\mu}\Gamma(\widehat{\phi}+1) = 9.1828$.

The estimated mixing probabilities are $\widehat{\pi}_1 = 0.4545$, $\widehat{\pi}_2 = 0.3435$, and $\widehat{\pi_3} = 0.2021$. In other words, the estimated distribution of log feeding intervals is a 45:35:20 mixture of an $\mathrm{N}(3.3415, 0.6718)$, a $\mathrm{N}(4.8940, 1.4497)$, and a $\mathrm{Weibull}(9.5174, 0.06848)$ distribution.

You can obtain a graphical display of the observed and estimated distribution of these data by enabling ODS Graphics. The PLOTS option in the PROC HPFMM statement modifies the default density plot by adding the densities of the mixture components:

```
ods select DensityPlot;
proc hpfmm data=cattle gconv=0;
   model LogInt = / dist=normal k=2 parms(3 1, 5 1);
   model        + / dist=weibull;
   freq count;
run;
```

Output 6.2.5 Observed and Estimated Densities in the Three-Component Model



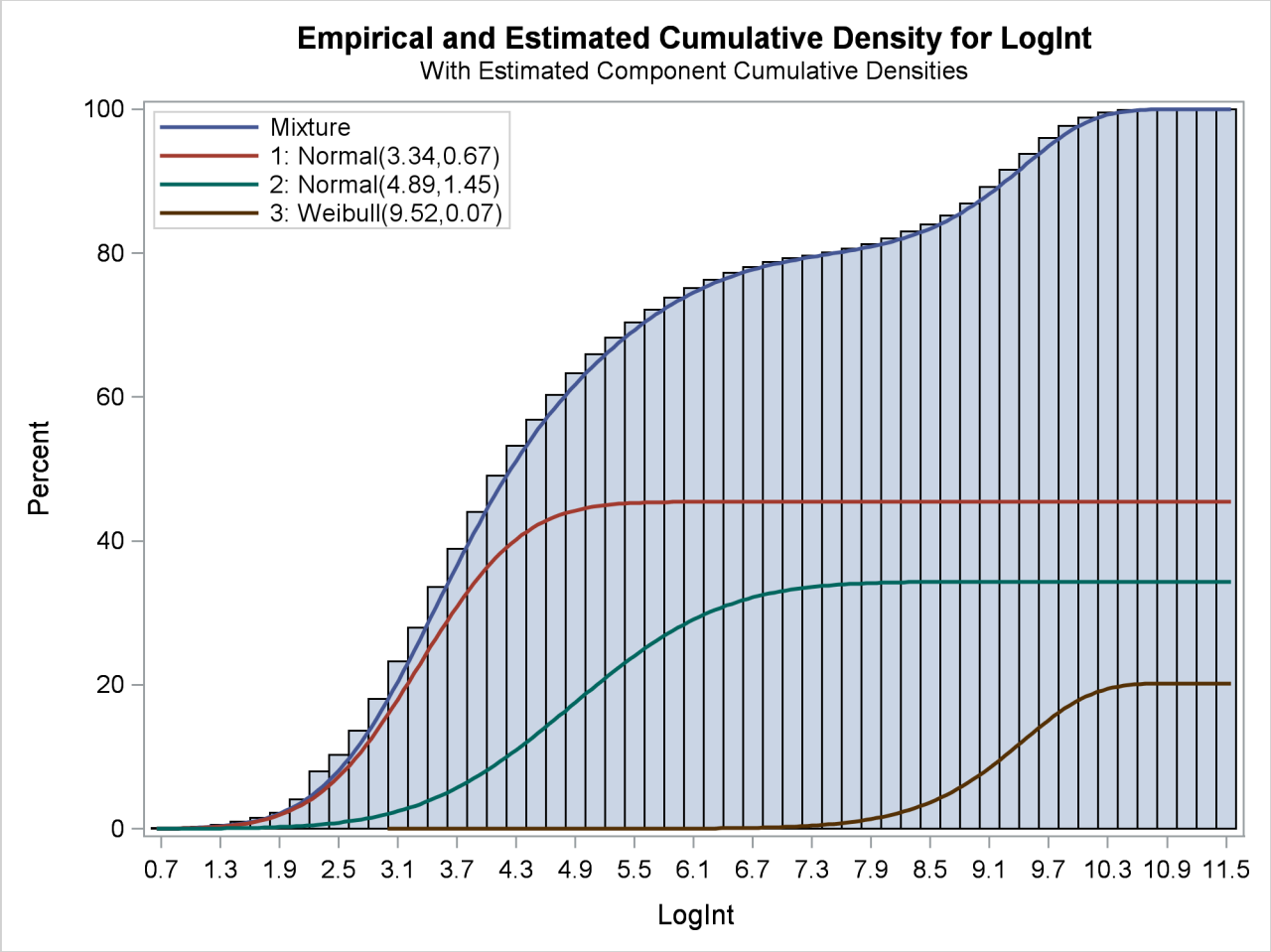
The estimated mixture density matches the histogram of the observed data closely (Output 6.2.5). The component densities are displayed in such a way that, at each point in the support of the LogInt variable, their sum combines to the overall mixture density. The three components in the mixtures are well separated.

The excellent quality of the fit is even more evident when the distributions are displayed cumulatively by adding the CUMULATIVE option in the DENSITY option (Output 6.2.6):

```
ods select DensityPlot;
proc hpfmm data=cattle plot=density(cumulative) gconv=0;
   model LogInt = / dist=normal k=2 parms(3 1, 5 1);
   model        + / dist=weibull;
   freq count;
run;
```

The component cumulative distribution functions are again scaled so that their sum produces the overall mixture cumulative distribution function. Because of this scaling, the percentage reached at the maximum value of LogInt corresponds to the mixing probabilities in Output 6.2.4.

Output 6.2.6 Observed and Estimated Cumulative Densities in the Three-Component Model



The importance of starting values for the parameter estimates was mentioned previously. Suppose that different starting values are selected for the three components (for example, the default starting values).

```
proc hpfmm data=cattle gconv=0;
   model LogInt = / dist=normal k=2;
   model        + / dist=weibull;
   freq count;
run;
ods graphics off;
```

The fit statistics and parameter estimates from this run are displayed in Output 6.2.7, and the density plot is shown in Output 6.2.8.

Output 6.2.7 Fit Statistics and Parameter Estimates

The HPFMM Procedure

Fit Statistics	
-2 Log Likelihood	564431
AIC (Smaller is Better)	564447
AICC (Smaller is Better)	564447
BIC (Smaller is Better)	564526
Pearson Statistic	141228
Effective Parameters	8
Effective Components	3

Parameter Estimates for Normal Model

Component	Parameter	Estimate	Standard Error	z Value	Pr > |z|
1	**Intercept**	4.9106	0.02604	188.56	<.0001
2	**Intercept**	9.2883	0.005031	1846.28	<.0001
1	**Variance**	1.7410	0.02753		
2	**Variance**	0.4158	0.005086		

Parameter Estimates for Weibull Model

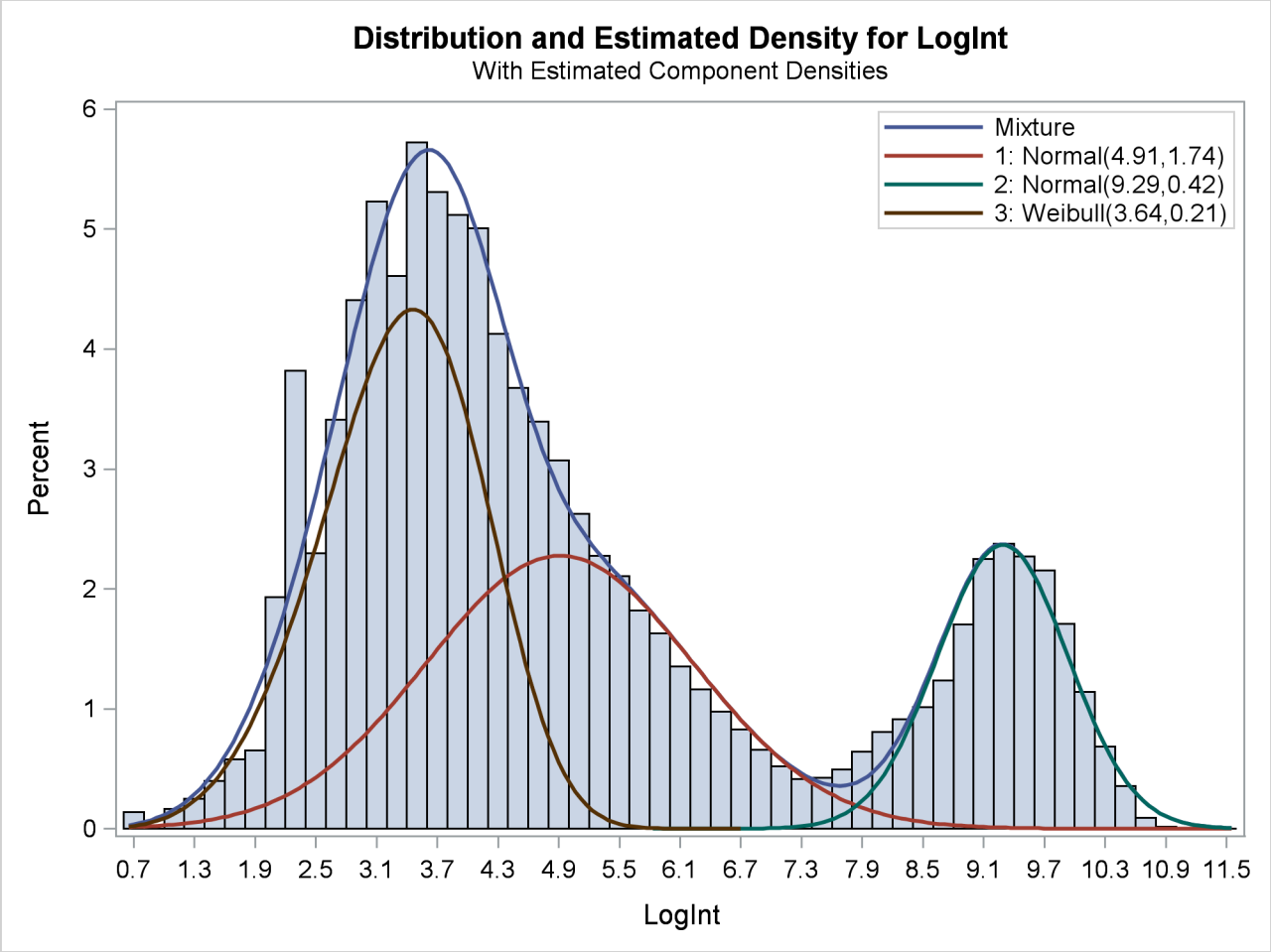
Component	Parameter	Estimate	Standard Error	z Value	Pr > |z|	Inverse Linked Estimate
3	**Intercept**	1.2908	0.002790	462.71	<.0001	3.6358
3	**Scale**	0.2093	0.001311			

Parameter Estimates for Mixing Probabilities

		Linked Scale			
Component	Mixing Probability	GLogit(Prob)	Standard Error	z Value	Pr > |z|
1	0.3745	-0.1505	0.03678	-4.09	<.0001
2	0.1902	-0.8280	0.01922	-43.08	<.0001
3	0.4353	0			

All components are active; no collapsing of components occurred. However, a closer look at the "Parameter Estimates" tables in Output 6.2.7 shows an important difference from the tables in Output 6.2.4. The means of the two normal distributions are now 4.9106 and 9.2883. Previously, the means were 3.3415 and 4.8940. The "position" of the Weibull distribution has moved from right to left, and the third component is now modeled by a symmetric normal distribution (Output 6.2.8). The mixture probabilities have also changed—in particular, for the first and third component.

Output 6.2.8 Three-Component Model with Default Starting Values



Such switching is not uncommon in mixture modeling. As judged by the information criteria, the model in which the Weibull distribution is the component with the smallest mean does not fit the data as well as the first model in which the specification of the starting values guided the optimization towards placing the normal distributions first. The converged solution found in the last HPFMM run represents a local minimum of the log-likelihood surface. There are other local minima—for example, when components are removed from the model, which is tantamount to estimating the associated mixture probabilities as zero.

Example 6.3: Enforcing Homogeneity Constraints: Count and Dispersion—It Is All Over!

The following example demonstrates how you can use either the EQUATE= option in the MODEL statement or the RESTRICT statement to impose homogeneity constraints on chosen model effects.

The data for this example were presented by Margolin, Kaplan, and Zeiger (1981) and analyzed by various authors applying a number of techniques. The following DATA step shows the number of revertant salmonella colonies (variable num) at six levels of quinoline dosing (variable dose). There are three replicate plates at each dose of quinoline.

```
data assay;
   label dose = 'Dose of quinoline (microg/plate)'
         num  = 'Observed number of colonies';
   input dose @;
   logd = log(dose+10);
   do i=1 to 3; input num@; output; end;
   datalines;
   0  15 21 29
  10  16 18 21
  33  16 26 33
 100  27 41 60
 333  33 38 41
1000  20 27 42
;
```

The basic notion is that the data are overdispersed relative to a Poisson distribution in which the logarithm of the mean count is modeled as a linear regression in dose (in μg/plate) and in the derived variable $\log\{\text{dose} + 10\}$ (Lawless 1987). The log of the expected count of revertants is thus

$$\beta_0 + \beta_1 \text{dose} + \beta_2 \log\{\text{dose} + 10\}$$

The following statements fit a standard Poisson regression model to these data:

```
proc hpfmm data=assay;
   model num = dose logd / dist=Poisson;
run;
```

The Pearson statistic for this model is rather large compared to the number of degrees of freedom ($18-3 = 15$). The ratio $46.2707/15 = 3.08$ indicates an overdispersion problem in the Poisson model (Output 6.3.1).

Output 6.3.1 Result of Fitting Poisson Regression Models

The HPFMM Procedure

Number of Observations Read	18
Number of Observations Used	18

Fit Statistics	
-2 Log Likelihood	136.3
AIC (Smaller is Better)	142.3
AICC (Smaller is Better)	144.0
BIC (Smaller is Better)	144.9
Pearson Statistic	46.2707

Parameter Estimates for Poisson Model

Effect	Estimate	Standard Error	z Value	Pr > |z|
Intercept	2.1728	0.2184	9.95	<.0001
dose	-0.00101	0.000245	-4.13	<.0001
logd	0.3198	0.05700	5.61	<.0001

Breslow (1984) accounts for overdispersion by including a random effect in the predictor for the log rate and applying a quasi-likelihood technique to estimate the parameters. Wang et al. (1996) examine these data using mixtures of Poisson regression models. They fit several two- and three-component Poisson regression mixtures. Examining the log likelihoods, AIC, and BIC criteria, they eventually settle on a two-component model in which the intercepts vary by category and the regression coefficients are the same. This mixture model can be written as

$$
\begin{aligned}
f(y) &= \pi \frac{1}{y!}\lambda_1^y \exp\{-\lambda_1\} + (1-\pi)\frac{1}{y!}\lambda_2^y \exp\{-\lambda_2\} \\
\lambda_1 &= \exp\{\beta_{01} + \beta_1 \text{dose} + \beta_2 \log\{\text{dose} + 10\}\} \\
\lambda_2 &= \exp\{\beta_{02} + \beta_1 \text{dose} + \beta_2 \log\{\text{dose} + 10\}\}
\end{aligned}
$$

This model is fit with the HPFMM procedure with the following statements:

```
proc hpfmm data=assay;
   model num = dose logd / dist=Poisson k=2
                           equate=effects(dose logd);
run;
```

The EQUATE= option in the MODEL statement places constraints on the optimization and makes the coefficients for dose and logd homogeneous across components in the model. Output 6.3.2 displays the "Fit Statistics" and parameter estimates in the mixture. The Pearson statistic is drastically reduced compared to the Poisson regression model in Output 6.3.1. With $18 - 5 = 13$ degrees of freedom, the ratio of the Pearson and the degrees of freedom is now $16.1573/13 = 1.2429$. Note that the effective number of parameters was used to compute the degrees of freedom, not the total number of parameters, because of the equality constraints.

Output 6.3.2 Result for Two-Component Poisson Regression Mixture

The HPFMM Procedure

Fit Statistics	
-2 Log Likelihood	121.8
AIC (Smaller is Better)	131.8
AICC (Smaller is Better)	136.8
BIC (Smaller is Better)	136.3
Pearson Statistic	16.1573
Effective Parameters	5
Effective Components	2

Parameter Estimates for Poisson Model

Component	Effect	Estimate	Standard Error	z Value	Pr > |z|
1	Intercept	1.9097	0.2654	7.20	<.0001
1	dose	-0.00126	0.000273	-4.62	<.0001
1	logd	0.3639	0.06602	5.51	<.0001
2	Intercept	2.4770	0.2731	9.07	<.0001
2	dose	-0.00126	0.000273	-4.62	<.0001
2	logd	0.3639	0.06602	5.51	<.0001

Output 6.3.2 *continued*

Parameter Estimates for Mixing Probabilities

Component	Mixing Probability	Linked Scale Logit(Prob)	Standard Error	z Value	Pr > |z|
1	0.8173	1.4984	0.6875	2.18	0.0293
2	0.1827	-1.4984			

You could also have used RESTRICT statements to impose the homogeneity constraints on the model fit, as shown in the following statements:

```
proc hpfmm data=assay;
   model num = dose logd / dist=Poisson k=2;
   restrict 'common dose' dose 1, dose -1;
   restrict 'common logd' logd 1, logd -1;
run;
```

The first RESTRICT statement equates the coefficients for the dose variable in the two components, and the second RESTRICT statement accomplishes the same for the coefficients of the logd variable. If the right-hand side of a restriction is not specified, PROC HPFMM defaults to equating the left-hand side of the restriction to zero. The "Linear Constraints" table in Output 6.3.3 shows that both linear equality constraints are active. The parameter estimates match the previous HPFMM run.

Output 6.3.3 Result for Two-Component Mixture with RESTRICT Statements

The HPFMM Procedure

Linear Constraints at Solution

Label	k = 1	k = 2		Constraint Active
common dose	dose	- dose	= 0	Yes
common logd	logd	- logd	= 0	Yes

Parameter Estimates for Poisson Model

Component	Effect	Estimate	Standard Error	z Value	Pr > |z|
1	Intercept	1.9097	0.2654	7.20	<.0001
1	dose	-0.00126	0.000273	-4.62	<.0001
1	logd	0.3639	0.06602	5.51	<.0001
2	Intercept	2.4770	0.2731	9.07	<.0001
2	dose	-0.00126	0.000273	-4.62	<.0001
2	logd	0.3639	0.06602	5.51	<.0001

Parameter Estimates for Mixing Probabilities

Component	Mixing Probability	Linked Scale Logit(Prob)	Standard Error	z Value	Pr > |z|
1	0.8173	1.4984	0.6875	2.18	0.0293
2	0.1827	-1.4984			

Wang et al. (1996) note that observation 12 with a revertant colony count of 60 is comparably high. The following statements remove the observation from the analysis and fit their selected model:

```
proc hpfmm data=assay(where=(num ne 60));
   model num = dose logd / dist=Poisson k=2
                           equate=effects(dose logd);
run;
```

Output 6.3.4 Result for Two-Component Model without Outlier

The HPFMM Procedure

Fit Statistics	
-2 Log Likelihood	111.5
AIC (Smaller is Better)	121.5
AICC (Smaller is Better)	126.9
BIC (Smaller is Better)	125.6
Pearson Statistic	16.5987
Effective Parameters	5
Effective Components	2

Parameter Estimates for Poisson Model

Component	Effect	Estimate	Standard Error	z Value	Pr > |z|
1	Intercept	2.2272	0.3022	7.37	<.0001
1	dose	-0.00065	0.000445	-1.46	0.1440
1	logd	0.2432	0.1045	2.33	0.0199
2	Intercept	2.5477	0.3331	7.65	<.0001
2	dose	-0.00065	0.000445	-1.46	0.1440
2	logd	0.2432	0.1045	2.33	0.0199

Parameter Estimates for Mixing Probabilities

		Linked Scale			
Component	Mixing Probability	Logit(Prob)	Standard Error	z Value	Pr > |z|
1	0.5777	0.3134	1.7261	0.18	0.8559
2	0.4223	-0.3134			

The ratio of Pearson Statistic over degrees of freedom (12) is only slightly worse than in the previous model; the loss of 5% of the observations carries a price (Output 6.3.4). The parameter estimates for the two intercepts are now fairly close. If the intercepts were identical, then the two-component model would collapse to the Poisson regression model:

```
proc hpfmm data=assay(where=(num ne 60));
   model num = dose logd / dist=Poisson;
run;
```

Output 6.3.5 Result of Fitting Poisson Regression Model without Outlier

The HPFMM Procedure

Number of Observations Read	17
Number of Observations Used	17

Fit Statistics	
-2 Log Likelihood	114.1
AIC (Smaller is Better)	120.1
AICC (Smaller is Better)	121.9
BIC (Smaller is Better)	122.5
Pearson Statistic	27.8008

Parameter Estimates for Poisson Model

Effect	Estimate	Standard Error	z Value	Pr > |z|
Intercept	2.3164	0.2244	10.32	<.0001
dose	-0.00072	0.000258	-2.78	0.0055
logd	0.2603	0.05996	4.34	<.0001

Compared to the same model applied to the full data, the Pearson statistic is much reduced (compare 46.2707 in Output 6.3.1 to 27.8008 in Output 6.3.5). The outlier—or *over*count, if you will—induces at least some of the *over*dispersion.

References

Aldrich, J. (1997), "R. A. Fisher and the Making of Maximum Likelihood, 1912–1922," *Statistical Science*, 12, 162–176.

Breslow, N. E. (1984), "Extra-Poisson Variation in Log-Linear Models," *Applied Statistics*, 33, 38–44.

Cameron, A. C. and Trivedi, P. K. (1998), *Regression Analysis of Count Data*, Cambridge: Cambridge University Press.

Celeux, G., Forbes, F., Robert, C. P., and Titterington, D. M. (2006), "Deviance Information Criteria for Missing Data Models," *Bayesian Analysis*, 1, 651–674.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977), "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society, Series B*, 39, 1–38.

Dennis, J. E., Gay, D. M., and Welsch, R. E. (1981), "An Adaptive Nonlinear Least-Squares Algorithm," *ACM Transactions on Mathematical Software*, 7, 348–368.

Dennis, J. E. and Mei, H. H. W. (1979), "Two New Unconstrained Optimization Algorithms Which Use Function and Gradient Values," *Journal of Optimization Theory and Applications*, 28, 453–482.

Eskow, E. and Schnabel, R. B. (1991), "Algorithm 695: Software for a New Modified Cholesky Factorization," *ACM Transactions on Mathematical Software*, 17, 306–312.

Everitt, B. S. and Hand, D. J. (1981), *Finite Mixture Distributions*, London: Chapman & Hall.

Ferrari, S. L. P. and Cribari-Neto, F. (2004), "Beta Regression for Modelling Rates and Proportions," *Journal of Applied Statistics*, 31, 799–815.

Fisher, R. A. (1921), "On the 'Probable Error' of a Coefficient of Correlation Deduced from a Small Sample," *Metron*, 1, 3–32.

Fletcher, R. (1987), *Practical Methods of Optimization*, 2nd Edition, Chichester, UK: John Wiley & Sons.

Frühwirth-Schnatter, S. (2006), *Finite Mixture and Markov Switching Models*, New York: Springer.

Gamerman, D. (1997), "Sampling from the Posterior Distribution in Generalized Linear Models," *Statistics and Computing*, 7, 57–68.

Gay, D. M. (1983), "Subroutines for Unconstrained Minimization," *ACM Transactions on Mathematical Software*, 9, 503–524.

Geweke, J. (1992), "Evaluating the Accuracy of Sampling-Based Approaches to Calculating Posterior Moments," in J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, eds., *Bayesian Statistics*, volume 4, Oxford: Clarendon Press.

Griffiths, D. A. (1973), "Maximum Likelihood Estimation for the Beta-Binomial Distribution and an Application to the Household Distribution of the Total Number of Cases of a Disease," *Biometrics*, 29, 637–648.

Haseman, J. K. and Kupper, L. L. (1979), "Analysis of Dichotomous Response Data from Certain Toxicological Experiments," *Biometrics*, 35, 281–293.

Joe, H. and Zhu, R. (2005), "Generalized Poisson Distribution: The Property of Mixture of Poisson and Comparison with Negative Binomial Distribution," *Biometrical Journal*, 47, 219–229.

Kass, R. E., Carlin, B. P., Gelman, A., and Neal, R. M. (1998), "Markov Chain Monte Carlo in Practice: A Roundtable Discussion," *American Statistician*, 52, 93–100.

Lawless, J. F. (1987), "Negative Binomial and Mixed Poisson Regression," *Canadian Journal of Statistics*, 15, 209–225.

Margolin, B. H., Kaplan, N. L., and Zeiger, E. (1981), "Statistical Analysis of the Ames Salmonella Microsome Test," *Proceedings of the National Academy of Sciences*, 76, 3779–3783.

McLachlan, G. J. and Peel, D. (2000), *Finite Mixture Models*, New York: John Wiley & Sons.

Moré, J. J. (1978), "The Levenberg-Marquardt Algorithm: Implementation and Theory," in G. A. Watson, ed., *Lecture Notes in Mathematics*, volume 30, 105–116, Berlin: Springer-Verlag.

Moré, J. J. and Sorensen, D. C. (1983), "Computing a Trust-Region Step," *SIAM Journal on Scientific and Statistical Computing*, 4, 553–572.

Morel, J. G. and Nagaraj, N. K. (1993), "A Finite Mixture Distribution for Modelling Multinomial Extra Variation," *Biometrika*, 80, 363–371.

Morel, J. G. and Neerchal, N. K. (1997), "Clustered Binary Logistic Regression in Teratology Data Using a Finite Mixture Distribution," *Statistics in Medicine*, 16, 2843–2853.

Neerchal, N. K. and Morel, J. G. (1998), "Large Cluster Results for Two Parametric Multinomial Extra Variation Models," *Journal of the American Statistical Association*, 93, 1078–1087.

Pearson, K. (1915), "On Certain Types of Compound Frequency Distributions in Which the Components Can Be Individually Described by Binomial Series," *Biometrika*, 11, 139–144.

Raftery, A. E. (1996), "Hypothesis Testing and Model Selection," in W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, eds., *Markov Chain Monte Carlo in Practice*, 163–188, London: Chapman & Hall.

Richardson, S. (2002), "Discussion of Spiegelhalter et al." *Journal of the Royal Statistical Society, Series B*, 64, 631.

Roeder, K. (1990), "Density Estimation with Confidence Sets Exemplified by Superclusters and Voids in the Galaxies," *Journal of the American Statistical Association*, 85, 617–624.

Spiegelhalter, D. J., Best, N. G., Carlin, B. P., and Van der Linde, A. (2002), "Bayesian Measures of Model Complexity and Fit," *Journal of the Royal Statistical Society, Series B*, 64(4), 583–616, with discussion.

Titterington, D. M., Smith, A. F. M., and Makov, U. E. (1985), *Statistical Analysis of Finite Mixture Distributions*, New York: John Wiley & Sons.

Viallefont, V., Richardson, S., and Greene, P. J. (2002), "Bayesian Analysis of Poisson Mixtures," *Journal of Nonparametric Statistics*, 14, 181–202.

Wang, P., Puterman, M. L., Cockburn, I., and Le, N. (1996), "Mixed Poisson Regression Models with Covariate Dependent Rates," *Biometrics*, 52, 381–400.

Williams, D. A. (1975), "The Analysis of Binary Responses from Toxicological Experiments Involving Reproduction and Teratogenicity," *Biometrics*, 31, 949–952.

Chapter 7
The HPGENSELECT Procedure

Contents

Overview: HPGENSELECT Procedure **224**
PROC HPGENSELECT Features 224
PROC HPGENSELECT Contrasted with PROC GENMOD 225
Getting Started: HPGENSELECT Procedure **225**
Syntax: HPGENSELECT Procedure **231**
PROC HPGENSELECT Statement 231
CLASS Statement 237
CODE Statement 237
FREQ Statement 238
ID Statement 238
MODEL Statement 238
OUTPUT Statement 245
PARTITION Statement 248
PERFORMANCE Statement 248
SELECTION Statement 249
WEIGHT Statement 250
ZEROMODEL Statement 250
Details: HPGENSELECT Procedure **251**
Missing Values 251
Exponential Family Distributions 252
Response Distributions 252
Response Probability Distribution Functions 254
Log-Likelihood Functions 257
Using Validation and Test Data 262
Computational Method: Multithreading 262
Choosing an Optimization Algorithm 264
First- or Second-Order Algorithms 264
Algorithm Descriptions 265
Displayed Output 267
ODS Table Names 271
Examples: HPGENSELECT Procedure **273**
Example 7.1: Model Selection 273
Example 7.2: Modeling Binomial Data 276
Example 7.3: Tweedie Model 280
References **283**

Overview: HPGENSELECT Procedure

The HPGENSELECT procedure is a high-performance procedure that provides model fitting and model building for generalized linear models. It fits models for standard distributions in the exponential family, such as the normal, Poisson, and Tweedie distributions. In addition, PROC HPGENSELECT fits multinomial models for ordinal and nominal responses, and it fits zero-inflated Poisson and negative binomial models for count data. For all these models, the HPGENSELECT procedure provides forward, backward, and stepwise variable selection.

PROC HPGENSELECT runs in either single-machine mode or distributed mode.

NOTE: Distributed mode requires SAS High-Performance Statistics.

PROC HPGENSELECT Features

The HPGENSELECT procedure does the following:

- estimates the parameters of a generalized linear regression model by using maximum likelihood techniques
- provides model-building syntax in the CLASS statement and the effect-based MODEL statement, which are familiar from SAS/STAT procedures (in particular, the GLM, GENMOD, LOGISTIC, GLIMMIX, and MIXED procedures)
- enables you to split classification effects into individual components by using the SPLIT option in the CLASS statement
- permits any degree of interaction effects that involve classification and continuous variables
- provides multiple link functions
- provides models for zero-inflated count data
- provides cumulative link modeling for ordinal data and generalized logit modeling for unordered multinomial data
- enables model building (variable selection) through the SELECTION statement
- provides a WEIGHT statement for weighted analysis
- provides a FREQ statement for grouped analysis
- provides an OUTPUT statement to produce a data set that has predicted values and other observation-wise statistics

Because the HPGENSELECT procedure is a high-performance analytical procedure, it also does the following:

- enables you to run in distributed mode on a cluster of machines that distribute the data and the computations
- enables you to run in single-machine mode on the server where SAS is installed
- exploits all the available cores and concurrent threads, regardless of execution mode

For more information, see the section "Processing Modes" on page 10 in Chapter 3, "Shared Concepts and Topics."

PROC HPGENSELECT Contrasted with PROC GENMOD

This section contrasts the HPGENSELECT procedure with the GENMOD procedure in SAS/STAT software.

The CLASS statement in the HPGENSELECT procedure permits two parameterizations: GLM parameterization and a reference parameterization. In contrast to the LOGISTIC, GENMOD, and other procedures that permit multiple parameterizations, the HPGENSELECT procedure does not mix parameterizations across the variables in the CLASS statement. In other words, all classification variables have the same parameterization, and this parameterization is either GLM parameterization or reference parameterization. The CLASS statement also enables you to split an effect that involves a classification variable into multiple effects that correspond to individual levels of the classification variable.

The default optimization technique used by the HPGENSELECT procedure is a modification of the Newton-Raphson algorithm with a ridged Hessian. You can choose different optimization techniques (including first-order methods that do not require a crossproducts matrix or Hessian) by specifying the TECHNIQUE= option in the PROC HPGENSELECT statement.

As in the GENMOD procedure, the default parameterization of CLASS variables in the HPGENSELECT procedure is GLM parameterization. You can change the parameterization by specifying the PARAM= option in the CLASS statement.

The GENMOD procedure offers a wide variety of postfitting analyses, such as contrasts, estimates, tests of model effects, and least squares means. The HPGENSELECT procedure is limited in postfitting functionality because it is primarily designed for large-data tasks, such as predictive model building, model fitting, and scoring.

Getting Started: HPGENSELECT Procedure

This example illustrates how you can use PROC HPGENSELECT to perform Poisson regression for count data. The following DATA step contains 100 observations for a count response variable (Y), a continuous variable (Total) to be used in a later analysis, and five categorical variables (C1–C5), each of which has four numerical levels:

```
data getStarted;
  input C1-C5 Y Total;
  datalines;
0 3 1 1 3 2 28.361
2 3 0 3 1 2 39.831
1 3 2 2 2 1 17.133
1 2 0 0 3 2 12.769
0 2 1 0 1 1 29.464
0 2 1 0 2 1 4.152
1 2 1 0 1 0 0.000
0 2 1 1 2 1 20.199
1 2 0 0 1 0 0.000
0 1 1 3 3 2 53.376
2 2 2 2 1 1 31.923
0 3 2 0 3 2 37.987
2 2 2 0 0 1 1.082
0 2 0 2 0 1 6.323
1 3 0 0 0 0 0.000
1 2 1 2 3 2 4.217
0 1 2 3 1 1 26.084
1 1 0 0 1 0 0.000
1 3 2 2 2 0 0.000
2 1 3 1 1 2 52.640
1 3 0 1 2 1 3.257
2 0 2 3 0 5 88.066
2 2 2 1 0 1 15.196
3 1 3 1 0 1 11.955
3 1 3 1 2 3 91.790
3 1 1 2 3 7 232.417
3 1 1 1 0 1 2.124
3 1 0 0 0 2 32.762
3 1 2 3 0 1 25.415
2 2 0 1 2 1 42.753
3 3 2 2 3 1 23.854
2 0 0 2 3 2 49.438
1 0 0 2 3 4 105.449
0 0 2 3 0 6 101.536
0 3 1 0 0 0 0.000
3 0 1 0 1 1 5.937
2 0 0 0 3 2 53.952
1 0 1 0 3 2 23.686
1 1 3 1 1 1 0.287
2 1 3 0 3 7 281.551
1 3 2 1 1 0 0.000
2 1 0 0 1 0 0.000
0 0 1 1 2 3 93.009
0 1 0 1 0 2 25.055
1 2 2 2 3 1 1.691
0 3 2 3 1 1 10.719
3 3 0 3 3 1 19.279
2 0 0 2 1 2 40.802
2 2 3 0 3 3 72.924
0 2 0 3 0 1 10.216
```

```
3 0 1 2 2 2 87.773
2 1 2 3 1 0 0.000
3 2 0 3 1 0 0.000
3 0 3 0 0 2 62.016
1 3 2 2 1 3 36.355
2 3 2 0 3 1 23.190
1 0 1 2 1 1 11.784
2 1 2 2 2 5 204.527
3 0 1 1 2 5 115.937
0 1 1 3 2 1 44.028
2 2 1 3 1 4 52.247
1 1 0 0 1 1 17.621
3 3 1 2 1 2 10.706
2 2 0 2 3 3 81.506
0 1 0 0 2 2 81.835
0 1 2 0 1 2 20.647
3 2 2 2 0 1 3.110
2 2 3 0 0 1 13.679
1 2 2 3 2 1 6.486
3 3 2 2 1 2 30.025
0 0 3 1 3 6 202.172
3 2 3 1 2 3 44.221
0 3 0 0 0 1 27.645
3 3 3 0 3 2 22.470
2 3 2 0 2 0 0.000
1 3 0 2 0 1 1.628
1 3 1 0 2 0 0.000
3 2 3 3 0 1 20.684
3 1 0 2 0 4 108.000
0 1 2 2 1 1 4.615
0 2 3 2 2 1 12.461
0 3 2 0 1 3 53.798
2 1 1 2 0 1 36.320
1 0 3 0 0 0 0.000
0 0 3 2 0 1 19.902
0 2 3 1 0 0 0.000
2 2 2 1 3 2 31.815
3 3 3 0 0 0 0.000
2 2 1 3 3 2 17.915
0 2 3 2 3 2 69.315
1 3 1 2 1 0 0.000
3 0 1 1 1 4 94.050
2 1 1 1 3 6 242.266
0 2 0 3 2 1 40.885
2 0 1 1 2 2 74.708
2 2 2 2 3 2 50.734
1 0 2 2 1 3 35.950
1 3 3 1 1 1 2.777
3 1 2 1 3 5 118.065
0 3 2 1 2 0 0.000
;
```

The following statements fit a log-linked Poisson model to these data by using classification effects for variables C1–C5:

```
proc hpgenselect data=getStarted;
   class C1-C5;
   model Y = C1-C5 /  Distribution=Poisson Link=Log;
run;
```

The default output from this analysis is presented in Figure 7.1 through Figure 7.8.

The "Performance Information" table in Figure 7.1 shows that the procedure executed in single-machine mode (that is, on the server where SAS is installed). When high-performance procedures run in single-machine mode, they use concurrently scheduled threads. In this case, four threads were used.

Figure 7.1 Performance Information

The HPGENSELECT Procedure

Performance Information	
Execution Mode	Single-Machine
Number of Threads	4

Figure 7.2 displays the "Model Information" table. The variable Y is an integer-valued variable that is modeled by using a Poisson probability distribution, and the mean of Y is modeled by using a log link function. The HPGENSELECT procedure uses a Newton-Raphson algorithm to fit the model. The CLASS variables C1–C5 are parameterized by using GLM parameterization, which is the default.

Figure 7.2 Model Information

Model Information	
Data Source	WORK.GETSTARTED
Response Variable	Y
Class Parameterization	GLM
Distribution	Poisson
Link Function	Log
Optimization Technique	Newton-Raphson with Ridging

Each of the CLASS variables C1–C5 has four unique formatted levels, which are displayed in the "Class Level Information" table in Figure 7.3.

Figure 7.3 Class Level Information

Class Level Information		
Class	**Levels**	**Values**
C1	4	0 1 2 3
C2	4	0 1 2 3
C3	4	0 1 2 3
C4	4	0 1 2 3
C5	4	0 1 2 3

Figure 7.4 displays the "Number of Observations" table. All 100 observations in the data set are used in the analysis.

Figure 7.4 Number of Observations

Number of Observations Read	100
Number of Observations Used	100

Figure 7.5 displays the "Dimensions" table for this model. This table summarizes some important sizes of various model components. For example, it shows that there are 21 columns in the design matrix $\mathbf{X}$: one column for the intercept and 20 columns for the effects that are associated with the classification variables C1–C5. However, the rank of the crossproducts matrix is only 16. Because the classification variables C1–C5 use GLM parameterization and because the model contains an intercept, there is one singularity in the crossproducts matrix of the model for each classification variable. Consequently, only 16 parameters enter the optimization.

Figure 7.5 Dimensions in Poisson Regression

Dimensions	
Number of Effects	6
Number of Parameters	16
Columns in X	21

Figure 7.6 displays the final convergence status of the Newton-Raphson algorithm. The GCONV= relative convergence criterion is satisfied.

Figure 7.6 Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

The "Fit Statistics" table is shown in Figure 7.7. The –2 log likelihood at the converged estimates is 290.16169. You can use this value to compare the model to nested model alternatives by means of a likelihood-ratio test. To compare models that are not nested, information criteria such as AIC (Akaike's information criterion), AICC (Akaike's bias-corrected information criterion), and BIC (Schwarz Bayesian information criterion) are used. These criteria penalize the –2 log likelihood for the number of parameters.

Figure 7.7 Fit Statistics

Fit Statistics	
-2 Log Likelihood	290.16
AIC (smaller is better)	322.16
AICC (smaller is better)	328.72
BIC (smaller is better)	363.84
Pearson Chi-Square	77.7694
Pearson Chi-Square/DF	0.9258

The "Parameter Estimates" table in Figure 7.8 shows that many parameters have fairly large *p*-values, indicating that one or more of the model effects might not be necessary.

Figure 7.8 Parameter Estimates

Parameter Estimates					
Parameter	**DF**	**Estimate**	**Standard Error**	**Chi-Square**	**Pr > ChiSq**
Intercept	1	0.881903	0.382730	5.3095	0.0212
C1 0	1	-0.196002	0.211482	0.8590	0.3540
C1 1	1	-0.605161	0.263508	5.2742	0.0216
C1 2	1	-0.068458	0.210776	0.1055	0.7453
C1 3	0	0	.	.	.
C2 0	1	0.961117	0.255485	14.1521	0.0002
C2 1	1	0.708188	0.246768	8.2360	0.0041
C2 2	1	0.161741	0.266365	0.3687	0.5437
C2 3	0	0	.	.	.
C3 0	1	-0.227016	0.252561	0.8079	0.3687
C3 1	1	-0.094775	0.229519	0.1705	0.6797
C3 2	1	0.044801	0.238127	0.0354	0.8508
C3 3	0	0	.	.	.
C4 0	1	-0.280476	0.263589	1.1322	0.2873
C4 1	1	0.028157	0.249652	0.0127	0.9102
C4 2	1	0.047803	0.240378	0.0395	0.8424
C4 3	0	0	.	.	.
C5 0	1	-0.817936	0.219901	13.8351	0.0002
C5 1	1	-0.710596	0.206265	11.8684	0.0006
C5 2	1	-0.602080	0.217724	7.6471	0.0057
C5 3	0	0	.	.	.

Syntax: HPGENSELECT Procedure

The following statements are available in the HPGENSELECT procedure:

PROC HPGENSELECT < *options* > **;**
CLASS *variable* < **(***options***)** >. . . < *variable* < **(***options***)** > > < / *global-options* > **;**
CODE < *options* > **;**
MODEL *response*< **(***response-options***)** > **=** < *effects* > < / *model-options* > **;**
MODEL *events/trials*< **(***response-options***)** > **=** < *effects* > < / *model-options* > **;**
OUTPUT < **OUT=***SAS-data-set* >
< *keyword* < **=***name* > >. . .
< *keyword* < **=***name* > > < / *options* > **;**
PARTITION < *partition-options* > **;**
PERFORMANCE *performance-options* **;**
SELECTION *selection-options* **;**
FREQ *variable* **;**
ID *variables* **;**
WEIGHT *variable* **;**
ZEROMODEL < *effects* >< / *zeromodel-options* > **;**

The PROC HPGENSELECT statement and at least one MODEL statement are required. The CLASS statement can appear multiple times. If a CLASS statement is specified, it must precede the MODEL statements.

PROC HPGENSELECT Statement

PROC HPGENSELECT < *options* > **;**

The PROC HPGENSELECT statement invokes the procedure. Table 7.1 summarizes the available options in the PROC HPGENSELECT statement by function. The options are then described fully in alphabetical order.

Table 7.1 PROC HPGENSELECT Statement Options

Option	Description
Basic Options	
ALPHA=	Specifies a global significance level
DATA=	Specifies the input data set
NAMELEN=	Limits the length of effect names

Table 7.1 *continued*

Option	Description
Output Options	
CORR	Displays the "Parameter Estimates Correlation Matrix" table
COV	Displays the "Parameter Estimates Covariance Matrix" table
ITDETAILS	Displays the "Iteration History" table when no model selection is performed
ITSELECT	Displays the summarized "Iteration History" table when model selection is performed
ITSUMMARY	Displays the summarized "Iteration History" table when no model selection is performed
NOPRINT	Suppresses ODS output
NOCLPRINT	Limits or suppresses the display of classification variable levels
NOSTDERR	Suppresses computation of the covariance matrix and standard errors
Optimization Options	
ABSCONV=	Tunes the absolute function convergence criterion
ABSFCONV=	Tunes the absolute function difference convergence criterion
ABSGCONV=	Tunes the absolute gradient convergence criterion
FCONV=	Tunes the relative function difference convergence criterion
GCONV=	Tunes the relative gradient convergence criterion
MAXITER=	Chooses the maximum number of iterations in any optimization
MAXFUNC=	Specifies the maximum number of function evaluations in any optimization
MAXTIME=	Specifies the upper limit of CPU time (in seconds) for any optimization
MINITER=	Specifies the minimum number of iterations in any optimization
NORMALIZE=	Specifies whether the objective function is normalized during optimization
TECHNIQUE=	Selects the optimization technique
Tolerance Options	
SINGCHOL=	Tunes the singularity criterion for Cholesky decompositions
SINGSWEEP=	Tunes the singularity criterion for the sweep operator
SINGULAR=	Tunes the general singularity criterion
User-Defined Format Options	
FMTLIBXML=	Specifies the file reference for a format stream

You can specify the following *options* in the PROC HPGENSELECT statement.

ABSCONV=*r*

ABSTOL=*r*

specifies an absolute function convergence criterion. For minimization, termination requires $f(\boldsymbol{\psi}^{(k)}) \leq r$, where $\boldsymbol{\psi}$ is the vector of parameters in the optimization and $f(\cdot)$ is the objective function. The default value of r is the negative square root of the largest double-precision value, which serves only as a protection against overflow.

ABSFCONV=*r* **<** *n* **>**

ABSFTOL=*r* **<** *n* **>**

specifies an absolute function difference convergence criterion. For all techniques except NMSIMP, termination requires a small change of the function value in successive iterations:

$$|f(\boldsymbol{\psi}^{(k-1)}) - f(\boldsymbol{\psi}^{(k)})| \leq r$$

Here, $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization, and $f(\cdot)$ is the objective function. The same formula is used for the NMSIMP technique, but $\boldsymbol{\psi}(k)$ is defined as the vertex that has the lowest function value and $\boldsymbol{\psi}^{(k-1)}$ is defined as the vertex that has the highest function value in the simplex. The default value is $r = 0$. The optional integer value *n* specifies the number of successive iterations for which the criterion must be satisfied before the process can be terminated.

ABSGCONV=*r* **<** *n* **>**

ABSGTOL=*r* **<** *n* **>**

specifies an absolute gradient convergence criterion. Termination requires the maximum absolute gradient element to be small:

$$\max_j |g_j(\boldsymbol{\psi}^{(k)})| \leq r$$

Here, $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization, and $g_j(\cdot)$ is the gradient of the objective function with respect to the *j*th parameter. This criterion is not used by the NMSIMP technique. The default value is r = 1E–8. The optional integer value *n* specifies the number of successive iterations for which the criterion must be satisfied before the process can be terminated.

ALPHA=*number*

specifies a global significance level for the construction of confidence intervals. The confidence level is 1 – *number*. The value of *number* must be between 0 and 1; the default is 0.05. You can override this global significance level by specifying the ALPHA= option in the MODEL statement or the ALPHA= option in the OUTPUT statement.

CORR

creates the "Parameter Estimates Correlation Matrix" table. The correlation matrix is computed by normalizing the covariance matrix $\boldsymbol{\Sigma}$. That is, if σ_{ij} is an element of $\boldsymbol{\Sigma}$, then the corresponding element of the correlation matrix is $\sigma_{ij}/\sigma_i\sigma_j$, where $\sigma_i = \sqrt{\sigma_{ii}}$.

COV

creates the "Parameter Estimates Covariance Matrix" table. The covariance matrix is computed as the inverse of the negative of the matrix of second derivatives of the log-likelihood function with respect to the model parameters (the Hessian matrix).

DATA=*SAS-data-set*

names the input SAS data set for PROC HPGENSELECT to use. The default is the most recently created data set.

If the procedure executes in distributed mode, the input data are distributed to memory on the appliance nodes and analyzed in parallel, unless the data are already distributed in the appliance database. In that case the procedure reads the data alongside the distributed database. For information about the various execution modes, see the section "Processing Modes" on page 10; for information about the alongside-the-database model, see the section "Alongside-the-Database Execution" on page 18.

FCONV=*r* **<** *n* **>**

FTOL=*r* **<** *n* **>**

specifies a relative function difference convergence criterion. For all techniques except NMSIMP, termination requires a small relative change of the function value in successive iterations:

$$\frac{|f(\boldsymbol{\psi}^{(k)}) - f(\boldsymbol{\psi}^{(k-1)})|}{|f(\boldsymbol{\psi}^{(k-1)})|} \leq r$$

Here, $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization, and $f(\cdot)$ is the objective function. The same formula is used for the NMSIMP technique, but $\boldsymbol{\psi}^{(k)}$ is defined as the vertex that has the lowest function value, and $\boldsymbol{\psi}^{(k-1)}$ is defined as the vertex that has the highest function value in the simplex.

The default value is $r = 2 \times \epsilon$, where ϵ is the machine precision. The optional integer value *n* specifies the number of successive iterations for which the criterion must be satisfied before the process can terminate.

FMTLIBXML=*file-ref*

specifies the file reference for the XML stream that contains the user-defined format definitions. User-defined formats are handled differently in a distributed computing environment than they are in other SAS products. For information about how to generate an XML stream for your formats, see the section "Working with Formats" on page 35 in Chapter 3, "Shared Concepts and Topics."

GCONV=*r* **<** *n* **>**

GTOL=*r* **<** *n* **>**

specifies a relative gradient convergence criterion. For all techniques except CONGRA and NMSIMP, termination requires that the normalized predicted function reduction be small:

$$\frac{\mathbf{g}(\boldsymbol{\psi}^{(k)})'[\mathbf{H}^{(k)}]^{-1}\mathbf{g}(\boldsymbol{\psi}^{(k)})}{|f(\boldsymbol{\psi}^{(k)})|} \leq r$$

Here, $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization, $f(\cdot)$ is the objective function, and $\mathbf{g}(\cdot)$ is the gradient. For the CONGRA technique (where a reliable Hessian estimate $\mathbf{H}$ is not available), the following criterion is used:

$$\frac{\| \mathbf{g}(\boldsymbol{\psi}^{(k)}) \|_2^2 \quad \| \mathbf{s}(\boldsymbol{\psi}^{(k)}) \|_2}{\| \mathbf{g}(\boldsymbol{\psi}^{(k)}) - \mathbf{g}(\boldsymbol{\psi}^{(k-1)}) \|_2 \; |f(\boldsymbol{\psi}^{(k)})|} \leq r$$

This criterion is not used by the NMSIMP technique. The default value is *r*=1E–8. The optional integer value *n* specifies the number of successive iterations for which the criterion must be satisfied before the process can terminate.

ITDETAILS

adds to the "Iteration History" table the current values of the parameter estimates and their gradients. These quantities are reported only for parameters that participate in the optimization. This option is not available when you perform model selection.

ITSELECT
generates the "Iteration History" table when you perform a model selection.

ITSUMMARY
generates the "Iteration History" table. This option is not available when you perform model selection.

MAXFUNC=*n*

MAXFU=*n*
specifies the maximum number of function calls in the optimization process. The default values are as follows, depending on the optimization technique:

- TRUREG, NRRIDG, NEWRAP: $n = 125$
- QUANEW, DBLDOG: $n = 500$
- CONGRA: $n = 1{,}000$
- NMSIMP: $n = 3{,}000$

The optimization can terminate only after completing a full iteration. Therefore, the number of function calls that are actually performed can exceed *n*. You can choose the optimization technique by specifying the TECHNIQUE= option.

MAXITER=*n*

MAXIT=*n*
specifies the maximum number of iterations in the optimization process. The default values are as follows, depending on the optimization technique:

- TRUREG, NRRIDG, NEWRAP: $n = 50$
- QUANEW, DBLDOG: $n = 200$
- CONGRA: $n = 400$
- NMSIMP: $n = 1{,}000$

These default values also apply when *n* is specified as a missing value. You can choose the optimization technique by specifying the TECHNIQUE= option.

MAXTIME=*r*
specifies an upper limit of *r* seconds of CPU time for the optimization process. The default value is the largest floating-point double representation of your computer. The time specified by this option is checked only once at the end of each iteration. Therefore, the actual running time can be longer than *r*.

MINITER=*n*

MINIT=*n*
specifies the minimum number of iterations. The default value is 0. If you request more iterations than are actually needed for convergence to a stationary point, the optimization algorithms might behave strangely. For example, the effect of rounding errors can prevent the algorithm from continuing for the required number of iterations.

NAMELEN=*number*
specifies the length to which long effect names are shortened. The default and minimum value is 20.

NOCLPRINT< =*number* **>**
suppresses the display of the "Class Level Information" table if you do not specify *number*. If you specify *number*, the values of the classification variables are displayed for only those variables whose number of levels is less than *number*. Specifying a *number* helps to reduce the size of the "Class Level Information" table if some classification variables have a large number of levels.

NOPRINT
suppresses the generation of ODS output.

NORMALIZE=YES | NO
specifies whether to normalize the objective function during optimization by the reciprocal of the frequency count of observations that are used in the analysis. This option affects the values that are reported in the "Iteration History" table. The results that are reported in the "Fit Statistics" are always displayed for the nonnormalized log-likelihood function. By default, NORMALIZE = NO.

NOSTDERR
suppresses the computation of the covariance matrix and the standard errors of the regression coefficients. When the model contains many variables (thousands), the inversion of the Hessian matrix to derive the covariance matrix and the standard errors of the regression coefficients can be time-consuming.

SINGCHOL=*number*
tunes the singularity criterion in Cholesky decompositions. The default is 1E4 times the machine epsilon; this product is approximately 1E–12 on most computers.

SINGSWEEP=*number*
tunes the singularity criterion for sweep operations. The default is 1E4 times the machine epsilon; this product is approximately 1E–12 on most computers.

SINGULAR=*number*
tunes the general singularity criterion that is applied in sweeps and inversions. The default is 1E4 times the machine epsilon; this product is approximately 1E–12 on most computers.

TECHNIQUE=*keyword*
TECH=*keyword*
specifies the optimization technique for obtaining maximum likelihood estimates. You can choose from the following techniques by specifying the appropriate *keyword*:

CONGRA	performs a conjugate-gradient optimization.
DBLDOG	performs a version of double-dogleg optimization.
NEWRAP	performs a Newton-Raphson optimization with line search.
NMSIMP	performs a Nelder-Mead simplex optimization.
NONE	performs no optimization.
NRRIDG	performs a Newton-Raphson optimization with ridging.
QUANEW	performs a dual quasi-Newton optimization.

TRUREG performs a trust-region optimization

The default value is TECHNIQUE=NRRIDG, except for the Tweedie distribution, for which the default value is TECHNIQUE=QUANEW.

For more information, see the section "Choosing an Optimization Algorithm" on page 264.

CLASS Statement

CLASS *variable* < **(***options***)** >. . . < *variable* < **(***options***)** > > < / *global-options* > **;**

The CLASS statement names the classification variables to be used as explanatory variables in the analysis. The CLASS statement must precede the MODEL statement. You can list the response variable for binary and multinomial models in the CLASS statement, but this is not necessary.

The CLASS statement is documented in the section "CLASS Statement" on page 42 of Chapter 4, "Shared Statistical Concepts."

The HPGENSELECT procedure additionally supports the following *global-option* in the CLASS statement:

UPCASE
uppercases the values of character-valued CLASS variables before levelizing them. For example, if the UPCASE option is in effect and a CLASS variable can take the values 'a', 'A', and 'b', then 'a' and 'A' represent the same level and the CLASS variable is treated as having only two values: 'A' and 'B'.

CODE Statement

CODE < *options* > **;**

The CODE statement writes SAS DATA step code for computing predicted values of the fitted model either to a file or to a catalog entry. This code can then be included in a DATA step to score new data.

Table 7.2 summarizes the *options* available in the CODE statement.

Table 7.2 CODE Statement Options

Option	Description
CATALOG=	Names the catalog entry where the generated code is saved
DUMMIES	Retains the dummy variables in the data set
ERROR	Computes the error function
FILE=	Names the file where the generated code is saved
FORMAT=	Specifies the numeric format for the regression coefficients
GROUP=	Specifies the group identifier for array names and statement labels
IMPUTE	Imputes predicted values for observations with missing or invalid covariates
LINESIZE=	Specifies the line size of the generated code
LOOKUP=	Specifies the algorithm for looking up CLASS levels
RESIDUAL	Computes residuals

For details about the syntax of the CODE statement, see the section "CODE Statement" (Chapter 19, *SAS/STAT User's Guide*) in Chapter 19, "Shared Concepts and Topics" (*SAS/STAT User's Guide*).

FREQ Statement

FREQ *variable* **;**

The *variable* in the FREQ statement identifies a numeric variable in the data set that contains the frequency of occurrence for each observation. PROC HPGENSELECT treats each observation as if it appeared f times, where the frequency value f is the value of the FREQ variable for the observation. If f is not an integer, then f is truncated to an integer. If f is less than 1 or missing, the observation is not used in the analysis. When the FREQ statement is not specified, each observation is assigned a frequency of 1.

ID Statement

ID *variables* **;**

The ID statement lists one or more variables from the input data set that are to be transferred to the output data set that is specified in the OUTPUT statement.

For more information, see the section "ID Statement" on page 46 in Chapter 4, "Shared Statistical Concepts."

MODEL Statement

MODEL *response* < **(***response-options***)** > **=** < *effects* > < / *model-options* > **;**

MODEL *events* / *trials* **=** < *effects* > < / *model-options* > **;**

The MODEL statement defines the statistical model in terms of a *response* variable (the target) or an *events/trials* specification. You can also specify model effects that are constructed from variables in the input data set, and you can specify options. An intercept is included in the model by default. You can remove the intercept by specifying the NOINT option.

You can specify a single *response* variable that contains your interval, binary, ordinal, or nominal response values. When you have binomial data, you can specify the *events/trials* form of the response, where one variable contains the number of positive responses (or events) and another variable contains the number of trials. The values of both *events* and (*trials* – *events*) must be nonnegative, and the value of *trials* must be positive. If you specify a single *response* variable that is in a CLASS statement, then the response is assumed to be either binary or multinomial, depending on the number of levels.

For information about constructing the model effects, see the section "Specification and Parameterization of Model Effects" on page 55 of Chapter 4, "Shared Statistical Concepts."

There are two sets of options in the MODEL statement. The *response-options* determine how the HPGENSELECT procedure models probabilities for binary and multinomial data. The *model-options* control other aspects of model formation and inference. Table 7.3 summarizes these options.

Table 7.3 MODEL Statement Options

Option	Description
Response Variable Options for Binary and Multinomial Models	
DESCENDING	Reverses the response categories
EVENT=	Specifies the event category
ORDER=	Specifies the sort order
REF=	Specifies the reference category
Model Options	
ALPHA=	Specifies the confidence level for confidence limits
CL	Requests confidence limits
DISPERSION | PHI=	Specifies a fixed dispersion parameter
DISTRIBUTION | DIST=	Specifies the response distribution
INCLUDE=	Includes effects in all models for model selection
INITIALPHI=	Specifies a starting value of the dispersion parameter
LINK=	Specifies the link function
NOCENTER	Requests that continuous main effects not be centered and scaled
NOINT	Suppresses the intercept
OFFSET=	Specifies the offset variable
SAMPLEFRAC=	Specifies the fraction of the data to be used to compute starting values for the Tweedie distribution
START=	Includes effects in the initial model for model selection

Response Variable Options

Response variable options determine how the HPGENSELECT procedure models probabilities for binary and multinomial data.

You can specify the following *response-options* by enclosing them in parentheses after the *response* or *trials* variable.

DESCENDING

DESC

reverses the order of the response categories. If both the DESCENDING and ORDER= options are specified, PROC HPGENSELECT orders the response categories according to the ORDER= option and then reverses that order.

EVENT='*category***' | FIRST | LAST**

specifies the event category for the binary response model. PROC HPGENSELECT models the probability of the event category. The EVENT= option has no effect when there are more than two response categories.

You can specify the event *category* (formatted, if a format is applied) in quotes, or you can specify one of the following:

FIRST
designates the first ordered category as the event. This is the default.

LAST
designates the last ordered category as the event.

For example, the following statements specify that observations that have a formatted value of '1' represent events in the data. The probability modeled by the HPGENSELECT procedure is thus the probability that the variable def takes on the (formatted) value '1'.

```
proc hpgenselect data=MyData;
   class A B C;
   model def(event ='1') = A B C x1 x2 x3;
run;
```

ORDER=DATA | FORMATTED | INTERNAL

ORDER=FREQ | FREQDATA | FREQFORMATTED | FREQINTERNAL
specifies the sort order for the levels of the *response* variable. When ORDER=FORMATTED (the default) for numeric variables for which you have supplied no explicit format (that is, for which there is no corresponding FORMAT statement in the current PROC HPGENSELECT run or in the DATA step that created the data set), the levels are ordered by their internal (numeric) value. Table 7.4 shows the interpretation of the ORDER= option.

Table 7.4 Sort Order

ORDER=	Levels Sorted By
DATA	Order of appearance in the input data set
FORMATTED	External formatted value, except for numeric variables that have no explicit format, which are sorted by their unformatted (internal) value
FREQ	Descending frequency count (levels that have the most observations come first in the order)
FREQDATA	Order of descending frequency count; within counts by order of appearance in the input data set when counts are tied
FREQFORMATTED	Order of descending frequency count; within counts by formatted value when counts are tied
FREQINTERNAL	Order of descending frequency count; within counts by unformatted value when counts are tied
INTERNAL	Unformatted value

By default, ORDER=FORMATTED. For the FORMATTED and INTERNAL orders, the sort order is machine-dependent.

For more information about sort order, see the chapter about the SORT procedure in *Base SAS Procedures Guide* and the discussion of BY-group processing in *SAS Language Reference: Concepts*.

REF='*category***' | FIRST | LAST**

specifies the reference category for the generalized logit model and the binary response model. For the generalized logit model, each logit contrasts a nonreference category with the reference category. For the binary response model, specifying one response category as the reference is the same as specifying the other response category as the event category. You can specify the reference *category* (formatted if a format is applied) in quotes, or you can specify one of the following:

FIRST

designates the first ordered category as the reference

LAST

designates the last ordered category as the reference. This is the default.

Model Options

ALPHA=*number*

requests that confidence intervals for each of the parameters that are requested by the CL option be constructed with confidence level 1–*number*. The value of *number* must be between 0 and 1; the default is 0.05.

CL

requests that confidence limits be constructed for each of the parameter estimates. The confidence level is 0.95 by default; this can be changed by specifying the ALPHA= option.

DISPERSION=*number*

specifies a fixed dispersion parameter for those distributions that have a dispersion parameter. The dispersion parameter used in all computations is fixed at *number*, and not estimated.

DISTRIBUTION=*keyword*

specifies the response distribution for the model. The *keywords* and the associated distributions are shown in Table 7.5.

Table 7.5 Built-In Distribution Functions

DISTRIBUTION=	Distribution Function
BINARY	Binary
BINOMIAL	Binary or binomial
GAMMA	Gamma
INVERSEGAUSSIAN | IG	Inverse Gaussian
MULTINOMIAL | MULT	Multinomial
NEGATIVEBINOMIAL | NB	Negative binomial
NORMAL | GAUSSIAN	Normal
POISSON	Poisson
TWEEDIE< (*Tweedie-options*) >	Tweedie
ZINB	Zero-inflated negative binomial
ZIP	Zero-inflated Poisson

When DISTRIBUTION=TWEEDIE, you can specify the following *Tweedie-options*:

INITIALP=
specifies a starting value for iterative estimation of the Tweedie power parameter.

OPTMETHOD=*Tweedie-optimization-option*
requests an optimization method for iterative estimation of the Tweedie model parameters. You can specify the following *Tweedie-optimization-options*:

EQL
requests that extended quasi-likelihood be used for a sample of the data, followed by extended quasi-likelihood for the full data. This is equivalent to the TWEEDIEEQL *Tweedie-option*.

EQLLHOOD
requests that extended quasi-likelihood be used for a sample of the data, followed by Tweedie log likelihood for the full data. This is the default method.

FINALLHOOD
requests a four-stage approach to estimating the Tweedie model parameters. The four stages are as follows:

1. extended quasi-likelihood for a sample of the data
2. Tweedie log likelihood for a sample of the data
3. extended quasi-likelihood for the full data
4. Tweedie log likelihood for the full data

LHOOD
requests that Tweedie log likelihood be used for a sample of the data, followed by Tweedie log likelihood for the full data.

P=
requests a fixed Tweedie power parameter.

TWEEDIEEQL | EQL
requests that extended quasi-likelihood be used instead of Tweedie log likelihood in parameter estimation.

If you do not specify a link function with the LINK= option, a default link function is used. The default link function for each distribution is shown in Table 7.6. For the binary and multinomial distributions, only the link functions shown in Table 7.6 are available. For the other distributions, you can use any link function shown in Table 7.7 by specifying the LINK= option. Other commonly used link functions for each distribution are shown in Table 7.6.

Table 7.6 Default and Commonly Used Link Functions

DISTRIBUTION=	Default Link Function	Other Commonly Used Link Functions
BINARY	Logit	Probit, complementary log-log, log-log
BINOMIAL	Logit	Probit, complementary log-log, log-log
GAMMA	Reciprocal	Log
INVERSEGAUSSIAN | IG	Reciprocal square	Log
MULTINOMIAL | MULT	Logit (ordinal)	Probit, complementary log-log, log-log
MULTINOMIAL | MULT	Generalized logit (nominal)	
NEGATIVEBINOMIAL | NB	Log	
NORMAL | GAUSSIAN	Identity	Log
POISSON	Log	
TWEEDIE	Log	
ZINB	Log	
ZIP	Log	

INCLUDE=*n*

INCLUDE=*single-effect*

INCLUDE=(*effects***)**

forces effects to be included in all models. If you specify INCLUDE=*n*, then the first *n* effects that are listed in the MODEL statement are included in all models. If you specify INCLUDE=*single-effect* or if you specify a list of effects within parentheses, then the specified effects are forced into all models. The effects that you specify in this option must be explanatory effects that are specified in the MODEL statement before the slash (/).

INITIAL-PHI=*number*

specifies a starting value for iterative maximum likelihood estimation of the dispersion parameter for distributions that have a dispersion parameter.

LINK=*keyword*

specifies the link function for the model. The *keywords* and the associated link functions are shown in Table 7.7. Default and commonly used link functions for the available distributions are shown in Table 7.6.

Table 7.7 Built-In Link Functions

LINK=	Link Function	$g(\mu) = \eta =$
CLOGLOG | CLL	Complementary log-log	$\log(-\log(1-\mu))$
GLOGIT | GENLOGIT	Generalized logit	
IDENTITY | ID	Identity	μ
INV | RECIP	Reciprocal	$\frac{1}{\mu}$
INV2	Reciprocal square	$\frac{1}{\mu^2}$
LOG	Logarithm	$\log(\mu)$
LOGIT	Logit	$\log(\mu/(1-\mu))$
LOGLOG	Log-log	$-\log(-\log(\mu))$
PROBIT	Probit	$\Phi^{-1}(\mu)$

$\Phi^{-1}(\cdot)$ denotes the quantile function of the standard normal distribution.

If a multinomial response variable has more than two categories, the HPGENSELECT procedure fits a model by using a cumulative link function that is based on the specified link. However, if you specify LINK=GLOGIT, the procedure assumes a generalized logit model for nominal (unordered) data, regardless of the number of response categories.

NOCENTER

requests that continuous main effects not be centered and scaled internally. (Continuous main effects are centered and scaled by default to aid in computing maximum likelihood estimates.) Parameter estimates and related statistics are always reported on the original scale.

NOINT

requests that no intercept be included in the model. (An intercept is included by default.) The NOINT option is not available in multinomial models.

OFFSET=*variable*

specifies a *variable* to be used as an offset to the linear predictor. An offset plays the role of an effect whose coefficient is known to be 1. The offset variable cannot appear in the CLASS statement or elsewhere in the MODEL statement. Observations that have missing values for the offset variable are excluded from the analysis.

SAMPLEFRAC=*number*

specifies a fraction of the data to be used to determine starting values for iterative estimation of the parameters of a Tweedie model. The sampled data are used in an extended quasi-likelihood estimation of the model parameters. The estimated parameters are then used as starting values in a full maximum likelihood estimation of the model parameters that uses all of the data.

START=*n*

START=*single-effect*

START=(*effects***)**

begins the selection process from the designated initial model for the FORWARD and STEPWISE selection methods. If you specify START=*n*, then the starting model includes the first *n* effects that are listed in the MODEL statement. If you specify START=*single-effect* or if you specify a list of

effects within parentheses, then the starting model includes those specified effects. The effects that you specify in the START= option must be explanatory effects that are specified in the MODEL statement before the slash (/). The START= option is not available when you specify METHOD=BACKWARD in the SELECTION statement.

OUTPUT Statement

OUTPUT < **OUT=***SAS-data-set* >
< *keyword* < =*name* > >. . . < *keyword* < =*name* > > < / *options* > **;**

The OUTPUT statement creates a data set that contains observationwise statistics that are computed after the model is fitted. The variables in the input data set are *not* included in the output data set to avoid data duplication for large data sets; however, variables that are specified in the ID statement are included.

If the input data are in distributed form, where accessing data in a particular order cannot be guaranteed, the HPGENSELECT procedure copies the distribution or partition key to the output data set so that its contents can be joined with the input data.

The computation of the output statistics is based on the final parameter estimates. If the model fit does not converge, missing values are produced for the quantities that depend on the estimates.

When there are more than two response levels for multinomial data, values are computed only for variables that are named by the XBETA and PREDICTED keywords; the other variables have missing values. These statistics are computed for every response category, and the automatic variable _LEVEL_ identifies the response category on which the computed values are based. If you also specify the OBSCAT option, then the observationwise statistics are computed only for the observed response category, as indicated by the value of the _LEVEL_ variable.

For observations in which only the response variable is missing, values of the XBETA and PREDICTED statistics are computed even though these observations do not affect the model fit. For zero-inflated models, ZBETA and PZERO are also computed. This practice enables predicted mean values or predicted probabilities to be computed for new observations.

You can specify the following syntax elements in the OUTPUT statement before the slash (/).

OUT=*SAS-data-set*

DATA=*SAS-data-set*

specifies the name of the output data set. If the OUT= (or DATA=) option is omitted, the procedure uses the DATA*n* convention to name the output data set.

keyword **< =***name* **>**

specifies a statistic to include in the output data set and optionally assigns a *name* to the variable. If you do not provide a *name*, the HPGENSELECT procedure assigns a default name based on the type of statistic requested.

You can specify the following *keywords* for adding statistics to the OUTPUT data set:

ADJPEARSON<=*name*>

ADJPEARS<=*name*>

STDRESCHI<=*name*>

requests the Pearson residual, adjusted to have unit variance. The adjusted Pearson residual is defined for the *i*th observation as $\frac{y_i-\mu_i}{\sqrt{\phi \mathrm{V}(\mu_i)(1-h_i)}}$, where $\mathrm{V}(\mu)$ is the response distribution variance function and h_i is the leverage. The leverage h_i of the *i*th observation is defined as the *i*th diagonal element of the hat matrix

$$\mathbf{H} = \mathbf{W}^{\frac{1}{2}}\mathbf{X}(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}^{\frac{1}{2}}$$

where $\mathbf{W}$ is the diagonal matrix whose *i*th diagonal is $w_{ei} = \frac{w_i}{\phi \mathrm{V}(\mu_i)(g'(\mu_i))^2}$, and w_i is a prior weight specified in a WEIGHT statement or 1 if no WEIGHT statement is specified. For the negative binomial, $\phi \mathrm{V}(\mu_i)$ in the denominator is replaced with the distribution variance, in both the definition of the leverage and the adjusted residual.

This statistic is not computed for multinomial models, nor is it computed for zero-modified models.

If you do not specify a *name*, PROC HPGENSELECT assigns Adjusted_Pearson as the *name*.

LINP<=*name*>

XBETA<=*name*>

requests the linear predictor $\eta = \mathbf{x}'\boldsymbol{\beta}$.

If you do not specify a *name*, PROC HPGENSELECT assigns Xbeta as the *name*.

LOWER<=name>

requests a lower confidence limit for the predicted value. This statistic is not computed for generalized logit multinomial models or zero-modified models.

If you do not specify a *name*, PROC HPGENSELECT assigns Lower as the *name*.

PEARSON<=*name*>

PEARS<=*name*>

RESCHI<=*name*>

requests the Pearson residual, $\frac{y-\mu}{\mathrm{V}(\mu)}$, where μ is the estimate of the predicted response mean and $\mathrm{V}(\mu)$ is the response distribution variance function. For the negative binomial defined in the section "Negative Binomial Distribution" on page 255 and the zero-inflated models defined in the sections "Zero-Inflated Poisson Distribution" on page 257 and "Zero-Inflated Negative Binomial Distribution" on page 256, the distribution variance is used in place of $\mathrm{V}(\mu)$.

This statistic is not computed for multinomial models.

If you do not specify a *name*, PROC HPGENSELECT assigns Pearson as the *name*.

PREDICTED<=*name*>

PRED<=*name*>

P<=*name*>

requests predicted values for the response variable.

If you do not specify a *name*, PROC HPGENSELECT assigns Pred as the *name*.

PZERO<=*name*>
requests zero-inflation probabilities for zero-inflated models.

If you do not specify a *name*, PROC HPGENSELECT assigns Pzero as the *name*.

RESIDUAL<=*name*>

RESID<=*name*>

R<=*name*>
requests the raw residual, $y - \mu$, where μ is the estimate of the predicted mean. This statistic is not computed for multinomial models.

If you do not specify a *name*, PROC HPGENSELECT assigns Residual as the *name*.

ROLE<=*name*>
requests a numeric variable that indicates the role played by each observation in fitting the model. Table 7.8 shows the interpretation of this variable for each observation.

Table 7.8 Role Interpretation

Value	Observation Role
0	Not used
1	Training
2	Validation
3	Testing

If you do not partition the input data by specifying a PARTITION statement, then the role variable value is 1 for observations that are used in fitting the model and 0 for observations that have at least one missing or invalid value for the response, regressors, frequency, or weight variable.

If you do not specify a *name*, PROC HPGENSELECT assigns Role as the *name*.

UPPER<=*name*>
requests an upper confidence limit for the predicted value. This statistic is not computed for generalized logit multinomial models or zero-modified models.

If you do not specify a *name*, PROC HPGENSELECT assigns Upper as the *name*.

ZBETA<=*name*>
requests the linear predictor for the zeros model in zero-modified models: $\kappa = \mathbf{z}'\boldsymbol{\gamma}$.

If you do not specify a *name*, PROC HPGENSELECT assigns Zbeta as the *name*.

You can specify the following *options* in the OUTPUT statement after the slash (/):

ALPHA=*number*
specifies the significance level for the construction of confidence intervals in the OUTPUT data set. The confidence level is $1 -$ *number*.

OBSCAT
requests (for multinomial models) that observationwise statistics be produced only for the response level. If the OBSCAT option is not specified and the response variable has J levels, then the following

outputs are created: for cumulative link models, $J - 1$ records are output for every observation in the input data that corresponds to the $J - 1$ lower-ordered response categories; for generalized logit models, J records are output that correspond to all J response categories.

PARTITION Statement

PARTITION < *partition-option* > **;**

The PARTITION statement specifies how observations in the input data set are to be logically partitioned into disjoint subsets for model training, validation, and testing. For more information, see the section "Using Validation and Test Data" on page 262. You can either designate a variable in the input data set and a set of formatted values of that variable to determine the role of each observation, or specify proportions to use for random assignment of observations for each role.

You can specify one of the following mutually exclusive *partition-options*:

ROLEVAR | ROLE=*variable***(< TEST='***value***' > < TRAIN='***value***' > < VALIDATE='***value***' >)**
names the variable in the input data set whose values are used to assign roles to each observation. The TEST=, TRAIN=, and VALIDATE= suboptions specify the formatted values of this variable that are used to assign observations roles. If you do not specify the TRAIN= suboption, then all observations whose role is not determined by the TEST= or VALIDATE= suboptions are assigned to training.

FRACTION(< TEST=*fraction* **> < VALIDATE=***fraction* **> < SEED=***number* **>)**
randomly assigns specified proportions of the observations in the input data set to the roles. You specify the proportions for testing and validation by using the TEST= and VALIDATE= suboptions. If you specify both the TEST= and the VALIDATE= suboptions, then the sum of the specified fractions must be less than 1 and the remaining fraction of the observations are assigned to the training role. The SEED= option specifies an integer that is used to start the pseudorandom number generator for random partitioning of data for training, testing, and validation. If you do not specify a seed, or if you specify a *number* less than or equal to 0, the seed is generated by reading the time of day from the computer's clock.

PERFORMANCE Statement

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

You can use the PERFORMANCE statement to control whether the procedure executes in single-machine or distributed mode. The default is single-machine mode.

You can also use this statement to define performance parameters for multithreaded and distributed computing, and you can request details about performance results.

The PERFORMANCE statement is documented in the section "PERFORMANCE Statement" on page 36 of Chapter 3, "Shared Concepts and Topics."

SELECTION Statement

SELECTION < *options* > **;**

The SELECTION statement performs model selection by examining whether effects should be added to or removed from the model according to rules that are defined by model selection methods. The statement is fully documented in the section "SELECTION Statement" on page 47 in Chapter 4, "Shared Statistical Concepts."

The HPGENSELECT procedure supports the following effect-selection methods in the SELECTION statement:

METHOD=NONE	results in no model selection. This method fits the full model.
METHOD=FORWARD	performs forward selection. This method starts with no effects in the model and adds effects.
METHOD=BACKWARD	performs backward elimination. This method starts with all effects in the model and deletes effects.
METHOD=STEPWISE	performs stepwise regression. This method is similar to the FORWARD method except that effects already in the model do not necessarily stay there.

The only effect-selection criterion supported by the HPGENSELECT procedure is SELECT=SL, where effects enter and leave the model based on an evaluation of the significance level. To determine this level of significance for each candidate effect, the HPGENSELECT procedure calculates an approximate chi-square test statistic.

The following criteria are available for the CHOOSE= option in the SELECT statement:

AIC	Akaike's information criterion (Akaike 1974)
AICC	a small-sample bias corrected version of Akaike's information criterion as promoted in Hurvich and Tsai (1989) and Burnham and Anderson (1998) among others
BIC | SBC	Schwarz Bayesian criterion (Schwarz 1978)

The following criteria are available for the STOP= option in the SELECT statement:

SL	the significance level of the test
AIC	Akaike's information criterion (Akaike 1974)
AICC	a small-sample bias corrected version of Akaike's information criterion as promoted in Hurvich and Tsai (1989) and Burnham and Anderson (1998) among others
BIC | SBC	Schwarz Bayesian criterion (Schwarz 1978)

The calculation of the information criteria uses the following formulas, where p denotes the number of effective parameters in the candidate model, f denotes the number of frequencies used, and l is the log likelihood evaluated at the converged estimates:

$$\mathrm{AIC} = -2l + 2p$$

$$\mathrm{AICC} = \begin{cases} -2l + 2pf/(f-p-1) & \text{when } f > p+2 \\ -2l + 2p(p+2) & \text{otherwise} \end{cases}$$

$$\mathrm{BIC} = -2l + p\log(f)$$

If the PARTITION statement is specified, then the AIC, AICC, BIC, and SL statistics are computed on the training data set; otherwise they are computed on the full data set.

When you specify one of the following DETAILS= options in the SELECTION statement, the HPGENSELECT procedure produces the indicated tables:

DETAILS=SUMMARY produces a summary table that shows which effect is added or removed at each step along with the p-value. The summary table is produced by default if the DETAILS= option is not specified.

DETAILS=STEPS produces a table of selection details that displays fit statistics for the model at each step of the selection process and the approximate log p-value. The summary table that results from the DETAILS=SUMMARY option is also produced.

DETAILS=ALL produces all the tables that are produced when DETAILS=STEPS and also produces a table that displays the effect that is added or removed at each step along with the p-value, chi-square statistic, and fit statistics for the model.

WEIGHT Statement

WEIGHT *variable* **;**

The *variable* in the WEIGHT statement is used as a weight to perform a weighted analysis of the data. Observations that have nonpositive or missing weights are not included in the analysis. If a WEIGHT statement is not included, then all observations used in the analysis are assigned a weight of 1.

ZEROMODEL Statement

ZEROMODEL < *effects* > < / *zeromodel-options* > **;**

The ZEROMODEL statement defines the statistical model for zero inflation probability in terms of model effects that are constructed from variables in the input data set. An intercept term is always included in the model.

You can specify the following *zeromodel-options*.

INCLUDE=*n*

INCLUDE=*single-effect*

INCLUDE=(*effects***)**

forces effects to be included in all models for zero inflation for all selection methods. If you specify INCLUDE=*n*, then the first *n* effects that are listed in the ZEROMODEL statement are included in all models. If you specify INCLUDE=*single-effect* or if you specify a list of effects within parentheses, then the specified effects are forced into all models. The effects that you specify in the INCLUDE= option must be explanatory effects that are specified in the ZEROMODEL statement before the slash (/).

LINK=*keyword*

specifies the link function for the zero inflation probability. The *keywords* and the associated link functions are shown in Table 7.9.

Table 7.9 Built-In Link Functions for Zero Inflation Probability

LINK=	Link Function	$g(\mu) = \eta =$
CLOGLOG | CLL	Complementary log-log	$\log(-\log(1-\mu))$
LOGIT	Logit	$\log(\mu/(1-\mu))$
LOGLOG	Log-log	$-\log(-\log(\mu))$
PROBIT	Probit	$\Phi^{-1}(\mu)$

$\Phi^{-1}(\cdot)$ denotes the quantile function of the standard normal distribution.

START=*n*

START=*single-effect*

START=(*effects***)**

begins the selection process from the designated initial zero inflation model for the FORWARD and STEPWISE selection methods. If you specify START=*n*, then the starting model includes the first *n* effects that are listed in the ZEROMODEL statement. If you specify START=*single-effect* or if you specify a list of effects within parentheses, then the starting model includes those specified effects. The effects that you specify in the START= option must be explanatory effects that are specified in the ZEROMODEL statement before the slash (/). The START= option is not available when you specify METHOD=BACKWARD in the SELECTION statement.

Details: HPGENSELECT Procedure

Missing Values

Any observation that has missing values for the response, frequency, weight, offset, or explanatory variables is excluded from the analysis; however, missing values are valid for response and explanatory variables that are specified in the MISSING option in the CLASS statement. Observations that have a nonpositive weight or a frequency less than 1 are also excluded.

The estimated linear predictor and the fitted probabilities are not computed for any observation that has missing offset or explanatory variable values. However, if only the response value is missing, the linear predictor and the fitted probabilities can be computed and output to a data set by using the OUTPUT statement.

Exponential Family Distributions

Many of the probability distributions that the HPGENSELECT procedure fits are members of an exponential family of distributions, which have probability distributions that are expressed as follows for some functions b and c that determine the specific distribution:

$$f(y) = \exp\left\{\frac{y\theta - b(\theta)}{\phi} + c(y,\phi)\right\}$$

For fixed ϕ, this is a one-parameter exponential family of distributions. The response variable can be discrete or continuous, so $f(y)$ represents either a probability mass function or a probability density function. A more useful parameterization of generalized linear models is by the mean and variance of the distribution:

$$\begin{aligned} \mathrm{E}(Y) &= b'(\theta) \\ \mathrm{Var}(Y) &= b''(\theta)\phi \end{aligned}$$

In generalized linear models, the mean μ of the response distribution is related to linear regression parameters through a link function,

$$g(\mu_i) = \mathbf{x}_i'\boldsymbol{\beta}$$

for the ith observation, where $\mathbf{x}_i$ is a fixed known vector of explanatory variables and $\boldsymbol{\beta}$ is a vector of regression parameters. The HPGENSELECT procedure parameterizes models in terms of the regression parameters $\boldsymbol{\beta}$ and either the dispersion parameter ϕ or a parameter that is related to ϕ, depending on the model. For exponential family models, the distribution variance is $\mathrm{Var}(Y) = \phi\mathrm{V}(\mu)$ where $\mathrm{V}(\mu)$ is a variance function that depends only on μ.

The zero-inflated models and the multinomial models are not exponential family models, but they are closely related models that are useful and are included in the HPGENSELECT procedure.

Response Distributions

The response distribution is the probability distribution of the response (target) variable. The HPGENSELECT procedure can fit data for the following distributions:

- binary distribution
- binomial distribution
- gamma distribution

- inverse Gaussian distribution
- multinomial distribution (ordinal and nominal)
- negative binomial distribution
- normal (Gaussian) distribution
- Poisson distribution
- Tweedie distribution
- zero-inflated negative binomial distribution
- zero-inflated Poisson distribution

Expressions for the probability distributions (probability density functions for continuous variables or probability mass functions for discrete variables) are shown in the section "Response Probability Distribution Functions" on page 254. The expressions for the log-likelihood functions of these distributions are given in the section "Log-Likelihood Functions" on page 257.

The binary (or Bernoulli) distribution is the elementary distribution of a discrete random variable that can take on two values that have probabilities p and $1 - p$. Suppose the random variable is denoted Y and

$$\Pr(Y = 1) = p$$
$$\Pr(Y = 0) = 1 - p$$

The value that is associated with probability p is often termed the *event* or "success"; the complementary event is termed the *non-event* or "failure." A Bernoulli experiment is a random draw from a binary distribution and generates events with probability p.

If $Y_1, \cdots, Y_n$ are n independent Bernoulli random variables, then their sum follows a binomial distribution. In other words, if $Y_i = 1$ denotes an event (success) in the ith Bernoulli trial, a binomial random variable is the number of events (successes) in n independent Bernoulli trials. If you use the events/trials syntax in the MODEL statement and you specify the DISTRIBUTION=BINOMIAL option, the HPGENSELECT procedure fits the model as if the data had arisen from a binomial distribution. For example, the following statements fit a binomial regression model that has regressors x1 and x2. The variables e and t represent the events and trials, respectively, for the binomial distribution:

```
proc hpgenselect;
   model e/t = x1 x2 / distribution=Binomial;
run;
```

If the events/trials syntax is used, then both variables must be numeric and the value of the events variable cannot be less than 0 or exceed the value of the trials variable. A "Response Profile" table is not produced for binomial data, because the response variable is not subject to levelization.

The multinomial distribution is a generalization of the binary distribution and allows for more than two outcome categories. Because there are more than two possible outcomes for the multinomial distribution, the terminology of "successes," "failures," "events," and "non-events" no longer applies. For multinomial data, these outcomes are generically referred to as "categories" or levels.

Whenever the HPGENSELECT procedure determines that the response variable is listed in a CLASS statement and has more than two levels (unless the events/trials syntax is used), the procedure fits the

model as if the data had arisen from a multinomial distribution. By default, it is then assumed that the response categories are ordered and a cumulative link model is fit by applying the default or specified link function. If the response categories are unordered, then you should fit a generalized logit model by choosing LINK=GLOGIT in the MODEL statement.

If the response variable is not listed in a CLASS statement and a response distribution is not specified in a DISTRIBUTION= option, then a normal distribution that uses the default or specified link function is assumed.

Response Probability Distribution Functions

Binary Distribution

$$\begin{aligned} f(y) &= \begin{cases} p & \text{for } y = 1 \\ 1-p & \text{for } y = 0 \end{cases} \\ \mathrm{E}(Y) &= p \\ \mathrm{Var}(Y) &= p(1-p) \end{aligned}$$

Binomial Distribution

$$\begin{aligned} f(y) &= \binom{n}{r} \mu^r (1-\mu)^{n-r} \quad \text{for } y = \frac{r}{n},\ r = 0, 1, 2, \ldots, n \\ \mathrm{E}(Y) &= \mu \\ \mathrm{Var}(Y) &= \frac{\mu(1-\mu)}{n} \end{aligned}$$

Gamma Distribution

$$\begin{aligned} f(y) &= \frac{1}{\Gamma(\nu) y} \left(\frac{y\nu}{\mu}\right)^{\nu} \exp\left(-\frac{y\nu}{\mu}\right) \quad \text{for } 0 < y < \infty \\ \phi &= \frac{1}{\nu} \\ \mathrm{E}(Y) &= \mu \\ \mathrm{Var}(Y) &= \frac{\mu^2}{\nu} \end{aligned}$$

For the gamma distribution, $\nu = \frac{1}{\phi}$ is the estimated dispersion parameter that is displayed in the output. The parameter ν is also sometimes called the gamma index parameter.

Inverse Gaussian Distribution

$$\begin{aligned} f(y) &= \frac{1}{\sqrt{2\pi y^3}\sigma} \exp\left[-\frac{1}{2y}\left(\frac{y-\mu}{\mu\sigma}\right)^2\right] \quad \text{for } 0 < y < \infty \\ \phi &= \sigma^2 \\ \mathrm{Var}(Y) &= \phi\mu^3 \end{aligned}$$

Multinomial Distribution

$$f(y_1, y_2, \cdots, y_k) = \frac{m!}{y_1! y_2! \cdots y_k!} p_1^{y_1} p_2^{y_2} \cdots p_k^{y_k}$$

Negative Binomial Distribution

$$\begin{aligned} f(y) &= \frac{\Gamma(y+1/k)}{\Gamma(y+1)\Gamma(1/k)} \frac{(k\mu)^y}{(1+k\mu)^{y+1/k}} \quad \text{for } y = 0, 1, 2, \ldots \\ \phi &= k \\ \mathrm{E}(Y) &= \mu \\ \mathrm{Var}(Y) &= \mu + \phi\mu^2 \end{aligned}$$

For the negative binomial distribution, k is the estimated dispersion parameter that is displayed in the output.

Normal Distribution

$$\begin{aligned} f(y) &= \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2\right] \quad \text{for } -\infty < y < \infty \\ \phi &= \sigma^2 \\ \mathrm{E}(Y) &= \mu \\ \mathrm{Var}(Y) &= \phi \end{aligned}$$

Poisson Distribution

$$
\begin{aligned}
f(y) &= \frac{\mu^y \mathrm{e}^{-\mu}}{y!} \quad \text{for } y = 0, 1, 2, \ldots \\
\mathrm{E}(Y) &= \mu \\
\mathrm{Var}(Y) &= \mu
\end{aligned}
$$

Tweedie Distribution

The Tweedie model is a generalized linear model from the exponential family. The Tweedie distribution is characterized by three parameters: the mean parameter μ, the dispersion ϕ, and the power p. The variance of the distribution is $\phi\mu^p$. For values of p in the range $1 < p < 2$, a Tweedie random variable can be represented as a Poisson sum of gamma distributed random variables. That is,

$$Y = \sum_{i=1}^{N} Y_i$$

where N has a Poisson distribution that has mean $\lambda = \frac{\mu^{2-p}}{\phi(2-p)}$ and the Y_is have independent, identical gamma distributions, each of which has an expected value $\mathrm{E}(Y_i) = \phi(2-p)\mu^{p-1}$ and an index parameter $\nu_i = \frac{2-p}{p-1}$.

In this case, Y has a discrete mass at 0, $\Pr(Y = 0) = \Pr(N = 0) = \exp(-\lambda)$, and the probability density of Y $f(y)$ is represented by an infinite series for $y > 0$. The HPGENSELECT procedure restricts the power parameter to satisfy $1.1 <= p$ for numerical stability in model fitting. The Tweedie distribution does not have a general closed form representation for all values of p. It can be characterized in terms of the distribution mean parameter μ, dispersion parameter ϕ, and power parameter p. For more information about the Tweedie distribution, see Frees (2010).

The distribution mean and variance are given by:

$$
\begin{aligned}
\mathrm{E}(Y) &= \mu \\
\mathrm{Var}(Y) &= \phi\mu^p
\end{aligned}
$$

Zero-Inflated Negative Binomial Distribution

$$
\begin{aligned}
f(y) &= \begin{cases} \omega + (1-\omega)(1+k\lambda)^{-\frac{1}{k}} & \text{for } y = 0 \\ (1-\omega)\frac{\Gamma(y+1/k)}{\Gamma(y+1)\Gamma(1/k)}\frac{(k\lambda)^y}{(1+k\lambda)^{y+1/k}} & \text{for } y = 1, 2, \ldots \end{cases} \\
\phi &= k \\
\mu = \mathrm{E}(Y) &= (1-\omega)\lambda \\
\mathrm{Var}(Y) &= (1-\omega)\lambda(1+\omega\lambda+k\lambda) \\
&= \mu + \left(\frac{\omega}{1-\omega} + \frac{k}{1-\omega}\right)\mu^2
\end{aligned}
$$

For the zero-inflated negative binomial distribution, k is the estimated dispersion parameter that is displayed in the output.

Zero-Inflated Poisson Distribution

$$
\begin{aligned}
f(y) &= \begin{cases} \omega + (1-\omega)\mathrm{e}^{-\lambda} & \text{for } y = 0 \\ (1-\omega)\frac{\lambda^y \mathrm{e}^{-\lambda}}{y!} & \text{for } y = 1, 2, \ldots \end{cases} \\
\mu = \mathrm{E}(Y) &= (1-\omega)\lambda \\
\mathrm{Var}(Y) &= (1-\omega)\lambda(1+\omega\lambda) \\
&= \mu + \frac{\omega}{1-\omega}\mu^2
\end{aligned}
$$

Log-Likelihood Functions

The HPGENSELECT procedure forms the log-likelihood functions of the various models as

$$L(\boldsymbol{\mu}; \mathbf{y}) = \sum_{i=1}^{n} f_i \, l(\mu_i; y_i, w_i)$$

where $l(\mu_i; y_i, w_i)$ is the log-likelihood contribution of the ith observation that has weight w_i, and f_i is the value of the frequency variable. For the determination of w_i and f_i, see the WEIGHT and FREQ statements. The individual log-likelihood contributions for the various distributions are as follows.

In the following, the mean parameter μ_i for each observation i is related to the regression parameters $\boldsymbol{\beta}_i$ through the linear predictor $\eta_i = \mathbf{x}_i'\boldsymbol{\beta}$ by

$$\mu_i = g^{-1}(\eta_i)$$

where g is the link function.

There are two link functions and linear predictors that are associated with zero-inflated Poisson and zero-inflated negative binomial distributions: one for the zero-inflation probability ω, and another for the parameter λ, which is the Poisson or negative binomial mean if there is no zero-inflation. Each of these parameters is related to regression parameters through an individual link function,

$$
\begin{aligned}
\eta_i &= \mathbf{x}_i'\boldsymbol{\beta} \\
\kappa_i &= \mathbf{z}_i'\boldsymbol{\gamma} \\
\lambda_i(\boldsymbol{\beta}) &= g^{-1}(\eta_i) \\
\omega_i(\boldsymbol{\gamma}) &= h^{-1}(\kappa_i)
\end{aligned}
$$

where h is one of the following link functions that are associated with binary data: complementary log-log, log-log, logit, or probit. These link functions are also shown in Table 7.9.

Binary Distribution

The HPGENSELECT procedure computes the log-likelihood function $l(\mu_i(\boldsymbol{\beta}); y_i)$ for the ith binary observation as

$$\begin{aligned} \eta_i &= \mathbf{x}_i'\boldsymbol{\beta} \\ \mu_i(\boldsymbol{\beta}) &= g^{-1}(\eta_i) \\ l(\mu_i(\boldsymbol{\beta}); y_i) &= y_i \log\{\mu_i\} + (1 - y_i)\log\{1 - \mu_i\} \end{aligned}$$

Here, μ_i is the probability of an event, and the variable y_i takes on the value 1 for an event and the value 0 for a non-event. The inverse link function $g^{-1}(\cdot)$ maps from the scale of the linear predictor η_i to the scale of the mean. For example, for the logit link (the default),

$$\mu_i(\boldsymbol{\beta}) = \frac{\exp\{\eta_i\}}{1 + \exp\{\eta_i\}}$$

You can control which binary outcome in your data is modeled as the event by specifying the *response-options* in the MODEL statement, and you can choose the link function by specifying the LINK= option in the MODEL statement.

If a WEIGHT statement is specified and w_i denotes the weight for the current observation, the log-likelihood function is computed as

$$l(\mu_i(\boldsymbol{\beta}); y_i, w_i) = w_i\, l(\mu_i(\boldsymbol{\beta}); y_i)$$

Binomial Distribution

The HPGENSELECT procedure computes the log-likelihood function $l(\mu_i(\boldsymbol{\beta}); y_i)$ for the ith binomial observation as

$$\begin{aligned} \eta_i &= \mathbf{x}_i'\boldsymbol{\beta} \\ \mu_i(\boldsymbol{\beta}) &= g^{-1}(\eta_i) \\ l(\mu_i(\boldsymbol{\beta}); y_i, w_i) &= w_i\,(y_i \log\{\mu_i\} + (n_i - y_i)\log\{1 - \mu_i\}) \\ &\quad + w_i\,(\log\{\Gamma(n_i + 1)\} - \log\{\Gamma(y_i + 1)\} - \log\{\Gamma(n_i - y_i + 1)\}) \end{aligned}$$

where y_i and n_i are the values of the events and trials of the ith observation, respectively. μ_i measures the probability of events (successes) in the underlying Bernoulli distribution whose aggregate follows the binomial distribution.

Gamma Distribution

The HPGENSELECT procedure computes the log-likelihood function $l(\mu_i(\boldsymbol{\beta}); y_i)$ for the ith observation as

$$\begin{aligned} \eta_i &= \mathbf{x}_i'\boldsymbol{\beta} \\ \mu_i(\boldsymbol{\beta}) &= g^{-1}(\eta_i) \\ l(\mu_i(\boldsymbol{\beta}); y_i, w_i) &= \frac{w_i}{\phi}\log\left(\frac{w_i y_i}{\phi\mu_i}\right) - \frac{w_i y_i}{\phi\mu_i} - \log(y_i) - \log\left(\Gamma\left(\frac{w_i}{\phi}\right)\right) \end{aligned}$$

For the gamma distribution, $\nu = \frac{1}{\phi}$ is the estimated dispersion parameter that is displayed in the output.

Inverse Gaussian Distribution

The HPGENSELECT procedure computes the log-likelihood function $l(\mu_i(\boldsymbol{\beta}); y_i)$ for the ith observation as

$$\begin{aligned} \eta_i &= \mathbf{x}_i'\boldsymbol{\beta} \\ \mu_i(\boldsymbol{\beta}) &= g^{-1}(\eta_i) \\ l(\mu_i(\boldsymbol{\beta}); y_i, w_i) &= -\frac{1}{2}\left[\frac{w_i(y_i-\mu_i)^2}{y_i\mu^2\phi} + \log\left(\frac{\phi y_i^3}{w_i}\right) + \log(2\pi)\right] \end{aligned}$$

where ϕ is the dispersion parameter.

Multinomial Distribution

The multinomial distribution that is modeled by the HPGENSELECT procedure is a generalization of the binary distribution; it is the distribution of a single draw from a discrete distribution with J possible values. The log-likelihood function for the ith observation is

$$l(\boldsymbol{\mu}_i; \mathbf{y}_i, w_i) = w_i \sum_{j=1}^{J} y_{ij}\log\{\mu_{ij}\}$$

In this expression, J denotes the number of response categories (the number of possible outcomes) and μ_{ij} is the probability that the ith observation takes on the response value that is associated with category j. The category probabilities must satisfy

$$\sum_{j=1}^{J}\mu_j = 1$$

and the constraint is satisfied by modeling $J-1$ categories. In models that have ordered response categories, the probabilities are expressed in cumulative form, so that the last category is redundant. In generalized logit models (multinomial models that have unordered categories), one category is chosen as the reference category and the linear predictor in the reference category is set to 0.

Negative Binomial Distribution

The HPGENSELECT procedure computes the log-likelihood function $l(\mu_i(\boldsymbol{\beta}); y_i)$ for the ith observation as

$$\begin{aligned}
\eta_i &= \mathbf{x}_i'\boldsymbol{\beta} \\
\mu_i(\boldsymbol{\beta}) &= g^{-1}(\eta_i) \\
l(\mu_i(\boldsymbol{\beta}); y_i, w_i) &= y_i \log\left(\frac{k\mu}{w_i}\right) - (y_i + w_i/k)\log\left(1 + \frac{k\mu}{w_i}\right) + \log\left(\frac{\Gamma(y_i + w_i/k)}{\Gamma(y_i+1)\Gamma(w_i/k)}\right)
\end{aligned}$$

where k is the negative binomial dispersion parameter that is displayed in the output.

Normal Distribution

The HPGENSELECT procedure computes the log-likelihood function $l(\mu_i(\boldsymbol{\beta}); y_i)$ for the ith observation as

$$\begin{aligned}
\eta_i &= \mathbf{x}_i'\boldsymbol{\beta} \\
\mu_i(\boldsymbol{\beta}) &= g^{-1}(\eta_i) \\
l(\mu_i(\boldsymbol{\beta}); y_i, w_i) &= -\frac{1}{2}\left[\frac{w_i(y_i - \mu_i)^2}{\phi} + \log\left(\frac{\phi}{w_i}\right) + \log(2\pi)\right]
\end{aligned}$$

where ϕ is the dispersion parameter.

Poisson Distribution

The HPGENSELECT procedure computes the log-likelihood function $l(\mu_i(\boldsymbol{\beta}); y_i)$ for the ith observation as

$$\begin{aligned}
\eta_i &= \mathbf{x}_i'\boldsymbol{\beta} \\
\mu_i(\boldsymbol{\beta}) &= g^{-1}(\eta_i) \\
l(\mu_i(\boldsymbol{\beta}); y_i, w_i) &= w_i[y_i \log(\mu_i) - \mu_i - \log(y_i!)]
\end{aligned}$$

Tweedie Distribution

The Tweedie distribution does not in general have a closed form log-likelihood function in terms of the mean, dispersion, and power parameters. The form of the log likelihood is

$$L(\boldsymbol{\mu}; \mathbf{y}) = \sum_{i=1}^{n} f_i\, l(\mu_i; y_i, w_i)$$

where

$$l(\mu_i, y_i, w_i) = \log(f(y_i; \mu_i, p, \frac{\phi}{w_i}))$$

and $f(y, \mu, p, \phi)$ is the Tweedie probability distribution, which is described in the section "Tweedie Distribution" on page 256. Evaluation of the Tweedie log-likelihood for model fitting is performed numerically as described in Dunn and Smyth (2005, 2008).

Quasi-likelihood

The extended quasi-likelihood (EQL) is constructed according to the definition of McCullagh and Nelder (1989, Chapter 9) as

$$Q_p(\mathbf{y}, \boldsymbol{\mu}, \phi, p) = \sum_i q(y_i, \mu_i, \phi, p)$$

where the contribution from an observation is

$$q(y_i, \mu_i, \phi, p) = -0.5 \log(2\pi \frac{\phi}{w_i} y_i^p) - w_i \left(\frac{y_i^{2-p} - (2-p) y_i \mu_i^{1-p} + (1-p)\mu_i^{2-p}}{(1-p)(1-p)} \right) / \phi$$

where $1 < p < 2$. This EQL is used in computing initial values for the iterative maximization of the Tweedie log likelihood, as specified using the OPTMETHOD= Tweedie option in Table 7.5. If you specify the OPTMETHOD=EQL *Tweedie-optimization-option* in Table 7.5, then the parameter estimates are computed by using the EQL instead of the log likelihood.

Zero-Inflated Negative Binomial Distribution

The HPGENSELECT procedure computes the log-likelihood function $l(\lambda_i(\boldsymbol{\beta}), \omega_i(\boldsymbol{\gamma}); y_i)$ for the ith observation as

$$\begin{aligned}
\eta_i &= \mathbf{x}_i' \boldsymbol{\beta} \\
\kappa_i &= \mathbf{z}_i' \boldsymbol{\gamma} \\
\lambda_i(\boldsymbol{\beta}) &= g^{-1}(\eta_i) \\
\omega_i(\boldsymbol{\gamma}) &= h^{-1}(\kappa_i) \\
l(\mu_i(\boldsymbol{\beta}), \omega_i(\boldsymbol{\gamma}); y_i, w_i) &= \begin{cases}
\log[\omega_i + (1-\omega_i)(1 + \frac{k}{w_i}\lambda)^{-\frac{1}{k}}] & y_i = 0 \\
\log(1-\omega_i) + y_i \log\left(\frac{k\lambda}{w_i}\right) & \\
-(y_i + \frac{w_i}{k}) \log\left(1 + \frac{k\lambda}{w_i}\right) & \\
+ \log\left(\frac{\Gamma(y_i + \frac{w_i}{k})}{\Gamma(y_i+1)\Gamma(\frac{w_i}{k})}\right) & y_i > 0
\end{cases}
\end{aligned}$$

where k is the zero-inflated negative binomial dispersion parameter that is displayed in the output.

Zero-Inflated Poisson Distribution

The HPGENSELECT procedure computes the log-likelihood function $l(\lambda_i(\boldsymbol{\beta}), \omega_i(\boldsymbol{\gamma}); y_i)$ for the ith observation as

$$\begin{aligned}
\eta_i &= \mathbf{x}_i' \boldsymbol{\beta} \\
\kappa_i &= \mathbf{z}_i' \boldsymbol{\gamma} \\
\lambda_i(\boldsymbol{\beta}) &= g^{-1}(\eta_i) \\
\omega_i(\boldsymbol{\gamma}) &= h^{-1}(\kappa_i) \\
l(\mu_i(\boldsymbol{\beta}), \omega_i(\boldsymbol{\gamma}); y_i, w_i) &= \begin{cases}
w_i \log[\omega_i + (1-\omega_i)\exp(-\lambda_i)] & y_i = 0 \\
w_i[\log(1-\omega_i) + y_i \log(\lambda_i) - \lambda_i - \log(y_i!)] & y_i > 0
\end{cases}
\end{aligned}$$

Using Validation and Test Data

When you have sufficient data, you can subdivide your data into three parts, which are called the training, validation, and test data. For a single model fit or during the model selection process, models are fit and selected based on the training data. After a model has been fit, the validation and test sets can be used to assess how the selected model generalizes on data that played no role in selecting the model. For example, Hastie, Tibshirani, and Friedman (2001) advocate using validation data in the model selection process to determine which effects to include in each step and when to terminate the selection process. PROC HPGENSELECT does not currently use validation data in this way, so the validation and test data subsets are equivalent.

You can use validation and test data to score data that were not used in fitting the model. Statistics in an output data set that is created by an OUTPUT statement are computed for validation and test data, using the model fit based on the training data. You can use the ROLE option in an OUTPUT statement to add a variable (named Role by default) to an output data set to indicate the role played by each observation.

You use a PARTITION statement to logically subdivide the DATA= data set into separate roles. You can specify the fractions of the data that you want to reserve as test data and validation data. For example, the following statements randomly subdivide the inData data set, reserving 50% for training and 25% each for validation and testing:

```
proc hpgenselect data=inData;
  partition fraction(test=0.25 validate=0.25);
  ...
run;
```

In some cases you might need to exercise more control over the partitioning of the input data set. You can do this by naming both a variable in the input data set and a formatted value of that variable that corresponds to each role. For example, the following statements assign roles to the observations in the inData data set based on the value of the variable group in that data set. Observations whose value of Group is `'group 1'` are assigned for testing, and those whose value is `'group 2'` are assigned to training. All other observations are ignored.

```
proc hpgenselect data=inData;
  partition roleVar=Group(test='group 1' train='group 2')
  ...
run;
```

When you have reserved observations for training, validation, and testing, a model that is fit on the training data is scored on the validation and test data, and fit statistics, including the average squared error (ASE), are computed separately for each of these subsets. The ASE for each data role is the sum of the squared differences between the responses and the predictions for observations in that role divided by the number of observations in that role.

Computational Method: Multithreading

Threading refers to the organization of computational work into multiple tasks (processing units that can be scheduled by the operating system). A task is associated with a thread. Multithreading refers to the concurrent execution of threads. When multithreading is possible, substantial performance gains can be realized compared to sequential (single-threaded) execution.

The number of threads spawned by the HPGENSELECT procedure is determined by the number of CPUs on a machine and can be controlled in the following ways:

- You can specify the number of CPUs in the CPUCOUNT= SAS system option. For example, if you specify the following statement, the HPGENSELECT procedure determines threading as if it executed on a system that has four CPUs, regardless of the actual CPU count:

```
options cpucount=4;
```

- You can specify the NTHREADS= option in the PERFORMANCE statement to control the number of threads. This specification overrides the CPUCOUNT= system option. Specify NTHREADS=1 to force single-threaded execution.

The number of threads per machine is displayed in the "Dimensions" table, which is part of the default output. The HPGENSELECT procedure allocates one thread per CPU by default.

The tasks that are multithreaded by the HPGENSELECT procedure are primarily defined by dividing the data that are processed on a single machine among the threads—that is, the HPGENSELECT procedure implements multithreading through a data-parallel model. For example, if the input data set has 1,000 observations and PROC HPGENSELECT is running with four threads, then 250 observations are associated with each thread. All operations that require access to the data are then multithreaded. These operations include the following:

- variable levelization
- effect levelization
- formation of the initial crossproducts matrix
- formation of approximate Hessian matrices for candidate evaluation during model selection
- objective function calculation
- gradient calculation
- Hessian calculation
- scoring of observations

In addition, operations on matrices such as sweeps can be multithreaded provided that the matrices are of sufficient size to realize performance benefits from managing multiple threads for the particular matrix operation.

Choosing an Optimization Algorithm

First- or Second-Order Algorithms

The factors that affect how you choose an optimization technique for a particular problem are complex. Although the default method works well for most problems, you might occasionally benefit from trying several different algorithms.

For many optimization problems, computing the gradient takes more computer time than computing the function value. Computing the Hessian sometimes takes *much* more computer time and memory than computing the gradient, especially when there are many decision variables. Unfortunately, optimization techniques that do not use some kind of Hessian approximation usually require many more iterations than techniques that do use a Hessian matrix; as a result, the total run time of these techniques is often longer. Techniques that do not use the Hessian also tend to be less reliable. For example, they can terminate more easily at stationary points than at global optima.

Table 7.10 shows which derivatives are required for each optimization technique.

Table 7.10 Derivatives Required

Algorithm	First-Order	Second-Order
TRUREG	x	x
NEWRAP	x	x
NRRIDG	x	x
QUANEW	x	-
DBLDOG	x	-
CONGRA	x	-
NMSIMP	-	-

The second-derivative methods TRUREG, NEWRAP, and NRRIDG are best for small problems for which the Hessian matrix is not expensive to compute. Sometimes the NRRIDG algorithm can be faster than the TRUREG algorithm, but TRUREG can be more stable. The NRRIDG algorithm requires only one matrix with $p(p+1)/2$ double words; TRUREG and NEWRAP require two such matrices. Here, p denotes the number of parameters in the optimization.

The first-derivative methods QUANEW and DBLDOG are best for medium-sized problems for which the objective function and the gradient can be evaluated much faster than the Hessian. In general, the QUANEW and DBLDOG algorithms require more iterations than TRUREG, NRRIDG, and NEWRAP, but each iteration can be much faster. The QUANEW and DBLDOG algorithms require only the gradient to update an approximate Hessian, and they require slightly less memory than TRUREG or NEWRAP.

The first-derivative method CONGRA is best for large problems for which the objective function and the gradient can be computed much faster than the Hessian and for which too much memory is required to store the (approximate) Hessian. In general, the CONGRA algorithm requires more iterations than QUANEW or DBLDOG, but each iteration can be much faster. Because CONGRA requires only a factor of p double-word memory, many large applications can be solved only by CONGRA.

The no-derivative method NMSIMP is best for small problems for which derivatives are not continuous or are very difficult to compute.

Each optimization method uses one or more convergence criteria that determine when it has converged. An algorithm is considered to have converged when any one of the convergence criteria is satisfied. For example, under the default settings, the QUANEW algorithm converges if ABSGCONV <1E–5, FCONV $< 2 \times \epsilon$, or GCONV <1E–8.

By default, the HPGENSELECT procedure applies the NRRIDG algorithm because it can take advantage of multithreading in Hessian computations and inversions. If the number of parameters becomes large, specifying the TECHNIQUE=QUANEW option (which is a first-order method with good overall properties), is recommended.

Algorithm Descriptions

The following subsections provide details about each optimization technique and follow the same order as Table 7.10.

Trust Region Optimization (TRUREG)

The trust region method uses the gradient $\mathbf{g}(\boldsymbol{\psi}^{(k)})$ and the Hessian matrix $\mathbf{H}(\boldsymbol{\psi}^{(k)})$; thus, it requires that the objective function $f(\boldsymbol{\psi})$ have continuous first- and second-order derivatives inside the feasible region.

The trust region method iteratively optimizes a quadratic approximation to the nonlinear objective function within a hyperelliptic trust region with radius Δ that constrains the step size that corresponds to the quality of the quadratic approximation. The trust region method is implemented based on Dennis, Gay, and Welsch (1981); Gay (1983); Moré and Sorensen (1983).

The trust region method performs well for small- to medium-sized problems, and it does not need many function, gradient, and Hessian calls. However, if the computation of the Hessian matrix is computationally expensive, one of the dual quasi-Newton or conjugate gradient algorithms might be more efficient.

Newton-Raphson Optimization with Line Search (NEWRAP)

The NEWRAP technique uses the gradient $\mathbf{g}(\boldsymbol{\psi}^{(k)})$ and the Hessian matrix $\mathbf{H}(\boldsymbol{\psi}^{(k)})$; thus, it requires that the objective function have continuous first- and second-order derivatives inside the feasible region.

If second-order derivatives are computed efficiently and precisely, the NEWRAP method can perform well for medium-sized to large problems, and it does not need many function, gradient, and Hessian calls.

This algorithm uses a pure Newton step when the Hessian is positive-definite and when the Newton step reduces the value of the objective function successfully. Otherwise, a combination of ridging and line search is performed to compute successful steps. If the Hessian is not positive-definite, a multiple of the identity matrix is added to the Hessian matrix to make it positive-definite (Eskow and Schnabel 1991).

In each iteration, a line search is performed along the search direction to find an approximate optimum of the objective function. The line-search method uses quadratic interpolation and cubic extrapolation.

Newton-Raphson Ridge Optimization (NRRIDG)

The NRRIDG technique uses the gradient $\mathbf{g}(\boldsymbol{\psi}^{(k)})$ and the Hessian matrix $\mathbf{H}(\boldsymbol{\psi}^{(k)})$; thus, it requires that the objective function have continuous first- and second-order derivatives inside the feasible region.

This algorithm uses a pure Newton step when the Hessian is positive-definite and when the Newton step reduces the value of the objective function successfully. If at least one of these two conditions is not satisfied, a multiple of the identity matrix is added to the Hessian matrix.

Because the NRRIDG technique uses an orthogonal decomposition of the approximate Hessian, each iteration of NRRIDG can be slower than an iteration of the NEWRAP technique, which works with a Cholesky decomposition. However, NRRIDG usually requires fewer iterations than NEWRAP.

The NRRIDG method performs well for small- to medium-sized problems, and it does not require many function, gradient, and Hessian calls. However, if the computation of the Hessian matrix is computationally expensive, one of the dual quasi-Newton or conjugate gradient algorithms might be more efficient.

Quasi-Newton Optimization (QUANEW)

The dual quasi-Newton method uses the gradient $\mathbf{g}(\boldsymbol{\psi}^{(k)})$, and it does not need to compute second-order derivatives because they are approximated. It works well for medium-sized to moderately large optimization problems, where the objective function and the gradient can be computed much faster than the Hessian. However, in general the QUANEW technique requires more iterations than the TRUREG, NEWRAP, and NRRIDG techniques, which compute second-order derivatives. The QUANEW technique provides an appropriate balance between the speed and stability that are required for most generalized linear model applications.

The QUANEW technique that is implemented by the HPGENSELECT procedure is the dual quasi-Newton algorithm, which updates the Cholesky factor of an approximate Hessian.

In each iteration, a line search is performed along the search direction to find an approximate optimum. The line-search method uses quadratic interpolation and cubic extrapolation to obtain a step size α that satisfies the Goldstein conditions (Fletcher 1987). One of the Goldstein conditions can be violated if the feasible region defines an upper limit of the step size. Violating the left-side Goldstein condition can affect the positive-definiteness of the quasi-Newton update. In that case, either the update is skipped or the iterations are restarted by using an identity matrix, resulting in the steepest descent or ascent search direction.

Double-Dogleg Optimization (DBLDOG)

The double-dogleg optimization method combines the ideas of the quasi-Newton and trust region methods. In each iteration, the double-dogleg algorithm computes the step $\mathbf{s}^{(k)}$ as the linear combination of the steepest descent or ascent search direction $\mathbf{s}_1^{(k)}$ and a quasi-Newton search direction $\mathbf{s}_2^{(k)}$:

$$\mathbf{s}^{(k)} = \alpha_1 \mathbf{s}_1^{(k)} + \alpha_2 \mathbf{s}_2^{(k)}$$

The step is requested to remain within a prespecified trust region radius (Fletcher 1987, p. 107). Thus, the DBLDOG subroutine uses the dual quasi-Newton update but does not perform a line search.

The double-dogleg optimization technique works well for medium-sized to moderately large optimization problems, where the objective function and the gradient can be computed much faster than the Hessian. The implementation is based on Dennis and Mei (1979); Gay (1983), but it is extended for dealing with boundary and linear constraints. The DBLDOG technique generally requires more iterations than the TRUREG, NEWRAP, and NRRIDG techniques, which require second-order derivatives; however, each of the DBLDOG iterations is computationally cheap. Furthermore, the DBLDOG technique requires only gradient calls for the update of the Cholesky factor of an approximate Hessian.

Conjugate Gradient Optimization (CONGRA)

Second-order derivatives are not required by the CONGRA algorithm and are not even approximated. The CONGRA algorithm can be expensive in function and gradient calls, but it requires only $O(p)$ memory for unconstrained optimization. In general, the algorithm must perform many iterations to obtain a precise solution, but each of the CONGRA iterations is computationally cheap.

The CONGRA algorithm should be used for optimization problems that have large p. For the unconstrained or boundary-constrained case, the CONGRA algorithm requires only $O(p)$ bytes of working memory, whereas all other optimization methods require order $O(p^2)$ bytes of working memory. During p successive iterations, uninterrupted by restarts or changes in the working set, the CONGRA algorithm computes a cycle of p conjugate search directions. In each iteration, a line search is performed along the search direction to find an approximate optimum of the objective function. The line-search method uses quadratic interpolation and cubic extrapolation to obtain a step size α that satisfies the Goldstein conditions. One of the Goldstein conditions can be violated if the feasible region defines an upper limit for the step size.

Nelder-Mead Simplex Optimization (NMSIMP)

The Nelder-Mead simplex method does not use any derivatives and does not assume that the objective function has continuous derivatives. The objective function itself needs to be continuous. This technique is quite expensive in the number of function calls, and it might be unable to generate precise results for $p \gg 40$.

The original Nelder-Mead simplex algorithm is implemented and extended to boundary constraints. This algorithm does not compute the objective for infeasible points, but it changes the shape of the simplex adapting to the nonlinearities of the objective function. This change contributes to an increased speed of convergence and uses a special termination criterion.

Displayed Output

The following sections describe the output that PROC HPGENSELECT produces by default. The output is organized into various tables, which are discussed in the order of their appearance.

Performance Information

The "Performance Information" table is produced by default. It displays information about the execution mode. For single-machine mode, the table displays the number of threads used. For distributed mode, the table displays the grid mode (symmetric or asymmetric), the number of compute nodes, and the number of threads per node.

If you specify the DETAILS option in the PERFORMANCE statement, the procedure also produces a "Timing" table in which elapsed times (absolute and relative) for the main tasks of the procedure are displayed.

Model Information

The "Model Information" table displays basic information about the model, such as the response variable, frequency variable, link function, and the model category that the HPGENSELECT procedure determined based on your input and options. The "Model Information" table also displays the distribution of the data that is assumed by the HPGENSELECT procedure. For information about how the procedure determines the response distribution, see the section "Response Distributions" on page 252.

Class Level Information

The "Class Level Information" table lists the levels of every variable that is specified in the CLASS statement. You should check this information to make sure that the data are correct. You can adjust the order of the CLASS variable levels by specifying the ORDER= option in the CLASS statement. You can suppress the

"Class Level Information" table completely or partially by specifying the NOCLPRINT= option in the PROC HPGENSELECT statement.

If the classification variables use reference parameterization, the "Class Level Information" table also displays the reference value for each variable.

Number of Observations

The "Number of Observations" table displays the number of observations that are read from the input data set and the number of observations that are used in the analysis. If a FREQ statement is present, the sum of the frequencies read and used is displayed. If the events/trials syntax is used, the number of events and trials is also displayed. If a PARTITION statement is specified, the table displays the values for each role.

Response Profile

The "Response Profile" table displays the ordered value from which the HPGENSELECT procedure determines the probability being modeled as an event in binary models and the ordering of categories in multinomial models. For each response category level, the frequency that is used in the analysis is reported. You can affect the ordering of the response values by specifying *response-options* in the MODEL statement. For binary and generalized logit models, the note that follows the "Response Profile" table indicates which outcome is modeled as the event in binary models and which value serves as the reference category.

The "Response Profile" table is not produced for binomial data. You can find information about the number of events and trials in the "Number of Observations" table. If a PARTITION statement is specified, the table displays the values for each role.

Entry and Removal Candidates

When you specify the DETAILS=ALL or DETAILS=STEPS option in the SELECTION statement, the HPGENSELECT procedure produces "Entry Candidates" and "Removal Candidates" tables that display the effect names and the values of the criterion that is used to select entering or departing effects at each step of the selection process. The effects are displayed in sorted order from best to worst of the selection criterion.

Selection Information

When you specify the SELECTION statement, the HPGENSELECT procedure produces by default a series of tables that have information about the model selection. The "Selection Information" table informs you about the model selection method, selection and stop criteria, and other parameters that govern the selection. You can suppress this table by specifying DETAILS=NONE in the SELECTION statement.

Selection Summary

When you specify the SELECTION statement, the HPGENSELECT procedure produces the "Selection Summary" table, which contains information about which effects were entered into or removed from the model at the steps of the model selection process. The p-value for the score chi-square test that led to the removal or entry decision is also displayed. You can request further details about the model selection steps by specifying DETAILS=STEPS or DETAILS=ALL in the SELECTION statement. You can suppress the display of the "Selection Summary" table by specifying DETAILS=NONE in the SELECTION statement.

Selection Details

When you specify the DETAILS=ALL option in the SELECTION statement, the HPGENSELECT procedure produces the "Selection Details" table, which contains information about which effects were entered into or removed from the model at the steps of the model selection process. The *p*-value and the chi-square test statistic that led to the removal or entry decision are also displayed. Fit statistics for the model at the steps are also displayed.

Stop Reason

When you specify the SELECTION statement, the HPGENSELECT procedure produces a simple table that tells you why model selection stopped.

Selection Reason

When you specify the SELECTION statement, the HPGENSELECT procedure produces a simple table that tells you why the final model was selected.

Selected Effects

When you specify the SELECTION statement, the HPGENSELECT procedure produces a simple table that tells you which effects were selected to be included in the final model.

Iteration History

For each iteration of the optimization, the "Iteration History" table displays the number of function evaluations (including gradient and Hessian evaluations), the value of the objective function, the change in the objective function from the previous iteration, and the absolute value of the largest (projected) gradient element. The objective function used in the optimization in the HPGENSELECT procedure is normalized by default to enable comparisons across data sets that have different sampling intensity. You can control normalization by specifying the NORMALIZE= option in the PROC HPGENSELECT statement.

If you specify the ITDETAILS option in the PROC HPGENSELECT statement, information about the parameter estimates and gradients in the course of the optimization is added to the "Iteration History" table. To generate the history from a model selection process, specify the ITSELECT option.

Convergence Status

The convergence status table is a small ODS table that follows the "Iteration History" table in the default output. In the listing it appears as a message that indicates whether the optimization succeeded and which convergence criterion was met. If the optimization fails, the message indicates the reason for the failure. If you save the convergence status table to an output data set, a numeric Status variable is added that enables you to programmatically assess convergence. The values of the Status variable encode the following:

0 Convergence was achieved, or an optimization was not performed because TECHNIQUE=NONE is specified.

1 The objective function could not be improved.

2 Convergence was not achieved because of a user interrupt or because a limit (such as the maximum number of iterations or the maximum number of function evaluations) was reached. To

modify these limits, see the MAXITER=, MAXFUNC=, and MAXTIME= options in the PROC HPGENSELECT statement.

3 Optimization failed to converge because function or derivative evaluations failed at the starting values or during the iterations or because a feasible point that satisfies the parameter constraints could not be found in the parameter space.

Dimensions

The "Dimensions" table displays size measures that are derived from the model and the environment. It displays the number of effects in the model, the number of columns in the design matrix, and the number of parameters for which maximum likelihood estimates are computed.

Optimization Stage Details

The "Optimization Stage Details" table displays the optimization stages that are used to fit Tweedie models. The type of optimization, the percentage of observations used, and the number of observations used are displayed for each stage.

Fit Statistics

The "Fit Statistics" table displays a variety of likelihood-based measures of fit. All statistics are presented in "smaller is better" form.

The calculation of the information criteria uses the following formulas, where p denotes the number of effective parameters, f denotes the number of frequencies used, and l is the log likelihood evaluated at the converged estimates:

$$\mathrm{AIC} = -2l + 2p$$

$$\mathrm{AICC} = \begin{cases} -2l + 2pf/(f-p-1) & \text{when } f > p+2 \\ -2l + 2p(p+2) & \text{otherwise} \end{cases}$$

$$\mathrm{BIC} = -2l + p\log(f)$$

If no FREQ statement is given, f equals n, the number of observations used.

If a PARTITION statement is specified, the table displays the values for each role. In addition, the average squared error (ASE) is computed separately for each role. The ASE for each data role is the sum of the squared differences between the responses and the predictions for observations in that role divided by the number of observations in that role.

The values displayed in the "Fit Statistics" table are not based on a normalized log-likelihood function.

Parameter Estimates

The "Parameter Estimates" table displays the parameter estimates, their estimated (asymptotic) standard errors, chi-square statistics, and p-values for the hypothesis that the parameter is 0.

If you request confidence intervals by specifying the CL option in the MODEL statement, confidence limits for regression parameters are produced for the estimate on the linear scale. Confidence limits for the dispersion parameter of those distributions that possess a dispersion parameter are produced on the log scale, because the dispersion must be greater than 0. Similarly, confidence limits for the power parameter of the Tweedie distribution are produced on the log scale.

Parameter Estimates Correlation Matrix

When you specify the CORR option in the PROC HPGENSELECT statement, the correlation matrix of the parameter estimates is displayed.

Parameter Estimates Covariance Matrix

When you specify the COV option in the PROC HPGENSELECT statement, the covariance matrix of the parameter estimates is displayed. The covariance matrix is computed as the inverse of the negative of the matrix of second derivatives of the log-likelihood function with respect to the model parameters (the Hessian matrix), evaluated at the parameter estimates.

Zero-Inflation Parameter Estimates

The parameter estimates for zero-inflation probability in zero-inflated models, their estimated (asymptotic) standard errors, chi-square statistics, and p-values for the hypothesis that the parameter is 0 are presented in the "Parameter Estimates" table. If you request confidence intervals by specifying the CL option in the MODEL statement, confidence limits for regression parameters are produced for the estimate on the linear scale.

ODS Table Names

Each table created by the HPGENSELECT procedure has a name that is associated with it, and you must use this name to refer to the table when you use ODS statements. These names are listed in Table 7.11.

Table 7.11 ODS Tables Produced by PROC HPGENSELECT

Table Name	Description	Required Statement and Option
ClassLevels	Level information from the CLASS statement	CLASS
ConvergenceStatus	Status of optimization at conclusion of optimization	Default output
CorrelationMatrix	Correlation matrix of parameter estimates	PROC HPGENSELECT CORR
CovarianceMatrix	Covariance matrix of parameter estimates	PROC HPGENSELECT COV
Dimensions	Model dimensions	Default output
EntryCandidates	Candidates for entry at step	SELECTION DETAILS=ALL | STEPS
FitStatistics	Fit statistics	Default output
IterHistory	Iteration history	PROC HPGENSELECT ITDETAILS or PROC HPGENSELECT ITSUMMARY or PROC HPGENSELECT ITSELECT

Table 7.11 *continued*

Table Name	Description	Required Statement / Option
ModelInfo	Information about the modeling environment	Default output
NObs	Number of observations read and used, and number of events and trials, if applicable	Default output
OptimizationStages	Optimization stages that are used to fit Tweedie models	MODEL DISTRIBUTION=TWEEDIE
ParameterEstimates	Solutions for the parameter estimates that are associated with effects in MODEL statements	Default output
PerformanceInfo	Information about the high-performance computing environment	Default output
RemovalCandidates	Candidates for removal at step	SELECTION DETAILS=ALL | STEPS
ResponseProfile	Response categories and the category that is modeled in models for binary and multinomial data	Default output
SelectedEffects	List of effects that are selected to be included in model	SELECTION
SelectionDetails	Details about model selection, including fit statistics by step	SELECTION DETAILS=ALL
SelectionInfo	Information about the settings for model selection	SELECTION
SelectionReason	Reason why the particular model was selected	SELECTION
SelectionSummary	Summary information about model selection steps	SELECTION
StopReason	Reason for termination of model selection	SELECTION
Timing	Absolute and relative times for tasks performed by the procedure	PERFORMANCE DETAILS
ZeroParameterEstimates	Solutions for the parameter estimates that are associated with effects in ZEROMODEL statements	ZEROMODEL

Examples: HPGENSELECT Procedure

Example 7.1: Model Selection

The following HPGENSELECT statements examine the same data that is used in the section "Getting Started: HPGENSELECT Procedure" on page 225, but they request model selection via the forward selection technique. Model effects are added in the order of their significance until no more effects make a significant improvement of the current model. The DETAILS=ALL option in the SELECTION statement requests that all tables that are related to model selection be produced.

The data set getStarted is shown in the section "Getting Started: HPGENSELECT Procedure" on page 225. It contains 100 observations on a count response variable (Y), a continuous variable (Total) to be used in Example 7.3, and five categorical variables (C1–C5), each of which has four numerical levels.

A log-linked Poisson regression model is specified by using classification effects for variables C1–C5. The following statements request model selection by using the forward selection method:

```
proc hpgenselect data=getStarted;
   class C1-C5;
   model Y = C1-C5 / Distribution=Poisson;
   selection method=forward details=all;
run;
```

The model selection tables are shown in Output 7.1.1 through Output 7.1.3.

The "Selection Information" table in Output 7.1.1 summarizes the settings for the model selection. Effects are added to the model only if they produce a significant improvement as judged by comparing the *p*-value of a score test to the entry significance level (SLE), which is 0.05 by default. The forward selection stops when no effect outside the model meets this criterion.

Output 7.1.1 Selection Information

The HPGENSELECT Procedure

Selection Information	
Selection Method	Forward
Select Criterion	Significance Level
Stop Criterion	Significance Level
Effect Hierarchy Enforced	None
Entry Significance Level (SLE)	0.05
Stop Horizon	1

The "Selection Summary" table in Output 7.1.2 shows the effects that were added to the model and their significance level. Step 0 refers to the null model that contains only an intercept. In the next step, effect C2 made the most significant contribution to the model among the candidate effects ($p < 0.0001$). In step 2, the most significant contribution when adding an effect to a model that contains the intercept and C2 was made by C5. In step 3, the variable C1 ($p = 0.0496$) was added. In the subsequent step, no effect could be added to the model that would produce a p-value less than 0.05, so variable selection stops.

Output 7.1.2 Selection Summary Information

The HPGENSELECT Procedure

Selection Summary

Step	Effect Entered	Number Effects In	p Value
0	Intercept	1	.
1	C2	2	<.0001
2	C5	3	<.0001
3	C1	4	0.0496

Selection stopped because no candidate for entry is significant at the 0.05 level.

Selected Effects: Intercept C1 C2 C5

The DETAILS=ALL option produces the "Selection Details" table, which provides fit statistics and the value of the score test chi-square statistic at each step.

Output 7.1.3 Selection Details

Selection Details

Step	Description	Effects In Model	Chi-Square	Pr > ChiSq	-2 LogL	AIC	AICC	BIC
0	Initial Model	1			350.193	352.193	352.234	354.798
1	C2 entered	2	25.7340	<.0001	324.611	332.611	333.032	343.032
2	C5 entered	3	23.0291	<.0001	303.580	317.580	318.798	335.817
3	C1 entered	4	7.8328	0.0496	295.263	315.263	317.735	341.315

Output 7.1.4 displays information about the selected model. Notice that the –2 log likelihood value in the "Fit Statistics" table is larger than the value for the full model in Figure 7.7. This is expected because the selected model contains only a subset of the parameters. Because the selected model is more parsimonious than the full model, the information criteria AIC, AICC and BIC are smaller than in the full model, indicating a better fit.

Output 7.1.4 Fit Statistics

Fit Statistics	
-2 Log Likelihood	295.26
AIC (smaller is better)	315.26
AICC (smaller is better)	317.74
BIC (smaller is better)	341.31
Pearson Chi-Square	85.0656
Pearson Chi-Square/DF	0.9452

The parameter estimates of the selected model are given in Output 7.1.5. Notice that the effects are listed in the "Parameter Estimates" table in the order in which they were specified in the MODEL statement and not in the order in which they were added to the model.

Output 7.1.5 Parameter Estimates

Parameter Estimates					
Parameter	**DF**	**Estimate**	**Standard Error**	**Chi-Square**	**Pr > ChiSq**
Intercept	1	0.775498	0.242561	10.2216	0.0014
C1 0	1	-0.211240	0.207209	1.0393	0.3080
C1 1	1	-0.685575	0.255713	7.1879	0.0073
C1 2	1	-0.127612	0.203663	0.3926	0.5309
C1 3	0	0	.	.	.
C2 0	1	0.958378	0.239731	15.9817	<.0001
C2 1	1	0.738529	0.237098	9.7024	0.0018
C2 2	1	0.211075	0.255791	0.6809	0.4093
C2 3	0	0	.	.	.
C5 0	1	-0.825545	0.214054	14.8743	0.0001
C5 1	1	-0.697611	0.202607	11.8555	0.0006
C5 2	1	-0.566706	0.213961	7.0153	0.0081
C5 3	0	0	.	.	.

Example 7.2: Modeling Binomial Data

If $Y_1, \cdots, Y_n$ are independent binary (Bernoulli) random variables that have common success probability π, then their sum is a binomial random variable. In other words, a binomial random variable that has parameters n and π can be generated as the sum of n Bernoulli(π) random experiments. The HPGENSELECT procedure uses a special syntax to express data in binomial form: the *events/trials* syntax.

Consider the following data, taken from Cox and Snell (1989, pp. 10–11), of the number, r, of ingots not ready for rolling, out of n tested, for a number of combinations of heating time and soaking time.

```
data Ingots;
   input Heat Soak r n @@;
   Obsnum= _n_;
   datalines;
7 1.0 0 10  14 1.0 0 31  27 1.0 1 56  51 1.0 3 13
7 1.7 0 17  14 1.7 0 43  27 1.7 4 44  51 1.7 0  1
7 2.2 0  7  14 2.2 2 33  27 2.2 0 21  51 2.2 0  1
7 2.8 0 12  14 2.8 0 31  27 2.8 1 22  51 4.0 0  1
7 4.0 0  9  14 4.0 0 19  27 4.0 1 16
;
```

If each test is carried out independently and if for a particular combination of heating and soaking time there is a constant probability that the tested ingot is not ready for rolling, then the random variable r follows a Binomial(n, π) distribution, where the success probability π is a function of heating and soaking time.

The following statements show the use of the events/trials syntax to model the binomial response. The *events* variable in this situation is r (the number of ingots not ready for rolling), and the *trials* variable is n (the number of ingots tested). The dependency of the probability of not being ready for rolling is modeled as a function of heating time, soaking time, and their interaction. The OUTPUT statement stores the linear predictors and the predicted probabilities in the Out data set along with the ID variable.

```
proc hpgenselect data=Ingots;
   model r/n = Heat Soak Heat*Soak / dist=Binomial;
   id Obsnum;
   output out=Out xbeta predicted=Pred;
run;
```

The "Performance Information" table in Output 7.2.1 shows that the procedure executes in single-machine mode.

Output 7.2.1 Performance Information

The HPGENSELECT Procedure

Performance Information	
Execution Mode	Single-Machine
Number of Threads	4

The "Model Information" table shows that the data are modeled as binomially distributed with a logit link function (Output 7.2.2). This is the default link function in the HPGENSELECT procedure for binary and binomial data. The procedure uses a ridged Newton-Raphson algorithm to estimate the parameters of the model.

Output 7.2.2 Model Information and Number of Observations

Model Information	
Data Source	WORK.INGOTS
Response Variable (Events)	r
Response Variable (Trials)	n
Distribution	Binomial
Link Function	Logit
Optimization Technique	Newton-Raphson with Ridging

Number of Observations Read	19
Number of Observations Used	19
Number of Events	12
Number of Trials	387

The second table in Output 7.2.2 shows that all 19 observations in the data set were used in the analysis and that the total number of events and trials equal 12 and 387, respectively. These are the sums of the variables r and n across all observations.

Output 7.2.3 displays the "Dimensions" table for the model. There are four columns in the design matrix of the model (the **X** matrix); they correspond to the intercept, the Heat effect, the Soak effect, and the interaction of the Heat and Soak effects. The model is nonsingular, because the rank of the crossproducts matrix equals the number of columns in **X**. All parameters are estimable and participate in the optimization.

Output 7.2.3 Dimensions in Binomial Logistic Regression

Dimensions	
Number of Effects	4
Number of Parameters	4
Columns in X	4

Output 7.2.4 displays the "Fit Statistics" table for this run. Evaluated at the converged estimates, –2 times the value of the log-likelihood function equals 27.9569. Further fit statistics are also given, all of them in "smaller is better" form. The AIC, AICC, and BIC criteria are used to compare non-nested models and to penalize the model fit for the number of observations and parameters. The –2 log-likelihood value can be used to compare nested models by way of a likelihood ratio test.

Output 7.2.4 Fit Statistics

Fit Statistics	
-2 Log Likelihood	27.9569
AIC (smaller is better)	35.9569
AICC (smaller is better)	38.8140
BIC (smaller is better)	39.7346
Pearson Chi-Square	13.4350
Pearson Chi-Square/DF	0.8957

The "Parameter Estimates" table in Output 7.2.5 displays the estimates and standard errors of the model effects.

Output 7.2.5 Parameter Estimates

Parameter Estimates					
Parameter	DF	Estimate	Standard Error	Chi-Square	Pr > ChiSq
Intercept	1	-5.990191	1.666622	12.9183	0.0003
Heat	1	0.096339	0.047067	4.1896	0.0407
Soak	1	0.299574	0.755068	0.1574	0.6916
Heat*Soak	1	-0.008840	0.025319	0.1219	0.7270

You can construct the prediction equation of the model from the "Parameter Estimates" table. For example, an observation with Heat equal to 14 and Soak equal to 1.7 has linear predictor

$$\widehat{\eta} = -5.9902 + 0.09634 \times 14 + 0.2996 \times 1.7 - 0.00884 \times 14 \times 7 = -4.34256$$

The probability that an ingot with these characteristics is not ready for rolling is

$$\widehat{\pi} = \frac{1}{1 + \exp\{-(-4.34256)\}} = 0.01284$$

The OUTPUT statement computes these linear predictors and probabilities and stores them in the Out data set. This data set also contains the ID variable, which is used by the following statements to attach the covariates to these statistics. Output 7.2.6 shows the probability that an ingot with Heat equal to 14 and Soak equal to 1.7 is not ready for rolling.

```
data Out;
   merge Out Ingots;
   by Obsnum;
proc print data=Out;
   where Heat=14 & Soak=1.7;
run;
```

Output 7.2.6 Predicted Probability for Heat=14 and Soak=1.7

Obs	Obsnum	Pred	Xbeta	Heat	Soak	r	n
6	6	0.012836	-4.34256	14	1.7	0	43

Binomial data are a form of grouped binary data where "successes" in the underlying Bernoulli trials are totaled. You can thus expand data for which you use the events/trials syntax and fit them with techniques for binary data.

The following DATA step expands the Ingots data set (which has 12 events in 387 trials) into a binary data set that has 387 observations.

```
data Ingots_binary;
   set Ingots;
   do i=1 to n;
     if i <= r then Y=1; else Y = 0;
     output;
   end;
run;
```

The following HPGENSELECT statements fit the model by using Heat effect, Soak effect, and their interaction to the binary data set. The **event='1'** response-variable option in the MODEL statement ensures that the HPGENSELECT procedure models the probability that the variable Y takes on the value '1'.

```
proc hpgenselect data=Ingots_binary;
   model Y(event='1') = Heat Soak Heat*Soak / dist=Binary;
run;
```

Output 7.2.7 displays the "Performance Information," "Model Information," "Number of Observations," and the "Response Profile" tables. The data are now modeled as binary (Bernoulli distributed) by using a logit link function. The "Response Profile" table shows that the binary response breaks down into 375 observations where Y equals 0 and 12 observations where Y equals 1.

Output 7.2.7 Model Information in Binary Model

The HPGENSELECT Procedure

Performance Information	
Execution Mode	Single-Machine
Number of Threads	4

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

Model Information	
Data Source	WORK.INGOTS_BINARY
Response Variable	Y
Distribution	Binary
Link Function	Logit
Optimization Technique	Newton-Raphson with Ridging

Number of Observations Read	387
Number of Observations Used	387

Response Profile		
Ordered Value	**Y**	**Total Frequency**
1	0	375
2	1	12

You are modeling the probability that Y='1'.

Output 7.2.8 displays the parameter estimates. These results match those in Output 7.2.5.

Output 7.2.8 Parameter Estimates

Parameter Estimates

Parameter	DF	Estimate	Standard Error	Chi-Square	Pr > ChiSq
Intercept	1	-5.990191	1.666622	12.9183	0.0003
Heat	1	0.096339	0.047067	4.1896	0.0407
Soak	1	0.299574	0.755068	0.1574	0.6916
Heat*Soak	1	-0.008840	0.025319	0.1219	0.7270

Example 7.3: Tweedie Model

The following HPGENSELECT statements examine the data set getStarted used in the section "Getting Started: HPGENSELECT Procedure" on page 225, but they request that a Tweedie model be fit by using the continuous variable Total as the response instead of the count variable Y. The following statements fit a log-linked Tweedie model to these data by using classification effects for variables C1–C5. In an insurance underwriting context, Y represents the total number of claims in each category that is defined by C1–C5, and Total represents the total cost of the claims (that is, the sum of costs for individual claims). The CODE statement requests that a text file named "Scoring Parameters.txt" be created. This file contains a SAS program that contains information from the model that allows scoring of a new data set based on the parameter estimates from the current model.

```
proc hpgenselect data=getStarted;
   class C1-C5;
   model Total = C1-C5 / Distribution=Tweedie Link=Log;
   code File='ScoringParameters.txt';
run;
```

The "Optimizations Stage Details" table in Output 7.3.1 shows the stages used in computing the maximum likelihood estimates of the parameters of the Tweedie model. Stage 1 uses quasi-likelihood and all of the data to compute starting values for stage 2, which uses all of the data and the Tweedie log-likelihood to compute the final estimates.

Output 7.3.1 Optimization Stage Details

The HPGENSELECT Procedure

Optimization Stage Details

Optimization Stage	Optimization Type	Sampling Percentage	Observations Used
1	Quasilikelihood	100.00	100
2	Full Likelihood	100.00	100

The "Parameter Estimates" table in Output 7.3.2 shows the resulting regression model parameter estimates, the estimated Tweedie dispersion parameter, and the estimated Tweedie power.

Output 7.3.2 Parameter Estimates

Parameter Estimates

Parameter	DF	Estimate	Standard Error	Chi-Square	Pr > ChiSq
Intercept	1	3.888904	0.435325	79.8044	<.0001
C1 0	1	-0.072400	0.240613	0.0905	0.7635
C1 1	1	-1.358456	0.324363	17.5400	<.0001
C1 2	1	0.154711	0.237394	0.4247	0.5146
C1 3	0	0	.	.	.
C2 0	1	1.350591	0.289897	21.7050	<.0001
C2 1	1	1.159242	0.275459	17.7106	<.0001
C2 2	1	0.033921	0.303204	0.0125	0.9109
C2 3	0	0	.	.	.
C3 0	1	-0.217763	0.272474	0.6387	0.4242
C3 1	1	-0.289425	0.259751	1.2415	0.2652
C3 2	1	-0.131961	0.276723	0.2274	0.6335
C3 3	0	0	.	.	.
C4 0	1	-0.258069	0.288840	0.7983	0.3716
C4 1	1	-0.057042	0.287566	0.0393	0.8428
C4 2	1	0.219697	0.272064	0.6521	0.4194
C4 3	0	0	.	.	.
C5 0	1	-1.314657	0.257806	26.0038	<.0001
C5 1	1	-0.996980	0.236881	17.7138	<.0001
C5 2	1	-0.481185	0.235614	4.1708	0.0411
C5 3	0	0	.	.	.
Dispersion	1	5.296966	0.773401	.	.
Power	1	1.425625	0.048981	.	.

Now suppose you want to compute predicted values for some different data. If $\mathbf{x}$ is a vector of explanatory variables that might not be in the original data and $\hat{\boldsymbol{\beta}}$ is the vector of estimated regression parameters from the model, then $\mu = g^{-1}(\mathbf{x}'\hat{\boldsymbol{\beta}})$ is the predicted value of the mean, where g is the log link function in this case.

The following data contain new values of the regression variables C1–C5, from which you can compute predicted values based on information in the SAS program that is created by the CODE statement. This is called *scoring* the new data set.

```
data ScoringData;
   input C1-C5;
   datalines;
3 3 1 0 2
1 1 2 2 0
3 2 2 2 0
1 1 2 3 2
1 1 2 3 3
3 1 1 0 1
0 2 1 0 0
2 1 3 1 3
3 2 3 2 0
3 0 2 0 1
;
```

The following SAS DATA step creates the new data set Scores, which contains a variable P_Total that represents the predicted values of Total, along with the variables C1–C5. The resulting data are shown in Output 7.3.3.

```
data Scores;
   set ScoringData;
   %inc 'ScoringParameters.txt';
;
proc print data=Scores;
run;
```

Output 7.3.3 Predicted Values for Scoring Data

Obs	C1	C2	C3	C4	C5	P_Total
1	3	3	1	0	2	17.465
2	1	1	2	2	0	11.737
3	3	2	2	2	0	14.819
4	1	1	2	3	2	21.683
5	1	1	2	3	3	35.083
6	3	1	1	0	1	33.237
7	0	2	1	0	0	7.303
8	2	1	3	1	3	171.711
9	3	2	3	2	0	16.909
10	3	0	2	0	1	47.110

References

Akaike, H. (1974), "A New Look at the Statistical Model Identification," *IEEE Transactions on Automatic Control*, AC-19, 716–723.

Burnham, K. P. and Anderson, D. R. (1998), *Model Selection and Inference: A Practical Information-Theoretic Approach*, New York: Springer-Verlag.

Cox, D. R. and Snell, E. J. (1989), *The Analysis of Binary Data*, 2nd Edition, London: Chapman & Hall.

Dennis, J. E., Gay, D. M., and Welsch, R. E. (1981), "An Adaptive Nonlinear Least-Squares Algorithm," *ACM Transactions on Mathematical Software*, 7, 348–368.

Dennis, J. E. and Mei, H. H. W. (1979), "Two New Unconstrained Optimization Algorithms Which Use Function and Gradient Values," *Journal of Optimization Theory and Applications*, 28, 453–482.

Dunn, P. K. and Smyth, G. K. (2005), "Series Evaluation of Tweedie Exponential Dispersion Model Densities," *Statistics and Computing*, 15, 267–280.

Dunn, P. K. and Smyth, G. K. (2008), "Series Evaluation of Tweedie Exponential Dispersion Model Densities by Fourier Inversion," *Statistics and Computing*, 18, 73–86.

Eskow, E. and Schnabel, R. B. (1991), "Algorithm 695: Software for a New Modified Cholesky Factorization," *ACM Transactions on Mathematical Software*, 17, 306–312.

Fletcher, R. (1987), *Practical Methods of Optimization*, 2nd Edition, Chichester, UK: John Wiley & Sons.

Frees, E. W. (2010), *Regression Modeling with Actuarial and Financial Applications*, Cambridge: Cambridge University Press.

Gay, D. M. (1983), "Subroutines for Unconstrained Minimization," *ACM Transactions on Mathematical Software*, 9, 503–524.

Hastie, T. J., Tibshirani, R. J., and Friedman, J. H. (2001), *The Elements of Statistical Learning*, New York: Springer-Verlag.

Hurvich, C. M. and Tsai, C.-L. (1989), "Regression and Time Series Model Selection in Small Samples," *Biometrika*, 76, 297–307.

McCullagh, P. and Nelder, J. A. (1989), *Generalized Linear Models*, 2nd Edition, London: Chapman & Hall.

Moré, J. J. and Sorensen, D. C. (1983), "Computing a Trust-Region Step," *SIAM Journal on Scientific and Statistical Computing*, 4, 553–572.

Schwarz, G. (1978), "Estimating the Dimension of a Model," *Annals of Statistics*, 6, 461–464.

Chapter 8
The HPLMIXED Procedure

Contents

Overview: HPLMIXED Procedure **286**
PROC HPLMIXED Features 287
Notation for the Mixed Model 288
PROC HPLMIXED Contrasted with Other SAS Procedures 289
Getting Started: HPLMIXED Procedure **289**
Mixed Model Analysis of Covariance with Many Groups 289
Syntax: HPLMIXED Procedure **292**
PROC HPLMIXED Statement 293
CLASS Statement 298
ID Statement 298
MODEL Statement 299
OUTPUT Statement 300
PARMS Statement 301
PERFORMANCE Statement 303
RANDOM Statement 303
REPEATED Statement 309
Details: HPLMIXED Procedure **310**
Linear Mixed Models Theory 310
Matrix Notation 310
Formulation of the Mixed Model 311
Estimating Covariance Parameters in the Mixed Model 316
Estimating Fixed and Random Effects in the Mixed Model 317
Statistical Properties 317
Computational Method 318
Distributed Computing 318
Multithreading 318
Displayed Output 319
Performance Information 319
Model Information 319
Class Level Information 320
Dimensions 320
Number of Observations 320
Optimization Information 320
Iteration History 320
Convergence Status 321
Covariance Parameter Estimates 321

Fit Statistics 321
Timing Information 322
ODS Table Names 322
Examples: HPLMIXED Procedure **323**
Example 8.1: Computing BLUPs for a Large Number of Subjects 323
References **325**

Overview: HPLMIXED Procedure

The HPLMIXED procedure fits a variety of mixed linear models to data and enables you to use these fitted models to make statistical inferences about the data. A *mixed linear model* is a generalization of the standard linear model used in the GLM procedure in SAS/STAT software; the generalization is that the data are permitted to exhibit correlation and nonconstant variability. Therefore, the mixed linear model provides you with the flexibility of modeling not only the means of your data (as in the standard linear model) but also their variances and covariances.

The primary assumptions underlying the analyses performed by PROC HPLMIXED are as follows:

- The data are normally distributed (Gaussian).
- The means (expected values) of the data are linear in terms of a certain set of parameters.
- The variances and covariances of the data are in terms of a different set of parameters, and they exhibit a structure that matches one of those available in PROC HPLMIXED.

Because Gaussian data can be modeled entirely in terms of their means and variances/covariances, the two sets of parameters in a mixed linear model actually specify the complete probability distribution of the data. The parameters of the mean model are referred to as *fixed-effects parameters*, and the parameters of the variance-covariance model are referred to as *covariance parameters*.

The fixed-effects parameters are associated with known explanatory variables, as in the standard linear model. These variables can be either qualitative (as in the traditional analysis of variance) or quantitative (as in standard linear regression). However, the covariance parameters are what distinguishes the mixed linear model from the standard linear model.

The need for covariance parameters arises quite frequently in applications; the following scenarios are the most typical:

- The experimental units on which the data are measured can be grouped into clusters, and the data from a common cluster are correlated. This scenario can be generalized to include one set of clusters nested within another. For example, if students are the experimental unit, they can be clustered into classes, which in turn can be clustered into schools. Each level of this hierarchy can introduce an additional source of variability and correlation.
- Repeated measurements are taken on the same experimental unit, and these repeated measurements are correlated or exhibit variability that changes. This scenario occurs in longitudinal studies, where repeated measurements are taken over time. Alternatively, the repeated measures could be spatial or multivariate in nature.

PROC HPLMIXED provides a variety of covariance structures to handle these two scenarios. The most common covariance structures arise from the use of *random-effects parameters*, which are additional unknown random variables that are assumed to affect the variability of the data. The variances of the random-effects parameters, commonly known as *variance components*, become the covariance parameters for this particular structure. Traditional mixed linear models contain both fixed- and random-effects parameters; in fact, it is the combination of these two types of effects that led to the name *mixed model*. PROC HPLMIXED fits not only these traditional variance component models but also numerous other covariance structures.

PROC HPLMIXED fits the structure you select to the data by using the method of *restricted maximum likelihood (REML)*, also known as *residual maximum likelihood*. It is here that the Gaussian assumption for the data is exploited.

PROC HPLMIXED runs in either single-machine mode or distributed mode.

NOTE: Distributed mode requires SAS High-Performance Statistics.

PROC HPLMIXED Features

PROC HPLMIXED provides easy accessibility to numerous mixed linear models that are useful in many common statistical analyses.

Here are some basic features of PROC HPLMIXED:

- covariance structures, including variance components, compound symmetry, unstructured, AR(1), Toeplitz, and factor analytic
- MODEL, RANDOM, and REPEATED statements for model specification as in the HPLMIXED procedure
- appropriate standard errors, t tests, and F tests for all specified estimable linear combinations of fixed and random effects
- a subject effect that enables blocking
- REML and ML (maximum likelihood) estimation methods implemented with a variety of optimization algorithms
- capacity to handle unbalanced data
- special dense and sparse algorithms that take advantage of distributed and multicore computing environments

Because the HPLMIXED procedure is a high-performance analytical procedure, it also does the following:

- enables you to run in distributed mode on a cluster of machines that distribute the data and the computations
- enables you to run in single-machine mode on the server where SAS is installed
- exploits all the available cores and concurrent threads, regardless of execution mode

For more information, see the section "Processing Modes" on page 10 in Chapter 3, "Shared Concepts and Topics."

PROC HPLMIXED uses the Output Delivery System (ODS), a SAS subsystem that provides capabilities for displaying and controlling the output from SAS procedures. ODS enables you to convert any output from PROC HPLMIXED into a SAS data set. See the section "ODS Table Names" on page 322.

Notation for the Mixed Model

This section introduces the mathematical notation used throughout this chapter to describe the mixed linear model and assumes familiarity with basic matrix algebra (for an overview, see Searle 1982). A more detailed description of the mixed model is contained in the section "Linear Mixed Models Theory" on page 310.

A statistical model is a mathematical description of how data are generated. The standard linear model, as used by the GLM procedure, is one of the most common statistical models:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

In this expression, $\mathbf{y}$ represents a vector of observed data, $\boldsymbol{\beta}$ is an unknown vector of fixed-effects parameters with a known design matrix $\mathbf{X}$, and $\boldsymbol{\epsilon}$ is an unknown random error vector that models the statistical noise around $\mathbf{X}\boldsymbol{\beta}$. The focus of the standard linear model is to model the mean of $\mathbf{y}$ by using the fixed-effects parameters $\boldsymbol{\beta}$. The residual errors $\boldsymbol{\epsilon}$ are assumed to be independent and identically distributed Gaussian random variables with mean 0 and variance σ^2.

The mixed model generalizes the standard linear model as follows:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\boldsymbol{\gamma} + \boldsymbol{\epsilon}$$

Here, $\boldsymbol{\gamma}$ is an unknown vector of random-effects parameters with a known design matrix $\mathbf{Z}$, and $\boldsymbol{\epsilon}$ is an unknown random error vector whose elements are no longer required to be independent and homogeneous.

To further develop this notion of variance modeling, assume that $\boldsymbol{\gamma}$ and $\boldsymbol{\epsilon}$ are Gaussian random variables that are uncorrelated, have expectations $\mathbf{0}$, and have variances $\mathbf{G}$ and $\mathbf{R}$, respectively. The variance of $\mathbf{y}$ is thus

$$\mathbf{V} = \mathbf{Z}\mathbf{G}\mathbf{Z}' + \mathbf{R}$$

Note that when $\mathbf{R} = \sigma^2\mathbf{I}$ and $\mathbf{Z} = \mathbf{0}$, the mixed model reduces to the standard linear model.

You can model the variance of the data $\mathbf{y}$ by specifying the structure of $\mathbf{Z}$, $\mathbf{G}$, and $\mathbf{R}$. The model matrix $\mathbf{Z}$ is set up in the same fashion as $\mathbf{X}$, the model matrix for the fixed-effects parameters. For $\mathbf{G}$ and $\mathbf{R}$, you must select some covariance structure. Possible covariance structures include the following:

- variance components
- compound symmetry (common covariance plus diagonal)
- unstructured (general covariance)
- autoregressive
- spatial

- general linear
- factor analytic

By appropriately defining the model matrices **X** and **Z** in addition to the covariance structure matrices **G** and **R**, you can perform numerous mixed model analyses.

PROC HPLMIXED Contrasted with Other SAS Procedures

The RANDOM and REPEATED statements of the HPLMIXED procedure follow the convention of the same statements in the MIXED procedure in SAS/STAT software. For information about how these statements differ from RANDOM and REPEATED statements in the MIXED procedure, see the documentation for the MIXED procedure in the *SAS/STAT User's Guide*.

The GLIMMIX procedure in SAS/STAT software fits generalized linear mixed models. Linear mixed models—where the data are normally distributed, given the random effects—are in the class of generalized linear mixed models. Therefore, PROC GLIMMIX accommodates nonnormal data with random effects.

Generalized linear mixed models have intrinsically nonlinear features because a nonlinear mapping (the link function) connects the conditional mean of the data (given the random effects) to the explanatory variables. The NLMIXED procedure also accommodates nonlinear structures in the conditional mean, but places no restrictions on the nature of the nonlinearity.

The HPMIXED procedure in SAS/STAT software is also termed a "high-performance" procedure, but it does not follow the general pattern of high-performance analytical procedures. The HPMIXED procedure does not take advantage of distributed or multicore computing environments; it derives high performance from applying sparse techniques to solving the mixed model equations. The HPMIXED procedure fits a small subset of the statistical models you can fit with the MIXED or HPLMIXED procedures and is particularly suited for problems in which the $[\mathbf{XZ}]'[\mathbf{XZ}]$ crossproducts matrix is sparse.

The HPLMIXED procedure employs algorithms that are specialized for distributed and multicore computing environments. The HPLMIXED procedure does not support BY processing.

Getting Started: HPLMIXED Procedure

Mixed Model Analysis of Covariance with Many Groups

Suppose you are an educational researcher who studies how student scores on math tests change over time. Students are tested four times, and you want to estimate the overall rise or fall, accounting for correlation between test response behaviors of students in the same neighborhood and school. One way to model this correlation is by using a random-effects analysis of covariance, where the scores for students from the same neighborhood and school are all assumed to share the same quadratic mean test response function, the parameters of this response function being random. The following statements simulate a data set with this structure:

```
data SchoolSample;
   do SchoolID = 1 to 300;
      do nID = 1 to 25;
         Neighborhood = (SchoolID-1)*5 + nId;
         bInt   = 5*ranuni(1);
         bTime  = 5*ranuni(1);
         bTime2 =   ranuni(1);
         do sID = 1 to 2;
            do Time = 1 to 4;
               Math = bInt + bTime*Time + bTime2*Time*Time + rannor(2);
               output;
               end;
            end;
         end;
      end;
run;
```

In this data, there are 300 schools and about 1,500 neighborhoods; neighborhoods are associated with more than one school and vice versa. The following statements use PROC HPLMIXED to fit a mixed analysis of covariance model to this data. To run these statements successfully, you need to set the macro variables GRIDHOST and GRIDINSTALLLOC to resolve to appropriate values, or you can replace the references to macro variables with appropriate values.

```
proc hplmixed data=SchoolSample;
   performance host="&GRIDHOST" install="&GRIDINSTALLLOC" nodes=20;
   class Neighborhood SchoolID;
   model Math = Time Time*Time / solution;
   random   int Time Time*Time / sub=Neighborhood(SchoolID) type=un;
run;
```

This model fits a quadratic mean response model with an unstructured covariance matrix to model the covariance between the random parameters of the response model. With 7,500 neighborhood/school combinations, this model can be computationally daunting to fit, but PROC HPLMIXED finishes quickly and displays the results shown in Figure 8.1.

Figure 8.1 Mixed Model Analysis of Covariance

The HPLMIXED Procedure

Performance Information	
Host Node	YourGridHost
Execution Mode	Distributed
Number of Compute Nodes	20
Number of Threads per Node	24

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

Figure 8.1 *continued*

Model Information	
Data Set	WORK.SCHOOLSAMPLE
Dependent Variable	Math
Covariance Structure	Unstructured
Subject Effect	Neighborho(SchoolID)
Estimation Method	Restricted Maximum Likelihood
Residual Variance Method	Profile
Fixed Effects SE Method	Model-Based
Degrees of Freedom Method	Residual

Class Level Information		
Class	**Levels**	**Values**
Neighborhood	1520	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 ...
SchoolID	300	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 ...

Dimensions	
Covariance Parameters	7
Columns in X	3
Columns in Z per Subject	3
Subjects	7500
Max Obs per Subject	8

Number of Observations Read	60000
Number of Observations Used	60000
Number of Observations Not Used	0
Number of Observations Swapped	52500
Number of Subjects Needing Swap	7500

Optimization Information	
Optimization Technique	Newton-Raphson with Ridging
Parameters in Optimization	6
Lower Boundaries	3
Upper Boundaries	0
Starting Values From	Data

Iteration History				
Iteration	**Evaluations**	**Objective Function**	**Change**	**Max Gradient**
0	2	225641.67142		6.741E-8

Convergence criterion (ABSGCONV=0.00001) satisfied.

Figure 8.1 *continued*

Covariance Parameter Estimates		
Cov Parm	**Subject**	**Estimate**
UN(1,1)	Neighborho(SchoolID)	2.0902
UN(2,1)	Neighborho(SchoolID)	0.000349
UN(2,2)	Neighborho(SchoolID)	2.0517
UN(3,1)	Neighborho(SchoolID)	0.01448
UN(3,2)	Neighborho(SchoolID)	0.01599
UN(3,3)	Neighborho(SchoolID)	0.08047
Residual		1.0083

Fit Statistics	
-2 Res Log Likelihood	225642
AIC (Smaller is Better)	225656
AICC (Smaller is Better)	225656
BIC (Smaller is Better)	225704

Solution for Fixed Effects					
Effect	**Estimate**	**Standard Error**	**DF**	**t Value**	**Pr > |t|**
Intercept	2.5070	0.02828	6E4	88.66	<.0001
Time	2.5124	0.02659	6E4	94.48	<.0001
Time*Time	0.5010	0.005247	6E4	95.48	<.0001

Syntax: HPLMIXED Procedure

The following statements are available in PROC HPLMIXED.

PROC HPLMIXED < *options* > **;**
CLASS *variables* **;**
ID *variables* **;**
MODEL *dependent* **=** < *fixed-effects* > < / *options* > **;**
OUTPUT < **OUT=***SAS-data-set* > < *keyword* < **=***name* > >. . . < *keyword* < **=***name* > > < / *options* > **;**
RANDOM *random-effects* < / *options* > **;**
REPEATED *repeated-effect* < / *options* > **;**
PARMS < **(***value-list***)** . . . > < / *options* > **;**
PERFORMANCE < *options* > **;**

Items within angle brackets (< >) are optional. The RANDOM statement can appear multiple times. Other statements can appear only once.

The PROC HPLMIXED and MODEL statements are required, and the MODEL statement must appear after the CLASS statement if a CLASS statement is included. The RANDOM statement must follow the MODEL statement.

Table 8.1 summarizes the basic functions and important options of the PROC HPLMIXED statements. The syntax of each statement in Table 8.1 is described in the following sections in alphabetical order after the description of the PROC HPLMIXED statement.

Table 8.1 Summary of PROC HPLMIXED Statements

Statement	Description	Important Options
PROC HPLMIXED	Invokes the procedure	DATA= specifies the input data set; METHOD= specifies the estimation method.
CLASS	Declares qualitative variables that create indicator variables in **X** and **Z** matrices.	None
ID	Lists additional variables to be included in predicted values tables	None
MODEL	Specifies dependent variable and fixed effects, setting up **X**	S requests a solution for fixed-effects parameters.
RANDOM	Specifies random effects, setting up **Z** and **G**	SUBJECT= creates block-diagonality; TYPE= specifies the covariance structure; S requests a solution for the random effects.
REPEATED	Sets up **R**	SUBJECT= creates block-diagonality; TYPE= specifies the covariance structure.
OUTPUT	Creates a data set that contains observationwise statistics	ALLSTATS requests that all statistics be computed.
PARMS	Specifies a grid of initial values for the covariance parameters	HOLD= and NOITER hold the covariance parameters or their ratios constant; PARMSDATA= reads the initial values from a SAS data set.
PERFORMANCE	Invokes the distributed computing connection	NODES= specifies the number of nodes to use.

PROC HPLMIXED Statement

PROC HPLMIXED < *options* > **;**

The PROC HPLMIXED statement invokes the procedure. Table 8.2 summarizes important options in the PROC HPLMIXED statement by function. These and other options in the PROC HPLMIXED statement are then described fully in alphabetical order.

Table 8.2 PROC HPLMIXED Statement Options

Option	Description
Basic Options	
DATA=	Specifies the input data set
METHOD=	Specifies the estimation method
NAMELEN=	Limits the length of effect names

Table 8.2 *continued*

Option	Description
BLUP	Computes the best linear unbiased prediction
Options Related to Output	
NOCLPRINT	Suppresses the "Class Level Information" table completely or in parts
MAXCLPRINT=	Specifies the maximum levels of CLASS variables to print
RANKS	Displays the rank of design matrix **X**
Optimization Options	
ABSCONV=	Tunes an absolute function convergence criterion
ABSFCONV=	Tunes an absolute function difference convergence criterion
ABSGCONV=	Tunes the absolute gradient convergence criterion
FCONV=	Tunes the relative function convergence criterion
GCONV=	Tunes the relative gradient convergence criterion
MAXITER=	Chooses the maximum number of iterations in any optimization
MAXFUNC=	Specifies the maximum number of function evaluations in any optimization
MAXTIME=	Specifies the upper limit on seconds of CPU time for any optimization
MINITER=	Specifies the minimum number of iterations in any optimization
TECHNIQUE=	Selects the optimization technique
XCONV=	Tunes the relative parameter convergence criterion

You can specify the following *options* in the PROC HPLMIXED statement.

ABSCONV=*r*

specifies an absolute function convergence criterion. For minimization, termination requires $f(\boldsymbol{\psi}^{(k)}) \leq r$, where $\boldsymbol{\psi}$ is the vector of parameters in the optimization and $f(\cdot)$ is the objective function. The default value of r is the negative square root of the largest double-precision value, which serves only as a protection against overflows.

ABSFCONV=*r*

specifies an absolute function difference convergence criterion. For all techniques except Nelder–Mead simplex (NMSIMP), termination requires a small change of the function value in successive iterations:

$$|f(\boldsymbol{\psi}^{(k-1)}) - f(\boldsymbol{\psi}^{(k)})| \leq r$$

Here, $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization and $f(\cdot)$ is the objective function. The same formula is used for the NMSIMP technique, but $\boldsymbol{\psi}^{(k)}$ is defined as the vertex with the lowest function value and $\boldsymbol{\psi}^{(k-1)}$ is defined as the vertex with the highest function value in the simplex. The default value is $r = 0$.

ABSGCONV=*r*

specifies an absolute gradient convergence criterion. Termination requires the maximum absolute gradient element to be small:

$$\max_j |g_j(\boldsymbol{\psi}^{(k)})| \leq r$$

Here, $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization and $g_j(\cdot)$ is the gradient of the objective function with respect to the j parameter. This criterion is not used by the NMSIMP technique. The default value is r=1E–5.

BLUP< *(suboptions)* >
requests that best linear unbiased predictions (BLUPs) for the random effects be displayed. To use this option, you must also use the PARMS statement to specify fixed values for the covariance parameters, which means that the NOITER option in the PARMS statement will be implied by the BLUP option. Also, the iterations in the ODS Table IterHistory will refer to iterations used to compute the BLUP rather than optimization iterations.

The BLUP solution might be sensitive to the order of observations, and hence to how the data are distributed on the grid. If there are multiple measures of a repeated effect, then the BLUP solution is not unique. If the order of these multiple measures on the grid differs for different runs, then different BLUP solutions will result.

You can specify the following *suboptions*:

ITPRINT=*number* specifies that the iteration history be displayed after every *number* of iterations. The default value is 10, which means the procedure displays the iteration history for every 10 iterations.

MAXITER=*number* specifies the maximum number of iterations allowed. The default value is the number of parameters in the BLUP option plus 2.

TOL=*number* specifies the tolerance value. The default value is the square root of machine precision.

DATA=*SAS-data-set*
names the SAS data set to be used as the input data set. The default is the most recently created data set.

FCONV=*r*
specifies a relative function convergence criterion. For all techniques except NMSIMP, termination requires a small relative change of the function value in successive iterations,

$$\frac{|f(\boldsymbol{\psi}^{(k)}) - f(\boldsymbol{\psi}^{(k-1)})|}{|f(\boldsymbol{\psi}^{(k-1)})|} \leq r$$

Here, $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization and $f(\cdot)$ is the objective function. The same formula is used for the NMSIMP technique, but $\boldsymbol{\psi}^{(k)}$ is defined as the vertex with the lowest function value and $\boldsymbol{\psi}^{(k-1)}$ is defined as the vertex with the highest function value in the simplex.

The default is $r = 10^{-\mathrm{FDIGITS}}$, where FDIGITS is $-\log_{10}(\epsilon)$ and ϵ is the machine precision.

GCONV=*r*
specifies a relative gradient convergence criterion. For all techniques except CONGRA and NMSIMP, termination requires that the normalized predicted function reduction be small,

$$\frac{\mathbf{g}(\boldsymbol{\psi}^{(k)})'[\mathbf{H}^{(k)}]^{-1}\mathbf{g}(\boldsymbol{\psi}^{(k)})}{|f(\boldsymbol{\psi}^{(k)})|} \leq r$$

Here, $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization, $f(\cdot)$ is the objective function, and $\mathbf{g}(\cdot)$ is the gradient. For the CONGRA technique (where a reliable Hessian estimate $\mathbf{H}$ is not available), the following criterion is used:

$$\frac{\| \mathbf{g}(\boldsymbol{\psi}^{(k)}) \|_2^2 \quad \| \mathbf{s}(\boldsymbol{\psi}^{(k)}) \|_2}{\| \mathbf{g}(\boldsymbol{\psi}^{(k)}) - \mathbf{g}(\boldsymbol{\psi}^{(k-1)}) \|_2 \, |f(\boldsymbol{\psi}^{(k)})|} \leq r$$

This criterion is not used by the NMSIMP technique. The default value is r=1E–8.

MAXCLPRINT=*number*

specifies the maximum levels of CLASS variables to print in the ODS table "ClassLevels." The default value is 20. MAXCLPRINT=0 enables you to print all levels of each CLASS variable. However, the option NOCLPRINT takes precedence over MAXCLPRINT.

MAXFUNC=*n*

specifies the maximum number *n* of function calls in the optimization process. The default values are as follows, depending on the optimization technique:

- TRUREG, NRRIDG, NEWRAP: 125
- QUANEW, DBLDOG: 500
- CONGRA: 1,000
- NMSIMP: 3,000

The optimization can terminate only after completing a full iteration. Therefore, the number of function calls that are actually performed can exceed *n*. You can choose the optimization technique with the TECHNIQUE= option.

MAXITER=*n*

specifies the maximum number *n* of iterations in the optimization process. The default values are as follows, depending on the optimization technique:

- TRUREG, NRRIDG, NEWRAP: 50
- QUANEW, DBLDOG: 200
- CONGRA: 400
- NMSIMP: 1,000

These default values also apply when *n* is specified as a missing value. You can choose the optimization technique with the TECHNIQUE= option.

MAXTIME=*r*

specifies an upper limit of *r* seconds of CPU time for the optimization process. The default value is the largest floating-point double representation of your computer. The time specified by the MAXTIME= option is checked only once at the end of each iteration. Therefore, the actual running time can be longer than *r*.

METHOD=REML

METHOD=ML

specifies the estimation method for the covariance parameters. METHOD=REML performs residual (restricted) maximum likelihood; it is the default method. METHOD=ML performs maximum likelihood.

MINITER=*n*

specifies the minimum number of iterations. The default value is 0. If you request more iterations than are actually needed for convergence to a stationary point, the optimization algorithms can behave strangely. For example, the effect of rounding errors can prevent the algorithm from continuing for the required number of iterations.

NAMELEN=*number*

specifies the length to which long effect names are shortened. The minimum value is 20, which is also the default.

NOCLPRINT< =*number* **>**

suppresses the display of the "Class Level Information" table if you do not specify *number*. If you specify *number*, the values of the classification variables are displayed for only those variables whose number of levels is less than *number*. Specifying a *number* helps to reduce the size of the "Class Level Information" table if some classification variables have a large number of levels.

NOPRINT

suppresses the generation of ODS output.

RANKS

displays the rank of design matrix $\mathbf{X}$.

SINGCHOL=*number*

tunes the singularity criterion in Cholesky decompositions. The default is 1E4 times the machine epsilon; this product is approximately 1E–12 on most computers.

SINGSWEEP=*number*

tunes the singularity criterion for sweep operations. The default is 1E4 times the machine epsilon; this product is approximately 1E–12 on most computers.

SINGULAR=*number*

tunes the general singularity criterion applied by the HPLMIXED procedure in sweeps and inversions. The default is 1E4 times the machine epsilon; this product is approximately 1E–12 on most computers.

TECHNIQUE=*keyword*

specifies the optimization technique for obtaining maximum likelihood estimates. You can specify any of the following *keywords*:

CONGRA	performs a conjugate-gradient optimization.
DBLDOG	performs a version of double-dogleg optimization.
NEWRAP	performs a Newton-Raphson optimization combining a line-search algorithm with ridging.
NMSIMP	performs a Nelder-Mead simplex optimization.

NONE	performs no optimization.
NRRIDG	performs a Newton-Raphson optimization with ridging.
QUANEW	performs a dual quasi-Newton optimization.
TRUREG	performs a trust-region optimization.

The default value is TECHNIQUE=NRRIDG.

XCONV=*r*

specifies the relative parameter convergence criterion:

- For all techniques except NMSIMP, termination requires a small relative parameter change in subsequent iterations:

$$\frac{\max_j |\psi_j^{(k)} - \psi_j^{(k-1)}|}{\max(|\psi_j^{(k)}|, |\psi_j^{(k-1)}|)} \leq r$$

- For the NMSIMP technique, the same formula is used, but $\psi_j^{(k)}$ is defined as the vertex with the lowest function value and $\psi_j^{(k-1)}$ is defined as the vertex with the highest function value in the simplex.

The default value is r = 1E–8 for the NMSIMP technique and r = 0 otherwise.

CLASS Statement

CLASS *variables* **;**

The CLASS statement names the classification variables to be used as explanatory variables in the analysis. These variables enter the analysis not through their values, but through levels to which the unique values are mapped. See the section "Levelization of Classification Variables" on page 52 of Chapter 4, "Shared Statistical Concepts" for details about these mappings.

If a CLASS statement is specified, it must precede the MODEL statement in high-performance analytical procedures that support a MODEL statement.

Levels of classification variables are ordered by their external formatted values, except for numeric variables with no explicit format, which are ordered by their unformatted (internal) values.

ID Statement

ID *variables* **;**

The ID statement specifies which variables from the input data set are to be included in the OUT= data set that is created by the OUTPUT statement. If you do not specify an ID statement, then no variables from the input data set are included in the OUT= data set.

MODEL Statement

MODEL *dependent* **=** < *fixed-effects* >< / *options* > **;**

The MODEL statement names a single dependent variable and the fixed effects, which determine the **X** matrix of the mixed model. (For details, see the section "Specification and Parameterization of Model Effects" on page 55 of Chapter 4, "Shared Statistical Concepts".) The MODEL statement is required.

An intercept is included in the fixed-effects model by default. If no fixed effects are specified, only this intercept term is fit. The intercept can be removed by using the NOINT option.

Table 8.3 summarizes options in the MODEL statement. These are subsequently discussed in detail in alphabetical order.

Table 8.3 Summary of Important MODEL Statement Options

Option	Description
Model Building	
NOINT	Excludes the fixed-effect intercept from model
Statistical Computations	
ALPHA=α	Determines the confidence level ($1-\alpha$) for fixed effects
DDFM=	Specifies the method for computing denominator degrees of freedom
Statistical Output	
CL	Displays confidence limits for fixed-effects parameter estimates
SOLUTION	Displays fixed-effects parameter estimates

You can specify the following *options* in the MODEL statement after a slash (/).

ALPHA=*number*

sets the confidence level to be $1-$*number* for each confidence interval of the fixed-effects parameters. The value of *number* must be between 0 and 1; the default is 0.05.

CL

requests that *t*-type confidence limits be constructed for each of the fixed-effects parameter estimates. The confidence level is 0.95 by default; this can be changed with the ALPHA= option.

DDFM=NONE | RESIDUAL

specifies the method for computing the denominator degrees of freedom for the tests of fixed effects.

The DDFM=RESIDUAL option performs all tests by using the residual degrees of freedom, $n - \mathrm{rank}(\mathbf{X})$, where n is the number of observations used. It is the default degrees-of-freedom method.

DDFM=NONE specifies that no denominator degrees of freedom be applied. PROC HPLMIXED then essentially assumes that infinite degrees of freedom are available in the calculation of *p*-values. The *p*-values for *t* tests are then identical to *p*-values that are derived from the standard normal distribution.

In the case of F tests, the p-values equal those of chi-square tests determined as follows: if F_{obs} is the observed value of the F test with l numerator degrees of freedom, then

$$p = \Pr\{F_{l,\infty} > F_{obs}\} = \Pr\{\chi^2_l > lF_{obs}\}$$

NOINT
requests that no intercept be included in the model. (An intercept is included by default.)

SOLUTION

S
requests that a solution for the fixed-effects parameters be produced. Using notation from the section "Linear Mixed Models Theory" on page 310, the fixed-effects parameter estimates are $\widehat{\boldsymbol{\beta}}$ and their approximate standard errors are the square roots of the diagonal elements of $(\mathbf{X}'\widehat{\mathbf{V}}^{-1}\mathbf{X})^{-}$.

Along with the estimates and their approximate standard errors, a t statistic is computed as the estimate divided by its standard error. The Pr > |t| column contains the two-tailed p-value that corresponds to the t statistic and associated degrees of freedom. You can use the CL option to request confidence intervals for all of the parameters; they are constructed around the estimate by using a radius that is the product of the standard error times a percentage point from the t distribution.

OUTPUT Statement

OUTPUT < **OUT=***SAS-data-set* > < *keyword* < **=***name* > > . . . < *keyword* < **=***name* > > < / *options* > **;**

The OUTPUT statement creates a data set that contains predicted values and residual diagnostics, which are computed after the model is fit. The variables in the input data set are *not* included in the output data set, in order to avoid data duplication for large data sets; however, variables that are specified in the ID statement are included. By default, only predicted values are included in the output data set.

If the input data are in distributed form, in which access of data in a particular order cannot be guaranteed, the HPLMIXED procedure copies the distribution or partition key to the output data set so that its contents can be joined with the input data.

For example, suppose that the data set Scores contains the variables Score, Machine, and Person. The following statements fit a model that has fixed machine and random person effects and save the predicted and residual values to the data set IgausOut:

```
proc hplmixed data = Scores;
   class machine person score;
   model score = machine;
   random person;
   output out=igausout pred=p resid=r;
run;
```

You can specify the following syntax element in the OUTPUT statement:

OUT=*SAS-data-set*
specifies the name of the output data set. If the OUT= option is omitted, PROC HPLMIXED uses the DATA*n* convention to name the output data set.

A *keyword* can appear multiple times in the OUTPUT statement. Table 8.4 lists the keywords and the default names that PROC HPLMIXED assigns if you do not specify a *name*. In this table, y denotes the response variable.

Table 8.4 Keywords for Output Statistics

Keyword	Description	Expression	Name
PRED	Linear predictor	$\widehat{\eta} = \mathbf{x}'\widehat{\boldsymbol{\beta}} + \mathbf{z}'\widehat{\boldsymbol{\gamma}}$	Pred
PREDPA	Marginal linear predictor	$\widehat{\eta}_m = \mathbf{x}'\widehat{\boldsymbol{\beta}}$	PredPA
RESIDUAL	Residual	$r = y - \widehat{\eta}$	Resid
RESIDUALPA	Marginal residual	$r_m = y - \widehat{\eta}_m$	ResidPA

The marginal linear predictor and marginal residual are also referred to as the predicted population average (PREDPA) and residual population average (RESIDUALPA), respectively. You can use the following shortcuts to request statistics: PRED for PREDICTED and RESID for RESIDUAL.

You can specify the following *option* in the OUTPUT statement after a slash (/):

ALLSTATS
requests that all statistics be computed. If you do not use a *keyword* to assign a name, PROC HPLMIXED uses the default name.

PARMS Statement

PARMS < *(value-list)*. . . > < / *options* > **;**

The PARMS statement specifies initial values for the covariance parameters, or it requests a grid search over several values of these parameters. You must specify the values in the order in which they appear in the "Covariance Parameter Estimates" table.

The *value-list* specification can take any of several forms:

m	a single value
$m_1, m_2, \ldots, m_n$	several values
m to n	a sequence in which m equals the starting value, n equals the ending value, and the increment equals 1
m to n by i	a sequence in which m equals the starting value, n equals the ending value, and the increment equals i
m_1, m_2 to m_3	mixed values and sequences

You can use the PARMS statement to input known parameters.

If you specify more than one set of initial values, PROC HPLMIXED performs a grid search of the likelihood surface and uses the best point on the grid for subsequent analysis. Specifying a large number of grid points can result in long computing times.

The results from the PARMS statement are the values of the parameters on the specified grid (denoted by CovP1 through CovP*n*), the residual variance (possibly estimated) for models with a residual variance parameter, and various functions of the likelihood.

You can specify the following *options* in the PARMS statement after a slash (/).

HOLD=*all*

EQCONS=*all*

specifies that all parameter values be held to equal the specified values.

For example, the following statement constrains all covariance parameters to equal 5, 3, 2, and 3:

```
parms (5) (3) (2) (3) / hold=all;
```

LOWERB=*value-list*

enables you to specify lower boundary constraints on the covariance parameters. The *value-list* specification is a list of numbers or missing values (.) separated by commas. You must list the numbers in the order that PROC HPLMIXED uses for the covariance parameters, and each number corresponds to the lower boundary constraint. A missing value instructs PROC HPLMIXED to use its default constraint. If you do not specify numbers for all of the covariance parameters, PROC HPLMIXED assumes the remaining ones are missing.

This option is useful when you want to constrain the $\mathbf{G}$ matrix to be positive definite in order to avoid the more computationally intensive algorithms that would be required when $\mathbf{G}$ becomes singular. The corresponding statements for a random coefficients model are as follows:

```
proc hplmixed;
   class person;
   model y = time;
   random int time / type=fa0(2) sub=person;
   parms / lowerb=1e-4,.,1e-4;
run;
```

The TYPE=FA0(2) structure specifies a Cholesky root parameterization for the 2×2 unstructured blocks in $\mathbf{G}$. This parameterization ensures that the $\mathbf{G}$ matrix is nonnegative definite, and the PARMS statement then ensures that it is positive definite by constraining the two diagonal terms to be greater than or equal to 1E–4.

NOITER

requests that no optimization iterations be performed and that PROC HPLMIXED use the best value from the grid search to perform inferences. By default, iterations begin at the best value from the PARMS grid search. The NOITER option will be implied by the specification of the BLUP option in the HPLMIXED statement.

PARMSDATA=*SAS-data-set*

PDATA=*SAS-data-set*

reads in covariance parameter values from a SAS data set. The data set should contain the Est or Covp1 through Covp*n* variables.

UPPERB=*value-list*

enables you to specify upper boundary constraints on the covariance parameters. The *value-list* specification is a list of numbers or missing values (.) separated by commas. You must list the numbers in the order that PROC HPLMIXED uses for the covariance parameters, and each number corresponds to the upper boundary constraint. A missing value instructs PROC HPLMIXED to use its default constraint. If you do not specify numbers for all of the covariance parameters, PROC HPLMIXED assumes that the remaining ones are missing.

PERFORMANCE Statement

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

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

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

The PERFORMANCE statement for SAS high-performance analytical procedures is documented in the section "PERFORMANCE Statement" on page 36 of Chapter 3, "Shared Concepts and Topics."

RANDOM Statement

RANDOM *random-effects* < / *options* > **;**

The RANDOM statement defines the random effects that constitute the $\boldsymbol{\gamma}$ vector in the mixed model. You can use this statement to specify traditional variance component models and to specify random coefficients. The random effects can be classification or continuous, and multiple RANDOM statements are possible.

Using notation from the section "Linear Mixed Models Theory" on page 310, the purpose of the RANDOM statement is to define the $\mathbf{Z}$ matrix of the mixed model, the random effects in the $\boldsymbol{\gamma}$ vector, and the structure of $\mathbf{G}$. The $\mathbf{Z}$ matrix is constructed exactly as the $\mathbf{X}$ matrix for the fixed effects is constructed, and the $\mathbf{G}$ matrix is constructed to correspond with the effects that constitute $\mathbf{Z}$. The structure of $\mathbf{G}$ is defined by using the TYPE= option.

You can specify INTERCEPT (or INT) as a random effect to indicate the intercept. PROC HPLMIXED does not include the intercept in the RANDOM statement by default as it does in the MODEL statement.

Table 8.5 summarizes important options in the RANDOM statement. All options are subsequently discussed in alphabetical order.

Table 8.5 Summary of Important RANDOM Statement Options

Option	Description
Construction of Covariance Structure	
SUBJECT=	Identifies the subjects in the model
TYPE=	Specifies the covariance structure

Table 8.5 *continued*

Option	Description
Statistical Output	
ALPHA=α	Determines the confidence level $(1-\alpha)$
CL	Requests confidence limits for predictors of random effects
SOLUTION	Displays solutions $\widehat{\boldsymbol{\gamma}}$ of the random effects

You can specify the following *options* in the RANDOM statement after a slash (/).

ALPHA=*number*

sets the confidence level to be $1-$*number* for each confidence interval of the random-effects estimates. The value of *number* must be between 0 and 1; the default is 0.05.

CL

requests that *t*-type confidence limits be constructed for each of the random-effect estimates. The confidence level is 0.95 by default; this can be changed with the ALPHA= option.

SOLUTION

S

requests that the solution for the random-effects parameters be produced. Using notation from the section "Linear Mixed Models Theory" on page 310, these estimates are the empirical best linear unbiased predictors (EBLUPs), $\widehat{\boldsymbol{\gamma}} = \widehat{\mathbf{G}}\mathbf{Z}'\widehat{\mathbf{V}}^{-1}(\mathbf{y}-\mathbf{X}\widehat{\boldsymbol{\beta}})$. They can be useful for comparing the random effects from different experimental units and can also be treated as residuals in performing diagnostics for your mixed model.

The numbers displayed in the SE Pred column of the "Solution for Random Effects" table are not the standard errors of the $\widehat{\boldsymbol{\gamma}}$ displayed in the Estimate column; rather, they are the standard errors of predictions $\widehat{\boldsymbol{\gamma}}_i - \boldsymbol{\gamma}_i$, where $\widehat{\boldsymbol{\gamma}}_i$ is the ith EBLUP and $\boldsymbol{\gamma}_i$ is the ith random-effect parameter.

SUBJECT=*effect*

SUB=*effect*

identifies the subjects in your mixed model. Complete independence is assumed across subjects; thus, for the RANDOM statement, the SUBJECT= option produces a block-diagonal structure in $\mathbf{G}$ with identical blocks. In fact, specifying a subject effect is equivalent to nesting all other effects in the RANDOM statement within the subject effect.

When you specify the SUBJECT= option and a classification random effect, computations are usually much quicker if the levels of the random effect are duplicated within each level of the SUBJECT= effect.

TYPE=*covariance-structure*

specifies the covariance structure of $\mathbf{G}$. Valid values for *covariance-structure* and their descriptions are listed in Table 8.6. Although a variety of structures are available, most applications call for either TYPE=VC or TYPE=UN. The TYPE=VC (variance components) option is the default structure, and it models a different variance component for each random effect.

The TYPE=UN (unstructured) option is useful for correlated random coefficient models. For example, the following statement specifies a random intercept-slope model that has different variances for the intercept and slope and a covariance between them:

```
random intercept age / type=un subject=person;
```

You can also use TYPE=FA0(2) here to request a **G** estimate that is constrained to be nonnegative definite.

If you are constructing your own columns of **Z** with continuous variables, you can use the TYPE=TOEP(1) structure to group them together to have a common variance component. If you want to have different covariance structures in different parts of **G**, you must use multiple RANDOM statements with different TYPE= options.

Table 8.6 Covariance Structures

Structure	Description	Parms	(i, j) element
ANTE(1)	Antedependence	$2t-1$	$\sigma_i\sigma_j\prod_{k=i}^{j-1}\rho_k$
AR(1)	Autoregressive(1)	2	$\sigma^2\rho^{\lvert i-j\rvert}$
ARH(1)	Heterogeneous AR(1)	$t+1$	$\sigma_i\sigma_j\rho^{\lvert i-j\rvert}$
ARMA(1,1)	Autoregressive moving average(1,1)	3	$\sigma^2[\gamma\rho^{\lvert i-j\rvert-1}1(i\neq j)+1(i=j)]$
CS	Compound symmetry	2	$\sigma_1+\sigma^2 1(i=j)$
CSH	Heterogeneous compound symmetry	$t+1$	$\sigma_i\sigma_j[\rho 1(i\neq j)+1(i=j)]$
FA(q)	Factor analytic	$\frac{q}{2}(2t-q+1)+t$	$\Sigma_{k=1}^{\min(i,j,q)}\lambda_{ik}\lambda_{jk}+\sigma_i^2 1(i=j)$
FA0(q)	No diagonal FA	$\frac{q}{2}(2t-q+1)$	$\Sigma_{k=1}^{\min(i,j,q)}\lambda_{ik}\lambda_{jk}$
FA1(q)	Equal diagonal FA	$\frac{q}{2}(2t-q+1)+1$	$\Sigma_{k=1}^{\min(i,j,q)}\lambda_{ik}\lambda_{jk}+\sigma^2 1(i=j)$
HF	Huynh-Feldt	$t+1$	$(\sigma_i^2+\sigma_j^2)/2+\lambda 1(i\neq j)$
SIMPLE	An alias for VC	q	$\sigma_k^2 1(i=j)$ for the kth effect
TOEP	Toeplitz	t	$\sigma_{\lvert i-j\rvert+1}$
TOEP(q)	Banded Toeplitz	q	$\sigma_{\lvert i-j\rvert+1}1(\lvert i-j\rvert<q)$
TOEPH	Heterogeneous TOEP	$2t-1$	$\sigma_i\sigma_j\rho_{\lvert i-j\rvert}$
TOEPH(q)	Banded heterogeneous TOEP	$t+q-1$	$\sigma_i\sigma_j\rho_{\lvert i-j\rvert}1(\lvert i-j\rvert<q)$
UN	Unstructured	$t(t+1)/2$	σ_{ij}
UN(q)	Banded	$\frac{q}{2}(2t-q+1)$	$\sigma_{ij}1(\lvert i-j\rvert<q)$
UNR	Unstructured correlation	$t(t+1)/2$	$\sigma_i\sigma_j\rho_{\max(i,j)\min(i,j)}$
UNR(q)	Banded correlations	$\frac{q}{2}(2t-q+1)$	$\sigma_i\sigma_j\rho_{\max(i,j)\min(i,j)}$
VC	Variance components	q	$\sigma_k^2 1(i=j)$ for the kth effect

In Table 8.6, the Parms column represents the number of covariance parameters in the structure, t is the overall dimension of the covariance matrix, and $1(A)$ equals 1 when A is true and 0 otherwise. For example, $1(i=j)$ equals 1 when $i=j$ and 0 otherwise, and $1(|i-j|<q)$ equals 1 when $|i-j|<q$ and 0 otherwise. For the TYPE=TOEPH structures, $\rho_0=1$; for the TYPE=UNR structures, $\rho_{ii}=1$ for all i.

Table 8.7 lists some examples of the structures in Table 8.6.

Table 8.7 Covariance Structure Examples

Description	Structure	Example
Variance components	VC (default)	$\begin{bmatrix} \sigma_B^2 & 0 & 0 & 0 \\ 0 & \sigma_B^2 & 0 & 0 \\ 0 & 0 & \sigma_{AB}^2 & 0 \\ 0 & 0 & 0 & \sigma_{AB}^2 \end{bmatrix}$
Compound symmetry	CS	$\begin{bmatrix} \sigma^2+\sigma_1 & \sigma_1 & \sigma_1 & \sigma_1 \\ \sigma_1 & \sigma^2+\sigma_1 & \sigma_1 & \sigma_1 \\ \sigma_1 & \sigma_1 & \sigma^2+\sigma_1 & \sigma_1 \\ \sigma_1 & \sigma_1 & \sigma_1 & \sigma^2+\sigma_1 \end{bmatrix}$
Unstructured	UN	$\begin{bmatrix} \sigma_1^2 & \sigma_{21} & \sigma_{31} & \sigma_{41} \\ \sigma_{21} & \sigma_2^2 & \sigma_{32} & \sigma_{42} \\ \sigma_{31} & \sigma_{32} & \sigma_3^2 & \sigma_{43} \\ \sigma_{41} & \sigma_{42} & \sigma_{43} & \sigma_4^2 \end{bmatrix}$
Banded main diagonal	UN(1)	$\begin{bmatrix} \sigma_1^2 & 0 & 0 & 0 \\ 0 & \sigma_2^2 & 0 & 0 \\ 0 & 0 & \sigma_3^2 & 0 \\ 0 & 0 & 0 & \sigma_4^2 \end{bmatrix}$
First-order autoregressive	AR(1)	$\sigma^2 \begin{bmatrix} 1 & \rho & \rho^2 & \rho^3 \\ \rho & 1 & \rho & \rho^2 \\ \rho^2 & \rho & 1 & \rho \\ \rho^3 & \rho^2 & \rho & 1 \end{bmatrix}$
Toeplitz	TOEP	$\begin{bmatrix} \sigma^2 & \sigma_1 & \sigma_2 & \sigma_3 \\ \sigma_1 & \sigma^2 & \sigma_1 & \sigma_2 \\ \sigma_2 & \sigma_1 & \sigma^2 & \sigma_1 \\ \sigma_3 & \sigma_2 & \sigma_1 & \sigma^2 \end{bmatrix}$
Toeplitz with two bands	TOEP(2)	$\begin{bmatrix} \sigma^2 & \sigma_1 & 0 & 0 \\ \sigma_1 & \sigma^2 & \sigma_1 & 0 \\ 0 & \sigma_1 & \sigma^2 & \sigma_1 \\ 0 & 0 & \sigma_1 & \sigma^2 \end{bmatrix}$
Heterogeneous autoregressive(1)	ARH(1)	$\begin{bmatrix} \sigma_1^2 & \sigma_1\sigma_2\rho & \sigma_1\sigma_3\rho^2 & \sigma_1\sigma_4\rho^3 \\ \sigma_2\sigma_1\rho & \sigma_2^2 & \sigma_2\sigma_3\rho & \sigma_2\sigma_4\rho^2 \\ \sigma_3\sigma_1\rho^2 & \sigma_3\sigma_2\rho & \sigma_3^2 & \sigma_3\sigma_4\rho \\ \sigma_4\sigma_1\rho^3 & \sigma_4\sigma_2\rho & \sigma_4\sigma_3\rho & \sigma_4^2 \end{bmatrix}$
First-order autoregressive moving average	ARMA(1,1)	$\sigma^2 \begin{bmatrix} 1 & \gamma & \gamma\rho & \gamma\rho^2 \\ \gamma & 1 & \gamma & \gamma\rho \\ \gamma\rho & \gamma & 1 & \gamma \\ \gamma\rho^2 & \gamma\rho & \gamma & 1 \end{bmatrix}$
Heterogeneous compound symmetry	CSH	$\begin{bmatrix} \sigma_1^2 & \sigma_1\sigma_2\rho & \sigma_1\sigma_3\rho & \sigma_1\sigma_4\rho \\ \sigma_2\sigma_1\rho & \sigma_2^2 & \sigma_2\sigma_3\rho & \sigma_2\sigma_4\rho \\ \sigma_3\sigma_1\rho & \sigma_3\sigma_2\rho & \sigma_3^2 & \sigma_3\sigma_4\rho \\ \sigma_4\sigma_1\rho & \sigma_4\sigma_2\rho & \sigma_4\sigma_3\rho & \sigma_4^2 \end{bmatrix}$

Table 8.7 *continued*

Description	Structure	Example
First-order factor analytic	FA(1)	$\begin{bmatrix} \lambda_1^2+d_1 & \lambda_1\lambda_2 & \lambda_1\lambda_3 & \lambda_1\lambda_4 \\ \lambda_2\lambda_1 & \lambda_2^2+d_2 & \lambda_2\lambda_3 & \lambda_2\lambda_4 \\ \lambda_3\lambda_1 & \lambda_3\lambda_2 & \lambda_3^2+d_3 & \lambda_3\lambda_4 \\ \lambda_4\lambda_1 & \lambda_4\lambda_2 & \lambda_4\lambda_3 & \lambda_4^2+d_4 \end{bmatrix}$
Huynh-Feldt	HF	$\begin{bmatrix} \sigma_1^2 & \frac{\sigma_1^2+\sigma_2^2}{2}-\lambda & \frac{\sigma_1^2+\sigma_3^2}{2}-\lambda \\ \frac{\sigma_2^2+\sigma_1^2}{2}-\lambda & \sigma_2^2 & \frac{\sigma_2^2+\sigma_3^2}{2}-\lambda \\ \frac{\sigma_3^2+\sigma_1^2}{2}-\lambda & \frac{\sigma_3^2+\sigma_2^2}{2}-\lambda & \sigma_3^2 \end{bmatrix}$
First-order antedependence	ANTE(1)	$\begin{bmatrix} \sigma_1^2 & \sigma_1\sigma_2\rho_1 & \sigma_1\sigma_3\rho_1\rho_2 \\ \sigma_2\sigma_1\rho_1 & \sigma_2^2 & \sigma_2\sigma_3\rho_2 \\ \sigma_3\sigma_1\rho_2\rho_1 & \sigma_3\sigma_2\rho_2 & \sigma_3^2 \end{bmatrix}$
Heterogeneous Toeplitz	TOEPH	$\begin{bmatrix} \sigma_1^2 & \sigma_1\sigma_2\rho_1 & \sigma_1\sigma_3\rho_2 & \sigma_1\sigma_4\rho_3 \\ \sigma_2\sigma_1\rho_1 & \sigma_2^2 & \sigma_2\sigma_3\rho_1 & \sigma_2\sigma_4\rho_2 \\ \sigma_3\sigma_1\rho_2 & \sigma_3\sigma_2\rho_1 & \sigma_3^2 & \sigma_3\sigma_4\rho_1 \\ \sigma_4\sigma_1\rho_3 & \sigma_4\sigma_2\rho_2 & \sigma_4\sigma_3\rho_1 & \sigma_4^2 \end{bmatrix}$
Unstructured correlations	UNR	$\begin{bmatrix} \sigma_1^2 & \sigma_1\sigma_2\rho_{21} & \sigma_1\sigma_3\rho_{31} & \sigma_1\sigma_4\rho_{41} \\ \sigma_2\sigma_1\rho_{21} & \sigma_2^2 & \sigma_2\sigma_3\rho_{32} & \sigma_2\sigma_4\rho_{42} \\ \sigma_3\sigma_1\rho_{31} & \sigma_3\sigma_2\rho_{32} & \sigma_3^2 & \sigma_3\sigma_4\rho_{43} \\ \sigma_4\sigma_1\rho_{41} & \sigma_4\sigma_2\rho_{42} & \sigma_4\sigma_3\rho_{43} & \sigma_4^2 \end{bmatrix}$

The following list provides some further information about these covariance structures:

TYPE=ANTE(1) specifies the first-order antedependence structure (Kenward 1987; Patel 1991; Macchiavelli and Arnold 1994). In Table 8.6, σ_i^2 is the i variance parameter, and ρ_k is the k autocorrelation parameter that satisfies $|\rho_k| < 1$.

TYPE=AR(1) specifies a first-order autoregressive structure. PROC HPLMIXED imposes the constraint $|\rho| < 1$ for stationarity.

TYPE=ARH(1) specifies a heterogeneous first-order autoregressive structure. As with TYPE=AR(1), PROC HPLMIXED imposes the constraint $|\rho| < 1$ for stationarity.

TYPE=ARMA(1,1) specifies the first-order autoregressive moving average structure. In Table 8.6, ρ is the autoregressive parameter, γ models a moving average component, and σ^2 is the residual variance. In the notation of Fuller (1976, p. 68), $\rho = \theta_1$ and

$$\gamma = \frac{(1 + b_1\theta_1)(\theta_1 + b_1)}{1 + b_1^2 + 2b_1\theta_1}$$

The example in Table 8.7 and $|b_1| < 1$ imply that

$$b_1 = \frac{\beta - \sqrt{\beta^2 - 4\alpha^2}}{2\alpha}$$

where $\alpha = \gamma - \rho$ and $\beta = 1 + \rho^2 - 2\gamma\rho$. PROC HPLMIXED imposes the constraints $|\rho| < 1$ and $|\gamma| < 1$ for stationarity, although the resulting covariance

matrix is not positive definite for some values of ρ and γ in this region. When the estimated value of ρ becomes negative, the computed covariance is multiplied by $\cos(\pi d_{ij})$ to account for the negativity.

TYPE=CS specifies the compound-symmetry structure, which has constant variance and constant covariance.

TYPE=CSH specifies the heterogeneous compound-symmetry structure. This structure has a different variance parameter for each diagonal element, and it uses the square roots of these parameters in the off-diagonal entries. In Table 8.6, σ_i^2 is the i variance parameter, and ρ is the correlation parameter that satisfies $|\rho| < 1$.

TYPE=FA(q) specifies the factor-analytic structure with q factors (Jennrich and Schluchter 1986). This structure is of the form $\mathbf{\Lambda}\mathbf{\Lambda}' + \mathbf{D}$, where $\mathbf{\Lambda}$ is a $t \times q$ rectangular matrix and $\mathbf{D}$ is a $t \times t$ diagonal matrix with t different parameters. When $q > 1$, the elements of $\mathbf{\Lambda}$ in its upper right corner (that is, the elements in the i row and j column for $j > i$) are set to zero to fix the rotation of the structure.

TYPE=FA0(q) is similar to the FA(q) structure except that no diagonal matrix $\mathbf{D}$ is included. When $q < t$ (that is, when the number of factors is less than the dimension of the matrix), this structure is nonnegative definite but not of full rank. In this situation, you can use this structure for approximating an unstructured $\mathbf{G}$ matrix in the RANDOM statement. When $q = t$, you can use this structure to constrain $\mathbf{G}$ to be nonnegative definite in the RANDOM statement.

TYPE=FA1(q) is similar to the TYPE=FA(q) structure except that all of the elements in $\mathbf{D}$ are constrained to be equal. This offers a useful and more parsimonious alternative to the full factor-analytic structure.

TYPE=HF specifies the Huynh-Feldt covariance structure (Huynh and Feldt 1970). This structure is similar to the TYPE=CSH structure in that it has the same number of parameters and heterogeneity along the main diagonal. However, it constructs the off-diagonal elements by taking arithmetic means rather than geometric means.

You can perform a likelihood ratio test of the Huynh-Feldt conditions by running PROC HPLMIXED twice, once with TYPE=HF and once with TYPE=UN, and then subtracting their respective values of -2 times the maximized likelihood.

If PROC HPLMIXED does not converge under your Huynh-Feldt model, you can specify your own starting values with the PARMS statement. The default MIVQUE(0) starting values can sometimes be poor for this structure. A good choice for starting values is often the parameter estimates that correspond to an initial fit that uses TYPE=CS.

TYPE=SIMPLE is an alias for TYPE=VC.

TYPE=TOEP<(q)> specifies a banded Toeplitz structure. This can be viewed as a moving average structure with order equal to $q - 1$. The TYPE=TOEP option is a full Toeplitz matrix, which can be viewed as an autoregressive structure with order equal to the dimension of the matrix. The specification TYPE=TOEP(1) is the same as $\sigma^2 I$, where I is an identity matrix, and it can be useful for specifying the same variance component for several effects.

TYPE=TOEPH<(q)> specifies a heterogeneous banded Toeplitz structure. In Table 8.6, σ_i^2 is the i variance parameter and ρ_j is the j correlation parameter that satisfies $|\rho_j| < 1$. If

you specify the order parameter q, then PROC HPLMIXED estimates only the first q bands of the matrix, setting all higher bands equal to 0. The option TOEPH(1) is equivalent to both the TYPE=UN(1) and TYPE=UNR(1) options.

TYPE=UN<(q)> specifies a completely general (unstructured) covariance matrix that is parameterized directly in terms of variances and covariances. The variances are constrained to be nonnegative, and the covariances are unconstrained. This structure is not constrained to be nonnegative definite in order to avoid nonlinear constraints. However, you can use the TYPE=FA0 structure if you want this constraint to be imposed by a Cholesky factorization. If you specify the order parameter q, then PROC HPLMIXED estimates only the first q bands of the matrix, setting all higher bands equal to 0.

TYPE=UNR<(q)> specifies a completely general (unstructured) covariance matrix that is parameterized in terms of variances and correlations. This structure fits the same model as the TYPE=UN(q) option but with a different parameterization. The i variance parameter is σ_i^2. The parameter ρ_{jk} is the correlation between the j and k measurements; it satisfies $|\rho_{jk}| < 1$. If you specify the order parameter r, then PROC HPLMIXED estimates only the first q bands of the matrix, setting all higher bands equal to zero.

TYPE=VC specifies standard variance components. This is the default structure for both the RANDOM and REPEATED statements. In the RANDOM statement, a distinct variance component is assigned to each effect.

Jennrich and Schluchter (1986) provide general information about the use of covariance structures, and Wolfinger (1996) presents details about many of the heterogeneous structures.

REPEATED Statement

REPEATED *repeated-effect* < / *options* > **;**

The REPEATED statement specifies the $\mathbf{R}$ matrix in the mixed model. If no REPEATED statement is specified, $\mathbf{R}$ is assumed to be equal to $\sigma^2\mathbf{I}$. For this release, you can use the REPEATED statement only with the BLUP option. The statement is ignored when no BLUP option is specified.

The *repeated-effect* is required, because the order of the input data is not necessarily reproducible in a distributed environment. The *repeated-effect* must contain only classification variables. Make sure that the levels of the *repeated-effect* are different for each observation within a subject; otherwise, PROC HPLMIXED constructs identical rows in $\mathbf{R}$ that correspond to the observations with the same level. This results in a singular $\mathbf{R}$ matrix and an infinite likelihood.

Table 8.8 summarizes important options in the REPEATED statement. All options are subsequently discussed in alphabetical order.

Table 8.8 Summary of Important REPEATED Statement Options

Option	Description
Construction of Covariance Structure	
SUBJECT=	Identifies the subjects in the R-side model
TYPE=	Specifies the R-side covariance structure

You can specify the following *options* in the REPEATED statement after a slash (/).

SUBJECT=*effect*

SUB=*effect*

identifies the subjects in your mixed model. Complete independence is assumed across subjects; therefore, the SUBJECT= option produces a block-diagonal structure in $\mathbf{R}$ with identical blocks. When the SUBJECT= effect consists entirely of classification variables, the blocks of $\mathbf{R}$ correspond to observations that share the same level of that effect. These blocks are sorted according to this effect as well.

If you want to model nonzero covariance among all of the observations in your SAS data set, specify SUBJECT=Dummy_Intercept to treat the data as if they are all from one subject. You need to create this Dummy_Intercept variable in the data set. However, be aware that in this case PROC HPLMIXED manipulates an $\mathbf{R}$ matrix with dimensions equal to the number of observations.

TYPE=*covariance-structure*

specifies the covariance structure of the $\mathbf{R}$ matrix. The SUBJECT= option defines the blocks of $\mathbf{R}$, and the TYPE= option specifies the structure of these blocks. The default structure is VC. You can specify any of the covariance structures that are described in the TYPE= option in the RANDOM statement.

Details: HPLMIXED Procedure

Linear Mixed Models Theory

This section provides an overview of a likelihood-based approach to linear mixed models. This approach simplifies and unifies many common statistical analyses, including those that involve repeated measures, random effects, and random coefficients. The basic assumption is that the data are linearly related to unobserved multivariate normal random variables. For extensions to nonlinear and nonnormal situations, see the documentation of the GLIMMIX and NLMIXED procedures in the *SAS/STAT User's Guide*. Additional theory and examples are provided in Littell et al. (2006); Verbeke and Molenberghs (1997, 2000); and Burdick and Graybill (1992).

Matrix Notation

Suppose that you observe n data points $y_1, \ldots, y_n$ and that you want to explain them by using n values for each of p explanatory variables $x_{11}, \ldots, x_{1p}, x_{21}, \ldots, x_{2p}, \ldots, x_{n1}, \ldots, x_{np}$. The x_{ij} values can be either regression-type continuous variables or dummy variables that indicate class membership. The standard linear model for this setup is

$$y_i = \sum_{j=1}^{p} x_{ij}\beta_j + \epsilon_i \quad i = 1, \ldots, n$$

where $\beta_1, \ldots, \beta_p$ are unknown fixed-effects parameters to be estimated and $\epsilon_1, \ldots, \epsilon_n$ are unknown independent and identically distributed normal (Gaussian) random variables with mean 0 and variance σ^2.

The preceding equations can be written simultaneously by using vectors and a matrix, as follows:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

For convenience, simplicity, and extendability, this entire system is written as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

where $\mathbf{y}$ denotes the vector of observed y_i's, $\mathbf{X}$ is the known matrix of x_{ij}'s, $\boldsymbol{\beta}$ is the unknown fixed-effects parameter vector, and $\boldsymbol{\epsilon}$ is the unobserved vector of independent and identically distributed Gaussian random errors.

In addition to denoting data, random variables, and explanatory variables in the preceding fashion, the subsequent development makes use of basic matrix operators such as transpose ($'$), inverse ($^{-1}$), generalized inverse ($^{-}$), determinant ($|\cdot|$), and matrix multiplication. See Searle (1982) for details about these and other matrix techniques.

Formulation of the Mixed Model

The previous general linear model is certainly a useful one (Searle 1971), and it is the one fitted by the GLM procedure. However, many times the distributional assumption about ϵ is too restrictive. The mixed model extends the general linear model by allowing a more flexible specification of the covariance matrix of ϵ. In other words, it allows for both correlation and heterogeneous variances, although you still assume normality.

The mixed model is written as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\boldsymbol{\gamma} + \boldsymbol{\epsilon}$$

where everything is the same as in the general linear model except for the addition of the known design matrix, $\mathbf{Z}$, and the vector of unknown random-effects parameters, $\boldsymbol{\gamma}$. The matrix $\mathbf{Z}$ can contain either continuous or dummy variables, just like $\mathbf{X}$. The name *mixed model* comes from the fact that the model contains both fixed-effects parameters, $\boldsymbol{\beta}$, and random-effects parameters, $\boldsymbol{\gamma}$. See Henderson (1990) and Searle, Casella, and McCulloch (1992) for historical developments of the mixed model.

A key assumption in the foregoing analysis is that $\boldsymbol{\gamma}$ and $\boldsymbol{\epsilon}$ are normally distributed with

$$\mathrm{E}\begin{bmatrix} \boldsymbol{\gamma} \\ \boldsymbol{\epsilon} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

$$\mathrm{Var}\begin{bmatrix} \boldsymbol{\gamma} \\ \boldsymbol{\epsilon} \end{bmatrix} = \begin{bmatrix} \mathbf{G} & \mathbf{0} \\ \mathbf{0} & \mathbf{R} \end{bmatrix}$$

Therefore, the variance of $\mathbf{y}$ is $\mathbf{V} = \mathbf{Z}\mathbf{G}\mathbf{Z}' + \mathbf{R}$. You can model $\mathbf{V}$ by setting up the random-effects design matrix $\mathbf{Z}$ and by specifying covariance structures for $\mathbf{G}$ and $\mathbf{R}$.

Note that this is a general specification of the mixed model, in contrast to many texts and articles that discuss only simple random effects. Simple random effects are a special case of the general specification with $\mathbf{Z}$ containing dummy variables, $\mathbf{G}$ containing variance components in a diagonal structure, and $\mathbf{R} = \sigma^2\mathbf{I}_n$, where $\mathbf{I}_n$ denotes the $n \times n$ identity matrix. The general linear model is a further special case with $\mathbf{Z} = \mathbf{0}$ and $\mathbf{R} = \sigma^2\mathbf{I}_n$.

The following two examples illustrate the most common formulations of the general linear mixed model.

Example: Growth Curve with Compound Symmetry

Suppose that you have three growth curve measurements for s individuals and that you want to fit an overall linear trend in time. Your $\mathbf{X}$ matrix is as follows:

$$\mathbf{X} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ \vdots & \vdots \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix}$$

The first column (coded entirely with 1s) fits an intercept, and the second column (coded with series of $1, 2, 3$) fits a slope. Here, $n = 3s$ and $p = 2$.

Suppose further that you want to introduce a common correlation among the observations from a single individual, with correlation being the same for all individuals. One way of setting this up in the general mixed model is to eliminate the $\mathbf{Z}$ and $\mathbf{G}$ matrices and let the $\mathbf{R}$ matrix be block-diagonal with blocks corresponding to the individuals and with each block having the *compound-symmetry* structure. This structure has two unknown parameters, one modeling a common covariance and the other modeling a residual variance. The form for $\mathbf{R}$ would then be

$$\mathbf{R} = \begin{bmatrix} \sigma_1^2+\sigma^2 & \sigma_1^2 & \sigma_1^2 & & & \\ \sigma_1^2 & \sigma_1^2+\sigma^2 & \sigma_1^2 & & & \\ \sigma_1^2 & \sigma_1^2 & \sigma_1^2+\sigma^2 & & & \\ & & & \ddots & & \\ & & & & \sigma_1^2+\sigma^2 & \sigma_1^2 & \sigma_1^2 \\ & & & & \sigma_1^2 & \sigma_1^2+\sigma^2 & \sigma_1^2 \\ & & & & \sigma_1^2 & \sigma_1^2 & \sigma_1^2+\sigma^2 \end{bmatrix}$$

where blanks denote zeros. There are $3s$ rows and columns altogether, and the common correlation is $\sigma_1^2/(\sigma_1^2+\sigma^2)$.

The following PROC HPLMIXED statements fit this model:

```
proc hplmixed;
   class indiv;
   model y = time;
   repeated morder/ type=cs subject=indiv;
run;
```

Here, INDIV is a classification variable that indexes individuals. The MODEL statement fits a straight line for TIME ; the intercept is fit by default just as in PROC GLM. The REPEATED statement models the $\mathbf{R}$ matrix: TYPE=CS specifies the compound symmetry structure, and SUBJECT=INDIV specifies the blocks of $\mathbf{R}$, and MORDER is the repeated effect that records the order of the measurements for each individual.

An alternative way of specifying the common intra-individual correlation is to let

$$\mathbf{Z} = \begin{bmatrix} 1 & & \\ 1 & & \\ 1 & & \\ & 1 & \\ & 1 & \\ & 1 & \\ & & \ddots \\ & & 1 \\ & & 1 \\ & & 1 \end{bmatrix}$$

$$\mathbf{G} = \begin{bmatrix} \sigma_1^2 & & & \\ & \sigma_1^2 & & \\ & & \ddots & \\ & & & \sigma_1^2 \end{bmatrix}$$

and $\mathbf{R} = \sigma^2 \mathbf{I}_n$. The $\mathbf{Z}$ matrix has $3s$ rows and s columns, and $\mathbf{G}$ is $s \times s$.

You can set up this model in PROC HPLMIXED in two different but equivalent ways:

```
proc hplmixed;
   class indiv;
   model y = time;
   random indiv;
run;

proc hplmixed;
   class indiv;
   model y = time;
   random intercept / subject=indiv;
run;
```

Both of these specifications fit the same model as the previous one that used the REPEATED statement. However, the RANDOM specifications constrain the correlation to be positive, whereas the REPEATED specification leaves the correlation unconstrained.

Example: Split-Plot Design

The split-plot design involves two experimental treatment factors, A and B, and two different sizes of experimental units to which they are applied (Winer 1971; Snedecor and Cochran 1980; Milliken and Johnson 1992; Steel, Torrie, and Dickey 1997). The levels of A are randomly assigned to the larger-sized experimental units, called *whole plots*, whereas the levels of B are assigned to the smaller-sized experimental units, the *subplots*. The subplots are assumed to be nested within the whole plots, so that a whole plot consists of a cluster of subplots and a level of A is applied to the entire cluster.

Such an arrangement is often necessary by nature of the experiment; the classical example is the application of fertilizer to large plots of land and different crop varieties planted in subdivisions of the large plots. For this example, fertilizer is the whole-plot factor A and variety is the subplot factor B.

The first example is a split-plot design for which the whole plots are arranged in a randomized block design. The appropriate PROC HPLMIXED statements are as follows:

```
proc hplmixed;
   class a b block;
   model y = a b a*b;
   random block a*block;
run;
```

Here

$$\mathbf{R} = \sigma^2 \mathbf{I}_{24}$$

and $\mathbf{X}$, $\mathbf{Z}$, and $\mathbf{G}$ have the following form:

$$\mathbf{X} = \begin{bmatrix}
1 & 1 & & & 1 & & 1 & & & & & \\
1 & 1 & & & & 1 & & 1 & & & & \\
1 & & 1 & & 1 & & & & 1 & & & \\
1 & & 1 & & & 1 & & & & 1 & & \\
1 & & & 1 & 1 & & & & & & 1 & \\
1 & & & 1 & & 1 & & & & & & 1 \\
\vdots & & \vdots & & \vdots & & & & \vdots & & & \\
1 & 1 & & & 1 & & 1 & & & & & \\
1 & 1 & & & & 1 & & 1 & & & & \\
1 & & 1 & & 1 & & & & 1 & & & \\
1 & & 1 & & & 1 & & & & 1 & & \\
1 & & & 1 & 1 & & & & & & 1 & \\
1 & & & 1 & & 1 & & & & & & 1
\end{bmatrix}$$

$$\mathbf{Z} = \begin{bmatrix}
1 & & & & 1 & & & & & & & & & & & & & & & & & & & \\
1 & & & & 1 & & & & & & & & & & & & & & & & & & & \\
1 & & & & & 1 & & & & & & & & & & & & & & & & & & \\
1 & & & & & 1 & & & & & & & & & & & & & & & & & & \\
1 & & & & & & 1 & & & & & & & & & & & & & & & & & \\
1 & & & & & & 1 & & & & & & & & & & & & & & & & & \\
 & 1 & & & & & & 1 & & & & & & & & & & & & & & & & \\
 & 1 & & & & & & 1 & & & & & & & & & & & & & & & & \\
 & 1 & & & & & & & 1 & & & & & & & & & & & & & & & \\
 & 1 & & & & & & & 1 & & & & & & & & & & & & & & & \\
 & 1 & & & & & & & & 1 & & & & & & & & & & & & & & \\
 & 1 & & & & & & & & 1 & & & & & & & & & & & & & & \\
 & & 1 & & & & & & & & 1 & & & & & & & & & & & & & \\
 & & 1 & & & & & & & & 1 & & & & & & & & & & & & & \\
 & & 1 & & & & & & & & & 1 & & & & & & & & & & & & \\
 & & 1 & & & & & & & & & 1 & & & & & & & & & & & & \\
 & & 1 & & & & & & & & & & 1 & & & & & & & & & & & \\
 & & 1 & & & & & & & & & & 1 & & & & & & & & & & & \\
 & & & 1 & & & & & & & & & & 1 & & & & & & & & & & \\
 & & & 1 & & & & & & & & & & 1 & & & & & & & & & & \\
 & & & 1 & & & & & & & & & & & 1 & & & & & & & & & \\
 & & & 1 & & & & & & & & & & & 1 & & & & & & & & & \\
 & & & 1 & & & & & & & & & & & & 1 & & & & & & & & \\
 & & & 1 & & & & & & & & & & & & 1 & & & & & & & & \\
 & & & & 1 & & & & & & & & & & & & 1 & & & & & & & \\
 & & & & 1 & & & & & & & & & & & & 1 & & & & & & & \\
 & & & & 1 & & & & & & & & & & & & & 1 & & & & & & \\
 & & & & 1 & & & & & & & & & & & & & 1 & & & & & & \\
 & & & & 1 & & & & & & & & & & & & & & 1 & & & & & \\
 & & & & 1 & & & & & & & & & & & & & & 1 & & & & & \\
 & & & & 1 & & & & & & & & & & & & & & & 1 & & & & \\
 & & & & 1 & & & & & & & & & & & & & & & 1 & & & &
\end{bmatrix}$$

$$\mathbf{G} = \begin{bmatrix} \sigma^2_B & & & & & & & \\ & \sigma^2_B & & & & & & \\ & & \sigma^2_B & & & & & \\ & & & \sigma^2_B & & & & \\ & & & & \sigma^2_{AB} & & & \\ & & & & & \sigma^2_{AB} & & \\ & & & & & & \ddots & \\ & & & & & & & \sigma^2_{AB} \end{bmatrix}$$

where σ^2_B is the variance component for Block and σ^2_{AB} is the variance component for A*Block. Changing the RANDOM statement as follows fits the same model, but with **Z** and **G** sorted differently:

```
random int a / subject=block;
```

$$\mathbf{Z} = \begin{bmatrix} 1 & 1 & & & & & & & & & & & & & & \\ 1 & 1 & & & & & & & & & & & & & & \\ 1 & & 1 & & & & & & & & & & & & & \\ 1 & & 1 & & & & & & & & & & & & & \\ 1 & & & 1 & & & & & & & & & & & & \\ 1 & & & 1 & & & & & & & & & & & & \\ & & & & 1 & 1 & & & & & & & & & & \\ & & & & 1 & 1 & & & & & & & & & & \\ & & & & 1 & & 1 & & & & & & & & & \\ & & & & 1 & & 1 & & & & & & & & & \\ & & & & 1 & & & 1 & & & & & & & & \\ & & & & 1 & & & 1 & & & & & & & & \\ & & & & & & & & 1 & 1 & & & & & & \\ & & & & & & & & 1 & 1 & & & & & & \\ & & & & & & & & 1 & & 1 & & & & & \\ & & & & & & & & 1 & & 1 & & & & & \\ & & & & & & & & 1 & & & 1 & & & & \\ & & & & & & & & 1 & & & 1 & & & & \\ & & & & & & & & & & & & 1 & 1 & & \\ & & & & & & & & & & & & 1 & 1 & & \\ & & & & & & & & & & & & 1 & & 1 & \\ & & & & & & & & & & & & 1 & & 1 & \\ & & & & & & & & & & & & 1 & & & 1 \\ & & & & & & & & & & & & 1 & & & 1 \end{bmatrix}$$

$$\mathbf{G} = \begin{bmatrix} \sigma^2_B & & & & & & & \\ & \sigma^2_{AB} & & & & & & \\ & & \sigma^2_{AB} & & & & & \\ & & & \sigma^2_{AB} & & & & \\ & & & & \ddots & & & \\ & & & & & \sigma^2_B & & \\ & & & & & & \sigma^2_{AB} & \\ & & & & & & & \sigma^2_{AB} \\ & & & & & & & & \sigma^2_{AB} \end{bmatrix}$$

Estimating Covariance Parameters in the Mixed Model

Estimation is more difficult in the mixed model than in the general linear model. Not only do you have $\boldsymbol{\beta}$ as in the general linear model, but you also have unknown parameters in $\boldsymbol{\gamma}$, $\mathbf{G}$, and $\mathbf{R}$. Least squares is no longer the best method. Generalized least squares (GLS) is more appropriate, minimizing

$$(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'\mathbf{V}^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

However, GLS requires knowledge of $\mathbf{V}$ and therefore knowledge of $\mathbf{G}$ and $\mathbf{R}$. Lacking such information, one approach is to use an *estimated* GLS, in which you insert some reasonable estimate for $\mathbf{V}$ into the minimization problem. The goal thus becomes finding a reasonable estimate of $\mathbf{G}$ and $\mathbf{R}$.

In many situations, the best approach is to use likelihood-based methods, exploiting the assumption that $\boldsymbol{\gamma}$ and $\boldsymbol{\epsilon}$ are normally distributed (Hartley and Rao 1967; Patterson and Thompson 1971; Harville 1977; Laird and Ware 1982; Jennrich and Schluchter 1986). PROC HPLMIXED implements two likelihood-based methods: maximum likelihood (ML) and restricted (residual) maximum likelihood (REML). A favorable theoretical property of ML and REML is that they accommodate data that are missing at random (Rubin 1976; Little 1995).

PROC HPLMIXED constructs an objective function associated with ML or REML and maximizes it over all unknown parameters. Using calculus, it is possible to reduce this maximization problem to one over only the parameters in $\mathbf{G}$ and $\mathbf{R}$. The corresponding log-likelihood functions are as follows:

$$\begin{aligned}
\text{ML}: \quad & l(\mathbf{G},\mathbf{R}) = -\frac{1}{2}\log|\mathbf{V}| - \frac{1}{2}\mathbf{r}'\mathbf{V}^{-1}\mathbf{r} - \frac{n}{2}\log(2\pi) \\
\text{REML}: \quad & l_R(\mathbf{G},\mathbf{R}) = -\frac{1}{2}\log|\mathbf{V}| - \frac{1}{2}\log|\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}| - \frac{1}{2}\mathbf{r}'\mathbf{V}^{-1}\mathbf{r} - \frac{n-p}{2}\log(2\pi)
\end{aligned}$$

where $\mathbf{r} = \mathbf{y} - \mathbf{X}(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-}\mathbf{X}'\mathbf{V}^{-1}\mathbf{y}$ and p is the rank of $\mathbf{X}$. By default, PROC HPLMIXED actually minimizes a normalized form of -2 times these functions by using a ridge-stabilized Newton-Raphson algorithm by default. Lindstrom and Bates (1988) provide reasons for preferring Newton-Raphson to the expectation-maximum (EM) algorithm described in Dempster, Laird, and Rubin (1977) and Laird, Lange, and Stram (1987), in addition to analytical details for implementing a QR-decomposition approach to the problem. Wolfinger, Tobias, and Sall (1994) present the sweep-based algorithms that are implemented in PROC HPLMIXED. You can change the optimization technique with the TECHNIQUE= option in the PROC HPLMIXED statement.

One advantage of using the Newton-Raphson algorithm is that the second derivative matrix of the objective function evaluated at the optima is available upon completion. Denoting this matrix $\mathbf{H}$, the asymptotic theory of maximum likelihood (Serfling 1980) shows that $2\mathbf{H}^{-1}$ is an asymptotic variance-covariance matrix of the estimated parameters of $\mathbf{G}$ and $\mathbf{R}$. Thus, tests and confidence intervals based on asymptotic normality can be obtained. However, these can be unreliable in small samples, especially for parameters such as variance components that have sampling distributions that tend to be skewed to the right.

If a residual variance σ^2 is a part of your mixed model, it can usually be *profiled* out of the likelihood. This means solving analytically for the optimal σ^2 and plugging this expression back into the likelihood formula (Wolfinger, Tobias, and Sall 1994). This reduces the number of optimization parameters by 1 and can improve convergence properties. PROC HPLMIXED profiles the residual variance out of the log likelihood.

Estimating Fixed and Random Effects in the Mixed Model

ML and REML methods provide estimates of $\mathbf{G}$ and $\mathbf{R}$, which are denoted $\widehat{\mathbf{G}}$ and $\widehat{\mathbf{R}}$, respectively. To obtain estimates of $\boldsymbol{\beta}$ and predicted values of $\boldsymbol{\gamma}$, the standard method is to solve the *mixed model equations* (Henderson 1984):

$$\begin{bmatrix} \mathbf{X}'\widehat{\mathbf{R}}^{-1}\mathbf{X} & \mathbf{X}'\widehat{\mathbf{R}}^{-1}\mathbf{Z} \\ \mathbf{Z}'\widehat{\mathbf{R}}^{-1}\mathbf{X} & \mathbf{Z}'\widehat{\mathbf{R}}^{-1}\mathbf{Z}+\widehat{\mathbf{G}}^{-1} \end{bmatrix} \begin{bmatrix} \widehat{\boldsymbol{\beta}} \\ \widehat{\boldsymbol{\gamma}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'\widehat{\mathbf{R}}^{-1}\mathbf{y} \\ \mathbf{Z}'\widehat{\mathbf{R}}^{-1}\mathbf{y} \end{bmatrix}$$

The solutions can also be written as

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}'\widehat{\mathbf{V}}^{-1}\mathbf{X})^{-}\mathbf{X}'\widehat{\mathbf{V}}^{-1}\mathbf{y}$$
$$\widehat{\boldsymbol{\gamma}} = \widehat{\mathbf{G}}\mathbf{Z}'\widehat{\mathbf{V}}^{-1}(\mathbf{y}-\mathbf{X}\widehat{\boldsymbol{\beta}})$$

and have connections with empirical Bayes estimators (Laird and Ware 1982; Carlin and Louis 1996). Note that the $\boldsymbol{\gamma}$ are random variables and not parameters (unknown constants) in the model. Technically, determining values for $\boldsymbol{\gamma}$ from the data is thus a prediction task, whereas determining values for $\boldsymbol{\beta}$ is an estimation task.

The mixed model equations are extended normal equations. The preceding expression assumes that $\widehat{\mathbf{G}}$ is nonsingular. For the extreme case where the eigenvalues of $\widehat{\mathbf{G}}$ are very large, $\widehat{\mathbf{G}}^{-1}$ contributes very little to the equations and $\widehat{\boldsymbol{\gamma}}$ is close to what it would be if $\boldsymbol{\gamma}$ actually contained fixed-effects parameters. On the other hand, when the eigenvalues of $\widehat{\mathbf{G}}$ are very small, $\widehat{\mathbf{G}}^{-1}$ dominates the equations and $\widehat{\boldsymbol{\gamma}}$ is close to 0. For intermediate cases, $\widehat{\mathbf{G}}^{-1}$ can be viewed as shrinking the fixed-effects estimates of $\boldsymbol{\gamma}$ toward 0 (Robinson 1991).

If $\widehat{\mathbf{G}}$ is singular, then the mixed model equations are modified (Henderson 1984) as follows:

$$\begin{bmatrix} \mathbf{X}'\widehat{\mathbf{R}}^{-1}\mathbf{X} & \mathbf{X}'\widehat{\mathbf{R}}^{-1}\mathbf{Z}\widehat{\mathbf{G}} \\ \widehat{\mathbf{G}}'\mathbf{Z}'\widehat{\mathbf{R}}^{-1}\mathbf{X} & \widehat{\mathbf{G}}'\mathbf{Z}'\widehat{\mathbf{R}}^{-1}\mathbf{Z}\widehat{\mathbf{G}}+\mathbf{G} \end{bmatrix} \begin{bmatrix} \widehat{\boldsymbol{\beta}} \\ \widehat{\boldsymbol{\tau}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'\widehat{\mathbf{R}}^{-1}\mathbf{y} \\ \widehat{\mathbf{G}}'\mathbf{Z}'\widehat{\mathbf{R}}^{-1}\mathbf{y} \end{bmatrix}$$

Denote the generalized inverses of the nonsingular $\widehat{\mathbf{G}}$ and singular $\widehat{\mathbf{G}}$ forms of the mixed model equations by $\mathbf{C}$ and $\mathbf{M}$, respectively. In the nonsingular case, the solution $\widehat{\boldsymbol{\gamma}}$ estimates the random effects directly. But in the singular case, the estimates of random effects are achieved through a back-transformation $\widehat{\boldsymbol{\gamma}} = \widehat{\mathbf{G}}\widehat{\boldsymbol{\tau}}$ where $\widehat{\boldsymbol{\tau}}$ is the solution to the modified mixed model equations. Similarly, while in the nonsingular case $\mathbf{C}$ itself is the estimated covariance matrix for $(\widehat{\boldsymbol{\beta}}, \widehat{\boldsymbol{\gamma}})$, in the singular case the covariance estimate for $(\widehat{\boldsymbol{\beta}}, \widehat{\mathbf{G}}\widehat{\boldsymbol{\tau}})$ is given by $\mathbf{PMP}$ where

$$\mathbf{P} = \begin{bmatrix} \mathbf{I} & \\ & \widehat{\mathbf{G}} \end{bmatrix}$$

An example of when the singular form of the equations is necessary is when a variance component estimate falls on the boundary constraint of 0.

Statistical Properties

If $\mathbf{G}$ and $\mathbf{R}$ are known, $\widehat{\boldsymbol{\beta}}$ is the best linear unbiased estimator (BLUE) of $\boldsymbol{\beta}$, and $\widehat{\boldsymbol{\gamma}}$ is the best linear unbiased predictor (BLUP) of $\boldsymbol{\gamma}$ (Searle 1971; Harville 1988, 1990; Robinson 1991; McLean, Sanders, and Stroup 1991). Here, "best" means minimum mean squared error. The covariance matrix of $(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}, \widehat{\boldsymbol{\gamma}} - \boldsymbol{\gamma})$ is

$$\mathbf{C} = \begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{Z} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z} + \mathbf{G}^{-1} \end{bmatrix}^{-}$$

where $^{-}$ denotes a generalized inverse (Searle 1971).

However, $\mathbf{G}$ and $\mathbf{R}$ are usually unknown and are estimated by using one of the aforementioned methods. These estimates, $\widehat{\mathbf{G}}$ and $\widehat{\mathbf{R}}$, are therefore simply substituted into the preceding expression to obtain

$$\widehat{\mathbf{C}} = \begin{bmatrix} \mathbf{X}'\widehat{\mathbf{R}}^{-1}\mathbf{X} & \mathbf{X}'\widehat{\mathbf{R}}^{-1}\mathbf{Z} \\ \mathbf{Z}'\widehat{\mathbf{R}}^{-1}\mathbf{X} & \mathbf{Z}'\widehat{\mathbf{R}}^{-1}\mathbf{Z} + \widehat{\mathbf{G}}^{-1} \end{bmatrix}^{-}$$

as the approximate variance-covariance matrix of $(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}, \widehat{\boldsymbol{\gamma}} - \boldsymbol{\gamma})$. In this case, the BLUE and BLUP acronyms no longer apply, but the word *empirical* is often added to indicate such an approximation. The appropriate acronyms thus become EBLUE and EBLUP.

McLean and Sanders (1988) show that $\widehat{\mathbf{C}}$ can also be written as

$$\widehat{\mathbf{C}} = \begin{bmatrix} \widehat{\mathbf{C}}_{11} & \widehat{\mathbf{C}}_{21}' \\ \widehat{\mathbf{C}}_{21} & \widehat{\mathbf{C}}_{22} \end{bmatrix}$$

where

$$\widehat{\mathbf{C}}_{11} = (\mathbf{X}'\widehat{\mathbf{V}}^{-1}\mathbf{X})^{-}$$
$$\widehat{\mathbf{C}}_{21} = -\widehat{\mathbf{G}}\mathbf{Z}'\widehat{\mathbf{V}}^{-1}\mathbf{X}\widehat{\mathbf{C}}_{11}$$
$$\widehat{\mathbf{C}}_{22} = (\mathbf{Z}'\widehat{\mathbf{R}}^{-1}\mathbf{Z} + \widehat{\mathbf{G}}^{-1})^{-1} - \widehat{\mathbf{C}}_{21}\mathbf{X}'\widehat{\mathbf{V}}^{-1}\mathbf{Z}\widehat{\mathbf{G}}$$

Note that $\widehat{\mathbf{C}}_{11}$ is the familiar estimated generalized least squares formula for the variance-covariance matrix of $\widehat{\boldsymbol{\beta}}$.

Computational Method

Distributed Computing

Distributed computing refers to the use of multiple autonomous computers that communicate through a secure network. Distributed computing solves computational problems by dividing them into many tasks, each of which is solved by one or more computers. Each computer in this distributed environment is referred to as a node.

You can specify the number of nodes to use with the NODES= option in the PERFORMANCE statement. Specify NODES=0 to force the execution to be done locally (often referred to as single-machine mode).

Multithreading

Threading refers to the organization of computational work into multiple tasks (processing units that can be scheduled by the operating system). A task is associated with a thread. Multithreading refers to the concurrent execution of threads. When multithreading is possible, substantial performance gains can be realized compared to sequential (single-threaded) execution.

The number of threads spawned by the HPLMIXED procedure is determined by the number of CPUs on a machine and can be controlled in the following ways:

You can specify the NTHREADS= option in the PERFORMANCE statement to determine the number of threads. This specification overrides the system option. Specify NTHREADS=1 to force single-threaded execution.

The number of threads per machine is displayed in the "Performance Information" table, which is part of the default output. The HPLMIXED procedure allocates two threads per CPU.

The tasks multithreaded by the HPLMIXED procedure are primarily defined by dividing the data processed on a single machine among the threads—that is, the HPLMIXED procedure implements multithreading through a data-parallel model. For example, if the input data set has 1,000 observations and you are running with four threads, then 250 observations are associated with each thread. All operations that require access to the data are then multithreaded. These operations include the following:

- variable levelization
- effect levelization
- formation of the crossproducts matrix
- the log-likelihood computation

In addition, operations on matrices such as sweeps might be multithreaded if the matrices are of sufficient size to realize performance benefits from managing multiple threads for the particular matrix operation.

Displayed Output

The following sections describe the output produced by PROC HPLMIXED. The output is organized into various tables, which are discussed in the order of their appearance.

Performance Information

The "Performance Information" table is produced by default. It displays information about the execution mode. For single-machine mode, the table displays the number of threads used. For distributed mode, the table displays the grid mode (symmetric or asymmetric), the number of compute nodes, and the number of threads per node.

If you specify the DETAILS option in the PERFORMANCE statement, PROC HPLMIXED also produces a "Timing" table that displays elapsed times (absolute and relative) for the main tasks of the procedure.

Model Information

The "Model Information" table describes the model, some of the variables it involves, and the method used in fitting it. The "Model Information" table also has a row labeled Fixed Effects SE Method. This row describes the method used to compute the approximate standard errors for the fixed-effects parameter estimates and related functions of them.

Class Level Information

The "Class Level Information" table lists the levels of every variable specified in the CLASS statement.

Dimensions

The "Dimensions" table lists the sizes of relevant matrices. This table can be useful in determining the requirements for CPU time and memory.

Number of Observations

The "Number of Observations" table shows the number of observations read from the data set and the number of observations used in fitting the model.

Optimization Information

The "Optimization Information" table displays important details about the optimization process.

The number of parameters that are updated in the optimization equals the number of parameters in this table minus the number of equality constraints. The number of constraints is displayed if you fix covariance parameters with the HOLD= option in the PARMS statement. The HPLMIXED procedure also lists the number of upper and lower boundary constraints. PROC HPLMIXED might impose boundary constraints for certain parameters, such as variance components and correlation parameters. If you specify the HOLD= option in the PARMS statement, covariance parameters have an upper and lower boundary equal to the parameter value.

Iteration History

The "Iteration History" table describes the optimization of the restricted log likelihood or log likelihood. The function to be minimized (the *objective function*) is $-2l$ for ML and $-2l_R$ for REML; the column name of the objective function in the "Iteration History" table is "-2 Log Like" for ML and "-2 Res Log Like" for REML. The minimization is performed by using a ridge-stabilized Newton-Raphson algorithm, and the rows of this table describe the iterations that this algorithm takes in order to minimize the objective function.

The Evaluations column of the "Iteration History" table tells how many times the objective function is evaluated during each iteration.

The Criterion column of the "Iteration History" table is, by default, a relative Hessian convergence quantity given by

$$\frac{\mathbf{g}_k' \mathbf{H}_k^{-1} \mathbf{g}_k}{|f_k|}$$

where f_k is the value of the objective function at iteration k, $\mathbf{g}_k$ is the gradient (first derivative) of f_k, and $\mathbf{H}_k$ is the Hessian (second derivative) of f_k. If $\mathbf{H}_k$ is singular, then PROC HPLMIXED uses the following relative quantity:

$$\frac{\mathbf{g}_k' \mathbf{g}_k}{|f_k|}$$

To prevent division by $|f_k|$, specify the ABSGCONV option in the PROC HPLMIXED statement. To use a relative function or gradient criterion, specify the FCONV or GCONV option, respectively.

The Hessian criterion is considered superior to function and gradient criteria because it measures orthogonality rather than lack of progress (Bates et al. 1987). Provided that the initial estimate is feasible and the maximum number of iterations is not exceeded, the Newton-Raphson algorithm is considered to have converged when the criterion is less than the tolerance specified with the FCONV or GCONV option in the PROC HPLMIXED statement. The default tolerance is 1E–8. If convergence is not achieved, PROC HPLMIXED displays the estimates of the parameters at the last iteration.

A convergence criterion that is missing indicates that a boundary constraint has been dropped; it is usually not a cause for concern.

Convergence Status

The "Convergence Status" table displays the status of the iterative estimation process at the end of the optimization. The status appears as a message in the listing, and this message is repeated in the log. The ODS object "ConvergenceStatus" also contains several nonprinting columns that can be helpful in checking the success of the iterative process, in particular during batch processing. The Status variable takes on the value 0 for a successful convergence (even if the Hessian matrix might not be positive definite). The values 1 and 2 of the Status variable indicate lack of convergence and infeasible initial parameter values, respectively. The variable pdG can be used to check whether the $\mathbf{G}$ matrix is positive definite.

For models that are not fit iteratively, such as models without random effects or when the NOITER option is in effect, the "Convergence Status" is not produced.

Covariance Parameter Estimates

The "Covariance Parameter Estimates" table contains the estimates of the parameters in $\mathbf{G}$ and $\mathbf{R}$. (See the section "Estimating Covariance Parameters in the Mixed Model" on page 316.) Their values are labeled in the table along with Subject information if applicable. The estimates are displayed in the Estimate column and are the results of either the REML or the ML estimation method.

Fit Statistics

The "Fit Statistics" table provides some statistics about the estimated mixed model. Expressions for -2 times the log likelihood are provided in the section "Estimating Covariance Parameters in the Mixed Model" on page 316. If the log likelihood is an extremely large number, then PROC HPLMIXED has deemed the estimated $\mathbf{V}$ matrix to be singular. In this case, all subsequent results should be viewed with caution.

In addition, the "Fit Statistics" table lists three information criteria: AIC, AICC, and BIC. All these criteria are in smaller-is-better form and are described in Table 8.9.

Table 8.9 Information Criteria

Criterion	Formula	Reference
AIC	$-2\ell + 2d$	Akaike (1974)
AICC	$-2\ell + 2dn^*/(n^* - d - 1)$	Hurvich and Tsai (1989) Burnham and Anderson (1998)
BIC	$-2\ell + d \log n$ for $n > 0$	Schwarz (1978)

Here ℓ denotes the maximum value of the (possibly restricted) log likelihood; d is the dimension of the

model; and n equals the number of effective subjects as displayed in the "Dimensions" table, unless this value equals 1, in which case n equals the number of levels of the first random effect specified in the first RANDOM statement or the number of levels of the interaction of the first random effect with noncommon subject effect specified in the first RANDOM statement. If the number of effective subjects equals 1 and you have no RANDOM statements, then n equals the number of valid observations for maximum likelihood estimation and $n - p$ for restricted maximum likelihood estimation, where p equals the rank of $\mathbf{X}$. For AICC (a finite-sample corrected version of AIC), n^* equals the number of valid observations for maximum likelihood estimation and $n - p$ equals the number of valid observations for restricted maximum likelihood estimation, unless this number is less than $d + 2$, in which case it equals $d + 2$. When $n = 0$, the value of the BIC is -2ℓ. For restricted likelihood estimation, d equals q, the effective number of estimated covariance parameters. For maximum likelihood estimation, d equals $q + p$.

Timing Information

If you specify the DETAILS option in the PERFORMANCE statement, the procedure also produces a "Timing" table in which the elapsed time for each main task of the procedure is displayed.

ODS Table Names

Each table created by PROC HPLMIXED has a name associated with it, and you must use this name to refer to the table when you use ODS statements. These names are listed in Table 8.10.

Table 8.10 ODS Tables Produced by PROC HPLMIXED

Table Name	Description	Required Statement / Option
ClassLevels	Level information from the CLASS statement	Default output
ConvergenceStatus	Convergence status	Default output
CovParms	Estimated covariance parameters	Default output
Dimensions	Dimensions of the model	Default output
FitStatistics	Fit statistics	Default output
IterHistory	Iteration history	Default output
ModelInfo	Model information	Default output
NObs	Number of observations read and used	Default output
OptInfo	Optimization information	Default output
ParmSearch	Parameter search values	PARMS
PerformanceInfo	Information about high-performance computing environment	Default output
Ranks	Rank of designed matrix X	PROC HPLMIXED RANKS
SolutionF	Fixed-effects solution vector	MODEL / S
SolutionR	Random-effects solution vector	RANDOM / S
Timing	Timing breakdown by task	DETAILS option in the PERFORMANCE statement

Examples: HPLMIXED Procedure

Example 8.1: Computing BLUPs for a Large Number of Subjects

Suppose you are using health measurements on patients treated by each medical center to monitor the performance of those centers. Different measurements within each patient are correlated, and there is enough data to fit the parameters of an unstructured covariance model for this correlation. In fact, long experience with historical data provides you with values for the covariance model that are essentially known, and the task is to apply these known values in order to compute best linear unbiased predictors (BLUPs) of the random effect of medical center. You can use these BLUPs to determine the best and worst performing medical centers, adjusting for other factors, on a weekly basis. Another reason why you want to do this with fixed values for the covariance parameters is to make the week-to-week BLUPs more comparable.

Although you cannot use the REPEATED statement in PROC HPLMIXED to fit models in this release, you can use it to compute BLUPs for such models with known values of the variance parameters. For illustration, the following statements create a simulated data set of a given week's worth of patient health measurements across 100 different medical centers. Measurements at three different times are simulated for each patient, and each center has about 50 patients. The simulated model includes a fixed gender effect, a random effect due to center, and covariance between different measurements on the same patient.

```
%let NCenter  = 100;
%let NPatient = %eval(&NCenter*50);
%let NTime    = 3;
%let SigmaC   = 2.0;
%let SigmaP   = 4.0;
%let SigmaE   = 8.0;
%let Seed     = 12345;

data WeekSim;
   keep Gender Center Patient Time Measurement;
   array PGender{&NPatient};
   array PCenter{&NPatient};
   array PEffect{&NPatient};
   array CEffect{&NCenter};
   array GEffect{2};

   do Center = 1 to &NCenter;
      CEffect{Center} = sqrt(&SigmaC)*rannor(&Seed);
      end;

   GEffect{1} = 10*ranuni(&Seed);
   GEffect{2} = 10*ranuni(&Seed);

   do Patient = 1 to &NPatient;
      PGender{Patient} = 1 + int(2       *ranuni(&Seed));
      PCenter{Patient} = 1 + int(&NCenter*ranuni(&Seed));
      PEffect{Patient} = sqrt(&SigmaP)*rannor(&Seed);
      end;
```

```
      do Patient = 1 to &NPatient;
         Gender = PGender{Patient};
         Center = PCenter{Patient};
         Mean = 1 + GEffect{Gender} + CEffect{Center} + PEffect{Patient};
         do Time = 1 to &nTime;
            Measurement = Mean + sqrt(&SigmaE)*rannor(&Seed);
         output;
         end;
      end;
   run;
```

Suppose that the known values for the covariance parameters are

$$\text{Var(Center)} = 1.7564$$

$$\text{Cov(Patient)} = \begin{bmatrix} 11.4555 & 3.6883 & 4.5951 \\ 3.6883 & 11.2071 & 3.6311 \\ 4.5951 & 3.6311 & 12.1050 \end{bmatrix}$$

Incidentally, these are not precisely the same estimates you would get if you estimated these parameters based on the preceding data (for example, with the HPLMIXED procedure).

The following statements use PROC HPLMIXED to compute the BLUPs for the random medical center effect. Instead of simply displaying them (as PROC HPMIXED does), PROC HPLMIXED sorts them and displays the five highest and lowest values. To run these statements successfully, you need to set the macro variables GRIDHOST and GRIDINSTALLLOC to resolve to appropriate values, or you can replace the references to macro variables with appropriate values.

```
ods listing close;
proc hplmixed data=WeekSim blup;
   performance host="&GRIDHOST" install="&GRIDINSTALLLOC" nodes=20;
   class Gender Center Patient Time;
   model Measurement = Gender;
   random   Center / s;
   repeated Time   / sub=Patient type=un;
   parms   1.7564
          11.4555
           3.6883 11.2071
           4.5951  3.6311 12.1050;
   ods output SolutionR=BLUPs;
run;
ods listing;

proc sort data=BLUPs;
   by Estimate;
run;

data BLUPs; set BLUPs;
   Rank = _N_;
run;
```

```
proc print data=BLUPs;
   where ((Rank <= 5) | (Rank >= 96));
   var Center Estimate;
run;
```

Three parts of the PROC HPLMIXED syntax are required in order to compute BLUPs for this model: the BLUP option in the HPLMIXED statement, the REPEATED statement, and the PARMS statement with fixed values for all parameters. The resulting values of the best and worst performing medical centers for this week are shown in Output 8.1.1. Apparently, for this week's data, medical center 54 had the most decreasing effect, and medical center 48 the most increasing effect, on patient measurements overall.

Output 8.1.1 Highest and Lowest Medical Center BLUPs

Obs	Center	Estimate
1	54	-2.9369
2	7	-2.4614
3	50	-2.2467
4	51	-2.2281
5	93	-2.1644
96	26	2.1603
97	99	2.2718
98	44	2.4222
99	60	2.6089
100	48	2.6443

References

Akaike, H. (1974), "A New Look at the Statistical Model Identification," *IEEE Transactions on Automatic Control*, AC-19, 716–723.

Bates, D. M., Lindstrom, M. J., Wahba, G., and Yandell, B. S. (1987), "GCVPACK-Routines for Generalized Cross Validation," *Communications in Statistics—Simulation and Computation*, 16, 263–297.

Burdick, R. K. and Graybill, F. A. (1992), *Confidence Intervals on Variance Components*, New York: Marcel Dekker.

Burnham, K. P. and Anderson, D. R. (1998), *Model Selection and Inference: A Practical Information-Theoretic Approach*, New York: Springer-Verlag.

Carlin, B. P. and Louis, T. A. (1996), *Bayes and Empirical Bayes Methods for Data Analysis*, London: Chapman & Hall.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977), "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society, Series B*, 39, 1–38.

Fuller, W. A. (1976), *Introduction to Statistical Time Series*, New York: John Wiley & Sons.

Hartley, H. O. and Rao, J. N. K. (1967), "Maximum-Likelihood Estimation for the Mixed Analysis of Variance Model," *Biometrika*, 54, 93–108.

Harville, D. A. (1977), "Maximum Likelihood Approaches to Variance Component Estimation and to Related Problems," *Journal of the American Statistical Association*, 72, 320–338.

Harville, D. A. (1988), "Mixed-Model Methodology: Theoretical Justifications and Future Directions," in *Proceedings of the Statistical Computing Section*, 41–49, Alexandria, VA: American Statistical Association.

Harville, D. A. (1990), "BLUP (Best Linear Unbiased Prediction), and Beyond," in D. Gianola and K. Hammond, eds., *Advances in Statistical Methods for Genetic Improvement of Livestock*, volume 18 of *Advanced Series in Agricultural Sciences*, 239–276, Berlin: Springer-Verlag.

Henderson, C. R. (1984), *Applications of Linear Models in Animal Breeding*, Guelph, ON: University of Guelph.

Henderson, C. R. (1990), "Statistical Method in Animal Improvement: Historical Overview," in *Advances in Statistical Methods for Genetic Improvement of Livestock*, 1–14, New York: Springer-Verlag.

Hurvich, C. M. and Tsai, C.-L. (1989), "Regression and Time Series Model Selection in Small Samples," *Biometrika*, 76, 297–307.

Huynh, H. and Feldt, L. S. (1970), "Conditions Under Which Mean Square Ratios in Repeated Measurements Designs Have Exact F-Distributions," *Journal of the American Statistical Association*, 65, 1582–1589.

Jennrich, R. I. and Schluchter, M. D. (1986), "Unbalanced Repeated-Measures Models with Structured Covariance Matrices," *Biometrics*, 42, 805–820.

Kenward, M. G. (1987), "A Method for Comparing Profiles of Repeated Measurements," *Applied Statistics*, 36, 296–308.

Laird, N. M., Lange, N. T., and Stram, D. O. (1987), "Maximum Likelihood Computations with Repeated Measures: Application of the EM Algorithm," *Journal of the American Statistical Association*, 82, 97–105.

Laird, N. M. and Ware, J. H. (1982), "Random-Effects Models for Longitudinal Data," *Biometrics*, 38, 963–974.

Lindstrom, M. J. and Bates, D. M. (1988), "Newton-Raphson and EM Algorithms for Linear Mixed-Effects Models for Repeated-Measures Data," *Journal of the American Statistical Association*, 83, 1014–1022.

Littell, R. C., Milliken, G. A., Stroup, W. W., Wolfinger, R. D., and Schabenberger, O. (2006), *SAS for Mixed Models*, 2nd Edition, Cary, NC: SAS Institute Inc.

Little, R. J. A. (1995), "Modeling the Drop-Out Mechanism in Repeated-Measures Studies," *Journal of the American Statistical Association*, 90, 1112–1121.

Macchiavelli, R. E. and Arnold, S. F. (1994), "Variable Order Ante-dependence Models," *Communications in Statistics—Theory and Methods*, 23, 2683–2699.

McLean, R. A. and Sanders, W. L. (1988), "Approximating Degrees of Freedom for Standard Errors in Mixed Linear Models," in *Proceedings of the Statistical Computing Section*, 50–59, Alexandria, VA: American Statistical Association.

McLean, R. A., Sanders, W. L., and Stroup, W. W. (1991), "A Unified Approach to Mixed Linear Models," *American Statistician*, 45, 54–64.

Milliken, G. A. and Johnson, D. E. (1992), *Designed Experiments*, volume 1 of *Analysis of Messy Data*, New York: Chapman & Hall.

Patel, H. I. (1991), "Analysis of Incomplete Data from a Clinical Trial with Repeated Measurements," *Biometrika*, 78, 609–619.

Patterson, H. D. and Thompson, R. (1971), "Recovery of Inter-block Information When Block Sizes Are Unequal," *Biometrika*, 58, 545–554.

Robinson, G. K. (1991), "That BLUP Is a Good Thing: The Estimation of Random Effects," *Statistical Science*, 6, 15–51.

Rubin, D. B. (1976), "Inference and Missing Data," *Biometrika*, 63, 581–592.

Schwarz, G. (1978), "Estimating the Dimension of a Model," *Annals of Statistics*, 6, 461–464.

Searle, S. R. (1971), *Linear Models*, New York: John Wiley & Sons.

Searle, S. R. (1982), *Matrix Algebra Useful for Statisticians*, New York: John Wiley & Sons.

Searle, S. R., Casella, G., and McCulloch, C. E. (1992), *Variance Components*, New York: John Wiley & Sons.

Serfling, R. J. (1980), *Approximation Theorems of Mathematical Statistics*, New York: John Wiley & Sons.

Snedecor, G. W. and Cochran, W. G. (1980), *Statistical Methods*, 7th Edition, Ames: Iowa State University Press.

Steel, R. G. D., Torrie, J. H., and Dickey, D. A. (1997), *Principles and Procedures of Statistics: A Biometrical Approach*, 3rd Edition, New York: McGraw-Hill.

Verbeke, G. and Molenberghs, G., eds. (1997), *Linear Mixed Models in Practice: A SAS-Oriented Approach*, New York: Springer.

Verbeke, G. and Molenberghs, G. (2000), *Linear Mixed Models for Longitudinal Data*, New York: Springer.

Winer, B. J. (1971), *Statistical Principles in Experimental Design*, 2nd Edition, New York: McGraw-Hill.

Wolfinger, R. D. (1996), "Heterogeneous Variance-Covariance Structures for Repeated Measures," *Journal of Agricultural, Biological, and Environmental Statistics*, 1, 205–230.

Wolfinger, R. D., Tobias, R. D., and Sall, J. (1994), "Computing Gaussian Likelihoods and Their Derivatives for General Linear Mixed Models," *SIAM Journal on Scientific Computing*, 15, 1294–1310.

Chapter 9
The HPLOGISTIC Procedure

Contents

Overview: HPLOGISTIC Procedure **330**
PROC HPLOGISTIC Features 330
PROC HPLOGISTIC Contrasted with Other SAS Procedures 331
Getting Started: HPLOGISTIC Procedure **332**
Binary Logistic Regression 332
Syntax: HPLOGISTIC Procedure **337**
PROC HPLOGISTIC Statement 338
BY Statement 343
CLASS Statement 344
CODE Statement 344
FREQ Statement 345
ID Statement 345
MODEL Statement 345
OUTPUT Statement 352
PARTITION Statement 354
PERFORMANCE Statement 354
SELECTION Statement 355
WEIGHT Statement 356
Details: HPLOGISTIC Procedure **356**
Missing Values 356
Response Distributions 356
Log-Likelihood Functions 357
Existence of Maximum Likelihood Estimates 359
Using Validation and Test Data 360
Model Fit and Assessment Statistics 361
The Hosmer-Lemeshow Goodness-of-Fit Test 366
Computational Method: Multithreading 366
Choosing an Optimization Algorithm 367
First- or Second-Order Algorithms 367
Algorithm Descriptions 368
Displayed Output 371
ODS Table Names 375
Examples: HPLOGISTIC Procedure **377**
Example 9.1: Model Selection 377
Example 9.2: Modeling Binomial Data 380
Example 9.3: Ordinal Logistic Regression 387
References **390**

Overview: HPLOGISTIC Procedure

The HPLOGISTIC procedure is a high-performance statistical procedure that fits logistic regression models for binary, binomial, and multinomial data on the SAS appliance.

The HPLOGISTIC procedure fits logistic regression models in the broader sense; the procedure permits several link functions and can handle ordinal and nominal data with more than two response categories (multinomial data).

PROC HPLOGISTIC runs in either single-machine mode or distributed mode.

NOTE: Distributed mode requires SAS High-Performance Statistics.

PROC HPLOGISTIC Features

The HPLOGISTIC procedure estimates the parameters of a logistic regression model by using maximum likelihood techniques. It also does the following:

- provides model-building syntax with the CLASS and effect-based MODEL statements, which are familiar from SAS/STAT analytic procedures (in particular, the GLM, LOGISTIC, GLIMMIX, and MIXED procedures)
- provides response-variable options as in the LOGISTIC procedure
- performs maximum likelihood estimation
- provides multiple link functions
- provides cumulative link models for ordinal data and generalized logit modeling for unordered multinomial data
- enables model building (variable selection) through the SELECTION statement
- provides a WEIGHT statement for weighted analysis
- provides a FREQ statement for grouped analysis
- provides an OUTPUT statement to produce a data set with predicted probabilities and other observationwise statistics

Because the HPLOGISTIC procedure is a high-performance statistical procedure, it also does the following:

- enables you to run in distributed mode on a cluster of machines that distribute the data and the computations

- enables you to run in single-machine mode on the server where SAS is installed
- exploits all the available cores and concurrent threads, regardless of execution mode
- performs parallel reads of input data and parallel writes of output data when the data source is the appliance database

For more information, see the section "Processing Modes" on page 10 in Chapter 3, "Shared Concepts and Topics."

PROC HPLOGISTIC Contrasted with Other SAS Procedures

For general contrasts, see the section "Common Features of SAS High-Performance Statistical Procedures" on page 42. The following remarks contrast the HPLOGISTIC procedure with the LOGISTIC procedure in SAS/STAT software.

The CLASS statement in the HPLOGISTIC procedure permits two parameterizations: the GLM parameterization and a reference parameterization. In contrast to the LOGISTIC, GENMOD, and other procedures that permit multiple parameterizations, the HPLOGISTIC procedure does not mix parameterizations across the variables in the CLASS statement. In other words, all classification variables have the same parameterization, and this parameterization is either the GLM or reference parameterization.

The default parameterization of CLASS variables in the HPLOGISTIC procedure is the GLM parameterization. The LOGISTIC procedure uses the EFFECT parameterization for the CLASS variables by default. In either procedure, you can change the parameterization with the PARAM= option in the CLASS statement.

The default optimization technique used by the LOGISTIC procedure is Fisher scoring; the HPLOGISTIC procedure uses by default a modification of the Newton-Raphson algorithm with a ridged Hessian. You can choose different optimization techniques, including first-order methods that do not require a crossproducts matrix or Hessian, with the TECHNIQUE= option in the PROC HPLOGISTIC statement.

The LOGISTIC procedure offers a wide variety of postfitting analyses, such as contrasts, estimates, tests of model effects, least squares means, and odds ratios. This release of the HPLOGISTIC procedure is limited in postfitting functionality, since with large data sets the focus is primarily on model fitting and scoring.

The HPLOGISTIC procedure is specifically designed to operate in the high-performance distributed environment. By default, PROC HPLOGISTIC performs computations in multiple threads. The LOGISTIC procedure executes in a single thread.

Getting Started: HPLOGISTIC Procedure

Binary Logistic Regression

The following DATA step contains 100 observations on a dichotomous response variable (y), a character variable (C), and 10 continuous variables (x1–x10):

```
data getStarted;
   input C$ y x1-x10;
   datalines;
   D  0  10.2  6  1.6  38  15  2.4  20  0.8  8.5  3.9
   F  1  12.2  6  2.6  42  61  1.5  10  0.6  8.5  0.7
   D  1   7.7  1  2.1  38  61    1  90  0.6  7.5  5.2
   J  1  10.9  7  3.5  46  42  0.3   0  0.2    6  3.6
   E  0  17.3  6  3.8  26  47  0.9  10  0.4  1.5  4.7
   A  0  18.7  4  1.8   2  34  1.7  80    1  9.5  2.2
   B  0   7.2  1  0.3  48  61  1.1  10  0.8  3.5    4
   D  0   0.1  3  2.4   0  65  1.6  70  0.8  3.5  0.7
   H  1   2.4  4  0.7  38  22  0.2  20    0    3  4.2
   J  0  15.6  7  1.4   0  98  0.3   0    1    5  5.2
   J  0  11.1  3  2.4  42  55  2.2  60  0.6  4.5  0.7
   F  0     4  6  0.9   4  36  2.1  30  0.8    9  4.6
   A  0   6.2  2  1.8  14  79  1.1  70  0.2    0  5.1
   H  0   3.7  3  0.8  12  66  1.3  40  0.4  0.5  3.3
   A  1   9.2  3  2.3  48  51  2.3  50    0    6  5.4
   G  0    14  3    2  18  12  2.2   0    0    3  3.4
   E  1  19.5  6  3.7  26  81  0.1  30  0.6    5  4.8
   C  0    11  3  2.8  38   9  1.7  50  0.8  6.5  0.9
   I  0  15.3  7  2.2  20  98  2.7 100  0.4    7  0.8
   H  1   7.4  4  0.5  28  65  1.3  60  0.2  9.5  5.4
   F  0  11.4  2  1.4  42  12  2.4  10  0.4    1  4.5
   C  1  19.4  1  0.4  42   4  2.4  10    0  6.5  0.1
   G  0   5.9  4  2.6  12  57  0.8  50  0.4    2  5.8
   G  1  15.8  6  3.7  34   8  1.3  90  0.6  2.5  5.7
   I  0    10  3  1.9  16  80    3  90  0.4  9.5  1.9
   E  0  15.7  1  2.7  32  25  1.7  20  0.2  8.5    6
   G  0    11  5  2.9  48  53  0.1  50    1  3.5  1.2
   J  1  16.8  0  0.9  14  86  1.4  40  0.8    9    5
   D  1    11  4  3.2  48  63  2.8  90  0.6    0  2.2
   J  1   4.8  7  3.6  24   1  2.2  20    1  8.5  0.5
   J  1  10.4  5    2  42  56    1  20    0  3.5  4.2
   G  0  12.7  7  3.6   8  56  2.1  70    1  4.5  1.5
   G  0   6.8  1  3.2  30  27  0.6   0  0.8    2  5.6
   E  0   8.8  0  3.2   2  67  0.7  10  0.4    1    5
   I  1   0.2  0  2.9  10  41  2.3  60  0.2    9  0.3
   J  1   4.6  7  3.9  50  61  2.1  50  0.4    3  4.9
   J  1   2.3  2  3.2  36  98  0.1  40  0.6  4.5  4.3
   I  0  10.8  3  2.7  28  58  0.8  80  0.8    3    6
   B  0   9.3  2  3.3  44  44  0.3  50  0.8  5.5  0.4
   F  0   9.2  6  0.6   4  64  0.1   0  0.6  4.5  3.9
```

```
D  0   7.4  0  2.9  14   0  0.2  30  0.8  7.5  4.5
G  0  18.3  3  3.1   8  60  0.3  60  0.2    7  1.9
F  0   5.3  4  0.2  48  63  2.3  80  0.2    8  5.2
C  0   2.6  5  2.2  24   4  1.3  20    0    2  1.4
F  0  13.8  4  3.6   4   7  1.1  10  0.4  3.5  1.9
B  1  12.4  6  1.7  30  44  1.1  60  0.2    6  1.5
I  0   1.3  1  1.3   8  53  1.1  70  0.6    7  0.8
F  0  18.2  7  1.7  26  92  2.2  30    1  8.5  4.8
J  0   5.2  2  2.2  18  12  1.4  90  0.8    4  4.9
G  1   9.4  2  0.8  22  86  0.4  30  0.4    1  5.9
J  1  10.4  2  1.7  26  31  2.4  10  0.2    7  1.6
J  0    13  1  1.8  14  11  2.3  50  0.6  5.5  2.6
A  0  17.9  4  3.1  46  58  2.6  90  0.6  1.5  3.2
D  1  19.4  6    3  20  50  2.8 100  0.2    9  1.2
I  0  19.6  3  3.6  22  19  1.2   0  0.6    5  4.1
I  1     6  2  1.5  30  30  2.2  20  0.4  8.5  5.3
G  0  13.8  1  2.7   0  52  2.4  20  0.8    6    2
B  0  14.3  4  2.9  30  11  0.6  90  0.6  0.5  4.9
E  0  15.6  0  0.4  38  79  0.4  80  0.4    1  3.3
D  0    14  2    1  22  61    3  90  0.6    2  0.1
C  1   9.4  5  0.4  12  53  1.7  40    0    3  1.1
H  0  13.2  1  1.6  40  15  0.7  40  0.2    9  5.5
A  0  13.5  5  2.4  18  89  1.6  20  0.4  9.5  4.7
E  0   2.6  4  2.3  38   6  0.8  20  0.4    5  5.3
E  0  12.4  3  1.3  26   8  2.8  10  0.8    6  5.8
D  0   7.6  2  0.9  44  89  1.3  50  0.8    6  0.4
I  0  12.7  1  2.3  42   6  2.4  10  0.4    1    3
C  1  10.7  4  3.2  28  23  2.2  90  0.8  5.5  2.8
H  0  10.1  2  2.3  10  62  0.9  50  0.4  2.5  3.7
C  1  16.6  1  0.5  12  88  0.1  20  0.6  5.5  1.8
I  1   0.2  3  2.2   8  71  1.7  80  0.4  0.5  5.5
C  0  10.8  4  3.5  30  70  2.3  60  0.4  4.5  5.9
F  0   7.1  4    3  14  63  2.4  70    0    7  3.1
D  0  16.5  1  3.3  30  80  1.6  40    0  3.5  2.7
H  0  17.1  7  2.1  30  45  1.5  60  0.6  0.5  2.8
D  0   4.3  1  1.5  24  44    0  70    0    5  0.5
H  0    15  2  0.2  14  87  1.8  50    0  4.5  4.7
G  0  19.7  3  1.9  36  99  1.5  10  0.6    3  1.7
H  1   2.8  6  0.6  34  21    2  60    1    9  4.7
G  0  16.6  3  3.3  46   1  1.4  70  0.6  1.5  5.3
E  0  11.7  5  2.7  48   4  0.9  60  0.8  4.5  1.6
F  0  15.6  3  0.2   4  79  0.5   0  0.8  1.5  2.9
C  1   5.3  6  1.4   8  64    2  80  0.4    9  4.2
B  1   8.1  7  1.7  40  36  1.4  60  0.6    6  3.9
I  0  14.8  2  3.2   8  37  0.4  10    0  4.5    3
D  0   7.4  4    3  12   3  0.6  60  0.6    7  0.7
D  0   4.8  3  2.3  44  41  1.9  60  0.2    3  3.1
A  0   4.5  0  0.2   4  48  1.7  80  0.8    9  4.2
D  0   6.9  6  3.3  14  92  0.5  40  0.4  7.5    5
B  0   4.7  4  0.9  14  99  2.4  80    1  0.5  0.7
I  1   7.5  4  2.1  20  79  0.4  40  0.4  2.5  0.7
C  0   6.1  0  1.4  38  18  2.3  60  0.8  4.5  0.7
C  0  18.3  1    1  26  98  2.7  20    1  8.5  0.5
F  0  16.4  7  1.2  32  94  2.9  40  0.4  5.5  2.1
```

```
   I  0   9.4  2  2.3  32  42  0.2  70  0.4  8.5  0.3
   F  1  17.9  4  1.3  32  42    2  40  0.2    1  5.4
   H  0  14.9  3  1.6  36  74  2.6  60  0.2    1  2.3
   C  0  12.7  0  2.6   0  88  1.1  80  0.8  0.5  2.1
   F  0   5.4  4  1.5   2   1  1.8  70  0.4  5.5  3.6
   J  1  12.1  4  1.8  20  59  1.3  60  0.4    3  3.8
;
```

The following statements fit a logistic model to these data by using a classification effect for variable C and 10 regressor effects for x1–x10:

```
proc hplogistic data=getStarted;
   class C;
   model y = C x1-x10;
run;
```

The default output from this analysis is presented in Figure 9.1 through Figure 9.11.

The "Performance Information" table in Figure 9.1 shows that the procedure executes in single-machine mode—that is, the model is fit on the machine where the SAS session executes. This run of the HPLOGISTIC procedure was performed on a multicore machine with the same number of CPUs as there are threads; that is, one computational thread was spawned per CPU.

Figure 9.1 Performance Information

The HPLOGISTIC Procedure

Performance Information	
Execution Mode	Single-Machine
Number of Threads	4

Figure 9.2 displays the "Model Information" table. The HPLOGISTIC procedure uses a Newton-Raphson algorithm to model a binary distribution for the variable y with a logit link function. The CLASS variable C is parameterized using the GLM parameterization, which is the default.

Figure 9.2 Model Information

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

The CLASS variable C has 10 unique formatted levels, and these are displayed in the "Class Level Information" table in Figure 9.3.

Figure 9.3 Class Level Information

Class Level Information		
Class	**Levels**	**Values**
C	10	A B C D E F G H I J

Figure 9.4 displays the "Number of Observations" table. All 100 observations in the data set are used in the analysis.

Figure 9.4 Number of Observations

Number of Observations Read	100
Number of Observations Used	100

The "Response Profile" table in Figure 9.5 is produced by default for binary and multinomial response variables. It shows the breakdown of the response variable levels by frequency. By default for binary data, the HPLOGISTIC procedure models the probability of the event with the lower-ordered value in the "Response Profile" table—this is indicated by the note that follows the table. In this example, the values represented by y = '0' are modeled as the "successes" in the Bernoulli experiments.

Figure 9.5 Response Profile

Response Profile		
Ordered Value	**y**	**Total Frequency**
1	0	69
2	1	31

You are modeling the probability that y='0'.

You can use the response-variable options in the MODEL statement to affect which value of the response variable is modeled.

Figure 9.6 displays the "Dimensions" table for this model. This table summarizes some important sizes of various model components. For example, it shows that there are 21 columns in the design matrix $\mathbf{X}$, which correspond to one column for the intercept, 10 columns for the effect associated with the classification variable C, and one column each for the continuous variables x1–x10. However, the rank of the crossproducts matrix is only 20. Because the classification variable C uses GLM parameterization and because the model contains an intercept, there is one singularity in the crossproducts matrix of the model. Consequently, only 20 parameters enter the optimization.

Figure 9.6 Dimensions in Binomial Logistic Regression

Dimensions	
Columns in X	21
Number of Effects	12
Max Effect Columns	10
Rank of Cross-product Matrix	20
Parameters in Optimization	20

The "Iteration History" table is shown in Figure 9.7. The Newton-Raphson algorithm with ridging converged after four iterations, not counting the initial setup iteration.

Figure 9.7 Iteration History

Iteration History				
Iteration	Evaluations	Objective Function	Change	Max Gradient
0	4	0.4493546916	.	0.410972
1	2	0.4436453992	0.00570929	0.081339
2	2	0.4435038109	0.00014159	0.003302
3	2	0.4435035933	0.00000022	5.623E-6
4	2	0.4435035933	0.00000000	1.59E-11

Figure 9.8 displays the final convergence status of the Newton-Raphson algorithm. The GCONV= relative convergence criterion is satisfied.

Figure 9.8 Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

The "Fit Statistics" table is shown in Figure 9.9. The –2 log likelihood at the converged estimates is 88.7007. You can use this value to compare the model to nested model alternatives by means of a likelihood-ratio test. To compare models that are not nested, information criteria such as AIC (Akaike's information criterion), AICC (Akaike's bias-corrected information criterion), and BIC (Schwarz' Bayesian information criterion) are used. These criteria penalize the –2 log likelihood for the number of parameters. Because of the large number of parameters relative to the number of observations, the discrepancy between the –2 log likelihood and, say, AIC, is substantial in this case.

Figure 9.9 Fit Statistics

Fit Statistics	
-2 Log Likelihood	88.7007
AIC (smaller is better)	128.70
AICC (smaller is better)	139.33
BIC (smaller is better)	180.80

Figure 9.10 shows the global test for the null hypothesis that all model effects jointly do not affect the probability of success of the binary response. The test is significant (p-value = 0.0135). One or more of the model effects thus significantly affects the probability of observing an event.

Figure 9.10 Null Test

Testing Global Null Hypothesis: BETA=0			
Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	35.1194	19	0.0135

However, a look at the "Parameter Estimates" table in Figure 9.11 shows that many parameters have fairly large p-values, indicating that one or more of the model effects might not be necessary.

Figure 9.11 Parameter Estimates

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept	1.2101	1.7507	Infty	0.69	0.4894
C A	3.4341	1.6131	Infty	2.13	0.0333
C B	2.1638	1.4271	Infty	1.52	0.1295
C C	0.6552	1.0810	Infty	0.61	0.5445
C D	2.4945	1.1094	Infty	2.25	0.0245
C E	3.2449	1.4321	Infty	2.27	0.0235
C F	3.6054	1.3070	Infty	2.76	0.0058
C G	2.0841	1.1898	Infty	1.75	0.0798
C H	2.9368	1.2939	Infty	2.27	0.0232
C I	1.3785	1.0319	Infty	1.34	0.1816
C J	0	.	.	.	.
x1	0.03218	0.05710	Infty	0.56	0.5730
x2	-0.3677	0.1538	Infty	-2.39	0.0168
x3	0.3146	0.3574	Infty	0.88	0.3787
x4	-0.05196	0.02443	Infty	-2.13	0.0334
x5	-0.00683	0.01056	Infty	-0.65	0.5177
x6	0.2539	0.3785	Infty	0.67	0.5024
x7	-0.00723	0.01073	Infty	-0.67	0.5004
x8	2.5370	0.9942	Infty	2.55	0.0107
x9	-0.1675	0.1068	Infty	-1.57	0.1168
x10	-0.2222	0.1577	Infty	-1.41	0.1590

Syntax: HPLOGISTIC Procedure

The following statements are available in the HPLOGISTIC procedure:

PROC HPLOGISTIC < *options* > **;**
BY *variables* **;**
CLASS *variable* < **(***options***)** >. . . < *variable* < **(***options***)** > > < / *global-options* > **;**
CODE < *options* > **;**
FREQ *variable* **;**
ID *variables* **;**
MODEL *response*< **(***response-options***)** > **=** < *effects* > < / *model-options* > **;**
MODEL *events/trials*< **(***response-options***)** > **=** < *effects* > < / *model-options* > **;**
OUTPUT < **OUT=***SAS-data-set* > < *keyword* < **=***name* > >. . . < *keyword* < **=***name* > > < / *options* > **;**
PARTITION < *partition-options* > **;**
PERFORMANCE *performance-options* **;**
SELECTION *selection-options* **;**
WEIGHT *variable* **;**

The PROC HPLOGISTIC statement and at least one MODEL statement is required. The CLASS statement can appear multiple times. If a CLASS statement is specified, it must precede the MODEL statements.

PROC HPLOGISTIC Statement

PROC HPLOGISTIC < *options* > **;**

The PROC HPLOGISTIC statement invokes the procedure. Table 9.1 summarizes the available options in the PROC HPLOGISTIC statement by function. The options are then described fully in alphabetical order.

Table 9.1 PROC HPLOGISTIC Statement Options

Option	Description
Basic Options	
ALPHA=	Specifies a global significance level
DATA=	Specifies the input data set
NAMELEN=	Limits the length of effect names
Options Related to Output	
ITDETAILS	Adds detail information to "Iteration History" table
ITSELECT	Displays the "Iteration History" table with model selection
NOPRINT	Suppresses ODS output
NOCLPRINT	Limits or suppresses the display of class levels
NOITPRINT	Suppresses generation of the iteration history table
NOSTDERR	Suppresses computation of covariance matrix and standard errors
Options Related to Optimization	
ABSCONV=	Tunes the absolute function convergence criterion
ABSFCONV=	Tunes the absolute function difference convergence criterion
ABSGCONV=	Tunes the absolute gradient convergence criterion
FCONV=	Tunes the relative function difference convergence criterion
GCONV=	Tunes the relative gradient convergence criterion
INEST=	Specifies the SAS data set that contains the starting values
MAXITER=	Chooses the maximum number of iterations in any optimization
MAXFUNC=	Specifies the maximum number of function evaluations in any optimization
MAXTIME=	Specifies the upper limit of CPU time (in seconds) for any optimization
MINITER=	Specifies the minimum number of iterations in any optimization
NORMALIZE=	Specifies whether the objective function is normalized during optimization
OUTEST	Adds parameter name to the "Parameter Estimates" table
TECHNIQUE=	Selects the optimization technique
Tolerances	
SINGCHOL=	Tunes the singularity criterion for Cholesky decompositions
SINGSWEEP=	Tunes the singularity criterion for the sweep operator
SINGULAR=	Tunes the general singularity criterion
User-Defined Formats	
FMTLIBXML=	Specifies the file reference for a format stream

You can specify the following options in the PROC HPLOGISTIC statement.

ABSCONV=*r*

ABSTOL=*r*

specifies an absolute function convergence criterion. For minimization, termination requires $f(\boldsymbol{\psi}^{(k)}) \leq r$, where $\boldsymbol{\psi}$ is the vector of parameters in the optimization and $f(\cdot)$ is the objective function. The default value of *r* is the negative square root of the largest double-precision value, which serves only as a protection against overflows.

ABSFCONV=*r* **<** *n* **>**

ABSFTOL=*r* **<** *n* **>**

specifies an absolute function difference convergence criterion. For all techniques except NMSIMP, termination requires a small change of the function value in successive iterations:

$$|f(\boldsymbol{\psi}^{(k-1)}) - f(\boldsymbol{\psi}^{(k)})| \leq r$$

Here, $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization, and $f(\cdot)$ is the objective function. The same formula is used for the NMSIMP technique, but $\boldsymbol{\psi}(k)$ is defined as the vertex with the lowest function value and $\boldsymbol{\psi}^{(k-1)}$ is defined as the vertex with the highest function value in the simplex. The default value is $r = 0$. The optional integer value *n* specifies the number of successive iterations for which the criterion must be satisfied before the process can be terminated.

ABSGCONV=*r* **<** *n* **>**

ABSGTOL=*r* **<** *n* **>**

specifies an absolute gradient convergence criterion. Termination requires the maximum absolute gradient element to be small:

$$\max_j |g_j(\boldsymbol{\psi}^{(k)})| \leq r$$

Here, $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization, and $g_j(\cdot)$ is the gradient of the objective function with respect to the *j*th parameter. This criterion is not used by the NMSIMP technique. The default value is *r*=1E–5. The optional integer value *n* specifies the number of successive iterations for which the criterion must be satisfied before the process can be terminated.

ALPHA=*number*

specifies a global significance level for the construction of confidence intervals. The confidence level is 1–*number*. The value of *number* must be between 0 and 1; the default is 0.05. You can override the global specification with the ALPHA= option in the MODEL statement.

DATA=*SAS-data-set*

names the input SAS data set for PROC HPLOGISTIC to use. The default is the most recently created data set.

If the procedure executes in distributed mode, the input data are distributed to memory on the appliance nodes and analyzed in parallel, unless the data are already distributed in the appliance database. In that case the procedure reads the data alongside the distributed database. For information about the various execution modes, see the section "Processing Modes" on page 10; for information about the alongside-the-database model, see the section "Alongside-the-Database Execution" on page 18.

FCONV=*r* **<** *n* **>**

FTOL=*r* **<** *n* **>**

specifies a relative function difference convergence criterion. For all techniques except NMSIMP, termination requires a small relative change of the function value in successive iterations,

$$\frac{|f(\boldsymbol{\psi}^{(k)}) - f(\boldsymbol{\psi}^{(k-1)})|}{|f(\boldsymbol{\psi}^{(k-1)})|} \leq r$$

Here, $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization, and $f(\cdot)$ is the objective function. The same formula is used for the NMSIMP technique, but $\boldsymbol{\psi}^{(k)}$ is defined as the vertex with the lowest function value, and $\boldsymbol{\psi}^{(k-1)}$ is defined as the vertex with the highest function value in the simplex.

The default value is r=$2 \times \epsilon$ where ϵ is the machine precision. The optional integer value *n* specifies the number of successive iterations for which the criterion must be satisfied before the process can terminate.

FMTLIBXML=*file-ref*

specifies the file reference for the XML stream that contains the user-defined format definitions. User-defined formats are handled differently in a distributed computing environment than they are in other SAS products. For more information about how to generate a XML stream for your formats, see the section "Working with Formats" on page 35.

GCONV=*r* **<** *n* **>**

GTOL=*r* **<** *n* **>**

specifies a relative gradient convergence criterion. For all techniques except CONGRA and NMSIMP, termination requires that the normalized predicted function reduction be small,

$$\frac{\mathbf{g}(\boldsymbol{\psi}^{(k)})'[\mathbf{H}^{(k)}]^{-1}\mathbf{g}(\boldsymbol{\psi}^{(k)})}{|f(\boldsymbol{\psi}^{(k)})|} \leq r$$

Here, $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization, $f(\cdot)$ is the objective function, and $\mathbf{g}(\cdot)$ is the gradient. For the CONGRA technique (where a reliable Hessian estimate $\mathbf{H}$ is not available), the following criterion is used:

$$\frac{\| \mathbf{g}(\boldsymbol{\psi}^{(k)}) \|_2^2 \quad \| \mathbf{s}(\boldsymbol{\psi}^{(k)}) \|_2}{\| \mathbf{g}(\boldsymbol{\psi}^{(k)}) - \mathbf{g}(\boldsymbol{\psi}^{(k-1)}) \|_2 \, |f(\boldsymbol{\psi}^{(k)})|} \leq r$$

This criterion is not used by the NMSIMP technique. The default value is r=1E–8. The optional integer value *n* specifies the number of successive iterations for which the criterion must be satisfied before the process can terminate.

INEST=*SAS-data-set*

names the TYPE=EST SAS data set that contains starting values for the parameters.

Your data set must include the _TYPE_ variable, a character variable in which the value 'PARMS' indicates the observation that contains your starting values. The data set also includes a numeric variable for each parameter for which you are specifying a starting value; the name of this numeric variable is the "parameter name." You can obtain parameter names by specifying the OUTEST option

and by using the ODS OUTPUT statement to output the "Parameter Estimates" table into a data set; the parameter name is contained in the ParmName variable in this data set. If you do not specify a starting value for a parameter, it is set to 0. PROC HPLOGISTIC uses only the first observation for which _TYPE_=PARMS, and it ignores BY variables. For more information about TYPE=EST data sets, see Chapter A, "Special SAS Data Sets" (*SAS/STAT User's Guide*),.

If you specify TECH=NONE or MAXITER=0, then the values in the INEST= data set are used as the parameter estimates, but the null model is still computed at the optimum value for the intercepts. If you specify TECH=NONE or MAXITER=0 and you specify a null model in the MODEL statement, then the null model is computed at the starting values for the intercept parameters.

ITDETAILS
adds to the "Iteration History" table the current values of the parameter estimates and their gradients. These quantities are reported only for parameters that participate in the optimization.

ITSELECT
generates the "Iteration History" table when you perform a model selection.

MAXFUNC=*n*

MAXFU=*n*
specifies the maximum number *n* of function calls in the optimization process. The default values are as follows, depending on the optimization technique:

- TRUREG, NRRIDG, NEWRAP: 125
- QUANEW, DBLDOG: 500
- CONGRA: 1,000
- NMSIMP: 3,000

The optimization can terminate only after completing a full iteration. Therefore, the number of function calls that are actually performed can exceed the number that is specified by the MAXFUNC= option. You can choose the optimization technique with the TECHNIQUE= option.

MAXITER=*n*

MAXIT=*n*
specifies the maximum number *n* of iterations in the optimization process. The default values are as follows, depending on the optimization technique:

- TRUREG, NRRIDG, NEWRAP: 50
- QUANEW, DBLDOG: 200
- CONGRA: 400
- NMSIMP: 1,000

These default values also apply when *n* is specified as a missing value. You can choose the optimization technique with the TECHNIQUE= option.

MAXTIME=*r*

specifies an upper limit of *r* seconds of CPU time for the optimization process. The default value is the largest floating-point double representation of your computer. The time specified by the MAXTIME= option is checked only once at the end of each iteration. Therefore, the actual running time can be longer than that specified by the MAXTIME= option.

MINITER=*n*

MINIT=*n*

specifies the minimum number of iterations. The default value is 0. If you request more iterations than are actually needed for convergence to a stationary point, the optimization algorithms can behave strangely. For example, the effect of rounding errors can prevent the algorithm from continuing for the required number of iterations.

NAMELEN=*number*

specifies the length to which long effect names are shortened. The default and minimum value is 20.

NOCLPRINT< =*number* **>**

suppresses the display of the "Class Level Information" table if you do not specify *number*. If you specify *number*, the values of the classification variables are displayed for only those variables whose number of levels is less than *number*. Specifying a *number* helps to reduce the size of the "Class Level Information" table if some classification variables have a large number of levels.

NOITPRINT

suppresses the generation of the "Iteration History" table.

NOPRINT

suppresses the generation of ODS output.

NORMALIZE=YES | NO

specifies whether the objective function should be normalized during the optimization by the reciprocal of the used frequency count. The default is to normalize the objective function. This option affects the values reported in the "Iteration History" table. The results reported in the "Fit Statistics" are always displayed for the nonnormalized log-likelihood function.

NOSTDERR

suppresses the computation of the covariance matrix and the standard errors of the logistic regression coefficients. When the model contains many variables (thousands), the inversion of the Hessian matrix to derive the covariance matrix and the standard errors of the regression coefficients can be time-consuming.

OUTEST

adds a column for the ParmName variable to the "Parameter Estimates" table. This column is not displayed, but you can use it to create a data set that you can specify in an INEST= option by first using the ODS OUTPUT statement to output the "Parameter Estimates" table and then submitting the following statements:

```
proc transpose data=parameterestimates out=inest(type=EST) label=_TYPE_;
   label Estimate=PARMS;
   var Estimate;
   id ParmName;
run;
```

SINGCHOL=*number*
tunes the singularity criterion in Cholesky decompositions. The default is 1E7 times the machine epsilon; this product is approximately 1E–9 on most computers.

SINGSWEEP=*number*
tunes the singularity criterion for sweep operations. The default is 1E7 times the machine epsilon; this product is approximately 1E–9 on most computers.

SINGULAR=*number*
tunes the general singularity criterion applied by the HPLOGISTIC procedure in sweeps and inversions. The default is 1E7 times the machine epsilon; this product is approximately 1E–9 on most computers.

TECHNIQUE=*keyword*
TECH=*keyword*
specifies the optimization technique for obtaining maximum likelihood estimates. You can choose from the following techniques by specifying the appropriate *keyword*:

CONGRA	performs a conjugate-gradient optimization.
DBLDOG	performs a version of double-dogleg optimization.
NEWRAP	performs a Newton-Raphson optimization with line search.
NMSIMP	performs a Nelder-Mead simplex optimization.
NONE	performs no optimization.
NRRIDG	performs a Newton-Raphson optimization with ridging.
QUANEW	performs a dual quasi-Newton optimization.
TRUREG	performs a trust-region optimization

The default value is TECHNIQUE=NRRIDG.

For more information, see the section "Choosing an Optimization Algorithm" on page 367.

BY Statement

BY *variables* **;**

You can specify a BY statement with PROC HPLOGISTIC to obtain separate analyses of observations in groups that are 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 you specify more than one BY statement, only the last one specified is used.

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 HPLOGISTIC 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 (in Base SAS software).

BY statement processing is not supported when the HPLOGISTIC procedure runs alongside the database or alongside the Hadoop Distributed File System (HDFS). These modes are used if the input data are stored in a database or HDFS and the grid host is the appliance that houses the data.

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

CLASS Statement

CLASS *variable* < **(***options***)** >. . . < *variable* < **(***options***)** > > < / *global-options* > **;**

The CLASS statement names the classification variables to be used as explanatory variables in the analysis. The CLASS statement must precede the MODEL statement. You can list the response variable for binary and multinomial models in the CLASS statement, but this is not necessary.

The CLASS statement for high-performance statistical procedures is documented in the section "CLASS Statement" on page 42 of Chapter 4, "Shared Statistical Concepts."

The HPLOGISTIC procedure does not support the SPLIT option in the CLASS statement. The HPLOGISTIC procedure additionally supports the following global-option in the CLASS statement:

UPCASE
uppercases the values of character-valued CLASS variables before levelizing them. For example, if the UPCASE option is in effect and a CLASS variable can take the values 'a', 'A', and 'b', then 'a' and 'A' represent the same level and the CLASS variable is treated as having only two values: 'A' and 'B'.

CODE Statement

CODE < *options* > **;**

The CODE statement writes SAS DATA step code for computing predicted values of the fitted model either to a file or to a catalog entry. This code can then be included in a DATA step to score new data.

Table 9.2 summarizes the *options* available in the CODE statement.

Table 9.2 CODE Statement Options

Option	Description
CATALOG=	Names the catalog entry where the generated code is saved
DUMMIES	Retains the dummy variables in the data set
ERROR	Computes the error function
FILE=	Names the file where the generated code is saved
FORMAT=	Specifies the numeric format for the regression coefficients
GROUP=	Specifies the group identifier for array names and statement labels
IMPUTE	Imputes predicted values for observations with missing or invalid covariates

Table 9.2 *continued*

Option	Description
LINESIZE=	Specifies the line size of the generated code
LOOKUP=	Specifies the algorithm for looking up CLASS levels
RESIDUAL	Computes residuals

For details about the syntax of the CODE statement, see the section "CODE Statement" (Chapter 19, *SAS/STAT User's Guide*) in Chapter 19, "Shared Concepts and Topics" (*SAS/STAT User's Guide*).

FREQ Statement

FREQ *variable* **;**

The *variable* in the FREQ statement identifies a numeric variable in the data set that contains the frequency of occurrence for each observation. High-performance statistical procedures that support the FREQ statement treat each observation as if it appeared f times, where the frequency value f is the value of the FREQ variable for the observation. If f is not an integer, then f is truncated to an integer. If f is less than 1 or missing, the observation is not used in the analysis. When the FREQ statement is not specified, each observation is assigned a frequency of 1.

ID Statement

ID *variables* **;**

The ID statement lists one or more variables from the input data set that are to be transferred to output data sets created by high-performance statistical procedures, provided that the output data set produces one (or more) records per input observation.

For documentation about the common ID statement in high-performance statistical procedures, see the section "ID Statement" on page 46 in Chapter 4, "Shared Statistical Concepts."

MODEL Statement

MODEL *response* < **(***response-options***)** > **=** < *effects* > < / *model-options* > **;**

MODEL *events* / *trials* < **(***response-options***)** > **=** < *effects* > < / *model-options* > **;**

The MODEL statement defines the statistical model in terms of a response variable (the target) or an *events/trials* specification, model effects constructed from variables in the input data set, and options. An intercept is included in the model by default. You can remove the intercept with the NOINT option.

You can specify a single *response* variable that contains your binary, ordinal, or nominal response values. When you have binomial data, you can specify the *events/trials* form of the response, where one variable contains the number of positive responses (or events) and another variable contains the number of trials.

Note that the values of both *events* and (*trials* – *events*) must be nonnegative and the value of *trials* must be positive.

For information about constructing the model effects, see the section "Specification and Parameterization of Model Effects" on page 55 of Chapter 4, "Shared Statistical Concepts."

There are two sets of options in the MODEL statement. The *response-options* determine how the HPLOGISTIC procedure models probabilities for binary data. The *model-options* control other aspects of model formation and inference. Table 9.3 summarizes these options.

Table 9.3 MODEL Statement Options

Option	Description
Response Variable Options	
DESCENDING	Reverses the response categories
EVENT=	Specifies the event category
ORDER=	Specifies the sort order
REF=	Specifies the reference category
Model Options	
ALPHA=	Specifies the confidence level for confidence limits
ASSOCIATION	Requests association statistics
CL	Requests confidence limits
CTABLE	Requests classification statistics
CUTPOINT=	Specifies a cutpoint for binary classification
DDFM=	Specifies the degrees-of-freedom method
INCLUDE=	Includes effects in all models for model selection
LACKFIT	Requests the Hosmer and Lemeshow goodness-of-fit test
LINK=	Specifies the link function
NOCHECK	Suppresses checking for infinite parameters
NOINT	Suppresses the intercept
OFFSET=	Specifies the offset variable
PRIOR=	Specifies prior probabilities
RSQUARE	Requests a generalized coefficient of determination
START=	Includes effects in the initial model for model selection

Response Variable Options

Response variable options determine how the HPLOGISTIC procedure models probabilities for binary and multinomial data.

You can specify the following *response-options* by enclosing them in parentheses after the *response* or *trials* variable.

DESCENDING

DESC

reverses the order of the response categories. If both the DESCENDING and ORDER= options are specified, PROC HPLOGISTIC orders the response categories according to the ORDER= option and then reverses that order.

EVENT='*category***' | FIRST | LAST**

specifies the event category for the binary response model. PROC HPLOGISTIC models the probability of the event category. The EVENT= option has no effect when there are more than two response categories.

You can specify the value (formatted, if a format is applied) of the event category in quotes, or you can specify one of the following:

FIRST

designates the first ordered category as the event. This is the default.

LAST

designates the last ordered category as the event.

For example, the following statements specify that observations with formatted value '1' represent events in the data. The probability modeled by the HPLOGISTIC procedure is thus the probability that the variable def takes on the (formatted) value '1'.

```
proc hplogistic data=MyData;
   class A B C;
   model def(event ='1') = A B C x1 x2 x3;
run;
```

ORDER=DATA | FORMATTED | INTERNAL

ORDER=FREQ | FREQDATA | FREQFORMATTED | FREQINTERNAL

specifies the sort order for the levels of the *response* variable. When ORDER=FORMATTED (the default) for numeric variables for which you have supplied no explicit format (that is, for which there is no corresponding FORMAT statement in the current PROC HPLOGISTIC run or in the DATA step that created the data set), the levels are ordered by their internal (numeric) value. The following table shows the interpretation of the ORDER= option:

ORDER=	Levels Sorted By
DATA	Order of appearance in the input data set
FORMATTED	External formatted value, except for numeric variables with no explicit format, which are sorted by their unformatted (internal) value
FREQ	Descending frequency count (levels with the most observations come first in the order)
FREQDATA	Order of descending frequency count; by order of appearance in the input data set when counts are tied
FREQFORMATTED	Order of descending frequency count; by formatted value (as above) when counts are tied
FREQINTERNAL	Order of descending frequency count; by unformatted value when counts are tied
INTERNAL	Unformatted value

By default, ORDER=FORMATTED. For the FORMATTED and INTERNAL orders, the sort order is machine-dependent.

For more information about sort order, see the chapter on the SORT procedure in the *Base SAS Procedures Guide* and the discussion of BY-group processing in *SAS Language Reference: Concepts*.

REF='*category***' | FIRST | LAST**
specifies the reference category for the generalized logit model and the binary response model. For the generalized logit model, each logit contrasts a nonreference category with the reference category. For the binary response model, specifying one response category as the reference is the same as specifying the other response category as the event category. You can specify the value (formatted if a format is applied) of the reference category in quotes, or you can specify one of the following:

FIRST
designates the first ordered category as the reference

LAST
designates the last ordered category as the reference. This is the default.

Model Options

ALPHA=*number*
requests that confidence intervals for each of the parameters be constructed with confidence level 1–*number*. The value of *number* must be between 0 and 1; the default is 0.05.

ASSOCIATION
displays measures of association between predicted probabilities and observed responses for binary or binomial response models. These measures assess the predictive ability of the model. The displayed statistics are the concordance index *C* (the area under the ROC curve, AUC), Somers' *D* statistic (Gini's coefficient), the Goodman-Kruskal gamma statistic, and Kendall's tau-a statistic. For more information, see the section "Association Statistics" on page 364.

CL
requests that confidence limits be constructed for each of the parameter estimates. The confidence level is 0.95 by default; this can be changed with the ALPHA= option.

CTABLE<=*SAS-data-set***>**
OUTROC<=*SAS-data-set***>**
displays a table for binary or binomial response models that contains the frequencies of observations that are correctly and incorrectly classified as events and nonevents, the sensitivity, the 1–specificity, the positive and negative predictive values, and the correct classification rate. For more information, see the section "Classification Table and ROC Curves" on page 363.

Classification is carried out by initially binning the predicted probabilities as discussed in the section "The Hosmer-Lemeshow Goodness-of-Fit Test" on page 366. The PRIOR= option does not change the reported predicted probabilities.

Because the number of cutpoints can be very large, you can store the table in an output data set. If you specify a PARTITION statement, then the statistics are computed by their roles, and a Role variable indicates to which partition the computations belong.

CUTPOINT=*value*

specifies a value between 0 and 1 used for classifying observations when you have a binary or binomial response variable. If the predicted probability of an observation equals or exceeds the cutpoint, the observation is classified as an event; otherwise it is classified as a nonevent. This option affects computation of the misclassification rate and the true positive and true negative fractions in the "Partition Fit Statistics" table. By default, CUTPOINT=0.5.

DDFM=RESIDUAL | NONE

specifies how degrees of freedom for statistical inference be determined in the "Parameter Estimates Table."

The HPLOGISTIC procedure always displays the statistical tests and confidence intervals in the "Parameter Estimates" tables in terms of a t test and a two-sided probability from a t distribution. With the DDFM= option, you can control the degrees of freedom of this t distribution and thereby switch between small-sample inference and large-sample inference based on the normal or chi-square distribution.

The default is DDFM=NONE, which leads to z-based statistical tests and confidence intervals. The HPLOGISTIC procedure then displays the degrees of freedom in the DF column as Infty, the p-values are identical to those from a Wald chi-square test, and the square of the t value equals the Wald chi-square statistic.

If you specify DDFM=RESIDUAL, the degrees of freedom are finite and determined by the number of usable frequencies (observations) minus the number of nonredundant model parameters. This leads to t-based statistical tests and confidence intervals. If the number of frequencies is large relative to the number of parameters, the inferences from the two degrees-of-freedom methods are almost identical.

INCLUDE=*n*

INCLUDE=*single-effect*

INCLUDE=(*effects***)**

forces effects to be included in all models. If you specify INCLUDE=*n*, then the first *n* effects that are listed in the MODEL statement are included in all models. If you specify INCLUDE=*single-effect* or if you specify a list of effects within parentheses, then the specified effects are forced into all models. The effects that you specify in the INCLUDE= option must be explanatory effects that are specified in the MODEL statement before the slash (/).

LACKFIT< (DFREDUCE=*r* **NGROUPS=***G***) >**

performs the Hosmer and Lemeshow goodness-of-fit test (Hosmer and Lemeshow 2000) for binary response models.

The subjects are divided into at most G groups of roughly the same size, based on the percentiles of the estimated probabilities. You can specify G as any integer greater than or equal to 5; by default, G=10. Let the actual number of groups created be g. The discrepancies between the observed and expected number of observations in these g groups are summarized by the Pearson chi-square statistic, which is then compared to a chi-square distribution with $g–r$ degrees of freedom. You can specify a nonnegative integer r that satisfies $g–r \geq 1$; by default, r=2.

A small p-value suggests that the fitted model is not an adequate model. For more information, see the section "The Hosmer-Lemeshow Goodness-of-Fit Test" on page 366.

LINK=*keyword*

specifies the link function for the model. The *keywords* and the associated link functions are shown in Table 9.4.

Table 9.4 Built-in Link Functions of the HPLOGISTIC Procedure

LINK=	Link Function	$g(\mu) = \eta =$
CLOGLOG | CLL	Complementary log-log	$\log(-\log(1-\mu))$
GLOGIT | GENLOGIT	Generalized logit	
LOGIT	Logit	$\log(\mu/(1-\mu))$
LOGLOG	Log-log	$-\log(-\log(\mu))$
PROBIT	Probit	$\Phi^{-1}(\mu)$

For the probit and cumulative probit links, $\Phi^{-1}(\cdot)$ denotes the quantile function of the standard normal distribution.

If the response variable has more than two categories, the HPLOGISTIC procedure fits a model with a cumulative link function based on the specified link. However, if you specify LINK=GLOGIT, the procedure assumes a generalized logit model for nominal (unordered) data, regardless of the number of response categories.

NOCHECK

disables the checking process that determines whether maximum likelihood estimates of the regression parameters exist. For more information, see the section "Existence of Maximum Likelihood Estimates" on page 359.

NOINT

requests that no intercept be included in the model. An intercept is included by default. The NOINT option is not available in multinomial models.

OFFSET=*variable*

specifies a *variable* to be used as an offset to the linear predictor. An offset plays the role of an effect whose coefficient is known to be 1. The offset variable cannot appear in the CLASS statement or elsewhere in the MODEL statement. Observations with missing values for the offset variable are excluded from the analysis.

PRIOR=*SAS-data-set*

PRIOR=*number*

PEVENT=*number*

PRIOR=ALLDATA

specifies prior probabilities (prevalences) that are used for computing posterior predicted probabilities. When you know what percentage of the population has a rare event and you oversample that rare event, specifying the prior probabilities as the prevalence of events in your population enables you to produce posterior probabilities that reflect the population, not the data.

You can specify your priors in a SAS data set in which a _PRIOR_ column contains the prior probabilities. For events/trials MODEL statement syntax, this data set should also include an _OUTCOME_ variable that contains the values EVENT and NONEVENT; for single-trial syntax, this data set should include the response variable that contains the unformatted response categories. Each row of the data set contains a unique response variable level and its prior. For binary and binomial response models, you can instead specify the probability of an event as *number*. If you also specify a PARTITION statement, you can specify PRIOR=ALLDATA to compute the prevalences as the observed proportions of the response levels gathered across all the roles.

If your response Y takes values $i = 1, \ldots, k$ that have observed empirical probabilities $\mathrm{OldPrior}_i = \frac{n_i}{n}$, you specify priors Prior_i, and your model predicted probabilities are $\hat{p}_i$, then the posterior predicted probabilities Post_i are computed as

$$\mathrm{Post}_i = \frac{\hat{p}_i \frac{\mathrm{Prior}_i}{\mathrm{OldPrior}_i}}{\sum_{j=1}^{k} \hat{p}_j \frac{\mathrm{Prior}_j}{\mathrm{OldPrior}_j}}$$

The POST= option in the OUTPUT statement writes the posterior to the output data set. If your priors are identical to the empirical probabilities, then the posteriors are identical to the model-predicted probabilities.

The priors do not affect the model-fitting process, but instead modify the following statistics in the "Partition Fit Statistics" table: false positive fraction, false negative fraction, false response fraction (for multinomial response models), and misclassification rate. The "Classification" table statistics PPV, NPV, and Percent Correct are also adjusted as described in the section "Classification Table and ROC Curves" on page 363.

RSQUARE

R2

requests a generalized coefficient of determination (R square, R^2) and a scaled version thereof for the fitted model. The results are added to the "Fit Statistics" table. For more information about the computation of these measures, see the section "Generalized Coefficient of Determination" on page 362.

START=*n*

START=*single-effect*

START=(*effects***)**

begins the selection process from the designated initial model for the FORWARD and STEPWISE selection methods. If you specify START=*n*, then the starting model includes the first *n* effects that are listed in the MODEL statement. If you specify START=*single-effect* or if you specify a list of effects within parentheses, then the starting model includes those specified effects. The effects that you specify in the START= option must be explanatory effects that are specified in the MODEL statement before the slash (/). The START= option is not available when you specify METHOD=BACKWARD in the SELECTION statement.

OUTPUT Statement

OUTPUT < **OUT=***SAS-data-set* >
< **COPYVARS=***(variables)* >
< *keyword* < *=name* > >. . . < *keyword* < *=name* > > < */ options* > **;**

The OUTPUT statement creates a data set that contains observationwise statistics that are computed after fitting the model. The variables in the input data set are *not* included in the output data set to avoid data duplication for large data sets; however, variables specified in the ID statement or COPYVAR= option are included.

If the input data are in distributed form, where access of data in a particular order cannot be guaranteed, the HPLOGISTIC procedure copies the distribution or partition key to the output data set so that its contents can be joined with the input data.

The output statistics are computed based on the final parameter estimates. If the optimization does not converge, then the output data set is not created.

When there are more than two response levels, values are computed only for variables that are named by the XBETA, POST, and PRED keywords; the other variables have missing values. These statistics are computed for every response category, and the automatic variable _LEVEL_ identifies the response category upon which the computed values are based. That is, every observation generates several rows in the output data set. If you also specify the OBSCAT option, then the observationwise statistics are computed only for the observed response category, as indicated by the value of the _LEVEL_ variable.

For observations in which only the response variable is missing, values of the XBETA, POST, and PRED statistics are computed even though these observations do not affect the model fit. This enables, for instance, predicted probabilities to be computed for new observations.

You can specify the following syntax elements in the OUTPUT statement before the slash (/).

OUT=*SAS-data-set*

DATA=*SAS-data-set*

specifies the name of the output data set. If the OUT= (or DATA=) option is omitted, the procedure uses the DATA*n* convention to name the output data set.

COPYVAR=*variable*

COPYVARS=(*variables***)**

transfers one or more *variables* from the input data set to the output data set. Variables named in an ID statement are also copied from the input data set to the output data set.

keyword **< =***name* **>**

specifies a statistic to include in the output data set and optionally names the variable *name*. If you do not provide a *name*, the HPLOGISTIC procedure assigns a default name based on the type of statistic requested.

The following are valid *keywords* for adding statistics to the OUTPUT data set:

LINP | XBETA

requests the linear predictor $\eta = \mathbf{x}'\boldsymbol{\beta}$. The default name is Xbeta.

PEARSON | PEARS | RESCHI
requests the Pearson residual, $\frac{\sqrt{wn}(y/n-\mu)}{\sqrt{\mu(1-\mu)}}$, where μ is the estimate of the predicted event probability, w is the weight of the observation, and n is the number of binomial trials (n=1 for binary observations). The default name is Pearson. This statistic is not computed for multinomial models.

POSTERIOR | POST
requests a numeric variable that contains the posterior predicted probability of each observation that is used in fitting the model. The default name is _POST_. If you do not specify the PRIOR option in the MODEL statement, then this value is the same as the predicted probability.

PREDICTED | PRED | P
requests predicted values (predicted probabilities of events) for the response variable. The default name is Pred.

RESIDUAL | RESID | R
requests the raw residual, $y - \mu$, where μ is the estimate of the predicted event probability. The default name is Residual. This statistic is not computed for multinomial models.

ROLE
requests a numeric variable that indicates the role played by each observation in fitting the model. The default name is _ROLE_. Table 9.5 shows how this variable is interpreted for each observation.

Table 9.5 Role Interpretation

Value	Observation Role
0	Not used
1	Training
2	Validation
3	Testing

If you do not partition the input data by specifying a PARTITION statement, then the role variable value is 1 for observations that are used in fitting the model and 0 for observations that have at least one missing or invalid value for the response, regressors, frequency or weight variables.

You can specify the following *options* in the OUTPUT statement after the slash (/):

OBSCAT
requests (for multinomial models) that observationwise statistics be produced for the response level only. If the OBSCAT option is not specified and the response variable has J levels, then the following outputs are created: for cumulative link models, $J - 1$ records are output for every observation in the input data that corresponds to the $J - 1$ lower-ordered response categories; for generalized logit models, J records are output that correspond to all J response categories.

PARTITION Statement

PARTITION < *partition-options* > **;**

The PARTITION statement specifies how observations in the input data set are logically partitioned into disjoint subsets for model training, validation, and testing. For more information, see the section "Using Validation and Test Data" on page 360. Either you can designate a variable in the input data set and a set of formatted values of that variable to determine the role of each observation, or you can specify proportions to use for random assignment of observations for each role.

The following mutually exclusive *partition-options* are available:

ROLEVAR | ROLE=*variable***(< TEST='***value***' > < TRAIN='***value***' > < VALIDATE='***value***' >)**
names the variable in the input data set whose values are used to assign roles to each observation. The formatted values of this variable that are used to assign observations roles are specified in the TEST=, TRAIN=, and VALIDATE= suboptions. If you do not specify the TRAIN= suboption, then all observations whose role is not determined by the TEST= or VALIDATE= suboptions are assigned to training.

FRACTION(< TEST=*fraction* **> < VALIDATE=***fraction* **> < SEED=***number* **>)**
randomly assigns specified proportions of the observations in the input data set to the roles. You specify the proportions for testing and validation by using the TEST= and VALIDATE= suboptions. If you specify both the TEST= and the VALIDATE= suboptions, then the sum of the specified fractions must be less than 1 and the remaining fraction of the observations are assigned to the training role. The SEED= option specifies an integer that is used to start the pseudorandom number generator for random partitioning of data for training, testing, and validation. If you do not specify a seed, or if you specify a value less than or equal to 0, the seed is generated from reading the time of day from the computer's clock.

PERFORMANCE Statement

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

The PERFORMANCE statement defines performance parameters for multithreaded and distributed computing, passes variables about the distributed computing environment, and requests detailed results about the performance characteristics of the HPLOGISTIC procedure.

With the PERFORMANCE statement you can also control whether the HPLOGISTIC procedure executes in single-machine mode or distributed mode.

The PERFORMANCE statement for high-performance statistical procedures is documented in the section "PERFORMANCE Statement" on page 36

SELECTION Statement

SELECTION < *options* > **;**

The SELECTION statement performs model selection by examining whether effects should be added to or removed from the model according to rules that are defined by model selection methods. The statement is fully documented in the section "SELECTION Statement" on page 47 in Chapter 4, "Shared Statistical Concepts."

The HPLOGISTIC procedure supports the following effect-selection methods in the SELECTION statement:

METHOD=NONE	results in no model selection. This method fits the full model.
METHOD=FORWARD	performs forward selection. This method starts with no effects in the model and adds effects.
METHOD=BACKWARD	performs backward elimination. This method starts with all effects in the model and deletes effects.
METHOD=BACKWARD(FAST)	performs fast backward elimination when SELECT=SL. This method starts with all effects in the model and deletes effects without refitting the model.
METHOD=STEPWISE	performs stepwise regression. This method is similar to the FORWARD method except that effects already in the model do not necessarily stay there.

The default criterion for the SELECT=, CHOOSE=, and STOP= options in the SELECTION statement is the significance level (SL), where effects enter and leave the model based on the significance level of an approximate chi-square test statistic. You can specify the following criteria in the SELECT=, CHOOSE=, and STOP= options:

AIC	uses Akaike's information criterion (Akaike 1974)
AICC	uses a small-sample bias corrected version of Akaike's information criterion, as promoted in Hurvich and Tsai (1989) and Burnham and Anderson (1998), for example
BIC | SBC	uses Schwarz' Bayesian criterion (Schwarz 1978)
SL	uses the significance level of the score test as the criterion (not available for a CHOOSE= option)
VALIDATE	uses the average square error (ASE) that is computed on the VALIDATE partition as the criterion (not available for a SELECT= option)

For more information, see the section "Information Criteria" on page 361. If you specify the PARTITION statement, then the AIC, AICC, BIC, and SL statistics are computed on the training data set; otherwise they are computed on the full data set.

NOTE: If you use the fast backward elimination method, the –2 log likelihood, AIC, AICC, and BIC statistics are approximated at each step where the model is not refit, and hence do not match the values that are computed when that model is fit outside of the selection routine. Similarly, if you specify SELECT=AIC, AICC, or BIC, the selection criteria are estimated (Lawless and Singhal 1978), and hence do not match the values that are computed when that model is fit outside of the selection routine.

When you specify the DETAILS= option in the SELECTION statement, the HPLOGISTIC procedure produces the following:

DETAILS=SUMMARY	produces a summary table that shows the effect that is added or removed at each step along with the *p*-value and the SELECT=, CHOOSE=, and STOP= criteria. The summary table is produced by default if the DETAILS= option is not specified.
DETAILS=STEPS	produces a detailed listing of all candidates at each step and their ranking in terms of the selection criterion for entry into or removal from the model.
DETAILS=ALL	produces the preceding two tables and a table of selection details, which displays fit statistics for the model at each step of the selection process and an approximate chi-square score or likelihood ratio statistic.

WEIGHT Statement

WEIGHT *variable* **;**

The *variable* in the WEIGHT statement is used as a weight to perform a weighted analysis of the data. Observations with nonpositive or missing weights are not included in the analysis. If a WEIGHT statement is not included, then all observations used in the analysis are assigned a weight of 1.

Details: HPLOGISTIC Procedure

Missing Values

Any observation with missing values for the response, frequency, weight, offset, or explanatory variables is excluded from the analysis; however, missing values are valid for response and explanatory variables that are specified with the MISSING option in the CLASS statement. Observations with a nonpositive weight or with a frequency less than 1 are also excluded.

The estimated linear predictor and the fitted probabilities are not computed for any observation that has missing offset or explanatory variable values. However, if only the response value is missing, the linear predictor and the fitted probabilities can be computed and output to a data set by using the OUTPUT statement.

Response Distributions

The response distribution is the probability distribution of the response (target) variable. The HPLOGISTIC procedure can fit data for the following distributions:

- binary distribution

- binomial distribution
- multinomial distribution

The expressions for the log-likelihood functions of these distributions are given in the next section.

The binary (or Bernoulli) distribution is the elementary distribution of a discrete random variable that can take on two values with probabilities p and $1 - p$. Suppose the random variable is denoted Y and

$$\Pr(Y = 1) = p$$
$$\Pr(Y = 0) = 1 - p$$

The value associated with probability p is often termed the *event* or "success"; the complementary event is termed the *non-event* or "failure." A Bernoulli experiment is a random draw from a binary distribution and generates events with probability p.

If $Y_1, \cdots, Y_n$ are n independent Bernoulli random variables, then their sum follows a binomial distribution. In other words, if $Y_i = 1$ denotes an event (success) in the ith Bernoulli trial, a binomial random variable is the number of events (successes) in n independent Bernoulli trials. If you use the events/trials syntax in the MODEL statement, the HPLOGISTIC procedure fits the model as if the data had arisen from a binomial distribution. For example, the following statements fit a binomial regression model with regressors x1 and x2. The variables e and t represent the events and trials for the binomial distribution:

```
proc hplogistic;
   model e/t = x1 x2;
run;
```

If the events/trials syntax is used, then both variables must be numeric and the value of the events variable cannot be less than 0 or exceed the value of the trials variable. A "Response Profile" table is not produced for binomial data.

The multinomial distribution is a generalization of the binary distribution and allows for more than two outcome categories. Because there are more than two possible outcomes for the multinomial distribution, the terminology of "successes," "failures," "events," and "non-events" no longer applies. With multinomial data, these outcomes are generically referred to as "categories" or levels.

Whenever the HPLOGISTIC procedure determines that the response variable has more than two levels (unless the events/trials syntax is used), the procedure fits the model as if the data had arisen from a multinomial distribution. By default, it is then assumed that the response categories are ordered and a cumulative link model is fit by applying the default or specified link function. If the response categories are unordered, then you should fit a generalized logit model by choosing LINK=GLOGIT in the MODEL statement.

Log-Likelihood Functions

The HPLOGISTIC procedure forms the log-likelihood functions of the various models as

$$L(\boldsymbol{\mu}; \mathbf{y}) = \sum_{i=1}^{n} f_i \, l(\mu_i; y_i, w_i)$$

where $l(\mu_i; y_i, w_i)$ is the log-likelihood contribution of the ith observation with weight w_i and f_i is the value of the frequency variable. For the determination of w_i and f_i, see the WEIGHT and FREQ statements. The individual log-likelihood contributions for the various distributions are as follows.

Binary Distribution

The HPLOGISTIC procedure computes the log-likelihood function $l(\mu_i(\boldsymbol{\beta}); y_i)$ for the ith binary observation as

$$\begin{aligned} \eta_i &= \mathbf{x}_i'\boldsymbol{\beta} \\ \mu_i(\boldsymbol{\beta}) &= g^{-1}(\eta_i) \\ l(\mu_i(\boldsymbol{\beta}); y_i) &= y_i \log\{\mu_i\} + (1 - y_i)\log\{1 - \mu_i\} \end{aligned}$$

Here, μ_i is the probability of an event, and the variable y_i takes on the value 1 for an event and the value 0 for a non-event. The inverse link function $g^{-1}(\cdot)$ maps from the scale of the linear predictor η_i to the scale of the mean. For example, for the logit link (the default),

$$\mu_i(\boldsymbol{\beta}) = \frac{\exp\{\eta_i\}}{1 + \exp\{\eta_i\}}$$

You can control which binary outcome in your data is modeled as the event with the *response-options* in the MODEL statement, and you can choose the link function with the LINK= option in the MODEL statement.

If a WEIGHT statement is given and w_i denotes the weight for the current observation, the log-likelihood function is computed as

$$l(\mu_i(\boldsymbol{\beta}); y_i, w_i) = w_i l(\mu_i(\boldsymbol{\beta}); y_i)$$

Binomial Distribution

The HPLOGISTIC procedure computes the log-likelihood function $l(\mu_i(\boldsymbol{\beta}); y_i)$ for the ith binomial observation as

$$\begin{aligned} \eta_i &= \mathbf{x}_i'\boldsymbol{\beta} \\ \mu_i(\boldsymbol{\beta}) &= g^{-1}(\eta_i) \\ l(\mu_i(\boldsymbol{\beta}); y_i, w_i) &= w_i \left(y_i \log\{\mu_i\} + (n_i - y_i)\log\{1 - \mu_i\}\right) \\ &\quad + w_i \left(\log\{\Gamma(n_i + 1)\} - \log\{\Gamma(y_i + 1)\} - \log\{\Gamma(n_i - y_i + 1)\}\right) \end{aligned}$$

where y_i and n_i are the values of the events and trials of the ith observation, respectively. μ_i measures the probability of events (successes) in the underlying Bernoulli distribution whose aggregate follows the binomial distribution.

Multinomial Distribution

The multinomial distribution modeled by the HPLOGISTIC procedure is a generalization of the binary distribution; it is the distribution of a single draw from a discrete distribution with J possible values. The log-likelihood function for the ith observation is thus deceptively simple:

$$l(\boldsymbol{\mu}_i; \mathbf{y}_i, w_i) = w_i \sum_{j=1}^{J} y_{ij} \log\{\mu_{ij}\}$$

In this expression, J denotes the number of response categories (the number of possible outcomes) and μ_{ij} is the probability that the ith observation takes on the response value associated with category j. The category probabilities must satisfy

$$\sum_{j=1}^{J} \mu_j = 1$$

and the constraint is satisfied by modeling $J - 1$ categories. In models that have ordered response categories, the probabilities are expressed in cumulative form, so that the last category is redundant. In generalized logit models (multinomial models that have unordered categories), one category is chosen as the reference category and the linear predictor in the reference category is set to zero. For more information, see the REF= *response-option* in the MODEL statement.

Existence of Maximum Likelihood Estimates

The likelihood equation for a logistic regression model does not always have a finite solution. Sometimes there is a nonunique maximum on the boundary of the parameter space, at infinity. The existence, finiteness, and uniqueness of maximum likelihood estimates for the logistic regression model depend on the patterns of data points in the observation space (Albert and Anderson 1984; Santner and Duffy 1986).

Consider a binary response model. Let Y_j be the response of the jth subject, and let $\mathbf{x}_j$ be the vector of explanatory variables (including the constant 1 that is associated with the intercept). There are three mutually exclusive and exhaustive types of data configurations: complete separation, quasi-complete separation, and overlap.

Complete Separation There is a complete separation of data points if there exists a vector $\mathbf{b}$ that correctly allocates all observations to their response groups; that is,

$$\begin{cases} \mathbf{b}'\mathbf{x}_j > 0 & Y_j = 1 \\ \mathbf{b}'\mathbf{x}_j < 0 & Y_j = 2 \end{cases}$$

This configuration produces nonunique infinite estimates. If the iterative process of maximizing the likelihood function is allowed to continue, the log likelihood diminishes to 0, and the dispersion matrix becomes unbounded.

Quasi-complete Separation The data are not completely separable, but there is a vector $\mathbf{b}$ such that

$$\begin{cases} \mathbf{b}'\mathbf{x}_j \geq 0 & Y_j = 1 \\ \mathbf{b}'\mathbf{x}_j \leq 0 & Y_j = 2 \end{cases}$$

and equality holds for at least one subject in each response group. This configuration also yields nonunique infinite estimates. If the iterative process of maximizing the likelihood function is allowed to continue, the dispersion matrix becomes unbounded and the log likelihood diminishes to a nonzero constant.

Overlap If neither complete nor quasi-complete separation exists in the sample points, there is an overlap of sample points. In this configuration, the maximum likelihood estimates exist and are unique.

The HPLOGISTIC procedure uses a simple empirical approach to recognize the data configurations that lead to infinite parameter estimates. The basis of this approach is that any convergence method of maximizing

the log likelihood must yield a solution that indicates complete separation, if such a solution exists. Upon convergence, if the predicted response equals the observed response for every observation, there is a complete separation of data points.

If the data are not completely separated, if an observation is identified to have an extremely large probability (≥ 0.95) of predicting the observed response, and if there have been at least eight iterations, then there are two possible situations. First, there is overlap in the data set, the observation is an atypical observation of its own group, and the iterative process stopped when a maximum was reached. Second, there is quasi-complete separation in the data set, and the asymptotic dispersion matrix is unbounded. If any of the diagonal elements of the dispersion matrix for the standardized observation vector (all explanatory variables standardized to zero mean and unit variance) exceeds 5,000, quasi-complete separation is declared. If either complete separation or quasi-complete separation is detected, a note is displayed in the procedure output.

Checking for quasi-complete separation is less foolproof than checking for complete separation. If neither type of separation is discovered and your parameter estimates have large standard errors, then this indicates that your data might be separable. The NOCHECK option in the MODEL statement turns off the process of checking for infinite parameter estimates.

Using Validation and Test Data

When you have sufficient data, you can subdivide your data into three parts called the training, validation, and test data. During the selection process, models are fit on the training data, and the prediction errors for the models so obtained are found by using the validation data. This prediction error on the validation data can be used to decide when to terminate the selection process and to decide which model. Finally, after a selected model has been obtained, the test set can be used to assess how the selected model generalizes on data that played no role in selecting the model.

In some cases you might want to use only training and test data. For example, you might decide to use an information criterion to decide which effects to include and when to terminate the selection process. In this case no validation data are required, but test data can still be useful in assessing the predictive performance of the selected model. In other cases you might decide to use validation data during the selection process but forgo assessing the selected model on test data. Hastie, Tibshirani, and Friedman (2001) note that it is difficult to provide a general rule for how many observations you should assign to each role. They note that a typical split might be 50% for training and 25% each for validation and testing.

You use a PARTITION statement to logically subdivide the DATA= data set into separate roles. You can specify the fractions of the data that you want to reserve as test data and validation data. For example, the following statements randomly subdivide the inData data set, reserving 50% for training and 25% each for validation and testing:

```
proc hplogistic data=inData;
  partition fraction(test=0.25 validate=0.25);
  ...
run;
```

In some cases you might need to exercise more control over the partitioning of the input data set. You can do this by naming both a variable in the input data set and a formatted value of that variable for each role. For example, the following statements assign roles to the observations in the inData data set that are based

on the value of the variable Group in that data set. Observations whose value of Group is `'Group 1'` are assigned for testing, and those whose value is `'Group 2'` are assigned to training. All other observations are ignored.

```
proc hplogistic data=inData;
  partition roleVar=Group(test='Group 1' train='Group 2')
  ...
run;
```

When you have reserved observations for training, validation, and testing, a model that is fit on the training data is scored on the validation and test data, and the average squared error (ASE) is computed separately for each of these subsets. The ASE for each data role is the sum of the squared differences between the responses and the predictions for observations in that role divided by the number of observations in that role.

Using the Validation Statistic as the CHOOSE= Criterion

When you specify the CHOOSE=VALIDATE suboption of the METHOD= option in the SELECTION statement, the ASE is computed on the validation data for the models at each step of the selection process. The smallest model at any step that yields the smallest validation ASE is selected.

Using the Validation Statistic as the STOP= Criterion

When you specify the STOP=VALIDATE suboption of the METHOD= option in the SELECTION statement, the ASE is computed on the validation data for the models at each step of the selection process. At step k of the selection process, the best candidate effect to enter or leave the current model is determined and the validation ASE for this new model is computed. If this validation ASE is greater than the validation ASE for the model at step k, then the selection process terminates at step k.

Partition Fit Statistics

Specifying a PARTITION statement modifies the display of many tables by adding separate rows or columns for the training, validation, and test data sets. A "Partition Fit Statistics" table is also produced, and it displays the following statistics, which are useful for assessing the model: average square error, misclassification rate, R^2, max-rescaled R^2, and McFadden's R^2. Binary and binomial response models also display the following statistics: area under the ROC curve, the Hosmer-Lemeshow test p-value, mean difference, Somers' D statistic, and the true positive and negative fractions. Polytomous response models also display the true fraction for each response level. For more information, see the sections "Classification Table and ROC Curves" on page 363 and "The Hosmer-Lemeshow Goodness-of-Fit Test" on page 366.

Model Fit and Assessment Statistics

Information Criteria

The calculation of the information criteria uses the following formulas, where p denotes the number of effective parameters in the candidate model, F denotes the sum of frequencies used, and l is the log likelihood

evaluated at the converged estimates:

$$\begin{aligned}
\text{AIC} &= -2l + 2p \\
\text{AICC} &= \begin{cases} -2l + 2pF/(F-p-1) & \text{when } F > p+2 \\ -2l + 2p(p+2) & \text{otherwise} \end{cases} \\
\text{BIC} &= -2l + p\log(F)
\end{aligned}$$

If you do not specify a FREQ statement, F equals n, the number of observations used.

Generalized Coefficient of Determination

The goal of a coefficient of determination, also known as an R-square measure, is to express the agreement between a stipulated model and the data in terms of variation in the data that is explained by the model. In linear models, the R-square measure is based on residual sums of squares; because these are additive, a measure bounded between 0 and 1 is easily derived.

In more general models where parameters are estimated by the maximum likelihood principle, Cox and Snell (1989, pp. 208–209) and Magee (1990) proposed the following generalization of the coefficient of determination:

$$R^2 = 1 - \left\{ \frac{L(\mathbf{0})}{L(\widehat{\boldsymbol{\beta}})} \right\}^{\frac{2}{n}}$$

Here, $L(\mathbf{0})$ is the likelihood of the intercept-only model, $L(\widehat{\boldsymbol{\beta}})$ is the likelihood of the specified model, and n denotes the number of observations used in the analysis. This number is adjusted for frequencies if a FREQ statement is present and is based on the trials variable for binomial models.

As discussed in Nagelkerke (1991), this generalized R-square measure has properties similar to the coefficient of determination in linear models. If the model effects do not contribute to the analysis, $L(\widehat{\boldsymbol{\beta}})$ approaches $L(\mathbf{0})$ and R^2 approaches zero.

However, R^2 does not have an upper limit of 1. Nagelkerke suggested a rescaled generalized coefficient of determination, $\tilde{R}^2$, which achieves an upper limit of 1 by dividing R^2 by its maximum value:

$$\begin{aligned}
R^2_{\max} &= 1 - \{L(\mathbf{0})\}^{\frac{2}{n}} \\
\tilde{R}^2 &= \frac{R^2}{R^2_{\max}}
\end{aligned}$$

Another measure from McFadden (1974) is also bounded by 0 and 1:

$$R^2_M = 1 - \left(\frac{\log L(\widehat{\boldsymbol{\beta}})}{\log L(\mathbf{0})} \right)$$

If you specify the RSQUARE option in the MODEL statement, the HPLOGISTIC procedure computes R^2 and $\tilde{R}^2$. All three measures are computed for each data role when you specify a PARTITION statement.

These measures are most useful for comparing competing models that are not necessarily nested—that is, models that cannot be reduced to one another by simple constraints on the parameter space. Larger values of the measures indicate better models.

Classification Table and ROC Curves

For binary response data, the response Y is either an event or a nonevent; let the response Y take the value 1 for an event and 2 for a nonevent. From the fitted model, a predicted event probability $\hat{\pi}_i$ can be computed for each observation i. If the predicted event probability equals or exceeds some cutpoint value $z \in [0, 1]$, the observation is classified as an event; otherwise, it is classified as a nonevent. Suppose n_1 of n individuals experience an event, such as a disease, and the remaining $n_2 = n - n_1$ individuals are nonevents. The 2×2 decision matrix in Table 9.6 is obtained by cross-classifying the observed and predicted responses, where n_{ij} is the total number of observations that are observed to have Y=i and are classified into j. In this table, let $Y = 1$ denote an observed event and $Y = 2$ denote a nonevent, and let $D = 1$ indicate that the observation is classified as an event and $D = 2$ denote that the observation is classified as a nonevent.

Table 9.6 Decision Matrix

	$D = 1\ (\hat{\pi} \geq z)$	$D = 2\ (\hat{\pi} < z)$	**Total**
$Y = 1$ (event)	n_{11}	n_{12}	n_1
$Y = 2$ (nonevent)	n_{21}	n_{22}	n_2

In the decision matrix, the *number of true positives*, n_{11}, is the number of event observations that are correctly classified as events; the *number of false positives*, n_{21}, is the number of nonevent observations that are incorrectly classified as events; the *number of false negatives*, n_{12}, is the number of event observations that are incorrectly classified as nonevents; and the *number of true negatives*, n_{22}, is the number of nonevent observations that are correctly classified as nonevents. The following statistics are computed from the preceding decision matrix:

Table 9.7 Statistics from the Decision Matrix with Cutpoint z

Statistic	**Equation**	**OUTROC Column**
Cutpoint	z	ProbLevel
Number of true positives	n_{11}	TruePos
Number of true negatives	n_{22}	TrueNeg
Number of false positives	n_{21}	FalsePos
Number of false negatives	n_{12}	FalseNeg
Sensitivity	n_{11}/n_1	TPF (true positive fraction)
1–specificity	n_{21}/n_2	FPF (false positive fraction)
Correct classification rate	$(n_{11} + n_{22})/n$	PercentCorrect (PC)
Misclassification rate	1 – PC	
Positive predictive value	$n_{11}/(n_{11} + n_{21})$	PPV
Negative predictive value	$n_{22}/(n_{12} + n_{22})$	NPV

The accuracy of the classification is measured by its ability to predict events and nonevents correctly. *Sensitivity* (TPF, true positive fraction) is the proportion of event responses that are predicted to be events. *Specificity* (1–FPF, true negative fraction) is the proportion of nonevent responses that are predicted to be nonevents.

You can also measure accuracy by how well the classification predicts the response. The *positive predictive value* (PPV, 1–false positive rate) is the proportion of nonevents that are classified as events. The *negative*

predictive value (NPV, 1–false negative rate) is the proportion of events that are classified as nonevents. The *correct classification rate* (PC) is the proportion of observations that are correctly classified.

If you also specify a PRIOR= option, then PROC HPLOGISTIC uses Bayes' theorem to modify the PPV, NPV, and PC as follows. Results of the classification are represented by two conditional probabilities: sensitivity, $\Pr(D=1|Y=1) = \frac{n_{11}}{n_1}$, and one minus the specificity, $\Pr(D=1|Y=2) = \frac{n_{21}}{n_2}$.

If the prevalence of the disease in the population $\Pr(D = 1)$ is provided by the value of the PRIOR= option, then the PPV, NPV, and PC are given by Fleiss (1981, pp. 4–5) as follows:

$$\begin{aligned}
\text{PPV} &= \Pr(Y=1|D=1) = \frac{\Pr(Y=1)\Pr(D=1|Y=1)}{\Pr(D=1|Y=2) + \Pr(Y=1)[\Pr(D=1|Y=1) - \Pr(D=1|Y=2)]} \\
\text{NPV} &= \Pr(Y=2|D=2) = \frac{[1-\Pr(D=1|Y=2)][1-\Pr(Y=1)]}{1-\Pr(D=1|Y=2) - \Pr(Y=1)[\Pr(D=1|Y=1) - \Pr(D=1|Y=2)]} \\
\text{PC} &= \Pr(Y=1|D=1) + \Pr(Y=2|D=2) \\
&= \Pr(D=1|Y=1)\Pr(Y=1) + \Pr(D=2|Y=2)[1-\Pr(Y=1)]
\end{aligned}$$

If you do not specify the PRIOR= option, then PROC HPLOGISTIC uses the sample proportion of diseased individuals; that is, $\Pr(Y = 1) = n_1/n$. In such a case, the preceding values reduce to those in Table 9.7. Note that for a stratified sampling situation in which n_1 and n_2 are chosen a priori, n_1/n is not a desirable estimate of $\Pr(Y=1)$, so you should specify a PRIOR= option.

PROC HPLOGISTIC constructs the data for a receiver operating characteristic (ROC) curve by initially binning the predicted probabilities as discussed in the section "The Hosmer-Lemeshow Goodness-of-Fit Test" on page 366, then moving the cutpoint from 0 to 1 along the bin boundaries (so that the cutpoints correspond to the predicted probabilities), and then selecting those cutpoints where a change in the decision matrix occurs. The CTABLE option produces a table that includes these cutpoints and the statistics in Table 9.7 that correspond to each cutpoint. You can output this table to a SAS data set by specifying the CTABLE= option (see Table 9.7 for the column names), and you can display the ROC curve by using the SGPLOT procedure as shown in Example 9.2.

The area under the ROC curve (AUC), as determined by the trapezoidal rule, is given by the concordance index C, which is described in the section "Association Statistics" on page 364.

For more information about the topics in this section, see Pepe (2003).

The "Partition Fit Statistics" table displays the misclassification rate, true positive fraction, true negative fraction, and AUC according to their roles. If you have a polytomous response, then instead of classifying according to a cutpoint, PROC HPLOGISTIC classifies the observation into the lowest response level (which has the largest predicted probability for that observation) and similarly computes a true response-level fraction.

Association Statistics

If you specify the ASSOCIATION option in the MODEL statement, PROC HPLOGISTIC displays measures of association between predicted probabilities and observed responses for binary or binomial response models. These measures assess the predictive ability of a model.

Of the n pairs of observations in the data set with different responses, let n_c be the number of pairs where the observation that has the lower ordered response value has a lower predicted probability, let n_d be the number of pairs where the observation that has the lower ordered response value has a higher predicted probability, and let $n_t = n - n_c - n_d$ be the rest. Let N be the sum of observation frequencies in the data. Then the following statistics are reported:

concordance index C (AUC)	=	$(n_c + 0.5n_t)/n$
Somers' D (Gini coefficient)	=	$(n_c - n_d)/n$
Goodman-Kruskal gamma	=	$(n_c - n_d)/(n_c + n_d)$
Kendall's tau-a	=	$(n_c - n_d)/(0.5N(N-1))$

Classification of the pairs is carried out by initially binning the predicted probabilities as discussed in the section "The Hosmer-Lemeshow Goodness-of-Fit Test" on page 366. The concordance index, C, is an estimate of the AUC, which is the area under the receiver operating characteristic (ROC) curve. If there are no ties, then Somers' D (Gini's coefficient) = $2C$–1.

If you specify a PARTITION statement, then PROC HPLOGISTIC displays the AUC and Somers' D in the "Association" and "Partition Fit Statistics" tables according to their roles.

Average Squared Error

The average squared error, ASE, is the average of the squared differences between the responses and the predictions. When you have a discrete number of response levels, the ASE is modified as shown in Table 9.8 (Brier 1950; Murphy 1973); it is also called the Brier score or Brier reliability:

Table 9.8 Average Square Error Computations

Response Type	ASE (Brier Score)
Polytomous	$\frac{1}{F}\sum_i f_i \sum_j (y_{ij} - \widehat{\pi}_{ij})^2$
Binary	$\frac{1}{F}\sum_i f_i (y_i(1-\widehat{\pi}_i)^2 + (1-y_i)\widehat{\pi}_i^2)$
Binomial	$\frac{1}{F}\sum_i f_i (r_i/t_i - \widehat{\pi}_i)^2$

In Table 9.8, $F = \sum_i f_i$, r_i is the number of events, t_i is the number of trials in binomial response models, and y_i=1 for events and 0 for nonevents in binary response models. For polytomous response models, y_{ij}=1 if the ith observation has response level j, and π_{ij} is the model-predicted probability of response level j for observation i.

Mean Difference

For a binary response model, write the mean of the model-predicted probabilities of event (Y=1) observations as $\overline{X}_1 = \frac{\sum_{i=1}^{n}(\pi_i | y_i=1)}{n_1}$ and of the nonevent (Y=2) observations as $\overline{X}_2 = \frac{\sum_{i=1}^{n}(\pi_i | y_i=2)}{n_1}$. The mean difference is $\overline{X}_1 - \overline{X}_2$, which Tjur (2009) relates to other R-square measures and calls the *coefficient of discrimination* because it is a measure of the model's ability to distinguish between the event and nonevent distributions. This mean difference is also the d' or Δm statistic (with unit standard error) that is discussed in signal detection literature (McNicol 2005).

The Hosmer-Lemeshow Goodness-of-Fit Test

To evaluate the fit of the model, Hosmer and Lemeshow (2000) proposed a statistic that they show, through simulation, is distributed as chi-square when there is no replication in any of the subpopulations. This goodness-of-fit test is available only for binary response models.

The unit interval is partitioned into 2,000 equal-sized bins, and each observation i is placed into the bin that contains its estimated event probability. This effectively sorts the observations in increasing order of their estimated event probability.

The observations (and frequencies) are further combined into G groups. By default $G = 10$, but you can specify $G \geq 5$ with the NGROUPS= suboption of the LACKFIT option in the MODEL statement. Let F be the total frequency. The target frequency for each group is $T = \lfloor F/G + 0.5 \rfloor$, which is the integer part of $F/G + 0.5$. Load the first group (g_j, $j = 1$) with the first of the 2,000 bins that has nonzero frequency f_1, and let the next nonzero bin have a frequency of f. PROC HPLOGISTIC performs the following steps for each nonzero bin to create the groups:

1. If $j = G$, then add this bin to group g_j.
2. Otherwise, if $f_j < T$ and $f_j + \lfloor f/2 \rfloor \leq T$, then add this bin to group g_j.
3. Otherwise, start loading the next group (g_{j+1}) with $f_{j+1} = f$, and set $j = j + 1$.

If the final group g_j has frequency $f_j < T/2$, then add these observations to the preceding group. The total number of groups actually created, g, can be less than G.

The Hosmer-Lemeshow goodness-of-fit statistic is obtained by calculating the Pearson chi-square statistic from the $2 \times g$ table of observed and expected frequencies. The statistic is written

$$\chi^2_{HL} = \sum_{j=1}^{g} \frac{(O_j - F_j \bar{\pi}_j)^2}{F_j \bar{\pi}_j (1 - \bar{\pi}_j)}$$

where, for the jth group g_j, $F_j = \sum_{i \in g_j} f_i$ is the total frequency of subjects, O_j is the total frequency of event outcomes, and $\bar{\pi}_j = \sum_{i \in g_j} f_i \widehat{p}_i / F_j$ is the average estimated predicted probability of an event outcome. Let ϵ be the square root of the machine epsilon divided by 4,000, which is about 2.5E–12. Any $\bar{\pi}_j < \epsilon$ is set to ϵ; similarly, any $\bar{\pi}_j > 1 - \epsilon$ is set to $1 - \epsilon$.

The Hosmer-Lemeshow statistic is compared to a chi-square distribution with $g - r$ degrees of freedom. You can specify r with the DFREDUCE= suboption of the LACKFIT option in the MODEL statement. By default, $r = 2$, and to compute the Hosmer-Lemeshow statistic you must have $g - r \geq 1$. Large values of χ^2_{HL} (and small p-values) indicate a lack of fit of the model.

Computational Method: Multithreading

Threading refers to the organization of computational work into multiple tasks (processing units that can be scheduled by the operating system). A task is associated with a thread. Multithreading refers to the concurrent execution of threads. When multithreading is possible, substantial performance gains can be realized compared to sequential (single-threaded) execution.

The number of threads spawned by the HPLOGISTIC procedure is determined by the number of CPUs on a machine and can be controlled by specifying the NTHREADS= option in the PERFORMANCE statement. This specification overrides the system option. Specify NTHREADS=1 to force single-threaded execution. The number of threads per machine is displayed in the "Dimensions" table, which is part of the default output. The HPLOGISTIC procedure allocates one thread per CPU by default.

The tasks that are multithreaded by the HPLOGISTIC procedure are primarily defined by dividing the data processed on a single machine among the threads—that is, the HPLOGISTIC procedure implements multithreading through a data-parallel model. For example, if the input data set has 1,000 observations and you are running with four threads, then 250 observations are associated with each thread. All operations that require access to the data are then multithreaded. These operations include the following:

- variable levelization
- effect levelization
- formation of the initial crossproducts matrix
- formation of approximate Hessian matrices for candidate evaluation during model selection
- objective function calculation
- gradient calculation
- Hessian calculation
- scoring of observations
- summarization of data for the Hosmer-Lemeshow test and association statistics

In addition, operations on matrices such as sweeps can be multithreaded provided that the matrices are of sufficient size to realize performance benefits from managing multiple threads for the particular matrix operation.

Choosing an Optimization Algorithm

First- or Second-Order Algorithms

The factors that go into choosing a particular optimization technique for a particular problem are complex. Trial and error can be involved.

For many optimization problems, computing the gradient takes more computer time than computing the function value. Computing the Hessian sometimes takes *much* more computer time and memory than computing the gradient, especially when there are many decision variables. Unfortunately, optimization techniques that do not use some kind of Hessian approximation usually require many more iterations than techniques that do use a Hessian matrix, and, as a result the total run time of these techniques is often longer. Techniques that do not use the Hessian also tend to be less reliable. For example, they can terminate more easily at stationary points than at global optima.

Table 9.9 shows which derivatives are required for each optimization technique.

Table 9.9 Derivatives Required

Algorithm	First-Order	Second-Order
TRUREG	x	x
NEWRAP	x	x
NRRIDG	x	x
QUANEW	x	-
DBLDOG	x	-
CONGRA	x	-
NMSIMP	-	-

The second-derivative methods TRUREG, NEWRAP, and NRRIDG are best for small problems for which the Hessian matrix is not expensive to compute. Sometimes the NRRIDG algorithm can be faster than the TRUREG algorithm, but TRUREG can be more stable. The NRRIDG algorithm requires only one matrix with $p(p+1)/2$ double words; TRUREG and NEWRAP require two such matrices. Here, p denotes the number of parameters in the optimization.

The first-derivative methods QUANEW and DBLDOG are best for medium-sized problems for which the objective function and the gradient can be evaluated much faster than the Hessian. In general, the QUANEW and DBLDOG algorithms require more iterations than TRUREG, NRRIDG, and NEWRAP, but each iteration can be much faster. The QUANEW and DBLDOG algorithms require only the gradient to update an approximate Hessian, and they require slightly less memory than TRUREG or NEWRAP.

The first-derivative method CONGRA is best for large problems for which the objective function and the gradient can be computed much faster than the Hessian and for which too much memory is required to store the (approximate) Hessian. In general, the CONGRA algorithm requires more iterations than QUANEW or DBLDOG, but each iteration can be much faster. Because CONGRA requires only a factor of p double-word memory, many large applications can be solved only by CONGRA.

The no-derivative method NMSIMP is best for small problems for which derivatives are not continuous or are very difficult to compute.

Each optimization method uses one or more convergence criteria that determine when it has converged. An algorithm is considered to have converged when any one of the convergence criteria is satisfied. For example, under the default settings, the QUANEW algorithm converges if ABSGCONV $<$ 1E–5, FCONV $< 2 \times \epsilon$, or GCONV $<$ 1E–8.

By default, the HPLOGISTIC procedure applies the NRRIDG algorithm because it can take advantage of multithreading in Hessian computations and inversions. If the number of parameters becomes large, specifying the TECHNIQUE=QUANEW option, which is a first-order method with good overall properties, is recommended.

Algorithm Descriptions

The following subsections provide details about each optimization technique and follow the same order as Table 9.9.

Trust Region Optimization (TRUREG)

The trust region method uses the gradient $\mathbf{g}(\boldsymbol{\psi}^{(k)})$ and the Hessian matrix $\mathbf{H}(\boldsymbol{\psi}^{(k)})$; thus, it requires that the objective function $f(\boldsymbol{\psi})$ have continuous first- and second-order derivatives inside the feasible region.

The trust region method iteratively optimizes a quadratic approximation to the nonlinear objective function within a hyperelliptic trust region with radius Δ that constrains the step size that corresponds to the quality of the quadratic approximation. The trust region method is implemented based on Dennis, Gay, and Welsch (1981), Gay (1983), and Moré and Sorensen (1983).

The trust region method performs well for small- to medium-sized problems, and it does not need many function, gradient, and Hessian calls. However, if the computation of the Hessian matrix is computationally expensive, one of the dual quasi-Newton or conjugate gradient algorithms might be more efficient.

Newton-Raphson Optimization with Line Search (NEWRAP)

The NEWRAP technique uses the gradient $\mathbf{g}(\boldsymbol{\psi}^{(k)})$ and the Hessian matrix $\mathbf{H}(\boldsymbol{\psi}^{(k)})$; thus, it requires that the objective function have continuous first- and second-order derivatives inside the feasible region. If second-order derivatives are computed efficiently and precisely, the NEWRAP method can perform well for medium-sized to large problems, and it does not need many function, gradient, and Hessian calls.

This algorithm uses a pure Newton step when the Hessian is positive-definite and when the Newton step reduces the value of the objective function successfully. Otherwise, a combination of ridging and line search is performed to compute successful steps. If the Hessian is not positive-definite, a multiple of the identity matrix is added to the Hessian matrix to make it positive-definite (Eskow and Schnabel 1991).

In each iteration, a line search is performed along the search direction to find an approximate optimum of the objective function. The line-search method uses quadratic interpolation and cubic extrapolation.

Newton-Raphson Ridge Optimization (NRRIDG)

The NRRIDG technique uses the gradient $\mathbf{g}(\boldsymbol{\psi}^{(k)})$ and the Hessian matrix $\mathbf{H}(\boldsymbol{\psi}^{(k)})$; thus, it requires that the objective function have continuous first- and second-order derivatives inside the feasible region.

This algorithm uses a pure Newton step when the Hessian is positive-definite and when the Newton step reduces the value of the objective function successfully. If at least one of these two conditions is not satisfied, a multiple of the identity matrix is added to the Hessian matrix.

Because the NRRIDG technique uses an orthogonal decomposition of the approximate Hessian, each iteration of NRRIDG can be slower than that of the NEWRAP technique, which works with a Cholesky decomposition. However, NRRIDG usually requires fewer iterations than NEWRAP.

The NRRIDG method performs well for small- to medium-sized problems, and it does not require many function, gradient, and Hessian calls. However, if the computation of the Hessian matrix is computationally expensive, one of the dual quasi-Newton or conjugate gradient algorithms might be more efficient.

Quasi-Newton Optimization (QUANEW)

The dual quasi-Newton method uses the gradient $\mathbf{g}(\boldsymbol{\psi}^{(k)})$, and it does not need to compute second-order derivatives because they are approximated. It works well for medium-sized to moderately large optimization problems, where the objective function and the gradient can be computed much faster than the Hessian. However, in general the QUANEW technique requires more iterations than the TRUREG, NEWRAP, and NRRIDG techniques, which compute second-order derivatives. The QUANEW technique provides an appropriate balance between the speed and stability required for most nonlinear mixed model applications.

The QUANEW technique implemented by the HPLOGISTIC procedure is the dual quasi-Newton algorithm, which updates the Cholesky factor of an approximate Hessian.

In each iteration, a line search is performed along the search direction to find an approximate optimum. The line-search method uses quadratic interpolation and cubic extrapolation to obtain a step size α that satisfies the Goldstein conditions (Fletcher 1987). One of the Goldstein conditions can be violated if the feasible region defines an upper limit of the step size. Violating the left-side Goldstein condition can affect the positive-definiteness of the quasi-Newton update. In that case, either the update is skipped or the iterations are restarted with an identity matrix, resulting in the steepest descent or ascent search direction.

Double-Dogleg Optimization (DBLDOG)

The double-dogleg optimization method combines the ideas of the quasi-Newton and trust region methods. In each iteration, the double-dogleg algorithm computes the step $\mathbf{s}^{(k)}$ as the linear combination of the steepest descent or ascent search direction $\mathbf{s}_1^{(k)}$ and a quasi-Newton search direction $\mathbf{s}_2^{(k)}$:

$$\mathbf{s}^{(k)} = \alpha_1 \mathbf{s}_1^{(k)} + \alpha_2 \mathbf{s}_2^{(k)}$$

The step is requested to remain within a prespecified trust region radius (Fletcher 1987, p. 107). Thus, the DBLDOG subroutine uses the dual quasi-Newton update but does not perform a line search.

The double-dogleg optimization technique works well for medium-sized to moderately large optimization problems, where the objective function and the gradient can be computed much faster than the Hessian. The implementation is based on Dennis and Mei (1979) and Gay (1983), but it is extended for dealing with boundary and linear constraints. The DBLDOG technique generally requires more iterations than the TRUREG, NEWRAP, and NRRIDG techniques, which require second-order derivatives; however, each of the DBLDOG iterations is computationally cheap. Furthermore, the DBLDOG technique requires only gradient calls for the update of the Cholesky factor of an approximate Hessian.

Conjugate Gradient Optimization (CONGRA)

Second-order derivatives are not required by the CONGRA algorithm and are not even approximated. The CONGRA algorithm can be expensive in function and gradient calls, but it requires only $O(p)$ memory for unconstrained optimization. In general, many iterations are required to obtain a precise solution, but each of the CONGRA iterations is computationally cheap.

The CONGRA subroutine should be used for optimization problems with large p. For the unconstrained or boundary-constrained case, CONGRA requires only $O(p)$ bytes of working memory, whereas all other optimization methods require order $O(p^2)$ bytes of working memory. During p successive iterations, uninterrupted by restarts or changes in the working set, the conjugate gradient algorithm computes a cycle of p conjugate search directions. In each iteration, a line search is performed along the search direction to find an approximate optimum of the objective function. The line-search method uses quadratic interpolation and cubic extrapolation to obtain a step size α that satisfies the Goldstein conditions. One of the Goldstein conditions can be violated if the feasible region defines an upper limit for the step size.

Nelder-Mead Simplex Optimization (NMSIMP)

The Nelder-Mead simplex method does not use any derivatives and does not assume that the objective function has continuous derivatives. The objective function itself needs to be continuous. This technique is quite expensive in the number of function calls, and it might be unable to generate precise results for $p \gg 40$.

The original Nelder-Mead simplex algorithm is implemented and extended to boundary constraints. This algorithm does not compute the objective for infeasible points, but it changes the shape of the simplex

adapting to the nonlinearities of the objective function. This change contributes to an increased speed of convergence and uses a special termination criterion.

Displayed Output

The following sections describe the output that PROC HPLOGISTIC produces. The output is organized into various tables, which are discussed in the order of appearance.

Performance Information

The "Performance Information" table is produced by default. It displays information about the execution mode. For single-machine mode, the table displays the number of threads used. For distributed mode, the table displays the grid mode (symmetric or asymmetric), the number of compute nodes, and the number of threads per node.

If you specify the DETAILS option in the PERFORMANCE statement, the procedure also produces a "Timing" table in which elapsed time (absolute and relative) for the main tasks of the procedure are displayed.

Model Information

The "Model Information" table displays basic information about the model, such as the response variable, frequency variable, link function, and the model category the HPLOGISTIC procedure determined based on your input and options. The "Model Information" table also displays the distribution of the data that is assumed by the HPLOGISTIC procedure. See the section "Response Distributions" on page 356 for how the procedure determines the response distribution.

Class Level Information

The "Class Level Information" table lists the levels of every variable specified in the CLASS statement. You should check this information to make sure that the data are correct. You can adjust the order of the CLASS variable levels with the ORDER= option in the CLASS statement. You can suppress the "Class Level Information" table completely or partially with the NOCLPRINT= option in the PROC HPLOGISTIC statement.

If the classification variables use reference parameterization, the "Class Level Information" table also displays the reference value for each variable.

Number of Observations

The "Number of Observations" table displays the number of observations read from the input data set and the number of observations used in the analysis. If a FREQ statement is present, the sum of the frequencies read and used is displayed. If the events/trials syntax is used, the number of events and trials is also displayed. If you specify a PARTITION statement, the table displays the values for each role.

Response Profile

The "Response Profile" table displays the ordered value from which the HPLOGISTIC procedure determines the probability being modeled as an event in binary models and the ordering of categories in multinomial models. For each response category level, the frequency used in the analysis is reported. You can affect

the ordering of the response values with the *response-options* in the MODEL statement. For binary and generalized logit models, the note that follows the "Response Profile" table indicates which outcome is modeled as the event in binary models and which value serves as the reference category.

The "Response Profile" table is not produced for binomial data. You can find information about the number of events and trials in the "Number of Observations" table. If you specify a PARTITION statement, the table displays the values for each role.

Selection Information

When you specify the SELECTION statement, the HPLOGISTIC procedure produces by default a series of tables with information about the model selection. The "Selection Information" table informs you about the model selection method, selection and stop criteria, and other parameters that govern the selection. You can suppress this table by specifying DETAILS=NONE in the SELECTION statement.

Selection Summary

When you specify the SELECTION statement, the HPLOGISTIC procedure produces the "Selection Summary" table with information about which effects were entered into or removed from the model at the steps of the model selection process. The statistic that led to the removal or entry decision is also displayed. You can request further details about the model selection steps by specifying DETAILS=STEPS or DETAILS=ALL in the SELECTION statement. You can suppress the display of the "Selection Summary" table by specifying DETAILS=NONE in the SELECTION statement.

Stop Reason

When you specify the SELECTION statement, the HPLOGISTIC procedure produces a simple table that tells you why model selection stopped.

Selection Reason

When you specify the SELECTION statement, the HPLOGISTIC procedure produces a simple table that tells you why the final model was selected.

Selected Effects

When you specify the SELECTION statement, the HPLOGISTIC procedure produces a simple table that tells you which effects were selected into the final model.

Candidate Entry and Removal Details

When you specify the DETAILS=ALL or DETAILS=STEPS option in the SELECTION statement, the HPLOGISTIC procedure produces the "Candidate Entry and Removal Details" table, which displays the effect names and values of the criterion used to select entering or departing effects at each step of the selection process. For each step, the effects are displayed in sorted order from best to worst of the selection criterion.

Selection Details

When you specify the DETAILS=ALL option in the SELECTION statement, the HPLOGISTIC procedure produces the "Selection Details" table, which contains information about which effects were entered into or

removed from the model at the steps of the model selection process. If you specify SELECT=AIC, AICC, or BIC then the likelihood ratio chi-square statistic is displayed along with the estimated selection criteria; otherwise the score or Wald chi-square statistic is displayed. Fit statistics computed at each step are also displayed.

Iteration History

For each iteration of the optimization, the "Iteration History" table displays the number of function evaluations (including gradient and Hessian evaluations), the value of the objective function, the change in the objective function from the previous iteration and the absolute value of the largest (projected) gradient element. The objective function used in the optimization in the HPLOGISTIC procedure is normalized by default to enable comparisons across data sets with different sampling intensity. You can control normalization with the NORMALIZE= option in the PROC HPLOGISTIC statement.

If you specify the ITDETAILS option in the PROC HPLOGISTIC statement, information about the parameter estimates and gradients in the course of the optimization is added to the "Iteration History" table.

The "Iteration History" table is displayed by default unless you specify the NOITPRINT option or perform a model selection. To generate the history from a model selection process, specify the ITSELECT or ITDETAILS option.

Convergence Status

The convergence status table is a small ODS table that follows the "Iteration History" table in the default output. In the listing it appears as a message that indicates whether the optimization succeeded and which convergence criterion was met. If the optimization fails, the message indicates the reason for the failure. If you save the convergence status table to an output data set, a numeric Status variable is added that enables you to assess convergence programmatically. The values of the Status variable encode the following:

0 Convergence was achieved, or an optimization was not performed (because TECHNIQUE=NONE is specified).

1 The objective function could not be improved.

2 Convergence was not achieved because of a user interrupt or because a limit was exceeded, such as the maximum number of iterations or the maximum number of function evaluations. To modify these limits, see the MAXITER=, MAXFUNC=, and MAXTIME= options in the PROC HPLOGISTIC statement.

3 Optimization failed to converge because function or derivative evaluations failed at the starting values or during the iterations or because a feasible point that satisfies the parameter constraints could not be found in the parameter space.

Dimensions

The "Dimensions" table displays size measures that are derived from the model and the environment. For example, it displays the number of columns in the design matrix, the rank of the matrix, the largest number of design columns associated with an effect, the number of compute nodes in distributed mode, and the number of threads per node.

Fit Statistics

The "Fit Statistics" table displays a variety of likelihood-based measures of fit. All statistics are presented in "smaller is better" form. If you specify a PARTITION statement, the table displays the values for each role. The values displayed in the "Fit Statistics" table are not based on a normalized log-likelihood function. For more information, see the section "Information Criteria" on page 361.

Partition Fit Statistics

If you specify the PARTITION statement, the "Partition Fit Statistics" table displays statistics for comparing the training, validation, and testing results. For more information about the statistics displayed in this table, see the sections "Partition Fit Statistics" on page 361, "Model Fit and Assessment Statistics" on page 361, and "The Hosmer-Lemeshow Goodness-of-Fit Test" on page 366.

Global Tests

The "Global Tests" table provides a statistical test for the hypothesis of whether the final model provides a better fit than a model without effects (an "intercept-only" model).

If you specify the NOINT option in the MODEL statement, the reference model is one where the linear predictor is 0 for all observations.

Partition for the Hosmer and Lemeshow Test

If you specify the LACKFIT option in the MODEL statement, the "Partition for the Hosmer and Lemeshow Test" table displays the grouping used in the Hosmer-Lemeshow test. If you specify a PARTITION statement, a table is displayed for each role. For more information, see the section "The Hosmer-Lemeshow Goodness-of-Fit Test" on page 366. For examples of using this partition, see Hosmer and Lemeshow (2000).

Hosmer and Lemeshow Goodness-of-Fit Test

If you specify the LACKFIT option in the MODEL statement, the "Hosmer and Lemeshow Goodness-of-Fit Test" table provides a test of the fit of the model; small p-values reject the null hypothesis that the fitted model is adequate. If you specify a PARTITION statement, a row is displayed for each role. For more information, see the section "The Hosmer-Lemeshow Goodness-of-Fit Test" on page 366.

Association Statistics

If you specify the ASSOCIATION option in the MODEL statement, the "Association Statistics" table displays the concordance index C (the area under the ROC curve, AUC), Somers' D statistic (Gini's coefficient), Goodman-Kruskal's gamma statistic, and Kendall's tau-a statistic. If you also specify a PARTITION statement, a row is displayed for each role. For more information, see the section "Association Statistics" on page 364.

Classification Table

The "Classification" table is displayed if you specify the CTABLE option without specifying an output data set. If you also specify a PARTITION statement, a table is displayed for each role. For more information, see the section "Classification Table and ROC Curves" on page 363.

Parameter Estimates

The parameter estimates, their estimated (asymptotic) standard errors, and p-values for the hypothesis that the parameter is 0 are presented in the "Parameter Estimates" table. If you request confidence intervals with the CL or ALPHA= options in the MODEL statement, confidence limits are produced for the estimate on the linear scale.

ODS Table Names

Each table created by the HPLOGISTIC procedure has a name associated with it, and you must use this name to refer to the table when you use ODS statements. These names are listed in Table 9.10.

Table 9.10 ODS Tables Produced by PROC HPLOGISTIC

Table Name	Description	Statement	Option
Association	Association of predicted probabilities and observed responses	MODEL	ASSOCIATION
CandidateDetails	Details about candidates for entry into or removal from the model	SELECTION	DETAILS=STEP
Classification	Classification table	MODEL	CTABLE
ClassLevels	Level information from the CLASS statement	CLASS	Default
ConvergenceStatus	Status of optimization at conclusion of optimization	PROC	Default
Dimensions	Model dimensions	PROC	Default
FitStatistics	Fit statistics	PROC	Default
GlobalTests	Test of the model versus the null model	PROC	Default
IterHistory	Iteration history	PROC	Default or ITSELECT
LackFitChiSq	Hosmer-Lemeshow chi-square test results	MODEL	LACKFIT
LackFitPartition	Partition for the Hosmer-Lemeshow test	MODEL	LACKFIT
ModelInfo	Information about the modeling environment	PROC	Default
NObs	Number of observations read and used, and number of events and trials, if applicable	PROC	Default
ParameterEstimates	Solutions for the parameter estimates associated with effects in MODEL statements	PROC	Default

Table 9.10 *continued*

Table Name	Description	Statement	Option
PartFitStats	Fit statistics for the data roles	PARTITION statement	Default
PerformanceInfo	Information about the high-performance computing environment	PROC	Default
ResponseProfile	Response categories and category modeled in models for binary and multinomial data	PROC	Default
SelectedEffects	List of effects selected into model	SELECTION	Default
SelectionDetails	Details about model selection, including fit statistics by step	SELECTION	DETAILS=ALL
SelectionInfo	Information about the settings for model selection	SELECTION	Default
SelectionReason	Reason why the particular model was selected	SELECTION	Default
SelectionSummary	Summary information about model selection steps	SELECTION	Default
StopReason	Reason for termination of model selection	SELECTION	Default
Timing	Absolute and relative times for tasks performed by the procedure	PERFORMANCE	DETAILS

Examples: HPLOGISTIC Procedure

Example 9.1: Model Selection

The following HPLOGISTIC statements examine the same data as in the section "Getting Started: HPLOGISTIC Procedure" on page 332, but they request model selection via the forward selection technique. Model effects are added in the order of their significance until no more effects make a significant improvement of the current model. The DETAILS=ALL option in the SELECTION statement requests that all tables related to model selection be produced.

```
proc hplogistic data=getStarted;
   class C;
   model y = C x1-x10;
   selection method=forward details=all;
run;
```

The model selection tables are shown in Output 9.1.1 through Output 9.1.4.

The "Selection Information" table in Output 9.1.1 summarizes the settings for the model selection. Effects are added to the model only if they produce a significant improvement as judged by comparing the p-value of a score test to the entry significance level (SLE), which is 0.05 by default. The forward selection stops when no effect outside the model meets this criterion.

Output 9.1.1 Selection Information

The HPLOGISTIC Procedure

Selection Information	
Selection Method	Forward
Select Criterion	Significance Level
Stop Criterion	Significance Level
Effect Hierarchy Enforced	None
Entry Significance Level (SLE)	0.05
Stop Horizon	1

The "Selection Summary" table in Output 9.1.2 shows the effects that were added to the model and their significance level. Step 0 refers to the null model that contains only an intercept. In the next step, effect x8 made the most significant contribution to the model among the candidate effects ($p = 0.0381$). In step 2 the most significant contribution when adding an effect to a model that contains the intercept and x8 was made by x2. In the subsequent step no effect could be added to the model that would produce a p-value less than 0.05, so variable selection stops.

Output 9.1.2 Selection Summary Information

Selection Summary

Step	Effect Entered	Number Effects In	p Value
0	Intercept	1	.
1	x8	2	0.0381
2	x2	3	0.0255

Selection stopped because no candidate for entry is significant at the 0.05 level.

Selected Effects: Intercept x2 x8

The DETAILS=ALL option requests further detail information about the steps of the model selection. The "Candidate Details" table in Output 9.1.3 list all candidates for each step in the order of significance of their score tests. The effect with smallest *p*-value less than the SLE level of 0.05 is added in each step.

Output 9.1.3 Candidate Details

Candidate Entry and Removal Details

Step	Rank	Effect	Candidate For	p Value
1	1	x8	Entry	0.0381
	2	x2	Entry	0.0458
	3	x4	Entry	0.0557
	4	x9	Entry	0.1631
	5	C	Entry	0.1858
	6	x1	Entry	0.2715
	7	x10	Entry	0.4434
	8	x5	Entry	0.7666
	9	x3	Entry	0.8006
	10	x7	Entry	0.8663
	11	x6	Entry	0.9626
2	1	x2	Entry	0.0255
	2	x4	Entry	0.0721
	3	x9	Entry	0.1080
	4	C	Entry	0.1241
	5	x1	Entry	0.2778
	6	x10	Entry	0.5250
	7	x5	Entry	0.6993
	8	x7	Entry	0.7103
	9	x3	Entry	0.8743
	10	x6	Entry	0.9577

The DETAILS=ALL option also produces the "Selection Details" table, which provides fit statistics and the value of the score test chi-square statistic at each step.

Output 9.1.4 Selection Details

Selection Details								
Step	Effect Entered	Number Effects In	Chi-Square	Pr > ChiSq	-2 LogL	AIC	AICC	BIC
0	Initial Model	1			123.82	125.82	125.86	128.43
1	x8	2	4.2986	0.0381	119.46	123.46	123.59	128.67
2	x2	3	4.9882	0.0255	114.40	120.40	120.65	128.21

Output 9.1.5 displays information about the selected model. Notice that the –2 log likelihood value in the "Fit Statistics" table is larger than the value for the full model in Figure 9.9. This is expected because the selected model contains only a subset of the parameters. Because the selected model is more parsimonious than the full model, the discrepancy between the –2 log likelihood and the information criteria is less severe than previously noted.

Output 9.1.5 Fit Statistics and Null Test

Fit Statistics	
-2 Log Likelihood	114.40
AIC (smaller is better)	120.40
AICC (smaller is better)	120.65
BIC (smaller is better)	128.21

Testing Global Null Hypothesis: BETA=0			
Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	9.4237	2	0.0090

The parameter estimates of the selected model are given in Output 9.1.6. Notice that the effects are listed in the "Parameter Estimates" table in the order in which they were specified in the MODEL statement and not in the order in which they were added to the model.

Output 9.1.6 Parameter Estimates

Parameter Estimates					
Parameter	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept	0.8584	0.5503	Infty	1.56	0.1188
x2	-0.2502	0.1146	Infty	-2.18	0.0290
x8	1.7840	0.7908	Infty	2.26	0.0241

You can construct the prediction equation for this model from the parameter estimates as follows. The estimated linear predictor for an observation is

$$\widehat{\eta} = 0.8584 - 0.2503 \times x_2 + 1.7840 \times x_8$$

and the predicted probability that variable y takes on the value 0 is

$$\widehat{\Pr}(Y = 0) = \frac{1}{1 + \exp\{-\widehat{\eta}\}}$$

Example 9.2: Modeling Binomial Data

If $Y_1, \cdots, Y_n$ are independent binary (Bernoulli) random variables with common success probability π, then their sum is a binomial random variable. In other words, a binomial random variable with parameters n and π can be generated as the sum of n Bernoulli(π) random experiments. The HPLOGISTIC procedure uses a special syntax to express data in binomial form, the *events/trials* syntax.

Consider the following data, taken from Cox and Snell (1989, pp. 10–11), of the number, r, of ingots not ready for rolling, out of n tested, for a number of combinations of heating time and soaking time. If each test is carried out independently and if for a particular combination of heating and soaking time there is a constant probability that the tested ingot is not ready for rolling, then the random variable r follows a Binomial(n, π) distribution, where the success probability π is a function of heating and soaking time.

```
data Ingots;
   input Heat Soak r n @@;
   Obsnum= _n_;
   datalines;
7 1.0 0 10  14 1.0 0 31  27 1.0 1 56  51 1.0 3 13
7 1.7 0 17  14 1.7 0 43  27 1.7 4 44  51 1.7 0  1
7 2.2 0  7  14 2.2 2 33  27 2.2 0 21  51 2.2 0  1
7 2.8 0 12  14 2.8 0 31  27 2.8 1 22  51 4.0 0  1
7 4.0 0  9  14 4.0 0 19  27 4.0 1 16
;
```

The following statements show the use of the events/trials syntax to model the binomial response. The *events* variable in this situation is r, the number of ingots not ready for rolling, and the *trials* variable is n, the number of ingots tested. The dependency of the probability of not being ready for rolling is modeled as a function of heating time, soaking time, and their interaction. The ASSOCIATION option displays ordinal measures of association between the observed responses and predicted probabilities. The CTABLE=ROC option stores statistics that are used for evaluating the predictive power of the model in the Roc data set. The LACKFIT option produces the Hosmer and Lemeshow goodness-of-fit test for binary response models. The OUTPUT statement stores the linear predictors and the predicted probabilities in the Out data set along with the ID variable.

```
proc hplogistic data=Ingots;
   model r/n = Heat Soak Heat*Soak / association ctable=Roc lackfit;
   id Obsnum;
   output out=Out xbeta predicted=Pred;
run;
```

The "Performance Information" table in Output 9.2.1 shows that the procedure executes in single-machine mode. The example is executed on a single machine with the same number of cores as the number of threads used; that is, one computational thread was spawned per CPU.

Output 9.2.1 Performance Information

The HPLOGISTIC Procedure

Performance Information	
Execution Mode	Single-Machine
Number of Threads	4

The "Model Information" table shows that the data are modeled as binomially distributed with a logit link function (Output 9.2.2). This is the default link function in the HPLOGISTIC procedure for binary and binomial data. The procedure estimates the parameters of the model by a Newton-Raphson algorithm.

Output 9.2.2 Model Information and Number of Observations

Model Information	
Data Source	WORK.INGOTS
Response Variable (Events)	r
Response Variable (Trials)	n
Distribution	Binomial
Link Function	Logit
Optimization Technique	Newton-Raphson with Ridging

Number of Observations Read	19
Number of Observations Used	19
Number of Events	12
Number of Trials	387

The second table in Output 9.2.2 shows that all 19 observations in the data set were used in the analysis, and that the total number of events and trials equal 12 and 387, respectively. These are the sums of the variables r and n across all observations.

Output 9.2.3 displays the "Iteration History" and convergence status tables for this run. The HPLOGISTIC procedure converged after four iterations (not counting the initial setup iteration) and meets the GCONV= convergence criterion.

Output 9.2.3 Iteration History and Convergence Status

Iteration History				
Iteration	**Evaluations**	**Objective Function**	**Change**	**Max Gradient**
0	4	0.7676329445	.	6.378002
1	2	0.7365832479	0.03104970	0.754902
2	2	0.7357086248	0.00087462	0.023623
3	2	0.7357075299	0.00000109	0.00003
4	2	0.7357075299	0.00000000	5.42E-11

Convergence criterion (GCONV=1E-8) satisfied.

Output 9.2.4 displays the "Dimensions" table for the model. There are four columns in the design matrix of the model (the **X** matrix); they correspond to the intercept, the Heat effect, the Soak effect, and the interaction of the Heat and Soak effects. The model is nonsingular, since the rank of the crossproducts matrix equals the number of columns in **X**. All parameters are estimable and participate in the optimization.

Output 9.2.4 Dimensions in Binomial Logistic Regression

Dimensions	
Columns in X	4
Number of Effects	4
Max Effect Columns	1
Rank of Cross-product Matrix	4
Parameters in Optimization	4

Output 9.2.5 displays the "Fit Statistics" table for this run. Evaluated at the converged estimates, –2 times the value of the log-likelihood function equals 27.9569. Further fit statistics are also given, all of them in "smaller is better" form. The AIC, AICC, and BIC criteria are used to compare non-nested models and to penalize the model fit for the number of observations and parameters. The –2 log-likelihood value can be used to compare nested models by way of a likelihood ratio test.

Output 9.2.5 Fit Statistics

Fit Statistics	
-2 Log Likelihood	27.9569
AIC (smaller is better)	35.9569
AICC (smaller is better)	38.8140
BIC (smaller is better)	39.7346

Output 9.2.6 shows the test of the global hypothesis that the effects jointly do not impact the probability of ingot readiness. The chi-square test statistic can be obtained by comparing the –2 log-likelihood value of the model with covariates to the value in the intercept-only model. The test is significant with a p-value of 0.0082. One or more of the effects in the model have a significant impact on the probability of ingot readiness.

Output 9.2.6 Null Test

Testing Global Null Hypothesis: BETA=0			
Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	11.7663	3	0.0082

Output 9.2.7 shows the tables that are produced when you specify the LACKFIT option in the MODEL statement. The first table displays the partition that PROC HPLOGISTIC uses to compute the chi-square test that is displayed in the second table; the large p-value does not reject the adequacy of the model fit. For more information, see the section "The Hosmer-Lemeshow Goodness-of-Fit Test" on page 366.

Output 9.2.7 Hosmer-Lemeshow Goodness-of-Fit Test

Partition for the Hosmer and Lemeshow Test					
	Events		Nonevents		
Group	Observed	Expected	Observed	Expected	Total
1	0	0.24	34	33.76	34
2	0	0.47	43	42.53	43
3	0	0.66	52	51.34	52
4	2	0.46	31	32.54	33
5	0	0.48	31	30.52	31
6	0	0.36	19	18.64	19
7	1	1.94	55	54.06	56
8	4	1.59	40	42.41	44
9	1	1.63	42	41.37	43
10	4	4.17	28	27.83	32

Hosmer and Lemeshow Goodness-of-Fit Test		
Chi-Square	DF	Pr > ChiSq
11.9771	8	0.1522

Output 9.2.8 displays the "Association Statistics" table, which is produced when you specify the ASSOCIATION option in the MODEL statement. The table contains four measures of association for assessing the predictive ability of a model. For more information, see the section "Association Statistics" on page 364.

Output 9.2.8 Association of Observed Responses and Predicted Probabilities

Association Statistics			
Concordance Index	Somers' D	Gamma	Tau-a
0.770556	0.541111	0.585759	0.032601

The "Parameter Estimates" table in Output 9.2.9 displays the estimates and standard errors of the model effects.

Output 9.2.9 Parameter Estimates

Parameter Estimates					
Parameter	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept	-5.9902	1.6666	Infty	-3.59	0.0003
Heat	0.09634	0.04707	Infty	2.05	0.0407
Soak	0.2996	0.7551	Infty	0.40	0.6916
Heat*Soak	-0.00884	0.02532	Infty	-0.35	0.7270

You can construct the prediction equation of the model from the "Parameter Estimates" table. For example, an observation with Heat equal to 14 and Soak equal to 1.7 has linear predictor

$$\widehat{\eta} = -5.9902 + 0.09634 \times 14 + 0.2996 \times 1.7 - 0.00884 \times 14 \times 7 = -4.34256$$

The probability that an ingot with these characteristics is not ready for rolling is

$$\widehat{\pi} = \frac{1}{1 + \exp\{-(-4.34256)\}} = 0.01284$$

The OUTPUT statement computes these linear predictors and probabilities and stores them in the Out data set. This data set also contains the ID variable, which is used by the following statements to attach the covariates to these statistics. Output 9.2.10 shows the probability that an ingot with Heat equal to 14 and Soak equal to 1.7 is not ready for rolling.

```
data Out;
   merge Out Ingots;
   by Obsnum;
proc print data=Out;
   where Heat=14 & Soak=1.7;
run;
```

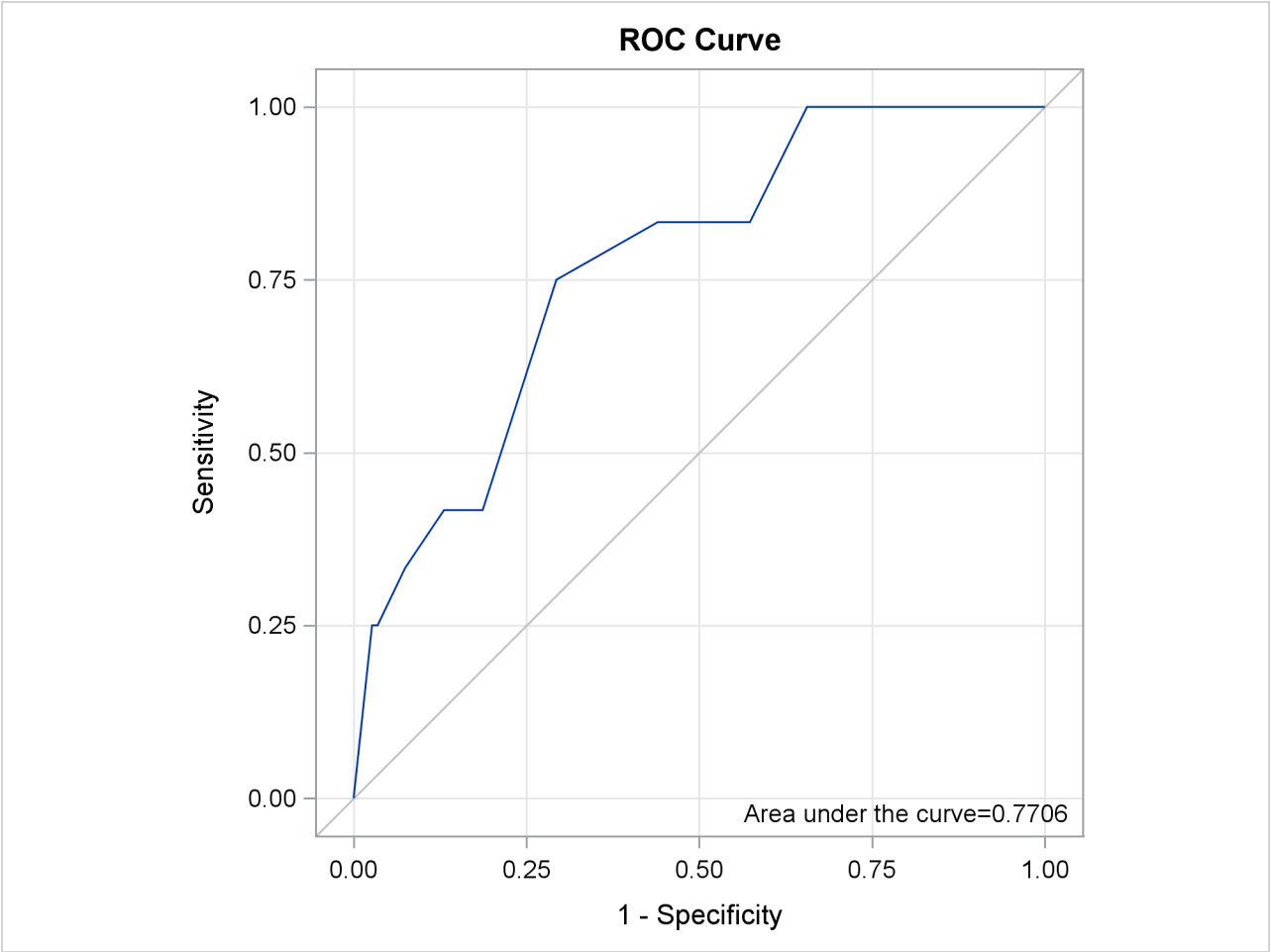
Output 9.2.10 Predicted Probability for Heat=14 and Soak=1.7

Obs	Obsnum	Pred	Xbeta	Heat	Soak	r	n
6	6	0.012836	-4.34256	14	1.7	0	43

The CTABLE=ROC option computes statistics for binary response models (based on classifying observations according to whether their predicted probabilities exceed certain values) and stores the results in the Roc data set. For more information, see the section "Classification Table and ROC Curves" on page 363. You can use this data set to display the ROC curve by using the SGPLOT procedure as follows:

```
proc sgplot data=Roc aspect=1 noautolegend;
   title 'ROC Curve';
   xaxis values=(0 to 1 by 0.25) grid offsetmin=.05 offsetmax=.05;
   yaxis values=(0 to 1 by 0.25) grid offsetmin=.05 offsetmax=.05;
   lineparm x=0 y=0 slope=1 / lineattrs=(color=ligr);
   series x=FPF y=TPF;
   inset 'Area under the curve=0.7706' / position=bottomright;
run;
```

Output 9.2.11 Receiver Operating Characteristics Curve



Binomial data are a form of grouped binary data where "successes" in the underlying Bernoulli trials are totaled. You can thus unwind data for which you use the events/trials syntax and fit it with techniques for binary data.

The following DATA step expands the Ingots data set with 12 events in 387 trials into a binary data set with 387 observations.

```
data Ingots_binary;
   set Ingots;
   do i=1 to n;
     if i <= r then y=1; else y = 0;
     output;
   end;
run;
```

The following HPLOGISTIC statements fit the model with Heat effect, Soak effect, and their interaction to the binary data set. The **event='1'** response-variable option in the MODEL statement ensures that the HPLOGISTIC procedure models the probability that the variable y takes on the value ('1').

```
proc hplogistic data=Ingots_binary;
   model y(event='1') = Heat Soak Heat*Soak;
run;
```

Output 9.2.12 displays the "Performance Information", "Model Information," "Number of Observations," and the "Response Profile" tables. The data are now modeled as binary (Bernoulli distributed) with a logit link function. The "Response Profile" table shows that the binary response breaks down into 375 observations where y equals zero and 12 observations where y equals 1.

Output 9.2.12 Model Information in Binary Model

ROC Curve

The HPLOGISTIC Procedure

Performance Information	
Execution Mode	Single-Machine
Number of Threads	4

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

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

Number of Observations Read	387
Number of Observations Used	387

Response Profile		
Ordered Value	**y**	**Total Frequency**
1	0	375
2	1	12

You are modeling the probability that y='1'.

Output 9.2.13 displays the result for the test of the global null hypothesis and the parameter estimates. These results match those in Output 9.2.6 and Output 9.2.9.

Output 9.2.13 Null Test and Parameter Estimates

Testing Global Null Hypothesis: BETA=0			
Test	**Chi-Square**	**DF**	**Pr > ChiSq**
Likelihood Ratio	11.7663	3	0.0082

Output 9.2.13 *continued*

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept	-5.9902	1.6666	Infty	-3.59	0.0003
Heat	0.09634	0.04707	Infty	2.05	0.0407
Soak	0.2996	0.7551	Infty	0.40	0.6916
Heat*Soak	-0.00884	0.02532	Infty	-0.35	0.7270

Example 9.3: Ordinal Logistic Regression

Consider a study of the effects of various cheese additives on taste. Researchers tested four cheese additives and obtained 52 response ratings for each additive. Each response was measured on a scale of nine categories ranging from strong dislike (1) to excellent taste (9). The data, given in McCullagh and Nelder (1989, p. 175) in the form of a two-way frequency table of additive by rating, are saved in the data set Cheese by using the following program. The variable y contains the response rating. The variable Additive specifies the cheese additive (1, 2, 3, or 4). The variable freq gives the frequency with which each additive received each rating.

```
data Cheese;
   do Additive = 1 to 4;
      do y = 1 to 9;
         input freq @@;
         output;
      end;
   end;
   label y='Taste Rating';
   datalines;
0  0  1  7  8  8 19  8  1
6  9 12 11  7  6  1  0  0
1  1  6  8 23  7  5  1  0
0  0  0  1  3  7 14 16 11
;
```

The response variable y is ordinally scaled. A cumulative logit model is used to investigate the effects of the cheese additives on taste. The following statements invoke PROC HPLOGISTIC to fit this model with y as the response variable and three indicator variables as explanatory variables, with the fourth additive as the reference level. With this parameterization, each Additive parameter compares an additive to the fourth additive.

```
proc hplogistic data=Cheese;
   freq freq;
   class Additive(ref='4') / param=ref ;
   model y=Additive;
   title 'Multiple Response Cheese Tasting Experiment';
run;
```

Results from the logistic analysis are shown in Output 9.3.1 through Output 9.3.3.

The "Response Profile" table in Output 9.3.1 shows that the strong dislike (y=1) end of the rating scale is associated with lower Ordered Values in the "Response Profile" table; hence the probability of disliking the additives is modeled.

Output 9.3.1 Proportional Odds Model Regression Analysis

Multiple Response Cheese Tasting Experiment

The HPLOGISTIC Procedure

Performance Information	
Execution Mode	Single-Machine
Number of Threads	4

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

Model Information	
Data Source	WORK.CHEESE
Response Variable	y
Frequency Variable	freq
Class Parameterization	Reference
Distribution	Multinomial
Link Function	Cumulative Logit
Optimization Technique	Newton-Raphson with Ridging

Class Level Information			
Class	**Levels**	**Reference Value**	**Values**
Additive	4	4	1 2 3 4

Number of Observations Read	36
Number of Observations Used	28
Sum of Frequencies Read	208
Sum of Frequencies Used	208

Response Profile		
Ordered Value	**y**	**Total Frequency**
1	1	7
2	2	10
3	3	19
4	4	27
5	5	41
6	6	28
7	7	39
8	8	25
9	9	12

You are modeling the probabilities of levels of y having lower Ordered Values in the Response Profile Table.

Output 9.3.2 Proportional Odds Model Regression Analysis

Iteration History

Iteration	Evaluations	Objective Function	Change	Max Gradient
0	4	2.0668312595	.	0.137412
1	2	1.7319560317	0.33487523	0.062757
2	2	1.7105150048	0.02144103	0.008919
3	2	1.7099716191	0.00054339	0.00035
4	2	1.7099709251	0.00000069	6.981E-7
5	2	1.7099709251	0.00000000	2.98E-12

Convergence criterion (GCONV=1E-8) satisfied.

Dimensions	
Columns in X	11
Number of Effects	2
Max Effect Columns	3
Rank of Cross-product Matrix	11
Parameters in Optimization	11

Fit Statistics	
-2 Log Likelihood	711.35
AIC (smaller is better)	733.35
AICC (smaller is better)	734.69
BIC (smaller is better)	770.06

Testing Global Null Hypothesis: BETA=0

Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	148.4539	3	<.0001

The positive value (1.6128) for the parameter estimate for Additive=1 in Output 9.3.3 indicates a tendency toward the lower-numbered categories of the first cheese additive relative to the fourth. In other words, the fourth additive tastes better than the first additive. Similarly, the second and third additives are both less favorable than the fourth additive. The relative magnitudes of these slope estimates imply the preference ordering: fourth, first, third, second.

Output 9.3.3 Proportional Odds Model Regression Analysis

Parameter Estimates						
Parameter	Taste Rating	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept	1	-7.0802	0.5640	Infty	-12.55	<.0001
Intercept	2	-6.0250	0.4764	Infty	-12.65	<.0001
Intercept	3	-4.9254	0.4257	Infty	-11.57	<.0001
Intercept	4	-3.8568	0.3880	Infty	-9.94	<.0001
Intercept	5	-2.5206	0.3453	Infty	-7.30	<.0001
Intercept	6	-1.5685	0.3122	Infty	-5.02	<.0001
Intercept	7	-0.06688	0.2738	Infty	-0.24	0.8071
Intercept	8	1.4930	0.3357	Infty	4.45	<.0001
Additive 1		1.6128	0.3805	Infty	4.24	<.0001
Additive 2		4.9646	0.4767	Infty	10.41	<.0001
Additive 3		3.3227	0.4218	Infty	7.88	<.0001

References

Akaike, H. (1974), "A New Look at the Statistical Model Identification," *IEEE Transactions on Automatic Control*, AC-19, 716–723.

Albert, A. and Anderson, J. A. (1984), "On the Existence of Maximum Likelihood Estimates in Logistic Regression Models," *Biometrika*, 71, 1–10.

Brier, G. W. (1950), "Verification of Forecasts Expressed in Terms of Probability," *Monthly Weather Review*, 78, 1–3.

Burnham, K. P. and Anderson, D. R. (1998), *Model Selection and Inference: A Practical Information-Theoretic Approach*, New York: Springer-Verlag.

Cox, D. R. and Snell, E. J. (1989), *The Analysis of Binary Data*, 2nd Edition, London: Chapman & Hall.

Dennis, J. E., Gay, D. M., and Welsch, R. E. (1981), "An Adaptive Nonlinear Least-Squares Algorithm," *ACM Transactions on Mathematical Software*, 7, 348–368.

Dennis, J. E. and Mei, H. H. W. (1979), "Two New Unconstrained Optimization Algorithms Which Use Function and Gradient Values," *Journal of Optimization Theory and Applications*, 28, 453–482.

Eskow, E. and Schnabel, R. B. (1991), "Algorithm 695: Software for a New Modified Cholesky Factorization," *ACM Transactions on Mathematical Software*, 17, 306–312.

Fleiss, J. L. (1981), *Statistical Methods for Rates and Proportions*, 2nd Edition, New York: John Wiley & Sons.

Fletcher, R. (1987), *Practical Methods of Optimization*, 2nd Edition, Chichester, UK: John Wiley & Sons.

Gay, D. M. (1983), "Subroutines for Unconstrained Minimization," *ACM Transactions on Mathematical Software*, 9, 503–524.

Hastie, T. J., Tibshirani, R. J., and Friedman, J. H. (2001), *The Elements of Statistical Learning*, New York: Springer-Verlag.

Hosmer, D. W., Jr. and Lemeshow, S. (2000), *Applied Logistic Regression*, 2nd Edition, New York: John Wiley & Sons.

Hurvich, C. M. and Tsai, C.-L. (1989), "Regression and Time Series Model Selection in Small Samples," *Biometrika*, 76, 297–307.

Lawless, J. F. and Singhal, K. (1978), "Efficient Screening of Nonnormal Regression Models," *Biometrics*, 34, 318–327.

Magee, L. (1990), "R^2 Measures Based on Wald and Likelihood Ratio Joint Significant Tests," *American Statistician*, 44, 250–253.

McCullagh, P. and Nelder, J. A. (1989), *Generalized Linear Models*, 2nd Edition, London: Chapman & Hall.

McFadden, D. (1974), "Conditional Logit Analysis of Qualitative Choice Behavior," in P. Zarembka, ed., *Frontiers in Econometrics*, New York: Academic Press.

McNicol, D. (2005), *A Primer of Signal Detection Theory*, Mahwah, NJ: Lawrence Erlbaum Associates.

Moré, J. J. and Sorensen, D. C. (1983), "Computing a Trust-Region Step," *SIAM Journal on Scientific and Statistical Computing*, 4, 553–572.

Murphy, A. H. (1973), "A New Vector Partition of the Probability Score," *Journal of Applied Meterology*, 12, 595–600.

Nagelkerke, N. J. D. (1991), "A Note on a General Definition of the Coefficient of Determination," *Biometrika*, 78, 691–692.

Pepe, M. S. (2003), *The Statistical Evaluation of Medical Tests for Classification and Prediction*, New York: Oxford University Press.

Santner, T. J. and Duffy, D. E. (1986), "A Note on A. Albert and J. A. Anderson's Conditions for the Existence of Maximum Likelihood Estimates in Logistic Regression Models," *Biometrika*, 73, 755–758.

Schwarz, G. (1978), "Estimating the Dimension of a Model," *Annals of Statistics*, 6, 461–464.

Tjur, T. (2009), "Coefficients of Determination in Logistic Regression Models—A New Proposal: The Coefficient of Discrimination," *American Statistician*, 63, 366–372.

Chapter 10
The HPNLMOD Procedure

Contents

Overview: HPNLMOD Procedure **393**
PROC HPNLMOD Features 394
PROC HPNLMOD Contrasted with the NLIN and NLMIXED Procedures 394
Getting Started: HPNLMOD Procedure **395**
Least Squares Model 395
Binomial Model 396
Syntax: HPNLMOD Procedure **397**
PROC HPNLMOD Statement 398
BOUNDS Statement 403
BY Statement 403
ESTIMATE Statement 404
MODEL Statement 404
PARAMETERS Statement 405
PERFORMANCE Statement 407
PREDICT Statement 408
RESTRICT Statement 409
Programming Statements 410
Details: HPNLMOD Procedure **412**
Least Squares Estimation 412
Built-In Log-Likelihood Functions 412
Computational Method 414
Choosing an Optimization Algorithm 415
Displayed Output 419
ODS Table Names 422
Examples: HPNLMOD Procedure **423**
Example 10.1: Segmented Model 423
References **426**

Overview: HPNLMOD Procedure

The HPNLMOD procedure is a high-performance procedure that uses either nonlinear least squares or maximum likelihood to fit nonlinear regression models. PROC HPNLMOD enables you to specify the model by using SAS programming statements, which give you greater flexibility in modeling the relationship

between the response variable and independent (regressor) variables than do SAS procedures that use a more structured MODEL statement.

PROC HPNLMOD runs in either single-machine mode or distributed mode.

NOTE: Distributed mode requires SAS High-Performance Statistics.

PROC HPNLMOD Features

The HPNLMOD procedure does the following:

- reads input data in parallel and writes output data in parallel when the data source is the appliance database
- is highly multithreaded during all phases of analytic execution
- computes analytical derivatives of user-provided expressions for more robust parameter estimations
- evaluates user-provided expressions and their confidence limits by using the ESTIMATE and PREDICT statements
- estimates parameters without specifying a particular distribution function by using the least squares method
- estimates parameters by using the maximum likelihood method when either a built-in distribution function is specified or a likelihood function is provided

Because the HPNLMOD procedure is a high-performance analytical procedure, it also does the following:

- enables you to run in distributed mode on a cluster of machines that distribute the data and the computations
- enables you to run in single-machine mode on the server where SAS is installed
- exploits all the available cores and concurrent threads, regardless of execution mode

For more information, see the section "Processing Modes" on page 10 in Chapter 3, "Shared Concepts and Topics."

PROC HPNLMOD Contrasted with the NLIN and NLMIXED Procedures

Like the NLIN procedure, the HPNLMOD procedure estimates parameters by using least squares minimization for models that are specified by SAS programming statements. However, PROC HPNLMOD can also perform maximum likelihood estimation when information about the response variable's distribution is provided. PROC HPNLMOD also has a RESTRICT statement for specifying restrictions on parameter estimates that are more general than those that are available in PROC NLIN. Because the HPNLMOD and

NLIN procedures use different optimization techniques, the available options that control the estimation process and resulting parameter estimates can differ between these procedures when equivalent models and data are analyzed.

Although it does not support the specification of random effects, PROC HPNLMOD is similar to PROC NLMIXED. Both procedures perform maximum likelihood estimation by using the same programming syntax and set of distributions to specify the model's mean term. In addition, both PROC HPNLMOD and PROC NLMIXED use the same optimization techniques and options. However, PROC NLMIXED does not support least squares parameter estimation.

Getting Started: HPNLMOD Procedure

The most common use of the HPNLMOD procedure is to estimate the parameters in a model in which the response variable is a nonlinear function of one or more of the parameters.

Least Squares Model

The Michaelis-Menten model of enzyme kinetics (Ratkowsky 1990, p. 59) relates a substrate's concentration to its catalyzed reaction rate. The Michaelis-Menten model can be analyzed using a least squares estimation because it does not specify how the reaction rate is distributed around its predicted value. The relationship between reaction rate and substrate concentration is

$$f(\mathbf{x}, \boldsymbol{\theta}) = \frac{\theta_1 x_i}{\theta_2 + x_i}, \quad \text{for } i = 1, 2, \ldots, n$$

where x_i represents the concentration for n trials and $f(\mathbf{x}, \boldsymbol{\theta})$ is the reaction rate. The vector $\boldsymbol{\theta}$ contains the rate parameters.

For this model, which has experimental measurements of reaction rate and concentration stored in the enzyme data set, the following SAS statements estimate the parameters θ_1 and θ_2:

```
proc hpnlmod data=enzyme;
   parms theta1=0 theta2=0;
   model rate ~ residual(theta1*conc / (theta2 + conc));
run;
```

The least squares estimation performed by PROC HPNLMOD for this enzyme kinetics problem produces the analysis of variance table that is displayed in Figure 10.1. The table displays the degrees of freedom, sums of squares, and mean squares along with the model F test.

Figure 10.1 Nonlinear Least Squares Analysis of Variance

The HPNLMOD Procedure

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Approx Pr > F
Model	2	290116	145058	88537.2	<.0001
Error	12	19.6606	1.6384		
Uncorrected Total	14	290135			

An intercept was not specified for this model.

Finally, Figure 10.2 displays the parameter estimates, standard errors, t statistics, and 95% confidence intervals for θ_1 and θ_2.

Figure 10.2 Parameter Estimates and Approximate 95% Confidence Intervals

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Approx Pr > |t|	Approximate 95% Confidence Limits	
theta1	158.1	0.6737	1	234.67	<.0001	156.6	159.6
theta2	0.0741	0.00313	1	23.69	<.0001	0.0673	0.0809

In the enzyme kinetics model, no information was supplied about the distribution of the reaction rate around the model's mean value. Therefore, the residual model distribution was specified to perform a least squares parameter fit.

Binomial Model

In Example 69.3 (*SAS/STAT User's Guide*) cancer remission is modeled by expressing the maximum likelihood function for a binary distribution as a nonlinear least squares optimization. The following statements show an equivalent formulation of this model that uses PROC HPNLMOD and specifies the binary distribution explicitly:

```
proc hpnlmod data=remiss corr;
   parms int=-10 a=-2 b=-1 c=6;
   linp = int + a*cell + b*li + c*temp;
   p = probnorm(linp);
   model remiss ~ binary(1-p);
run;
```

This binary distribution model displays information about the quality of the estimation that is different from the information displayed in the section "Least Squares Model" on page 395. No analysis of variance table is produced for this model; fit statistics that are based on the value of the likelihood function are displayed in Figure 10.3.

Figure 10.3 Nonlinear Likelihood Function Statistics

The HPNLMOD Procedure

Fit Statistics	
-2 Log Likelihood	21.9002
AIC (smaller is better)	29.9002
AICC (smaller is better)	31.7183
BIC (smaller is better)	35.0835

Parameter estimates for the binary distribution model that uses the same quantities as are used in the section "Least Squares Model" on page 395 are displayed in Figure 10.4.

Figure 10.4 Parameter Estimates and Approximate 95% Confidence Intervals

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Approx Pr > |t|	Approximate 95% Confidence Limits	
int	-36.7548	32.3607	1	-1.14	0.2660	-103.2	29.6439
a	-5.6298	4.6376	1	-1.21	0.2353	-15.1454	3.8858
b	-2.2513	0.9790	1	-2.30	0.0294	-4.2599	-0.2426
c	45.1815	34.9095	1	1.29	0.2065	-26.4469	116.8

Syntax: HPNLMOD Procedure

The following statements are available in the HPNLMOD procedure:

PROC HPNLMOD < *options* > **;**
BOUNDS *constraint* < *,. . . ,constraint* > **;**
BY *variables* **;**
ESTIMATE '*label*' *expression* < *options* > **;**
MODEL *dependent-variable* ∼ *distribution* **;**
PARAMETERS < *parameter-specification* > < *,. . . , parameter-specification* > < */ options* > **;**
PERFORMANCE < *performance-options* > **;**
PREDICT '*label*' *expression* < *options* > **;**
RESTRICT *restriction1* < *, restriction2 . . .* > **;**
Programming Statements **;**

The PROC HPNLMOD statement and exactly one MODEL statement are required.

PROC HPNLMOD Statement

PROC HPNLMOD < *options* > **;**

The PROC HPNLMOD statement invokes the procedure. Table 10.1 summarizes important options in the PROC HPNLMOD statement by function. These and other options in the PROC HPNLMOD statement are then described fully in alphabetical order.

Table 10.1 PROC HPNLMOD Statement Options

Option	Description
Basic Options	
DATA=	Specifies the input data set
OUT=	Specifies the output data set
Output Options	
CORR	Specifies the correlation matrix
COV	Specifies the covariance matrix
ECORR	Specifies the correlation matrix of additional estimates
ECOV	Specifies the covariance matrix of additional estimates
DF	Specifies the default degrees of freedom
NOPRINT	Suppresses ODS output
NOITPRINT	Suppresses output about iterations within the optimization process
Optimization Options	
ABSCONV=	Tunes an absolute function convergence criterion
ABSFCONV=	Tunes an absolute difference function convergence criterion
ABSGCONV=	Tunes the absolute gradient convergence criterion
FCONV=	Tunes the relative function convergence criterion
GCONV=	Tunes the relative gradient convergence criterion
MAXITER=	Chooses the maximum number of iterations in any optimization
MAXFUNC=	Specifies the maximum number of function evaluations in any optimization
MAXTIME=	Specifies the upper limit seconds of CPU time for any optimization
MINITER=	Specifies the minimum number of iterations in any optimization
TECHNIQUE=	Selects the optimization technique
Tolerance Options	
SINGULAR=	Tunes the general singularity criterion
User-Defined Format Options	
FMTLIBXML=	Specifies a file reference for a format stream
XMLFORMAT=	Specifies a file name for a format stream

You can specify the following *options* in the PROC HPNLMOD statement.

ABSCONV=*r*

ABSTOL=*r*

specifies an absolute function convergence criterion. For minimization, termination requires $f(\boldsymbol{\psi}^{(k)}) \leq r$, where $\boldsymbol{\psi}$ is the vector of parameters in the optimization and $f(\cdot)$ is the objective function. The default value of r is the negative square root of the largest double-precision value, which serves only as a protection against overflow.

ABSFCONV=*r* **<** *n* **>**

ABSFTOL=*r***<** *n* **>**

specifies an absolute difference function convergence criterion. For all techniques except the Nelder-Mead simplex (NMSIMP) technique, termination requires a small change of the function value in successive iterations:

$$|f(\boldsymbol{\psi}^{(k-1)}) - f(\boldsymbol{\psi}^{(k)})| \leq r$$

Here, $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization, and $f(\cdot)$ is the objective function. The same formula is used for the NMSIMP technique, but $\boldsymbol{\psi}(k)$ is defined as the vertex that has the lowest function value, and $\boldsymbol{\psi}^{(k-1)}$ is defined as the vertex that has the highest function value in the simplex. The default value is $r = 0$. The optional integer value n specifies the number of successive iterations for which the criterion must be satisfied before the process can be terminated.

ABSGCONV=*r* **<** *n* **>**

ABSGTOL=*r***<** *n* **>**

specifies an absolute gradient convergence criterion. Termination requires the maximum absolute gradient element to be small:

$$\max_j |g_j(\boldsymbol{\psi}^{(k)})| \leq r$$

Here, $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization, and $g_j(\cdot)$ is the gradient of the objective function with respect to the jth parameter. This criterion is not used by the NMSIMP technique. The default value is r =1E−5. The optional integer value n specifies the number of successive iterations for which the criterion must be satisfied before the process can be terminated.

ALPHA=α

specifies the level of significance α that is used in the construction of $100(1-\alpha)\%$ confidence intervals. The value must be strictly between 0 and 1; the default value of $\alpha = 0.05$ results in 95% intervals. This value is used as the default confidence level for limits that are computed in the "Parameter Estimates" table and is used in the LOWER and UPPER options in the PREDICT statement.

CORR

requests the approximate correlation matrix for the parameter estimates.

COV

requests the approximate covariance matrix for the parameter estimates.

DATA=*SAS-data-set*

names the SAS data set to be used by PROC HPNLMOD. The default is the most recently created data set.

If PROC HPNLMOD executes in distributed mode, the input data are distributed to memory on the appliance nodes and analyzed in parallel, unless the data are already distributed in the appliance database. In the latter case, PROC HPNLMOD reads the data alongside the distributed database. For more information about the various execution modes, see the section "Processing Modes" on page 10; for more information about the alongside-the-database model, see the section "Alongside-the-Database Execution" on page 18.

DF=*n*

specifies the default number of degrees of freedom to use in the calculation of p-values and confidence limits for additional parameter estimates.

ECORR

requests the approximate correlation matrix for all expressions that are specified in ESTIMATE statements.

ECOV

requests the approximate covariance matrix for all expressions that are specified in ESTIMATE statements.

FCONV=*r*<*n*>

FTOL=*r*<*n*>

specifies a relative function convergence criterion. For all techniques except NMSIMP, termination requires a small relative change of the function value in successive iterations:

$$\frac{|f(\boldsymbol{\psi}^{(k)}) - f(\boldsymbol{\psi}^{(k-1)})|}{|f(\boldsymbol{\psi}^{(k-1)})|} \leq r$$

Here, $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization, and $f(\cdot)$ is the objective function. The same formula is used for the NMSIMP technique, but $\boldsymbol{\psi}^{(k)}$ is defined as the vertex that has the lowest function value, and $\boldsymbol{\psi}^{(k-1)}$ is defined as the vertex that has the highest function value in the simplex. The default is $r = 10^{-\mathrm{FDIGITS}}$, where FDIGITS is by default $-\log_{10}\{\epsilon\}$ and ϵ is the machine precision. The optional integer value n specifies the number of successive iterations for which the criterion must be satisfied before the process can terminate.

FMTLIBXML=*file-ref*

specifies the file reference for the XML stream that contains the user-defined format definitions. User-defined formats are handled differently in a distributed computing environment than they are handled in other SAS products. For information about how to generate a XML stream for your formats, see the section "Working with Formats" on page 35.

GCONV=*r*<*n*>

GTOL=*r*<*n*>

specifies a relative gradient convergence criterion. For all techniques except the conjugate gradient (CONGRA) and NMSIMP techniques, termination requires that the normalized predicted function reduction be small:

$$\frac{\mathbf{g}(\boldsymbol{\psi}^{(k)})'[\mathbf{H}^{(k)}]^{-1}\mathbf{g}(\boldsymbol{\psi}^{(k)})}{|f(\boldsymbol{\psi}^{(k)})|} \leq r$$

Here, $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization, $f(\cdot)$ is the objective function, and $\mathbf{g}(\cdot)$ is the gradient. For the CONGRA technique (where a reliable Hessian estimate $\mathbf{H}$ is not available), the following criterion is used:

$$\frac{\| \mathbf{g}(\boldsymbol{\psi}^{(k)}) \|_2^2 \quad \| \mathbf{s}(\boldsymbol{\psi}^{(k)}) \|_2}{\| \mathbf{g}(\boldsymbol{\psi}^{(k)}) - \mathbf{g}(\boldsymbol{\psi}^{(k-1)}) \|_2 \, |f(\boldsymbol{\psi}^{(k)})|} \leq r$$

This criterion is not used by the NMSIMP technique. The default value is $r =$1E−8. The optional integer value n specifies the number of successive iterations for which the criterion must be satisfied before the process can terminate.

MAXFUNC=*n*

MAXFU=*n*

specifies the maximum number of function calls in the optimization process. The default values are as follows, depending on the optimization technique (which you specify in the TECHNIQUE= option):

- TRUREG, NRRIDG, NEWRAP: $n = 125$
- QUANEW, DBLDOG: $n = 500$
- CONGRA: $n = 1,000$
- NMSIMP: $n = 3,000$

Optimization can terminate only after completing a full iteration. Therefore, the number of function calls that are actually performed can exceed *n*.

MAXITER=*n*

MAXIT=*n*

specifies the maximum number of iterations in the optimization process. The default values are as follows, depending on the optimization technique (which you specify in the TECHNIQUE= option):

- TRUREG, NRRIDG, NEWRAP: $n = 50$
- QUANEW, DBLDOG: $n = 200$
- CONGRA: $n = 400$
- NMSIMP: $n = 1,000$

These default values also apply when *n* is specified as a missing value.

MAXTIME=*r*

specifies an upper limit of *r* seconds of CPU time for the optimization process. The default value is the largest floating-point double representation of your computer. This time that is specified by *r* is checked only once at the end of each iteration. Therefore, the actual running time can be longer than *r*.

MINITER=*n*

MINIT=*n*

specifies the minimum number of iterations. The default value is 0. If you request more iterations than are actually needed for convergence to a stationary point, the optimization algorithms can behave strangely. For example, the effect of rounding errors can prevent the algorithm from continuing for the required number of iterations.

NOITPRINT
suppresses the display of the "Iteration History" table.

NOPRINT
suppresses the generation of ODS output.

OUT=*SAS-data-set*
names the SAS data set to be created when one or more PREDICT statements are specified. A single OUT= data set is created to contain all predicted values when more than one PREDICT statement is specified. An error message is produced if a PREDICT statement is specified and an OUT= data set is not specified. For more information about output data sets in SAS high-performance analytical procedures, see the section "Output Data Sets" on page 34.

SINGULAR=*number*
tunes the general singularity criterion that is applied in sweeps and inversions. The default is 1E4 times the machine epsilon; this product is approximately 1E−12 on most computers.

TECHNIQUE=*keyword*

TECH=*keyword*
specifies the optimization technique for obtaining maximum likelihood estimates. You can choose from the following techniques by specifying the appropriate *keyword*:

CONGRA	performs a conjugate-gradient optimization.
DBLDOG	performs a version of double-dogleg optimization.
LEVMAR	performs a Levenberg-Marquardt optimization.
NEWRAP	performs a Newton-Raphson optimization that combines a line-search algorithm with ridging.
NMSIMP	performs a Nelder-Mead simplex optimization.
NONE	performs no optimization.
NRRIDG	performs a Newton-Raphson optimization with ridging.
QUANEW	performs a quasi-Newton optimization.
TRUREG	performs a trust-region optimization.

The default value is TECHNIQUE=LEVMAR for least squares regression models and TECHNIQUE=NRRIDG for models where the distribution is specified.

XMLFORMAT=*filename*
specifies the file name for the XML stream that contains the user-defined format definitions. User-defined formats are handled differently in a distributed computing environment than they are handled in other SAS products. For information about how to generate a XML stream for your formats, see the section "Working with Formats" on page 35.

BOUNDS Statement

BOUNDS *constraint* < , *constraint . . .* > **;**

where *constraint* represents

< *number operator* > *parameter-list* < *operator number* >

Boundary constraints are specified in a BOUNDS statement. One- or two-sided boundary constraints are allowed. Elements in a list of boundary constraints are separated by commas. For example:

```
bounds 0 <= a1-a9 X <= 1, -1 <= c2-c5;
bounds b1-b10 y >= 0;
```

You can specify more than one BOUNDS statement. If you specify more than one lower (or upper) bound for the same parameter, the maximum (or minimum) of these is taken.

If the maximum l_j of all lower bounds is larger than the minimum of all upper bounds u_j for the same parameter θ_j, the boundary constraint is replaced by $\theta_j := l_j := \min(u_j)$, which is defined by the minimum of all upper bounds specified for θ_j.

BY Statement

BY *variables* **;**

You can specify a BY statement with PROC HPNLMOD to obtain separate analyses of observations in groups that are 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 you specify more than one BY statement, only the last one specified is used.

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 HPNLMOD 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 (in Base SAS software).

BY statement processing is not supported when the HPNLMOD procedure runs alongside the database or alongside the Hadoop distributed file system (HDFS). These modes are used if the input data are stored in a database or HDFS and the grid host is the appliance that houses the data.

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

ESTIMATE Statement

ESTIMATE '*label*' *expression* < *options* > **;**

The ESTIMATE statement enables you to compute an additional estimate that is a function of the parameter values. You must provide a quoted string to identify the estimate and then provide a valid SAS expression. Multiple ESTIMATE statements are permitted, and results from all ESTIMATE statements are listed in a common table. PROC HPNLMOD computes approximate standard errors for the estimates by using the delta method (Billingsley 1986). It uses these standard errors to compute corresponding t statistics, p-values, and confidence limits.

The ECOV option in the PROC HPNLMOD statement produces a table that contains the approximate covariance matrix of all the additional estimates you specify. The ECORR option produces the corresponding correlation matrix.

You can specify the following *options* in the ESTIMATE statement:

ALPHA=α
specifies the alpha level to be used to compute confidence limits. The default value corresponds to the ALPHA= option in the PROC HPNLMOD statement.

DF=d
specifies the degrees of freedom to be used to compute p-values and confidence limits. The default value corresponds to the DF= option in the PROC HPNLMOD statement.

MODEL Statement

MODEL *dependent-variable* ∼ *distribution* **;**

The MODEL statement is the mechanism for either using a distribution specification to specify the distribution of the data or using the RESIDUAL distribution to specify a predicted value. You must specify a single dependent variable from the input data set, a tilde (∼), and then a distribution along with its parameters. You can specify the following values for *distribution*:

RESIDUAL(*m*) or **LS**(*m*) specifies no particular distribution. Instead the sum of squares of the differences between *m* and the dependent variable is minimized.

NORMAL(*m*, *v*) specifies a normal (Gaussian) distribution that has mean *m* and variance *v*.

BINARY(*p*) specifies a binary (Bernoulli) distribution that has probability *p*.

BINOMIAL(*n*, *p*) specifies a binomial distribution that has count *n* and probability *p*.

GAMMA(*a*, *b*) specifies a gamma distribution that has shape *a* and scale *b*.

NEGBIN(*n*, *p*) specifies a negative binomial distribution that has count *n* and probability *p*.

POISSON(*m*) specifies a Poisson distribution that has mean *m*.

GENERAL(*ll*) specifies a general log-likelihood function that you construct by using SAS programming statements.

The MODEL statement must follow any SAS programming statements that you specify for computing parameters of the preceding distributions. For information about the built-in log-likelihood functions, see the section "Built-In Log-Likelihood Functions" on page 412 .

PARAMETERS Statement

PARAMETERS < *parameter-specification* > < *,. . . , parameter-specification* > < */ options* > **;**

PARMS < *parameter-specification* > < *,. . . , parameter-specification* > < */ options* > **;**

The purpose of the PARAMETERS statement is to provide starting values for the HPNLMOD procedure. You can provide values that define a single point in the parameter space or that define a set of points. For more information about *parameter-specification*, see the section "Assigning Starting Values by Using Parameter Specification" on page 406.

You can specify the following *options* in the PARAMETERS statement after the slash (/).

BEST=*i* > 0

specifies the maximum number of parameter grid points and the corresponding objective function values to display in the "Parameters" table. If you specify this option, the parameter grid points are listed in ascending order of objective function value. By default, all parameter grid points are displayed.

PDATA=*SAS-data-set*

DATA=*SAS-data-set*

specifies the data set that provides parameter starting values.

START=*value*

DEFSTART=*value*

specifies a default starting value for all parameters.

There are four methods available for providing starting values to the optimization process. In descending order of precedence, the methods are as follows:

1. Specify values directly in the PARAMETERS statement.
2. Specify values in the PDATA= data set option.
3. Specify a single value for all parameters by using the START= option.
4. Use the default value 1.0.

The names that are assigned to parameters must be valid SAS names and must not coincide with names of variables in the input data set (see the DATA= option in the PROC HPNLMOD statement). Parameters that are assigned starting values through the PARAMETERS statement can be omitted from the estimation if the expression in the MODEL statement does not depend on them.

Assigning Starting Values by Using Parameter Specification

A *parameter-specification* has the following general form, where *name* identifies the parameter and *value-list* provides the set of starting values for the parameter:

name **=** *value-list*

Often the *value-list* contains only a single value, but more general and flexible list specifications such as the following are possible:

m	a single value
m1, *m2*, . . . , *mn*	several values
m TO *n*	a sequence in which *m* equals the starting value, *n* equals the ending value, and the increment equals 1
m TO *n* BY *i*	a sequence in which *m* equals the starting value, *n* equals the ending value, and the increment is *i*
m1, *m2* TO *m3*	mixed values and sequences

When you specify more than one value for a parameter, PROC HPNLMOD sorts the values in ascending sequence and removes duplicate values from the parameter list before forming the grid for the parameter search. If you specify several values for each parameter, PROC HPNLMOD evaluates the model at each point on the grid. The iterations then commence from the point on the grid that yields the smallest objective function value.

For example, the following PARAMETERS statement specifies five parameters and sets their possible starting values as shown in the following table:

```
parms  b0 = 0
       b1 = 4 to 8
       b2 = 0 to .6 by .2
       b3 = 1, 10, 100
       b4 = 0, .5, 1 to 4;
```

Possible Starting Values

B0	B1	B2	B3	B4
0	4	0.0	1	0.0
	5	0.2	10	0.5
	6	0.4	100	1.0
	7	0.6		2.0
	8			3.0
				4.0

The objective function values are calculated for each of the $1 \times 5 \times 4 \times 3 \times 6 = 360$ combinations of possible starting values. Each grid point's objective function value is computed by using the execution mode that is specified in the PERFORMANCE statement.

If you specify a starting value by using a *parameter-specification*, any starting values that are provided for this parameter through the PDATA= data set are ignored. The *parameter-specification* overrides the information in the PDATA= data set.

Assigning Starting Values from a SAS Data Set

The PDATA= option in the PARAMETERS statement enables you to assign starting values for parameters by using a SAS data set. The data set must contain at least two variables: a character variable named Parameter or Parm that identifies the parameter, and a numeric variable named Estimate or Est that contains the starting values. For example, the PDATA= option enables you to use the contents of the "ParameterEstimates" table from one PROC HPNLMOD run to supply starting values for a subsequent run, as follows:

```
proc hpnlmod data=d(obs=30);
   parameters alpha=100 beta=3 gamma=4;
   Switch = 1/(1+gamma*exp(beta*log(dose)));
   model y ~ residual(alpha*Switch);
   ods output ParameterEstimates=pest;
run;

proc hpnlmod data=d;
   parameters / pdata=pest;
   Switch = 1/(1+gamma*exp(beta*log(dose)));
   model y ~ residual(alpha*Switch);
run;
```

You can specify multiple values for a parameter in the PDATA= data set, and the parameters can appear in any order. The starting values are collected by parameter and arranged in ascending order, and duplicate values are removed. The parameter names in the PDATA= data set are not case sensitive. For example, the following DATA step defines starting values for three parameters and a starting grid with $1 \times 3 \times 1 = 3$ points:

```
data test;
   input Parameter $ Estimate;
   datalines;
alpha  100
 BETA   4
 beta   4.1
beta    4.2
beta    4.1
 gamma 30
;
```

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 the procedure.

You can also use the PERFORMANCE statement to control whether PROC HPNLMOD executes in single-machine mode or distributed mode.

The PERFORMANCE statement is documented further in the section "PERFORMANCE Statement" on page 36 of Chapter 3, "Shared Concepts and Topics."

PREDICT Statement

PREDICT '*label*' *expression* < *options* > **;**

PREDICT '*label*' **MEAN** < *options* > **;**

The PREDICT statement enables you to construct predictions of an expression across all of the observations in the input data set. Multiple PREDICT statements are permitted. You must provide a quoted string to identify the predicted expression and then provide the predicted value. You can specify the predicted value either by using a SAS programming expression that involves the input data set variables and parameters or by using the keyword MEAN. If you specify the keyword MEAN, the predicted mean value for the distribution specified in the MODEL statement is used. Predicted values are computed using the final parameter estimates. Standard errors of prediction are computed using the delta method (Billingsley 1986; Cox 1998). Results for all PREDICT statements are placed in the output data set that you specify in the OUT= option in the PROC HPNLMOD statement. For more information, see the section "Output Data Sets" on page 34.

The following *options* are available in the PREDICT statement.

ALPHA=α
specifies the alpha level to be used to compute confidence limits. The default value corresponds to the ALPHA= option in the PROC HPNLMOD statement.

DF=*d*
specifies the degrees of freedom to be used to compute confidence limits. The default value corresponds to the DF= option in the PROC HPNLMOD statement.

LOWER=*name*
specifies a variable that contains the lower confidence limit of the predicted value.

PRED=*name*
specifies a variable that contains the predicted value.

PROBT=*name*
specifies a variable that contains the p-value of the predicted value.

STDERR=*name*
specifies a variable that contains the standard error of the predicted value.

TVALUE=*name*
specifies a variable that contains the t statistic for the predicted value.

UPPER=*name*
specifies a variable that contains the upper confidence limit of the predicted value.

RESTRICT Statement

RESTRICT *restriction1* < *, restriction2 . . .* > **;**

The RESTRICT statement imposes linear restrictions on the model's parameters estimates. You can specify any number of RESTRICT statements.

Each *restriction* is written as an expression, optionally followed by an equality operator (=) or an inequality operator (<, >, <=, >=), followed by a second expression as follows:

expression **<** *operator expression* **>**

The *operator* can be =, <, >, <= , or >=. The operator and second expression are optional. When they are omitted, the *operator* defaults to = and the second *expression* defaults to the value 0.

Restriction expressions can be composed of parameter names, arithmetic operators, functions, and constants. Comparison operators (such as = or <) and logical operators (such as &) cannot be used in RESTRICT statement expressions. Parameters that are named in restriction expressions must be among the parameters that are estimated by the model. Restriction expressions cannot refer to other variables that are defined in the program or the DATA= data set. The restriction expressions must be linear functions of the parameters.

The following example illustrates how to use the RESTRICT statement to impose a linear constraint on parameters:

```
proc hpnlmod;
   parms alpha beta;
   f = (x/alpha + beta)**2
   model y ~ residual(f);
   restrict beta < 2*(alpha + constant('pi'));
run;
```

The preceding RESTRICT statement represents the following model constraint:

$$\beta < 2(\alpha + \pi)$$

Programming Statements

Programming statements define the arguments of the MODEL, ESTIMATE, and PREDICT statements in PROC HPNLMOD. Most of the programming statements that can be used in the SAS DATA step can also be used in the HPNLMOD procedure. See *SAS Language Reference: Concepts* for a description of SAS programming statements. The following are valid programming statements:

ABORT;
CALL *name* **[(** *expression* **[,** *expression* **. . .])];**
DELETE;
DO[*variable* **=** *expression*
[TO *expression***] [BY** *expression***]**
[, *expression* **[TO** *expression***] [BY** *expression* **] . . .]**
]
[WHILE *expression* **] [UNTIL** *expression* **] ;**
END;
GOTO *statement_label***;**
IF *expression***;**
IF *expression* **THEN** *program_statement***;**
ELSE *program_statement***;**
variable **=** *expression***;**
variable **+** *expression***;**
LINK *statement_label***;**
PUT [*variable***] [=] [...];**
RETURN;
SELECT[(*expression***)];**
STOP;
SUBSTR(*variable***,** *index***,** *length* **)=** *expression***;**
WHEN (*expression***)** *program_statement***;**
OTHERWISE *program_statement***;**

For the most part, the SAS programming statements work the same as they do in the SAS DATA step, as documented in *SAS Language Reference: Concepts*. However, they differ as follows:

- The ABORT statement does not allow any arguments.
- The DO statement does not allow a character index variable. Thus, the first of the following statements is supported, but the second is not:

```
do i = 1,2,3;
```

```
do i = 'A','B','C';
```

- In contrast to other procedures that share PROC HPNLMOD's programming syntax, PROC HPNLMOD does not support the LAG function. Because observations are not processed sequentially when high-performance analytical procedures perform the parameter optimization, information for computing lagged values is not available.

- The PUT statement, used mostly for program debugging in PROC HPNLMOD, supports only some of the features of the DATA step PUT statement, and it has some new features that the DATA step PUT statement does not have:
  - The PROC HPNLMOD PUT statement does not support line pointers, factored lists, iteration factors, overprinting, _INFILE_, the colon (:) format modifier, or "$".
  - The PROC HPNLMOD PUT statement supports expressions, but the expression must be enclosed in parentheses. For example, the following statement displays the square root of x:

```
put  (sqrt(x));
```

  - The PROC HPNLMOD PUT statement supports the item _PDV_, which displays a formatted listing of all variables in the program. For example, the following statement displays a much more readable listing of the variables than the _ALL_ print item:

```
put _pdv_;
```

- The WHEN and OTHERWISE statements enable you to specify more than one programming statement. That is, DO/END groups are not necessary for multiple WHEN statements. For example, the following syntax is valid:

```
select;
   when (exp1) stmt1;
               stmt2;
   when (exp2) stmt3;
               stmt4;
end;
```

When you code your programming statements, avoid defining variables that begin with an underscore (_) because they might conflict with internal variables that are created by PROC HPNLMOD. The MODEL statement must come after any SAS programming statements that define or modify terms that are used to specify the model.

Details: HPNLMOD Procedure

Least Squares Estimation

Models that are estimated by PROC HPNLMOD can be represented by using the equations

$$\begin{aligned} \mathbf{Y} &= \mathbf{f}(\boldsymbol{\beta}; \mathbf{z}_1, \cdots, \mathbf{z}_k) + \boldsymbol{\epsilon} \\ \mathrm{E}[\boldsymbol{\epsilon}] &= \mathbf{0} \\ \mathrm{Var}[\boldsymbol{\epsilon}] &= \sigma^2 \mathbf{I} \end{aligned}$$

where

$\mathbf{Y}$	is the $(n \times 1)$ vector of observed responses
$\mathbf{f}$	is the nonlinear prediction function of parameters and regressor variables
$\boldsymbol{\beta}$	is the vector of model parameters to be estimated
$\mathbf{z}_1, \cdots, \mathbf{z}_k$	are the $(n \times 1)$ vectors for each of the k regressor variables
$\boldsymbol{\epsilon}$	is the $(n \times 1)$ vector of residuals
σ^2	is the variance of the residuals

In these models, the distribution of the residuals is not specified and the model parameters are estimated using the least squares method. For the standard errors and confidence limits in the "ParameterEstimates" table to apply, the errors are assumed to be homoscedastic, uncorrelated, and have zero mean.

Built-In Log-Likelihood Functions

For models in which the distribution of model errors is specified, the HPNLMOD procedure estimates parameters by maximizing the value of a log-likelihood function for the specified distribution. The log-likelihood functions used by PROC HPNLMOD for the supported error distributions are as follows:

$Y \sim \mathrm{normal}(m, v)$

$$\begin{aligned} l(m, v; y) &= -\frac{1}{2}\left(\log\{2\pi\} + \frac{(y-m)^2}{v} + \log\{v\}\right) \\ \mathrm{E}[Y] &= m \\ \mathrm{Var}[Y] &= v \\ v &> 0 \end{aligned}$$

$Y \sim \text{binary}(p)$

$$
\begin{aligned}
l_1(p;y) &= \begin{cases} y \ \log\{p\} & y > 0 \\ 0 & \text{otherwise} \end{cases} \\
l_2(p;y) &= \begin{cases} (1-y) \ \log\{1-p\} & y < 1 \\ 0 & \text{otherwise} \end{cases} \\
l(p;y) &= l_1(p;y) + l_2(p;y) \\
\text{E}[Y] &= p \\
\text{Var}[Y] &= p\,(1-p) \\
0 &< p < 1
\end{aligned}
$$

$Y \sim \text{binomial}(n, p)$

$$
\begin{aligned}
l_c &= \log\{\Gamma(n+1)\} - \log\{\Gamma(y+1)\} - \log\{\Gamma(n-y+1)\} \\
l_1(n,p;y) &= \begin{cases} y \ \log\{p\} & y > 0 \\ 0 & \text{otherwise} \end{cases} \\
l_2(n,p;y) &= \begin{cases} (n-y) \ \log\{1-p\} & n-y > 0 \\ 0 & \text{otherwise} \end{cases} \\
l(n,p;y) &= l_c + l_1(n,p;y) + l_2(n,p;y) \\
\text{E}[Y] &= n\,p \\
\text{Var}[Y] &= n\,p\,(1-p) \\
0 &< p < 1
\end{aligned}
$$

$Y \sim \text{gamma}(a, b)$

$$
\begin{aligned}
l(a,b;y) &= -a\log\{b\} - \log\{\Gamma(a)\} + (a-1)\log\{y\} - y/b \\
\text{E}[Y] &= ab \\
\text{Var}[Y] &= ab^2 \\
a &> 0 \\
b &> 0
\end{aligned}
$$

This parameterization of the gamma distribution differs from the parameterization that the GLIMMIX and GENMOD procedures use. The scale parameter in PROC HPNLMOD is expressed as the inverse of the scale parameter that PROC GLIMMIX and PROC GENMOD use. The PROC HPNLMOD parameter represents the scale of the magnitude of the residuals. The scale parameter in PROC GLIMMIX can be estimated by using the following statements:

```
proc glimmix;
   model y = x / dist=gamma s;
run;
```

The following statements show how to use PROC HPNLMOD to estimate the equivalent scale parameter:

```
proc hpnlmod;
   parms b0=1 b1=0 scale=14;
   linp = b0 + b1*x;
   mu   = exp(linp);
   b    = mu*scale;
   model y ~ gamma(1/scale,b);
run;
```

$Y \sim \text{negbin}(n, p)$

$$
\begin{aligned}
l(n, p; y) &= \log\{\Gamma(n + y)\} - \log\{\Gamma(n)\} - \log\{\Gamma(y + 1)\} \\
&\quad + n \log\{p\} + y \log\{1 - p\} \\
\text{E}[Y] &= n\left(\frac{1 - p}{p}\right) \\
\text{Var}[Y] &= n\left(\frac{1 - p}{p^2}\right) \\
n &\geq 0 \\
0 &< p < 1
\end{aligned}
$$

The parameter n can be real-numbered; it does not have to be integer-valued.

$Y \sim \text{Poisson}(m)$

$$
\begin{aligned}
l(m; y) &= y \log\{m\} - m - \log\{\Gamma(y + 1)\} \\
\text{E}[Y] &= m \\
\text{Var}[Y] &= m \\
m &> 0
\end{aligned}
$$

Computational Method

Multithreading

Threading refers to the organization of computational work into multiple tasks (processing units that can be scheduled by the operating system). A task is associated with a thread. Multithreading refers to the concurrent execution of threads. When multithreading is possible, substantial performance gains can be realized compared to sequential (single-threaded) execution.

The number of threads that the HPNLMOD procedure spawns is determined by the number of CPUs on a machine and can be controlled in the following ways:

- You can specify the CPU count by using the CPUCOUNT= SAS system option. For example, if you specify the following statement, the HPNLMOD procedure determines threading as if it executed on a system that has four CPUs, regardless of the actual CPU count:

```
options cpucount=4;
```

- You can specify the NTHREADS= option in the PERFORMANCE statement to determine the number of threads. This specification overrides the CPUCOUNT= system option. Specify NTHREADS=1 to force single-threaded execution.

The number of threads per machine is displayed in the "Performance Information" table, which is part of the default output. The HPNLMOD procedure allocates one thread per CPU.

The HPNLMOD procedure divides the data that are processed on a single machine among the threads—that is, the HPNLMOD procedure implements multithreading by distributing computations across the data. For example, if the input data set has 1,000 observations and PROC HPNLMOD is running with four threads, then 250 observations are associated with each thread. All operations that require access to the data are then multithreaded. These operations include the following:

- calculation of objective function values for the initial parameter grid
- objective function calculation
- gradient calculation
- Hessian calculation
- scoring of observations

In addition, operations on matrices such as sweeps might be multithreaded, provided that the matrices are of sufficient size to realize performance benefits from managing multiple threads for the particular matrix operation.

Choosing an Optimization Algorithm

First- or Second-Order Algorithms

The factors that affect how you choose a particular optimization technique for a particular problem are complex. Occasionally, you might benefit from trying several algorithms.

For many optimization problems, computing the gradient takes more computer time than computing the function value. Computing the Hessian sometimes takes *much* more computer time and memory than computing the gradient, especially when there are many decision variables. Unfortunately, optimization techniques that do not use some kind of Hessian approximation usually require many more iterations than techniques that do use a Hessian matrix; as a result, the total run time of these techniques is often longer. Techniques that do not use the Hessian also tend to be less reliable. For example, they can terminate more easily at stationary points than at global optima.

Table 10.3 shows which derivatives are required for each optimization technique.

Table 10.3 Derivatives Required

Algorithm	First-Order	Second-Order
TRUREG	x	x
NEWRAP	x	x
NRRIDG	x	x
QUANEW	x	-
DBLDOG	x	-
CONGRA	x	-
LEVMAR	x	-
NMSIMP	-	-

The second-derivative methods (TRUREG, NEWRAP, and NRRIDG) are best for small problems for which the Hessian matrix is not expensive to compute. Sometimes the NRRIDG algorithm can be faster than the TRUREG algorithm, but TRUREG can be more stable. The NRRIDG algorithm requires only one matrix with $p(p+1)/2$ double words; TRUREG and NEWRAP require two such matrices. Here, p denotes the number of parameters in the optimization.

The first-derivative methods QUANEW and DBLDOG are best for medium-sized problems for which the objective function and the gradient are much faster to evaluate than the Hessian. In general, the QUANEW and DBLDOG algorithms require more iterations than TRUREG, NRRIDG, and NEWRAP, but each iteration can be much faster. The QUANEW and DBLDOG algorithms require only the gradient to update an approximate Hessian, and they require slightly less memory than TRUREG or NEWRAP.

The first-derivative method CONGRA is best for large problems for which the objective function and the gradient can be computed much faster than the Hessian and for which too much memory is required to store the (approximate) Hessian. In general, the CONGRA algorithm requires more iterations than QUANEW or DBLDOG, but each iteration can be much faster. Because CONGRA requires only a factor of p double-word memory, many large applications can be solved only by CONGRA.

The no-derivative method NMSIMP is best for small problems for which derivatives are not continuous or are very difficult to compute.

The LEVMAR method is appropriate only for least squares optimization problems.

Each optimization method uses one or more convergence criteria that determine when it has converged. An algorithm is considered to have converged when any one of the convergence criteria is satisfied. For example, under the default settings, the QUANEW algorithm converges if ABSGCONV <1E−5, FCONV $< 2 \times \epsilon$, or GCONV <1E−8.

By default, the HPNLMOD procedure applies the NRRIDG algorithm because it can take advantage of multithreading in Hessian computations and inversions. If the number of parameters becomes large, specifying TECHNIQUE=QUANEW (which is a first-order method with good overall properties) is recommended.

Algorithm Descriptions

The following subsections provide details about each optimization technique and follow the same order as Table 10.3.

Trust Region Optimization (TRUREG)

The trust region method uses the gradient $\mathbf{g}(\boldsymbol{\psi}^{(k)})$ and the Hessian matrix $\mathbf{H}(\boldsymbol{\psi}^{(k)})$; thus, it requires that the objective function $f(\boldsymbol{\psi})$ have continuous first- and second-order derivatives inside the feasible region.

The trust region method iteratively optimizes a quadratic approximation to the nonlinear objective function within a hyperelliptic trust region that has radius Δ. Th radius constrains the step size that corresponds to the quality of the quadratic approximation. The trust region method is implemented based on Dennis, Gay, and Welsch (1981); Gay (1983); Moré and Sorensen (1983).

The trust region method performs well for small- to medium-sized problems, and it does not need many function, gradient, and Hessian calls. However, if the computation of the Hessian matrix is computationally expensive, one of the quasi-Newton or conjugate gradient algorithms might be more efficient.

Newton-Raphson Optimization with Line Search (NEWRAP)

The NEWRAP technique uses the gradient $\mathbf{g}(\boldsymbol{\psi}^{(k)})$ and the Hessian matrix $\mathbf{H}(\boldsymbol{\psi}^{(k)})$; thus, it requires that the objective function have continuous first- and second-order derivatives inside the feasible region.

If second-order derivatives are computed efficiently and precisely, the NEWRAP method can perform well for medium-sized to large problems, and it does not need many function, gradient, and Hessian calls.

This algorithm uses a pure Newton step when the Hessian is positive-definite and when the Newton step reduces the value of the objective function successfully. Otherwise, a combination of ridging and line search is performed to compute successful steps. If the Hessian is not positive-definite, a multiple of the identity matrix is added to the Hessian matrix to make it positive-definite (Eskow and Schnabel 1991).

In each iteration, a line search is performed along the search direction to find an approximate optimum of the objective function. The default line-search method uses quadratic interpolation and cubic extrapolation (LIS=2).

Newton-Raphson Ridge Optimization (NRRIDG)

The NRRIDG technique uses the gradient $\mathbf{g}(\boldsymbol{\psi}^{(k)})$ and the Hessian matrix $\mathbf{H}(\boldsymbol{\psi}^{(k)})$; thus, it requires that the objective function have continuous first- and second-order derivatives inside the feasible region.

This algorithm uses a pure Newton step when the Hessian is positive-definite and when the Newton step reduces the value of the objective function successfully. If at least one of these two conditions is not satisfied, a multiple of the identity matrix is added to the Hessian matrix.

The NRRIDG method performs well for small- to medium-sized problems, and it does not require many function, gradient, and Hessian calls. However, if the computation of the Hessian matrix is computationally expensive, one of the quasi-Newton or conjugate gradient algorithms might be more efficient.

Because the NRRIDG technique uses an orthogonal decomposition of the approximate Hessian, each iteration of NRRIDG can be slower than an iteration of the NEWRAP technique, which works with a Cholesky decomposition. However, NRRIDG usually requires fewer iterations than NEWRAP.

Quasi-Newton Optimization (QUANEW)

The (dual) quasi-Newton method uses the gradient $\mathbf{g}(\boldsymbol{\psi}^{(k)})$, and it does not need to compute second-order derivatives because they are approximated. It works well for medium-sized to moderately large optimization problems, where the objective function and the gradient can be computed much faster than the Hessian. However, in general it requires more iterations than the TRUREG, NEWRAP, and NRRIDG techniques, which compute second-order derivatives. QUANEW is the default optimization algorithm because it provides an appropriate balance between the speed and stability that are required for most nonlinear mixed model applications.

The QUANEW technique that is implemented by the HPNLMOD procedure is the dual quasi-Newton algorithm, which updates the Cholesky factor of an approximate Hessian.

In each iteration, a line search is performed along the search direction to find an approximate optimum. The line-search method uses quadratic interpolation and cubic extrapolation to obtain a step size α that satisfies the Goldstein conditions (Fletcher 1987). One of the Goldstein conditions can be violated if the feasible region defines an upper limit of the step size. Violating the left-side Goldstein condition can affect the positive-definiteness of the quasi-Newton update. In that case, either the update is skipped or the iterations are restarted by using an identity matrix, resulting in the steepest descent or ascent search direction.

The QUANEW algorithm uses its own line-search technique.

Double-Dogleg Optimization (DBLDOG)

The double-dogleg optimization method combines the ideas of the quasi-Newton and trust region methods. In each iteration, the double-dogleg algorithm computes the step $\mathbf{s}^{(k)}$ as the linear combination of the steepest descent or ascent search direction $\mathbf{s}_1^{(k)}$ and a quasi-Newton search direction $\mathbf{s}_2^{(k)}$:

$$\mathbf{s}^{(k)} = \alpha_1 \mathbf{s}_1^{(k)} + \alpha_2 \mathbf{s}_2^{(k)}$$

The step is requested to remain within a prespecified trust region radius (Fletcher 1987, p. 107). Thus, the DBLDOG subroutine uses the dual quasi-Newton update but does not perform a line search.

The double-dogleg optimization technique works well for medium-sized to moderately large optimization problems, where the objective function and the gradient are much faster to compute than the Hessian. The implementation is based on Dennis and Mei (1979); Gay (1983), but it is extended for dealing with boundary and linear constraints. The DBLDOG technique generally requires more iterations than the TRUREG, NEWRAP, and NRRIDG techniques, which require second-order derivatives; however, each of the DBLDOG iterations is computationally cheap. Furthermore, the DBLDOG technique requires only gradient calls for the update of the Cholesky factor of an approximate Hessian.

Conjugate Gradient Optimization (CONGRA)

Second-order derivatives are not required by the CONGRA algorithm and are not even approximated. The CONGRA algorithm can be expensive in function and gradient calls, but it requires only $O(p)$ memory for unconstrained optimization. In general, the algorithm must perform many iterations to obtain a precise solution, but each of the CONGRA iterations is computationally cheap.

The CONGRA algorithm should be used for optimization problems that have large p. For the unconstrained or boundary-constrained case, the CONGRA algorithm requires only $O(p)$ bytes of working memory, whereas all other optimization methods require order $O(p^2)$ bytes of working memory. During p successive iterations, uninterrupted by restarts or changes in the working set, the CONGRA algorithm computes a cycle of p conjugate search directions. In each iteration, a line search is performed along the search direction to find an

approximate optimum of the objective function. The default line-search method uses quadratic interpolation and cubic extrapolation to obtain a step size α that satisfies the Goldstein conditions. One of the Goldstein conditions can be violated if the feasible region defines an upper limit for the step size. Other line-search algorithms can be specified with the LIS= option.

Levenberg-Marquardt Optimization (LEVMAR)

The LEVMAR algorithm performs a highly stable optimization; however, for large problems, it consumes more memory and takes longer than the other techniques. The Levenberg-Marquardt optimization technique is a slightly improved variant of the Moré (1978) implementation.

Nelder-Mead Simplex Optimization (NMSIMP)

The Nelder-Mead simplex method does not use any derivatives and does not assume that the objective function has continuous derivatives. The objective function itself needs to be continuous. This technique is quite expensive in the number of function calls, and it might be unable to generate precise results for $p \gg 40$.

The original Nelder-Mead simplex algorithm is implemented and extended to boundary constraints. This algorithm does not compute the objective for infeasible points, but it changes the shape of the simplex by adapting to the nonlinearities of the objective function. This adaptation contributes to an increased speed of convergence. NMSIMP uses a special termination criterion.

Displayed Output

The following sections describe the output that PROC HPNLMOD produces by default. The output is organized into various tables, which are discussed in the order of their appearance.

Performance Information

The "Performance Information" table is produced by default. It displays information about the execution mode. For single-machine mode, the table displays the number of threads used. For distributed mode, the table displays the grid mode (symmetric or asymmetric), the number of compute nodes, and the number of threads per node.

If you specify the DETAILS option in the PERFORMANCE statement, the procedure also produces a "Timing" table in which elapsed times (absolute and relative) for the main tasks of the procedure are displayed.

Specifications

The "Specifications" table displays basic information about the model such as the data source, the dependent variable, the distribution being modeled, and the optimization technique.

Number of Observations

The "Number of Observations" table displays the number of observations that are read from the input data set and the number of observations that are used in the analysis.

Dimensions

The "Dimensions" table displays the number of parameters that are estimated in the model and the number of upper and lower bounds that are imposed on the parameters.

Parameters

The "Parameters" table displays the initial values of parameters that are used to start the estimation process. You can limit this information by specifying the BEST= option in the PARAMETERS statement when you specify a large number of initial parameter value combinations. The parameter combinations and their corresponding objective function values are listed in increasing order of objective function value.

Iteration History

For each iteration of the optimization, the "Iteration History" table displays the number of function evaluations (including gradient and Hessian evaluations), the value of the objective function, the change in the objective function from the previous iteration, and the absolute value of the largest (projected) gradient element.

Convergence Status

The convergence status table is a small ODS table that follows the "Iteration History" table in the default output. In the listing it appears as a message that identifies whether the optimization succeeded and which convergence criterion was met. If the optimization fails, the message indicates the reason for the failure. If you save the convergence status table to an output data set, a numeric Status variable is added that enables you to programmatically assess convergence. The values of the Status variable encode the following:

0	Convergence was achieved or an optimization was not performed because TECHNIQUE=NONE.
1	The objective function could not be improved.
2	Convergence was not achieved because of a user interrupt or because a limit (such as the maximum number of iterations or the maximum number of function evaluations) was reached. To modify these limits, see the MAXITER=, MAXFUNC=, and MAXTIME= options in the PROC HPNLMOD statement.
3	Optimization failed to converge because function or derivative evaluations failed at the starting values or during the iterations or because a feasible point that satisfies the parameter constraints could not be found in the parameter space.

Linear Constraints

The "Linear Constraints" table summarizes the linear constraints that are applied to the model by using the RESTRICT statements. All the constraints that are specified in the model are listed in the "Linear Constraints" table, together with information about whether each constraint represents an inequality or equality condition and whether that constraint is active for the final parameter estimates.

Fit Statistics

The "Fit Statistics" table displays a variety of measures of fit, depending on whether the model was estimated using least squares or maximum likelihood. In both cases, smaller values of the fit statistics indicate better fit.

For least squares estimations, the "Fit Statistics" table displays the sum of squares of errors and the variance of errors.

For maximum likelihood estimations, the table uses the following formulas to display information criteria, where p denotes the number of effective parameters, n denotes the number of observations used, and l is the log likelihood that is evaluated at the converged estimates:

$$\begin{aligned}
\mathrm{AIC} &= -2l + 2p \\
\mathrm{AICC} &= \begin{cases} -2l + 2pn/(n-p-1) & f > p+2 \\ -2l + 2p(p+2) & \text{otherwise} \end{cases} \\
\mathrm{BIC} &= -2l + p\log(f)
\end{aligned}$$

The information criteria values that are displayed in the "Fit Statistics" table are not based on a normalized log-likelihood function.

ANOVA

The "Analysis of Variance" table is displayed only for least squares estimations. The ANOVA table displays the number of degrees of freedom and the sum of squares that are attributed to the model, the error, and the total. The ANOVA table also reports the variance of the model and the errors, the F statistic, and its probability for the model.

Parameter Estimates

The "Parameter Estimates" table displays the parameter estimates, their estimated (asymptotic) standard errors t statistics, and associated p-values for the hypothesis that the parameter is 0. Confidence limits are displayed for each parameter and are based on the value of the ALPHA= option in the PROC HPNLMOD statement.

Additional Estimates

The "Additional Estimates" table displays the same information as the "Parameter Estimates" table for the expressions that appear in the optional ESTIMATE statements. The table is generated when one or more ESTIMATE statements are specified. Because a separate ALPHA= option can be specified for each ESTIMATE statement, the "Additional Estimates" table also includes a column that indicates each confidence interval's corresponding significance level.

Covariance

The "Covariance" table appears when the COV option is specified in the PROC HPNLMOD statement. The "Covariance" table displays a matrix of covariances between each pair of estimated parameters.

Correlation

The "Correlation" table appears when the CORR option is specified in the PROC HPNLMOD statement. The "Correlation" table displays the correlation matrix for the estimated parameters.

Additional Estimates Covariance

The "Covariance of Additional Estimates" table appears when the ECOV option is specified in the PROC HPNLMOD statement. The "Covariance of Additional Estimates" table displays a matrix of covariances between each pair of expressions that are specified in ESTIMATE statements.

Additional Estimates Correlation

The "Correlation of Additional Estimates" table appears when the ECORR option is specified in the PROC HPNLMOD statement. The "Correlation of Additional Estimates" table displays the correlation matrix for the expressions that are specified in ESTIMATE statements.

Procedure Task Timing

If you specify the DETAILS option in the PERFORMANCE statement, the procedure also produces a "Procedure Task Timing" table in which elapsed times (absolute and relative) for the main tasks of the procedure are displayed.

ODS Table Names

Each table that is created by the HPNLMOD procedure has a name associated with it, and you must use this name to refer to the table when you use ODS statements. These names are listed in Table 10.4.

Table 10.4 ODS Tables Produced by PROC HPNLMOD

Table Name	Description	Required Statement and Option
AdditionalEstimates	Functions of estimated parameters and their associated statistics	ESTIMATE statement
ANOVA	Least squares analysis of variance information	RESIDUAL option in the MODEL statement
Constraints	Information about the model's linear constraints	RESTRICT statement
ConvergenceStatus	Optimization success and convergence information	Default output
CorrB	Parameter correlation matrix	CORR option in the PROC HPNLMOD statement
CovB	Parameter covariance matrix	COV option in the PROC HPNLMOD statement
Dimensions	Number of parameters and their bounds	Default output
ECorrB	Additional estimates' correlation matrix	ECORR option in the PROC HPNLMOD statement
ECovB	Additional estimates' covariance matrix	ECOV option in the PROC HPNLMOD statement

Table 10.4 *continued*

Table Name	Description	Required Statement and Option
FitStatistics	Statistics about the quality of the fit	Default output
IterHistory	Optimizer iteration information	Default output
NObs	Number of observations read and used	Default output
ParameterEstimates	Parameter estimates and associated statistics	Default output
Parameters	Initial parameter values	Default output
PerformanceInfo	Information about high-performance computing environment	Default output
Specifications	Basic model characteristics	Default output

Examples: HPNLMOD Procedure

Example 10.1: Segmented Model

Suppose you are interested in fitting a model that consists of two segments that connect in a smooth fashion. For example, the following model states that the mean of Y is a quadratic function in x for values of x less than x_0 and that the mean of Y is constant for values of x greater than x_0:

$$\mathrm{E}[Y|x] = \begin{cases} \alpha + \beta x + \gamma x^2 & \text{if } x < x_0 \\ c & \text{if } x \geq x_0 \end{cases}$$

In this model equation, α, β, and γ are the coefficients of the quadratic segment, and c is the plateau of the mean function. The HPNLMOD procedure can fit such a segmented model even when the join point, x_0, is unknown.

Suppose you also want to impose conditions on the two segments of the model. First, the curve should be continuous—that is, the quadratic and the plateau section need to meet at x_0. Second, the curve should be smooth—that is, the first derivative of the two segments with respect to x needs to coincide at x_0.

The continuity condition requires that

$$c = \mathrm{E}[Y|x_0] = \alpha + \beta x_0 + \gamma x_0^2$$

The smoothness condition requires that

$$\frac{\partial \mathrm{E}[Y|x_0]}{\partial x} = \beta + 2\gamma x_0 \equiv 0$$

If you solve for x_0 and substitute into the expression for c, the two conditions jointly imply that

$$x_0 = -\beta/2\gamma$$
$$c = \alpha - \beta^2/4\gamma$$

Although there are five unknowns, the model contains only three independent parameters. The continuity and smoothness restrictions together completely determine two parameters, given the other three.

The following DATA step creates the SAS data set for this example:

```
data a;
   input y x @@;
   datalines;
.46 1  .47  2 .57  3 .61  4 .62  5 .68  6 .69  7
.78 8  .70  9 .74 10 .77 11 .78 12 .74 13 .80 13
.80 15 .78 16
 ;
```

The following PROC HPNLMOD statements fit this segmented model:

```
proc hpnlmod data=a out=b;
   parms alpha=.45 beta=.05 gamma=-.0025;

   x0 = -.5*beta / gamma;

   if (x < x0) then
        yp = alpha + beta*x  + gamma*x*x;
   else
        yp = alpha + beta*x0 + gamma*x0*x0;

   model y ~ residual(yp);

   estimate 'join point' -beta/2/gamma;
   estimate 'plateau value c' alpha - beta**2/(4*gamma);
   predict 'predicted' yp pred=yp;
   predict 'response' y pred=y;
   predict 'x' x pred=x;
run;
```

The parameters of the model are α, β, and γ. They are represented in the PROC HPNLMOD statements by the variables alpha, beta, and gamma, respectively. In order to model the two segments, a conditional statement assigns the appropriate expression to the mean function, depending on the value of x_0. The ESTIMATE statements compute the values of x_0 and c. The PREDICT statement computes predicted values for plotting and saves them to data set b.

The results from fitting this model are shown in Output 10.1.1 through Output 10.1.3. The iterative optimization converges after six iterations (Output 10.1.1). Output 10.1.2 shows the estimated parameters. Output 10.1.3 indicates that the join point is 12.7477 and the plateau value is 0.7775.

Output 10.1.1 Nonlinear Least Squares Iterative Phase

Quadratic Model with Plateau

The HPNLMOD Procedure

Iteration History

Iteration	Evaluations	Objective Function	Change	Max Gradient
0	5	0.0035144531		7.184063
1	2	0.0007352716	0.00277918	2.145337
2	2	0.0006292751	0.00010600	0.032551
3	2	0.0006291261	0.00000015	0.002952
4	2	0.0006291244	0.00000000	0.000238
5	2	0.0006291244	0.00000000	0.000023
6	2	0.0006291244	0.00000000	2.313E-6

Convergence criterion (GCONV=1E-8) satisfied.

Output 10.1.2 Least Squares Analysis for the Quadratic Model

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Approx Pr > F
Model	2	0.1769	0.0884	114.22	<.0001
Error	13	0.0101	0.000774		
Corrected Total	15	0.1869			

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Approx Pr > |t|	Approximate 95% Confidence Limits	
alpha	0.3921	0.0267	1	14.70	<.0001	0.3345	0.4497
beta	0.0605	0.00842	1	7.18	<.0001	0.0423	0.0787
gamma	-0.00237	0.000551	1	-4.30	0.0009	-0.00356	-0.00118

Output 10.1.3 Additional Estimates for the Quadratic Model

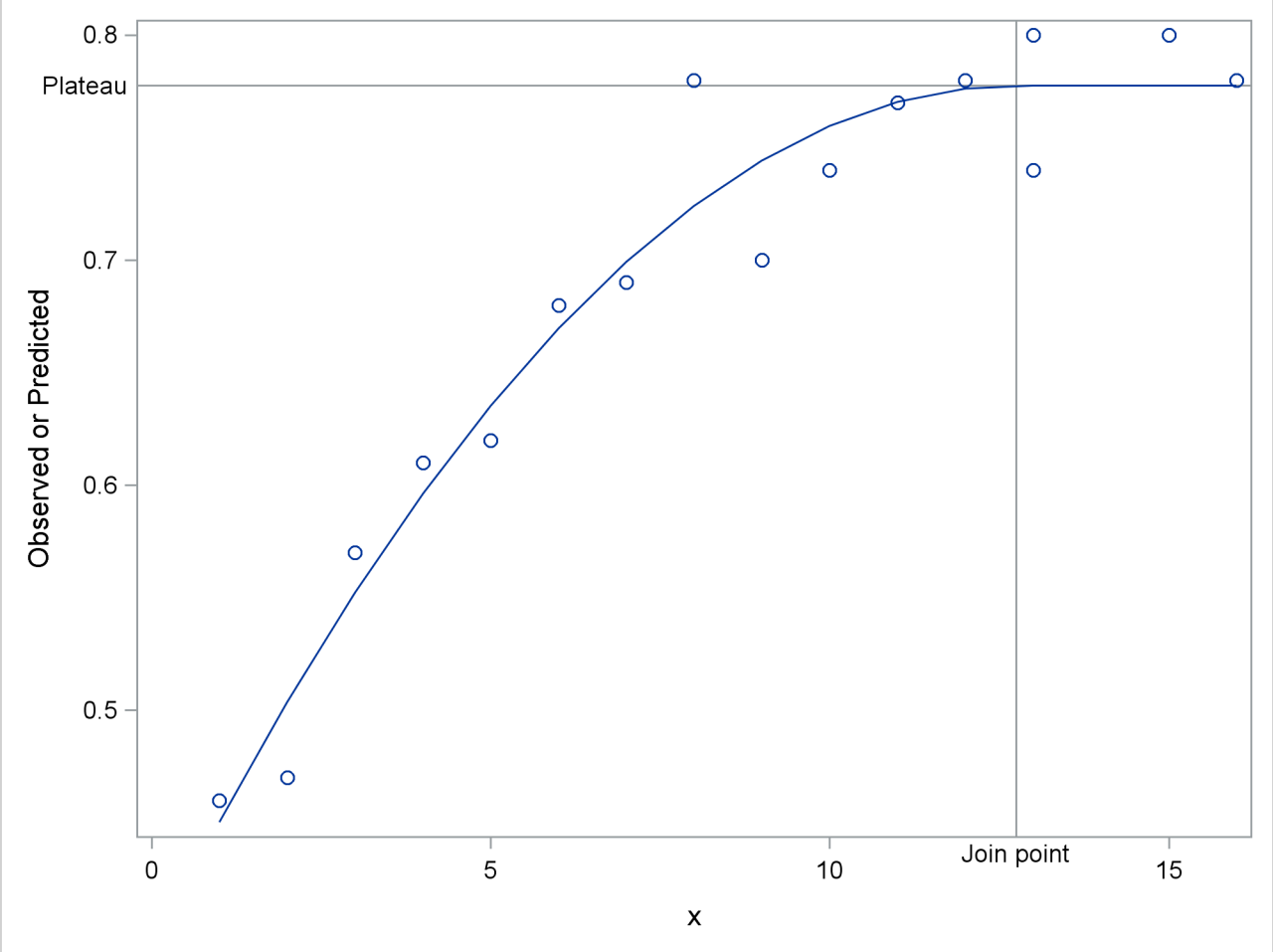
Additional Estimates

Label	Estimate	Standard Error	DF	t Value	Approx Pr > |t|	Alpha	Approximate Confidence Limits	
join point	12.7477	1.2781	1	9.97	<.0001	0.05	9.9864	15.5089
plateau value c	0.7775	0.0123	1	63.11	<.0001	0.05	0.7509	0.8041

The following statements produce a graph of the observed and predicted values along with reference lines for the join point and plateau estimates (Output 10.1.4):

```
proc sgplot data=b noautolegend;
   yaxis label='Observed or Predicted';
   refline 0.7775  / axis=y label="Plateau"    labelpos=min;
   refline 12.7477 / axis=x label="Join point" labelpos=min;
   scatter y=y  x=x;
   series  y=yp x=x;
run;
```

Output 10.1.4 Observed and Predicted Values for the Quadratic Model



References

Billingsley, P. (1986), *Probability and Measure*, 2nd Edition, New York: John Wiley & Sons.

Cox, C. (1998), "Delta Method," in P. Armitage and T. Colton, eds., *Encyclopedia of Biostatistics*, 1125–1127, New York: John Wiley & Sons.

Dennis, J. E., Gay, D. M., and Welsch, R. E. (1981), "An Adaptive Nonlinear Least-Squares Algorithm," *ACM Transactions on Mathematical Software*, 7, 348–368.

Dennis, J. E. and Mei, H. H. W. (1979), "Two New Unconstrained Optimization Algorithms Which Use Function and Gradient Values," *Journal of Optimization Theory and Applications*, 28, 453–482.

Eskow, E. and Schnabel, R. B. (1991), "Algorithm 695: Software for a New Modified Cholesky Factorization," *ACM Transactions on Mathematical Software*, 17, 306–312.

Fletcher, R. (1987), *Practical Methods of Optimization*, 2nd Edition, Chichester, UK: John Wiley & Sons.

Gay, D. M. (1983), "Subroutines for Unconstrained Minimization," *ACM Transactions on Mathematical Software*, 9, 503–524.

Moré, J. J. (1978), "The Levenberg-Marquardt Algorithm: Implementation and Theory," in G. A. Watson, ed., *Lecture Notes in Mathematics*, volume 30, 105–116, Berlin: Springer-Verlag.

Moré, J. J. and Sorensen, D. C. (1983), "Computing a Trust-Region Step," *SIAM Journal on Scientific and Statistical Computing*, 4, 553–572.

Ratkowsky, D. (1990), *Handbook of Nonlinear Regression Models*, New York: Marcel Dekker.

Chapter 11
The HPPLS Procedure

Contents

Overview: HPPLS Procedure **430**
PROC HPPLS Features 430
PROC HPPLS Contrasted with PROC PLS 431
Getting Started: HPPLS Procedure **432**
Spectrometric Calibration 432
Fitting a PLS Model 433
Selecting the Number of Factors by Test Set Validation 435
Predicting New Observations 438
Syntax: HPPLS Procedure **439**
PROC HPPLS Statement 440
BY Statement 443
CLASS Statement 444
ID Statement 444
MODEL Statement 444
OUTPUT Statement 445
PARTITION Statement 447
PERFORMANCE Statement 448
Details: HPPLS Procedure **448**
Regression Methods 448
Partial Least Squares 448
SIMPLS 449
Principal Components Regression 449
Reduced Rank Regression 449
Relationships between Methods 450
Test Set Validation 452
Centering and Scaling 454
Missing Values 454
Computational Method 455
Multithreading 455
Output Data Set 456
Displayed Output 456
Performance Information 456
Data Access Information 456
Centering and Scaling Information 456
Model Information 456
Number of Observations 457

Class Level Information 457
Dimensions 457
Test Set Validation 457
Percent Variation Accounted for by Extracted Factors 457
Model Details 457
Parameter Estimates 458
Timing Information 458
ODS Table Names 458
Examples: HPPLS Procedure **459**
Example 11.1: Choosing a PLS Model by Test Set Validation 459
Example 11.2: Fitting a PLS Model in Single-Machine and Distributed Modes 463
References **466**

Overview: HPPLS Procedure

The HPPLS procedure is a high-performance version of the PLS procedure in SAS/STAT software, which fits models by using any one of a number of linear predictive methods, including *partial least squares* (PLS). Ordinary least squares regression, as implemented in SAS/STAT procedures such as the GLM and REG procedures, has the single goal of minimizing sample response prediction error, and it seeks linear functions of the predictors that explain as much variation in each response as possible. The HPPLS procedure implements techniques that have the additional goal of accounting for variation in the predictors, under the assumption that directions in the predictor space that are well sampled should provide better prediction for *new* observations when the predictors are highly correlated. All the techniques that the HPPLS procedure implements work by extracting successive linear combinations of the predictors, called *factors* (also called *components*, *latent vectors*, or *latent variables*), which optimally address one or both of these two goals: explaining response variation and explaining predictor variation. In particular, the method of partial least squares balances the two objectives by seeking factors that explain both response and predictor variation.

The name "partial least squares" also applies to a more general statistical method that is *not* implemented in this procedure. The partial least squares method was originally developed in the 1960s by the econometrician Herman Wold (1966) for modeling "paths" of causal relation between any number of "blocks" of variables. However, the HPPLS procedure fits only *predictive* partial least squares models that have one "block" of predictors and one "block" of responses. If you are interested in fitting more general path models, you should consider using the CALIS procedure.

PROC HPPLS runs in either single-machine mode or distributed mode.

NOTE: Distributed mode requires SAS High-Performance Statistics.

PROC HPPLS Features

The main features of the HPPLS procedure are as follows:

- supports GLM and reference parameterization for classification effects

- permits any degree of interaction effects that involve classification and continuous variables
- supports partitioning of data into training and testing roles
- supports test set validation to choose the number of extracted factors, where the model is fit to only part of the available data (the training set) and the fit is evaluated over the other part of the data (the test set)
- produces an output data set that contains predicted values and other observationwise statistics

The HPPLS procedure implements the following techniques:

- principal components regression, which extracts factors to explain as much predictor sample variation as possible
- reduced rank regression, which extracts factors to explain as much response variation as possible. This technique, also known as (maximum) redundancy analysis, differs from multivariate linear regression only when there are multiple responses.
- partial least squares regression, which balances the two objectives of explaining response variation and explaining predictor variation. Two different formulations for partial least squares are available: the original predictive method of Wold (1966) and the straightforward implementation of a statistically inspired modification of the partial least squares (SIMPLS) method of De Jong (1993).

Because the HPPLS procedure is a high-performance analytical procedure, it also does the following:

- enables you to run in distributed mode on a cluster of machines that distribute the data and the computations when you license SAS High-Performance Statistics
- enables you to run in single-machine mode on the server where SAS is installed
- exploits all the available cores and concurrent threads, regardless of execution mode

For more information, see the section "Processing Modes" on page 10 in Chapter 3, "Shared Concepts and Topics."

PROC HPPLS Contrasted with PROC PLS

The HPPLS procedure and the PLS procedure have the following similarities and differences:

- All the general factor extraction methods that are available in PROC PLS are supported by PROC HPPLS.
- The RLGW algorithm, which is available in PROC PLS to compute extracted PLS factors, is not supported by PROC HPPLS.
- PROC PLS can specify various methods to be used for cross validation. PROC HPPLS supports test set validation only by using the PARTITION statement.

- The CLASS statement in PROC HPPLS permits two parameterizations: the GLM-type parameterization and a reference parameterization. The HPPLS procedure does not mix parameterizations across the variables in the CLASS statement. In other words, all classification variables are in the same parameterization, and this parameterization is either the GLM or reference parameterization. In PROC PLS, only the GLM-type parameterization is supported.
- The HPPLS procedure does not support the EFFECT statement, the MISSING= option, the VARSCALE option, and the PLOTS option that are available in PROC PLS.
- The syntax of the OUTPUT statement in the HPPLS procedure is different from the syntax of the OUTPUT statement in PROC PLS. In the HPPLS procedure, you do not need to provide a prefix in the OUTPUT statement. A default prefix is used if you do not provide one. If you do not specify any output statistics in the OUTPUT statement in PROC HPPLS, the output data set includes the predicted values for response variables. Furthermore, although the OUTPUT statement in the PLS procedure includes the input and BY variables in the output data by default, PROC HPPLS does not include them by default so that it can avoid data duplication for large data sets. In order to include any input or BY variables in the output data set, you must list these variables in the ID statement.
- The HPPLS procedure is primarily designed to operate in the high-performance distributed environment for large-data tasks. By default, PROC HPPLS performs computations on multiple threads. The PLS procedure executes on a single thread.

Getting Started: HPPLS Procedure

Spectrometric Calibration

The example in this section illustrates basic features of the HPPLS procedure. The data are reported in Umetrics (1995); the original source is Lindberg, Persson, and Wold (1983). Suppose you are researching pollution in the Baltic Sea and you want to use the spectra of samples of seawater to determine the amounts of three compounds present in seawater samples: lignin sulfonate (ls: pulp industry pollution), humic acids (ha: natural forest products), and optical whitener from detergent (dt). Spectrometric calibration is a type of problem in which partial least squares can be very effective. The predictors are the spectra emission intensities at different frequencies in a sample spectrum, and the responses are the amounts of various chemicals in the sample.

For the purposes of calibrating the model, samples that have known compositions are used. The calibration data consist of 16 samples of known concentrations of ls, ha, and dt, with spectra based on 27 frequencies (or, equivalently, wavelengths). The following statements create a SAS data set named Sample for these data. In order to demonstrate the use of test set validation, the data set contains a variable Role that is used to assign observations to the training and testing roles. In this case, the training role has nine samples and the testing role has seven samples.

```
data Sample;
   input obsnam $ v1-v27 ls ha dt Role $5. @@;
   datalines;
EM1   2766 2610 3306 3630 3600 3438 3213 3051 2907 2844 2796
```

```
      2787 2760 2754 2670 2520 2310 2100 1917 1755 1602 1467
      1353 1260 1167 1101 1017         3.0110  0.0000   0.00 TRAIN
EM2   1492 1419 1369 1158  958  887  905  929  920  887  800
       710  617  535  451  368  296  241  190  157  128  106
        89   70   65   56   50         0.0000  0.4005   0.00 TEST
EM3   2450 2379 2400 2055 1689 1355 1109  908  750  673  644
       640  630  618  571  512  440  368  305  247  196  156
       120   98   80   61   50         0.0000  0.0000  90.63 TRAIN
EM4   2751 2883 3492 3570 3282 2937 2634 2370 2187 2070 2007
      1974 1950 1890 1824 1680 1527 1350 1206 1080  984  888
       810  732  669  630  582         1.4820  0.1580  40.00 TEST
EM5   2652 2691 3225 3285 3033 2784 2520 2340 2235 2148 2094
      2049 2007 1917 1800 1650 1464 1299 1140 1020  909  810
       726  657  594  549  507         1.1160  0.4104  30.45 TEST
EM6   3993 4722 6147 6720 6531 5970 5382 4842 4470 4200 4077
      4008 3948 3864 3663 3390 3090 2787 2481 2241 2028 1830
      1680 1533 1440 1314 1227         3.3970  0.3032  50.82 TRAIN
EM7   4032 4350 5430 5763 5490 4974 4452 3990 3690 3474 3357
      3300 3213 3147 3000 2772 2490 2220 1980 1779 1599 1440
      1320 1200 1119 1032  957         2.4280  0.2981  70.59 TRAIN
EM8   4530 5190 6910 7580 7510 6930 6150 5490 4990 4670 4490
      4370 4300 4210 4000 3770 3420 3060 2760 2490 2230 2060
      1860 1700 1590 1490 1380         4.0240  0.1153  89.39 TRAIN
EM9   4077 4410 5460 5857 5607 5097 4605 4170 3864 3708 3588
      3537 3480 3330 3192 2910 2610 2325 2064 1830 1638 1476
      1350 1236 1122 1044  963         2.2750  0.5040  81.75 TEST
EM10  3450 3432 3969 4020 3678 3237 2814 2487 2205 2061 2001
      1965 1947 1890 1776 1635 1452 1278 1128  981  867  753
       663  600  552  507  468         0.9588  0.1450 101.10 TRAIN
EM11  4989 5301 6807 7425 7155 6525 5784 5166 4695 4380 4197
      4131 4077 3972 3777 3531 3168 2835 2517 2244 2004 1809
      1620 1470 1359 1266 1167         3.1900  0.2530 120.00 TRAIN
EM12  5340 5790 7590 8390 8310 7670 6890 6190 5700 5380 5200
      5110 5040 4900 4700 4390 3970 3540 3170 2810 2490 2240
      2060 1870 1700 1590 1470         4.1320  0.5691 117.70 TEST
EM13  3162 3477 4365 4650 4470 4107 3717 3432 3228 3093 3009
      2964 2916 2838 2694 2490 2253 2013 1788 1599 1431 1305
      1194 1077  990  927  855         2.1600  0.4360  27.59 TRAIN
EM14  4380 4695 6018 6510 6342 5760 5151 4596 4200 3948 3807
      3720 3672 3567 3438 3171 2880 2571 2280 2046 1857 1680
      1548 1413 1314 1200 1119         3.0940  0.2471  61.71 TRAIN
EM15  4587 4200 5040 5289 4965 4449 3939 3507 3174 2970 2850
      2814 2748 2670 2529 2328 2088 1851 1641 1431 1284 1134
      1020  918  840  756  714         1.6040  0.2856 108.80 TEST
EM16  4017 4725 6090 6570 6354 5895 5346 4911 4611 4422 4314
      4287 4224 4110 3915 3600 3240 2913 2598 2325 2088 1917
      1734 1587 1452 1356 1257         3.1620  0.7012  60.00 TEST
;
```

Fitting a PLS Model

To isolate a few underlying spectral factors that provide a good predictive model, you can fit a PLS model to the 16 samples by using the following SAS statements:

```
proc hppls data=sample;
   model ls ha dt = v1-v27;
run;
```

By default, the HPPLS procedure extracts at most 15 factors. The default output from this analysis is presented in Figure 11.1 through Figure 11.3.

Figure 11.1 displays the "Performance Information," "Data Access Information," and "Model Information" tables.

The "Performance Information" table shows that PROC HPPLS executes in single-machine mode—that is, the model is fit on the machine where the SAS session executes. This run of the HPPLS procedure was performed on a multicore machine that has four CPUs; one computational thread was spawned per CPU.

The "Data Access Information" table shows that the input data set is accessed with the V9 (base) engine on the client machine where the MVA SAS session executes.

The "Model Information" table identifies the data source and shows that the factor extraction method is partial least squares regression (which is the default) and that the nonlinear iterative partial least squares (NIPALS) algorithm (which is also the default) is used to compute extracted PLS factors.

Figure 11.1 Performance, Data Access, and Model Information

The HPPLS Procedure

Performance Information	
Execution Mode	Single-Machine
Number of Threads	4

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

Model Information	
Data Source	WORK.SAMPLE
Factor Extraction Method	Partial Least Squares
PLS Algorithm	NIPALS
Validation Method	None

Figure 11.2 displays the "Number of Observations" and "Dimensions" tables. The "Number of Observations" table shows that all 16 of the sample observations in the input data are used in the analysis because all samples contain complete data. The "Dimensions" table shows the number of dependent variables, the number of effects, the number of predictor parameters, and the number of factors to extract.

Figure 11.2 Number of Observations and Dimensions

Number of Observations Read	16
Number of Observations Used	16

Figure 11.2 *continued*

Dimensions	
Number of Response Variables	3
Number of Effects	27
Number of Predictor Parameters	27
Number of Factors	15

Figure 11.3 lists the amount of variation, both individual and cumulative, that is accounted for by each of the 15 factors. All the variation in both the predictors and the responses is accounted for by only 15 factors because there are only 16 sample observations. More important, almost all the variation is accounted for with even fewer factors—one or two for the predictors and three to eight for the responses.

Figure 11.3 PLS Variation Summary

Percent Variation Accounted for by Partial Least Squares Factors				
	Model Effects		Dependent Variables	
Number of Extracted Factors	Current	Total	Current	Total
1	97.46068	97.46068	41.91546	41.91546
2	2.18296	99.64365	24.24355	66.15900
3	0.17806	99.82170	24.53393	90.69293
4	0.11973	99.94143	3.78978	94.48271
5	0.04146	99.98289	1.00454	95.48725
6	0.01058	99.99347	2.28084	97.76809
7	0.00168	99.99515	1.16935	98.93744
8	0.00097586	99.99613	0.50410	99.44153
9	0.00142	99.99755	0.12292	99.56446
10	0.00097037	99.99852	0.11027	99.67472
11	0.00032725	99.99884	0.15227	99.82699
12	0.00029338	99.99914	0.12907	99.95606
13	0.00024792	99.99939	0.03121	99.98727
14	0.00042742	99.99981	0.00651	99.99378
15	0.00018639	100.00000	0.00622	100.00000

Selecting the Number of Factors by Test Set Validation

A PLS model is not complete until you choose the number of factors. You can choose the number of factors by using test set validation, in which the data set is divided into two groups called the training data and test data. You fit the model to the training data, and then you check the capability of the model to predict responses for the test data. The predicted residual sum of squares (PRESS) statistic is based on the residuals that are generated by this process.

To select the number of extracted factors by test set validation, you use the PARTITION statement to specify how observations in the input data set are logically divided into two subsets for model training and testing. For example, you can designate a variable in the input data set and a set of formatted values of that variable to determine the role of each observation, as in the following SAS statements:

```
proc hppls data=sample;
   model ls ha dt = v1-v27;
   partition roleVar = Role(train='TRAIN' test='TEST');
run;
```

The resulting output is shown in Figure 11.4 through Figure 11.6.

Figure 11.4 Model Information and Number of Observations with Test Set Validation

The HPPLS Procedure

Model Information	
Data Source	WORK.SAMPLE
Factor Extraction Method	Partial Least Squares
PLS Algorithm	NIPALS
Validation Method	Test Set Validation

Number of Observations Read	16
Number of Observations Used	16
Number of Observations Used for Training	9
Number of Observations Used for Testing	7

Figure 11.5 Test-Set-Validated PRESS Statistics for Number of Factors

The HPPLS Procedure

Test Set Validation for the Number of Extracted Factors	
Number of Extracted Factors	**Root Mean PRESS**
0	1.426362
1	1.276694
2	1.181752
3	0.656999
4	0.43457
5	0.420916
6	0.585031
7	0.576586
8	0.563935
9	0.563935

Minimum Root Mean PRESS	0.420916
Minimizing Number of Factors	5

Figure 11.6 PLS Variation Summary for Test-Set-Validated Model

Percent Variation Accounted for by Partial Least Squares Factors				
	Model Effects		Dependent Variables	
Number of Extracted Factors	Current	Total	Current	Total
1	95.92495	95.92495	37.27071	37.27071
2	3.86407	99.78903	32.38167	69.65238
3	0.10170	99.89073	20.76882	90.42120
4	0.08979	99.98052	4.66666	95.08787
5	0.01142	99.99194	3.88184	98.96971

In Figure 11.4, the "Model Information" table indicates that test set validation is used. The "Number of Observations" table shows that nine sample observations are assigned for training roles and seven are assigned for testing roles.

Figure 11.5 provides details about the results from test set validation. These results show that the absolute minimum PRESS is achieved with five extracted factors. Notice, however, that this is not much smaller than the PRESS for three factors. By using the CVTEST option, you can perform a statistical model comparison that is suggested by van der Voet (1994) to test whether this difference is significant, as shown in the following SAS statements:

```
proc hppls data=sample cvtest(pval=0.15 seed=12345);
   model ls ha dt = v1-v27;
   partition roleVar = Role(train='TRAIN' test='TEST');
run;
```

The model comparison test is based on a rerandomization of the data. By default, the seed for this randomization is based on the system clock, but it is specified here. The resulting output is presented in Figure 11.7 through Figure 11.9.

Figure 11.7 Model Information with Model Comparison Test

The HPPLS Procedure

Model Information	
Data Source	WORK.SAMPLE
Factor Extraction Method	Partial Least Squares
PLS Algorithm	NIPALS
Validation Method	Test Set Validation
Validation Testing Criterion	Prob T**2 > 0.15
Number of Random Permutations	1000
Random Number Seed for Permutation	12345

Figure 11.8 Testing Test Set Validation for Number of Factors

The HPPLS Procedure

Test Set Validation for the Number of Extracted Factors

Number of Extracted Factors	Root Mean PRESS	T**2	Prob > T**2
0	1.426362	5.191629	0.0440
1	1.276694	6.174825	0.0440
2	1.181752	4.60203	0.1020
3	0.656999	3.09999	0.6340
4	0.43457	4.980227	0.7430
5	0.420916	0	1.0000
6	0.585031	2.05496	1.5300
7	0.576586	3.009172	1.9990
8	0.563935	2.416635	2.7550
9	0.563935	2.416635	3.4800

Minimum Root Mean PRESS	0.420916
Minimizing Number of Factors	5
Smallest Number of Factors with p > 0.15	3

Figure 11.9 PLS Variation Summary for Tested Test-Set-Validated Model

Percent Variation Accounted for by Partial Least Squares Factors

	Model Effects		Dependent Variables	
Number of Extracted Factors	Current	Total	Current	Total
1	95.92495	95.92495	37.27071	37.27071
2	3.86407	99.78903	32.38167	69.65238
3	0.10170	99.89073	20.76882	90.42120

The "Model Information" table in Figure 11.7 displays information about the options that are used in the model comparison test. In Figure 11.8, the *p*-value in comparing the test-set validated residuals from models that have five and three factors indicates that the difference between the two models is insignificant; therefore, the model with fewer factors is preferred. The variation summary in Figure 11.9 shows that more than 99% of the predictor variation and more than 90% of the response variation are accounted for by the three factors.

Predicting New Observations

Now that you have chosen a two-factor PLS model for predicting pollutant concentrations that are based on sample spectra, suppose that you have two new samples. The following SAS statements create a data set that contains the spectra for the new samples:

```
data newobs;
   input obsnam $ v1-v27 @@;
   datalines;
EM17  3933 4518 5637 6006 5721 5187 4641 4149 3789
      3579 3447 3381 3327 3234 3078 2832 2571 2274
      2040 1818 1629 1470 1350 1245 1134 1050  987
EM25  2904 2997 3255 3150 2922 2778 2700 2646 2571
      2487 2370 2250 2127 2052 1713 1419 1200  984
       795  648  525  426  351  291  240  204  162
;
```

You can apply the PLS model to these samples to estimate pollutant concentration by appending the new samples to the original 16 and specifying that the predicted values for all 18 be output to a data set, as shown in the following statements:

```
data all;
   set sample newobs;
run;

proc hppls data=all nfac=2;
   model ls ha dt = v1-v27;
   partition roleVar = Role(train='TRAIN' test='TEST');
   output out=result pred=p;
   id obsnam;
run;

proc print data=result;
   where (obsnam in ('EM17','EM25'));
   var obsnam p_ls p_ha p_dt;
run;
```

The ID statement lists the variable obsnam from the input data set that is transferred to the output data set. The new observations are not used in calculating the PLS model because they have no response values. Their predicted concentrations are shown in Figure 11.10.

Figure 11.10 Predicted Concentrations for New Observations

Obs	obsnam	p_ls	p_ha	p_dt
17	EM17	2.63326	0.22343	80.2027
18	EM25	0.69865	0.14308	98.9937

Syntax: HPPLS Procedure

The following statements are available in the HPPLS procedure:

```
PROC HPPLS < options > ;
   BY variables ;
   CLASS variable < (options) > . . . < variable < (options) > > < / global-options > ;
   MODEL response-variables = predictor-effects < / options > ;
   OUTPUT < OUT=SAS-data-set >
          < keyword < =prefix > > . . . < keyword < =prefix > > ;
   PARTITION < partition-options > ;
   PERFORMANCE < performance-options > ;
   ID variables ;
```

The PROC HPPLS statement and a single MODEL statement are required. All other statements are optional. The CLASS statement can appear multiple times. If a CLASS statement is specified, it must precede the MODEL statement. The following sections describe the PROC HPPLS statement and then describe the other statements in alphabetical order.

PROC HPPLS Statement

PROC HPPLS < *options* > ;

The PROC HPPLS statement invokes the HPPLS procedure. Table 11.1 summarizes the options available in the PROC HPPLS statement.

Table 11.1 PROC HPPLS Statement Options

Option	Description
Basic Options	
DATA=	Specifies the input data set
NAMELEN=	Limits the length of effect names
Model Fitting Options	
CVTEST	Requests that van der Voet's (1994) randomization-based model comparison test be performed
METHOD=	Specifies the general factor extraction method to be used
NFAC=	Specifies the number of factors to extract
NOCENTER	Suppresses centering of the responses and predictors before fitting
NOCVSTDIZE	Suppresses re-centering and rescaling of the responses and predictors when cross-validating
NOSCALE	Suppresses scaling of the responses and predictors before fitting
Output Options	
CENSCALE	Displays the centering and scaling information
DETAILS	Displays the details of the fitted model
NOCLPRINT	Limits or suppresses the display of class levels
NOPRINT	Suppresses ODS output
VARSS	Displays the amount of variation accounted for in each response and predictor

The following list provides details about these *options*.

CENSCALE
lists the centering and scaling information for each response and predictor.

CVTEST <(*cvtest-options*)>
requests that van der Voet's (1994) randomization-based model comparison test be performed to test models that have different numbers of extracted factors against the model that minimizes the predicted residual sum of squares. For more information, see the section "Test Set Validation" on page 452. You can also specify the following *cvtest-options* in parentheses after the CVTEST option:

PVAL=*n*
specifies the cutoff probability for declaring an insignificant difference. By default, PVAL=0.10.

STAT=PRESS | T2
specifies the test statistic for the model comparison. You can specify either T2, for Hotelling's T^2 statistic, or PRESS, for the predicted residual sum of squares. By default, STAT=T2.

NSAMP=*n*
specifies the number of randomizations to perform. By default, NSAMP=1000.

SEED=*n*
specifies the seed value for the random number stream. If you do not specify a seed, or if you specify a value less than or equal to 0, the seed is generated from reading the time of day from the computer's clock.

Analyses that use the same (nonzero) seed are not completely reproducible if they are executed on a different number of threads because the random number streams in separate threads are independent. You can control the number of threads on which the HPPLS procedure executes by using SAS system options or by using the PERFORMANCE statement in the HPPLS procedure.

DATA=*SAS-data-set*
names the input SAS data set to be used by PROC HPPLS. The default is the most recently created data set.

If PROC HPPLS executes in distributed mode, the input data are distributed to memory on the appliance nodes and analyzed in parallel, unless the data are already distributed in the appliance database. In that case PROC HPPLS reads the data alongside the distributed database. For more information about the various execution modes, see the section "Processing Modes" on page 10 in Chapter 3, "Shared Concepts and Topics." For more information about the alongside-the-database model, see the section "Alongside-the-Database Execution" on page 18 in Chapter 3, "Shared Concepts and Topics."

DETAILS
lists the details of the fitted model for each successive factor. The listed details are different for different extraction methods. For more information, see the section "Displayed Output" on page 456.

METHOD=PLS<(*PLS-options*)> | SIMPLS | PCR | RRR
specifies the general factor extraction method to be used. You can specify the following values:

PCR
requests principal components regression.

PLS<(*PLS-options*)>
requests partial least squares. You can also specify the following optional *PLS-options* in parentheses after METHOD=PLS:

ALGORITHM=NIPALS | SVD | EIG
names the specific algorithm used to compute extracted PLS factors. NIPALS requests the usual iterative NIPALS algorithm, SVD bases the extraction on the singular value decomposition of $\mathbf{X}'\mathbf{Y}$, and EIG bases the extraction on the eigenvalue decomposition of $\mathbf{Y}'\mathbf{X}\mathbf{X}'\mathbf{Y}$. ALGORITHM=SVD is the most accurate but least efficient approach. By default, ALGORITHM=NIPALS.

EPSILON=*n*
specifies the convergence criterion for the NIPALS algorithm. By default, EPSILON=10^{-12}.

MAXITER=*n*
specifies the maximum number of iterations for the NIPALS algorithm. By default, MAXITER=200.

RRR
requests reduced rank regression.

SIMPLS
requests the straightforward implementation of a statistically inspired modification of the partial least squares (SIMPLS) method of De Jong (1993).

By default, METHOD=PLS.

NAMELEN=*number*
specifies the length to which long effect names are shortened. By default, NAMELEN=20. If you specify a value less than 20 for *number*, the default is used.

NFAC=*n*
specifies the number of factors to extract. The default is $\min\{15, p, N\}$, where p is the number of predictors (or the number of dependent variables when METHOD=RRR) and N is the number of runs (observations). You probably do not need to extract this many factors for most applications. Extracting too many factors can lead to an overfit model (one that matches the training data too well), sacrificing predictive ability. Thus, if you use the default, you should also either specify the PARTITION statement to select the appropriate number of factors for the final model or consider the analysis to be preliminary and examine the results to determine the appropriate number of factors for a subsequent analysis.

NOCENTER
suppresses centering of the responses and predictors before fitting. This option is useful if the analysis variables are already centered and scaled. For more information, see the section "Centering and Scaling" on page 454.

NOCLPRINT<=*number*>
suppresses the display of the "Class Level Information" table if you do not specify *number*. If you specify *number*, the values of the classification variables are displayed only for variables whose number of levels is less than *number*. Specifying a *number* helps to reduce the size of the "Class Level Information" table if some classification variables have a large number of levels.

NOCVSTDIZE
suppresses re-centering and rescaling of the responses and predictors before each model is fit in the cross validation. For more information, see the section "Centering and Scaling" on page 454.

NOPRINT
suppresses the normal display of results. This option is useful when you want only the output statistics saved in a data set. This option temporarily disables the Output Delivery System (ODS). For more information, see Chapter 20, "Using the Output Delivery System" (*SAS/STAT User's Guide*).

NOSCALE
suppresses scaling of the responses and predictors before fitting. This option is useful if the analysis variables are already centered and scaled. For more information, see the section "Centering and Scaling" on page 454.

VARSS
lists, in addition to the average response and predictor sum of squares accounted for by each successive factor, the amount of variation accounted for in each response and predictor.

BY Statement

BY *variables* **;**

You can specify a BY statement with PROC HPPLS to obtain separate analyses of observations in groups that are 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 you specify more than one BY statement, only the last one specified is used.

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 HPPLS 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 (in Base SAS software).

BY-statement processing is not supported when the HPPLS procedure runs alongside the database or alongside the Hadoop Distributed File System (HDFS). Alongside-the-database mode or alongside-HDFS mode is used if the input data are stored in a database or HDFS and the grid host is the appliance that houses the data. For more information about these modes, see Chapter 3, "Shared Concepts and Topics."

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

CLASS Statement

CLASS *variable* <(*options*)>. . . <*variable* <(*options*)>> </ *global-options*> **;**

The CLASS statement names the classification variables to be used as explanatory variables in the analysis. The CLASS statement must precede the MODEL statement.

The CLASS statement for SAS high-performance statistical procedures is documented in the section "CLASS Statement" on page 42 of Chapter 4, "Shared Statistical Concepts." The HPPLS procedure also supports the following *global-option* in the CLASS statement:

UPCASE
uppercases the values of character-valued CLASS variables before levelizing them. For example, if the UPCASE option is in effect and a CLASS variable can take the values 'a', 'A', and 'b', then 'a' and 'A' represent the same level and that CLASS variable is treated as having only two values: 'A' and 'B'.

ID Statement

ID *variables* **;**

The ID statement lists one or more variables from the input data set to be transferred to output data sets that are created by SAS high-performance analytical procedures, provided that the output data set produces one (or more) records per input observation.

For information about the common ID statement in SAS high-performance analytical procedures, see the section "ID Statement" on page 46 in Chapter 4, "Shared Statistical Concepts."

MODEL Statement

MODEL *response-variables* **=** *predictor-effects* </ *options*> **;**

The MODEL statement names the responses and the predictors, which determine the **Y** and **X** matrices of the model, respectively. You can simply list the names of the predictor variables as the model effects, but you can also specify other types of effects, including polynomial effects and interactions. For information about constructing the model effects, see the section "Specification and Parameterization of Model Effects" on page 55, of Chapter 4, "Shared Statistical Concepts."

The MODEL statement is required. You can specify only one MODEL statement.

You can specify the following *options* in the MODEL statement after a slash (/).

INTERCEPT
overrides the default, in which the responses and predictors are centered. When responses and predictors are centered, no intercept is required in the model.

SOLUTION
lists the coefficients of the final predictive model for the responses. The coefficients for predicting the centered and scaled responses that are based on the centered and scaled predictors are displayed, in addition to the coefficients for predicting the raw responses that are based on the raw predictors.

OUTPUT Statement

OUTPUT < **OUT=***SAS-data-set* >
< *keyword* < =*prefix* > >. . . < *keyword* < =*prefix* > > **;**

The OUTPUT statement creates a data set that contains observationwise statistics, which are computed after fitting the model. If you do not specify any *keyword*, then only the predicted values for responses are included.

The variables in the input data set are *not* included in the output data set in order to avoid data duplication for large data sets; however, variables specified in the ID statement are included. If the input data are in distributed form, where accessing data in a particular order cannot be guaranteed, the HPPLS procedure copies the distribution or partition key to the output data set so that its contents can be joined with the input data.

You can specify the following syntax elements in the OUTPUT statement:

OUT=*SAS-data-set*

DATA=*SAS-data-set*

specifies the name of the output data set. If the OUT= (or DATA=) option is omitted, the procedure uses the DATA*n* convention to name the output data set.

keyword **< =***prefix* **>**

specifies a statistic to include in the output data set and optionally a *prefix* for naming the output variables. If you do not provide a *prefix*, the HPPLS procedure assigns a default prefix based on the type of statistic requested. For example, for response variables y1 and y2, a specification of PREDICTED produces two predicted value variables Pred_y1 and Pred_y2.

You can specify the following *keywords* for adding statistics to the OUTPUT data set:

H

requests the approximate leverage. The default prefix is H.

PREDICTED

PRED

P

requests predicted values for each response. The default prefix is Pred.

PRESS

requests approximate predicted residuals for each response. The default prefix is PRESS.

ROLE

requests numeric values that indicate the role played by each observation in fitting the model. The default prefix is _ROLE_. Table 11.2 shows the interpretation of this variable for each observation.

Table 11.2 Role Interpretation

Value	Observation Role
0	Not used
1	Training
2	Testing

If you do not partition the input data by using a PARTITION statement, then the role variable value is 1 for observations that are used in fitting the model, and 0 for observations that have at least one missing or invalid value for the responses or predictors.

STDX
requests standardized (centered and scaled) predictor values for each predictor. The default prefix is StdX.

STDXSSE
requests the sum of squares of residuals for standardized predictors. The default prefix is StdXSSE.

STDY
requests standardized (centered and scaled) response values for each response. The default prefix is StdY.

STDYSSE
requests the sum of squares of residuals for standardized responses. The default prefix is StdYSSE.

TSQUARE
T2
requests scaled sum of squares of score values. The default prefix is TSquare.

XRESIDUAL
XRESID
XR
requests residuals for each predictor. The default prefix is XResid.

XSCORE
requests extracted factors (X-scores, latent vectors, latent variables, T) for each selected model factor. The default prefix is XScore.

YRESIDUAL
YRESID
YR
requests residuals for each response. The default prefix is YResid.

YSCORE
requests extracted responses (Y-scores, U) for each selected model factor. The default prefix is YScore.

According to the *keyword* specified, the output variables that contain the requested statistic are named as follows:

- The *keywords* XRESIDUAL and STDX define an output variable for each predictor, so the variables that correspond to each predictor are named by appending a number (which starts from 1) to the prefix. For each defined variable, a label is also generated automatically; the label contains the prefix of the variable and the name of the predictor. For example, if the model has three predictors, then a specification of XRESIDUAL=XR produces the variables XR1, XR2, and XR3.

- The *keywords* PREDICTED, YRESIDUAL, STDY, and PRESS define an output variable for each response, so the variables that correspond to each response are named by appending the name of the response variable to the prefix. For example, if the model has response variables y1 and y2, then a specification of PREDICTED=P produces the variables P_y1 and P_y2.
- The *keywords* XSCORE and YSCORE define an output variable for each selected model factor, so the variables that correspond to each successive factor are named by appending the factor number to the prefix. For example, if the model has three selected factors, then a specification of XSCORE=T produces the variables T1, T2, and T3.
- The *keywords* H, TSQUARE, STDXSSE, STDYSSE, and ROLE each define a single output variable, so the variable name matches the prefix.

PARTITION Statement

PARTITION < *partition-options* > **;**

The PARTITION statement specifies how observations in the input data set are logically partitioned into disjoint subsets for model training and testing. Either you can designate a variable in the input data set and a set of formatted values of that variable to determine the role of each observation, or you can specify proportions to use for random assignment of observations for each role.

You can specify one (but not both) of the following *partition-options*:

ROLEVAR | ROLE=*variable* **(< TEST='***value***' > < TRAIN='***value***' >)**

names the variable in the input data set whose values are used to assign roles to each observation. The formatted values of this variable that are used to assign observations roles are specified in the TEST= and TRAIN= suboptions. If you specify only the TEST= suboption, then all observations whose role is not determined by the TEST= suboption are assigned to training. If you specify only the TRAIN= suboption, then all observations whose role is not determined by the TRAIN= suboption are assigned to testing.

FRACTION(< TEST=*fraction* **> < SEED=***n* **>)**

requests that specified proportions of the observations in the input data set be randomly assigned training and testing roles. You specify the proportions for testing by using the TEST= suboption; the specified fraction must be less than 1 and the remaining fraction of the observations are assigned to the training role. If you do not specify the TEST= suboption, the default fraction is 0.5. The SEED= suboption specifies an integer that is used to start the pseudorandom number generator for random partitioning of data for training and testing. If you do not specify a seed, or if you specify a value less than or equal to 0, the seed is generated from reading the time of day from the computer's clock.

Because *fraction* is a per-observation probability (which means that any particular observation has a probability of *fraction* of being assigned the testing role), using the FRACTION option can cause different numbers of observations to be assigned training and testing roles. Different partitions can be observed when the number of nodes or threads changes or when PROC HPPLS runs in alongside-the-database mode.

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 the HPPLS procedure.

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

The PERFORMANCE statement is documented further in the section "PERFORMANCE Statement" on page 36.

Details: HPPLS Procedure

Regression Methods

All the predictive methods that PROC HPPLS implements work essentially by finding linear combinations of the predictors (factors) to use to predict the responses linearly. The methods differ only in how the factors are derived, as explained in the following sections.

Partial Least Squares

Partial least squares (PLS) works by extracting one factor at a time. Let $\mathbf{X} = \mathbf{X}_0$ be the centered and scaled matrix of predictors, and let $\mathbf{Y} = \mathbf{Y}_0$ be the centered and scaled matrix of response values. The PLS method starts with a linear combination $\mathbf{t} = \mathbf{X}_0\mathbf{w}$ of the predictors, where $\mathbf{t}$ is called a *score vector* and $\mathbf{w}$ is its associated *weight vector*. The PLS method predicts both $\mathbf{X}_0$ and $\mathbf{Y}_0$ by regression on $\mathbf{t}$:

$$\begin{aligned} \hat{\mathbf{X}}_0 &= \mathbf{t}\mathbf{p}', \quad \text{where} \quad \mathbf{p}' = (\mathbf{t}'\mathbf{t})^{-1}\mathbf{t}'\mathbf{X}_0 \\ \hat{\mathbf{Y}}_0 &= \mathbf{t}\mathbf{c}', \quad \text{where} \quad \mathbf{c}' = (\mathbf{t}'\mathbf{t})^{-1}\mathbf{t}'\mathbf{Y}_0 \end{aligned}$$

The vectors $\mathbf{p}$ and $\mathbf{c}$ are called the X- and Y-*loadings*, respectively.

The specific linear combination $\mathbf{t} = \mathbf{X}_0\mathbf{w}$ is the one that has maximum covariance $\mathbf{t}'\mathbf{u}$ with some response linear combination $\mathbf{u} = \mathbf{Y}_0\mathbf{q}$. Another characterization is that the X-weight, $\mathbf{w}$, and the Y-weight, $\mathbf{q}$, are proportional to the first left and right singular vectors, respectively, of the covariance matrix $\mathbf{X}_0'\mathbf{Y}_0$ or, equivalently, the first eigenvectors of $\mathbf{X}_0'\mathbf{Y}_0\mathbf{Y}_0'\mathbf{X}_0$ and $\mathbf{Y}_0'\mathbf{X}_0\mathbf{X}_0'\mathbf{Y}_0$, respectively.

This accounts for how the first PLS factor is extracted. The second factor is extracted in the same way by replacing $\mathbf{X}_0$ and $\mathbf{Y}_0$ with the X- and Y-residuals from the first factor:

$$\begin{aligned} \mathbf{X}_1 &= \mathbf{X}_0 - \hat{\mathbf{X}}_0 \\ \mathbf{Y}_1 &= \mathbf{Y}_0 - \hat{\mathbf{Y}}_0 \end{aligned}$$

These residuals are also called the *deflated* $\mathbf{X}$ and $\mathbf{Y}$ blocks. The process of extracting a score vector and deflating the data matrices is repeated for as many extracted factors as are wanted.

SIMPLS

Note that each extracted PLS factor is defined in terms of different X-variables $\mathbf{X}_i$. This leads to difficulties in comparing different scores, weights, and so on. The SIMPLS method of De Jong (1993) overcomes these difficulties by computing each score $\mathbf{t}_i = \mathbf{X}\mathbf{r}_i$ in terms of the original (centered and scaled) predictors $\mathbf{X}$. The SIMPLS X-weight vectors r_i are similar to the eigenvectors of $\mathbf{S}\mathbf{S}' = \mathbf{X}'\mathbf{Y}\mathbf{Y}'\mathbf{X}$, but they satisfy a different orthogonality condition. The $\mathbf{r}_1$ vector is just the first eigenvector $\mathbf{e}_1$ (so that the first SIMPLS score is the same as the first PLS score). However, the second eigenvector maximizes

$$\mathbf{e}_1'\mathbf{S}\mathbf{S}'\mathbf{e}_2 \text{ subject to } \mathbf{e}_1'\mathbf{e}_2 = 0$$

whereas the second SIMPLS weight $\mathbf{r}_2$ maximizes

$$\mathbf{r}_1' S S' \mathbf{r}_2 \text{ subject to } \mathbf{r}_1'\mathbf{X}'\mathbf{X}\mathbf{r}_2 = \mathbf{t}_1'\mathbf{t}_2 = 0$$

The SIMPLS scores are identical to the PLS scores for one response but slightly different for more than one response; see De Jong (1993) for details. The X- and Y-loadings are defined as in PLS, but because the scores are all defined in terms of $\mathbf{X}$, it is easy to compute the overall model coefficients $\mathbf{B}$:

$$\begin{aligned}
\hat{\mathbf{Y}} &= \sum_i \mathbf{t}_i \mathbf{c}_i' \\
&= \sum_i \mathbf{X}\mathbf{r}_i \mathbf{c}_i' \\
&= \mathbf{X}\mathbf{B}, \text{ where } \mathbf{B} = \mathbf{R}\mathbf{C}'
\end{aligned}$$

Principal Components Regression

Like the SIMPLS method, principal component regression (PCR) defines all the scores in terms of the original (centered and scaled) predictors $\mathbf{X}$. However, unlike both the PLS and SIMPLS methods, the PCR method chooses the X-weights and X-scores without regard to the response data. The X-scores are chosen to explain as much variation in $\mathbf{X}$ as possible; equivalently, the X-weights for the PCR method are the eigenvectors of the predictor covariance matrix $\mathbf{X}'\mathbf{X}$. Again, the X- and Y-loadings are defined as in PLS; but, as in SIMPLS, it is easy to compute overall model coefficients for the original (centered and scaled) responses $\mathbf{Y}$ in terms of the original predictors $\mathbf{X}$.

Reduced Rank Regression

As discussed in the preceding sections, partial least squares depends on selecting factors $\mathbf{t} = \mathbf{X}\mathbf{w}$ of the predictors and $\mathbf{u} = \mathbf{Y}\mathbf{q}$ of the responses that have maximum covariance, whereas principal components regression effectively ignores $\mathbf{u}$ and selects $\mathbf{t}$ to have maximum variance, subject to orthogonality constraints. In contrast, reduced rank regression selects $\mathbf{u}$ to account for as much variation in the *predicted* responses as possible, effectively ignoring the predictors for the purposes of factor extraction. In reduced rank regression, the Y-weights $\mathbf{q}_i$ are the eigenvectors of the covariance matrix $\hat{\mathbf{Y}}_{\mathrm{LS}}'\hat{\mathbf{Y}}_{\mathrm{LS}}$ of the responses that are predicted by ordinary least squares regression, and the X-scores are the projections of the Y-scores $\mathbf{Y}\mathbf{q}_i$ onto the X space.

Relationships between Methods

When you develop a predictive model, it is important to consider not only the explanatory power of the model for current responses, but also how well sampled the predictive functions are, because the sampling affects how well the model can extrapolate to future observations. All the techniques that the HPPLS procedure implements work by extracting successive factors (linear combinations of the predictors) that optimally address one or both of these two goals: explaining response variation and explaining predictor variation. In particular, principal components regression selects factors that explain as much predictor variation as possible, reduced rank regression selects factors that explain as much response variation as possible, and partial least squares balances the two objectives, seeking factors that explain both response and predictor variation.

To see the relationships between these methods, consider how each one extracts a single factor from the following artificial data set, which consists of two predictors and one response:

```
data data;
   input x1 x2 y;
   datalines;
    3.37651  2.30716        0.75615
    0.74193 -0.88845        1.15285
    4.18747  2.17373        1.42392
    0.96097  0.57301        0.27433
   -1.11161 -0.75225       -0.25410
   -1.38029 -1.31343       -0.04728
    1.28153 -0.13751        1.00341
   -1.39242 -2.03615        0.45518
    0.63741  0.06183        0.40699
   -2.52533 -1.23726       -0.91080
    2.44277  3.61077       -0.82590
;

proc hppls data=data nfac=1 method=rrr;
   model y = x1 x2;
run;

proc hppls data=data nfac=1 method=pcr;
   model y = x1 x2;
run;

proc hppls data=data nfac=1 method=pls;
   model y = x1 x2;
run;
```

The amount of model and response variation that are explained by the first factor for each method is shown in Figure 11.11 through Figure 11.13.

Figure 11.11 Variation Explained by the First Reduced Rank Regression Factor

The HPPLS Procedure

Percent Variation Accounted for by Reduced Rank Regression Factors

	Model Effects		Dependent Variables	
Number of Extracted Factors	Current	Total	Current	Total
1	15.06605	15.06605	100.00000	100.00000

Figure 11.12 Variation Explained by the First Principal Components Regression Factor

The HPPLS Procedure

Percent Variation Accounted for by Principal Components

	Model Effects		Dependent Variables	
Number of Extracted Factors	Current	Total	Current	Total
1	92.99959	92.99959	9.37874	9.37874

Figure 11.13 Variation Explained by the First Partial Least Squares Regression Factor

The HPPLS Procedure

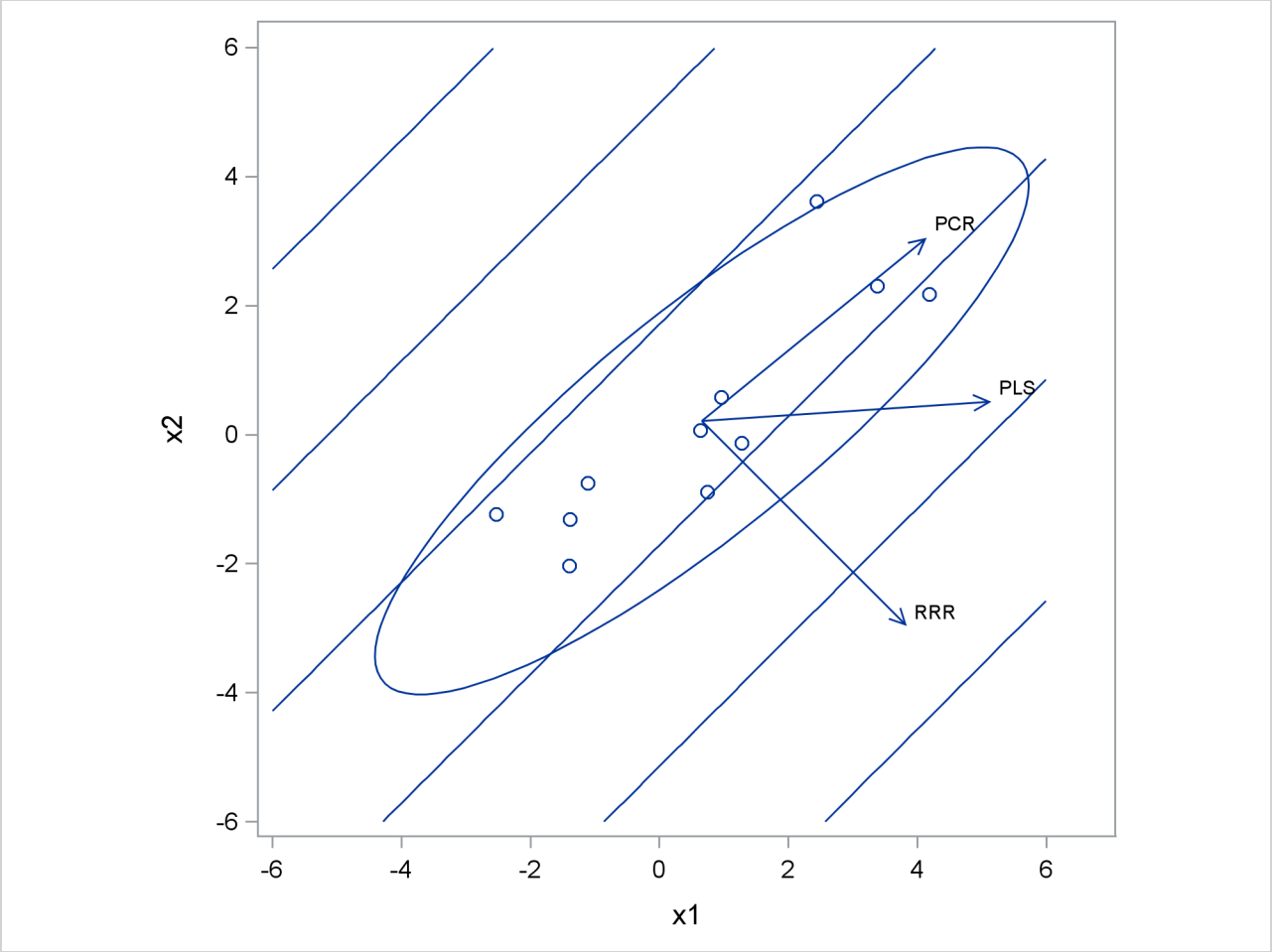
Percent Variation Accounted for by Partial Least Squares Factors

	Model Effects		Dependent Variables	
Number of Extracted Factors	Current	Total	Current	Total
1	88.53567	88.53567	26.53038	26.53038

Notice that although the first reduced rank regression factor explains *all* of the response variation, it accounts for only about 15% of the predictor variation. In contrast, the first principal component regression factor accounts for most of the predictor variation (93%) but only 9% of the response variation. The first partial least squares factor accounts for only slightly less predictor variation than principal components but about three times as much response variation.

Figure 11.14 illustrates how partial least squares balances the goals of explaining response and predictor variation in this case.

Figure 11.14 Depiction of the First Factors for Three Different Regression Methods



The ellipse shows the general shape of the 11 observations in the predictor space, with the contours of increasing y overlaid. Also shown are the directions of the first factor for each of the three methods. Notice that although the predictors vary most in the x1 = x2 direction, the response changes most in the orthogonal x1 = –x2 direction. This explains why the first principal component accounts for little variation in the response and why the first reduced rank regression factor accounts for little variation in the predictors. The direction of the first partial least squares factor represents a compromise between the other two directions.

Test Set Validation

None of the regression methods that the HPPLS procedure implements fit the observed data any better than ordinary least squares (OLS) regression; in fact, all the methods approach OLS as more factors are extracted. Basing the model on more extracted factors improves the model fit to the observed data, but extracting too many factors can cause *overfitting*—that is, tailoring the model too much to the current data to the detriment of future predictions. So the crucial point is that when there are many predictors, OLS can *overfit* the observed data; biased regression methods that use fewer extracted factors can provide better predictability of *future* observations. However, as the preceding observations imply, the quality of the observed data fit cannot be

used to choose the number of factors to extract; the number of extracted factors must be chosen on the basis of how well the model fits observations that are not involved in the modeling procedure itself.

The method of choosing the number of extracted factors that PROC HPPLS implements is called *test set validation*. When you have sufficient data, you can subdivide your data into two parts: training data and test data. During the validation process, the model is fit on the training data, and the predicted residual sum of squares (PRESS) for models that have different numbers of extracted factors is found by using the test data. The number of factors chosen is usually the one that minimizes PRESS.

You use a PARTITION statement to logically subdivide the DATA= data set into separate roles. You can name the fractions of the data that you want to reserve as training data and test data. For example, the following statements randomly subdivide the inData data set, reserving 50% each for training and testing:

```
proc hppls data=inData;
   partition fraction(test=0.5);
   ...
run;
```

In some cases you might need to exercise more control over the partitioning of the input data set. You can do this by naming both a variable in the input data set and a formatted value of that variable for each role. For example, the following statements assign roles to the observations in the inData data set based on the value of the variable Group in that data set. Observations whose value of Group is `'group 1'` are assigned for training, and those whose value is `'group 2'` are assigned to testing. All other observations are ignored.

```
proc hppls data=inData;
   partition roleVar=Group(train='group 1' test='group 2')
   ...
run;
```

By default, the number of extracted factors is chosen to be the one that minimizes PRESS. However, models that have fewer factors often have PRESS statistics that are only marginally larger than the absolute minimum. To address this, van der Voet (1994) proposed a statistical test for comparing the predicted residuals from different models; when you apply van der Voet's test, the number of factors chosen is the fewest while still producing residuals that are insignificantly larger than the residuals of the model that has a minimum PRESS.

To see how van der Voet's test works, let $R_{i,jk}$ be the jth predicted residual for response k for the model that has i extracted factors. Then, the PRESS statistic is $\sum_{jk} R^2_{i,jk}$. Also, let $i_{\min}$ be the number of factors for which PRESS is minimized. The critical value for van der Voet's test is based on the differences between squared predicted residuals:

$$D_{i,jk} = R^2_{i,jk} - R^2_{i_{\min},jk}$$

One alternative for the critical value is $C_i = \sum_{jk} D_{i,jk}$, which is simply the difference between the PRESS statistics for i and $i_{\min}$ factors; alternatively, van der Voet suggests Hotelling's T^2 statistic $C_i = \mathbf{d}'_{i,.}\mathbf{S}_i^{-1}\mathbf{d}_{i,.}$, where $\mathbf{d}_{i,.}$ is the sum of the vectors $\mathbf{d}_{i,j} = \{D_{i,j1}, \ldots, D_{i,jN_y}\}'$ and $\mathbf{S}_i$ is the sum of squares and crossproducts matrix,

$$\mathbf{S}_i = \sum_j \mathbf{d}_{i,j}\mathbf{d}'_{i,j}$$

Virtually, the significance level for van der Voet's test is obtained by comparing C_i with the distribution of values that result from randomly exchanging $R^2_{i,jk}$ and $R^2_{i_{\min},jk}$. In practice, a Monte Carlo sample of such values is simulated and the significance level is approximated as the proportion of simulated critical values that are greater than C_i. If you apply van der Voet's test by specifying the CVTEST option, then, by default, the number of extracted factors that are chosen is the least number of factors that have an approximate significance level that is greater than 0.10.

Centering and Scaling

By default, the predictors and the responses are centered and scaled to have mean 0 and standard deviation 1. Centering the predictors and the responses ensures that the criterion for choosing successive factors is based on how much *variation* they explain in either the predictors or the responses or in both. (For more information about how different methods explain variation, see the section "Regression Methods" on page 448.) Without centering, both the mean variable value and the variation around that mean are involved in selecting factors. Scaling serves to place all predictors and responses on an equal footing relative to their variation in the data. For example, if Time and Temp are two of the predictors, then scaling says that a change of $\mathrm{std}(\mathsf{Time})$ in Time is approximately equivalent to a change of $\mathrm{std}(\mathsf{Temp})$ in Temp.

Usually, both the predictors and responses should be centered and scaled. However, if their values already represent variation around a nominal or target value, then you can use the NOCENTER option in the PROC HPPLS statement to suppress centering. Likewise, if the predictors or responses are already all on comparable scales, then you can use the NOSCALE option to suppress scaling.

If the predictors involve crossproduct terms, PROC HPPLS does not standardize the variables before it standardizes the crossproduct. That is, if the *i*th values of two predictors are denoted x^1_i and x^2_i, then the default standardized *i*th value of the crossproduct is

$$\frac{x^1_i x^2_i - \mathrm{mean}_j(x^1_j x^2_j)}{\mathrm{std}_j(x^1_j x^2_j)}$$

When test set validation is performed for the number of effects, some practitioners disagree as to whether the training data should be retransformed. By default, PROC HPPLS does retransform the training data, but you can suppress this behavior by specifying the NOCVSTDIZE option in the PROC HPPLS statement.

Missing Values

Observations that have any missing independent variables (including all classification variables) are excluded from the analysis, and no predictions are computed for such observations. Observations that have no missing independent variables but do have missing dependent variables are also excluded from the analysis, but predictions are computed. If you use the PARTITION statement and specify the ROLEVAR= option, observations that contain the missing ROLEVAR= variable are excluded from the analysis, but predictions are computed for them.

Computational Method

Multithreading

Threading refers to the organization of computational work into multiple tasks (processing units that can be scheduled by the operating system). A task is associated with a thread. Multithreading refers to the concurrent execution of threads. When multithreading is possible, substantial performance gains can be realized compared to sequential (single-threaded) execution.

The number of threads that the HPPLS procedure spawns is determined by the number of CPUs on a machine and can be controlled in the following ways:

- You can specify the CPU count by using the CPUCOUNT= SAS system option. For example, if you specify the following statements, the HPPLS procedure schedules threads as if it executed on a system that has four CPUs, regardless of the actual CPU count.

```
options cpucount=4;
```

- You can specify the NTHREADS= option in the PERFORMANCE statement to determine the number of threads. This specification overrides the system option. Specify NTHREADS=1 to force single-threaded execution.

The number of threads per machine is displayed in the "Performance Information" table, which is part of the default output. The HPPLS procedure allocates one thread per CPU.

The tasks that the HPPLS procedure multithreads are primarily defined by dividing the data that are processed on a single machine among the threads—that is, the HPPLS procedure implements multithreading through a data-parallel model. For example, if the input data set has 1,000 observations and PROC HPPLS runs on four threads, then 250 observations are associated with each thread. All operations that require access to the data are then multithreaded. These operations include the following:

- variable levelization
- effect levelization
- formation of the crossproducts matrix
- computation of loadings, weights, scores, generalized inverse, and residual sums of squares
- scoring of observations

In addition, operations on matrices such as sweeps might be multithreaded if the matrices are of sufficient size to realize performance benefits from managing multiple threads for the particular matrix operation.

Output Data Set

When an observationwise output data set is created, many procedures in SAS software add the variables from the input data set to the output data set. High-performance statistical procedures assume that the input data sets can be large and contain many variables. For performance reasons, the output data set contains only the following:

- variables that are explicitly created by the statement
- variables that are listed in the ID statement
- distribution keys or hash keys that are transferred from the input data set

Including these variables and keys enables you to add output data set information that is necessary for subsequent SQL joins without copying the entire input data set to the output data set. For more information about output data sets that are produced when PROC HPPLS runs in distributed mode, see the section "Output Data Sets" on page 34 in Chapter 3, "Shared Concepts and Topics."

Displayed Output

The following sections describe the output that PROC HPPLS produces. The output is organized into various tables, which are discussed in the order of their appearance.

Performance Information

The "Performance Information" table is produced by default. It displays information about the execution mode. For single-machine mode, the table displays the number of threads used. For distributed mode, the table displays the grid mode (symmetric or asymmetric), the number of compute nodes, and the number of threads per node.

Data Access Information

The "Data Access Information" table is produced by default. For the input and output data sets, it displays the libref and data set name, the engine used to access the data, the role (input or output) of the data set, and the path that data followed to reach the computation.

Centering and Scaling Information

If you specify the CENSCALE option in the PROC HPPLS statement, the HPPLS procedure produces "Model Effect Centering and Scaling" and "Dependent Variable Centering and Scaling" tables, which display the centering and scaling information for each response and predictor.

Model Information

The "Model Information" table displays basic information about the model, such as the input data set, the factor extraction method, the validation method, and the type of parameterization used for classification variables that are named in the CLASS statement. If you use the PARTITION statement, the table also displays

the random number seed for partition, the validation testing criterion, the number of random permutations, and the random number seed for permutation, depending on whether you specify the FRACTION option in the PARTITION statement and the CVTEST option in the PROC HPPLS statement.

Number of Observations

The "Number of Observations" table displays the number of observations that are read from the input data set and the number of observations that are used in the analysis. If you use a PARTITION statement, the table also displays the number of observations that are used for each data role.

Class Level Information

The "Class Level Information" table lists the levels of every variable that is specified in the CLASS statement. You should check this information to make sure that the data are correct. You can adjust the order of the CLASS variable levels by specifying the ORDER= option in the CLASS statement. You can suppress the "Class Level Information" table completely or partially by specifying the NOCLPRINT= option in the PROC HPPLS statement.

If the classification variables are in the reference parameterization, the "Class Level Information" table also displays the reference value for each variable.

Dimensions

The "Dimensions" table displays information about the number of response variables, the number of effects, and the number of predictor parameters. It also displays the number of factors to extract.

Test Set Validation

If you use the PARTITION statement to perform a test set validation for choosing the number of extracted factors, the HPPLS procedure produces a "Test Set Validation Residual Summary" table to display a residual summary of the validation for each number of factors. It also produces a "Test Set Validation Results" table to display information about the optimal number of factors.

Percent Variation Accounted for by Extracted Factors

By default, the HPPLS procedure produces the "Percent Variation Accounted for by Extracted Factors" table to display just the amount of predictor variation and response variation that are accounted for by each factor. If you specify the VARSS option in the PROC HPPLS statement, the HPPLS procedure also produces the "Model Effect Percent Variation Accounted for by Extracted Factors" table and the "Dependent Variable Percent Variation Accounted for by Extracted Factors" table to display the amount of variation that is accounted for in each response and predictor, in addition to the average response and predictor sum of squares that are accounted for by each successive factor.

Model Details

If you specify the DETAILS option in the PROC HPPLS statement, the HPPLS procedure produces tables to display details about the fitted model for each successive factor. These tables include the following:

- "Model Effect Loadings" table, which displays the predictor loadings

- "Model Effect Weights" table, which displays predictor weights
- "Dependent Variable Weights" table, which displays the response weights
- "Coded Regression Coefficients" tables, which display the coded regression coefficients, if you specify METHOD=SIMPLS, METHOD=PCR, or METHOD=RRR in the PROC HPPLS statement.

Parameter Estimates

If you specify the SOLUTION option in the MODEL statement, the HPPLS procedure produces a "Parameter Estimates" table to display the coefficients of the final predictive model for the responses. The coefficients for predicting the centered and scaled responses based on the centered and scaled predictors are displayed, in addition to the coefficients for predicting the raw responses based on the raw predictors.

Timing Information

If you specify the DETAILS option in the PERFORMANCE statement, the HPPLS procedure produces a "Timing" table, which displays the elapsed time (absolute and relative) of each main task of the procedure.

ODS Table Names

PROC HPPLS assigns a name to each table that it creates. You can use these names to refer to the ODS table when you use the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in Table 11.3. For more information about ODS, see Chapter 20, "Using the Output Delivery System" (*SAS/STAT User's Guide*).

Table 11.3 ODS Tables Produced by PROC HPPLS

Table Name	Description	Required Statement and Option
CVResults	Results of test set validation	PARTITITON statement
CenScaleParms	Parameter estimates for centered and scaled data	SOLUTION option in MODEL statement
ClassLevels	Level information from the CLASS statement	CLASS statement
CodedCoef	Coded regression coefficients	DETAILS option in PROC HPPLS statement
DataAccessInfo	Information about modes of data access	Default output
Dimensions	Model dimensions	Default output
ModelInfo	Model information	Default output
NObs	Number of observations read and used	Default output
ParameterEstimates	Parameter estimates for raw data	SOLUTION option in MODEL statement
PercentVariation	Predictor and response variation that are accounted for by each factor	Default output

Table 11.3 *continued*

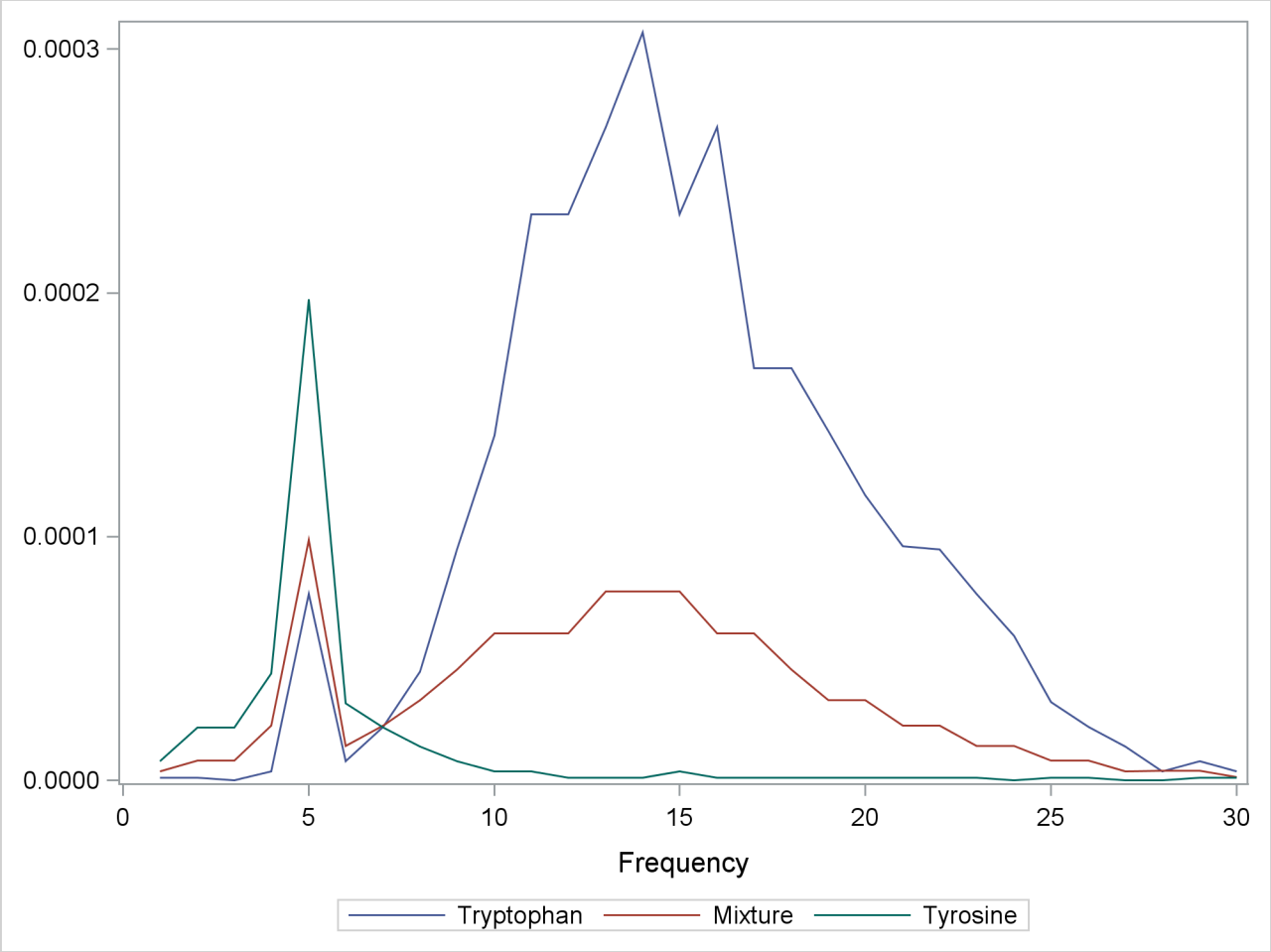
Table Name	Description	Required Statement and Option
PerformanceInfo	Information about the high-performance computing environment	Default output
ResidualSummary	Residual summary from test set validation	PARTITION statement
Timing	Absolute and relative times of tasks that are performed by the procedure	DETAILS option in PERFORMANCE statement
XEffectCenScale	Centering and scaling information for predictor effects	CENSCALE option in PROC HPPLS statement
XLoadings	Loadings for predictor effects	DETAILS option in PROC HPPLS statement
XPercentVariation	Variation that is accounted for by each factor for predictor effects	VARSS option in PROC HPPLS statement
XWeights	Weights for predictor effects	DETAILS option in PROC HPPLS statement
YPercentVariation	Variation that is accounted for by each factor for responses	VARSS option in PROC HPPLS statement
YVariableCenScale	Centering and scaling information for responses	CENSCALE option in PROC HPPLS statement
YWeights	Weights for responses	DETAILS option in PROC HPPLS statement

Examples: HPPLS Procedure

Example 11.1: Choosing a PLS Model by Test Set Validation

This example demonstrates issues in spectrometric calibration. The data (Umetrics 1995) consist of spectrographic readings on 33 samples that contain known concentrations of two amino acids, tyrosine and tryptophan. The spectra are measured at 30 frequencies across the overall range of frequencies. For example, Output 11.1.1 shows the observed spectra for three samples: one with only tryptophan, one with only tyrosine, and one with a mixture of the two, all at a total concentration of 10^{-6}.

Output 11.1.1 Spectra for Three Samples of Tyrosine and Tryptophan



Of the 33 samples, 18 are used as a training set and 15 as a test set. The data originally appear in McAvoy et al. (1989).

These data were created in a lab, where the concentrations are fixed in order to provide a wide range of applicability for the model. You want to use a linear function of the logarithms of the spectra to predict the logarithms of tyrosine and tryptophan concentration, in addition to the logarithm of the total concentration. Actually, because zeros are possible in both the responses and the predictors, slightly different transformations are used. The following statements create a SAS data set named ex1Data for these data. The data set also contains a variable Role that is used to assign samples to the training and testing roles.

```
data ex1Data;
   input obsnam $ Role : $5. tot tyr f1-f30 @@;
   try = tot - tyr;
   if (tyr) then tyr_log = log10(tyr); else tyr_log = -8;
   if (try) then try_log = log10(try); else try_log = -8;
   tot_log = log10(tot);
   datalines;
17mix35 TRAIN 0.00003 0
 -6.215 -5.809 -5.114 -3.963 -2.897 -2.269 -1.675 -1.235
 -0.900 -0.659 -0.497 -0.395 -0.335 -0.315 -0.333 -0.377
```

```
 -0.453 -0.549 -0.658 -0.797 -0.878 -0.954 -1.060 -1.266
 -1.520 -1.804 -2.044 -2.269 -2.496 -2.714
19mix35 TRAIN 0.00003 3E-7
 -5.516 -5.294 -4.823 -3.858 -2.827 -2.249 -1.683 -1.218
 -0.907 -0.658 -0.501 -0.400 -0.345 -0.323 -0.342 -0.387
 -0.461 -0.554 -0.665 -0.803 -0.887 -0.960 -1.072 -1.272
 -1.541 -1.814 -2.058 -2.289 -2.496 -2.712
21mix35 TRAIN 0.00003 7.5E-7
 -5.519 -5.294 -4.501 -3.863 -2.827 -2.280 -1.716 -1.262
 -0.939 -0.694 -0.536 -0.444 -0.384 -0.369 -0.377 -0.421
 -0.495 -0.596 -0.706 -0.824 -0.917 -0.988 -1.103 -1.294
 -1.565 -1.841 -2.084 -2.320 -2.521 -2.729
23mix35 TRAIN 0.00003 1.5E-6

   ... more lines ...

26tyro5 TEST 0.00001 0.00001
 -3.037 -2.696 -2.464 -2.321 -2.239 -2.444 -2.602 -2.823
 -3.144 -3.396 -3.742 -4.063 -4.398 -4.699 -4.893 -5.138
 -5.140 -5.461 -5.463 -5.945 -5.461 -5.138 -5.140 -5.138
 -5.138 -5.463 -5.461 -5.461 -5.461 -5.461
tyro2   TEST 0.0001 0.0001
 -1.081 -0.710 -0.470 -0.337 -0.327 -0.433 -0.602 -0.841
 -1.119 -1.423 -1.750 -2.121 -2.449 -2.818 -3.110 -3.467
 -3.781 -4.029 -4.241 -4.366 -4.501 -4.366 -4.501 -4.501
 -4.668 -4.668 -4.865 -4.865 -5.109 -5.111
;
```

The following statements fit a PLS model that has 10 factors.

```
proc hppls data=ex1Data nfac=10;
   model tot_log tyr_log try_log = f1-f30;
run;
```

The "Model Information" table in Output 11.1.2 shows that no validation method is used. The "Number of Observations" table confirms that all 33 sample observations are used in the analysis.

The table in Output 11.1.3 indicates that only four or five factors are required to explain almost all of the variation in both the predictors and the responses.

Output 11.1.2 Model Information and Number of Observations

The HPPLS Procedure

Model Information	
Data Source	WORK.EX1DATA
Factor Extraction Method	Partial Least Squares
PLS Algorithm	NIPALS
Validation Method	None

Number of Observations Read	33
Number of Observations Used	33

Output 11.1.3 Amount of Variation Explained

Percent Variation Accounted for by Partial Least Squares Factors				
	Model Effects		Dependent Variables	
Number of Extracted Factors	Current	Total	Current	Total
1	77.67903	77.67903	47.80217	47.80217
2	20.62719	98.30622	38.96826	86.77043
3	1.00143	99.30766	6.89262	93.66305
4	0.24930	99.55696	1.77222	95.43528
5	0.13077	99.68773	1.71762	97.15290
6	0.08970	99.77742	0.58619	97.73909
7	0.05684	99.83426	0.29079	98.02988
8	0.06730	99.90156	0.13857	98.16845
9	0.01521	99.91676	0.68214	98.85059
10	0.02627	99.94304	0.14388	98.99447

In order to choose the optimal number of PLS factors, you can explore how well models that are based on data in training roles and have different numbers of factors fit the data in testing roles. To do so, you can use the PARTITION statement to assign observations to training and testing roles based on the values of the input variable named Role.

```
proc hppls data=ex1Data nfac=10 cvtest(stat=press seed=12345);
   model tot_log tyr_log try_log = f1-f30;
   partition roleVar = Role(train='TRAIN' test='TEST');
run;
```

Output 11.1.4 shows the "Model Information" table and the "Number of Observations" table. The "Model Information" table indicates that test set validation is used and displays information about the options that are used in the model comparison test. The "Number of Observations" table confirms that there are 18 observations for the training role and 15 for the testing role.

Output 11.1.5 displays the results of the test set validation. They indicate that although five PLS factors produce the minimum predicted residual sum of squares, the residuals for four factors are insignificantly different from the residuals for five factors. Thus, the smaller model is preferred.

Output 11.1.4 Model Information and Number of Observations with Test Set Validation

The HPPLS Procedure

Model Information	
Data Source	WORK.EX1DATA
Factor Extraction Method	Partial Least Squares
PLS Algorithm	NIPALS
Validation Method	Test Set Validation
Validation Testing Criterion	Prob PRESS > 0.1
Number of Random Permutations	1000
Random Number Seed for Permutation	12345

Output 11.1.4 *continued*

Number of Observations Read	33
Number of Observations Used	33
Number of Observations Used for Training	18
Number of Observations Used for Testing	15

Output 11.1.5 Test Set Validation for the Number of PLS Factors

The HPPLS Procedure

Test Set Validation for the Number of Extracted Factors

Number of Extracted Factors	Root Mean PRESS	Prob > PRESS
0	3.056797	<.0001
1	2.630561	<.0001
2	1.00706	0.0060
3	0.664603	0.0070
4	0.521578	0.3510
5	0.500034	1.0000
6	0.513561	0.8430
7	0.501431	1.5360
8	1.055791	1.6920
9	1.435085	1.7840
10	1.720389	1.8200

Minimum Root Mean PRESS	0.500034
Minimizing Number of Factors	5
Smallest Number of Factors with p > 0.1	4

Percent Variation Accounted for by Partial Least Squares Factors

	Model Effects		Dependent Variables	
Number of Extracted Factors	Current	Total	Current	Total
1	81.16545	81.16545	48.33854	48.33854
2	16.81131	97.97676	32.54654	80.88508
3	1.76391	99.74067	11.44380	92.32888
4	0.19507	99.93574	3.83631	96.16519

Example 11.2: Fitting a PLS Model in Single-Machine and Distributed Modes

This example shows how you can run PROC HPPLS in single-machine and distributed modes. For more information about the execution modes of SAS high-performance analytical procedures, see the section "Processing Modes" on page 10 in Chapter 3, "Shared Concepts and Topics." The focus of this example is to show how you can switch the modes of execution in PROC HPPLS. The following DATA step generates the

data:

```
data ex2Data;
   drop i j k sign n n1 n2 n3 n4;

   n  = 100000;
   n1 = n*0.1;
   n2 = n*0.25;
   n3 = n*0.45;
   n4 = n*0.7;

   array y{10};
   array x{100};

   do i=1 to n;
      do j=1 to dim(y);
         y{j} = 1;
      end;
      sign = 1;

      do j=1 to dim(x);
         x{j} = ranuni(1);
         do k=1 to dim(y);
            y{k} = y{k} + sign*j*x{j};
            sign = -sign;
         end;
      end;

      do j=1 to dim(y);
         y{j} = y{j} + 7*rannor(1);
      end;

      if      i <= n1 then z='verytiny';
      else if i <= n2 then z='small';
      else if i <= n3 then z='medium';
      else if i <= n4 then z='large';
      else                 z='huge';

      output;
   end;
run;
```

The following statements use PROC HPPLS to fit a PLS model by using the SIMPLS method and test set validation:

```
proc hppls data=ex2Data method=simpls cvtest(stat=press seed=12345);
   class z;
   model y: = x: z:;
   partition fraction(test=0.4 seed=67890);
   performance details;
run;
```

In this example, any particular observation has a 40% probability of being assigned the testing role. All nonassigned observations are in training roles.

Output 11.2.1 shows the "Performance Information" table. This table shows that the HPPLS procedure executes in single-machine mode on four threads (the client machine has four CPUs). You can force a certain number of threads on any machine to be involved in the computations by specifying the NTHREADS= option in the PERFORMANCE statement.

Output 11.2.1 Performance Information in Single-Machine Mode

The HPPLS Procedure

Performance Information	
Execution Mode	Single-Machine
Number of Threads	4

Output 11.2.2 shows timing information for the PROC HPPLS run. This table is produced when you specify the DETAILS option in the PERFORMANCE statement. You can see that, in this case, the majority of time is spent fitting a PLS model.

Output 11.2.2 Timing in Single-Machine Mode

Procedure Task Timing		
Task	**Seconds**	**Percent**
Reading and Levelizing Data	1.16	1.00%
Fitting Model	114.37	99.00%

To switch to running PROC HPPLS in distributed mode, specify valid values for the NODES=, INSTALL=, and HOST= options in the PERFORMANCE statement. An alternative to specifying the INSTALL= and HOST= options in the PERFORMANCE statement is to use the OPTIONS SET commands to set appropriate values for the GRIDHOST and GRIDINSTALLLOC environment variables. For information about setting these options or environment variables, see the section "Processing Modes" on page 10 in Chapter 3, "Shared Concepts and Topics."

NOTE: Distributed mode requires SAS High-Performance Statistics.

The following statements provide an example. To run these statements successfully, you need to set the macro variables GRIDHOST and GRIDINSTALLLOC to resolve to appropriate values, or you can replace the references to macro variables with appropriate values.

```
proc hppls data=ex2Data method=simpls cvtest(stat=press seed=12345);
   class z;
   model y: = x: z:;
   partition fraction(test=0.4 seed=67890);
   performance details nodes = 4
               host="&GRIDHOST" install="&GRIDINSTALLLOC";
run;
```

The execution mode in the "Performance Information" table shown in Output 11.2.3 indicates that the calculations were performed in a distributed environment that uses four nodes, each of which uses 32 threads.

Output 11.2.3 Performance Information in Distributed Mode

Performance Information	
Host Node	<< your grid host >>
Install Location	<< your grid install location >>
Execution Mode	Distributed
Number of Compute Nodes	4
Number of Threads per Node	32

Another indication of distributed execution is the following message, which is issued by all high-performance analytical procedures (with the corresponding procedure name) in the SAS log:

```
NOTE: The HPPLS procedure is executing in the distributed
      computing environment with 4 worker nodes.
```

Output 11.2.4 shows timing information for this distributed run of the HPPLS procedure. The majority of time in the distributed mode run is also spent fitting a model.

Output 11.2.4 Timing in Distributed Mode

Procedure Task Timing		
Task	**Seconds**	**Percent**
Distributing Data	0.85	16.48%
Reading and Levelizing Data	0.04	0.85%
Fitting Model	4.12	80.21%
Waiting on Client	0.13	2.45%

References

De Jong, S. (1993), "SIMPLS: An Alternative Approach to Partial Least Squares Regression," *Chemometrics and Intelligent Laboratory Systems*, 18, 251–263.

De Jong, S. and Kiers, H. (1992), "Principal Covariates Regression," *Chemometrics and Intelligent Laboratory Systems*, 14, 155–164.

Dijkstra, T. K. (1983), "Some Comments on Maximum Likelihood and Partial Least Squares Methods," *Journal of Econometrics*, 22, 67–90.

Dijkstra, T. K. (1985), *Latent Variables in Linear Stochastic Models: Reflections on Maximum Likelihood and Partial Least Squares Methods*, 2nd Edition, Amsterdam: Sociometric Research Foundation.

Frank, I. and Friedman, J. (1993), "A Statistical View of Some Chemometrics Regression Tools," *Technometrics*, 35, 109–135.

Geladi, P. and Kowalski, B. (1986), "Partial Least-Squares Regression: A Tutorial," *Analytica Chimica Acta*, 185, 1–17.

Haykin, S. (1994), *Neural Networks: A Comprehensive Foundation*, New York: Macmillan.

Helland, I. S. (1988), "On the Structure of Partial Least Squares Regression," *Communications in Statistics—Simulation and Computation*, 17, 581–607.

Hoerl, A. and Kennard, R. (1970), "Ridge Regression: Biased Estimation for Non-orthogonal Problems," *Technometrics*, 12, 55–67.

Lindberg, W., Persson, J.-A., and Wold, S. (1983), "Partial Least-Squares Method for Spectrofluorimetric Analysis of Mixtures of Humic Acid and Ligninsulfonate," *Analytical Chemistry*, 55, 643–648.

McAvoy, T. J., Wang, N. S., Naidu, S., Bhat, N., Gunter, J., and Simmons, M. (1989), "Interpreting Biosensor Data via Backpropagation," *International Joint Conference on Neural Networks*, 1, 227–233.

Naes, T. and Martens, H. (1985), "Comparison of Prediction Methods for Multicollinear Data," *Communications in Statistics—Simulation and Computation*, 14, 545–576.

Ränner, S., Lindgren, F., Geladi, P., and Wold, S. (1994), "A PLS Kernel Algorithm for Data Sets with Many Variables and Fewer Objects," *Journal of Chemometrics*, 8, 111–125.

Sarle, W. S. (1994), "Neural Networks and Statistical Models," in *Proceedings of the Nineteenth Annual SAS Users Group International Conference*, Cary, NC: SAS Institute Inc.

Shao, J. (1993), "Linear Model Selection by Cross-Validation," *Journal of the American Statistical Association*, 88, 486–494.

Tobias, R. D. (1995), "An Introduction to Partial Least Squares Regression," in *Proceedings of the Twentieth Annual SAS Users Group International Conference*, Cary, NC: SAS Institute Inc.

Umetrics (1995), *Multivariate Analysis*, three-day course, Winchester, MA: Umetrics.

van den Wollenberg, A. L. (1977), "Redundancy Analysis: An Alternative to Canonical Correlation Analysis," *Psychometrika*, 42, 207–219.

van der Voet, H. (1994), "Comparing the Predictive Accuracy of Models Using a Simple Randomization Test," *Chemometrics and Intelligent Laboratory Systems*, 25, 313–323.

Wold, H. (1966), "Estimation of Principal Components and Related Models by Iterative Least Squares," in P. R. Krishnaiah, ed., *Multivariate Analysis*, New York: Academic Press.

Chapter 12
The HPPRINCOMP Procedure

Contents

Overview: HPPRINCOMP Procedure **470**
PROC HPPRINCOMP Features 470
PROC HPPRINCOMP Contrasted with PROC PRINCOMP 471
Getting Started: HPPRINCOMP Procedure **471**
Syntax: HPPRINCOMP Procedure **475**
PROC HPPRINCOMP Statement 476
BY Statement 479
CODE Statement 479
FREQ Statement 480
ID Statement 480
PARTIAL Statement 480
PERFORMANCE Statement 480
VAR Statement 481
WEIGHT Statement 481
Details: HPPRINCOMP Procedure **481**
Missing Values 481
Output Data Sets 481
OUT= Data Set 481
OUTSTAT= Data Set 482
Computational Method 484
Multithreading 484
Displayed Output 485
Performance Information 485
Data Access Information 485
Number of Observations 485
Number of Variables 485
Simple Statistics 485
Correlation Matrix 485
Regression Statistics 485
Regression Coefficients 485
Partial Correlation Matrix 486
Total Variance 486
Eigenvalues 486
Eigenvectors 486
Timing Information 486
ODS Table Names 486

Examples: HPPRINCOMP Procedure **487**
Example 12.1: Analyzing Mean Temperatures of US Cities 487
Example 12.2: Computing Principal Components in Single-Machine and Distributed Modes 489
References **491**

Overview: HPPRINCOMP Procedure

The HPPRINCOMP procedure is a high-performance procedure that performs principal component analysis. It is a high-performance version of the PRINCOMP procedure in SAS/STAT software. PROC HPPRINCOMP accepts raw data as input and can create output data sets that contain eigenvalues, eigenvectors, and standardized or unstandardized principal component scores.

Principal component analysis is a multivariate technique for examining relationships among several quantitative variables. The choice between using factor analysis and using principal component analysis depends in part on your research objectives. You should use the HPPRINCOMP procedure if you are interested in summarizing data and detecting linear relationships. You can use principal component analysis to reduce the number of variables in regression, clustering, and so on.

Principal component analysis was originated by Pearson (1901) and later developed by Hotelling (1933). The application of principal components is discussed by Rao (1964); Cooley and Lohnes (1971); Gnanadesikan (1977). Excellent statistical treatments of principal components are found in Kshirsagar (1972); Morrison (1976); Mardia, Kent, and Bibby (1979).

If you have a data set that contains p numeric variables, you can compute p principal components. Each principal component is a linear combination of the original variables, with coefficients equal to the eigenvectors of the correlation or covariance matrix. The eigenvectors are usually taken with unit length. The principal components are sorted by descending order of the eigenvalues, which are equal to the variances of the components.

PROC HPPRINCOMP runs in either single-machine mode or distributed mode.

NOTE: Distributed mode requires SAS High-Performance Statistics.

PROC HPPRINCOMP Features

The main features of the HPPRINCOMP procedure are as follows:

- can perform analysis on a massively parallel SAS high-performance appliance
- reads input data in parallel and writes output data in parallel when the data source is the appliance database
- is highly multithreaded during calculations of the sum-of-squares-and-crossproducts (SSCP) matrix and the principal component scores
- supports a PARTIAL statement for analyzing a partial correlation or covariance matrix

- supports a FREQ statement for grouped analysis
- supports a WEIGHT statement for weighted analysis
- produces an output data set that contains principal component scores
- produces an output data set that contains means, standard deviations, number of observations, correlations or covariances, eigenvalues, and eigenvectors

PROC HPPRINCOMP Contrasted with PROC PRINCOMP

The HPPRINCOMP procedure and the PRINCOMP procedure in SAS/STAT have the following similarities and differences:

- All statements that are available in PROC PRINCOMP are supported by the HPPRINCOMP procedure.
- PROC PRINCOMP can accept ordinary SAS data set and other types of special SAS data sets as input. The HPPRINCOMP procedure can accept only the ordinary SAS data set (raw data) as input.
- This release of the HPPRINCOMP procedure does not support the PLOTS option that is available in PROC PRINCOMP.
- The HPPRINCOMP procedure is specifically designed to operate in the high-performance distributed environment. By default, PROC HPPRINCOMP performs computations on multiple threads. The PRINCOMP procedure executes on a single thread.

Getting Started: HPPRINCOMP Procedure

The following data provide crime rates per 100,000 people in seven categories for each of the 50 US states in 1977:

```
title 'Crime Rates per 100,000 Population by State';

data Crime;
   input State $1-15 Murder Rape Robbery Assault
         Burglary Larceny Auto_Theft;
   datalines;
Alabama        14.2 25.2  96.8 278.3 1135.5 1881.9 280.7
Alaska         10.8 51.6  96.8 284.0 1331.7 3369.8 753.3
Arizona         9.5 34.2 138.2 312.3 2346.1 4467.4 439.5
Arkansas        8.8 27.6  83.2 203.4  972.6 1862.1 183.4
California     11.5 49.4 287.0 358.0 2139.4 3499.8 663.5
Colorado        6.3 42.0 170.7 292.9 1935.2 3903.2 477.1
Connecticut     4.2 16.8 129.5 131.8 1346.0 2620.7 593.2
Delaware        6.0 24.9 157.0 194.2 1682.6 3678.4 467.0
Florida        10.2 39.6 187.9 449.1 1859.9 3840.5 351.4
```

```
Georgia        11.7 31.1 140.5 256.5 1351.1 2170.2 297.9
Hawaii          7.2 25.5 128.0  64.1 1911.5 3920.4 489.4
Idaho           5.5 19.4  39.6 172.5 1050.8 2599.6 237.6
Illinois        9.9 21.8 211.3 209.0 1085.0 2828.5 528.6
Indiana         7.4 26.5 123.2 153.5 1086.2 2498.7 377.4
Iowa            2.3 10.6  41.2  89.8  812.5 2685.1 219.9
Kansas          6.6 22.0 100.7 180.5 1270.4 2739.3 244.3
Kentucky       10.1 19.1  81.1 123.3  872.2 1662.1 245.4
Louisiana      15.5 30.9 142.9 335.5 1165.5 2469.9 337.7
Maine           2.4 13.5  38.7 170.0 1253.1 2350.7 246.9
Maryland        8.0 34.8 292.1 358.9 1400.0 3177.7 428.5
Massachusetts   3.1 20.8 169.1 231.6 1532.2 2311.3 1140.1
Michigan        9.3 38.9 261.9 274.6 1522.7 3159.0 545.5
Minnesota       2.7 19.5  85.9  85.8 1134.7 2559.3 343.1
Mississippi    14.3 19.6  65.7 189.1  915.6 1239.9 144.4
Missouri        9.6 28.3 189.0 233.5 1318.3 2424.2 378.4
Montana         5.4 16.7  39.2 156.8  804.9 2773.2 309.2
Nebraska        3.9 18.1  64.7 112.7  760.0 2316.1 249.1
Nevada         15.8 49.1 323.1 355.0 2453.1 4212.6 559.2
New Hampshire   3.2 10.7  23.2  76.0 1041.7 2343.9 293.4
New Jersey      5.6 21.0 180.4 185.1 1435.8 2774.5 511.5
New Mexico      8.8 39.1 109.6 343.4 1418.7 3008.6 259.5
New York       10.7 29.4 472.6 319.1 1728.0 2782.0 745.8
North Carolina 10.6 17.0  61.3 318.3 1154.1 2037.8 192.1
North Dakota    0.9  9.0  13.3  43.8  446.1 1843.0 144.7
Ohio            7.8 27.3 190.5 181.1 1216.0 2696.8 400.4
Oklahoma        8.6 29.2  73.8 205.0 1288.2 2228.1 326.8
Oregon          4.9 39.9 124.1 286.9 1636.4 3506.1 388.9
Pennsylvania    5.6 19.0 130.3 128.0  877.5 1624.1 333.2
Rhode Island    3.6 10.5  86.5 201.0 1489.5 2844.1 791.4
South Carolina 11.9 33.0 105.9 485.3 1613.6 2342.4 245.1
South Dakota    2.0 13.5  17.9 155.7  570.5 1704.4 147.5
Tennessee      10.1 29.7 145.8 203.9 1259.7 1776.5 314.0
Texas          13.3 33.8 152.4 208.2 1603.1 2988.7 397.6
Utah            3.5 20.3  68.8 147.3 1171.6 3004.6 334.5
Vermont         1.4 15.9  30.8 101.2 1348.2 2201.0 265.2
Virginia        9.0 23.3  92.1 165.7  986.2 2521.2 226.7
Washington      4.3 39.6 106.2 224.8 1605.6 3386.9 360.3
West Virginia   6.0 13.2  42.2    .  597.4 1341.7 163.3
Wisconsin       2.8 12.9  52.2  63.7  846.9 2614.2 220.7
Wyoming          .  21.9  39.7 173.9  811.6 2772.2 282.0
;
```

The following statements invoke the HPPRINCOMP procedure, which requests a principal component analysis of the data and produces Figure 12.1 through Figure 12.4:

```
proc hpprincomp data=Crime;
run;
```

Figure 12.1 displays the "Performance Information," "Data Access Information," "Number of Observations," "Number of Variables," and "Simple Statistics" tables.

The "Performance Information" table shows the procedure executes in single-machine mode—that is, the data reside and the computation is performed on the machine where the SAS session executes. This run of the HPPRINCOMP procedure took place on a multicore machine with four CPUs; one computational thread was spawned per CPU.

The "Data Access Information" table shows that the input data set is accessed with the V9 (base) engine on the client machine where the MVA SAS session executes.

The "Number of Observations" table shows that of the 50 observations in the input data, only 48 observations are used in the analysis because some observations have incomplete data.

The "Number of Variables" table indicates that there are seven variables to be analyzed. By default, if the VAR statement is omitted, all numeric variables that are not listed in other statements are used in the analysis.

The "Simple Statistics" table displays the mean and standard deviation of the analysis variables.

Figure 12.1 Performance Information and Simple Statistics

Crime Rates per 100,000 Population by State

The HPPRINCOMP Procedure

Performance Information	
Execution Mode	Single-Machine
Number of Threads	4

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

Number of Observations Read	50
Number of Observations Used	48

Number of Variables	7

Simple Statistics		
	Mean	**Standard Deviation**
Murder	7.51667	3.93059
Rape	26.07500	10.81304
Robbery	127.55625	88.49374
Assault	214.58750	100.64360
Burglary	1316.37917	423.31261
Larceny	2696.88542	714.75023
Auto_Theft	383.97917	194.37033

Figure 12.2 displays the "Correlation Matrix" table. By default, the PROC HPPRINCOMP statement requests that principal components be computed from the correlation matrix, so the total variance is equal to the number of variables, 7.

Figure 12.2 Correlation Matrix Table

Correlation Matrix							
	Murder	**Rape**	**Robbery**	**Assault**	**Burglary**	**Larceny**	**Auto_Theft**
Murder	1.0000	0.6000	0.4768	0.6485	0.3778	0.0925	0.0555
Rape	0.6000	1.0000	0.5817	0.7316	0.7038	0.6009	0.3282
Robbery	0.4768	0.5817	1.0000	0.5452	0.6200	0.4371	0.5787
Assault	0.6485	0.7316	0.5452	1.0000	0.6082	0.3791	0.2520
Burglary	0.3778	0.7038	0.6200	0.6082	1.0000	0.7932	0.5390
Larceny	0.0925	0.6009	0.4371	0.3791	0.7932	1.0000	0.4246
Auto_Theft	0.0555	0.3282	0.5787	0.2520	0.5390	0.4246	1.0000

Figure 12.3 displays the "Eigenvalues" table. The first principal component accounts for about 57.8% of the total variance, the second principal component accounts for about 18.1%, and the third principal component accounts for about 10.7%. Note that the eigenvalues sum to the total variance.

The eigenvalues indicate that two or three components provide a good summary of the data: two components account for 76% of the total variance, and three components account for 87%. Subsequent components account for less than 5% each.

Figure 12.3 Eigenvalues Table

Eigenvalues of the Correlation Matrix				
	Eigenvalue	**Difference**	**Proportion**	**Cumulative**
1	4.045824	2.781795	0.5780	0.5780
2	1.264030	0.516529	0.1806	0.7586
3	0.747500	0.421175	0.1068	0.8653
4	0.326325	0.061119	0.0466	0.9120
5	0.265207	0.036843	0.0379	0.9498
6	0.228364	0.105613	0.0326	0.9825
7	0.122750		0.0175	1.0000

Figure 12.4 displays the "Eigenvectors" table. From the eigenvectors matrix, you can represent the first principal component, Prin1, as a linear combination of the original variables:

$$\begin{aligned}\text{Prin1} &= 0.302888 \times (\text{Murder}) \\ &\quad + 0.434103 \times (\text{Rape}) \\ &\quad + 0.397055 \times (\text{Robbery}) \\ &\quad \vdots \\ &\quad + 0.288343 \times (\text{Auto_Theft})\end{aligned}$$

Similarly, the second principal component, Prin2, is

$$\begin{aligned}\text{Prin2} = &-0.618929 \times \text{(Murder)}\\ &- 0.170526 \times \text{(Rape)}\\ &+ 0.047125 \times \text{(Robbery)}\\ &.\\ &.\\ &.\\ &+ 0.504003 \times \text{(Auto_Theft)}\end{aligned}$$

where the variables are standardized.

Figure 12.4 Eigenvectors Table

	Eigenvectors						
	Prin1	**Prin2**	**Prin3**	**Prin4**	**Prin5**	**Prin6**	**Prin7**
Murder	0.30289	-0.61893	0.17353	-0.23308	0.54896	0.26371	0.26428
Rape	0.43410	-0.17053	-0.23539	0.06540	0.18075	-0.78232	-0.27946
Robbery	0.39705	0.04713	0.49208	-0.57470	-0.50808	-0.09452	-0.02497
Assault	0.39622	-0.35142	-0.05343	0.61743	-0.51525	0.17395	0.19921
Burglary	0.44164	0.20861	-0.22454	-0.02750	0.11273	0.52340	-0.65085
Larceny	0.35634	0.40570	-0.53681	-0.23231	0.02172	0.04085	0.60346
Auto_Theft	0.28834	0.50400	0.57524	0.41853	0.35939	-0.06024	0.15487

The first component is a measure of the overall crime rate, because the first eigenvector shows approximately equal loadings on all variables. The second eigenvector has high positive loadings on the variables Auto_Theft and Larceny and high negative loadings on the variables Murder and Assault. There is also a small positive loading on the variable Burglary and a small negative loading on the variable Rape. This component seems to measure the preponderance of property crime compared to violent crime. The interpretation of the third component is not obvious.

Syntax: HPPRINCOMP Procedure

The following statements are available in the HPPRINCOMP procedure:

PROC HPPRINCOMP < *options* > ;
BY *variables* ;
CODE < *options* > ;
FREQ *variable* ;
ID *variables* ;
PARTIAL *variables* ;
PERFORMANCE *performance-options* ;
VAR *variables* ;
WEIGHT *variable* ;

The rest of this section provides detailed syntax information for each of the preceding statements, beginning with the PROC HPPRINCOMP statement. The remaining statements are described in alphabetical order.

PROC HPPRINCOMP Statement

PROC HPPRINCOMP < *options* > **;**

The PROC HPPRINCOMP statement invokes the HPPRINCOMP procedure. Optionally, it also identifies input and output data sets, specifies the analyses that are performed, and controls displayed output. Table 12.1 summarizes the options available in the PROC HPPRINCOMP statement.

Table 12.1 Summary of PROC HPPRINCOMP Statement Options

Option	Description
Specify Data Sets	
DATA=	Specifies the name of the input data set
OUT=	Specifies the name of the output data set
OUTSTAT=	Specifies the name of the output data set that contains various statistics
Specify Details of Analysis	
COV	Computes the principal components from the covariance matrix
N=	Specifies the number of principal components to be computed
NOINT	Omits the intercept from the model
PREFIX=	Specifies a prefix for naming the principal components
PARPREFIX=	Specifies a prefix for naming the residual variables
SINGULAR=	Specifies the singularity criterion
STD	Standardizes the principal component scores
VARDEF=	Specifies the divisor used in calculating variances and standard deviations
Suppress the Display of Output	
NOPRINT	Suppresses the display of all output

The following list provides details about these *options*.

COVARIANCE

COV

computes the principal components from the covariance matrix. If you omit the COV option, the correlation matrix is analyzed. The COV option causes variables that have large variances to be more strongly associated with components that have large eigenvalues, and it causes variables that have small variances to be more strongly associated with components that have small eigenvalues. You should not specify the COV option unless the units in which the variables are measured are comparable or the variables are standardized in some way.

DATA=*SAS-data-set*

specifies the SAS data set to be analyzed. The data set can only be an ordinary SAS data set (raw data). If you omit the DATA= option, the HPPRINCOMP procedure uses the most recently created SAS data set.

If PROC HPPRINCOMP executes in distributed mode, the input data are distributed to memory on the appliance nodes and analyzed in parallel, unless the data are already distributed in the appliance database. In that case PROC HPPRINCOMP reads the data alongside the distributed database. For more information about the various execution modes, see the section "Processing Modes" on page 10 in Chapter 3, "Shared Concepts and Topics." For more information about the alongside-the-database model, see the section "Alongside-the-Database Execution" on page 18 in Chapter 3, "Shared Concepts and Topics."

N=*number*

specifies the number of principal components to be computed. The default is the number of variables. The value of the N= option must be an integer greater than or equal to 0.

NOINT

omits the intercept from the model. In other words, the NOINT option requests that the covariance or correlation matrix not be corrected for the mean. When you specify the NOINT option in the HPPRINCOMP procedure, the covariance matrix and, hence, the standard deviations are not corrected for the mean. If you want to obtain the standard deviations corrected for the mean, you can obtain them by using a procedure such as PROC MEANS.

If you use the NOINT option and also create an OUTSTAT= data set, the data set is TYPE=UCORR or TYPE=UCOV rather than TYPE=CORR or TYPE=COV.

NOPRINT

suppresses the display of all output. This option temporarily disables the Output Delivery System (ODS). For more information, see Chapter 20, "Using the Output Delivery System" (*SAS/STAT User's Guide*).

OUT=*SAS-data-set*

creates an output SAS data set to contain observationwise principal component scores. To avoid data duplication when you have large data sets, the variables in the input data set are *not* included in the output data set; however, variables that are specified in the ID statement are included.

If the input data are in distributed form, in which access of data in a particular order cannot be guaranteed, the HPPRINCOMP procedure copies the distribution or partition key to the output data set so that its contents can be joined with the input data.

If you want to create a SAS data set in a permanent library, you must specify a two-level name. For more information about permanent libraries and SAS data sets, see *SAS Language Reference: Concepts*. For more information about OUT= data sets, see the section "Output Data Sets" on page 481.

OUTSTAT=*SAS-data-set*

creates an output SAS data set to contain means, standard deviations, number of observations, correlations or covariances, eigenvalues, and eigenvectors. If you specify the COV option, the data set is TYPE=COV or TYPE=UCOV, depending on the NOINT option, and it contains covariances; otherwise, the data set is TYPE=CORR or TYPE=UCORR, depending on the NOINT option, and it contains correlations. If you specify the PARTIAL statement, the OUTSTAT= data set also contains R squares.

If you want to create a SAS data set in a permanent library, you must specify a two-level name. For more information about permanent libraries and SAS data sets, see *SAS Language Reference: Concepts*. For more information about OUTSTAT= data sets, see the section "Output Data Sets" on page 481.

PREFIX=*name*

specifies a prefix for naming the principal components. By default, the *names* are Prin1, Prin2, . . . , Prin*n*. If you specify PREFIX=Abc, the components are named Abc1, Abc2, Abc3, and so on. The number of characters in the prefix plus the number of digits required to designate the variables should not exceed the current name length that is defined by the VALIDVARNAME= system option.

PARPREFIX=*name*

PPREFIX=*name*

RPREFIX=*name*

specifies a prefix for naming the residual variables in the OUT= data set and the OUTSTAT= data set. By default, the prefix is R_. The number of characters in the prefix plus the maximum length of the variable names should not exceed the current name length that is defined by the VALIDVARNAME= system option.

SINGULAR=*p*

SING=*p*

specifies the singularity criterion, where $0 < p < 1$. If a variable in a PARTIAL statement has an R square as large as $1 - p$ when predicted from the variables listed before it in the statement, the variable is assigned a standardized coefficient of 0. By default, SINGULAR=1E–8.

STANDARD

STD

standardizes the principal component scores in the OUT= data set to unit variance. If you omit the STANDARD option, the scores have a variance equal to the corresponding eigenvalue. Note that the STANDARD option has no effect on the eigenvalues themselves.

VARDEF=DF | N | WDF | WEIGHT | WGT

specifies the divisor to be used in calculating variances and standard deviations. By default, VARDEF=DF. The following table displays the values and associated divisors:

Value	Divisor	Formula	
DF	Error degrees of freedom	$n - i$	(before partialing)
		$n - p - i$	(after partialing)
N	Number of observations	n	
WEIGHT | WGT	Sum of weights	$\sum_{j=1}^{n} w_j$	
WDF	Sum of weights minus one	$\left(\sum_{j=1}^{n} w_j\right) - i$	(before partialing)
		$\left(\sum_{j=1}^{n} w_j\right) - p - i$	(after partialing)

In the formulas for VARDEF=DF and VARDEF=WDF, p is the number of degrees of freedom of the variables in the PARTIAL statement, and i is 0 if the NOINT option is specified and 1 otherwise.

BY Statement

BY *variables* **;**

You can specify a BY statement with PROC HPPRINCOMP to obtain separate analyses of observations in groups that are 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 you specify more than one BY statement, only the last one specified is used.

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 HPPRINCOMP 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 (in Base SAS software).

BY-statement processing is not supported when the HPPRINCOMP procedure runs alongside the database or alongside the Hadoop Distributed File System (HDFS). Alongside-the-database mode or alongside-the-HDFS mode is used if the input data are stored in a database or HDFS and the grid host is the appliance that houses the data.

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

CODE Statement

CODE < *options* > **;**

The CODE statement enables you to write SAS DATA step code for computing the principal component scores either to a file or to a catalog entry. This code can then be included in a DATA step to score new data.

The CODE statement is not supported when the PARTIAL statement is specified. If you specify more than one CODE statement, only the last one specified is used.

Table 12.2 summarizes the *options* available in the CODE statement.

Table 12.2 CODE Statement Options

Option	Description
CATALOG=	Names the catalog entry where the generated code is saved
FILE=	Names the file where the generated code is saved
FORMAT=	Specifies the numeric format for the eigenvectors
GROUP=	Specifies the group identifier for array names and statement labels
LINESIZE=	Specifies the line size of the generated code

For more information about the syntax of the CODE statement, see the section "CODE Statement" in Chapter 19, "Shared Concepts and Topics" (*SAS/STAT User's Guide*).

FREQ Statement

FREQ *variable* **;**

The *variable* in the FREQ statement identifies a numeric variable in the data set that contains the frequency of occurrence of each observation. SAS high-performance analytics procedures that support the FREQ statement treat each observation as if it appeared f times, where f is the value of the FREQ variable for the observation. If the frequency value is not an integer, it is truncated to an integer. If the frequency value is less than 1 or missing, the observation is not used in the analysis. When the FREQ statement is not specified, each observation is assigned a frequency of 1.

ID Statement

ID *variables* **;**

The ID statement lists one or more variables from the input data set that are transferred to output data sets created by SAS high-performance analytics procedures, provided that the output data set produces one (or more) records per input observation.

For information about the common ID statement in SAS high-performance analytics procedures, see the section "ID Statement" on page 46 in Chapter 4, "Shared Statistical Concepts."

PARTIAL Statement

PARTIAL *variables* **;**

If you want to analyze a partial correlation or covariance matrix, specify the names of the numeric variables to be partialed out in the PARTIAL statement. The HPPRINCOMP procedure computes the principal components of the residuals from the prediction of the VAR variables by the PARTIAL variables. If you request an OUT= or OUTSTAT= data set, the residual variables are named by prefixing either the characters R_ (by default) or the string specified in the PARPREFIX= option to the VAR variables.

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 the HPPRINCOMP procedure.

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

The PERFORMANCE statement is documented further in the section "PERFORMANCE Statement" on page 36 of Chapter 3, "Shared Concepts and Topics."

VAR Statement

VAR *variables* **;**

The VAR statement lists the numeric variables to be analyzed. If you omit the VAR statement, all numeric variables that are not specified in other statements are analyzed.

WEIGHT Statement

WEIGHT *variable* **;**

The *variable* in the WEIGHT statement is used as a weight to perform a weighted analysis of the data. Observations that have nonpositive or missing weights are not included in the analysis. If you do not specify a WEIGHT statement, all observations that are used in the analysis are assigned a weight of 1.

Details: HPPRINCOMP Procedure

Missing Values

Observations that have missing values for any variable in the VAR, PARTIAL, FREQ, or WEIGHT statement are omitted from the analysis and are given missing values for principal component scores in the OUT= data set.

Output Data Sets

OUT= Data Set

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

- variables that are explicitly created by the statement
- variables that are listed in the ID statement
- distribution keys or hash keys that are transferred from the input data set

Having these variables and keys in the OUT= data set enables you to add output data set information that is necessary for subsequent SQL joins without copying the entire input data set to the output data set. For more

information about output data sets that are produced when PROC HPPRINCOMP is run in distributed mode, see the section "Output Data Sets" on page 34 in Chapter 3, "Shared Concepts and Topics."

The new variables that are created for the OUT= data set contain the principal component scores. The N= option determines the number of new variables. The names of the new variables are formed by concatenating the value given by the PREFIX= option (or Prin if PREFIX= is omitted) to the numbers 1, 2, 3, and so on. The new variables have mean 0 and a variance equal to the corresponding eigenvalue, unless you specify the STANDARD option to standardize the scores to unit variance. Also, if you specify the COV option, PROC HPPRINCOMP computes the principal component scores from the corrected or uncorrected (if the NOINT option is specified) variables rather than from the standardized variables.

If you use a PARTIAL statement, the OUT= data set also contains the residuals from predicting the VAR variables from the PARTIAL variables.

OUTSTAT= Data Set

The OUTSTAT= data set is similar to the TYPE=CORR data set that the CORR procedure produces. The following table relates the TYPE= value for the OUTSTAT= data set to the options that are specified in the PROC HPPRINCOMP statement:

Options	TYPE=
(Default)	CORR
COV	COV
NOINT	UCORR
COV NOINT	UCOV

Note that the default (neither the COV nor NOINT option) produces a TYPE=CORR data set.

The new data set contains the following variables:

- the BY variables, if any
- two new variables, _TYPE_ and _NAME_, both character variables
- the variables that are analyzed (that is, those in the VAR statement); or, if there is no VAR statement, all numeric variables not listed in any other statement; or, if there is a PARTIAL statement, the residual variables as described in the section "OUT= Data Set"

Each observation in the new data set contains some type of statistic, as indicated by the _TYPE_ variable. The values of the _TYPE_ variable are as follows:

TYPE	Contents
MEAN	mean of each variable. If you specify the PARTIAL statement, this observation is omitted.
STD	standard deviations. If you specify the COV option, this observation is omitted, so the SCORE procedure does not standardize the variables before computing scores. If you use the PARTIAL statement, the standard deviation of a variable is computed as its root mean squared error as predicted from the PARTIAL variables.

USTD	uncorrected standard deviations. When you specify the NOINT option in the PROC HPPRINCOMP statement, the OUTSTAT= data set contains standard deviations not corrected for the mean. However, if you also specify the COV option in the PROC HPPRINCOMP statement, this observation is omitted.
N	number of observations on which the analysis is based. This value is the same for each variable. If you specify the PARTIAL statement and the value of the VARDEF= option is DF or unspecified, then the number of observations is decremented by the degrees of freedom for the PARTIAL variables.
SUMWGT	the sum of the weights of the observations. This value is the same for each variable. If you specify the PARTIAL statement and VARDEF=WDF, then the sum of the weights is decremented by the degrees of freedom for the PARTIAL variables. This observation is output only if the value is different from that in the observation for which _TYPE_='N'.
CORR	correlations between each variable and the variable specified by the _NAME_ variable. The number of observations for which _TYPE_='CORR' is equal to the number of variables being analyzed. If you specify the COV option, no _TYPE_='CORR' observations are produced. If you use the PARTIAL statement, the partial correlations, not the raw correlations, are output.
UCORR	uncorrected correlation matrix. When you specify the NOINT option without the COV option in the PROC HPPRINCOMP statement, the OUTSTAT= data set contains a matrix of correlations not corrected for the means. However, if you also specify the COV option in the PROC HPPRINCOMP statement, this observation is omitted.
COV	covariances between each variable and the variable specified by the _NAME_ variable. _TYPE_='COV' observations are produced only if you specify the COV option. If you use the PARTIAL statement, the partial covariances, not the raw covariances, are output.
UCOV	uncorrected covariance matrix. When you specify the NOINT and COV options in the PROC HPPRINCOMP statement, the OUTSTAT= data set contains a matrix of covariances not corrected for the means.
EIGENVAL	eigenvalues. If the N= option requests less than the maximum number of principal components, only the specified number of eigenvalues are produced, with missing values filling out the observation.
SCORE	eigenvectors. The _NAME_ variable contains the name of the corresponding principal component as constructed from the PREFIX= option. The number of observations for which _TYPE_='SCORE' equals the number of principal components computed. The eigenvectors have unit length unless you specify the STD option, in which case the unit-length eigenvectors are divided by the square roots of the eigenvalues to produce scores that have unit standard deviations. To obtain the principal component scores, if the COV option is not specified, these coefficients should be multiplied by the standardized data. For the COV option, these coefficients should be multiplied by the centered data. To center and standardize the data, you should use means that are obtained from the observation for which _TYPE_='MEAN' and standard deviations that are obtained from the observation for which _TYPE_='STD'.
USCORE	scoring coefficients to be applied without subtracting the mean from the raw variables. Observations for which _TYPE_='USCORE' are produced when you specify the NOINT option in the PROC HPPRINCOMP statement.

	To obtain the principal component scores, these coefficients should be multiplied by the data that are standardized by the uncorrected standard deviations obtained from the observation for which _TYPE_='USTD'.
RSQUARED	R squares for each VAR variable as predicted by the PARTIAL variables.
B	regression coefficients for each VAR variable as predicted by the PARTIAL variables. This observation is produced only if you specify the COV option.
STB	standardized regression coefficients for each VAR variable as predicted by the PARTIAL variables. If you specify the COV option, this observation is omitted.

You can use the data set in the SCORE procedure to compute principal component scores, or you can use it as input to the FACTOR procedure and specify METHOD=SCORE to rotate the components. If you use the PARTIAL statement, the scoring coefficients should be applied to the residuals, not to the original variables.

Computational Method

Multithreading

Threading is the organization of computational work into multiple tasks (processing units that can be scheduled by the operating system). A task is associated with a thread. Multithreading is the concurrent execution of threads. When multithreading is possible, you can realize substantial performance gains compared to the performance that you get from sequential (single-threaded) execution.

The number of threads that the HPPRINCOMP procedure spawns is determined by the number of CPUs on a machine and can be controlled in the following ways:

- You can specify the CPU count by using the CPUCOUNT= SAS system option. For example, if you specify the following statements, the HPPRINCOMP procedure schedules threads as if it were executing on a system that had four CPUs, regardless of the actual CPU count:

```
options cpucount=4;
```

- You can specify the NTHREADS= option in the PERFORMANCE statement to determine the number of threads. This specification overrides the system option. Specify NTHREADS=1 to force single-threaded execution.

The number of threads per machine is displayed in the "Performance Information" table, which is part of the default output. The HPPRINCOMP procedure allocates one thread per CPU.

The tasks that are multithreaded by the HPPRINCOMP procedure are primarily defined by dividing the data processed on a single machine among the threads; that is, PROC HPPRINCOMP implements multithreading through a data-parallel model. For example, if the input data set has 1,000 observations and you are running on four threads, then 250 observations are associated with each thread. All operations that require access to the data are then multithreaded. Those operations include the following:

- formation of the crossproducts matrix
- principal component scoring of observations

Displayed Output

The following sections describe the output that PROC HPPRINCOMP produces. The output is organized into various tables, which are discussed in order of appearance.

Performance Information

The "Performance Information" table is produced by default. It displays information about the execution mode. For single-machine mode, the table displays the number of threads used. For distributed mode, the table displays the grid mode (symmetric or asymmetric), the number of compute nodes, and the number of threads per node.

Data Access Information

The "Data Access Information" table is produced by default. For the input and output data sets, it displays the libref and data set name, the engine used to access the data, the role (input or output) of the data set, and the path that data followed to reach the computation.

Number of Observations

The "Number of Observations" table displays the number of observations read from the input data set and the number of observations used in the analysis. If you specify a FREQ statement, the table also displays the sum of frequencies read and used.

Number of Variables

The "Number of Variables" table displays the number of VAR variables and the number of PARTIAL variables.

Simple Statistics

The "Simple Statistics" table displays the mean and standard deviation (std) for each variable. If you specify the NOINT option, the uncorrected standard deviation (ustd) is displayed.

Correlation Matrix

The "Correlation Matrix" table displays the correlation or, if you specify the COV option, the covariance matrix.

Regression Statistics

When you specify the PARTIAL statement, the HPPRINCOMP procedure produces a "Regression Statistics" table that displays the R square and root mean squared error (RMSE) for each VAR variable as predicted by the PARTIAL variables.

Regression Coefficients

When you specify the PARTIAL statement, the HPPRINCOMP procedure produces a "Regression Coefficients" table that displays standardized regression coefficients or, if you specify the COV option, regression coefficients for predicting the VAR variables from the PARTIAL variables.

Partial Correlation Matrix

When you specify the PARTIAL statement, the HPPRINCOMP procedure produces a "Partial Correlation Matrix" table that displays the partial correlation matrix or, if you specify the COV option, the partial covariance matrix.

Total Variance

If you specify the COV option, the HPPRINCOMP procedure produces a simple table that displays the total variance.

Eigenvalues

Unless you specify the NOPRINT option, the HPPRINCOMP procedure produces an "Eigenvalues" table that displays eigenvalues of the correlation or covariance matrix, along with the difference between successive eigenvalues, the proportion of variance explained by each eigenvalue, and the cumulative proportion of variance explained.

Eigenvectors

Unless you specify the NOPRINT option, the HPPRINCOMP procedure produces an "Eigenvectors" table that displays the eigenvectors.

Timing Information

If you specify the DETAILS option in the PERFORMANCE statement, the HPPRINCOMP procedure produces a "Timing" table that displays the elapsed time of each main task of the procedure.

ODS Table Names

PROC HPPRINCOMP assigns a name to each table that it creates. You can use these names to reference the ODS table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in Table 12.3. For more information about ODS, see Chapter 20, "Using the Output Delivery System" (*SAS/STAT User's Guide*).

Table 12.3 ODS Tables Produced by PROC HPPRINCOMP

Table Name	Description	Required Statement / Option
Corr	Correlation matrix	Default
Cov	Covariance matrix	COV
DataAccessInfo	Information about modes of data access	Default
Eigenvalues	Eigenvalues	Default
Eigenvectors	Eigenvectors	Default
NObs	Number of observations read and used	Default
NVars	Number of variables and partial variables	Default
ParCorr	Partial correlation matrix	PARTIAL statement
ParCov	Uncorrected partial covariance matrix	PARTIAL statement and COV

Table 12.3 *continued*

Table Name	Description	Required Statement / Option
PerformanceInfo	Information about the high-performance computing environment	Default
RegCoef	Regression coefficients	PARTIAL statement and COV
RSquareRMSE	Regression statistics: R squares and RMSEs	PARTIAL statement
SimpleStatistics	Simple statistics	Default
StdRegCoef	Standardized regression coefficients	PARTIAL statement
Timing	Absolute and relative times of tasks performed by the procedure	PERFORMANCE DETAILS
TotalVariance	Total variance	COV

Examples: HPPRINCOMP Procedure

Example 12.1: Analyzing Mean Temperatures of US Cities

This example analyzes mean daily temperatures of selected US cities in January and July. The following statements create the Temperature data set:

```
data Temperature;
   length Cityid $ 2;
   title 'Mean Temperature in January and July for Selected Cities ';
   input City $1-15 January July;
   Cityid = substr(City,1,2);
   datalines;
Mobile          51.2 81.6
Phoenix         51.2 91.2
Little Rock     39.5 81.4
Sacramento      45.1 75.2
Denver          29.9 73.0

   ... more lines ...

Cheyenne        26.6 69.1
;
```

The following statements invoke the HPPRINCOMP procedure, which requests a principal component analysis of the Temperature data set and outputs the scores to the Scores data set (OUT= Scores). The Cityid variable in the ID statement is also included in the output data set.

```
title 'Mean Temperature in January and July for Selected Cities';
proc hpprincomp data=Temperature cov out=Scores;
   var July January;
   id Cityid;
run;
```

Output 12.1.1 displays the PROC HPPRINCOMP output. The standard deviation of January (11.712) is higher than the standard deviation of July (5.128). The COV option in the PROC HPPRINCOMP statement requests that the principal components be computed from the covariance matrix. The total variance is 163.474. The first principal component accounts for about 94% of the total variance, and the second principal component accounts for only about 6%. The eigenvalues sum to the total variance.

Note that January receives a higher loading on Prin1 because it has a higher standard deviation than July. Also note that the HPPRINCOMP procedure calculates the scores by using the centered variables rather than the standardized variables.

Output 12.1.1 Results of Principal Component Analysis

Mean Temperature in January and July for Selected Cities

The HPPRINCOMP Procedure

Performance Information	
Execution Mode	Single-Machine
Number of Threads	4

Data Access Information			
Data	**Engine**	**Role**	**Path**
WORK.TEMPERATURE	V9	Input	On Client
WORK.SCORES	V9	Output	On Client

Number of Observations Read	64
Number of Observations Used	64

Number of Variables	2

Simple Statistics		
	Mean	**Standard Deviation**
July	75.60781	5.12762
January	32.09531	11.71243

Covariance Matrix		
	July	**January**
July	26.29248	46.82829
January	46.82829	137.18109

Total Variance	163.47356647

Eigenvalues of the Covariance Matrix				
	Eigenvalue	**Difference**	**Proportion**	**Cumulative**
1	154.310607	145.147647	0.9439	0.9439
2	9.162960		0.0561	1.0000

Eigenvectors		
	Prin1	**Prin2**
July	0.34353	0.93914
January	0.93914	-0.34353

Example 12.2: Computing Principal Components in Single-Machine and Distributed Modes

PROC HPPRINCOMP shows its real power when the computation is conducted with multiple threads or in a distributed environment. This example shows how you can run PROC HPPRINCOMP in single-machine and distributed modes. For more information about the execution modes of SAS high-performance analytics procedures, see the section "Processing Modes" on page 10 in Chapter 3, "Shared Concepts and Topics." The focus of this example is to show how you can switch the modes of execution in PROC HPPRINCOMP. The following DATA step generates the data:

```
data ex2Data;
   array x{100};
   do i = 1 to 5000000;
      do j = 1 to dim(x);
         x[j] = ranuni(1);
      end;
      output;
   end;
run;
```

The following statements use PROC HPPRINCOMP to perform a principal component analysis and to output various statistics to the Stats data set (OUTSTAT= Stats):

```
proc hpprincomp data=ex2Data n=20 outstat=Stats;
   var x:;
   performance details;
run;
```

Output 12.2.1 shows the "Performance Information" table. This table shows that the HPPRINCOMP procedure executes in single-machine mode on four threads, because the client machine has four CPUs. You can force a certain number of threads on any machine to be involved in the computations by specifying the NTHREADS= option in the PERFORMANCE statement.

Output 12.2.1 Performance Information in Single-Machine Mode

The HPPRINCOMP Procedure

Performance Information	
Execution Mode	Single-Machine
Number of Threads	4

Output 12.2.2 shows timing information for the PROC HPPRINCOMP run. This table is produced when you specify the DETAILS option in the PERFORMANCE statement. You can see that, in this case, the majority of time is spent reading the data and computing the moments.

Output 12.2.2 Timing in Single-Machine Mode

Procedure Task Timing		
Task	**Seconds**	**Percent**
Reading Data and Computing Moments	53.24	71.82%
Computing Correlation Matrix	20.72	27.95%
Performing Eigenvalue Decomposition	0.15	0.21%
Producing Output Statistics Data Set	0.01	0.02%

To switch to running PROC HPPRINCOMP in distributed mode, specify valid values for the NODES=, INSTALL=, and HOST= options in the PERFORMANCE statement. An alternative to specifying the INSTALL= and HOST= options in the PERFORMANCE statement is to use OPTIONS SET commands to set appropriate values for the GRIDHOST and GRIDINSTALLLOC environment variables. For information about setting these options or environment variables, see the section "Processing Modes" on page 10 in Chapter 3, "Shared Concepts and Topics."

The following statements provide an example. To run these statements successfully, you need to set the macro variables GRIDHOST and GRIDINSTALLLOC to resolve to appropriate values, or you can replace the references to macro variables with appropriate values.

```
proc hpprincomp data=ex2Data n=20 outstat=Stats;
   var x:;
   performance details nodes = 4
               host="&GRIDHOST" install="&GRIDINSTALLLOC";
run;
```

The execution mode in the "Performance Information" table shown in Output 12.2.3 indicates that the calculations were performed in a distributed environment that uses four nodes, each of which uses 32 threads.

Output 12.2.3 Performance Information in Distributed Mode

Performance Information	
Host Node	<< your grid host >>
Install Location	<< your grid install location >>
Execution Mode	Distributed
Number of Compute Nodes	4
Number of Threads per Node	32

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

```
NOTE: The HPPRINCOMP procedure is executing in the distributed
      computing environment with 4 worker nodes.
```

Output 12.2.4 shows timing information for this distributed run of the HPPRINCOMP procedure. In contrast with the single-machine mode (where reading the data and computing the moments dominate the time spent), the majority of time in the distributed-mode run is spent distributing the data.

Output 12.2.4 Timing in Distributed Mode

Procedure Task Timing		
Task	**Seconds**	**Percent**
Obtaining Settings	0.00	0.00%
Distributing Data	35.44	95.09%
Reading Data and Computing Moments	1.27	3.40%
Computing Correlation Matrix	0.29	0.78%
Performing Eigenvalue Decomposition	0.00	0.00%
Producing Output Statistics Data Set	0.01	0.04%
Waiting on Client	0.26	0.69%

References

Cooley, W. W. and Lohnes, P. R. (1971), *Multivariate Data Analysis*, New York: John Wiley & Sons.

Gnanadesikan, R. (1977), *Methods for Statistical Data Analysis of Multivariate Observations*, New York: John Wiley & Sons.

Hotelling, H. (1933), "Analysis of a Complex of Statistical Variables into Principal Components," *Journal of Educational Psychology*, 24, 417–441, 498–520.

Kshirsagar, A. M. (1972), *Multivariate Analysis*, New York: Marcel Dekker.

Mardia, K. V., Kent, J. T., and Bibby, J. M. (1979), *Multivariate Analysis*, London: Academic Press.

Morrison, D. F. (1976), *Multivariate Statistical Methods*, 2nd Edition, New York: McGraw-Hill.

Pearson, K. (1901), "On Lines and Planes of Closest Fit to Systems of Points in Space," *Philosophical Magazine*, 6, 559–572.

Rao, C. R. (1964), "The Use and Interpretation of Principal Component Analysis in Applied Research," *Sankhyā, Series A*, 26, 329–358.

Chapter 13
The HPQUANTSELECT Procedure

Contents

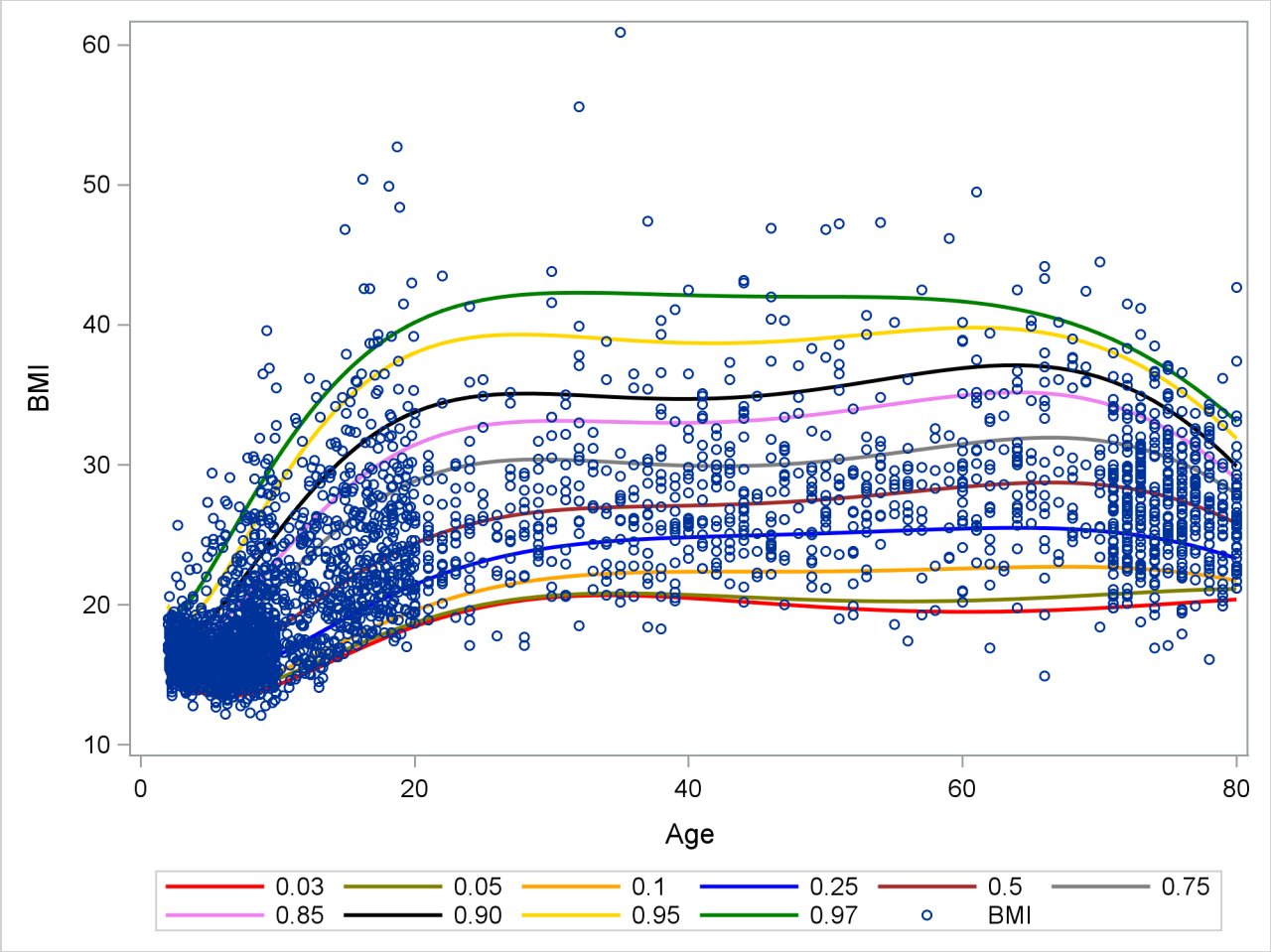
Overview: HPQUANTSELECT Procedure **494**
PROC HPQUANTSELECT Features 496
PROC HPQUANTSELECT Contrasted with Other SAS Procedures 497
Getting Started: HPQUANTSELECT Procedure **498**
Syntax: HPQUANTSELECT Procedure **507**
PROC HPQUANTSELECT Statement 508
BY Statement 509
CLASS Statement 510
CODE Statement 510
ID Statement 511
MODEL Statement 511
OUTPUT Statement 513
PARTITION Statement 515
PERFORMANCE Statement 515
SELECTION Statement 516
WEIGHT Statement 517
Details: HPQUANTSELECT Procedure **517**
Quantile Regression 517
Linear Model with iid Errors 517
Linear-in-Parameter Model with Non-iid Settings 519
Criteria Used in Model Selection 519
Quasi-likelihood Information Criteria 519
Statistical Tests for Significance Level 520
Diagnostic Statistics 522
Classification Variables and the SPLIT Option 523
Macro Variables That Contain Selected Effects 524
Using Validation and Test Data 525
Using the Validation ACL as the STOP= Criterion 526
Using the Validation ACL as the CHOOSE= Criterion 526
Using the Validation ACL as the SELECT= Criterion 526
Computational Method 527
Multithreading 527
Output Data Set 528
Displayed Output 528
Performance Information 528
Data Access Information 528

Model Information 528
Selection Information 528
Number of Observations 529
Class Level Information 529
Dimensions 529
Entry and Removal Candidates 529
Selection Summary 529
Stop Reason 529
Selection Reason 530
Selected Effects 530
Fit Statistics 530
Parameter Estimates 531
Timing Information 531
ODS Table Names 531
Examples: HPQUANTSELECT Procedure **532**
Example 13.1: Simulation Study 532
Example 13.2: Growth Charts for Body Mass Index 535
References **538**

Overview: HPQUANTSELECT Procedure

Quantile regression is a systematic statistical methodology for modeling conditional quantile functions of a response variable on explanatory covariate effects. Although modern quantile regression was introduced by Koenker and Bassett (1978), simple quantile regression that uses only the intercept as the explanatory effect has been practiced for much longer, because quantile is no more than a generalized notion for terms such as percentile, decile, quintile, and quartile. A conditional quantile of a response variable at quantile level τ denotes the value below which the proportion of the conditional response population is τ. Unlike linear regression, which exclusively focuses on the conditional mean, quantile regression can anatomize the entire response distribution and examine how the covariate effects influence the shape of the response distribution over the entire range of quantile levels $[0, 1]$. Therefore, quantile regression provides a more comprehensive view of the regression relationship. Figure 13.1 shows an example of quantile regression that creates growth charts for the men's body mass index (BMI) as quantile curves. Each entry in the legend shows the quantile level for the corresponding quantile curve. For example, the curve whose quantile level $\tau = 0.85$ corresponds to the 85th conditional percentile. For more information about the BMI example, see "Example 13.2: Growth Charts for Body Mass Index" on page 535.

Figure 13.1 Growth Chart for Body Mass Index



The HPQUANTSELECT procedure is a high-performance procedure that fits and performs effect selection for quantile regression analysis. PROC HPQUANTSELECT supports continuous variables, CLASS variables, and the interactions of these variables. PROC HPQUANTSELECT supports statistical inferences on quantile regression models with or without the assumption of independently and identically distributed (iid) errors. PROC HPQUANTSELECT also offers extensive capabilities for customizing the effect selection by using a wide variety of selection and stopping criteria.

PROC HPQUANTSELECT runs in either single-machine mode or distributed mode. NOTE: Distributed mode requires SAS High-Performance Statistics.

PROC HPQUANTSELECT Features

The main features of the HPQUANTSELECT procedure are as follows:

- **Model specification**
  - supports quantile regression for single or multiple quantile levels
  - supports GLM and reference cell parameterization for classification effects
  - supports any degree of interaction (crossed effects) and nested effects
  - supports statistical inferences with or without iid errors assumption
  - supports hierarchy among effects
  - supports partitioning of data into training, validation, and testing roles
  - supports a CODE statement to write SAS DATA step code to a file or catalog entry for computing predicted quantiles
  - supports a WEIGHT statement for a weighted analysis
- **Selection control**
  - provides multiple effect-selection methods
  - offers selection of individual levels of classification effects
  - provides effect selection based on a variety of selection criteria
  - provides stopping rules based on a variety of model evaluation criteria
- **Display and output**
  - produces output data sets that contain predicted values, residuals, standardized errors, and confidence limits of predicted values

The HPQUANTSELECT procedure supports the following effect-selection methods. For more information about these methods, see the section "Methods" on page 63 in Chapter 4, "Shared Statistical Concepts."

forward selection	starts with no effects in the model and adds effects.
backward elimination	starts with all effects in the model and deletes effects.
stepwise selection	is similar to the forward selection method except that effects already in the model do not necessarily stay there.

PROC HPQUANTSELECT Contrasted with Other SAS Procedures

For general contrasts between SAS High-Performance Analytics procedures and other SAS procedures, see the section "Common Features of SAS High-Performance Statistical Procedures" on page 42 in Chapter 4, "Shared Statistical Concepts." The following remarks contrast the HPQUANTSELECT procedure with the QUANTSELECT and QUANTREG procedures in SAS/STAT.

The major functional differences between the HPQUANTSELECT and QUANTSELECT procedures are as follows:

- The HPQUANTSELECT procedure uses an interior point algorithm for model fitting. The QUANTSELECT procedure uses a flexible simplex algorithm for model fitting.
- The HPQUANTSELECT procedure can output confidence limits for parameter estimates.
- The HPQUANTSELECT procedure does not support the LASSO and adaptive LASSO effect-selection methods.
- The HPQUANTSELECT procedure does not support the TESTDATA= and VALDATA= options in its PROC statement.
- The HPQUANTSELECT procedure does not support graphical summaries for the effect selection processes.

Both the HPQUANTSELECT and QUANTSELECT procedures support the forward, backward, and stepwise effect-selection methods and the ability to use separate validation and test data via the PARTITION statement. For more information about the QUANTSELECT procedure, see Chapter 84, "The QUANTSELECT Procedure" (*SAS/STAT User's Guide*).

The major functional differences between the HPQUANTSELECT and QUANTREG procedures are as follows:

- The QUANTREG procedure does not support any effect-selection methods. It does not output the following fit statistics: AIC, AICC, SBS, VALIDATE, TEST, R1, and adjusted R1. And it does not support the PARTITION statement.
- The QUANTREG procedure provides three algorithms for fitting quantile regression models: simplex algorithm, interior point algorithm, and smoothing algorithm. The HPQUANTSELECT procedure supports only the interior point algorithm.
- The QUANTREG procedure supports the rank test, which is not available for the HPQUANTSELECT procedure. Both the QUANTREG and HPQUANTSELECT procedures support the Wald test and the likelihood test.
- The QUANTREG procedure supports two methods of estimating the covariance matrix of the parameter estimates: an asymptotic method and a bootstrap method. The HPQUANTSELECT procedure supports only the asymptotic method.

For more information about the QUANTREG procedure, see Chapter 83, "The QUANTREG Procedure" (*SAS/STAT User's Guide*).

The HPQUANTSELECT procedure is also different from the QUANTSELECT and QUANTREG procedures in the following respects:

- The HPQUANTSELECT procedure supports the CODE statement, which is not available in the QUANTSELECT and QUANTREG procedures.
- The HPQUANTSELECT procedure does not support quantile process regression, whereas the QUANTREG procedure does support quantile process regression. The QUANTSELECT procedure supports effect selection for quantile process regression.
- The HPQUANTSELECT procedure does not support the EFFECT statement, which provides constructed effects such as polynomial effects, spline effects, collection effects, and multimember classification effects.
- The HPQUANTSELECT procedure can output confidence limits for mean predicted quantiles; this functionality is not available in the QUANTSELECT and QUANTREG procedures.

In addition to having many similarities to the QUANTSELECT and QUANTREG procedures, the HPQUANTSELECT procedure also compares closely to the HPREG procedure. PROC HPREG is a high-performance procedure that performs effect selection in the framework of general linear models. The HPQUANTSELECT procedure inherits most of its syntax from the HPREG and QUANTREG procedures. The HPQUANTSELECT procedure provides results that are similar to those of the HPREG and QUANTSELECT procedures.

Getting Started: HPQUANTSELECT Procedure

The following example is modeled on the example in the section "Getting Started: QUANTSELECT Procedure" in the *SAS/STAT User's Guide*. The Sashelp.baseball data set contains salary and performance information for Major League Baseball (MLB) players, excluding pitchers, who played in at least one game in both the 1986 and 1987 seasons. The salaries (Time Inc. 1987) are for the 1987 season, and the performance measures are for the 1986 season (Reichler 1987).

The following statements display the variables in the data set. Figure 13.2 shows the results.

```
proc contents varnum data=sashelp.baseball;
   ods select position;
run;
```

Figure 13.2 Sashelp.Baseball Data Set

The CONTENTS Procedure

Variables in Creation Order				
#	Variable	Type	Len	Label
1	Name	Char	18	Player's Name
2	Team	Char	14	Team at the End of 1986
3	nAtBat	Num	8	Times at Bat in 1986
4	nHits	Num	8	Hits in 1986
5	nHome	Num	8	Home Runs in 1986
6	nRuns	Num	8	Runs in 1986
7	nRBI	Num	8	RBIs in 1986
8	nBB	Num	8	Walks in 1986
9	YrMajor	Num	8	Years in the Major Leagues
10	CrAtBat	Num	8	Career Times at Bat
11	CrHits	Num	8	Career Hits
12	CrHome	Num	8	Career Home Runs
13	CrRuns	Num	8	Career Runs
14	CrRbi	Num	8	Career RBIs
15	CrBB	Num	8	Career Walks
16	League	Char	8	League at the End of 1986
17	Division	Char	8	Division at the End of 1986
18	Position	Char	8	Position(s) in 1986
19	nOuts	Num	8	Put Outs in 1986
20	nAssts	Num	8	Assists in 1986
21	nError	Num	8	Errors in 1986
22	Salary	Num	8	1987 Salary in $ Thousands
23	Div	Char	16	League and Division
24	logSalary	Num	8	Log Salary

Suppose you want to investigate how the MLB players' salaries for the 1987 season depend on performance measures for the players' previous season and MLB career. You might worry that some players who are outliers could dominate your least squares analysis. To address this concern, you can use the following statements to obtain a median regression model, which is equivalent to the 50th conditional percentile or the quantile regression model at quantile level 0.5:

```
proc hpquantselect data=sashelp.baseball;
   class league division;
   model Salary = nAtBat nHits nHome nRuns nRBI nBB
                  yrMajor crAtBat crHits crHome crRuns crRbi
                  crBB league division nOuts nAssts nError
         / clb;
run;
```

The CLB option in the MODEL statement requests 95% confidence limits for the parameter estimates. If you do not use the SELECTION statement, the HPQUANTSELECT procedure fits the full model that is specified by the MODEL statement without any effect selection.

Figure 13.3 Performance, Data Access, and Model Information

The HPQUANTSELECT Procedure

Performance Information	
Execution Mode	Single-Machine
Number of Threads	4

Model Information	
Data Source	SASHELP.BASEBALL
Dependent Variable	Salary
Class Parameterization	GLM
Optimization Algorithm	Interior

Number of Observations Read	322
Number of Observations Used	263

Class Level Information		
Class	**Levels**	**Values**
League	2	American National
Division	2	East West

Dimensions	
Number of Effects	19
Number of Parameters	21

Figure 13.3 displays the "Performance Information," "Data Access Information," and "Model Information" tables.

The "Performance Information" table shows that the HPQUANTSELECT procedure executes in client mode—that is, the model is fit on the machine where the SAS session executes. This step was performed on a multicore machine that contained four CPUs; one computational thread was spawned per CPU.

The "Data Access Information" table shows that the input data set is accessed with the V9 (base) engine on the client machine.

The "Model Information" table identifies the data source and response and shows that the CLASS variables are parameterized in the GLM parameterization, which is the default.

Figure 13.4 Number of Observations, Class Level Information, and Dimensions Tables

Number of Observations Read	322
Number of Observations Used	263

Class Level Information

Class	Levels	Values
League	2	American National
Division	2	East West

Dimensions	
Number of Effects	19
Number of Parameters	21

Figure 13.4 displays the "Number of Observations," "Class Level Information," and "Dimensions" tables.

The "Number of Observations" table shows that, of the 322 observations, PROC HPQUANTSELECT uses only 263 observations for model fitting and ignores 59 incomplete observations.

The "Class Level Information" table shows level information for two CLASS effects that the CLASS statement identifies: League and Division. League has two levels: American League and National League. Division also has two levels: East Division and West Division.

The "Dimensions" table shows that the MODEL statement identifies 19 effects for model fitting besides the intercept effect. Because the 19 effects include two CLASS effects and each level of a CLASS effect corresponds to a parameter, the 19 effects contain a total of 21 parameters.

Figure 13.5 Fit Statistics

The HPQUANTSELECT Procedure

Quantile Level = 0.5

Fit Statistics	
Objective Function	25977
R1	0.40584
Adj R1	0.36200
AIC	2453.81587
AICC	2456.94344
SBC	2521.68680
ACL	98.77118

Figure 13.5 displays the "Fit Statistics" table, which shows the values of model fitting criteria for the fitted median model. For more information about model fitting criteria for quantile regression, see the section "Details: HPQUANTSELECT Procedure" on page 517.

Figure 13.6 Parameter Estimates

Parameter Estimates

Parameter	DF	Estimate	Standard Error	95% Confidence Limits	
Intercept	1	-67.75322	39.95908	-146.46197	10.95553
nAtBat	1	-1.57112	0.44700	-2.45160	-0.69064
nHits	1	8.82192	1.94990	4.98114	12.66270
nHome	1	-5.91757	4.91015	-15.58926	3.75412
nRuns	1	-5.17076	2.14914	-9.40400	-0.93753
nRBI	1	0.77547	2.15469	-3.46870	5.01963
nBB	1	5.28866	1.67603	1.98732	8.59000
YrMajor	1	6.61877	6.61798	-6.41690	19.65444
CrAtBat	1	-0.04463	0.15485	-0.34964	0.26038
CrHits	1	0.07896	0.73594	-1.37064	1.52857
CrHome	1	3.78231	1.90065	0.03854	7.52607
CrRuns	1	1.23105	0.77137	-0.28833	2.75044
CrRbi	1	-0.70695	0.76888	-2.22144	0.80754
CrBB	1	-0.68911	0.41382	-1.50423	0.12601
League American	1	-34.39136	24.37175	-82.39724	13.61451
League National	0	0	.	.	.
Division East	1	60.30856	27.28730	6.55984	114.05728
Division West	0	0	.	.	.
nOuts	1	0.23273	0.12110	-0.00581	0.47126
nAssts	1	0.09824	0.18888	-0.27381	0.47029
nError	1	-0.81574	3.51436	-7.73810	6.10661

Figure 13.6 displays the "Parameter Estimates" table, which shows the parameter estimates of the fitted median model. You can see that, of the 19 effective parameters whose degrees of freedom are not zero, the fitted model contains 13 insignificant parameters whose 95% confidence intervals cover zeros. Because more than half of the 19 effective parameters are insignificant, you might worry that the model is overfitted.

It is well known that both overfitting and underfitting harm the prediction performance of a model. You can prevent overfitting and underfitting by using a good effect-selection technique. The following statements apply the forward selection method and the SL (significance level) criterion to choose a parsimonious model for the Sashelp.baseball data set:

```
proc hpquantselect data=sashelp.baseball;
   class league division;
   model Salary = nAtBat nHits nHome nRuns nRBI nBB
                  yrMajor crAtBat crHits crHome crRuns crRbi
                  crBB league division nOuts nAssts nError
         / clb;
   selection method=forward(select=sl sle=0.1);
run;
```

The SLE=0.1 option in the SELECTION statement specifies the significance level for entry. A candidate effect can enter the model at a certain selection step only if the following conditions are met:

- Its *p*-value is the smallest among all the valid candidate effects.
- Its *p*-value is smaller than 0.1 (the significance level for entry).

For more information about using significance levels in effect selection, see the section "Statistical Tests for Significance Level" on page 520.

Figure 13.7 Selection Information

The HPQUANTSELECT Procedure

Selection Information	
Selection Method	Forward
Select Criterion	Significance Level
Stop Criterion	Significance Level
Effect Hierarchy Enforced	None
Entry Significance Level (SLE)	0.1
Stop Horizon	1

Figure 13.7 displays the "Selection Information" table. The "Selection Information" provides details about the method and criteria used to perform the model selection. The requested selection method is the forward selection method where the decisions about what effects to add at any step and when to terminate the selection are both based on the significance level criterion.

Figure 13.8 Selection Summary

The HPQUANTSELECT Procedure

Quantile Level = 0.5

Selection Summary			
Step	**Effect Entered**	**Number Effects In**	**p Value**
0	**Intercept**	1	.
1	**CrHome**	2	<.0001
2	**nHits**	3	<.0001
3	**CrHits**	4	<.0001
4	**nOuts**	5	0.0185
5	**nAtBat**	6	0.0182
6	**Division**	7	0.0118
7	**nBB**	8	0.0647
8	**nRuns**	9	0.0558

Figure 13.8 displays the "Selection Summary" table. Each row in the "Selection Summary" table shows the effect that enters the model at the corresponding step of the effect selection process together with its p-value for adding the effect into the model at that step.

Figure 13.9 Stopping and Selection Reasons

Selection stopped because no candidate for entry is significant at the 0.1 level.

The model at step 8 is selected.

Figure 13.9 *continued*

The HPQUANTSELECT Procedure

Quantile Level = 0.5
Selected Model

Selected Effects:	Intercept nAtBat nHits nRuns nBB CrHits CrHome Division nOuts

Figure 13.9 displays the "Stop Reason," "Selection Reason," and "Selected Effects" tables. The "Stop Reason" and "Selection Reason" tables indicate that effect selection stopped because no candidate for entry was significant at the 0.1 level after step 8. The "Selected Effects" table lists the effects that are included in the selected model.

Figure 13.10 Details of the Selected Model

Fit Statistics	
Objective Function	26568
R1	0.39232
Adj R1	0.37318
AIC	2445.64547
AICC	2446.35693
SBC	2477.79485
ACL	101.01768

Parameter Estimates

Parameter	DF	Estimate	Standard Error	95% Confidence Limits	
Intercept	1	-130.65536	33.04546	-195.73336	-65.57735
nAtBat	1	-1.20522	0.41690	-2.02624	-0.38419
nHits	1	7.76667	1.81045	4.20127	11.33207
nRuns	1	-3.92180	1.87567	-7.61566	-0.22795
nBB	1	3.92049	1.04341	1.86565	5.97532
CrHits	1	0.17697	0.06856	0.04195	0.31198
CrHome	1	1.66939	0.73083	0.23012	3.10866
Division East	1	65.14327	24.48038	16.93289	113.35364
Division West	0	0	.	.	.
nOuts	1	0.23719	0.11403	0.01261	0.46176

The "Fit Statistics" and "Parameter Estimates" tables in Figure 13.10 give details of the final selected model. You can see that all nine effective parameters (excluding Division West) are significant at the 5% significance level, corresponding to the 95% confidence limits.

Like the sample median, a median regression model is robust to extreme observations, because it depends only on a small middle subset of all the observations in the data set. However, it is less representative of the entire conditional distribution of the response variable. You might want to further investigate the Sashelp.baseball data set at other quantile levels. The following statements select quantile regression models at the quantile levels 0.1 and 0.9, which correspond to the 10% and 90% conditional percentiles of the players' salaries:

```
proc hpquantselect data=sashelp.baseball alpha=0.1;
   class league division;
   model Salary = nAtBat nHits nHome nRuns nRBI nBB
                  yrMajor crAtBat crHits crHome crRuns crRbi
                  crBB league division nOuts nAssts nError
         / quantile=0.1 0.9 clb;
   selection method=backward(select=sl sls=0.1);
run;
```

The ALPHA=0.1 option in the PROC statement sets the significance level to 0.1. Combined with the CLB option in the MODEL statement, the ALPHA=0.1 option requests 90% confidence limits for parameter estimates. The QUANTILE= option in the MODEL statement specifies two quantile levels, 0.1 and 0.9, for fitting quantile regression models. The METHOD=BACKWARD option in the SELECTION statement specifies the backward elimination method of effect selection.

Figure 13.11 Parameter Estimates at Quantile Level 0.1

The HPQUANTSELECT Procedure

Quantile Level = 0.1
Selected Model

Selected Effects: Intercept nAtBat nHits nBB CrRuns CrBB Division nAssts

Parameter Estimates

Parameter	DF	Estimate	Standard Error	90% Confidence Limits	
Intercept	1	4.75224	18.94983	-26.53111	36.03558
nAtBat	1	-0.73670	0.17958	-1.03316	-0.44024
nHits	1	2.69396	0.63510	1.64551	3.74241
nBB	1	1.81807	0.40595	1.14791	2.48823
CrRuns	1	0.65476	0.09560	0.49694	0.81259
CrBB	1	-0.44622	0.16254	-0.71454	-0.17790
Division East	1	28.35406	10.68922	10.70774	46.00038
Division West	0	0	.	.	.
nAssts	1	0.14958	0.05897	0.05223	0.24694

Figure 13.11 displays the "Selected Effects" and "Parameter Estimates" tables at quantile level 0.1.

Figure 13.12 Parameter Estimates at Quantile Level 0.9

The HPQUANTSELECT Procedure

Quantile Level = 0.9
Selected Model

Selected Effects: Intercept nHits nBB CrAtBat CrHits CrHome CrRbi League Division nOuts

Figure 13.12 *continued*

Parameter Estimates					
Parameter	DF	Estimate	Standard Error	90% Confidence Limits	
Intercept	1	20.39804	58.17164	-75.63745	116.43353
nHits	1	2.30897	0.55640	1.39042	3.22752
nBB	1	3.09799	1.44414	0.71386	5.48213
CrAtBat	1	-0.44914	0.14651	-0.69101	-0.20727
CrHits	1	2.48064	0.51725	1.62672	3.33457
CrHome	1	6.29896	1.37134	4.03502	8.56291
CrRbi	1	-2.12293	0.76546	-3.38662	-0.85923
League American	1	-103.28955	33.68480	-158.89975	-47.67935
League National	0	0	.	.	.
Division East	1	107.46694	50.82797	23.55512	191.37876
Division West	0	0	.	.	.
nOuts	1	0.39766	0.12820	0.18601	0.60931

Figure 13.11 displays the "Selected Effects" and "Parameter Estimates" tables at quantile level 0.9.

You might want to compute the 90th percentile predictions for players' salaries and find out which players were overpaid based on the quantile regression model at quantile level 0.9. The following statements repeat the backward elimination method at quantile level 0.9, compute and sort the overpaid players' salaries, and output the observations for the top 10 overpaid players in the Sashelp.baseball data set:

```
proc hpquantselect data=sashelp.baseball alpha=0.1;
   id Name;
   class league division;
   model Salary = nAtBat nHits nHome nRuns nRBI nBB
                  yrMajor crAtBat crHits crHome crRuns crRbi
                  crBB league division nOuts nAssts nError
         / quantile=0.9 clb;
   selection method=backward(select=sl sls=0.1);
   output out=BaseballOverpaid copyvar=Salary r=Overpaid
          p=PredictedSalary lclm uclm;
run;

proc sort data=BaseballOverpaid;
   by descending Overpaid;
run;

proc print data=BaseballOverpaid(obs=10);
   var Name Salary Overpaid PredictedSalary lclm uclm;
run;
```

The ID statement adds the variable Name from the input data set to the BaseballOverpaid data set that is produced by the OUTPUT statement. The LCLM and UCLM options, respectively, request lower and upper bounds of $100(1-\alpha)\%$ confidence intervals for the expected conditional quantile predictions of players' salaries at quantile level 0.9.

Figure 13.13 Top 10 Overpaid Baseball Players at Quantile Level 0.9

Obs	Name	Salary	Overpaid	PredictedSalary	LCLM	UCLM
1	Smith, Ozzie	1940.0	1084.54	855.46	611.09	1099.83
2	Wiggins, Alan	700.0	220.74	479.26	345.93	612.60
3	Murray, Eddie	2460.0	213.10	2246.90	1982.06	2511.74
4	Strawberry, Darryl	1220.0	187.64	1032.36	914.99	1149.72
5	Gibson, Kirk	1300.0	177.24	1122.76	1011.97	1233.56
6	Trevino, Alex	512.5	149.70	362.80	273.17	452.43
7	Ramirez, Rafael	875.0	141.09	733.91	599.52	868.31
8	Romero, Ed	375.0	128.71	246.29	144.88	347.70
9	Mattingly, Don	1975.0	124.82	1850.18	1557.81	2142.55
10	Puhl, Terry	900.0	104.34	795.66	625.51	965.81

Output 13.13 shows the information about the top 10 overpaid players according to the final selected quantile regression model at quantile level 0.9. Ozzie Smith is in first place. This might be because, although Smith was known for his defensive brilliance, the model weights offensive performance measures much more than defensive performance measures.

Syntax: HPQUANTSELECT Procedure

The following statements are available in the HPQUANTSELECT procedure:

```
PROC HPQUANTSELECT < options > ;
   BY variables ;
   CLASS variable < (options) >. . . < variable < (options) > > < / global-options > ;
   CODE < options > ;
   ID variables ;
   MODEL dependent = < effects > < / model-options > ;
   OUTPUT < OUT=SAS-data-set >
            < keyword < =name > >. . .
            < keyword < =name > > < / options > ;
   PARTITION < partition-options > ;
   PERFORMANCE performance-options ;
   SELECTION selection-options ;
   WEIGHT variable ;
```

The PROC HPQUANTSELECT statement and a single MODEL statement are required. All other statements are optional. The CLASS statement can appear multiple times. If a CLASS statement is specified, it must precede the MODEL statement.

The rest of this section provides detailed syntax information about each of the preceding statements, beginning with the PROC HPQUANTSELECT statement. The remaining statements are described in alphabetical order.

PROC HPQUANTSELECT Statement

PROC HPQUANTSELECT < *options* > **;**

The PROC HPQUANTSELECT statement invokes the HPQUANTSELECT procedure. Table 13.1 summarizes the options in the PROC HPQUANTSELECT statement by function.

Table 13.1 PROC HPQUANTSELECT Statement Options

Option	Description
Basic Options	
DATA=	Specifies the input data set
MAXMACRO=	Specifies the maximum number of macro variables to produce
NAMELEN=	Limits the length of effect names
Options Related to Output	
NOCLPRINT	Limits or suppresses the display of CLASS levels
NOPRINT	Suppresses ODS output
User-Defined Formats	
FMTLIBXML=	Specifies a file reference for a format stream
Other Options	
ALPHA=	Sets the significance level to use for the construction of confidence intervals
SEED=	Sets the seed used for pseudorandom number generation

The following list describes these *options* in alphabetical order:

ALPHA=*number*

sets the significance level to use for the construction of confidence intervals. The value must be between 0 and 1; the default value of 0.05 results in 95% intervals. This option affects the STDP, LCLM, and UCLM keywords in the OUTPUT statement and the CLB option in the MODEL statement.

DATA=*SAS-data-set*

names the input SAS data set to be used by PROC HPQUANTSELECT. The default is the most recently created data set.

If PROC HPQUANTSELECT executes in distributed mode, the input data are distributed to memory on the appliance nodes and analyzed in parallel, unless the data are already distributed in the appliance database. In that case, PROC HPQUANTSELECT reads the data alongside the distributed database. For more information, see the section "Processing Modes" on page 10 about the various execution modes and the section "Alongside-the-Database Execution" on page 18 about the alongside-the-database model. Both sections are in Chapter 3, "Shared Concepts and Topics."

FMTLIBXML=*file-ref*
specifies the file reference for the XML stream that contains the user-defined format definitions. User-defined formats are handled differently in a distributed computing environment than they are in other SAS products. For more information about how to generate an XML stream for your formats, see the section "Working with Formats" on page 35 in Chapter 3, "Shared Concepts and Topics."

MAXMACRO=*n*
specifies the total maximum number of macro variables to produce. Each macro variable contains selected effects for a selected model. For more information about the macro variables, see the section "Macro Variables That Contain Selected Effects" on page 524. By default, MAXMACRO=100.

NAMELEN=*number*
specifies the length to which long effect names are to be shortened. The default and minimum value is 20.

NOCLPRINT< =*number* **>**
suppresses the display of the "Class Level Information" table if you do not specify *number*. If you specify *number*, the values of the classification variables are displayed for only those variables whose number of levels is less than *number*. Specifying *number* helps reduce the size of the "Class Level Information" table if some classification variables have a large number of levels.

NOPRINT
suppresses the generation of ODS output.

SEED=*number*
specifies an integer to be used to start the pseudorandom number generator for random partitioning of data for training, testing, and validation. If you do not specify a seed, or if you specify a value less than or equal to 0, the seed is generated from reading the time of day from the computer's clock.

BY Statement

BY *variables* **;**

You can specify a BY statement with PROC HPQUANTSELECT to obtain separate analyses of observations in groups that are 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 you specify more than one BY statement, only the last one specified is used.

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 HPQUANTSELECT 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 (in Base SAS software).

Processing of BY statements is not supported when the HPQUANTSELECT procedure runs alongside the database or alongside the Hadoop Distributed File System (HDFS). These modes are used if the input data are stored in a database or HDFS and the grid host is the appliance that houses the data.

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

CLASS Statement

CLASS *variable* <(*options*)>. . . <*variable* <(*options*)>> </ *global-option*> ;

The CLASS statement names the classification variables to be used as explanatory variables in the analysis. The CLASS statement must precede the MODEL statement.

The CLASS statement for SAS high-performance analytical procedures is documented in the section "CLASS Statement" on page 42 of Chapter 4, "Shared Statistical Concepts." The HPQUANTSELECT procedure also supports the following *global-option* in the CLASS statement:

UPCASE
uppercases the values of character-valued CLASS variables before levelizing them. For example, if you specify the UPCASE option and a CLASS variable can take the values "a," "A," and "b," then "a" and "A" represent the same level and the CLASS variable is treated as having only two values: "A" and "B."

CODE Statement

CODE <*options*> ;

The CODE statement writes SAS DATA step code for computing predicted values of the fitted model either to a file or to a catalog entry. This code can then be included in a DATA step to score new data.

Table 13.2 summarizes the *options* available in the CODE statement.

Table 13.2 CODE Statement Options

Option	Description
CATALOG=	Names the catalog entry where the generated code is saved
DUMMIES	Retains the dummy variables in the data set
ERROR	Computes the error function
FILE=	Names the file where the generated code is saved
FORMAT=	Specifies the numeric format for the regression coefficients
GROUP=	Specifies the group identifier for array names and statement labels
IMPUTE	Imputes predicted values for observations with missing or invalid covariates
LINESIZE=	Specifies the line size of the generated code
LOOKUP=	Specifies the algorithm for looking up CLASS levels
RESIDUAL	Computes residuals

For details about the syntax of the CODE statement, see the section "CODE Statement" (Chapter 19, *SAS/STAT User's Guide*) in Chapter 19, "Shared Concepts and Topics" (*SAS/STAT User's Guide*).

ID Statement

ID *variables* **;**

The ID statement lists one or more variables from the input data set that are transferred to output data sets created by SAS High-Performance Analytics procedures, provided that the output data set produces one (or more) records per input observation.

For information about the common ID statement in SAS high-performance analytical procedures, see the section "ID Statement" on page 46 in Chapter 4, "Shared Statistical Concepts."

MODEL Statement

MODEL *dependent*=< *effects* > / < *options* > **;**

The MODEL statement names the dependent variable and the explanatory effects, including covariates, main effects, interactions, and nested effects. If you omit the explanatory effects, PROC HPQUANTSELECT fits an intercept-only model.

After the keyword MODEL, the dependent (response) variable is specified, followed by an equal sign. The explanatory effects follow the equal sign. For information about constructing the model effects, see the section "Specification and Parameterization of Model Effects" on page 55 of Chapter 4, "Shared Statistical Concepts."

You can specify the following *options* in the MODEL statement after a slash (/):

CLB
requests the $100(1-\alpha)\%$ upper and lower confidence limits for the parameter estimates. By default, the 95% limits are computed; you can use the ALPHA= option in the PROC HPQUANTSELECT statement to change the α level.

INCLUDE=*n*

INCLUDE=*single-effect*

INCLUDE=(*effects***)**
forces effects to be included in all models. If you specify INCLUDE=*n*, then the first *n* effects that are listed in the MODEL statement are included in all models. If you specify INCLUDE=*single-effect* or if you specify a list of effects within parentheses, then the specified effects are forced into all models. The effects that you specify in the INCLUDE= option must be explanatory effects that are defined in the MODEL statement.

NOINT
suppresses the intercept term that is otherwise included in the model.

ORDERSELECT
specifies that, for the selected model, effects be displayed in the order in which they first entered the model. If you do not specify this option, then effects in the selected model are displayed in the order in which they appear in the MODEL statement.

QUANTILES=*number-list*

QUANTILE=*number-list*
specifies the quantile levels for the quantile regression. You can specify any number of quantile levels in $(0, 1)$. If you do not specify this option, the HPQUANTSELECT procedure performs median regression effect selection that corresponds to QUANTILE=0.5.

SPARSITY(< BF | HS > < IID >)
specifies the suboptions for estimating the sparsity function. You can specify the Bofinger method by using the BF suboption or the Hall-Sheather method by using the HS suboption. By default, the Hall-Sheather method is used. You can also specify the IID suboption to assume that the quantile regression errors satisfy the independently and identically distributed (iid) assumption. Let f_i and F_i, respectively, denote the probability density function and the cumulative distribution function of the ith error for $i = 1, \ldots, n$. The iid assumption means that there exist f and F such that $f = f_1 = \cdots = f_n$ and $F = F_1 = \cdots = F_n$. If you specify the IID option, the covariance matrix of the parameter estimates, $\omega^2(\tau, F)(\mathbf{X}'\mathbf{X})^-$, is adopted for computing the confidence limits and the Wald statistics, where $\omega^2(\tau, F) = \tau(1-\tau)/f^2(F^{-1}(\tau))$. By default, the covariance matrix of the parameter estimates is non-iid and takes the sandwich form: $n^{-2}\tau(1-\tau)\mathbf{H}_n^-(\mathbf{X}'\mathbf{X})\mathbf{H}_n^-$, where $\mathbf{H}_n = n^{-1}\sum_{i=1}^n f_i(F_i^{-1}(\tau))\mathbf{x}_i\mathbf{x}_i'$. For more information, see the section "Details: HPQUANTSELECT Procedure" on page 517.

START=*n*

START=*single-effect*

START=(*effects***)**
begins the effect-selection process in the forward and stepwise selection methods from the initial model that you designate. If you specify START=*n*, then the starting model consists of the first *n* effects listed in the MODEL statement. If you specify START=*single-effect* or if you specify a list of effects within parentheses, then the starting model consists of these specified effects. The effects that you specify in the START= option must be explanatory effects defined in the MODEL statement. The START= option is not available when you specify METHOD=BACKWARD in the SELECTION statement.

STB
produces standardized regression coefficients. A standardized regression coefficient is computed by dividing a parameter estimate by the ratio of the sample standard deviation of the dependent variable to the sample standard deviation of the regressor.

TOL
produces tolerance values for the estimates. Tolerance for a parameter is defined as $1 - R^2$, where R^2 is obtained from the ordinary least squares regression of the parameter on all other parameters in the model.

VIF
produces variance inflation factors in the parameter estimates table. Variance inflation is the reciprocal of tolerance.

OUTPUT Statement

OUTPUT < **OUT=***SAS-data-set* >
< **COPYVARS=(***variables***)** >
< *keyword* < **=***name* > >. . . < *keyword* < **=***name* > > **;**

The OUTPUT statement creates a data set that contains observationwise statistics, which are computed after the final selected model is fit. To avoid data duplication for large data sets, the variables in the input data set are *not* included in the output data set; however, variables that are specified in the ID statement or COPYVARS= option are included.

If the input data are in distributed form, where access of data in a particular order cannot be guaranteed, the HPQUANTSELECT procedure copies the distribution or partition key to the output data set so that its contents can be joined with the input data.

The output statistics are computed based on the parameter estimates for the selected model. If you specify multiple quantile levels by using the QUANTILE= option in the MODEL statement, then for each appropriate keyword that is specified in the OUTPUT statement, one variable is generated for each specified quantile level. These variables appear in the sorted order of the specified quantile levels. For example, the following statements generate the Out data set, which contains the two predicted quantile variables p1 and p2:

```
proc hpquantselect data=one;
   model y = x1-x4 /quantile=0.5 0.3;
   output out=out pred=p;
run;
```

The variable p1 is for quantile level 0.3, and the variable p2 is for quantile level 0.5, because the sorted quantile levels are (0.3 0.5), not (0.5 0.3).

You can specify the following options in the OUTPUT statement:

OUT=*SAS-data-set*

DATA=*SAS-data-set*

specifies the name of the output data set. If you omit the OUT= (or DATA=) option, PROC HPQUANTSELECT uses the DATA*n* convention to name the output data set.

COPYVAR=*variable*

COPYVARS=(*variables***)**

transfers one or more *variables* from the input data set to the output data set. Variables that you name in an ID statement are also copied from the input data set to the output data set.

keyword **< =***name* **>**

specifies the statistics to include in the output data set and optionally names the new variables that contain the statistics. Specify a *keyword* for each desired statistic (see the following list of *keywords*), followed optionally by an equal sign and a variable to contain the statistic.

If you specify *keyword***=***name*, the new variable that contains the requested statistic has the specified name. If you omit the optional **=***name* after a *keyword*, then a default name is used.

You can specify the following *keywords* to request statistics that are available with for selection methods:

LCLM
requests the lower bound of a $100(1-\alpha)\%$ confidence interval for the expected quantile of the dependent variable. The default variable name is LCLM.

PREDICTED
PRED
P
requests predicted values for the response variable. The default variable name is Pred.

RESIDUAL
RESID
R
requests the residual, calculated as ACTUAL – PREDICTED. The default variable name is Residual.

ROLE
requests a numeric variable that indicates the role that each observation plays in fitting the model. The default variable name is _ROLE_. For each observation, the interpretation of this variable is shown in Table 13.3.

Table 13.3 Role Interpretation

Value	Observation Role
0	Not used
1	Training
2	Validation
3	Testing

If you do not partition the input data by using a PARTITION statement, then the role variable value is 1 for observations that are used in fitting the model, and 0 for observations that have at least one missing or invalid value for the response, regressor, or weight variables.

You can use the following statements to display the values of the _ROLE_ variable in the output data set:

```
proc format;
   value role 0 = 'Not Used'
              1 = 'Training'
              2 = 'Validation'
              3 = 'Testing';
run;

proc freq;
   tables _role_;
   format _role_ role.;
run;
```

STDP
requests standard error of the mean predicted quantiles. The default variable name is STDP.

UCLM
requests the upper bound of a $100(1-\alpha)\%$ confidence interval for the expected quantile of the dependent variable. The default variable name is UCLM.

PARTITION Statement

PARTITION *partition-options* **;**

The PARTITION statement specifies how observations in the input data set are logically partitioned into disjoint subsets for model training, validation, and testing. Either you can designate a variable in the input data set and a set of formatted values of that variable to determine the role of each observation, or you can specify proportions to use for random assignment of observations for each role.

You can specify either of the following mutually exclusive *partition-options*:

FRACTION(< TEST=*fraction* **> < VALIDATE=***fraction* **>)**
randomly assigns the specified proportions of observations in the input data set to testing, validation, and training roles. You specify the proportions for testing and validation by using the TEST= and VALIDATE= suboptions. If you specify both the TEST= and VALIDATE= suboptions, then the sum of the specified fractions must be less than 1 and the remaining fraction of the observations are assigned to the training role.

ROLEVAR | ROLE=*variable***(< TEST='***value***' > < TRAIN='***value***' > < VALIDATE='***value***' >)**
names the variable in the input data set whose values are used to assign roles to each observation. Use the TEST=, TRAIN=, and VALIDATE= suboptions to specify the formatted values of this variable that are used to assign observation roles are specified in the TEST=, TRAIN=, and VALIDATE= suboptions. If you do not specify the TRAIN= suboption, then all observations whose roles are not determined by the TEST= and VALIDATE= suboptions are assigned to training.

To create an output data set variable that indicates the role assignment for either *partition-option*, specify the ROLE=*variable* option in the OUTPUT statement.

PERFORMANCE Statement

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

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

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

The PERFORMANCE statement for SAS high-performance analytical procedures is documented in the section "PERFORMANCE Statement" on page 36 of Chapter 3, "Shared Concepts and Topics."

SELECTION Statement

SELECTION < *options* > **;**

The SELECTION statement performs variable selection. The statement is fully documented in the section "SELECTION Statement" on page 47 in Chapter 4, "Shared Statistical Concepts."

The HPQUANTSELECT procedure supports the following suboptions in the METHOD= option in the SELECTION statement to specify the corresponding effect selection methods:

NONE	specifies no model selection.
FORWARD	specifies the forward selection method, which starts with no effects in the model and adds effects.
BACKWARD	specifies the backward elimination method, which starts with all effects in the model and deletes effects.
STEPWISE	specifies the stepwise regression method, which is similar to the forward selection method except that effects already in the model do not necessarily stay there.

By default, the METHOD=STEPWISE option is used in the SELECTION statement. If you do not use the SELECTION statement, the HPQUANTSELECT procedure fits the full model that is specified by the MODEL statement; this is equivalent to specifying the METHOD=NONE option in the SELECTION statement. For information about all the selection criteria that are used in PROC HPQUANTSELECT, see the section "Criteria Used in Model Selection" on page 519.

The DETAILS=ALL and DETAILS=STEPS options produce "Fit Statistics" and "Parameter Estimates" tables, which provide information about the model that is selected at each step of the selection process.

TEST=*value*

specifies a method to compute significance levels for the selection process. Table 13.4 summarizes these methods.

Table 13.4 Options for Significance Levels

Value of TEST=	Method
LR1	Likelihood ratio test Type I
LR2	Likelihood ratio test Type II
WALD	Wald score test

By default, PROC HPQUANTSELECT uses the Wald score to compute significance levels. If you specify the IID suboption in the SPARSITY option of the MODEL statement, the Wald score test uses the iid form of the covariance matrix to compute the Wald score and the associated significance levels. Otherwise, the non-iid form of the covariance matrix is used. The sparsity functions for both Type I and Type II likelihood ratio tests are estimated under the iid assumption no matter whether you specify the IID suboption.

WEIGHT Statement

WEIGHT *variable* **;**

The *variable* in the WEIGHT statement is used as a weight to perform a weighted analysis of the data. Observations that have nonpositive or missing weights are not included in the analysis. If you do not use a WEIGHT statement, all observations that are used in the analysis are assigned a weight of 1.

Details: HPQUANTSELECT Procedure

Quantile Regression

This section describes the basic concepts and notations for quantile regression and quantile regression model selection.

Let $\{(y_i, \mathbf{x}_i) : i = 1, \ldots, n\}$ denote a data set of observations, where y_i are responses and $\mathbf{x}_i$ are regressors. Koenker and Bassett (1978) define the *regression quantile* at quantile level $\tau \in (0, 1)$ as any solution to the minimization problem

$$\min_{\boldsymbol{\beta} \in \mathbf{R}^p} \sum_{i=1}^{n} \rho_\tau \left(y_i - \mathbf{x}_i' \boldsymbol{\beta} \right)$$

where $\rho_\tau(r) = \tau r^+ + (1 - \tau) r^-$ is a check loss function in which $r^+ = \max(r, 0)$ and $r^- = \max(-r, 0)$.

If you specify weights $w_i, i = 1, \ldots, n$, in the WEIGHT statement, then weighted quantile regression is carried out by solving

$$\min_{\boldsymbol{\beta} \in \mathbf{R}^p} \sum_{i=1}^{n} \rho_\tau \left(w_i (y_i - \mathbf{x}_i' \boldsymbol{\beta}) \right)$$

The HPQUANTSELECT procedure fits a quantile regression model by using a predictor-corrector interior point algorithm, which was originally designed to solve support vector machine classifiers for large data sets (Gertz and Griffin 2005, 2010).

Linear Model with iid Errors

You can specify the SPARSITY(IID) option in the MODEL statement to assume that the distribution of Y_i conditional on $\mathbf{x}_i$ follows the linear model

$$Y_i = \mathbf{x}_i' \boldsymbol{\beta} + \epsilon_i$$

where ϵ_i for $i = 1, \ldots, n$ are iid in the distribution function F. Let $f = F'$ denote the density function of F. Further assume that $f(F^{-1}(\tau)) > 0$ in a neighborhood of τ. Then, under some mild conditions, Koenker and Bassett (1982) prove that the asymptotic distribution of the quantile regression estimates is

$$\sqrt{n}(\hat{\boldsymbol{\beta}}(\tau) - \boldsymbol{\beta}(\tau)) \rightarrow N(0, \omega^2(\tau, F) \boldsymbol{\Omega}^{-1})$$

where $\omega^2(\tau, F) = \tau(1-\tau)/f^2(F^{-1}(\tau))$ and $\mathbf{\Omega} = \lim_{n\to\infty} n^{-1}\sum \mathbf{x}_i\mathbf{x}_i'$. The reciprocal of the density function, $s(\tau) = 1/f(F^{-1}(\tau))$, is called the *sparsity function*.

Accordingly, the covariance matrix of $\hat{\boldsymbol{\beta}}(\tau)$ can be estimated as

$$\hat{\Sigma}(\tau) = \tau(1-\tau)\hat{s}^2(\tau)(\mathbf{X}'\mathbf{X})^-$$

where $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)'$ is the design matrix and $\hat{s}(\tau)$ is an estimate of $s(\tau)$. Under the iid assumption, the algorithm for computing $\hat{s}(\tau)$ is as follows:

1. Fit a quantile regression model and compute the residuals. Each residual $r_i = y_i - \mathbf{x}_i'\hat{\boldsymbol{\beta}}(\tau)$ can be viewed as an estimated realization of the corresponding error ϵ_i.
2. Compute the quantile level bandwidth h_n. The HPQUANTSELECT procedure provides two bandwidth methods:
   - The Bofinger bandwidth is an optimizer of mean squared error for standard density estimation:
     $$h_n = n^{-1/5}(4.5v^2(\tau))^{1/5}$$
   - The Hall-Sheather bandwidth is based on Edgeworth expansions for studentized quantiles,
     $$h_n = n^{-1/3}z_\alpha^{2/3}(1.5v(\tau))^{1/3}$$
     z_α satisfies $T(z_\alpha, df) = 1-\alpha/2$ for the construction of $1-\alpha$ confidence intervals, where T is the cumulative distribution function for the t distribution and df is the residual degrees of freedom.

   The quantity
   $$v(\tau) = \frac{s(\tau)}{s^{(2)}(\tau)} = \frac{f^2}{2(f^{(1)}/f)^2 + [(f^{(1)}/f)^2 - f^{(2)}/f]}$$
   is not sensitive to f and can be estimated by assuming f is Gaussian as
   $$\hat{v}(\tau) = \frac{\exp(-q^2)}{2\pi(q^2+1)}$$
   where $q = \Phi^{-1}(\tau)$.
3. Compute residual quantiles $\hat{F}^{-1}(\tau_0)$ and $\hat{F}^{-1}(\tau_1)$ as follows:
   a) Set $\tau_0 = \max(0, \tau - h_n)$ and $\tau_1 = \min(1, \tau + h_n)$.
   b) Use the equation
   $$\hat{F}^{-1}(t) = \begin{cases} r_{(1)} & \text{if } t \in [0, 1/2n) \\ \lambda r_{(i+1)} + (1-\lambda)r_{(i)} & \text{if } t \in [(i-0.5)/n, (i+0.5)/n) \\ r_{(n)} & \text{if } t \in [(2n-1), 1] \end{cases}$$
   where $r_{(i)}$ is the ith smallest residual and $\lambda = t - (i-0.5)/n$.
   c) If $\hat{F}^{-1}(\tau_0) = \hat{F}^{-1}(\tau_1)$, find i that satisfies $r_{(i)} < \hat{F}^{-1}(\tau_0)$ and $r_{(i+1)} \geq \hat{F}^{-1}(\tau_0)$. If such an i exists, reset $\tau_0 = (i-0.5)/n$ so that $\hat{F}^{-1}(\tau_0) = r_{(i)}$. Also find j that satisfies $r_{(j)} > \hat{F}^{-1}(\tau_1)$ and $r_{(j-1)} \leq \hat{F}^{-1}(\tau_1)$. If such a j exists, reset $\tau_1 = (j-0.5)/n$ so that $\hat{F}^{-1}(\tau_1) = r_{(j)}$.
4. Estimate the sparsity function $s(\tau)$ as
   $$\hat{s}(\tau) = \frac{\hat{F}^{-1}(\tau_1) - \hat{F}^{-1}(\tau_0)}{\tau_1 - \tau_0}$$

Linear-in-Parameter Model with Non-iid Settings

The general form of a linear quantile regression model is

$$Q_Y(\tau|\mathbf{x}) = \mathbf{x}'\boldsymbol{\beta}(\tau)$$

where the iid assumption is not necessary. Under some regularity conditions, the asymptotic distribution of the general form of quantile regression estimates is

$$\sqrt{n}(\hat{\boldsymbol{\beta}}(\tau) - \boldsymbol{\beta}(\tau)) \rightarrow N(0, \tau(1-\tau)\mathbf{H}_n^{-}\boldsymbol{\Omega}\mathbf{H}_n^{-})$$

where $\mathbf{H}_n = \lim_{n\rightarrow\infty} n^{-1}\sum \mathbf{x}_i\mathbf{x}_i' f_i(F_i^{-1}(\tau))$.

Accordingly, the covariance matrix of $\hat{\boldsymbol{\beta}}(\tau)$ can be estimated as

$$\hat{\Sigma}(\tau) = n^{-2}\tau(1-\tau)\hat{\mathbf{H}}_n^{-}(\mathbf{X}'\mathbf{X})\hat{\mathbf{H}}_n^{-}$$

where $\hat{\mathbf{H}}_n = n^{-1}\sum\left(\mathbf{x}_i\mathbf{x}_i'/\hat{s}_i(\tau)\right)$.

The sparsity value of the ith observation, $\hat{s}_i(\tau)$, can be estimated as

$$\hat{s}_i(\tau) = \frac{\hat{F}_i^{-1}(\tau + h_n) - \hat{F}_i^{-1}(\tau - h_n)}{2h_n}$$

where $\hat{F}_i^{-1}(\tau \pm h_n) = \mathbf{x}_i'\hat{\boldsymbol{\beta}}(\tau \pm h_n)$ are the ith predicted quantile values at quantile levels $(\tau \pm h_n)$.

Criteria Used in Model Selection

The HPQUANTSELECT procedure supports the following fit statistics that you can use as criteria for the CHOOSE=, SELECT=, and STOP= options in the SELECTION statement:

ADJR1	specifies the adjusted R1 statistic.
AIC	specifies Akaike's information criterion (Akaike 1969; Koenker 2005).
AICC	specifies the corrected Akaike's information criterion (Hurvich and Tsai 1989).
BIC | SBC	specifies the Schwarz Bayesian information criterion (Schwarz 1978; Koenker 2005).
R1	specifies the R1 statistic (Koenker and Machado 1999). The R1 statistic is not valid for the STOP= or CHOOSE= option.
SL	specifies the significance level that is used to assess an effect's contribution to the fit when it is added to or removed from a model. SL is not valid for the CHOOSE= option.
VALIDATE	specifies the average check loss over the validation data.

Quasi-likelihood Information Criteria

Given the quantile level τ, assume that the distribution of Y_i conditional on $\mathbf{x}_i$ follows the linear model

$$Y_i = \mathbf{x}_i'\boldsymbol{\beta} + \epsilon_i$$

where ϵ_i for $i = 1, \ldots, n$ are iid in distribution F. Further assume that F is an asymmetric Laplace distribution whose density function is

$$f_\tau(r) = \frac{\tau(1-\tau)}{\sigma} \exp\left(-\frac{\rho_\tau(r)}{\sigma}\right)$$

where σ is the scale parameter. Then, the negative log-likelihood function is

$$l_\tau(\boldsymbol{\beta}, \sigma) = n \log(\sigma) + \sigma^{-1} \sum_{i=1}^{n} \rho_\tau(y_i - \mathbf{x}_i' \boldsymbol{\beta}) - n \log(\tau(1-\tau))$$

Under these settings, the maximum likelihood estimate (MLE) of $\boldsymbol{\beta}$ is the same as the relevant level-τ quantile regression solution $\hat{\boldsymbol{\beta}}(\tau)$, and the MLE for σ is

$$\hat{\sigma}(\tau) = n^{-1} \sum_{i=1}^{n} \rho_\tau\left(y_i - \mathbf{x}_i' \hat{\boldsymbol{\beta}}(\tau)\right)$$

where $\hat{\sigma}(\tau)$ equals the level-τ average check loss $\mathrm{ACL}(\tau)$ for the quantile regression solution.

Because the general form of Akaike's information criterion (AIC) is $\mathrm{AIC} = (-2l + 2p)$, the quasi-likelihood AIC for quantile regression is

$$\mathrm{AIC}(\tau) = 2n \ln\left(\mathrm{ACL}(\tau)\right) + 2p$$

where p is the degrees of freedom for the fitted model.

Similarly, the quasi-likelihood AICC (corrected AIC) and SBC (Schwarz Bayesian information criterion) can be formulated as follows:

$$\mathrm{AICC}(\tau) = 2n \ln\left(\mathrm{ACL}(\tau)\right) + \frac{2pn}{n-p-1}$$

$$\mathrm{SBC}(\tau) = 2n \ln\left(\mathrm{ACL}(\tau)\right) + p \ln(n)$$

In fact, the quasi-likelihood AIC, AICC, and SBC are fairly robust, and you can use them to select effects for data sets without the iid assumption in asymmetric Laplace distribution. For a simulation study that applies SBC for effect selection, see "Example 13.1: Simulation Study" on page 532. The study generates a data set by using a naive instrumental model (Chernozhukov and Hansen 2008).

Statistical Tests for Significance Level

The HPQUANTSELECT procedure supports the significance level (SL) criterion for effect selection. Consider the general form of a linear quantile regression model:

$$Q_Y(\tau | \mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1' \boldsymbol{\beta}_1(\tau) + \mathbf{x}_2' \boldsymbol{\beta}_2(\tau)$$

At each step of an effect-selection process, a candidate effect can be represented as $\mathbf{x}_2$, and the significance level of the candidate effect can be calculated by testing the null hypothesis: $H_0 : \boldsymbol{\beta}_2(\tau) = \mathbf{0}$.

When you use SL as a criterion for effect selection, you can further use the TEST= option in the SELECTION statement to specify a statistical test method to compute the significance-level values as follows:

- The TEST=WALD option specifies the Wald test. Let $\hat{\boldsymbol{\beta}}(\tau) = \left(\hat{\boldsymbol{\beta}}_1'(\tau), \hat{\boldsymbol{\beta}}_2'(\tau)\right)'$ be the parameter estimates for the extended model, and denote the estimated covariance matrix of $\hat{\boldsymbol{\beta}}(\tau)$ as

$$\hat{\Sigma}(\tau) = \begin{bmatrix} \hat{\Sigma}_{11}(\tau) & \hat{\Sigma}_{12}(\tau) \\ \hat{\Sigma}_{21}(\tau) & \hat{\Sigma}_{22}(\tau) \end{bmatrix}$$

  where $\hat{\Sigma}_{22}(\tau)$ is the covariance matrix for $\hat{\boldsymbol{\beta}}_2(\tau)$. Then the Wald test score is defined as

$$\hat{\boldsymbol{\beta}}_2'(\tau)\hat{\Sigma}_{22}^{-1}(\tau)\hat{\boldsymbol{\beta}}_2(\tau)$$

  If you specify the SPARSITY(IID) option in the MODEL statement, $\hat{\Sigma}(\tau)$ is estimated under the iid errors assumption. Otherwise, $\hat{\Sigma}(\tau)$ is estimated by using non-iid settings. For more information about the linear model with iid errors and non-iid settings, see the section "Quantile Regression" on page 517.

- The TEST=LR1 or TEST=LR2 option specifies the Type I or Type II quasi-likelihood ratio test, respectively. Under the iid assumption, Koenker and Machado (1999) propose two types of quasi-likelihood ratio tests for quantile regression, where the error distribution is flexible but not limited to the asymmetric Laplace distribution. The Type I test score, LR1, is defined as

$$\frac{2(D_1(\tau) - D_2(\tau))}{\tau(1-\tau)\hat{s}}$$

  where $D_1(\tau) = \sum \rho_\tau \left(y_i - \mathbf{x}_{1i}\hat{\boldsymbol{\beta}}_{1_1}(\tau)\right)$ is the sum of check losses for the reduced model, $D_2(\tau) = \sum \rho_\tau \left(y_i - \mathbf{x}_{1i}\hat{\boldsymbol{\beta}}_{1_2}(\tau) - \mathbf{x}_{2i}\hat{\boldsymbol{\beta}}_2(\tau)\right)$ is the sum of check losses for the extended model, and $\hat{s}$ is the estimated sparsity function. The Type II test score, LR2, is defined as

$$\frac{2D_2(\tau)\left(\log(D_1(\tau)) - \log(D_2(\tau))\right)}{\tau(1-\tau)\hat{s}}$$

Under the null hypothesis that the reduced model is the true model, the Wald score, LR1 score, and LR2 score all follow a χ^2 distribution with degrees of freedom $df = df_2 - df_1$, where df_1 and df_2 are the degrees of freedom for the reduced model and the extended model, respectively .

When you use SL as a criterion for effect selection, the algorithm for estimating sparsity function depends on whether an effect is being considered as an add or a drop candidate. For testing an add candidate effect, the sparsity function, which is $s(\tau)$ under the iid error assumption or $s_i(\tau)$ for non-iid settings, is estimated on the reduced model that does not include the add candidate effect. For testing a drop candidate effect, the sparsity function is estimated on the extended model that does not exclude the drop candidate effect. Then, these estimated sparsity function values are used to compute LR1 or LR2 and the covariance matrix of the parameter estimates for the extended model. However, for the model that is selected at each step, the sparsity function for estimating standard errors and confidence limits of the parameter estimates is estimated on that model itself, but not on the model that was selected at the preceding step.

Because the null hypotheses usually do not hold, the SLENTRY and SLSTAY values cannot reliably be viewed as probabilities. One way to address this difficulty is to replace hypothesis testing as a means of selecting a model with information criteria or out-of-sample prediction criteria.

Table 13.5 provides formulas and definitions for these fit statistics.

Table 13.5 Formulas and Definitions for Model Fit Summary Statistics for Single Quantile Effect Selection

Statistic	Definition or Formula
n	Number of observations
p	Number of parameters, including the intercept
$r_i(\tau)$	Residual for the ith observation; $r_i(\tau) = y_i - \mathbf{x}_i\hat{\boldsymbol{\beta}}(\tau)$
$D(\tau)$	Total sum of check losses; $D(\tau) = \sum_{i=1}^{n} \rho_\tau(r_i)$. $D(\tau)$ is labeled as Objective Function in the "Fit Statistics" table.
$D_0(\tau)$	Total sum of check losses for intercept-only model if the intercept is a forced-in effect; otherwise for empty model.
$\mathrm{ACL}(\tau)$	Average check loss; $\mathrm{ACL}(\tau) = \dfrac{D(\tau)}{n}$
$\mathrm{R1}(\tau)$	Counterpart of linear regression R square for quantile regression; $\mathrm{R1}(\tau) = 1 - \dfrac{D(\tau)}{D_0(\tau)}$
$\mathrm{ADJR1}(\tau)$	Adjusted R1; $1 - \dfrac{(n-1)D(\tau)}{(n-p)D_0(\tau)}$ if intercept is a forced-in effect; otherwise $1 - \dfrac{nD(\tau)}{(n-p)D_0(\tau)}$.
$\mathrm{AIC}(\tau)$	$2n\ln(\mathrm{ACL}(\tau)) + 2p$
$\mathrm{AICC}(\tau)$	$2n\ln(\mathrm{ACL}(\tau)) + \dfrac{2pn}{n-p-1}$
$\mathrm{SBC}(\tau)$	$2n\ln(\mathrm{ACL}(\tau)) + p\ln(n)$

The $\mathrm{ADJR1}(\tau)$ criterion is equivalent to the generalized approximate cross validation (GACV) criterion for quantile regression (Yuan 2006). The GACV criterion is defined as

$$\mathrm{GACV}(\tau) = D(\tau)/(n-p)$$

which is proportional to $1 - \mathrm{ADJR1}(\tau)$.

Diagnostic Statistics

This section gathers the formulas for the statistics available in the OUTPUT statement. All the statistics available in the OUTPUT statement are conditional on the selected model and do not take into account the variability that is introduced by doing model selection.

The model to be fit is $Q_Y(\tau|\mathbf{x}) = \mathbf{x}'\boldsymbol{\beta}(\tau)$, and the parameter estimate $\hat{\boldsymbol{\beta}}(\tau)$ is the solution that minimizes $\sum_{i=1}^{n} \rho_\tau\left(y_i - \mathbf{x}_i'\boldsymbol{\beta}\right)$. The subscript i is for the ith observation. The subscript j is for the jth-smallest quantile level among all the specified QUANTILE= levels in the MODEL statement. $\hat{\Sigma}(\tau)$ denotes the covariance estimation for $\hat{\boldsymbol{\beta}}(\tau)$.

The ALPHA= option in the PROC HPQUANTSELECT statement sets the α value for the confidence limit statistics. The degrees of freedom for $t_{\frac{\alpha}{2}}$ are $n - p$.

Table 13.6 contains the diagnostic statistics and their formulas. Each statistic is computed for each observation.

Table 13.6 Formulas and Definitions for Diagnostic Statistics

MODEL Option or Statistic	Formula
PREDj	$\widehat{y}_{ji} = \mathbf{x}_i' \hat{\boldsymbol{\beta}}(\tau_j)$
RESj	$y_i - \widehat{y}_{ji}$
STDPj	$\sqrt{\mathbf{x}_i' \hat{\Sigma}(\tau_j) \mathbf{x}_i}$
LCLMj	$\widehat{y}_{ji} - t_{\frac{\alpha}{2}} \text{STDP}j_i$
UCLMj	$\widehat{y}_{ji} + t_{\frac{\alpha}{2}} \text{STDP}j_i$

Classification Variables and the SPLIT Option

PROC HPQUANTSELECT supports the ability to split classification variables when you do model selection. You use the SPLIT option in the CLASS statement to specify that the columns of the design matrix that correspond to effects that contain a split classification variable can enter or leave a model independently of the other design columns of that effect. The following statements illustrate the use of the SPLIT option:

```
data splitExample;
   length C2 $6;
   drop i;
   do i=1 to 1000;
     C1 = 1 + mod(i,6);
     if      i < 250 then C2 = 'Low';
     else if i < 500 then C2 = 'Medium';
     else                 C2 = 'High';
     x1 = ranuni(1);
     x2 = ranuni(1);
     y = x1+3*(C2 ='low')  + 10*(C1=3) +5*(C1=5) + rannor(1);
     output;
   end;
run;

proc hpquantselect data=splitExample;
   class C1(split) C2(order=data);
   model y = C1 C2 x1 x2/orderselect clb;
   selection method=forward;
run;
```

The "Class Levels" table in Figure 13.14 is produced by default whenever you specify a CLASS statement.

Figure 13.14 Class Levels

The HPQUANTSELECT Procedure

Class Level Information

Class	Levels	Values
C1	6	* 1 2 3 4 5 6
C2	3	Low Medium High

*** Associated Parameters Split**

The SPLIT option has been specified for the classification variable C1. This permits the parameters that are associated with the effect C1 to enter or leave the model individually. The "Parameter Estimates" table in Figure 13.15 shows that for this example the parameters that correspond to only levels 3 and 5 of C1 are in the selected model. Finally, note that the ORDERSELECT option in the MODEL statement displays the parameters in the order in which they first entered the model.

Figure 13.15 Parameter Estimates

Parameter Estimates

Parameter	DF	Estimate	Standard Error	95% Confidence Limits	
Intercept	1	-0.21596	0.09024	-0.39304	-0.03887
C1_3	1	10.08952	0.09852	9.89619	10.28285
C1_5	1	5.04115	0.10835	4.82854	5.25376
x1	1	1.29863	0.14014	1.02363	1.57363

Macro Variables That Contain Selected Effects

PROC HPQUANTSELECT saves the list of selected effects in a macro variable for each selected model so that you can use other SAS procedures to perform postselection analyses. This list does not explicitly include the intercept so that you can use it in the MODEL statement of other SAS/STAT regression procedures.

When multiple quantile levels or BY processing are used, one macro variable, indexed by the quantile level order and the BY group number (as shown in Table 13.7), is created for each quantile level and BY group combination.

Table 13.7 Macro Variables Created for Subsequent Processing

Macro Variable	Description
Single Quantile Level and No BY Processing	
_HPQRSIND	Selected model
Multiple Quantile Levels and No BY Processing	
_HPQRSINDT1	Selected model for the first quantile level
_HPQRSINDT2	Selected model for the second quantile level
...	
Single Quantile Level and BY Processing	
_HPQRSIND1	Selected model for BY group 1
_HPQRSIND2	Selected model for BY group 2
...	
Multiple Quantile Levels and BY Processing	
_HPQRSIND1T1	Selected model for the first quantile level and BY group 1
_HPQRSIND1T2	Selected model for the second quantile level and BY group 1
...	
_HPQRSIND2T1	Selected model for the first quantile level and BY group 2
_HPQRSIND2T2	Selected model for the second quantile level and BY group 2
...	

The macro variables _HPQRSIND, _HPQRSINDT1, _HPQRSIND1, and _HPQRSIND1T1 are all synonyms.

The following statements generate a simulation data set, use PROC HPQUANTSELECT to select a median regression model, and print the macro variables for the selected model:

```
%let seed=321;
%let p=20;
%let n=3000;

data analysisData;
   array x{&p} x1-x&p;
   do i=1 to &n;
      do j=1 to &p;
         x{j} = ranuni(&seed);
      end;
      e  = ranuni(&seed);
      y  = x1 + x2 + x3 + e;
      output;
   end;
run;

proc hpquantselect data=analysisData;
   model y = x1-x&p;
   selection method=forward;
run;

%put _HPQRSIND    = &_hpqrsind;
%put _HPQRSIND1   = &_hpqrsind1;
%put _HPQRSIND1T1 = &_hpqrsind1t1;
```

The following statements use PROC HPREG to fit a linear regression model on the effects that are selected by the HPQUANTSELECT procedure:

```
proc hpreg data=analysisData;
   model y = &_hpqrsind;
run;
```

Using Validation and Test Data

When you have sufficient data, you can subdivide your data into three parts, called the training, validation, and test data. During the selection process, models are fit on the training data, and the prediction error for the models is found by using the validation data. This prediction error on the validation data can be used to decide when to terminate the selection process or what effects to include as the selection process proceeds. Finally, after you have obtained a selected model, you can use the test set to assess how the selected model generalizes on data that played no role in selecting the model.

In some cases, you might want to use only training and test data. For example, you might decide to use an information criterion to decide what effects to include and when to terminate the selection process. In this case, no validation data are required, but test data can still be useful in assessing the predictive performance of the selected model. In other cases, you might decide to use validation data during the selection process but forgo assessing the selected model on test data. Hastie, Tibshirani, and Friedman (2001) state that it is

difficult to give a general rule for how many observations you should assign to each role. They note that a typical split might be 50% for training and 25% each for validation and testing.

You use a PARTITION statement to logically subdivide the DATA= data set into separate roles. You can name the fractions of the data that you want to reserve as test data and validation data. For example, the following statements randomly subdivide the inData data set, reserving 50% for training and 25% each for validation and testing:

```
proc hpquantselect data=inData;
  partition fraction(test=0.25 validate=0.25);
  ...
run;
```

In some cases, you might need to exercise more control over the partitioning of the input data set. You can do this by naming both a variable in the input data set and a formatted value of that variable that correspond to each role. For example, the following statements assign roles to the observations in the inData data set based on the value of the variable Group in that data set. Observations in which the value of Group is `'group 1'` are assigned for testing, and those whose value is `'group 2'` are assigned to training. All other observations are ignored.

```
proc hpquantselect data=inData;
  partition roleVar=Group(test='group 1' train='group 2')
  ...
run;
```

After you reserve observations for training, validation, and testing, a model fit on the training data is scored on the validation and test data, and the average check loss (ACL) is computed separately for each of these subsets. The ACL for each data role is the error sum of squares for observations in that role divided by the number of observations in that role.

Using the Validation ACL as the STOP= Criterion

If you have provided observations for validation, then you can specify STOP=VALIDATE as a suboption of the METHOD= option in the SELECTION statement. At step k of the selection process, the best candidate effect to enter or leave the current model is determined. Here "best candidate" means the effect that gives the best value of the SELECT= criterion; this criterion need not be based on the validation data. The validation ACL is computed for the model in which this candidate effect is added or removed. If this validation ACL is greater than the validation ACL for the model at step k, then the selection process terminates at step k.

Using the Validation ACL as the CHOOSE= Criterion

When you specify the CHOOSE=VALIDATE suboption of the METHOD= option in the SELECTION statement, the validation ACL is computed for the models at each step of the selection process. The smallest model at any step that yields the smallest validation ACL is selected.

Using the Validation ACL as the SELECT= Criterion

You request the validation ACL as the selection criterion by specifying the SELECT=VALIDATE suboption of the METHOD= option in the SELECTION statement. At step k of the selection process, the validation ACL is computed for each model in which a candidate for entry is added or a candidate for removal is dropped. The selected candidate for entry or removal is the one that yields a model that has the minimal validation ACL.

Computational Method

Multithreading

Threading refers to the organization of computational work into multiple tasks (processing units that can be scheduled by the operating system). A task is associated with a thread. Multithreading refers to the concurrent execution of threads. When multithreading is possible, substantial performance gains can be realized compared to those that occur in sequential (single-threaded) execution.

The number of threads that the HPQUANTSELECT procedure spawns is determined by the number of CPUs on a machine and can be controlled in the following ways:

- You can specify the CPU count by using the CPUCOUNT= SAS system option. For example, if you specify the following statement, the HPQUANTSELECT procedure schedules threads as if it were executing on a system that had at most four CPUs:

```
options cpucount=4;
```

- You can specify the NTHREADS= option in the PERFORMANCE statement to determine the number of threads. This specification overrides the system option. Specify NTHREADS=1 to force single-threaded execution.

The number of threads is displayed in the "Performance Information" table, which is part of the default output. The HPQUANTSELECT procedure allocates one thread per CPU.

PROC HPQUANTSELECT divides the data processing on a single machine among the threads—that is, the HPQUANTSELECT procedure implements multithreading through a data-parallel model. For example, if the input data set has 1,000 observations and you are running on four threads, then 250 observations are associated with each thread. All operations that require access to the data are then multithreaded. These operations include the following:

- variable levelization
- effect levelization
- formation of the crossproducts matrix
- quantile regression model fitting
- estimation of covariance matrix for parameter estimates
- evaluation of predicted residual sums of check losses on validation and test data
- scoring of observations

In addition, operations on matrices such as sweeps might be multithreaded if the matrices are of sufficient size to realize performance benefits from managing multiple threads for the particular matrix operation.

Output Data Set

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

- variables that are explicitly created by the statement
- variables that are listed in the ID statement or specified by using the COPYVAR= option
- distribution keys or hash keys that are transferred from the input data set

The high-performance analytical procedures enable you to add output data set information that is necessary for subsequent SQL joins without copying the entire input data set to the output data set. For further details about output data sets when PROC HPQUANTSELECT is run in distributed mode, see the section "Output Data Sets" on page 34 in Chapter 3, "Shared Concepts and Topics."

Displayed Output

The following sections describe the output that PROC HPQUANTSELECT produces. The output is organized into various tables, which are discussed in their order of appearance.

Performance Information

The "Performance Information" table is produced by default and displays information about the grid host for distributed execution and about whether PROC HPQUANTSELECT executes in single-machine mode, distributed mode, or alongside-the-database mode. The numbers of compute nodes and threads are also displayed, depending on the environment.

Data Access Information

The "Data Access Information" table is produced by default. For the input and output data sets, it displays the libref and data set name, the engine used to access the data, the role (input or output) of the data set, and path that data followed to the reach the computation.

Model Information

The "Model Information" table displays basic information about the model, such as the response variable, the weight variable, and the type of parameterization that is used for classification variables named in the CLASS statement.

Selection Information

When you specify the SELECTION statement, by default the HPQUANTSELECT procedure produces a series of tables that contain information about the model selection. The "Selection Information" table informs you about the model selection method; select, stop, and choose criteria; and other parameters that govern the selection. You can suppress this table by specifying DETAILS=NONE in the SELECTION statement.

Number of Observations

The "Number of Observations" table displays the number of observations that are read from the input data set and the number of observations that are used in the analysis. If you use a PARTITION statement, the table also displays the number of observations that are used for each data role.

Class Level Information

The "Class Level Information" table lists the levels of every variable that is specified in the CLASS statement. You should check this information to ensure that the data are correct. You can adjust the order of the CLASS variable levels by specifying the ORDER= option in the CLASS statement. You can suppress the "Class Level Information" table completely or partially by specifying the NOCLPRINT= option in the PROC HPQUANTSELECT statement.

If the classification variables are in the reference parameterization, the "Class Level Information" table also displays the reference value for each variable. This table also indicates which, if any, of the classification variables are split by using the SPLIT option in the CLASS statement.

Dimensions

The "Dimensions" table displays information about the number of effects and the number of parameters from which the selected model is chosen. If you use split classification variables, then this table also includes the number of effects after splitting is taken into account.

Entry and Removal Candidates

When you specify the DETAILS=ALL or DETAILS=STEPS option in the SELECTION statement, the HPQUANTSELECT procedure produces "Entry Candidates" and "Removal Candidates" tables that display the effect names and values of the criterion used to select entering or departing effects at each step of the selection process. The effects are displayed in order from best to worst of the selection criterion.

Selection Summary

When you specify the SELECTION statement, the HPQUANTSELECT procedure produces the "Selection Summary" table, which contains information about the sequence of steps of the selection process. For each step, the effect that was entered or dropped is displayed along with the statistics used to select the effect, stop the selection, and choose the selected model. For all criteria that you can use for model selection, the steps at which the optimal values of these criteria occur are also indicated.

You can suppress the "Selection Summary" table by specifying DETAILS=NONE in the SELECTION statement.

Stop Reason

The "Stop Reason" table displays the reason why the selection stopped. To facilitate programmatic use of this table, an integer code is assigned to each reason and is included if you output this table by using an ODS OUTPUT statement. The reasons and their associated codes follow:

Code	Stop Reason
1	All eligible effects are in the model.
2	All eligible effects have been removed.
3	Specified maximum number of steps have been done.
4	The model contains the specified maximum number of effects.
5	The model contains the specified minimum number of effects (for backward selection).
6	The stopping criterion is at a local optimum.
7	No suitable add or drop candidate could be found.
8	Adding or dropping any effect does not improve the selection criterion.
9	No candidate meets the appropriate SLE or SLS significance level.
10	Stepwise selection is cycling.
11	The model is an exact fit.
12	Dropping an effect would result in an empty model.

You can suppress the "Stop Reason" table by specifying DETAILS=NONE in the SELECTION statement.

Selection Reason

When you specify the SELECTION statement, the HPQUANTSELECT procedure produces a simple table that contains text that explains why the final model was selected.

You can suppress the "Selection Reason" table by specifying DETAILS=NONE in the SELECTION statement.

Selected Effects

When you specify the SELECTION statement, the HPQUANTSELECT procedure produces a simple table that contains text that lists the selected effects in the final model.

Fit Statistics

The "Fit Statistics" table displays fit statistics for the selected model. The statistics include the following:

- Objective Function, total sum of check losses. Objective Function is denoted as $D(\tau)$ in Table 13.5.
- R1, a measure between 0 and 1 that indicates the portion of the (corrected) total check losses attributed to the fit rather than left to residuals. It is calculated as $1 - \frac{D(\tau)}{D_0(\tau)}$. It is the quantile regression counterpart of the linear regression R square.
- Adj R1, the adjusted R1, a version of R1 that has been adjusted for degrees of freedom. It is calculated as

$$\text{Adj R1} = 1 - \frac{n-i}{n-p}(1 - \text{R1})$$

where $i = 1$ if the intercept is forced in and $i = 0$ otherwise , n is the number of observations used to fit the model, and p is the number of parameters in the model.

- the fit criteria AIC, AICC, and SBC if they are used in the selection process. For the formulas to evaluate these criteria, see Table 13.5.
- the average check loss (ACL) on the training, validation, and test data

You can request the "Fit Statistics" tables for the model at each step of the selection process by specifying the DETAILS= option in the SELECTION statement.

Parameter Estimates

The "Parameter Estimates" table displays the parameters in the selected model and their estimates. The information that is displayed for each parameter in the selected model includes the following:

- the parameter label, which includes the effect name and level information for effects that contain classification variables
- the degrees of freedom (DF) for the parameter. There is one degree of freedom unless the model is not full rank.
- the parameter estimate
- the standardized parameter estimate. PROC HPQUANTSELECT outputs the standardized parameter estimate only if you specify the STB option in the MODEL statement.

If you specify the CLB option in the MODEL statement, PROC HPQUANTSELECT also outputs the following information:

- the standard error, which is the estimate of the standard deviation of the parameter estimate
- the $100(1-\alpha)\%$ confidence limits for the parameter estimate

You can request "Parameter Estimates" tables for the model at each step of the selection process by specifying the DETAILS= option in the SELECTION statement.

Timing Information

If you specify the DETAILS option in the PERFORMANCE statement, PROC HPQUANTSELECT also produces a "Timing" table that displays the elapsed time for each main task of PROC HPQUANTSELECT.

ODS Table Names

Each table that the HPQUANTSELECT procedure creates has a name associated with it. You must use this name to refer to the table when you use ODS statements. These names are listed in Table 13.8.

Table 13.8 ODS Tables Produced by PROC HPQUANTSELECT

Table Name	Description	Required Statement / Option
ClassLevels	Level information from the CLASS statement	CLASS
DataAccessInfo	Information about modes of data access	Default output
Dimensions	Model dimensions	Default output
EntryCandidates	Candidates for entry at step	SELECTION DETAILS=ALL | STEPS
FitStatistics	Fit statistics	Default output
ModelInfo	Information about the modeling environment	Default output
NObs	Number of observations read and used	Default output
ParameterEstimates	Solutions for the parameter estimates associated with effects in the MODEL statement	Default output
PerformanceInfo	Information about high-performance computing environment	Default output
RemovalCandidates	Candidates for removal at step	SELECTION DETAILS=ALL | STEPS
SelectedEffects	List of selected effects	SELECTION
SelectionInfo	Information about selection settings	Default output
SelectionReason	Reason for selecting the final model	SELECTION
SelectionSummary	Summary information about the model selection steps	SELECTION
StopReason	Reason selection was terminated	SELECTION
Timing	Timing breakdown by task	PERFORMANCE DETAILS

Examples: HPQUANTSELECT Procedure

Example 13.1: Simulation Study

This example is based on "Example 84.1: Simulation Study" (Chapter 84, *SAS/STAT User's Guide*). This simulation study shows how you can use the forward selection method to select quantile regression models for single quantile levels. The following statements simulate a data set from a naive instrumental model (Chernozhukov and Hansen 2008):

```
%let seed=321;
%let p=20;
%let n=3000;

data analysisData;
   array x{&p} x1-x&p;
   do i=1 to &n;
      U  = ranuni(&seed);
      x1 = ranuni(&seed);
      x2 = ranexp(&seed);
      x3 = abs(rannor(&seed));
      y  = x1*(U-0.1) + x2*(U*U-0.25) + x3*(exp(U)-exp(0.9));
      do j=4 to &p;
         x{j} = ranuni(&seed);
      end;
      output;
   end;
run;
```

Variable U in the data set indicates the true quantile level of the response y conditional on $\mathbf{x} = (x_1, \ldots, x_p)$.

Let $Q_Y(\tau|\mathbf{x}) = \mathbf{x}\boldsymbol{\beta}(\tau)$ denote the underlying quantile regression model, where $\boldsymbol{\beta}(\tau) = (\beta_1(\tau), \ldots, \beta_p(\tau))'$. Then, the true parameter functions are

$$\begin{aligned}
\beta_1(\tau) &= \tau - 0.1 \\
\beta_2(\tau) &= \tau^2 - 0.25 \\
\beta_3(\tau) &= \exp(\tau) - \exp(0.9) \\
\beta_4(\tau) &= \cdots = \beta_p(\tau) = 0
\end{aligned}$$

It is easy to see that, at $\tau = 0.1$, only $\beta_2(0.1) = -0.24$ and $\beta_3(0.1) = \exp(0.1) - \exp(0.9) \approx -1.354432$ are nonzero parameters. Therefore, an effective effect-selection method should select x2 and x3 and drop all the other effects in this data set at $\tau = 0.1$. By the same rationale, x1 and x3 should be selected at $\tau = 0.5$ with $\beta_1(0.5) = 0.4$ and $\beta_3(0.5) \approx -0.810882$, and x1 and x2 should be selected at $\tau = 0.9$ with $\beta_1(0.9) = 0.8$ and $\beta_2(0.9) = 0.56$.

The following statements use PROC HPQUANTSELECT with the forward selection method. The STB option and the CLB option in the MODEL statement request the standardized parameter estimates and the confidence limits of parameter estimates, respectively.

```
proc hpquantselect data=analysisData;
   model y= x1-x&p / quantile=0.1 0.5 0.9 stb clb;
   selection method=forward;
   output out=out p=pred;
run;
```

Output 13.1.1 shows that, by default, the CHOOSE= and STOP= options are both set to SBC.

Output 13.1.1 Model Information

The HPQUANTSELECT Procedure

Selection Information	
Selection Method	Forward
Select Criterion	SBC
Stop Criterion	SBC
Effect Hierarchy Enforced	None
Stop Horizon	3

Output 13.1.2, Output 13.1.3, and Output 13.1.4 display the selected effects and the parameter estimates for $\tau = 0.1$, $\tau = 0.5$, and $\tau = 0.9$, respectively. You can see that the forward selection method correctly selects active effects for all three quantile levels.

Output 13.1.2 Parameter Estimates at $\tau = 0.1$

The HPQUANTSELECT Procedure

Quantile Level = 0.1
Selected Model

Selected Effects: Intercept x2 x3

Parameter Estimates

Parameter	DF	Estimate	Standardized Estimate	Standard Error	95% Confidence Limits	
Intercept	1	0.01179	0	0.01192	-0.01158	0.03516
x2	1	-0.22871	-0.21829	0.00946	-0.24725	-0.21017
x3	1	-1.37991	-0.78452	0.01556	-1.41042	-1.34939

Output 13.1.3 Parameter Estimates at $\tau = 0.5$

The HPQUANTSELECT Procedure

Quantile Level = 0.5
Selected Model

Selected Effects: Intercept x1 x3

Parameter Estimates

Parameter	DF	Estimate	Standardized Estimate	Standard Error	95% Confidence Limits	
Intercept	1	0.01178	0	0.03418	-0.05524	0.07879
x1	1	0.42584	0.11879	0.06237	0.30355	0.54814
x3	1	-0.86332	-0.49082	0.04765	-0.95674	-0.76989

Output 13.1.4 Parameter Estimates at $\tau = 0.9$

The HPQUANTSELECT Procedure

Quantile Level = 0.9
Selected Model

Selected Effects: Intercept x1 x2

Parameter Estimates

Parameter	DF	Estimate	Standardized Estimate	Standard Error	95% Confidence Limits	
Intercept	1	-0.00774	0	0.03292	-0.07228	0.05680
x1	1	0.78294	0.21841	0.05134	0.68228	0.88360
x2	1	0.57644	0.55018	0.03422	0.50935	0.64354

Example 13.2: Growth Charts for Body Mass Index

This example is modeled on the example in the section "Getting Started: QUANTSELECT Procedure" (Chapter 84, *SAS/STAT User's Guide*). It highlights the use of the HPQUANTSELECT procedure for multiple-level quantile regression by creating growth charts for men's body mass index (BMI).

BMI, which is defined as the ratio of weight (kg) to squared height (m^2), is a standard measure for categorizing individuals as overweight or underweight. The percentiles of BMI for specified ages are of particular interest. This example draws smooth BMI quantile curves conditional on Age, which can serve as BMI growth charts in medical diagnosis to identify BMI percentiles for subjects.

The BMIMen data set is from the 1999–2000 and 2001–2002 survey results for men that are published by the National Center for Health Statistics. It contains the two variables BMI and Age with 3,264 observations.

```
data bmimen;
   input BMI Age @@;
   SqrtAge = sqrt(Age);
   InveAge = 1/Age;
   LogBMI  = log(BMI);
   datalines;
18.6  2.0 17.1  2.0 19.0  2.0 16.8  2.0 19.0  2.1  15.5   2.1
16.7  2.1 16.1  2.1 18.0  2.1 17.8  2.1 18.3  2.1  16.9   2.1
15.9  2.1 20.6  2.1 16.7  2.1 15.4  2.1 15.9  2.1  17.7   2.1

   ... more lines ...

29.0 80.0 24.1 80.0 26.6 80.0 24.2 80.0 22.7 80.0  28.4  80.0
26.3 80.0 25.6 80.0 24.8 80.0 28.6 80.0 25.7 80.0  25.8  80.0
22.5 80.0 25.1 80.0 27.0 80.0 27.9 80.0 28.5 80.0  21.7  80.0
33.5 80.0 26.1 80.0 28.4 80.0 22.7 80.0 28.0 80.0  42.7  80.0
;
```

The logarithm of BMI is used as the response. (Although this approach does not improve the quantile regression fit, it helps with statistical inference.) The following statements fit quantile regression models for the BMIMen data set at 10 quantile levels:

```
%let quantile=0.03 0.05 0.1 0.25 0.5 0.75 0.85 0.90 0.95 0.97;
%let nq=10;

proc hpquantselect data=BMIMen;
   model logBMI = InveAge SqrtAge Age SqrtAge*Age Age*Age Age*Age*Age
         / quantile=&quantile;
   code file='bmicode.sas';
   output out=Bmiout copyvars=(BMI Age) pred=P_LogBMI;
run;
```

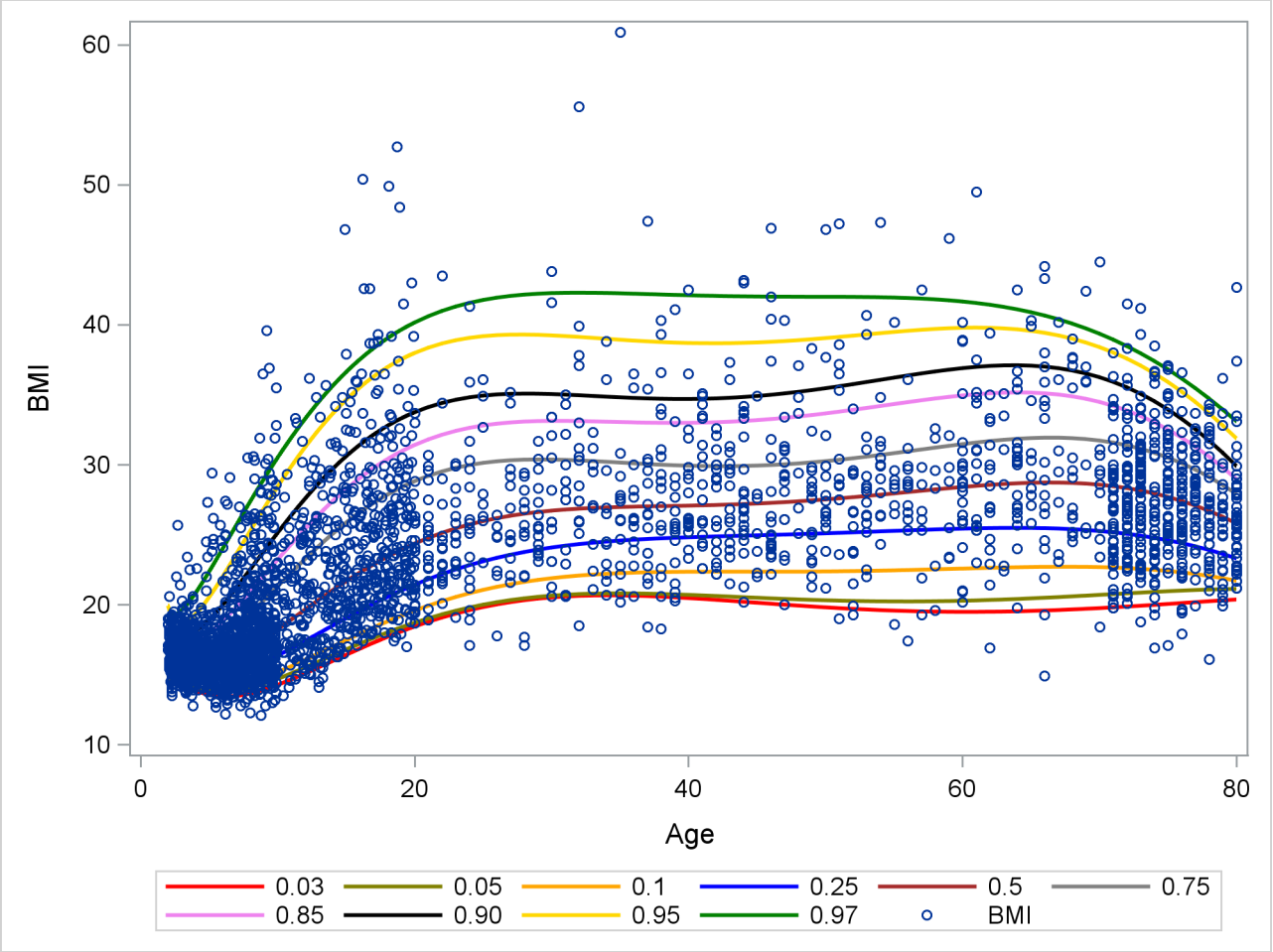
The CODE statement enables you to write a SAS DATA step to compute quantile predictions of the fitted model. The OUTPUT statement outputs the mean predicted quantiles for the 10 specified quantile levels. The PRED= option in the OUTPUT statement specifies the variable names for the quantile predictions. For examples, p1 is for quantile level 0.03, and p2 is for quantile level 0.05.

The following statements define and apply a SAS macro function to create a quantile curves plot for the BMIMen data set:

```
%let BMIcolor=red olive orange blue brown gray violet black gold green;

%macro plotBMI;
   data BmiPred;
      set Bmiout;
      %do j=1 %to &nq;
         predBMI&j = exp(P_LogBMI&j);
      %end;
      label %do j=1 %to &nq;
               predBMI&j=%qscan(&quantile,&j,%str( ))
            %end;;
   run;

   proc sort data=BmiPred;
      by Age;
   run;

   proc sgplot data=BmiPred;
      %do j=1 %to &nq;
         series y=predBMI&j x=Age/lineattrs=(thickness=2
         color=%qscan(&BMIcolor,&j,%str( )));
      %end;
      scatter y=BMI x=Age/markerattrs=(size=5);
   run;
%mend;

%plotBMI;
```

Figure 13.2.1 shows the BMI quantile curves, which can serve as BMI growth charts. For example, the percentiles of any observations (small blue circles) that are located between the top 0.95 quantile (gold) curve and the 0.97 quantile (green) curve are between the 95th percentile and the 97th percentile. By using this rule, you can measure the percentile range for any observations of interest.

Output 13.2.1 Growth Chart for Body Mass Index



Other than using the OUTPUT statement, you can also calculate quantile predictions by using the CODE statement. The following statements show how to use the SAS DATA step and the SAS file *bmicode*, which the CODE statement requests, to calculate quantile predictions for the BMIMen data set:

```
data Newmen;
   set BMIMen;
   %inc bmicode;
run;
```

The SET statement in the SAS DATA step specifies a data set for computing quantile predictions. This is usually a new data set that you want to score. This example uses the BMIMen data set again, so the quantile predictions in the Newmen data set are identical to those in the Bmiout data set. The following statements compare the Bmiout data set with the Newmen data set:

```
proc compare data=Bmiout compare=Newmen criterion=0.00001;
run;
```

References

Akaike, H. (1969), "Fitting Autoregressive Models for Prediction," *Annals of the Institute of Statistical Mathematics*, 21, 243–247.

Chernozhukov, V. and Hansen, C. (2008), "Instrumental Variable Quantile Regression: A Robust Inference Approach," *Journal of Econometrics*, 142, 379–398.

Collier Books (1987), *The 1987 Baseball Encyclopedia Update*, New York: Macmillan.

Gertz, E. M. and Griffin, J. D. (2005), *Support Vector Machine Classifiers for Large Data Sets*, Technical Report ANL/MCS-TM-289, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL.

Gertz, E. M. and Griffin, J. D. (2010), "Using an Iterative Linear Solver in an Interior-Point Method for Generating Support Vector Machines," *Computational Optimization and Applications*, 47, 431–453.

Hastie, T. J., Tibshirani, R. J., and Friedman, J. H. (2001), *The Elements of Statistical Learning*, New York: Springer-Verlag.

Hurvich, C. M. and Tsai, C.-L. (1989), "Regression and Time Series Model Selection in Small Samples," *Biometrika*, 76, 297–307.

Koenker, R. (2005), *Quantile Regression*, New York: Cambridge University Press.

Koenker, R. and Bassett, G. W. (1978), "Regression Quantiles," *Econometrica*, 46, 33–50.

Koenker, R. and Bassett, G. W. (1982), "Tests of Linear Hypotheses and l_1 Estimation," *Econometrica*, 50, 1577–1583.

Koenker, R. and Machado, A. F. (1999), "Goodness of Fit and Related Inference Processes for Quantile Regression," *Journal of the American Statistical Association*, 94, 1296–1310.

Reichler, J. L., ed. (1987), *The 1987 Baseball Encyclopedia Update*, New York: Macmillan.

Schwarz, G. (1978), "Estimating the Dimension of a Model," *Annals of Statistics*, 6, 461–464.

Time Inc. (1987), "What They Make," *Sports Illustrated*, April, 54–81.

Yuan, M. (2006), "GACV for Quantile Smoothing Splines," *Computational Statistics and Data Analysis*, 50, 813–829.
URL http://EconPapers.repec.org/RePEc:eee:csdana:v:50:y:2006:i:3:p:813-829

Chapter 14
The HPREG Procedure

Contents

Overview: HPREG Procedure **540**
PROC HPREG Features 540
PROC HPREG Contrasted with Other SAS Procedures 541
Getting Started: HPREG Procedure **542**
Syntax: HPREG Procedure **548**
PROC HPREG Statement 548
BY Statement 549
CLASS Statement 550
CODE Statement 550
FREQ Statement 551
ID Statement 551
MODEL Statement 551
OUTPUT Statement 553
PARTITION Statement 555
PERFORMANCE Statement 555
SELECTION Statement 556
WEIGHT Statement 558
Details: HPREG Procedure **558**
Criteria Used in Model Selection 558
Diagnostic Statistics 560
Classification Variables and the SPLIT Option 561
Using Validation and Test Data 562
Computational Method 564
Output Data Set 565
Screening 565
Displayed Output 566
ODS Table Names 570
Examples: HPREG Procedure **571**
Example 14.1: Model Selection with Validation 571
Example 14.2: Backward Selection in Single-Machine and Distributed Modes 578
Example 14.3: Forward-Swap Selection 582
Example 14.4: Forward Selection with Screening 585
References **591**

Overview: HPREG Procedure

The HPREG procedure is a high-performance procedure that fits and performs model selection for ordinary linear least squares models. The models supported are standard independently and identically distributed general linear models, which can contain main effects that consist of both continuous and classification variables and interaction effects of these variables. The procedure offers extensive capabilities for customizing the model selection with a wide variety of selection and stopping criteria, from traditional and computationally efficient significance-level-based criteria to more computationally intensive validation-based criteria. PROC HPREG also provides a variety of regression diagnostics that are conditional on the selected model.

PROC HPREG runs in either single-machine mode or distributed mode.

NOTE: Distributed mode requires SAS High-Performance Statistics.

PROC HPREG Features

The main features of the HPREG procedure are as follows:

- **Model specification**
  - supports GLM and reference parameterization for classification effects
  - supports any degree of interaction (crossed effects) and nested effects
  - supports hierarchy among effects
  - supports partitioning of data into training, validation, and testing roles
  - supports a FREQ statement for grouped analysis
  - supports a WEIGHT statement for weighted analysis
- **Selection control**
  - provides multiple effect-selection methods
  - enables selection from a very large number of effects (tens of thousands)
  - offers selection of individual levels of classification effects
  - provides effect selection based on a variety of selection criteria
  - provides stopping rules based on a variety of model evaluation criteria
  - supports stopping and selection rules based on external validation and leave-one-out cross validation
- **Display and output**
  - produces output data sets that contain predicted values, residuals, studentized residuals, confidence limits, and influence statistics

The HPREG procedure supports the following effect selection methods. For a more detailed description of these methods, see the section "Methods" on page 63 in Chapter 4, "Shared Statistical Concepts."

- Forward selection starts with no effects in the model and adds effects.
- Backward elimination starts with all effects in the model and deletes effects.
- Stepwise regression is similar to forward selection except that effects already in the model do not necessarily stay there.
- Forward-swap selection is a modification of forward selection. Before any addition step, PROC HPREG makes all pairwise swaps of effects in and out of the current model that improve the selection criterion. When the selection criterion is R square, this method coincides with the MAXR method in the REG procedure in SAS/STAT software.
- Least angle regression, like forward selection, starts with no effects in the model and adds effects. The parameter estimates at any step are "shrunk" when compared to the corresponding least squares estimates.
- Lasso adds and deletes parameters based on a version of ordinary least squares in which the sum of the absolute regression coefficients is constrained. PROC HPREG also supports adaptive lasso selection where weights are applied to each of the parameters in forming the lasso constraint.

Hybrid versions of LAR and LASSO methods are also supported. They use LAR or LASSO to select the model, but then estimate the regression coefficients by ordinary weighted least squares.

Because the HPREG procedure is a high-performance analytical procedure, it also does the following:

- enables you to run in distributed mode on a cluster of machines that distribute the data and the computations
- enables you to run in single-machine mode on the server where SAS is installed
- exploits all the available cores and concurrent threads, regardless of execution mode

For more information, see the section "Processing Modes" on page 10 in Chapter 3, "Shared Concepts and Topics."

PROC HPREG Contrasted with Other SAS Procedures

For general contrasts between SAS high-performance statistical procedures and other SAS procedures, see the section "Common Features of SAS High-Performance Statistical Procedures" on page 42 in Chapter 4, "Shared Statistical Concepts." The following remarks contrast the HPREG procedure with the GLMSELECT, GLM, and REG procedures in SAS/STAT software.

A major functional difference between the HPREG and REG procedures is that the HPREG procedure enables you to specify general linear models that include classification variables. In this respect it is similar to the GLM and GLMSELECT procedures. In terms of the supported model selection methods, the HPREG procedure most resembles the GLMSELECT procedure. Like the GLMSELECT procedure but different from the REG procedure, the HPREG procedure supports the LAR and LASSO methods, the ability to use external validation data and cross validation as selection criteria, and extensive options to customize the selection process. The HPREG procedure does not support the MAXR and MINR methods that are available

in the REG procedure. Nor does the HPREG procedure include any support for the all-subset-based methods that you can find in the REG procedure.

The CLASS statement in the HPREG procedure permits two parameterizations: the GLM-type parameterization and a reference parameterization. In contrast to the GLMSELECT, GENMOD, LOGISTIC, and other procedures that permit multiple parameterizations, the HPREG procedure does not mix parameterizations across the variables in the CLASS statement. In other words, all classification variables are in the same parameterization, and this parameterization is either the GLM or reference parameterization.

Like the REG procedure but different from the GLMSELECT procedure, the HPREG procedure does not perform model selection by default. If you request model selection by using the SELECTION statement then the default selection method is stepwise selection based on the SBC criterion. This default matches the default method used in PROC GLMSELECT.

As with the REG procedure but not supported with the GLMSELECT procedure, you can request observation-wise residual and influence diagnostics in the OUTPUT statement and variance inflation and tolerance statistics for the parameter estimates. If the fitted model has been obtained by performing model selection, then these statistics are conditional on the selected model and do not take the variability introduced by the selection process into account.

Getting Started: HPREG Procedure

The following example is closely modeled on the example in the section "Getting Started: GLMSELECT Procedure" in the *SAS/STAT User's Guide*.

The Sashelp.Baseball data set contains salary and performance information for Major League Baseball players who played at least one game in both the 1986 and 1987 seasons, excluding pitchers. The salaries (*Sports Illustrated,* April 20, 1987) are for the 1987 season and the performance measures are from 1986 (Collier Books, *The 1987 Baseball Encyclopedia Update*). The following step displays in Figure 14.1 the variables in the data set:

```
proc contents varnum data=sashelp.baseball;
   ods select position;
run;
```

Figure 14.1 Sashelp.Baseball Data Set

The CONTENTS Procedure

Variables in Creation Order				
#	**Variable**	**Type**	**Len**	**Label**
1	Name	Char	18	Player's Name
2	Team	Char	14	Team at the End of 1986
3	nAtBat	Num	8	Times at Bat in 1986
4	nHits	Num	8	Hits in 1986
5	nHome	Num	8	Home Runs in 1986
6	nRuns	Num	8	Runs in 1986
7	nRBI	Num	8	RBIs in 1986
8	nBB	Num	8	Walks in 1986
9	YrMajor	Num	8	Years in the Major Leagues
10	CrAtBat	Num	8	Career Times at Bat
11	CrHits	Num	8	Career Hits
12	CrHome	Num	8	Career Home Runs
13	CrRuns	Num	8	Career Runs
14	CrRbi	Num	8	Career RBIs
15	CrBB	Num	8	Career Walks
16	League	Char	8	League at the End of 1986
17	Division	Char	8	Division at the End of 1986
18	Position	Char	8	Position(s) in 1986
19	nOuts	Num	8	Put Outs in 1986
20	nAssts	Num	8	Assists in 1986
21	nError	Num	8	Errors in 1986
22	Salary	Num	8	1987 Salary in $ Thousands
23	Div	Char	16	League and Division
24	logSalary	Num	8	Log Salary

Suppose you want to investigate whether you can model the players' salaries for the 1987 season based on performance measures for the previous season. The aim is to obtain a parsimonious model that does not overfit this particular data, making it useful for prediction. This example shows how you can use PROC HPREG as a starting point for such an analysis. Since the variation of salaries is much greater for the higher salaries, it is appropriate to apply a log transformation to the salaries before doing the model selection.

The following statements select a model with the default settings for stepwise selection:

```
proc hpreg data=sashelp.baseball;
  class league division;
  model logSalary = nAtBat nHits nHome nRuns nRBI nBB
                    yrMajor crAtBat crHits crHome crRuns crRbi
                    crBB league division nOuts nAssts nError;
  selection method=stepwise;
run;
```

The default output from this analysis is presented in Figure 14.2 through Figure 14.6.

Figure 14.2 Performance, Data Access, Model, and Selection Information

The HPREG Procedure

Performance Information	
Execution Mode	Single-Machine
Number of Threads	4

Data Access Information			
Data	**Engine**	**Role**	**Path**
SASHELP.BASEBALL	V9	Input	On Client

Model Information	
Data Source	SASHELP.BASEBALL
Dependent Variable	logSalary
Class Parameterization	GLM

Selection Information	
Selection Method	Stepwise
Select Criterion	SBC
Stop Criterion	SBC
Effect Hierarchy Enforced	None
Stop Horizon	3

Figure 14.2 displays the "Performance Information," "Data Access Information," "Model Information," and "Selection Information" tables. The "Performance Information" table shows that procedure executes in single-machine mode—that is, the model is fit on the machine where the SAS session executes. This run of the HPREG procedure was performed on a multicore machine with four CPUs; one computational thread was spawned per CPU.

The "Data Access Information" table shows that the input data set is accessed with the V9 (base) engine on the client machine.

The "Model Information" table identifies the data source and response and shows that the CLASS variables are parameterized in the GLM parameterization, which is the default.

The "Selection Information" provides details about the method and criteria used to perform the model selection. The requested selection method is a variant of the traditional stepwise selection where the decisions about what effects to add or drop at any step and when to terminate the selection are both based on the Schwarz Bayesian information criterion (SBC). The effect in the current model whose removal yields the maximal decrease in the SBC statistic is dropped provided this lowers the SBC value. When no further decrease in the SBC value can be obtained by dropping an effect in the model, the effect whose addition to the model yields the lowest SBC statistic is added and the whole process is repeated. The method terminates when dropping or adding any effect increases the SBC statistic.

Figure 14.3 displays the "Number of Observations," "Class Levels," and "Dimensions" tables. The "Number of Observations" table shows that of the 322 observations in the input data, only 263 observations are used in the analysis because there are observations with incomplete data. The "Class Level Information" table lists the levels of the classification variables "division" and "league." When you specify effects that contain classification variables, the number of parameters is usually larger than the number of effects. The "Dimensions" table shows the number of effects and the number of parameters considered.

Figure 14.3 Number of Observations, Class Levels, and Dimensions

Number of Observations Read	322
Number of Observations Used	263

Class Level Information

Class	Levels	Values
League	2	American National
Division	2	East West

Dimensions	
Number of Effects	19
Number of Parameters	21

The "Stepwise Selection Summary" table in Figure 14.4 shows the effect that was added or dropped at each step of the selection process together with fit statistics for the model at each step. In this case, both selection and stopping are based on the SBC statistic.

Figure 14.4 Selection Summary Table

The HPREG Procedure

Selection Summary

Step	Effect Entered	Effect Removed	Number Effects In	SBC
0	**Intercept**		1	-57.2041
1	**CrRuns**		2	-194.3166
2	**nHits**		3	-252.5794
3	**YrMajor**		4	-262.7322
4		**CrRuns**	3	-262.8353
5	**nBB**		4	-269.7804*

*** Optimal Value of Criterion**

Figure 14.5 displays the "Stop Reason," "Selection Reason," and "Selected Effects" tables. Note that these tables are displayed without any titles. The "Stop Reason" table indicates that selection stopped because adding or removing any effect would worsen the SBC value that is used as the selection criterion. In this case, because no CHOOSE= criterion is specified in the SELECTION statement, the final model is the selected model; this is indicated in the "Selection Reason" table. The "Selected Effects" table lists the effects in the selected model.

Figure 14.5 Stopping and Selection Reasons

Stepwise selection stopped because adding or removing an effect does not improve the SBC criterion.

The model at step 5 is selected.

Selected Effects:	Intercept nHits nBB YrMajor

The "Analysis of Variance," "Fit Statistics," and "Parameter Estimates" tables shown in Figure 14.6 give details of the selected model.

Figure 14.6 Details of the Selected Model

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	3	120.52553	40.17518	120.12	<.0001
Error	259	86.62820	0.33447		
Corrected Total	262	207.15373			

Root MSE	0.57834
R-Square	0.58182
Adj R-Sq	0.57697
AIC	-19.06903
AICC	-18.83557
SBC	-269.78041
ASE	0.32938

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	4.013911	0.111290	36.07	<.0001
nHits	1	0.007929	0.000994	7.98	<.0001
nBB	1	0.007280	0.002049	3.55	0.0005
YrMajor	1	0.100663	0.007551	13.33	<.0001

You might want to examine regression diagnostics for the selected model to investigate whether collinearity among the selected parameters or the presence of outlying or high leverage observations might be impacting the fit produced. The following statements include some options and statements to obtain these diagnostics:

```
proc hpreg data=sashelp.baseball;
  id name;
  class league division;
  model logSalary = nAtBat nHits nHome nRuns nRBI nBB
                    yrMajor crAtBat crHits crHome crRuns crRbi
                    crBB league division nOuts nAssts nError / vif clb;
  selection method=stepwise;
  output out=baseballOut p=predictedLogSalary r h cookd rstudent;
run;
```

The VIF and CLB options in the MODEL statement request variance inflation factors and 95% confidence limits for the parameter estimates. Figure 14.7 shows the "Parameter Estimates" with these requested statistics. The variance inflation factors (VIF) measure the inflation in the variances of the parameter estimates due to collinearities that exist among the regressor (independent) variables. Although there are no formal criteria for deciding whether a VIF is large enough to affect the predicted values, the VIF values for the selected effects in this example are small enough to indicate that there are no collinearity issues among the selected regressors.

Figure 14.7 Parameter Estimates with Additional Statistics

The HPREG Procedure

Selected Model

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Pr > |t|	Variance Inflation	95% Confidence Limits	
Intercept	1	4.013911	0.111290	36.07	<.0001	0	3.79476	4.23306
nHits	1	0.007929	0.000994	7.98	<.0001	1.49642	0.00597	0.00989
nBB	1	0.007280	0.002049	3.55	0.0005	1.52109	0.00325	0.01131
YrMajor	1	0.100663	0.007551	13.33	<.0001	1.02488	0.08579	0.11553

By default, high-performance statistical procedures do not include all variables from the input data set in output data sets. The ID statement specifies that the variable name in the input data set be added as an identification variable in the baseballOut data set that is produced by the OUTPUT statement. In addition to this variable, the OUTPUT statement requests that predicted values, raw residuals, leverage values, Cook's D statistics, and studentized residuals be added in the output data set. Note that default names are used for these statistics except for the predicted values for which a specified name, predictedLogSalary, is supplied. The following statements use PROC PRINT to display the first five observations of this output data set:

```
proc print data=baseballOut(obs=5);
run;
```

Figure 14.8 First 5 Observations of the baseballOut Data Set

Obs	Name	predictedLogSalary	Residual	H	COOKD	RSTUDENT
1	Allanson, Andy	4.73980	.	0.016087	.	.
2	Ashby, Alan	6.34935	-0.18603	0.012645	.000335535	-0.32316
3	Davis, Alan	5.89993	0.27385	0.019909	.001161794	0.47759
4	Dawson, Andre	6.50852	-0.29392	0.011060	.000730178	-0.51031
5	Galarraga, Andres	5.12344	-0.60711	0.009684	.002720358	-1.05510

Syntax: HPREG Procedure

The following statements are available in the HPREG procedure:

PROC HPREG < *options* > **;**
BY *variables* **;**
CODE < *options* > **;**
CLASS *variable* < **(***options***)** >. . . < *variable* < **(***options***)** > > < / *global-options* > **;**
MODEL *dependent* **=** < *effects* > < / *model-options* > **;**
OUTPUT < **OUT=***SAS-data-set* >
< *keyword* < **=***name* > >. . .
< *keyword* < **=***name* > > < / *options* > **;**
PARTITION < *partition-options* > **;**
PERFORMANCE < *performance-options* > **;**
SELECTION *options* **;**
FREQ *variable* **;**
ID *variables* **;**
WEIGHT *variable* **;**

The PROC HPREG statement and a single MODEL statement are required. All other statements are optional. The CLASS statement can appear multiple times. If a CLASS statement is specified, it must precede the MODEL statement.

PROC HPREG Statement

PROC HPREG < *options* > **;**

The PROC HPREG statement invokes the procedure. Table 14.1 summarizes the options in the PROC HPREG statement by function.

Table 14.1 PROC HPREG Statement Options

Option	Description
Basic Options	
DATA=	Specifies the input data set
NAMELEN=	Limits the length of effect names
Options Related to Output	
NOPRINT	Suppresses ODS output
NOCLPRINT	Limits or suppresses the display of class levels
User-Defined Formats	
FMTLIBXML=	Specifies a file reference for a format stream
Other Options	
ALPHA=	Sets the significance level used for the construction of confidence intervals
SEED=	Sets the seed used for pseudorandom number generation

Following are explanations of the *options* that you can specify in the PROC HPREG statement (in alphabetical order):

ALPHA=*number*
sets the significance level used for the construction of confidence intervals. The value must be between 0 and 1; the default value of 0.05 results in 95% intervals. This option affects the OUTPUT statement keywords LCL, LCLM, UCL, and UCLM, and the CLB option in the MODEL statement.

DATA=*SAS-data-set*
names the input SAS data set to be used by PROC HPREG. The default is the most recently created data set.

If the procedure executes in distributed mode, the input data are distributed to memory on the appliance nodes and analyzed in parallel, unless the data are already distributed in the appliance database. In that case the procedure reads the data alongside the distributed database. See the section "Processing Modes" on page 10 about the various execution modes and the section "Alongside-the-Database Execution" on page 18 about the alongside-the-database model. Both sections are in Chapter 3, "Shared Concepts and Topics."

FMTLIBXML=*file-ref*
specifies the file reference for the XML stream that contains the user-defined format definitions. User-defined formats are handled differently in a distributed computing environment than they are in other SAS products. See the section "Working with Formats" on page 35 in Chapter 3, "Shared Concepts and Topics," for details about how to generate a XML stream for your formats.

NAMELEN=*number*
specifies the length to which long effect names are shortened. The default and minimum value is 20.

NOCLPRINT<=*number***>**
suppresses the display of the "Class Level Information" table if you do not specify *number*. If you specify *number*, the values of the classification variables are displayed for only those variables whose number of levels is less than *number*. Specifying a *number* helps to reduce the size of the "Class Level Information" table if some classification variables have a large number of levels.

NOPRINT
suppresses the generation of ODS output.

SEED=*number*
specifies an integer used to start the pseudorandom number generator for random partitioning of data for training, testing, and validation. If you do not specify a seed, or if you specify a value less than or equal to 0, the seed is generated from reading the time of day from the computer's clock.

BY Statement

BY *variables* **;**

You can specify a BY statement with PROC HPREG to obtain separate analyses of observations in groups that are 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 you specify more than one BY statement, only the last one specified is used.

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 HPREG 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 (in Base SAS software).

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

CLASS Statement

CLASS *variable* <**(***options***)**>. . . <*variable* <**(***options***)**>> </ *global-options*> **;**

The CLASS statement names the classification variables to be used as explanatory variables in the analysis. The CLASS statement must precede the MODEL statement.

The CLASS statement for SAS high-performance statistical procedures is documented in the section "CLASS Statement" on page 42 of Chapter 4, "Shared Statistical Concepts." The HPREG procedure also supports the following *global-option* in the CLASS statement:

UPCASE
uppercases the values of character-valued CLASS variables before levelizing them. For example, if the UPCASE option is in effect and a CLASS variable can take the values 'a', 'A', and 'b', then 'a' and 'A' represent the same level and the CLASS variable is treated as having only two values: 'A' and 'B'.

CODE Statement

CODE <*options*> **;**

The CODE statement writes SAS DATA step code for computing predicted values of the fitted model either to a file or to a catalog entry. This code can then be included in a DATA step to score new data.

Table 14.2 summarizes the *options* available in the CODE statement.

Table 14.2 CODE Statement Options

Option	Description
CATALOG=	Names the catalog entry where the generated code is saved
DUMMIES	Retains the dummy variables in the data set
ERROR	Computes the error function
FILE=	Names the file where the generated code is saved
FORMAT=	Specifies the numeric format for the regression coefficients

Table 14.2 *continued*

Option	Description
GROUP=	Specifies the group identifier for array names and statement labels
IMPUTE	Imputes predicted values for observations with missing or invalid covariates
LINESIZE=	Specifies the line size of the generated code
LOOKUP=	Specifies the algorithm for looking up CLASS levels
RESIDUAL	Computes residuals

For details about the syntax of the CODE statement, see the section "CODE Statement" (Chapter 19, *SAS/STAT User's Guide*) in Chapter 19, "Shared Concepts and Topics" (*SAS/STAT User's Guide*).

FREQ Statement

FREQ *variable* **;**

The *variable* in the FREQ statement identifies a numeric variable in the data set that contains the frequency of occurrence for each observation. SAS high-performance statistical procedures that support the FREQ statement treat each observation as if it appeared f times, where f is the value of the FREQ variable for the observation. If the frequency value is not an integer, it is truncated to an integer. If the frequency value is less than 1 or missing, the observation is not used in the analysis. When the FREQ statement is not specified, each observation is assigned a frequency of 1.

ID Statement

ID *variables* **;**

The ID statement lists one or more variables from the input data set that are transferred to output data sets created by SAS high-performance statistical procedures, provided that the output data set produces one (or more) records per input observation.

For documentation on the common ID statement in SAS high-performance statistical procedures, see the section "ID Statement" on page 46 in Chapter 4, "Shared Statistical Concepts."

MODEL Statement

MODEL *dependent=< effects > / < options >* **;**

The MODEL statement names the dependent variable and the explanatory effects, including covariates, main effects, interactions, and nested effects. If you omit the explanatory effects, the procedure fits an intercept-only model.

After the keyword MODEL, the dependent (response) variable is specified, followed by an equal sign. The explanatory effects follow the equal sign. For information about constructing the model effects, see the

section "Specification and Parameterization of Model Effects" on page 55, of Chapter 4, "Shared Statistical Concepts."

You can specify the following *options* in the MODEL statement after a slash (/):

CLB
requests the $100(1-\alpha)\%$ upper and lower confidence limits for the parameter estimates. By default, the 95% limits are computed; the ALPHA= option in the PROC HPREG statement can be used to change the α level. The CLB option is not supported when you request METHOD=LAR or METHOD=LASSO in the SELECTION statement.

INCLUDE=*n*

INCLUDE=*single-effect*

INCLUDE=(*effects***)**
forces effects to be included in all models. If you specify INCLUDE=*n*, then the first *n* effects listed in the MODEL statement are included in all models. If you specify INCLUDE=*single-effect* or if you specify a list of effects within parentheses, then the specified effects are forced into all models. The effects that you specify in the INCLUDE= option must be explanatory effects defined in the MODEL statement before the slash (/). The INCLUDE= option is not available when you specify METHOD=LAR or METHOD=LASSO in the SELECTION statement.

NOINT
suppresses the intercept term that is otherwise included in the model.

ORDERSELECT
specifies that, for the selected model, effects be displayed in the order in which they first entered the model. If you do not specify the ORDERSELECT option, then effects in the selected model are displayed in the order in which they appear in the MODEL statement.

START=*n*

START=*single-effect*

START=(*effects***)**
is used to begin the selection process in the FORWARD, FORWARDSWAP, and STEPWISE selection methods from the initial model that you designate. If you specify START=*n*, then the starting model consists of the first *n* effects listed in the MODEL statement. If you specify START=*single-effect* or if you specify a list of effects within parentheses, then the starting model consists of these specified effects. The effects that you specify in the START= option must be explanatory effects defined in the MODEL statement before the slash (/). The START= option is not available when you specify METHOD=BACKWARD, METHOD=LAR, or METHOD=LASSO in the SELECTION statement.

STB
produces standardized regression coefficients. A standardized regression coefficient is computed by dividing a parameter estimate by the ratio of the sample standard deviation of the dependent variable to the sample standard deviation of the regressor.

TOL
produces tolerance values for the estimates. Tolerance for a parameter is defined as $1 - R^2$, where R^2 is obtained from the regression of the parameter on all other parameters in the model. The TOL option is not supported when you request METHOD=LAR or METHOD=LASSO in the SELECTION statement.

VIF

produces variance inflation factors with the parameter estimates. Variance inflation is the reciprocal of tolerance. The VIF option is not supported when you request METHOD=LAR or METHOD=LASSO in the SELECTION statement.

OUTPUT Statement

OUTPUT < **OUT=***SAS-data-set* >
< **COPYVARS=***(variables)* >
< *keyword* < *=name* > >. . . < *keyword* < *=name* > > **;**

The OUTPUT statement creates a data set that contains observationwise statistics, which are computed after fitting the model. The variables in the input data set are *not* included in the output data set to avoid data duplication for large data sets; however, variables specified in the ID statement or COPYVARS= option are included.

If the input data are in distributed form, where access of data in a particular order cannot be guaranteed, the HPREG procedure copies the distribution or partition key to the output data set so that its contents can be joined with the input data.

The output statistics are computed based on the parameter estimates for the selected model.

You can specify the following syntax elements in the OUTPUT statement:

OUT=*SAS-data-set*

DATA=*SAS-data-set*

specifies the name of the output data set. If the OUT= (or DATA=) option is omitted, the procedure uses the DATA*n* convention to name the output data set.

COPYVAR=*variable*

COPYVARS=(*variables***)**

transfers one or more *variables* from the input data set to the output data set. Variables named in an ID statement are also copied from the input data set to the output data set.

keyword **< =***name* **>**

specifies the statistics to include in the output data set and optionally names the new variables that contain the statistics. Specify a keyword for each desired statistic (see the following list of keywords), followed optionally by an equal sign and a variable to contain the statistic.

If you specify *keyword=name*, the new variable that contains the requested statistic has the specified name. If you omit the optional *=name* after a *keyword*, then a default name is used.

The following are valid values for *keyword* to request statistics that are available with all selection methods:

PREDICTED

PRED

P

requests predicted values for the response variable. The default name is Pred.

RESIDUAL

RESID

R

requests the residual, calculated as ACTUAL–PREDICTED. The default name is Residual.

ROLE

requests a numeric variable that indicates the role played by each observation in fitting the model. The default name is _ROLE_. For each observation the interpretation of this variable is shown in Table 14.3:

Table 14.3 Role Interpretation

Value	Observation Role
0	Not used
1	Training
2	Validation
3	Testing

If you do not partition the input data by using a PARTITION statement, then the role variable value is 1 for observations used in fitting the model, and 0 for observations that have at least one missing or invalid value for the response, regressors, frequency or weight variables.

In addition to the preceding statistics, you can also use the *keywords* listed in Table 14.4 in the OUTPUT statement to obtain additional statistics. These statistics are not available if you use METHOD=LAR or METHOD=LASSO in the SELECTION statement, unless you also specify the LSCOEFFS option. See the section "Diagnostic Statistics" on page 560 for computational formulas. All the statistics available in the OUTPUT statement are conditional on the selected model and do not take into account the variability introduced by doing model selection.

Table 14.4 Keywords for OUTPUT Statement

Keyword	Description
COOKD	Cook's D influence statistic
COVRATIO	Standard influence of observation on covariance of betas
DFFIT	Standard influence of observation on predicted value
H	Leverage, $\mathbf{x}_i(\mathbf{X}'\mathbf{X})^{-}\mathbf{x}_i'$
LCL	Lower bound of a $100(1-\alpha)\%$ confidence interval for an individual prediction. This includes the variance of the error, as well as the variance of the parameter estimates.
LCLM	Lower bound of a $100(1-\alpha)\%$ confidence interval for the expected value (mean) of the dependent variable
PRESS	ith residual divided by $(1-h)$, where h is the leverage, and where the model has been refit without the ith observation
RSTUDENT	A studentized residual with the current observation deleted
STDI	Standard error of the individual predicted value
STDP	Standard error of the mean predicted value
STDR	Standard error of the residual

Table 14.4 *continued*

Keyword	Description
STUDENT	Studentized residuals, which are the residuals divided by their standard errors
UCL	Upper bound of a $100(1-\alpha)\%$ confidence interval for an individual prediction
UCLM	Upper bound of a $100(1-\alpha)\%$ confidence interval for the expected value (mean) of the dependent variable

PARTITION Statement

PARTITION < *partition-options* > **;**

The PARTITION statement specifies how observations in the input data set are logically partitioned into disjoint subsets for model training, validation, and testing. Either you can designate a variable in the input data set and a set of formatted values of that variable to determine the role of each observation, or you can specify proportions to use for random assignment of observations for each role.

The following mutually exclusive *partition-options* are available:

ROLEVAR | ROLE=*variable***(< TEST='***value***' > < TRAIN='***value***' > < VALIDATE='***value***' >)**
names the variable in the input data set whose values are used to assign roles to each observation. The formatted values of this variable that are used to assign observations roles are specified in the TEST=, TRAIN=, and VALIDATE= suboptions. If you do not specify the TRAIN= suboption, then all observations whose role is not determined by the TEST= or VALIDATE= suboptions are assigned to training.

FRACTION(< TEST=*fraction* > < VALIDATE=*fraction* >)**
requests that specified proportions of the observations in the input data set be randomly assigned training and validation roles. You specify the proportions for testing and validation by using the TEST= and VALIDATE= suboptions. If you specify both the TEST= and the VALIDATE= suboptions, then the sum of the specified fractions must be less than 1 and the remaining fraction of the observations are assigned to the training role.

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 the HPREG procedure.

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

The PERFORMANCE statement is documented further in the section "PERFORMANCE Statement" on page 36.

SELECTION Statement

SELECTION < *options* > **;**

The SELECTION statement performs variable selection. All *options* except the SCREEN option are fully documented in the section "SELECTION Statement" on page 47 in Chapter 4, "Shared Statistical Concepts." The SCREEN option is described in the following section. The remainder of this section describes specific information about how PROC HPREG implements the METHOD= option and the DETAILS= option.

The HPREG procedure supports the following values of the METHOD= option in the SELECTION statement:

NONE specifies no model selection.

FORWARD specifies the forward selection method, which starts with no effects in the model and adds effects.

BACKWARD specifies the backward elimination method, which starts with all effects in the model and deletes effects.

STEPWISE specifies the stepwise regression method, which is similar to the forward selection method except that effects already in the model do not necessarily stay there.

FORWARDSWAP specifies the forward-swap selection method, which is an extension of the forward selection method. Before any addition step, PROC HPREG makes all pairwise swaps of effects in and out of the current model that improve the selection criterion. When the selection criterion is R square, this method is the same as the MAXR method in the REG procedure in SAS/STAT software.

LAR specifies the least angle regression method. Like forward selection, this method starts with no effects in the model and adds effects. The parameter estimates at any step are "shrunk" when compared to the corresponding least squares estimates. If the model contains classification variables, then these classification variables are split. For more information, see the SPLIT option in the CLASS statement.

LASSO specifies the lasso method, which adds and deletes parameters based on a version of ordinary least squares in which the sum of the absolute regression coefficients is constrained. If the model contains classification variables, then these classification variables are split. For more information, see the SPLIT option in the CLASS statement.

The DETAILS=ALL and DETAILS=STEPS options produce the "ANOVA," "Fit Statistics," and "Parameter Estimates" tables, which provide information about the model that is selected at each step of the selection process.

In addition to other options, which are fully documented in the section "SELECTION Statement" on page 47 in Chapter 4, "Shared Statistical Concepts," PROC HPREG also supports a SCREEN option, which has the following syntax:

SCREEN < **(***global-screen-options***)** > < **=***screen-options* >

You can specify following *global-screen-options*:

DETAILS=NONE | SUMMARY | ALL
specifies the level of detail to be produced about the screening process. You can specify the following values:

NONE suppresses all tables that provide details of the screening process.

ALL produces the following output and shows model selection details at each stage of the screening process:

- a screening table that shows the correlations that are used to obtain the screened effects for the first two stages of the screening process
- a screened effects table that lists the effects that are chosen at each stage of the screening process

SUMMARY produces the following output and shows details about the model selection only for the final stage of the screening process:

- a screening table that shows the correlations that are used to obtain the screened effects for the first two stages of the screening process
- a screened effects table that lists the effects that are chosen at each stage of the screening process

By default, DETAILS=SUMMARY.

SINGLESTAGE
screens effects and selects a model only once.

MULTISTAGE
performs multiple stages, each of which contains a screening and a model selection step.

You can specify the following *screen-options* after an = sign:

SCREEN=*n1* **<** *n2* **>**
specifies the number of effects to be chosen at the first two stages of the screening process. If you specify only *n1*, then *n1* is used for both the first and second stages. If you specify both *n1* and *n2*, then *n1* is used at the first stage and *n2* is used at the second stage. At the first stage, effects are ranked in decreasing order of the magnitude of their pairwise correlations with the response, and the first *n1* effects are used in the selection process at that stage. At the second stage, effects are ranked in decreasing order of the magnitude of their pairwise correlations with the residuals obtained at the first stage, and the first *n2* effects are used in the selection process at that stage.

SCREEN=PERCENT(*p1* **<** *p2* **>)**
specifies the percentage of effects in the MODEL statement to be chosen at the first two stages of the screening process. If you specify only *p1*, then *p1* is used for both the first and second stages. If you specify *p1* and *p2*, then *p1* is used at the first stage and *p2* is used at the second stage.

SCREEN=CUTOFF(*c1* **<** *c2* **>)**
specifies the minimum value of the screening statistic that effects must have in order to be chosen at the first two stages of the screening process. If you specify only *c1*, then *c1* is used for both the first and second stages. If you specify both *c1* and *c2*, then *c1* is used at the first stage and *c2* is used at the second stage. At the first stage, any effect whose absolute pairwise correlation with the response is less than the first-stage cutoff is not used in the selection process at that stage. At the second stage, any

effect whose absolute pairwise correlation with the residuals obtained from the first stage is less than the second-stage cutoff is not used in the selection process at that stage.

If you do not specify any *screen-options*, SCREEN=PERCENT(10) by default.

For a classification effect that has multiple degrees of freedom, pairwise correlations with the response at the first stage and the first stage residuals at the second stage are computed separately for each dummy variable that corresponds to the levels of the classification variables in the effect. The largest magnitude of these correlations is used as a proxy for the correlation statistic for that effect.

WEIGHT Statement

WEIGHT *variable* **;**

The *variable* in the WEIGHT statement is used as a weight to perform a weighted analysis of the data. Observations with nonpositive or missing weights are not included in the analysis. If a WEIGHT statement is not included, all observations used in the analysis are assigned a weight of 1.

Details: HPREG Procedure

Criteria Used in Model Selection

The HPREG procedure supports a variety of fit statistics that you can specify as criteria for the CHOOSE=, SELECT=, and STOP= options in the SELECTION statement. The following statistics are available:

ADJRSQ	Adjusted R-square statistic (Darlington 1968; Judge et al. 1985)
AIC	Akaike's information criterion (Akaike 1969; Judge et al. 1985)
AICC	Corrected Akaike's information criterion (Hurvich and Tsai 1989)
BIC | SBC	Schwarz Bayesian information criterion (Schwarz 1978; Judge et al. 1985)
CP	Mallows C_p statistic (Mallows 1973; Hocking 1976)
PRESS	Predicted residual sum of squares statistic
RSQUARE	R-square statistic (Darlington 1968; Judge et al. 1985)
SL	Significance used to assess an effect's contribution to the fit when it is added to or removed from a model
VALIDATE	Average square error over the validation data

When you use SL as a criterion for effect selection, the definition depends on whether an effect is being considered as a drop or an add candidate. If the current model has p parameters excluding the intercept, and if you denote its residual sum of squares by RSS_p and you add an effect with k degrees of freedom and denote the residual sum of squares of the resulting model by RSS_{p+k}, then the F statistic for entry with k numerator degrees of freedom and $n-(p+k)-1$ denominator degrees of freedom is given by

$$F = \frac{(\mathrm{RSS}_p - \mathrm{RSS}_{p+k})/k}{\mathrm{RSS}_{p+k}/(n-(p+k)-1)}$$

where n is number of observations used in the analysis. The significance level for entry is the p-value of this F statistic, and is deemed significant if it is smaller than the SLENTRY limit. Among several such add candidates, the effect with the smallest p-value (most significant) is deemed best.

If you drop an effect with k degrees of freedom and denote the residual sum of squares of the resulting model by RSS_{p-k}, then the F statistic for removal with k numerator degrees of freedom and $n-p-k$ denominator degrees of freedom is given by

$$F = \frac{(\text{RSS}_{p-k} - \text{RSS}_p)/k}{\text{RSS}_p/(n-p-k)}$$

where n is number of observations used in the analysis. The significance level for removal is the p-value of this F statistic, and the effect is deemed not significant if this p-value is larger than the SLSTAY limit. Among several such removal candidates, the effect with the largest p-value (least significant) is deemed the best removal candidate.

It is known that the "F-to-enter" and "F-to-delete" statistics do not follow an F distribution (Draper, Guttman, and Kanemasu 1971).. Hence the SLENTRY and SLSTAY values cannot reliably be viewed as probabilities. One way to address this difficulty is to replace hypothesis testing as a means of selecting a model with information criteria or out-of-sample prediction criteria. While Harrell (2001) points out that information criteria were developed for comparing only prespecified models, Burnham and Anderson (2002) note that AIC criteria have routinely been used for several decades for performing model selection in time series analysis.

Table 14.5 provides formulas and definitions for these fit statistics.

Table 14.5 Formulas and Definitions for Model Fit Summary Statistics

Statistic	Definition or Formula
n	Number of observations
p	Number of parameters including the intercept
$\hat{\sigma}^2$	Estimate of pure error variance from fitting the full model
SST	Total sum of squares corrected for the mean for the dependent variable
SSE	Error sum of squares
ASE	$\frac{\text{SSE}}{n}$
MSE	$\frac{\text{SSE}}{n-p}$
R^2	$1 - \frac{\text{SSE}}{\text{SST}}$
ADJRSQ	$1 - \frac{(n-1)(1-R^2)}{n-p}$
AIC	$n \ln\left(\frac{\text{SSE}}{n}\right) + 2p$
AICC	$1 + \ln\left(\frac{\text{SSE}}{n}\right) + \frac{2(p+1)}{n-p-2}$
CP (C_p)	$\frac{\text{SSE}}{\hat{\sigma}^2} + 2p - n$

Table 14.5 *continued*

Statistic	Definition or Formula
PRESS	$\sum_{i=1}^{n} \frac{r_i^2}{(1-h_i)^2}$ where $r_i =$ residual at observation i and $h_i =$ leverage of observation $i = \mathbf{x}_i(\mathbf{X}'\mathbf{X})^{-}\mathbf{x}_i'$
RMSE	$\sqrt{\text{MSE}}$
SBC	$n \ln\left(\frac{\text{SSE}}{n}\right) + p \ln(n)$

Diagnostic Statistics

This section gathers the formulas for the statistics available in the OUTPUT statement. All the statistics available in the OUTPUT statement are conditional on the selected model and do not take into account the variability introduced by doing model selection.

The model to be fit is $\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, and the parameter estimate is denoted by $\mathbf{b} = (\mathbf{X}'\mathbf{X})^{-}\mathbf{X}'\mathbf{Y}$. The subscript i denotes values for the ith observation, and the parenthetical subscript (i) means that the statistic is computed by using all observations except the ith observation.

The ALPHA= option in the PROC HPREG statement is used to set the α value for the confidence limit statistics.

Table 14.6 contains the diagnostic statistics and their formulas. Each statistic is computed for each observation.

Table 14.6 Formulas and Definitions for Diagnostic Statistics

MODEL Option or Statistic	Formula
PRED $(\widehat{\mathbf{Y}}_i)$	$\mathbf{X}_i\mathbf{b}$
RES (r_i)	$\mathbf{Y}_i - \widehat{\mathbf{Y}}_i$
H (h_i)	$\mathbf{x}_i(\mathbf{X}'\mathbf{X})^{-}\mathbf{x}_i'$
STDP	$\sqrt{h_i\widehat{\sigma}^2}$
STDI	$\sqrt{(1+h_i)\widehat{\sigma}^2}$
STDR	$\sqrt{(1-h_i)\widehat{\sigma}^2}$
LCL	$\widehat{Y}_i - t_{\frac{\alpha}{2}}\text{STDI}$
LCLM	$\widehat{Y}_i - t_{\frac{\alpha}{2}}\text{STDP}$
UCL	$\widehat{Y}_i + t_{\frac{\alpha}{2}}\text{STDI}$
UCLM	$\widehat{Y}_i + t_{\frac{\alpha}{2}}\text{STDP}$
STUDENT	$\frac{r_i}{\text{STDR}_i}$

Table 14.6 *continued*

MODEL Option or Statistic	Formula
RSTUDENT	$\frac{r_i}{\hat{\sigma}_{(i)}\sqrt{1-h_i}}$
COOKD	$\frac{1}{p}\text{STUDENT}^2\frac{\text{STDP}^2}{\text{STDR}^2}$
COVRATIO	$\frac{\det(\hat{\sigma}^2_{(i)}(\mathbf{x}'_{(i)}\mathbf{x}_{(i)})^{-1}}{\det(\hat{\sigma}^2(\mathbf{X}'\mathbf{X})^{-1})}$
DFFITS	$\frac{(\widehat{\mathbf{Y}}_i - \widehat{\mathbf{Y}}_{(i)})}{(\hat{\sigma}_{(i)}\sqrt{h_i})}$
PRESS(predr$_i$)	$\frac{r_i}{1-h_i}$

Classification Variables and the SPLIT Option

PROC HPREG supports the ability to split classification variables when doing model selection. You use the SPLIT option in the CLASS statement to specify that the columns of the design matrix that correspond to effects that contain a split classification variable can enter or leave a model independently of the other design columns of that effect. The following statements illustrate the use of SPLIT option:

```
data splitExample;
   length c2 $6;
   drop i;
   do i=1 to 1000;
     c1 = 1 + mod(i,6);
     if      i < 250 then c2 = 'low';
     else if i < 500 then c2 = 'medium';
     else                 c2 = 'high';
     x1 = ranuni(1);
     x2 = ranuni(1);
     y = x1+3*(c2 ='low')  + 10*(c1=3) +5*(c1=5) + rannor(1);
     output;
   end;
run;

proc hpreg data=splitExample;
   class c1(split) c2(order=data);
   model y = c1 c2 x1 x2/orderselect;
   selection method=forward;
run;
```

The "Class Levels" table shown in Figure 14.9 is produced by default whenever you specify a CLASS statement.

Figure 14.9 Class Levels

The HPREG Procedure

Class Level Information		
Class	Levels	Values
c1	6	* 1 2 3 4 5 6
c2	3	low medium high

*** Associated Parameters Split**

The SPLIT option has been specified for the classification variable c1. This permits the parameters associated with the effect c1 to enter or leave the model individually. The "Parameter Estimates" table in Figure 14.10 shows that for this example the parameters that correspond to only levels 3 and 5 of c1 are in the selected model. Finally, note that the ORDERSELECT option in the MODEL statement specifies that the parameters be displayed in the order in which they first entered the model.

Figure 14.10 Parameter Estimates

Parameter Estimates					
Parameter	DF	Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	-0.308111	0.075387	-4.09	<.0001
c1_3	1	10.161702	0.087601	116.00	<.0001
c1_5	1	5.018407	0.087587	57.30	<.0001
c2 low	1	3.139941	0.078495	40.00	<.0001
c2 medium	1	0.221539	0.078364	2.83	0.0048
c2 high	0	0	.	.	.
x1	1	1.317420	0.109510	12.03	<.0001

Using Validation and Test Data

When you have sufficient data, you can subdivide your data into three parts called the training, validation, and test data. During the selection process, models are fit on the training data, and the prediction error for the models so obtained is found by using the validation data. This prediction error on the validation data can be used to decide when to terminate the selection process or to decide what effects to include as the selection process proceeds. Finally, after a selected model has been obtained, the test set can be used to assess how the selected model generalizes on data that played no role in selecting the model.

In some cases you might want to use only training and test data. For example, you might decide to use an information criterion to decide what effects to include and when to terminate the selection process. In this case no validation data are required, but test data can still be useful in assessing the predictive performance of the selected model. In other cases you might decide to use validation data during the selection process but forgo assessing the selected model on test data. Hastie, Tibshirani, and Friedman (2001) note that it is difficult to give a general rule for how many observations you should assign to each role. They note that a typical split might be 50% for training and 25% each for validation and testing.

You use a PARTITION statement to logically subdivide the DATA= data set into separate roles. You can name the fractions of the data that you want to reserve as test data and validation data. For example, the

following statements randomly subdivide the "inData" data set, reserving 50% for training and 25% each for validation and testing:

```
proc hpreg data=inData;
  partition fraction(test=0.25 validate=0.25);
  ...
run;
```

In some cases you might need to exercise more control over the partitioning of the input data set. You can do this by naming both a variable in the input data set and also a formatted value of that variable that correspond to each role. For example, the following statements assign roles to the observations in the "inData" data set based on the value of the variable group in that data set. Observations where the value of group is 'group 1' are assigned for testing, and those with value 'group 2' are assigned to training. All other observations are ignored.

```
proc hpreg data=inData;
  partition roleVar=group(test='group 1' train='group 2')
  ...
run;
```

When you have reserved observations for training, validation, and testing, a model fit on the training data is scored on the validation and test data, and the average squared error (ASE) is computed separately for each of these subsets. The ASE for each data role is the error sum of squares for observations in that role divided by the number of observations in that role.

Using the Validation ASE as the STOP= Criterion

If you have provided observations for validation, then you can specify STOP=VALIDATE as a suboption of the METHOD= option in the SELECTION statement. At step *k* of the selection process, the best candidate effect to enter or leave the current model is determined. Here "best candidate" means the effect that gives the best value of the SELECT= criterion; this criterion need not be based on the validation data. The validation ASE for the model with this candidate effect added or removed is computed. If this validation ASE is greater than the validation ASE for the model at step *k*, then the selection process terminates at step *k*.

Using the Validation ASE as the CHOOSE= Criterion

When you specify the CHOOSE=VALIDATE suboption of the METHOD= option in the SELECTION statement, the validation ASE is computed for the models at each step of the selection process. The smallest model at any step that yields the smallest validation ASE is selected.

Using the Validation ASE as the SELECT= Criterion

You request the validation ASE as the selection criterion by specifying the SELECT=VALIDATE suboption of the METHOD= option in the SELECTION statement. At step *k* of the selection process, the validation ASE is computed for each model in which a candidate for entry is added or candidate for removal is dropped. The selected candidate for entry or removal is the one that yields a model with the minimal validation ASE. This method is computationally very expensive because validation statistics need to be computed for every candidate at every step; it should be used only with small data sets or models with a small number of regressors.

Computational Method

Multithreading

Threading refers to the organization of computational work into multiple tasks (processing units that can be scheduled by the operating system). A task is associated with a thread. Multithreading refers to the concurrent execution of threads. When multithreading is possible, substantial performance gains can be realized compared to sequential (single-threaded) execution.

The number of threads spawned by the HPREG procedure is determined by the number of CPUs on a machine and can be controlled in the following ways:

- You can specify the CPU count with the CPUCOUNT= SAS system option. For example, if you specify the following statements, the HPREG procedure schedules threads as if it executes on a system with four CPUs, regardless of the actual CPU count.

```
options cpucount=4;
```

- You can specify the NTHREADS= option in the PERFORMANCE statement to determine the number of threads. This specification overrides the system option. Specify NTHREADS=1 to force single-threaded execution.

The number of threads per machine is displayed in the "Performance Information" table, which is part of the default output. The HPREG procedure allocates one thread per CPU.

The tasks multithreaded by the HPREG procedures are primarily defined by dividing the data processed on a single machine among the threads—that is, the HPREG procedure implements multithreading through a data-parallel model. For example, if the input data set has 1,000 observations and you are running with four threads, then 250 observations are associated with each thread. All operations that require access to the data are then multithreaded. This operations include the following:

- variable levelization
- effect levelization
- formation of the crossproducts matrix
- evaluation of predicted residual sums of squares on validation and test data
- scoring of observations

In addition, operations on matrices such as sweeps might be multithreaded if the matrices are of sufficient size to realize performance benefits from managing multiple threads for the particular matrix operation.

Output Data Set

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

- those variables explicitly created by the statement
- variables listed in the ID statement
- distribution keys or hash keys that are transferred from the input data set

This enables you to add output data set information that is necessary for subsequent SQL joins without copying the entire input data set to the output data set. For more information about output data sets that are produced when PROC HPREG is run in distributed mode, see the section "Output Data Sets" on page 34 in Chapter 3, "Shared Concepts and Topics."

Screening

Model selection from a very large number of effects is computationally demanding. For example, in analyzing microarray data, where each dot in the array corresponds to a regressor, having 35,000 such regressors is not uncommon. Another source of such large regression problems arises when you want to consider all possible two-way interactions of your main effects as candidates for inclusion in a selected model. See Foster and Stine (2004) for an example that uses this approach to build a predictive model for bankruptcy.

In recent years, there has been a resurgence of interest in combining variable selection methods with an initial screening step that reduces the large number of regressors to a much smaller subset from which the final model is chosen. You can find theoretical underpinnings of this approach in Fan and Lv (2008). See El Ghaoui, Viallon, and Rabbani (2012) and Tibshirani et al. (2012) for examples where screening has also been incorporated in the context of penalized regression methods (such as lasso) for performing model selection.

Screening uses a screening statistic that is inexpensive to compute in order to eliminate from consideration regressors that are unlikely to be selected if you included them in variable selection. For linear regression, you can use the magnitude of the correlation between each individual regressor and the response as such a screening statistic. The square of the correlation between a regressor that has one degree of freedom and the response is the R-square value for the univariate regression for the response with this regressor. Hence, screening by the magnitude of the pairwise correlations is equivalent to fitting univariate models to do the screening.

The first stage of the screening method chooses only the subset of regressors whose screening statistic is larger than a specified cutoff value or by choosing those regressors whose screening statistics are among a specified number or percentage of the largest screening statistic values. Then you perform model selection for the response from this screened subset of the original regressors.

One problem with this approach is that a regressor that is pairwise (marginally) uncorrelated or has very small correlation with the response can nevertheless be an important predictor, but it would be eliminated in the screening. You can address this problem by switching to a multistage approach. The first stage consists

of screening the regressors and selecting the model for the response from the screened subset. The second stage repeats the first stage except that you use the residuals from the first stage as the response variable in this second stage. You can iterate this process by using the residuals from the previous stage as the response for the next stage. The final stage forms the union of all the screened regressors from the first stage with all the selected regressors at the subsequent stages and selects a model for the original response variable from this union.

Experimentation has shown that there is little benefit in practice in using more than one stage where the response is the residual from the previous stage. Hence, PROC HPREG implements a three-stage process by default. However, if you specify the SINGLESTAGE suboption in the SCREEN option in the SELECTION statement, then only the first screening stage is performed.

Displayed Output

The following sections describe the output produced by PROC HPREG. The output is organized into various tables, which are discussed in the order of appearance.

Performance Information

The "Performance Information" table is produced by default. It displays information about the execution mode. For single-machine mode, the table displays the number of threads used. For distributed mode, the table displays the the number of compute nodes, and the number of threads per node.

Data Access Information

The "Data Access Information" table is produced by default. For the input and output data sets, it displays the libref and data set name, the engine used to access the data, the role (input or output) of the data set, and path that data followed to the reach the computation.

Model Information

The "Model Information" table displays basic information about the model, such as the response variable, frequency variable, weight variable, and the type of parameterization used for classification variables named in the CLASS statement.

Selection Information

When you specify the SELECTION statement, the HPREG procedure produces by default a series of tables with information about the model selection. The "Selection Information" table informs you about the model selection method; select, stop, and choose criteria; and other parameters that govern the selection. You can suppress this table by specifying DETAILS=NONE in the SELECTION statement.

Screening Information

When you specify the SCREEN option in the SELECTION statement, the "Screening Information" table informs you about the the number of screening stages used and informs you about the method and values that are used to determine how many screened effects are chosen at each screening stage.

Screening

When you specify the DETAILS=ALL suboption of the SCREEN option in the SELECTION statement, the the "Screening" table displays the model effects and their screening statistic values in descending order of the screening statistic values.

Screened Effects

When you specify the SCREEN option in the SELECTION statement, the "Screened Effects" table displays a list of the screened model effects at each stage of the screening process.

Number of Observations

The "Number of Observations" table displays the number of observations read from the input data set and the number of observations used in the analysis. If you specify a FREQ statement, the table also displays the sum of frequencies read and used. If you use a PARTITION statement, the table also displays the number of observations used for each data role.

Class Level Information

The "Class Level Information" table lists the levels of every variable specified in the CLASS statement. You should check this information to make sure that the data are correct. You can adjust the order of the CLASS variable levels with the ORDER= option in the CLASS statement. You can suppress the "Class Level Information" table completely or partially with the NOCLPRINT= option in the PROC HPREG statement.

If the classification variables are in the reference parameterization, the "Class Level Information" table also displays the reference value for each variable. The "Class Level Information" table also indicates which, if any, of the classification variables are split by using the SPLIT option in the CLASS statement.

Dimensions

The "Dimensions" table displays information about the number of effects and the number of parameters from which the selected model is chosen. If you use split classification variables, then this table also includes the number of effects after splitting is taken into account.

Entry and Removal Candidates

When you specify the DETAILS=ALL or DETAILS=STEPS option in the SELECTION statement, the HPREG procedure produces "Entry Candidates" and "Removal Candidates" tables that display the effect names and values of the criterion used to select entering or departing effects at each step of the selection process. The effects are displayed in sorted order from best to worst of the selection criterion.

Selection Summary

When you specify the SELECTION statement, the HPREG procedure produces the "Selection Summary" table with information about the sequence of steps of the selection process. For each step, the effect that was entered or dropped is displayed along with the statistics used to select the effect, stop the selection, and choose the selected model. For all criteria that you can use for model selection, the steps at which the optimal values of these criteria occur are also indicated.

The display of the "Selection Summary" table can be suppressed by specifying DETAILS=NONE in the SELECTION statement.

Stop Reason

The "Stop Reason" table displays the reason why the selection stopped. To facilitate programmatic use of this table, an integer code is assigned to each reason and is included if you output this table by using an ODS OUTPUT statement. The reasons and their associated codes follow:

Code	Stop Reason
1	All eligible effects are in the model.
2	All eligible effects have been removed.
3	Specified maximum number of steps done.
4	The model contains the specified maximum number of effects.
5	The model contains the specified minimum number of effects (for backward selection).
6	The stopping criterion is at a local optimum.
7	No suitable add or drop candidate could be found.
8	Adding or dropping any effect does not improve the selection criterion.
9	No candidate meets the appropriate SLE or SLS significance level.
10	Stepwise selection is cycling.
11	The model is an exact fit.
12	Dropping an effect would result in an empty model.

The display of the "Stop Reason" table can be suppressed by specifying DETAILS=NONE in the SELECTION statement.

Selection Reason

When you specify the SELECTION statement, the HPREG procedure produces a simple table that contains text informing you about the reason why the final model was selected.

The display of the "Selection Reason" table can be suppressed by specifying DETAILS=NONE in the SELECTION statement.

Selected Effects

When you specify the SELECTION statement, the HPREG procedure produces a simple table that contains text informing you about which effects were selected into the final model.

ANOVA

The "ANOVA" table displays an analysis of variance for the selected model. This table includes the following:

- the Source of the variation, Model for the fitted regression, Error for the residual error, and C Total for the total variation after correcting for the mean. The Uncorrected Total Variation is produced when the NOINT option is used.

- the degrees of freedom (DF) associated with the source
- the Sum of Squares for the term
- the Mean Square, the sum of squares divided by the degrees of freedom
- the *F* Value for testing the hypothesis that all parameters are 0 except for the intercept. This is formed by dividing the mean square for Model by the mean square for Error.
- the Prob>*F*, the probability of getting a greater *F* statistic than that observed if the hypothesis is true. When you do model selection, these *p*-values are generally liberal because they are not adjusted for the fact that the terms in the model have been selected.

You can request "ANOVA" tables for the model at each step of the selection process with the DETAILS= option in the SELECTION statement.

Fit Statistics

The "Fit Statistics" table displays fit statistics for the selected model. The statistics displayed include the following:

- Root MSE, an estimate of the standard deviation of the error term. It is calculated as the square root of the mean square error.
- R-square, a measure between 0 and 1 that indicates the portion of the (corrected) total variation attributed to the fit rather than left to residual error. It is calculated as SS(Model) divided by SS(Total). It is also called the *coefficient of determination*. It is the square of the multiple correlation—in other words, the square of the correlation between the dependent variable and the predicted values.
- Adj R-Sq, the adjusted R-square, a version of R-square that has been adjusted for degrees of freedom. It is calculated as

$$\bar{R}^2 = 1 - \frac{(n-i)(1-R^2)}{n-p}$$

  where i is equal to 1 if there is an intercept and 0 otherwise, n is the number of observations used to fit the model, and p is the number of parameters in the model.
- fit criteria AIC, AICC, BIC, CP, and PRESS if they are used in the selection process. See Table 14.5 for the formulas for evaluating these criteria.
- the average square errors (ASE) on the training, validation, and test data.

You can request "Fit Statistics" tables for the model at each step of the selection process with the DETAILS= option in the SELECTION statement.

Parameter Estimates

The "Parameter Estimates" table displays the parameters in the selected model and their estimates. The information displayed for each parameter in the selected model includes the following:

- the parameter label that includes the effect name and level information for effects that contain classification variables
- the degrees of freedom (DF) for the parameter. There is one degree of freedom unless the model is not full rank.
- the parameter estimate
- the standard error, which is the estimate of the standard deviation of the parameter estimate
- t Value, the *t* test that the parameter is 0. This is computed as the parameter estimate divided by the standard error.
- the Pr > |t|, the probability that a *t* statistic would obtain a greater absolute value than that observed given that the true parameter is 0. This is the two-tailed significance probability.

  When you do model selection, these *p*-values are generally liberal because they are not adjusted for the fact that the terms in the model have been selected.

You can request "Parameter Estimates" tables for the model at each step of the selection process with the DETAILS= option in the SELECTION statement.

Timing Information

If you specify the DETAILS option in the PERFORMANCE statement, the procedure also produces a "Timing" table in which elapsed time (absolute and relative) for the main tasks of the procedure are displayed.

ODS Table Names

Each table created by the HPREG procedure has a name associated with it, and you must use this name to refer to the table when you use ODS statements. These names are listed in Table 14.7.

Table 14.7 ODS Tables Produced by PROC HPREG

Table Name	Description	Required Statement / Option
ANOVA	Selected model ANOVA table	Default output
Candidates	Swap candidates at step	SELECTION DETAILS=ALL|STEPS
ClassLevels	Level information from the CLASS statement	CLASS
DataAccessInfo	Information about modes of data access	Default output
Dimensions	Model dimensions	Default output

Table 14.7 *continued*

Table Name	Description	Required Statement / Option
EntryCandidates	Candidates for entry at step	SELECTION DETAILS=ALL|STEPS
FitStatistics	Fit statistics	Default output
ModelInfo	Information about the modeling environment	Default output
NObs	Number of observations read and used	Default output
ParameterEstimates	Solutions for the parameter estimates associated with effects in MODEL statement	Default output
PerformanceInfo	Information about high-performance computing environment	Default output
RemovalCandidates	Candidates for removal at step	SELECTION DETAILS=ALL|STEPS
ScreenedEffects	List of screened effects	SELECTION SCREEN
ScreenInfo	Information about the screening method	SELECTION SCREEN
Screening	Screening statistic values for model effects	SELECTION SCREEN(DETAILS=ALL)
SelectedEffects	List of selected effects	SELECTION
SelectionInfo	Information about selection settings	Default output
SelectionReason	Reason for selecting the final model	SELECTION
SelectionSummary	Summary information about the model selection steps	SELECTION
StopReason	Reason selection was terminated	SELECTION
Timing	Timing breakdown by task	SELECTION DETAILS

Examples: HPREG Procedure

Example 14.1: Model Selection with Validation

This example is based on the example "Using Validation and Cross Validation" in the documentation for the GLMSELECT procedure in the *SAS/STAT User's Guide*. This example shows how you can use validation data to monitor and control variable selection. It also demonstrates the use of split classification variables.

The following DATA step produces analysis data that contains a variable that you can use to assign observations to the training, validation, and testing roles. In this case, each role has 5,000 observations.

```
data analysisData;
   drop i j c3Num;
   length c3$ 7;

   array x{20} x1-x20;

   do i=1 to 15000;
      do j=1 to 20;
         x{j} = ranuni(1);
      end;

      c1 = 1 + mod(i,8);
      c2 = ranbin(1,3,.6);

      if      i < 50   then do; c3 = 'tiny';     c3Num=1;end;
      else if i < 250  then do; c3 = 'small';    c3Num=1;end;
      else if i < 600  then do; c3 = 'average';  c3Num=2;end;
      else if i < 1200 then do; c3 = 'big';      c3Num=3;end;
      else                  do; c3 = 'huge';     c3Num=5;end;

      yTrue = 10 + x1 + 2*x5 + 3*x10 + 4*x20  + 3*x1*x7 + 8*x6*x7
                 + 5*(c1=3)*c3Num + 8*(c1=7);

      error = 5*rannor(1);

      y = yTrue + error;

           if mod(i,3)=1 then Role = 'TRAIN';
      else if mod(i,3)=2 then Role = 'VAL';
      else                    Role = 'TEST';

      output;
  end;
run;
```

By construction, the true model consists of main effects x1, x5, x10, x20, and c1 and interaction effects x1*x7, x6*x7, and c1*c3. Furthermore, you can see that only levels 3 and 7 of the classification variable c1 are systematically related to the response.

Because the error term for each observation is five times a value drawn from a standard normal distribution, the expected error variance is 25. For the data in each role, you can compute an estimate of this error variance by forming the average square error (ASE) for the observations in the role. Output 14.1.1 shows the ASE for each role that you can compute with the following statements:

```
proc summary data=analysisData;
   class role;
   ways 1;
   var error;
   output out=ASE uss=uss n=n;
data ASE; set ASE;
   OracleASE = uss / n;
   label OracleASE = 'Oracle ASE';
   keep Role OracleASE;
proc print data=ASE label noobs;
run;


proc print data=ASE label noobs;
run;
```

Output 14.1.1 Oracle ASE Values by Role

Role	Oracle ASE
TEST	25.5784
TRAIN	25.4008
VAL	25.8993

The ASE values shown Output 14.1.1 are labeled as "Oracle ASE" because you need to know the true underlying model if you want to compute these values from the response and underlying regressors. In a modeling context, a good predictive model produces values that are close to these oracle values. An overfit model produces a smaller ASE on the training data but higher values on the validation and test data. An underfit model exhibits higher values for all data roles.

Suppose you suspect that the dependent variable depends on both main effects and two-way interactions. You can use the following statements to select a model:

```
proc hpreg data=analysisData;
   partition roleVar=role(train='TRAIN' validate='VAL' test='TEST');
   class c1 c2 c3(order=data);
   model y =  c1|c2|c3|x1|x2|x3|x4|x5|x5|x6|x7|x8|x9|x10
             |x11|x12|x13|x14|x15|x16|x17|x18|x19|x20 @2 /stb;
   selection method = stepwise(select=sl sle=0.1 sls=0.15 choose=validate)
                      hierarchy=single details=steps;
run;
```

A PARTITION statement assigns observations to training, validation, and testing roles based on the values of the input variable named role. The SELECTION statement requests STEPWISE selection based on significance level where the SLE and SLS values are set to use the defaults of PROC REG. The CHOOSE=VALIDATE option selects the model that yields the smallest ASE value on the validation data.

The "Number Of Observation" table in Output 14.1.2 confirms that there are 5,000 observations for each data role. The "Dimensions" table shows that the selection is from 278 effects with a total of 661 parameters.

Output 14.1.2 Number of Observations, Class Levels, and Dimensions

The HPREG Procedure

Number of Observations Read	15000
Number of Observations Used	15000
Number of Observations Used for Training	5000
Number of Observations Used for Validation	5000
Number of Observations Used for Testing	5000

Class Level Information

Class	Levels	Values
c1	8	1 2 3 4 5 6 7 8
c2	4	0 1 2 3
c3	5	tiny small average big huge

Dimensions	
Number of Effects	278
Number of Parameters	661

Output 14.1.3 shows the "Selection Summary" table. You see that 18 steps are done, at which point all effects in the model are significant at the SLS value of 0.15 and all the remaining effects if added individually would not be significant at the SLE significance level of 0.1. However, because you have specified the CHOOSE=VALIDATE option, the model at step 18 is not used as the selected model. Instead the model at step 10 (where the validation ASE achieves a local minimum value) is selected. The "Stop Reason," "Selection Reason," and "Selected Effects" in Output 14.1.4 provide this information.

Output 14.1.3 Selection Summary

The HPREG Procedure

Selection Summary

Step	Effect Entered	Number Effects In	Validation ASE	p Value
0	**Intercept**	1	98.3895	1.0000
1	**c1**	2	34.8572	<.0001
2	**x7**	3	32.5531	<.0001
3	**x6**	4	31.0646	<.0001
4	**x20**	5	29.7078	<.0001
5	**x6*x7**	6	29.2210	<.0001
6	**x10**	7	28.6683	<.0001
7	**x1**	8	28.3250	<.0001
8	**x5**	9	27.9766	<.0001
9	**c3**	10	27.8288	<.0001
10	**c1*c3**	11	25.9701*	<.0001
11	**x10*c1**	12	26.0696	0.0109
12	**x4**	13	26.1594	0.0128
13	**x4*x10**	14	26.1814	0.0035
14	**x20*c1**	15	26.3294	0.0156
15	**x1*c3**	16	26.3945	0.0244
16	**x1*x7**	17	26.3632	0.0270
17	**x7*x10**	18	26.4120	0.0313
18	**x1*x20**	19	26.4330	0.0871

*** Optimal Value of Criterion**

Output 14.1.4 Stopping and Selection Reasons

Selection stopped because all candidates for removal are significant at the 0.15 level and no candidate for entry is significant at the 0.1 level.

The model at step 10 is selected where Validation ASE is 25.9701.

Selected Effects: Intercept c1 c3 c1*c3 x1 x5 x6 x7 x6*x7 x10 x20

You can see that the selected effects include all the main effects in the true model and two of the three true interaction terms. Furthermore, the selected model does not include any variables that are not in the true model. Note that these statements are not true of the larger model at the final step of the selection process.

Output 14.1.5 shows the fit statistics of the selected model. You can see that the ASE values on the training, validation, and test data are all similar, which is indicative of a reasonable predictive model. In this case where the true model is known, you can see that all three ASE values are close to oracle values for the true model, as shown in Output 14.1.1.

Output 14.1.5 Fit Statistics for the Selected Model

Root MSE	5.03976
R-Square	0.74483
Adj R-Sq	0.74246
AIC	21222
AICC	21223
SBC	16527
ASE (Train)	25.16041
ASE (Validate)	25.97010
ASE (Test)	25.83436

Because you specified the DETAILS=STEPS option in the SELECTION statement, you can see the "Fit Statistics" for the model at each step of the selection process. Output 14.1.6 shows these fit statistics for final model at step 18. You see that for this model, the ASE value on the training data is smaller than the ASE values on the validation and test data. This is indicative an overfit model that might not generalize well to new data. You see the ASE values on the validation and test data are now worse in comparison to the oracle values than the values for the selected model at step 10.

Output 14.1.6 Fit Statistics for the Model at Step 18

Root MSE	5.01386
R-Square	0.74862
Adj R-Sq	0.74510
AIC	21194
AICC	21196
SBC	16648
ASE (Train)	24.78688
ASE (Validate)	26.43304
ASE (Test)	26.07078

Output 14.1.7 shows part of the "Parameter Estimates" table for the selected model at step 10 that includes the estimates for the main effect `c1`. Because the STB option is specified in the MODEL statement, this table includes standardized estimates.

Output 14.1.7 Part of the Parameter Estimates Table for the Selected Model

Parameter Estimates						
Parameter	DF	Estimate	Standardized Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	9.479114	0	0.422843	22.42	<.0001
c1 1	1	0.279417	0.009306	0.297405	0.94	0.3475
c1 2	1	0.615589	0.020502	0.297332	2.07	0.0385
c1 3	1	25.678601	0.855233	0.297280	86.38	<.0001
c1 4	1	0.420360	0.014000	0.297283	1.41	0.1574
c1 5	1	0.473986	0.015786	0.297265	1.59	0.1109
c1 6	1	0.394044	0.013124	0.297299	1.33	0.1851
c1 7	1	8.469793	0.282089	0.297345	28.48	<.0001
c1 8	0	0	0	.	.	.

The magnitudes of the standardized estimates and the *t* statistics of the parameters of the effect c1 reveal that only levels 3 and 7 of this effect contribute appreciably to the model. This suggests that a more parsimonious model with similar or better predictive power might be obtained if parameters that correspond to the levels of c1 can enter or leave the model independently. You request this with the SPLIT option in the CLASS statement as shown in the following statements:

```
proc hpreg data=analysisData;
   partition roleVar=role(train='TRAIN' validate='VAL' test='TEST');
   class c1(split) c2 c3(order=data);
   model y =  c1|c2|c3|x1|x2|x3|x4|x5|x5|x6|x7|x8|x9|x10
             |x11|x12|x13|x14|x15|x16|x17|x18|x19|x20 @2 /stb;
   selection method = stepwise(select=sl sle=0.1 sls=0.15 choose=validate)
                      hierarchy=single details=steps;
run;
```

Output 14.1.8 shows the "Dimensions" table. You can see that because the columns in the design matrix that correspond to levels of c1 are treated as separate effects, the selection is now from 439 effects, even though the number of parameters is unchanged.

Output 14.1.8 Dimensions with c1 Split

The HPREG Procedure

Dimensions	
Number of Effects	278
Number of Effects after Splits	439
Number of Parameters	661

Output 14.1.9 shows the selected effects. You can see that as anticipated the selected model now depends on only levels 3 and 7 of c1.

Output 14.1.9 Selected Effects with c1 Split

Selected Effects:	Intercept c1_3 c1_7 c3 c1_3*c3 x1 x5 x6 x7 x6*x7 x10 x20

Finally, the fit statistics for the selected model are shown Output 14.1.10.

Output 14.1.10 Fit Statistics for the Selected Model with c1 Split

Root MSE	5.04060
R-Square	0.74325
Adj R-Sq	0.74238
AIC	21195
AICC	21195
SBC	16311
ASE (Train)	25.31622
ASE (Validate)	25.98055
ASE (Test)	25.76059

If you compare the ASE values for this model in Output 14.1.10 with the oracle values in Output 14.1.1 and the values for the model without splitting c1 in Output 14.1.5, you see that this more parsimonious model produces the best predictive performance on the test data of all the models considered in this example.

Example 14.2: Backward Selection in Single-Machine and Distributed Modes

This example shows how you can run PROC HPREG in single-machine and distributed modes. See the section "Processing Modes" on page 10 in Chapter 3, "Shared Concepts and Topics," for details about the execution modes of SAS High-Performance Statistics procedures. The focus of this example is to simply show how you can switch the modes of execution of PROC HPREG, rather than on any statistical features of the procedure. The following DATA step generates the data for this example. The response y depends on 20 of the 1,000 regressors.

```
data ex2Data;
   array x{1000};

   do i=1 to 10000;
      y=1;
      sign=1;

      do j=1 to 1000;
         x{j} = ranuni(1);
         if j<=20  then do;
           y = y + sign*j*x{j};
           sign=-sign;
         end;
      end;
      y = y + 5*rannor(1);
      output;
  end;
run;
```

The following statements use PROC HPREG to select a model by using BACKWARD selection:

```
proc hpreg data=ex2Data;
    model y = x: ;
    selection method = backward;
    performance details;
run;
```

Output 14.2.1 shows the "Performance Information" table. This shows that the HPREG procedure executes in single-machine mode using four threads because the client machine has four CPUs. You can force a certain number of threads on any machine involved in the computations with the NTHREADS option in the PERFORMANCE statement.

Output 14.2.1 Performance Information

The HPREG Procedure

Performance Information	
Execution Mode	Single-Machine
Number of Threads	4

Output 14.2.2 shows the parameter estimates for the selected model. You can see that the default BACKWARD selection with selection and stopping based on the SBC criterion retains all 20 of the true effects but also keeps two extraneous effects.

Output 14.2.2 Parameter Estimates for the Selected Model

Parameter Estimates					
Parameter	DF	Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	1.506615	0.419811	3.59	0.0003
x1	1	1.054402	0.176930	5.96	<.0001
x2	1	-1.996080	0.176967	-11.28	<.0001
x3	1	3.293331	0.177032	18.60	<.0001
x4	1	-3.741273	0.176349	-21.22	<.0001
x5	1	4.908310	0.176047	27.88	<.0001
x6	1	-5.772356	0.176642	-32.68	<.0001
x7	1	7.398822	0.175792	42.09	<.0001
x8	1	-7.958471	0.176281	-45.15	<.0001
x9	1	8.899407	0.177624	50.10	<.0001
x10	1	-9.687667	0.176431	-54.91	<.0001
x11	1	11.083373	0.175195	63.26	<.0001
x12	1	-12.046504	0.176324	-68.32	<.0001
x13	1	13.009052	0.176967	73.51	<.0001
x14	1	-14.456393	0.175968	-82.15	<.0001
x15	1	14.928731	0.174868	85.37	<.0001
x16	1	-15.762907	0.177651	-88.73	<.0001
x17	1	16.842889	0.177037	95.14	<.0001
x18	1	-18.468844	0.176502	-104.64	<.0001
x19	1	18.810193	0.176616	106.50	<.0001
x20	1	-20.212291	0.176325	-114.63	<.0001
x87	1	-0.542384	0.176293	-3.08	0.0021
x362	1	-0.560999	0.176594	-3.18	0.0015

Output 14.2.3 shows timing information for the PROC HPREG run. This table is produced when you specify the DETAILS option in the PERFORMANCE statement. You can see that, in this case, the majority of time is spent forming the crossproducts matrix for the model that contains all the regressors.

Output 14.2.3 Timing

Procedure Task Timing		
Task	Seconds	Percent
Reading and Levelizing Data	0.20	6.72%
Loading Design Matrix	0.03	1.06%
Computing Moments	0.02	0.82%
Computing Cross Products Matrix	2.14	72.79%
Performing Model Selection	0.55	18.62%

You can switch to running PROC HPREG in distributed mode by specifying valid values for the NODES=, INSTALL=, and HOST= options in the PERFORMANCE statement. An alternative to specifying the INSTALL= and HOST= options in the PERFORMANCE statement is to set appropriate values for the GRIDHOST and GRIDINSTALLLOC environment variables by using OPTIONS SET commands. See the section "Processing Modes" on page 10 in Chapter 3, "Shared Concepts and Topics," for details about setting these options or environment variables.

The following statements provide an example. To run these statements successfully, you need to set the macro variables GRIDHOST and GRIDINSTALLLOC to resolve to appropriate values, or you can replace the references to macro variables with appropriate values.

```
proc hpreg data=ex2Data;
    model y = x: ;
    selection method = backward;
    performance details nodes = 10
                host="&GRIDHOST" install="&GRIDINSTALLLOC";
run;
```

The execution mode in the "Performance Information" table shown in Output 14.2.4 indicates that the calculations were performed in a distributed environment that uses 10 nodes, each of which uses eight threads.

Output 14.2.4 Performance Information in Distributed Mode

Performance Information	
Host Node	<< your grid host >>
Install Location	<< your grid install location >>
Execution Mode	Distributed
Number of Compute Nodes	10
Number of Threads per Node	32

Another indication of distributed execution is the following message issued by all high-performance statistical procedures in the SAS Log:

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

Output 14.2.5 shows timing information for this distributed run of the HPREG procedure. In contrast to the single-machine mode (where forming the crossproducts matrix dominated the time spent), the majority of time in distributed mode is spent distributing the data and performing the model selection.

Output 14.2.5 Timing

Procedure Task Timing		
Task	**Seconds**	**Percent**
Distributing Data	0.73	46.40%
Reading and Levelizing Data	0.02	1.07%
Loading Design Matrix	0.00	0.29%
Computing Moments	0.00	0.13%
Computing Cross Products Matrix	0.18	11.18%
Performing Model Selection	0.32	20.37%
Waiting on Client	0.32	20.55%

Example 14.3: Forward-Swap Selection

This example highlights the use of the forward-swap selection method, which is a generalization of the maximum R-square improvement (MAXR) method that is available in the REG procedure in SAS/STAT software. This example also demonstrates the use of the INCLUDE and START options.

The following DATA step produces the simulated data in which the response y depends on six main effects and three 2-way interactions from a set of 20 regressors.

```
data ex3Data;
   array x{20};
   do i=1 to 10000;
      do j=1 to 20;
         x{j} = ranuni(1);
      end;
      y = 3*x1 + 7*x2 -5*x3 + 5*x1*x3 +
          4*x2*x13 + x7 + x11 -x13  + x1*x4 + rannor(1);
      output;
   end;
run;
```

Suppose you want to find the best model of each size in a range of sizes for predicting the response y. You can use the forward-swap selection method to produce good models of each size without the computational expense of examining all possible models of each size. In this example, the criterion used to evaluate the models of each size is the model R square. With this criterion, the forward-swap method coincides with the MAXR method that is available in the REG procedure in SAS/STAT software. The model of a given size for which no pairwise swap of an effect in the model with any candidate effect improves the R-square value is deemed to be the best model of that size.

Suppose that you have prior knowledge that the regressors x1, x2, and x3 are needed in modeling the response y. Suppose that you also believe that some of the two-way interactions of these variables are likely to be important in predicting y and that some other two-way interactions might also be needed. You can use this prior information by specifying the selection process shown in the following statements:

```
proc hpreg data=ex3Data;
    model y = x1|x2|x3|x4|x5|x6|x7|x8|x9|x10|X11|
              x12|x13|x14|x5|x16|x7|x18|x19|x20@2 /
                  include=(x1 x2 x3) start=(x1*x2 x1*x3 x2*x3);
    selection method=forwardswap(select=rsquare maxef=15 choose=sbc) details=all;
run;
```

The MODEL statement specifies that all main effects and two-way interactions are candidates for selection. The INCLUDE= option specifies that the effects x1, x2, and x3 must appear in all models that are examined. The START= option specifies that all the two-way interactions of these variables should be used in the initial model that is considered but that these interactions are eligible for removal during the forward-swap selection.

The "Selection Summary" table is shown in Output 14.3.1.

Output 14.3.1 Selection Summary

The HPREG Procedure

Selection Summary

Step	Effect Entered	Effect Removed	Number Effects In	SBC	Model R-Square
0	Intercept		1		
	x1		2		
	x2		3		
	x1*x2		4		
	x3		5		
	x1*x3		6		
	x2*x3		7	3307.6836	0.8837
1	x2*x13		8	1892.8403	0.8992
2	x7*x11	x1*x2	8	618.9298	0.9112
3	x1*x4	x2*x3	8	405.3751	0.9131
4	x13		9	213.6140	0.9148
5	x7		10	180.4457	0.9152
6	x11	x7*x11	10	1.4039*	0.9167
7	x10*x11		11	2.3393	0.9168
8	x3*x7		12	4.5000	0.9168
9	x6*x7		13	10.0589	0.9169
10	x3*x6		14	13.1113	0.9169
11	x5*x20		15	19.4612	0.9169
12	x13*x20	x3*x6	15	18.3678	0.9169
13	x5*x5	x6*x7	15	12.1398	0.9170*

*** Optimal Value of Criterion**

You see that starting from the model with an intercept and the effects specified in the INCLUDE= and START= options at step 0, the forward-swap selection method adds the effect x2*x13 at step one, because this yields the maximum improvement in R square that can be obtained by adding a single effect. The forward-swap selection method now evaluates whether any effect swap yields a better eight-effect model (one with a higher R-square value). Because you specified the DETAILS=ALL option in the SELECTION statement, at each step where a swap is made you obtain a "Candidates" table that shows the R-square values for the evaluated swaps. Output 14.3.2 shows the "Candidates" for step 2. By default, only the best 10 swaps are displayed.

Output 14.3.2 Swap Candidates at Step 2

Best 10 Candidates			
Rank	Effect Dropped	Effect Added	R-Square
1	x1*x2	x7*x11	0.9112
2	x2*x3	x7*x11	0.9112
3	x1*x2	x7	0.9065
4	x2*x3	x7	0.9065
5	x1*x2	x7*x7	0.9060
6	x2*x3	x7*x7	0.9060
7	x1*x2	x4*x7	0.9060
8	x2*x3	x4*x7	0.9060
9	x1*x2	x11	0.9058
10	x2*x3	x11	0.9058

You see that the best swap adds x7*x11 and drops x1*x2. This yields an eight-effect model whose R-square value (0.9112) is larger than the R-square value (0.8992) of the eight-effect model at step 1. Hence this swap is made at step 2. At step 3, an even better eight-effect model than the model at step 2 is obtained by dropping x2*x3 and adding x1*x4. No additional swap improves the R-square value, and so the model at step 3 is deemed to be the best eight-effect model. Although this is the best eight-effect model that can be found by this method given the starting model, it is not guaranteed that this model that has the highest R-square value among all possible models that consist of seven effects and an intercept.

Because the DETAILS=ALL option is specified in the SELECTION statement, details for the model at each step of the selection process are displayed. Output 14.3.3 provides details of the model at step 3.

Output 14.3.3 Model Details at Step 3

Analysis of Variance					
Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	7	108630	15519	15000.3	<.0001
Error	9992	10337	1.03455		
Corrected Total	9999	118967			

Root MSE	1.01713
R-Square	0.91311
Adj R-Sq	0.91305
AIC	10350
AICC	10350
SBC	405.37511
ASE	1.03373

Output 14.3.3 *continued*

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	0.012095	0.045712	0.26	0.7913
x1	1	3.087078	0.076390	40.41	<.0001
x2	1	7.775180	0.046815	166.08	<.0001
x3	1	-4.957140	0.070995	-69.82	<.0001
x1*x3	1	4.910115	0.122503	40.08	<.0001
x1*x4	1	0.890436	0.060523	14.71	<.0001
x7*x11	1	1.708469	0.045939	37.19	<.0001
x2*x13	1	2.584078	0.061506	42.01	<.0001

The forward-swap method continues to find the best nine-effect model, best 10-effect model, and so on until it obtains the best 15-effect model. At this point the selection terminates because you specified the MAXEF=15 option in the SELECTION statement. The R-square value increases at each step of the selection process. However, because you specified the CHOOSE=SBC criterion in the SELECTION statement, the final model selected is the model at step 6.

Example 14.4: Forward Selection with Screening

This example shows how you can use the SCREEN option in the SELECTION statement to greatly speed up model selection from a large number of regressors. In order to demonstrate the efficacy of model selection with screening, this example uses simulated data in which the response y depends systematically on a relatively small subset of a much larger set of regressors, which is described in Table 14.8.

Table 14.8 Complete Set of Regressors

Regressor Name	Type	Number of Levels	In True Model
xIn1–xIn25	Continuous		Yes
xWeakIn1–xWeakIn2	Continuous		Yes
xOut1–xOut500	Continuous		No
cIn1–cIn5	Classification	From two to five	Yes
cOut1–cOut500	Classification	From two to five	No

The labels In and Out, which are part of the variable names, make it easy to identify whether the selected model succeeds or fails in capturing the true underlying model. The regressors that are labeled xWeakIn1 and xWeakIn2 are predictive, but their influence is substantially smaller than the influence of the other regressors in the true model.

The following DATA step generates the data:

```
%let nObs      = 50000;
%let nContIn   = 25;
%let nContOut  = 500;
%let nClassIn  = 5;
%let nClassOut = 500;
```

```
%let maxLevs   = 5;
%let noiseScale= 1;

 data ex4Data;
   array xIn{&nContIn};
   array xOut{&nContOut};
   array cIn{&nClassIn};
   array cOut{&nClassOut};

   drop i j sign nLevs xBeta;

   do i=1 to &nObs;
      sign  = -1;
      xBeta = 0;
      do j=1 to dim(xIn);
         xIn{j} = ranuni(1);
         xBeta  = xBeta + j*sign*xIn{j};
         sign   = -sign;
      end;
      do j=1 to dim(xOut);
         xOut{j} = ranuni(1);
      end;

      xWeakIn1 = ranuni(1);
      xWeakin2 = ranuni(1);

      xBeta  = xBeta + 0.1*xWeakIn1+ 0.1*xWeakIn2;

      do j=1 to dim(cIn);
         nLevs  = 2 + mod(j,&maxlevs-1);
         cIn{j} = 1+int(ranuni(1)*nLevs);
         xBeta  = xBeta + j*sign*(cIn{j}-nLevs/2);
         sign   = -sign;
      end;

      do j=1 to dim(cOut);
         nLevs  = 2 + mod(j,&maxlevs-1);
         cOut{j} = 1+int(ranuni(1)*nLevs);
      end;

      y = xBeta + &noiseScale*rannor(1);

      output;
   end;
run;
```

When you have insufficient prior knowledge of what effects need to be included in a parsimonious predictive model, a reasonable starting point is to use model selection to build a such a model. In such cases, you might want to consider a large number of possible model effects, even though you know that a successful model that generalizes well for predicting unseen data depends on a relatively small number of effects. In such cases, you can dramatically reduce the computational task by including screening in the model selection process. The following statements show how you do this:

```
proc hpreg data=ex4Data;
    class c: ;
    model y = x: c: ;
    selection method=forward screen(details=all)=100 20;
    performance details;
run;
```

The ordered pair of integers that is specified in the SCREEN option in the SELECTION statement requests that screening be used to reduce the set of regressors to 100 regressors at the first screening stage and to 20 regressors at the second screening stage. This information is reflected in the "Screening Information" table shown in Output 14.4.1.

Output 14.4.1 Screening Information

The HPREG Procedure

Screening Information	
Screening Stages	Multiple
Screening Criterion	Maximum Absolute Correlation
Stage 1 Number of Screened Effects	100
Stage 2 Number of Screened Effects	20

The "Number Of Observations" table in Output 14.4.2 confirms that the data contain 50,000 observations and the "Dimensions" table shows that the selection is from 1,033 effects that have a total of 2,295 parameters.

Output 14.4.2 Number of Observations and Dimensions

Number of Observations Read	50000
Number of Observations Used	50000

Dimensions	
Number of Effects	1033
Number of Parameters	2295

Because you specified the DETAILS=ALL suboption of the SCREEN option, you obtain the "Screening" table in Output 14.4.3, which shows how the screened subset of 100 effects is obtained at the first screening stage. For display purposes, some ranks in this table have been suppressed.

Output 14.4.3 First Stage Screening Details

Effect Screening for Response		
Rank	Effect	Maximum Absolute Correlation
1	xIn25	0.31785
2	xIn24	0.30697
3	xIn23	0.29734
.	.	.
.	.	.
98	xOut338	0.00932
99	cOut363	0.00922
100	cOut194	0.00920
101	xOut125	0.00919*
102	xOut220	0.00916*
103	cOut310	0.00916*
104	cOut49	0.00915*
105	cOut11	0.00915*

The "Screened Effects" table shown in Output 14.4.4 lists the effects from which a model is selected at the first screening stage.

Output 14.4.4 First Stage Screened Effects

Screened Effects:	xIn25 xIn24 xIn23 xIn22 xIn21 xIn20 xIn19 xIn18 xIn17 xIn16 xIn15 xIn14 xIn13 cIn5 xIn12 cIn3 xIn11 xIn10 xIn8 xIn9 xIn7 cIn4 cIn2 xIn6 xIn5 xIn4 xIn3 cIn1 xIn2 cOut498 cOut110 cOut450 cOut441 cOut272 xOut82 cOut45 cOut6 cOut281 cOut134 cOut15 xOut310 xOut252 xOut485 xOut365 cOut138 cOut123 cOut337 cOut195 cOut423 cOut283 cOut62 cOut114 xOut489 cOut14 cOut158 cOut437 xOut64 cOut301 cOut311 cOut187 cOut431 cOut464 cOut388 cOut213 cOut46 xOut329 cOut403 cOut305 cOut171 cOut85 cOut99 cOut249 xOut267 cOut455 cOut457 cOut271 cOut78 xOut93 cOut259 cOut417 cOut258 cOut326 cOut291 cOut263 cOut107 cOut402 cOut17 cOut237 cOut129 cOut198 cOut58 cOut428 cOut135 cOut206 cOut139 cOut113 cOut486 xOut338 cOut363 cOut194

You see that the magnitude of the pairwise correlations of effects xIn1, xWeakIn1, and xWeakIn2 with response are too small for those effects to be included as candidates for selection at the first screening stage.

The first stage continues with forward selection from the screened effects that are shown in Output 14.4.4. The effects in the selected model at this stage are shown in Output 14.4.5.

Output 14.4.5 First Stage Selected Effects

Selected Effects:	Intercept xIn2 xIn3 xIn4 xIn5 xIn6 xIn7 xIn8 xIn9 xIn10 xIn11 xIn12 xIn13 xIn14 xIn15 xIn16 xIn17 xIn18 xIn19 xIn20 xIn21 xIn22 xIn23 xIn24 xIn25 cIn1 cIn2 cIn3 cIn4 cIn5

You see that the selected model at this stage includes only effects that are systematically related to the response. If you had requested that only a single-stage screening method be used by specifying the SINGLESTAGE suboption of the SCREEN option, then the selected model at this stage would have been the final selected model. However, multistage screening is used in this example. The second stage repeats the steps of the first stage except that the modeled response is the residuals from the selected model at the first stage.

Output 14.4.6 shows the screening details at the second stage. You see that 20 effects are chosen by screening at this stage as specified. Because the selected effects from the first stage are orthogonal to the residuals at

the first stage, none of these effects are in the screened subset. Furthermore, you see that although the effects xIn1, xWeakIn1, and xWeakIn2 are weakly correlated with y, they are the most strongly correlated effects with the residuals from the first stage.

Output 14.4.6 Second Stage Screening Details

Screening Stage 2: Residual Fit

Effect Screening for Stage 1 Residuals

Rank	Effect	Maximum Absolute Correlation
1	xIn1	0.27373
2	xWeakIn1	0.02352
3	xWeakin2	0.02132
4	cOut295	0.01524
5	cOut35	0.01443
6	cOut323	0.01417
7	cOut202	0.01406
8	xOut6	0.01401
9	cOut154	0.01263
10	cOut54	0.01160
11	cOut181	0.01159
12	cOut115	0.01150
13	cOut403	0.01144
14	xOut332	0.01142
15	xOut409	0.01141
16	cOut267	0.01137
17	cOut374	0.01132
18	cOut254	0.01128
19	xOut204	0.01121
20	cOut147	0.01120
21	xOut113	0.01116*
22	xOut427	0.01115*
23	cOut259	0.01111*
24	cOut170	0.01106*
25	cOut107	0.01102*

*** Screened Out**

Screened Effects:	xIn1 xWeakIn1 xWeakin2 cOut295 cOut35 cOut323 cOut202 xOut6 cOut154 cOut54 cOut181 cOut115 cOut403 xOut332 xOut409 cOut267 cOut374 cOut254 xOut204 cOut147

Output 14.4.7 shows the selected effects at the second screening stage. You see that the selected effects are precisely the remaining effects that are systematically predictive of y but that were not in the screened subset at the first screening stage.

Output 14.4.7 Second Stage Selected Effects

Selected Effects:	Intercept xIn1 xOut6 xWeakIn1 xWeakin2

In the third and final screening stage, model selection is performed from the union of the screened effects from the first stage (which are shown in Output 14.4.4) and the selected effects from the second stage (which are shown in Output 14.4.7). The selected effects from this final stage are shown in Output 14.4.8.

Output 14.4.8 Final Stage Selected Effects

Selected Effects:	Intercept xIn1 xIn2 xIn3 xIn4 xIn5 xIn6 xIn7 xIn8 xIn9 xIn10 xIn11 xIn12 xIn13 xIn14 xIn15 xIn16 xIn17 xIn18 xIn19 xIn20 xIn21 xIn22 xIn23 xIn24 xIn25 xOut6 xWeakIn1 xWeakin2 cIn1 cIn2 cIn3 cIn4 cIn5

You see that the final selected model contains all the true underlying model effects and just one noise effect (xOut6). Because you specified the DETAILS option in the PERFORMANCE statement, the "Timing" table shown in Output 14.4.9 is displayed.

Output 14.4.9 Timing for Model Selection with Screening

Procedure Task Timing		
Task	**Seconds**	**Percent**
Reading and Levelizing Data	2.23	26.76%
Loading Design Matrix	0.45	5.41%
Computing Moments	0.72	8.60%
Computing Cross Products Matrix	2.00	24.00%
Performing Model Selection	2.94	35.23%

You see that even though the selected model was obtained by selecting from thousands of effects, screening enabled the entire modeling task to be completed in about 10 seconds. You can perform the same model selection without screening as shown in the following statements:

```
proc hpreg data=ex4Data;
    class c: ;
    model y = x: c: ;
    selection method=forward;
    performance details;
run;
```

In this case, the model that is selected without screening is identical to model that is obtained with screening. However, there is no guarantee that you will get identical selected models. Output 14.4.10 shows the "Timing" table for the model selection without screening.

Output 14.4.10 Timing for Model Selection without Screening

Procedure Task Timing		
Task	**Seconds**	**Percent**
Reading and Levelizing Data	2.28	2.76%
Loading Design Matrix	0.32	0.39%
Computing Moments	0.27	0.33%
Computing Cross Products Matrix	32.70	39.62%
Performing Model Selection	46.96	56.91%

You see that the model selection without screening took about 83 seconds, which is substantially slower than the approximately 10 seconds it took when screening was included in the selection process.

References

Akaike, H. (1969), "Fitting Autoregressive Models for Prediction," *Annals of the Institute of Statistical Mathematics*, 21, 243–247.

Burnham, K. P. and Anderson, D. R. (2002), *Model Selection and Multimodel Inference*, 2nd Edition, New York: Springer-Verlag.

Collier Books (1987), *The 1987 Baseball Encyclopedia Update*, New York: Macmillan.

Darlington, R. B. (1968), "Multiple Regression in Psychological Research and Practice," *Psychological Bulletin*, 69, 161–182.

Draper, N. R., Guttman, I., and Kanemasu, H. (1971), "The Distribution of Certain Regression Statistics," *Biometrika*, 58, 295–298.

El Ghaoui, L., Viallon, V., and Rabbani, T. (2012), "Safe Feature Elimination for the Lasso and Sparse Supervised Learning Problems," *Pacific Journal of Optimization*, 8, 667–698, special Issue on Conic Optimization.

Fan, J. and Lv, J. (2008), "Sure Independence Screening for Ultrahigh Dimensional Feature Space," *Journal of the Royal Statistical Society, Series B*, 70, 849–911.

Foster, D. P. and Stine, R. A. (2004), "Variable Selection in Data Mining: Building a Predictive Model for Bankruptcy," *Journal of the American Statistical Association*, 303–313.

Harrell, F. E. (2001), *Regression Modeling Strategies*, New York: Springer-Verlag.

Hastie, T. J., Tibshirani, R. J., and Friedman, J. H. (2001), *The Elements of Statistical Learning*, New York: Springer-Verlag.

Hocking, R. R. (1976), "The Analysis and Selection of Variables in a Linear Regression," *Biometrics*, 32, 1–50.

Hurvich, C. M. and Tsai, C.-L. (1989), "Regression and Time Series Model Selection in Small Samples," *Biometrika*, 76, 297–307.

Judge, G. G., Griffiths, W. E., Hill, R. C., Lütkepohl, H., and Lee, T.-C. (1985), *The Theory and Practice of Econometrics*, 2nd Edition, New York: John Wiley & Sons.

Mallows, C. L. (1973), "Some Comments on C_p," *Technometrics*, 15, 661–675.

Schwarz, G. (1978), "Estimating the Dimension of a Model," *Annals of Statistics*, 6, 461–464.

Tibshirani, R., Bien, J., Friedman, J., Hastie, T., Simon, N., Taylor, J., and Tibshirani, R. J. (2012), "Strong Rules for Discarding Predictors in Lasso-Type Problems," *Journal of the Royal Statistical Society, Series B*, 74, 245–266.

Time Inc. (1987), "What They Make," *Sports Illustrated*, April, 54–81.

Chapter 15
The HPSPLIT Procedure

Contents

Overview: HPSPLIT Procedure **594**
PROC HPSPLIT Features 594
Getting Started: HPSPLIT Procedure **595**
Syntax: HPSPLIT Procedure **598**
PROC HPSPLIT Statement 598
CODE Statement 601
CRITERION Statement 601
ID Statement 603
INPUT Statement 603
OUTPUT Statement 604
PARTITION Statement 604
PERFORMANCE Statement 605
PRUNE Statement 606
RULES Statement 608
SCORE Statement 608
TARGET Statement 609
Details: HPSPLIT Procedure **609**
Building a Tree 609
Interval Input Binning Details 610
Input Variable Splitting and Selection 610
Pruning 612
Memory Considerations 614
Handling Missing Values 614
Handling Unknown Levels in Scoring 615
Splitting Criteria 615
Pruning Criteria 617
Subtree Statistics 618
Variable Importance 625
Outputs 626
Examples: HPSPLIT Procedure **628**
Example 15.1: Creating a Node Rules Description of a Tree 628
Example 15.2: Assessing Variable Importance 630
References **632**

Overview: HPSPLIT Procedure

The HPSPLIT procedure is a high-performance utility procedure that creates a decision or regression tree model and saves results in output data sets and files for use in SAS Enterprise Miner. Decision trees model a target which has a discrete set of levels by recursively partitioning the input variable space. Regression trees model a target which is continuous and also recursively partition the input space.

PROC HPSPLIT runs in either single-machine mode or distributed mode.

NOTE: Distributed mode requires SAS High-Performance Statistics.

PROC HPSPLIT Features

The main features of the HPSPLIT procedure are as follows:

- Model creation
  - supports interval and nominal inputs
  - supports nominal targets (decision trees)
  - supports interval targets (regression trees)
  - provides the entropy, Gini, FastCHAID, CHAID, information gain ratio (IGR), and chi-square methods for decision tree growth (for nominal targets)
  - provides the variance, CHAID, and *F* test methods for regression tree growth (for interval targets)
  - provides multiple statistical metrics for decision tree pruning
  - provides C4.5-style decision tree pruning
  - provides ASE-based regression tree pruning
  - provides cost-complexity pruning
  - provides minimum metric subtree selection
  - partitions the input data set into training and validation sets
  - provides surrogate rules in addition to popularity, similarity, or a dedicated branch for missing value assignments
- Score output data set
  - saves scored results for the training data
  - provides predicted levels and posterior probabilities

- Score code file
  - saves SAS DATA step code, which can be used for scoring new data with the tree model
- Rules file
  - saves node rules that describe the leaves of the tree
- Node output data set
  - saves statistics and descriptive information for the nodes in the tree
- Variable importance output data set
  - saves the importance of the input variables in creating the pruned decision tree
  - provides variable importance for the validation set
- Subtree monitoring output data sets
  - save statistical metrics for each subtree that is created during growth
  - save statistical metrics for each subtree that is created during pruning

Because the HPSPLIT procedure is a high-performance analytical procedure, it also does the following:

- enables you to run in distributed mode on a cluster of machines that distribute the data and the computations
- enables you to run in single-machine mode on the server where SAS is installed
- exploits all of the available cores and concurrent threads, regardless of execution mode.

For more information, see the section "Processing Modes" on page 10 in Chapter 3, "Shared Concepts and Topics."

Getting Started: HPSPLIT Procedure

Decision trees are commonly used in banking to predict default in mortgage applications. The data set HMEQ, which is in the sample library, contains observations for 5,960 mortgage applicants. A variable named BAD indicates whether the applicant paid or defaulted on the loan.

This example uses HMEQ to build a tree model that is used to score the data and can be used to score data on new applicants. Table 15.1 describes the variables in HMEQ.

Table 15.1 Variables in the Home Equity (HMEQ) Data Set

Variable	Role	Level	Description
BAD	Target	Binary	1 = applicant defaulted on the loan or is seriously delinquent 0 = applicant paid the loan
CLAGE	Input	Interval	Age of oldest credit line in months
CLNO	Input	Interval	Number of credit lines
DEBTINC	Input	Interval	Debt-to-income ratio
DELINQ	Input	Interval	Number of delinquent credit lines
DEROG	Input	Interval	Number of major derogatory reports
JOB	Input	Nominal	Occupational category
LOAN	Input	Interval	Requested loan amount
MORTDUE	Input	Interval	Amount due on existing mortgage
NINQ	Input	Interval	Number of recent credit inquiries
REASON	Input	Binary	DebtCon = debt consolidation HomeImp = home improvement
VALUE	Input	Interval	Value of current property
YOJ	Input	Interval	Years at present job

Figure 15.1 shows a partial listing of HMEQ.

Figure 15.1 Partial Listing of the HMEQ Data

Obs	BAD	LOAN	MORTDUE	VALUE	REASON	JOB	YOJ	DEROG	DELINQ	CLAGE	NINQ	CLNO	DEBTINC
1	1	1100	25860	39025	HomeImp	Other	10.5	0	0	94.367	1	9	.
2	1	1300	70053	68400	HomeImp	Other	7.0	0	2	121.833	0	14	.
3	1	1500	13500	16700	HomeImp	Other	4.0	0	0	149.467	1	10	.
4	1	1500	.	.			.	.	.	.	.	.	.
5	0	1700	97800	112000	HomeImp	Office	3.0	0	0	93.333	0	14	.
6	1	1700	30548	40320	HomeImp	Other	9.0	0	0	101.466	1	8	37.1136
7	1	1800	48649	57037	HomeImp	Other	5.0	3	2	77.100	1	17	.
8	1	1800	28502	43034	HomeImp	Other	11.0	0	0	88.766	0	8	36.8849
9	1	2000	32700	46740	HomeImp	Other	3.0	0	2	216.933	1	12	.
10	1	2000	.	62250	HomeImp	Sales	16.0	0	0	115.800	0	13	.

The target variable for the tree model is BAD, a nominal variable that has two values (0 indicates payment of loan, and 1 indicates default). The other variables are input variables for the model.

The following statements use the HPSPLIT procedure to create a decision tree and an output file that contains SAS DATA step code for predicting the probability of default:

```
proc hpsplit data=sampsio.hmeq maxdepth=7 maxbranch=2;
   target BAD;
   input DELINQ DEROG JOB NINQ REASON / level=nom;
   input CLAGE CLNO DEBTINC LOAN MORTDUE VALUE YOJ  / level=int;
   prune misc / N <= 10;
   partition fraction(validate=0.2);
   code file='hpsplhme-code.sas';
run;
```

The TARGET statement specifies the target variable, and the INPUT statements specify the input variables and their levels. The MAXDEPTH= option specifies the maximum depth of the tree to be grown, and the MAXBRANCH= option specifies the maximum number of children per node.

By default, the entropy metric is used to grow the tree. The PRUNE statement requests the misclassification rate metric for choosing a node to prune back to a leaf. The option N<=10 stops the pruning when the number of leaves is less than or equal to 10.

The PARTITION statement specifies the probability (0.2) of randomly selecting a given observation in HMEQ for validation; the remaining observations are used for training.

The CODE statement specifies a file named *hpsplmhe-code.sas*, to which SAS DATA step code for scoring is saved.

The following statements score the data in HMEQ and save the results in a SAS data set named SCORED.

```
data scored;
set sampsio.hmeq;
  %include 'hpsplhme-code.sas';
run;
```

A partial listing of SCORED is shown in Figure 15.2.

Figure 15.2 Partial Listing of the Scored HMEQ Data

Obs	BAD	LOAN	MORTDUE	VALUE	REASON	JOB	YOJ	DEROG	DELINQ	CLAGE	NINQ	CLNO
1	1	1100	25860	39025	HomeImp	Other	10.5	0	0	94.367	1	9
2	1	1300	70053	68400	HomeImp	Other	7.0	0	2	121.833	0	14
3	1	1500	13500	16700	HomeImp	Other	4.0	0	0	149.467	1	10
4	1	1500	.	.			.	.	.	.	.	.
5	0	1700	97800	112000	HomeImp	Office	3.0	0	0	93.333	0	14
6	1	1700	30548	40320	HomeImp	Other	9.0	0	0	101.466	1	8
7	1	1800	48649	57037	HomeImp	Other	5.0	3	2	77.100	1	17
8	1	1800	28502	43034	HomeImp	Other	11.0	0	0	88.766	0	8
9	1	2000	32700	46740	HomeImp	Other	3.0	0	2	216.933	1	12
10	1	2000	.	62250	HomeImp	Sales	16.0	0	0	115.800	0	13

Obs	DEBTINC	_NODE_	_LEAF_	_WARN_	P_BAD1	P_BAD0	V_BAD1	V_BAD0
1	.	9	2		0.18923	0.81077	0.16996	0.83004
2	.	14	6		0.30818	0.69182	0.28750	0.71250
3	.	9	2		0.18923	0.81077	0.16996	0.83004
4	.	9	2		0.18923	0.81077	0.16996	0.83004
5	.	9	2		0.18923	0.81077	0.16996	0.83004
6	37.1136	9	2		0.18923	0.81077	0.16996	0.83004
7	.	13	5		0.58125	0.41875	0.60784	0.39216
8	36.8849	9	2		0.18923	0.81077	0.16996	0.83004
9	.	14	6		0.30818	0.69182	0.28750	0.71250
10	.	9	2		0.18923	0.81077	0.16996	0.83004

The data set contains the original variables and new variables that are created by the score statements. The variable P_BAD1 is the proportion of training observations at this leaf that have BAD=1, and this variable can be interpreted as the probability of default. The variable V_BAD1 is the proportion of validation observations at this leaf that have BAD=1. The other new variables are described in the section "Outputs" on page 626

The preceding statements can be used to score new data by including the new data set in place of HMEQ. The new data set must contain the same variables as the data that are used to build the tree model.

Syntax: HPSPLIT Procedure

The following statements and options are available in the HPSPLIT procedure:

PROC HPSPLIT < *options* > **;**
CODE FILE=*filename* **;**
CRITERION *criterion* < / *options* > **;**
ID *variables* **;**
INPUT *variables* < / *option* > **;**
OUTPUT < *output-options* > < / *subtreestat-option* > **;**
PARTITION < *partition-options* > **;**
PERFORMANCE *performance-options* **;**
PRUNE < *prune-options* > **;**
RULES FILE=*filename* **;**
SCORE OUT=*SAS-data-set* **;**
TARGET *variable* < / *option* > **;**

The PROC HPSPLIT statement, the TARGET statement, and the INPUT statement are required. All other statements are optional. You can specify the INPUT statement multiple times.

It is recommended that you use at least one of the following statements: OUTPUT, RULES, or CODE.

The following sections describe the PROC HPSPLIT statement and then describe the other statements in alphabetical order.

PROC HPSPLIT Statement

PROC HPSPLIT < *options* > **;**

The PROC HPSPLIT statement invokes the procedure. Table 15.2 summarizes the options in the PROC HPSPLIT statement.

Table 15.2 PROC HPSPLIT Statement Options

Option	Description
Basic Options	
DATA=	Specifies the input data set
EVENT=	Specifies the formatted value of the target event (decision trees)
INTERVALBINS=	Specifies the number of bins for interval variables
MINVARIANCE=	Sets the minimum variance for a regression tree leaf to be split
NSURROGATES=	Specifies the number of surrogate rules to create
USEVARONCE	Specifies that variables should be split only once per branch
Splitting Options	
LEAFSIZE=	Specifies the minimum number of observations per leaf
MAXBRANCH=	Specifies the maximum leaves per node
MAXDEPTH=	Specifies the maximum tree depth
MINCATSIZE=	Specifies the number of observations per level to consider a level for splitting
MISSING=	Specifies how to handle missing values in an input variable
FastCHAID and chi-square Options	
ALPHA=	Specifies the maximum *p*-value for a split to be considered
BONFERRONI	Enables the Bonferroni adjustment to after-split *p*-values
***F* Test Options**	
ALPHA=	Specifies the maximum *p*-value for a split to be considered
FastCHAID Options	
MINDIST=	Specifies the minimum Kolmogorov-Smirnov distance

You can specify the following *options*:

ALPHA=*number*

specifies the maximum *p*-value for a split to be considered if you specify FASTCHAID, CHISQUARE, or FTEST in the CRITERION statement. Otherwise, this option is ignored.

By default, ALPHA=0.3.

BONFERRONI

enables the Bonferroni adjustment to the *p*-value of each variable's split after the split has been determined by the Kolmogorov-Smirnov distance or chi-squared value if you specify FASTCHAID or CHISQUARE in the CRITERION statement. Otherwise, this option is ignored.

By default, there is no Bonferroni adjustment.

DATA=*SAS-data-set*

names the input SAS data set to be used by PROC HPSPLIT. The default is the most recently created data set.

If the procedure executes in distributed mode, the input data are distributed to memory on the appliance nodes and analyzed in parallel, unless the data are already distributed in the appliance database. In that case the procedure reads the data alongside the distributed database. See the section "Processing Modes" on page 10 about the various execution modes and the section "Alongside-the-Database Execution" on

page 18 about the alongside-the-database model. Both sections are in Chapter 3, "Shared Concepts and Topics."

EVENT='*value***'**
specifies a formatted *value* of the target level variable to use for sorting nominal input levels when PROC HPSPLIT uses the fastest splitting category. (FastCHAID always uses the fastest splitting category.) Ties are broken in internal order.

See section "Input Variable Splitting and Selection" on page 610 for details on splitting categories and the FastCHAID criterion.

The default is the first level of the target as specified by the ORDER= option in the TARGET statement.

INTERVALBINS=*number*
specifies the *number* of bins for interval variables. For more information about interval variable binning, see the section "Details: HPSPLIT Procedure" on page 609.

By default, INTERVALBINS=100.

LEAFSIZE=*number*
specifies the minimum *number* of observations that a split must contain in the training data set in order for the split to be considered.

By default, LEAFSIZE=1.

MAXBRANCH=*number*
specifies the maximum *number* of children per node in the tree. PROC HPSPLIT tries to create this number of children unless it is impossible (for example, if a split variable does not have enough levels).

The default is the number of target levels.

MAXDEPTH=*number*
specifies the maximum depth of the tree to be grown.

The default depends on the value of the MAXBRANCH= option. If MAXBRANCH=2, the default is MAXDEPTH=10. Otherwise, the MAXDEPTH= option is set using the following equation:

$$\text{MAXDEPTH} = \left\lceil \frac{10}{\log_2 (\text{MAXBRANCH})} \right\rceil$$

MINCATSIZE=*number*
specifies the *number* of observations that a nominal variable level must have in order to be considered in the split. Input variable levels that have fewer observations than *number* receive the non-surrogate missing value assignment for that split.

By default, MINCATSIZE=1.

MINDIST=*number*
specifies the minimum Kolmogorov-Smirnov distance for a split to be considered when you specify FASTCHAID in the CRITERION statement. Otherwise, this option is ignored.

By default, MINDIST=0.01.

MINVARIANCE=*value*
specifies the minimum variance for a regression tree leaf to be eligible for splitting. That is, leaves whose variance is less than *value* are not split any further.

By default, MINVARIANCE=1×10^{-8}.

MISSING=BRANCH | POPULARITY | SIMILARITY
specifies how a splitting rule handles missing values. You can specify one of the following:

BRANCH
requests that missing values be assigned to their own branch.

POPULARITY
requests that missing values be assigned to the most popular, or largest, branch.

SIMILARITY
requests that missing values be assigned to the branch they are most similar to (using the chi-square or F test criterion)

By default, MISSING=POPULARITY.

NSURROGATES=*number* (Experimental)
specifies the *number* of surrogate rules to create for each splitting rule.

By default, NSURROGATES=0.

USEVARONCE
specifies that variables are to be split only once on a branch. However, a variable can be used more than once across branches. That is, a variable cannot be split more than once on the path from any leaf to the root node.

CODE Statement

CODE FILE=*filename* **;**

The CODE statement converts the final tree into SAS DATA step code that can be used for scoring. The code is written to the file that is specified by *filename*.

If no CODE statement is specified, no SAS DATA step code is output.

CRITERION Statement

CRITERION *criterion* < / *options* > **;**

The CRITERION statement specifies the criterion by which to grow the tree.

For nominal targets, you can set the *criterion* to one of the following:

CHAID
uses values of a chi-square test (decision tree) or an F test (regression tree) to merge similar levels of nominal inputs until the number of children in the proposed split reaches the value of the MAXBRANCH= option. It then uses the p-values of the final split to determine the variable on which to split. For interval inputs, CHAID chooses the best single split until the number of children in the proposed split reaches the value of the MAXBRANCH= option.

This criterion is available for both interval and nominal targets.

CHISQUARE
uses the p-values to split each variable and then to determine the split.

ENTROPY
uses the gain in information (decrease in entropy) to split each variable and then to determine the split.

FASTCHAID
uses a Kolmogorov-Smirnov splitter to determine splits for each variable, following a recursive method similar to that of Friedman (1977) (after ordering the levels of nominal variables by the level specified in the EVENT= option), and then uses the lowest of each variable's resulting p-values to determine the variable on which to split.

GINI
uses the decrease in Gini statistic to split each variable and then to determine the split.

IGR
uses the entropy metric to split each variable and then uses the information gain ratio to determine the split.

This criterion is available only for nominal targets.

The default *criterion* for nominal targets is ENTROPY.

For interval targets, you can set the *criterion* to one of the following:

FTEST
uses an F test to split each variable and then to determine the split.

VARIANCE
uses the change in target variance to split each variable and then to determine the split.

The default *criterion* for interval targets is VARIANCE.

You can also specify the following *options* after a slash (/):

LEVTHRESH1=*number*
specifies the maximum number of computations to perform for an exhaustive search for a nominal input. If the input variable being examined is a nominal variable, the splitter tries to fall back to the fast algorithm. Otherwise, it falls back to a greedy algorithm. The LEVTHRESH1= option does not affect interval inputs.

By default, LEVTHRESH1=500,000.

LEVTHRESH2=*number*

specifies the maximum number of computations to perform in a greedy search for nominal input variables. If the input variable that is being examined is an interval variable, the LEVTHRESH2= option specifies the number of computations to perform for an exhaustive search of all possible split points.

If the number of computations in either case is greater than *number*, the splitter uses a much faster greedy algorithm.

Although this option is similar to the LEVTHRESH1= option, it specifies the computations of the nominal variable fallback algorithm for finding the best splits of a nominal variable, a calculation that has a much different computational complexity.

By default, LEVTHRESH2=1,000,000.

ID Statement

ID *variables* **;**

The ID statement is used only if an output data set is requested in the SCORE statement. The data set contains the variables that are specified in the ID statement in addition to the target variable and tree leaf information.

INPUT Statement

INPUT *variables* < / *option* > **;**

The INPUT statement specifies input *variables* to the decision tree. The value of *variable* can be a range such as "g_1–g_1000" or the special "_ALL_" value to include all variables in the data set.

Use the LEVEL=NOM option to request that PROC HPSPLIT treat a numeric variable as a nominal input.

Use multiple INPUT statements if you have a set of numeric variables that you want treated as interval inputs and a second set of numeric variables that you want treated as nominal inputs. For example, the following INPUT statements cause NUMVAR1 to be treated as an interval input and NUMVAR2, CHARVAR1, and CHARVAR2 to be treated as a nominal inputs:

```
input numvar1 charvar1;
input numvar2 charvar2 / level=nom;
```

The following two statements are equivalent to the previous two statements:

```
input numvar1 charvar1 / level=int;
input numvar2 charvar2 / level=nom;
```

PROC HPSPLIT treats CHARVAR1 as a nominal input despite the LEVEL=INT option because CHARVAR1 is a character variable type.

You can specify the following *option*:

LEVEL=INT | NOM
specifies whether the specified input *variables* are interval or nominal.

INT
treats all numeric *variables* as interval inputs.

NOM
treats all *variables* as nominal inputs.

Unless the LEVEL= option is specified, numeric *variables* are treated as interval inputs and character *variables* are treated as nominal inputs. Specifying LEVEL=NOM forces all *variables* in that statement to be treated as nominal. PROC HPSPLIT ignores the LEVEL=INT option for character variables.

OUTPUT Statement

OUTPUT < *output-options* > **;**

The OUTPUT statement allows several SAS data sets to be created.

You can specify the following *output-options*:

GROWTHSUBTREE=*SAS-data-set*
writes to the specified *SAS-data-set* a table that contains the requested statistical metrics of the subtrees that are created during growth.

IMPORTANCE=*SAS-data-set*
writes the importance of each variable to the specified *SAS-data-set*.

NODESTATS=*SAS-data-set*
writes a description of the final tree to the specified *SAS-data-set*.

PRUNESUBTREE=*SAS-data-set*
writes to the specified *SAS-data-set* a table that contains the requested statistical metrics of the subtrees that are created during pruning.

PARTITION Statement

PARTITION < *partition-options* > **;**

The PARTITION statement specifies how observations in the input data set are logically partitioned into disjoint subsets for model training and validation. Either you can designate a variable in the input data set and a set of formatted values of that variable to determine the role of each observation, or you can specify proportions to use for random assignment of observations to each role.

You can specify one (but not both) of the following:

FRACTION(VALIDATE=*fraction* **< SEED=***number* **>)**
requests that specified proportions of the observations in the input data set be randomly assigned to training and validation roles. You specify the proportions for testing and validation by using the

VALIDATE= suboption. The SEED suboption sets the seed. Because *fraction* is a per-observation probability, setting *fraction* too low can result in an empty or nearly empty validation set.

The default seed is based on the date and time.

Using the FRACTION option can cause different numbers of observations to be selected for the validation set because this option specifies a per-observation probability. Different partitions can be observed when the number of nodes or threads changes or when PROC HPSPLIT runs in alongside-the-database mode.

The following PARTITION statement shows how to use a probability of choosing a particular observation for the validation set:

```
partition fraction(validate=0.1 seed=1234);
```

In this example, any particular observation has a probability of 10% of being selected for the validation set. All nonselected records are in the training set. The seed that is used for the random number generator is specified by the SEED= option.

ROLEVAR=*variable***(TRAIN='***value***' VALIDATE='***value***')**
names the *variable* in the input data set whose values are used to assign roles to each observation. The formatted values of this *variable*, which are used to assign observations roles, are specified in the TRAIN= and VALIDATE= suboptions.

In the following example, the ROLEVAR= option specifies _PARTIND_ as the variable in the input data set that is used to select the data set.

```
partition rolevar=_partind_(TRAIN='1' VALIDATE='0');
```

The TRAIN= and VALIDATE= options provide the values that indicate whether an observation is in the training or validation set, respectively. Observations in which the variable is missing or a value that corresponds to neither argument are ignored. Formatting and normalization are performed before comparison, so you should specify numeric variable values as formatted values, as in the preceding example.

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 HPSPLIT.

You can also use the PERFORMANCE statement to control whether PROC HPSPLIT executes in single-machine mode or distributed mode.

The PERFORMANCE statement is documented further in the section "PERFORMANCE Statement" on page 36 of Chapter 3, "Shared Concepts and Topics."

PRUNE Statement

PRUNE C45 < / *confidence* > **;**

PRUNE NONE ;

PRUNE COSTCOMPLEXITY < / γ > **;**

PRUNE *by-metric* < / *until-metric operator value* > **;**

PRUNE *by-metric* / < *until-metric* > **MIN ;**

The PRUNE statement controls pruning. It has five different syntaxes: one for C4.5-style pruning, one for no pruning, one for cost-complexity pruning, one for pruning by using a specified metric and choosing the subtree based on the change in a specified metric, and one for pruning by using a specified metric and choosing the subtree based on the minimum of a specified metric.

The default decision tree pruning method is entropy, using the change-in-entropy subtree selection method. The following PRUNE statement example is equivalent to having no PRUNE statement for a nominal target:

```
prune entropy;
```

The default regression tree pruning method is ASE, using the change-in-ASE subtree selection method. The following PRUNE statement example is equivalent to having no PRUNE statement for an interval target:

```
prune ASE;
```

The preceding statements are also equivalent to the following statements, respectively:

```
prune entropy / entropy >= 1.0;
prune ASE / ASE >= 1.0;
```

You can specify the following pruning options:

C45 < / *confidence* >
: requests C4.5-based pruning (Quinlan 1993) based on the upper error rate from the binomial distribution (Wilson 1927; Blyth and Still 1983; Agresti and Coull 1998) at the *confidence* limit. The default *confidence* is 0.25.

  This option is available only for decision trees (nominal targets).

COSTCOMPLEXITY < / γ >
: requests cost-complexity pruning (Breiman et al. 1984). The optional argument, γ, is the penalty for the size of the tree. The final per-leaf penalty value is γ multiplied by the sum of squares error (SSE) of the largest tree and divided by the number of leaves in that tree.

NONE
: turns off pruning.

by-metric **<** / *until-metric operator value* **>**
chooses a node to prune back to a leaf by the specified *by-metric*. Optionally, you can specify an *until-metric*, *operator*, and *value* to control pruning. If you do not specify these arguments, *until-metric* is set to the same metric as *by-metric*, *operator* is set to ">=," and *value* is set to 1.

You can specify the following value for *by-metric* for decision trees (nominal target) or for regression trees (interval target):

ASE chooses the leaf that has the smallest change in the average square error (ASE).

You can specify the following values for *by-metric* only for decision trees (nominal target):

ENTROPY chooses the leaf that has the smallest change in the entropy.

GINI chooses the leaf that has the smallest change in the Gini statistic.

MISC chooses the leaf that has the smallest change in the misclassification rate.

You can specify the following values for *until-metric* for decision trees (nominal target) or for regression trees (interval target):

ASE stops pruning when the per-leaf change in average square error rate (ASE) is *operator value* times the per-leaf change in the ASE of pruning the whole initial tree to a leaf.

N stops pruning when the number of leaves is *operator value*.

You can specify the following values for *until-metric* only for decision trees (nominal target):

ENTROPY stops pruning when the per-leaf change in entropy is *operator value* times the per-leaf change in the entropy of pruning the whole initial tree to a leaf.

GINI stops pruning when the per-leaf change in the Gini statistic is *operator value* times the per-leaf change in the Gini statistic of pruning the whole initial tree to a leaf.

MISC stops pruning when the per-leaf change in misclassification rate is *operator value* times the per-leaf change in the misclassification rate of pruning the whole initial tree to a leaf.

You can specify any of the following values for *operator* for decision trees (nominal target) or for regression trees (interval target):

<= less than or equal to

LE less than or equal to

>= greater than or equal to

GE greater than or equal to

< less than

LT less than

> greater than

GT greater than

= equal to

EQ equal to

by-metric | **<** *until-metric* **>** **MIN**
chooses a node to prune back to a leaf by the specified *by-metric* and selects the subtree at the minimum of the specified *until-metric* as the optimal subtree. Optionally, you can specify an *until-metric*. If you do not specify an *until-metric*, *until-metric* is set to the same metric as *by-metric*.

You can specify the following value for *by-metric* for decision trees (nominal target) or for regression trees (interval target):

ASE chooses the leaf that has the smallest change in the average square error (ASE).

You can specify any of the following values for *by-metric* for nominal targets only:

ENTROPY chooses the leaf that has the smallest change in the entropy.

GINI chooses the leaf that has the smallest change in the Gini statistic.

MISC chooses the leaf that has the smallest change in the misclassification rate.

You can specify the following value for *until-metric* for decision trees (nominal target) or for regression trees (interval target):

ASE selects the subtree with the minimum absolute square error (ASE) as the optimal subtree.

You can specify any of the following values for *until-metric* for nominal targets only:

ENTROPY selects the subtree with the minimum entropy as the optimal subtree.

GINI selects the subtree with the minimum Gini statistic as the optimal subtree.

MISC selects the subtree with the minimum misclassification rate as the optimal subtree.

RULES Statement

RULES FILE=*filename* **;**

The RULES statement writes the final tree's leaves to the file that is specified by *filename*.

If no RULES statement is specified, no rules are output.

SCORE Statement

SCORE OUT=*SAS-data-set* **;**

The SCORE statement scores the training data set by using the tree model that was trained by PROC HPSPLIT and outputs a *SAS-data-set* that contains the scored results. The output data set contains the ID variables that are specified in the ID statement, predictions, and decisions.

For each level of the target, a posterior probability variable is generated in addition to the final predicted level.

TARGET Statement

TARGET *variable* < / *options* > **;**

The TARGET statement names the *variable* whose values PROC HPSPLIT tries to predict. Missing values in the target are ignored except during scoring.

You can specify the following *options*:

LEVEL=INT | NOM
specifies whether the specified target *variable* is interval or nominal.

INT
treats the target as an interval variable and creates a regression tree.

NOM
treats the target as a nominal variable and creates a decision tree.

By default, PROC HPSPLIT creates a decision tree (nominal target).

ORDER=*ordering*
ensures that the target values are levelized in the specified order. You can specify one of the following values for *ordering*:

ASC | ASCENDING levelizes target values in ascending order.

DESC | DESCENDING levelizes target values in descending order. This is the default.

FMTASC | ASCFORMATTED levelizes target values in ascending order of the formatted value.

FMTDESC | DESFORMATTED levelizes target values in descending order of the formatted value.

Details: HPSPLIT Procedure

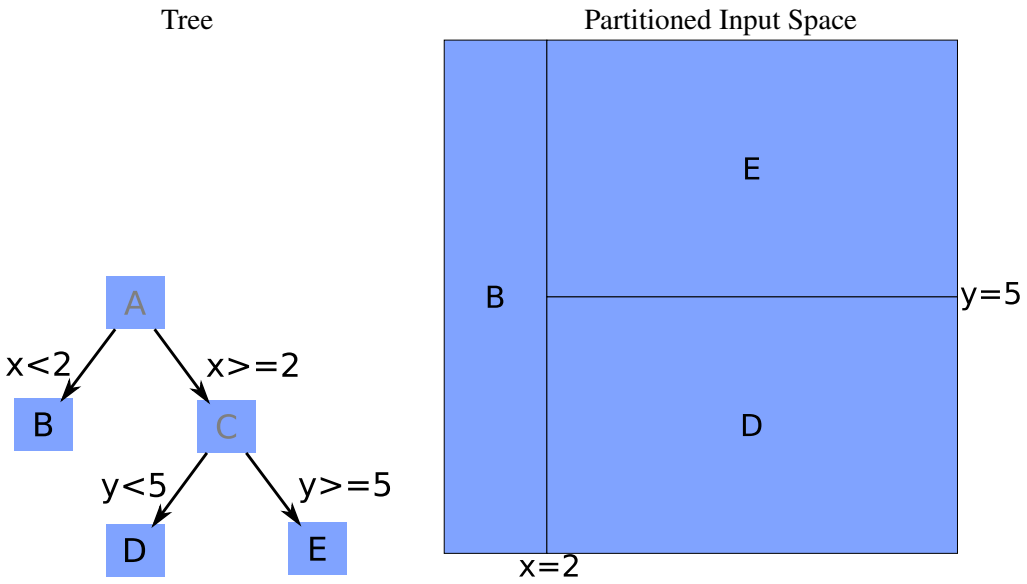
Building a Tree

A decision tree splits the input data into regions by choosing one variable at a time on which to split the data. The splits are hierarchical, so a new split subdivides a previously created region. The simplest situation is a binary split, where only two regions are created from an input region. An interval variable is split by whether the region is less than or is greater than or equal to the split value. Nominal values are collected into two groups.

These hierarchical splits form a tree: the splits are represented by the tree nodes, and the resulting regions are represented by the leaves. Figure 15.3 shows an illustration of a tree and how the space is partitioned by it. The left diagram shows the tree (subdivided region letters are shaded). The splits occur at the tree nodes, and

the leaves are the final regions of the input space. The right diagram shows how the input space is partitioned by the tree. The original data set is the region A, which does not appear on the right. Region A is split into regions B and C by the interval variable X. Region C is subdivided again, this time by the variable Y, into regions D and E. Because the largest number of splits that occur in a path from the top of the tree to the bottommost region is two, the depth of this example tree is two.

Figure 15.3 Conceptual Drawing of a Decision Tree



Interval Input Binning Details

PROC HPSPLIT places interval input variables into bins. You can specify the number of bins by using the INTERVALBINS= option in the PROC HPSPLIT statement. Each bin except the last spans the range

$$\left[\frac{v_{max} - v_{vmin}}{intervalbins} bin + v_{min}, \frac{v_{max} - v_{min}}{intervalbins} (bin + 1) + v_{min}\right)$$

where v_{max} and v_{min} are the maximum and minimum value of the respective variable and *bin* is the bin, which is an integer in the range

$$[0, intervalbins)$$

For the largest bin, the end of the bin range is inclusive.

Input Variable Splitting and Selection

You can use the following criteria to determine a split for nominal targets:

- entropy
- FastCHAID
- Gini

- chi-square

You can use the following criteria to determine a split for interval targets.

- *F* test
- variance

PROC HPSPLIT determines the best split in two stages. First, the splitter uses a splitting algorithm category to find the best split for each variable according to the criterion. Next, the variable that has the best split determines the split of the leaf.

The splitter uses different algorithms called *splitting categories* to find the best split for a variable. Three categories are available: exhaustive, a C4.5-like greedy algorithm that groups levels together using the criterion that is specified in the CRITERION statement until the value specified in the MAXBRANCH= option is reached, and a fast sort–based greedy algorithm. The splitter switches between the different algorithms as the number of levels increases because each splitting category has a different computational complexity that depends on the number of levels.

Splitting Categories and Types

The number of available levels in the variable to be split determines the splitting category. A variable's level is "available" if the variable has not yet been used in the path from the root of the tree to the leaf that is being split, or if a given level has not been switched to a different branch along that path. This definition of "available" allows a variable to be split multiple times. Adjusting the splitting category based on the number of available levels obtains the best split possible according to the statistical criterion while still enabling the splitter to perform quickly despite dealing with an arbitrary number of variable levels or bins.

An exhaustive split search of an interval variable has a much different computational complexity than an exhaustive split search of a nominal variable does. Because of this difference, only two splitter categories are used for interval variables: the exhaustive search and a fast, greedy search. The exhaustive search examines every possible arrangement of splits, up to one less than the value specified in the MAXBRANCH option. The best one is chosen as that variable's split.

If an exhaustive search is computationally infeasible—that is, it requires more operations to perform than the value specified in the LEVTHRESH2= option—the splitter falls back to a faster, greedy algorithm. The greedy algorithm finds the best single split. It then finds the best split of the resulting two regions, choosing the best region and the best split of that region. This process continues until the number of regions equals the value specified in the MAXBRANCH= option or until no further splits are possible.

An exhaustive search of nominal variable splits requires checking every possible assignment of levels to resulting regions. Therefore, the number of operations that are required to perform this search is exponential as a function of the number of variable levels. If the number of operations that are required to perform the exhaustive search is greater than the value specified in the LEVTHRESH1= option, then the splitter uses a faster, greedy search.

The fast greedy algorithm examines each possible pairwise combination of levels. The splitter looks at the best pairwise combination of levels and merges the best pair. This process continues until the number of splits is below one less than the value specified in the MAXBRANCH= option. However, if the number of levels is huge, even this method is infeasible, and the splitter falls back to an even faster method.

After ordering the nominal variable levels based on the EVENT= option, the splitter finds the best splits iteratively. At each iteration, the best split is chosen using the statistical metric for each previously split range of bins or levels. In effect, this combines a number of binary-split nodes into one ensemble of one less than the number of splits specified in the MAXBRANCH= option.

For FastCHAID, the splitter uses only the fastest algorithm regardless of the number of levels. The statistic that is used for choosing split goodness is the Kolmogorov-Smirnov (K-S) distance for the empirical cumulative distribution function. The K-S splitter follows Friedman's (1977) proposal, splitting once at the point that has the maximum K-S distance between all the levels. The splitter then finds the maximum K-S distance of the resulting regions and splits there. The splitter continues until the number of splits is equal to the value specified in the MAXBRANCH= option minus 1.

The chi-square criterion uses the *p*-value of the two-way contingency table of possible split assignments to find the best split of each variable. This criterion can use any splitting category—exhaustive, intermediate, and fastest.

The variance criterion uses the change in variance due to a split to determine each variable's split. Negative changes are not permitted. This criterion can use the exhaustive, intermediate, and fastest splitting categories.

The *F* test criterion uses the *p*-value of the *F* test of possible splits to determine how to split each variable. This criterion can use the exhaustive, intermediate, and fastest splitting categories.

Selecting the Split Variable

After it finds the split for each variable, the splitter chooses the best split variable to use for the final tree node according to the criterion that is specified in the CRITERION statement:

- The entropy and Gini criteria use the named metric to guide the decision.
- The FastCHAID and chi-square criteria use the *p*-value of the two-way table of target-child counts of the proposed split. The ALPHA= option in the PROC HPSPLIT statement specifies the value below which the *p*-value must fall in order to be accepted as a candidate split. In addition, the BONFERRONI keyword in the PROC HPSPLIT statement adjusts the *p*-value of the split by the Bonferroni adjustment.
- The variance criterion uses the change in target variance to choose the best variable on which to split.
- The *F* test criterion uses the *p*-value of an *F* test to determine the best variable on which to split.

The splitting metrics are based on the population that lands at the *node*, not the whole tree. For example, the change in entropy when you split the leaf is determined by the number of observations at that leaf. Although subtle, this distinction makes it potentially useful to grow and prune according to the entropy, even when no validation data set is present. This is because the metric that is used in pruning is based on the partition of the entire data set.

Pruning

You can choose to prune by the following pattern, which uses the *by-metric* to choose a node to prune back to a leaf at each iteration until the per-leaf change in the *until-metric* is *operator value* times the per-leaf change in the *until-metric* of replacing the full tree with a single leaf:

```
PRUNE by-metric / until-metric operator value;
```

For example, the following statement prunes by average square error until the number of leaves falls below (or is equal to) 3:

```
PRUNE ASE / N <= 3;
```

The inequality is necessary because PROC HPSPLIT prunes by removing entire nodes and replacing them with leaves.

The *by-metric* is used to choose which node to replace with a leaf during each pruning step. The smallest global increase (or largest decrease) in the specified metric is the criterion that is used to choose the node to replace. After the pruner chooses the leaf, it uses the *until-metric* to determine whether to terminate pruning.

For example, consider the following statement:

```
PRUNE GINI / ENTROPY >= 1.0;
```

This statement chooses the node that has the least global change (smallest increase or largest decrease) in the Gini statistic when the node is made into a leaf. It also instructs PROC HPSPLIT to terminate when removing the node causes a change in global entropy per leaf that is greater than or equal to the per-leaf change in entropy that is caused by replacing the original tree by a single leaf.

To be more precise, if the original tree has an entropy of E_0 and has N_0 leaves, and if trimming away the whole tree to a stump results in a tree that has an entropy of E_s (and, because it is a stump, just one leaf), then the per-leaf change in entropy is

$$\left.\frac{\Delta E}{\Delta N}\right|_0 = \frac{E_0 - E_s}{N_0 - 1}$$

If the node, n, that is chosen by the pruner has an entropy of E_n, N_n leaves, and an entropy of E_λ if it were replaced by a leaf, then

$$\left.\frac{\Delta E}{\Delta N}\right|_n = \frac{E_n - E_\lambda}{N_n - 1}$$

The preceding statement would cause the pruner to terminate when

$$\frac{\left.\frac{\Delta E}{\Delta N}\right|_n}{\left.\frac{\Delta E}{\Delta N}\right|_0} \geq 1$$

If the original tree's *until-metric* is less than the full tree's *until-metric*, as can be true for a significantly overfitted tree that has a validation set, the denominator in the preceding equation is set to 0 (and the equation is suitably transformed). Setting the denominator to 0 selects the subtree that immediately precedes the first increase (or no change) in *until-metric*.

For all *until-metric*s except N, the default *operator* is >= and the default *value* is 1.0. For the N *until-metric*, the default *operator* is <= and the default *value* is 5.

Memory Considerations

PROC HPSPLIT is built for high-performance computing. As a result, it does not create utility files but rather stores all the data in memory. Data sets that have a large number of variables and few observations, particularly if they have a large number of target levels, can cause PROC HPSPLIT to run out of memory. One way to overcome this is to give SAS more memory to use. Another way to deal with this is to use fewer threads.

Handling Missing Values

When building and pruning a tree, PROC HPSPLIT ignores observations that have a missing value in the target. However, it includes these observations when using the SCORE statement to score the data, and it includes them in the SAS DATA step code.

PROC HPSPLIT always includes observations that have missing values in input variables. It uses a special level or bin for them that is not used in per-variable split determination.

Each split handles missing values by assigning them to one of the children. This ensures that data scored by the SAS DATA step score code can always assign a target to any record.

If you request surrogate rules, these surrogate rules provide a set of backup splitting rules to assign targets. If the variable for a given rule (primary or surrogate) is missing, the backup is used. Thus, the set of surrogate rules provides a progressive set of backups for the preceding splitting rules. If no splitting rule (primary or any surrogate) is usable, the MISSING= option provides a default rule to use.

The MISSING= in the PROC HPSPLIT statement option controls missing value assignment as follows:

- The BRANCH option requests that missing values be assigned to their own branch.
- The POPULARITY option requests that missing values be assigned to the most popular leaf (default).
- The SIMILARITY option requests that missing values be assigned to the branch they are most similar to (using the chi-square or F test criterion).

Observations that correspond to unknown levels are assigned using the default missing value assignment. Nominal variable levels whose counts are less than the value of the MINCATSIZE= option are also assigned using the default missing value assignment.

Surrogate rules are based on the description in Breiman et al. (1984). After splitting a node, PROC HPSPLIT tries to create a surrogate rule for that split from every input variable that is not involved in the split. Only variables that have enough bins or levels to assign one to each new leaf are eligible to be used as surrogates. PROC HPSPLIT first restricts the observations to those that are not missing in both the primary split and in the candidate surrogate. Then it selects the requested number of surrogate-split variables based on the agreement, in order of agreement. That is, the surrogate split that has the largest agreement is first, followed by the second-largest, until it finds the number of surrogate rules that are requested in the NSURROGATES= option or until it runs out of candidates. The agreement is simply the fraction of eligible (that is, nonmissing) observations that are assigned to each new leaf by both the primary and candidate surrogate splitting rules.

Handling Unknown Levels in Scoring

PROC HPSPLIT treats nominal variable values that do not occur in the input data set (either in the validation or in the training set) as missing values in the generated SAS DATA step scoring code.

PROC HPSPLIT assigns interval variable values that are outside the minimum and maximum range in the input data (the training and validation sets together) to either of the end bins. PROC HPSPLIT assigns a value that is less than the minimum to the first bin and a value that is greater than the maximum to the last bin.

Splitting Criteria

When you specify entropy or the Gini statistic as the splitting criterion, the value of the split is judged by the decrease in the specified criterion. Thus, the criterion for the original leaf is computed, as is the criterion for the final, split leaf. The per-variable split and then the variable on which to split are chosen based on the gain.

When you specify FastCHAID as the splitting criterion, splitting is based on the Kolmogorov-Smirnov distance of the variables.

Entropy Splitting Criterion

The entropy is related to the amount of information that a split contains. The entropy of a single leaf λ is given by the equation

$$\text{Entropy}_\lambda = -\sum_t \frac{N_t^\lambda}{N_\lambda} \log_2 \left(\frac{N_t^\lambda}{N_\lambda} \right)$$

where N_t^λ is the number of observations with the target level t on leaf λ and N_λ is the number of observations on the leaf (Hastie, Tibshirani, and Friedman 2001; Quinlan 1993).

When a leaf is split, the total entropy is then

$$\text{Entropy} = -\sum_\lambda \frac{N_\lambda}{N_0} \sum_t \frac{N_t^\lambda}{N_\lambda} \log_2 \left(\frac{N_t^\lambda}{N_\lambda} \right)$$

where N_0 is the number of observations on the original unsplit leaf.

Gini Splitting Criterion

Split Gini is similar to split entropy. First, the per-leaf Gini statistic or index is given by Hastie, Tibshirani, and Friedman (2001) as

$$\text{Gini}_\lambda = \sum_t \frac{N_t^\lambda}{N_\lambda} \left(1 - \frac{N_t^\lambda}{N_\lambda} \right)$$

When split, the Gini statistic is then

$$\text{Gini} = \sum \frac{N_\lambda}{N_0} \sum_t \frac{N_t^\lambda}{N_\lambda} \left(1 - \frac{N_t^\lambda}{N_\lambda} \right)$$

Kolmogorov-Smirnov (FastCHAID) Splitting Criterion

The Kolmogorov-Smirnov (K-S) distance is the maximum distance between the cumulative distribution functions (CDFs) of two or more target levels (Friedman 1977; Rokach and Maimon 2008; Utgoff and Clouse 1996). To create a meaningful CDF for nominal inputs, nominal target levels are ordered first by the level that is specified in the EVENT= option in the PROC HPSPLIT statement (if specified) and then by the other levels in internal order.

After the CDFs have been created, the maximum K-S distance is given by

$$\text{MAXKS} = \text{MAX}_{ijk} \left| CDF_i^{\tau_j} - CDF_i^{\tau_k} \right|$$

where i is an interval variable bin or an explanatory variable level, τ_j is the jth target level, and τ_k is the kth target level.

At each step of determining each variable's split, the maximum K-S distance is computed, resulting in a single split. The splitting continues recursively until the value specified in the MAXBRANCH= option has been reached.

After each variable's split has been determined, the variable that has the lowest p-value is chosen as the variable on which to split. Because this operation is similar to another established tree algorithm (Kass 1980; Soman, Diwakar, and Ajay 2010), this overall criterion is called "FastCHAID."

Chi-Square Splitting Criterion

The chi-square criterion uses the sum of squares of the two-way contingency table that is formed by an input variable's levels and the target levels.

For C children of the split, the sum is equal to

$$\chi^2 = \frac{1}{N} \sum_{\tau}^{T} \sum_{c}^{C} \frac{(N_\tau N_c - N N_{\tau c})^2}{N_\tau N_c}$$

where N is the number of observations under consideration, T is the number of target levels, N_τ is the number of observations that have target level τ, N_c is the number of observations assigned to child c, and $N_{\tau c}$ is the number of observations that have target level τ and are assigned to child c.

After splitting each variable, the variable that has the lowest p-value is chosen as the variable on which to split.

Variance Splitting Criterion

The variance criterion chooses the split that has the maximum reduction in variance. For some leaf λ, variance is calculated as

$$V = \sum_{i}^{\lambda} (y_i - \hat{y}_\lambda)^2$$

So the reduction in variance then becomes

$$\Delta V = \sum_{i}^{\lambda} \left(y_i - y_{\lambda_0}\right)^2 - \sum_{\lambda}^{\lambda_0} \sum_{i}^{\lambda} (y_i - y_\lambda)^2$$

where λ_0 is the leaf being split and λ are the leaves after the split.

This is a type of change in purity criterion.

F Test Splitting Criterion

The F test criterion uses the F sum to choose the best split for a variable:

$$F = \frac{\sum_{\lambda}^{\Lambda} N_{\lambda} \left(\hat{y}_{\lambda} - \hat{y}_{\lambda_0}\right)^2 \left(N_{\lambda_0} - |\Lambda|\right)}{\sum_{\lambda}^{\Lambda} \sum_{i}^{\lambda} \left(y_i - \hat{y}_{\lambda}\right)^2 \left(|\Lambda| - 1\right)}$$

After splitting each variable, the variable that has the lowest F test p-value is chosen as the variable on which to split. The p-value is the probability that $z \geq F$, where z is a random variable from an F distribution that has $N_{\lambda_0} - |\Lambda|, |\Lambda| - 1$ degrees of freedom. Because the value N_{λ_0} is the sum of weights on the unsplit leaf, this is not a strictly correct degrees of freedom.

Pruning Criteria

Pruning criteria are similar to growth criteria, except that they use the global change of a metric instead of the per-leaf change. In addition, partition is present, pruning statistics are calculated from the validation partition if one is present.

Decision Tree Entropy Pruning Criterion

When you prune by entropy, the entropy is calculated as though the entire data set were a single leaf that is partitioned into the final number of leaves. Thus, it can be expected that the path taken during pruning might not correspond to the reverse of the path taken during growth, even if the pruning and growth metrics are identical.

The change is then based on the global entropy, comparing the entropy when node is preserved to the entropy when the node is pruned back to a leaf.

Decision Tree Gini Pruning Criterion

As with entropy, the change in Gini statistic is calculated based on the change in the global Gini statistic. The equations for this criterion are otherwise identical to the equations shown in the section "Gini Splitting Criterion" on page 615.

Decision Tree Misclassification Rate Pruning Criterion

The misclassification rate (MISC) is simply the number of mispredictions divided by the number of predictions. Thus, for a leaf that has a predicted target level τ_P, the misclassification rate is

$$\text{MISC}_{\lambda} = \sum_{\tau_i \neq \tau_P} \frac{N_{\tau_i}^{\lambda}}{N_{\lambda}}$$

For all the leaves in the tree, it is

$$\text{MISC} = \sum_{\lambda} \frac{N_{\lambda}}{N_0} \sum_{\tau_i \neq \tau_P} \frac{N_{\tau_i}^{\lambda}}{N_{\lambda}}$$

The predicted target level is always based on the training data set.

Decision Tree Average Square Error Pruning Criterion

The average square error (ASE) is based on the sum of squares error (SSE). For a perfect assignment, you would expect that the proportion of observations at a leaf λ would be 1 for the predicted target level and 0 for the remainder. Thus, for a single leaf, the equation for the average of this error is

$$\mathrm{ASE}_\lambda = 1 - 2\sum_{\tau_i} \frac{N^{\Lambda}_{\tau_i}}{N_\Lambda}\frac{N^{\lambda}_{\tau_i}}{N_\lambda} + \sum_{\tau_i}\left(\frac{N^{\lambda}_{\tau_i}}{N_\lambda}\right)^2$$

where λ is for a leaf in the training set and Λ is for a leaf in the validation set. If there is no validation set, the training set is used.

Thus, for an ensemble of leaves, the ASE becomes

$$\mathrm{ASE} = \sum_{\Lambda}\frac{N_\lambda}{N_0}\left[1 - 2\sum_{\tau_i} \frac{N^{\Lambda}_{\tau_i}}{N_\Lambda}\frac{N^{\lambda}_{\tau_i}}{N_\lambda} + \sum_{\tau_i}\left(\frac{N^{\lambda}_{\tau_i}}{N_\lambda}\right)^2\right]$$

This summation is over the validation counts set at the leaves, Λ.

Regression Tree Average Square Error Pruning Criterion

Because the predicted value at each leaf is the average at that leaf, the average square error for a regression tree is simply the standard deviation. Thus, for an ensemble of leaves, the ASE becomes

$$\mathrm{ASE} = \frac{1}{N_0}\sum_{\Lambda}\sum_{i\in\lambda}\left(y_i - \hat{y}^T_\lambda\right)^2$$

where y_i is the target value at observation i and $\hat{y}^T_\lambda$ is the predicted value at leaf λ (that is, the average of the target within the training set at that leaf). The T is present to emphasize that it always comes from the training set. This summation is over the validation counts set at the leaves (Λ) and over the observations i in the leaf λ.

Subtree Statistics

Statistics that are printed in the subtree tables are similar to the pruning statistics. There are two ways to calculate the subtree statistics: one is based on a scored data set (using the SCORE statement or the SAS DATA step score code that the CODE statement produces), and the other is based on the internal observation counts at each leaf of the tree. The two methods should provide identical results unless the target is missing.

NOTE: The per-observation and per-leaf methods of calculating the subtree statistics might not agree if the input data set contains observations that have a missing value for the target.

Decision Tree Per-Observation Methods

In scoring, whether you use the SCORE statement or you use the CODE statement with a SAS DATA step, each observation is assigned a posterior probability, P_τ, where τ is a target level. These posterior probabilities are then used to calculate the subtree statistics of the final tree.

For a leaf λ, the posterior probability is the fraction of observations at that leaf that have the target level τ. That is, for leaf λ,

$$P_\tau^\lambda = \frac{N_\tau^\lambda}{N_\lambda}$$

When a record is scored, it is assigned to a leaf, and all posterior probabilities for that leaf are assigned along with it. Thus, for observation ω assigned to leaf λ, the posterior probability is

$$P_\tau^\omega = P_\tau^\lambda = \frac{N_\tau^\lambda}{N_\lambda}$$

The variable N_0 continues to indicate the total number of observations in the input data set, and ω is the observation number (ω is used to prevent confusion with 0).

If a validation set is selected, the statistics are calculated separately for the validation set and for the training set. In addition, the per-observation validation posterior probabilities should be used. The validation posterior probabilities, V_τ^ω, are the same as the posterior probabilities from the training set, but they are the fraction of the observations from the validation set that are in each target level,

$$V_\tau^\omega = V_\tau^\lambda = \frac{N_\tau^\lambda}{N_\lambda}$$

where N_τ^λ and N_λ are now observation counts from the validation set. For calculating the statistics on the validation set, the same equations can be used but substituting V for P where appropriate (for example, V_τ^λ for P_τ^λ).

Decision Tree Observationwise Entropy Statistic

The entropy at each observation is calculated from the posterior probabilities:

$$\text{Entropy} = -\sum_\omega \frac{1}{N_0} \sum_\tau P_\tau^\omega \log_2 \left(P_\tau^\omega\right)$$

Decision Tree Observationwise Gini Statistic

Like the entropy, the Gini statistic is also calculated from the posterior probabilities:

$$\text{Gini} = \sum_\omega \frac{1}{N_0} \sum_\tau P_\tau^\omega \left(1 - P_\tau^\omega\right)$$

Decision Tree Observationwise Misclassification Rate

The misclassification rate is the average number of incorrectly predicted observations in the input data set. Predictions are always based on the training set. Thus, each scored record's predicted target level τ_P^ω is compared against the actual level τ_π^ω:

$$\text{MISC} = \sum_\omega \frac{1 - \delta_{\tau_P^\omega \tau_\pi^\omega}}{N_0}$$

$\delta_{\tau_P^\omega \tau_\pi^\omega}$ is the Kronecker delta:

$$\delta_{\tau_P^\omega \tau_\pi^\omega} = \begin{cases} 1 & \text{if } \tau_p^\omega = \tau_\pi^\omega \\ 0 & \text{otherwise} \end{cases}$$

Phrased slightly differently, the misclassification rate is the fraction of incorrectly predicted observations:

$$\text{MISC} = \frac{1}{N_0} \sum_\omega \begin{cases} 0 & \text{if } \tau_p^\omega = \tau_\pi^\omega \\ 1 & \text{otherwise} \end{cases}$$

Decision Tree Observationwise Sum of Squares Error

For the sum of squares error (SSE), N_τ predictions are made for every observation: that the correct posterior is 1 and that the incorrect posteriors is 0. Thus the SSE is as follows, where τ_π^ω is again the actual target level for observation ω:

$$\text{SSE} = \sum_\omega \left[\sum_{\tau \neq \tau_\pi^\omega} \left(P_\tau^\omega\right)^2 + \left(1 - P_{\tau_\pi^\omega}^\omega\right)^2 \right]$$

Decision Tree Observationwise Average Square Error

The average square error (ASE) is simply the SSE divided by the number of predictions (there are N_τ predictions per observation):

$$\text{ASE} = \frac{1}{N_\tau N_0} \sum_\omega \left[\sum_{\tau \neq \tau_\pi^\omega} \left(P_\tau^\omega\right)^2 + \left(1 - P_{\tau_\pi^\omega}\right)^2 \right]$$

Decision Tree Per-Leaf Methods

The subtree statistics that are calculated by PROC HPSPLIT are calculated per leaf. That is, instead of scanning through the entire data set, PROC HPSPLIT examines the proportions of observations at the leaves. Barring missing target values, which are not handled by the tree, the per-leaf and per-observation methods for calculating the subtree statistics are the same.

As with the per-observation method, observation counts N (N_τ^λ, N_λ, and N_0) can come from either the training set or the validation set. The growth subtree table always produces statistics from the training set. The pruning subtree table produces both sets of data if they are both present.

Unless otherwise marked, counts N can come from either set.

Decision Tree Leafwise Entropy Statistic

Because there are N_λ observations on the leaf λ, entropy takes the following form:

$$\text{Entropy} = -\sum_\lambda \frac{N_\lambda}{N_0} \sum_\tau P_\tau^\lambda \log_2 \left(P_\tau^\lambda\right)$$

Rephrased in terms of N, this becomes

$$\text{Entropy} = -\sum_\lambda \frac{N_\lambda}{N_0} \sum_\tau \frac{N_\tau^\lambda}{N_\lambda} \log_2 \left(\frac{N_\tau^\lambda}{N_\lambda}\right)$$

Decision Tree Leafwise Gini Statistic

The Gini statistic is similar to entropy in its leafwise form:

$$\text{Gini} = \sum_{\lambda} \frac{N_\lambda}{N_0} \sum_{\tau} P_\tau^\lambda \left(1 - P_\tau^\lambda\right)$$

Rephrased in terms of N, this becomes

$$\text{Gini} = \sum_{\lambda} \frac{N_\lambda}{N_0} \sum_{\tau} \frac{N_\tau^\lambda}{N_\lambda} \left(1 - \frac{N_\tau^\lambda}{N_\lambda}\right)$$

Decision Tree Leafwise Misclassification Rate

Misclassification comes from the number of incorrectly predicted observations. Thus, it is necessary to count the proportion of observations at each leaf in each target level. Similar to the misprediction rate of an entire data set, the misprediction rate of a single leaf, is

$$\text{MISC}_\lambda = \frac{1}{N_\lambda} \sum_{\omega} \begin{cases} 0 & \text{if } \tau_p^\omega = \tau_\pi^\omega \\ 1 & \text{otherwise} \end{cases}$$

where the summation is over the observations that arrive at a leaf λ.

All observations at a leaf are assigned the same prediction because they are all assigned the same leaf. Therefore, the summation reduces to simply the number of observations at leaf λ that have a target level other than the predicted target level for that leaf, τ_π. Thus,

$$\begin{aligned} \text{MISC}_\lambda &= \frac{N_\lambda - N_{\tau_\pi}^\lambda}{N_\lambda} \\ &= \sum_{\tau \neq \tau_\pi} \frac{N_\tau^\lambda}{N_\lambda} \\ &= \sum_{\tau \neq \tau_\pi} P_\tau^\lambda \end{aligned}$$

where P_τ^λ is V_τ^λ if the validation set is being examined.

Thus, for the entire data set, the misclassification rate is

$$\begin{aligned} \text{MISC} &= \sum \frac{N_\lambda}{N_0} \sum_{\tau \neq \tau_\pi} P_\tau^\lambda \\ &= \sum \frac{N_\lambda}{N_0} \sum_{\tau \neq \tau_\pi} \frac{N_\tau^\lambda}{N_\lambda} \end{aligned}$$

where again P_τ^λ is V_τ^λ for the validation set.

Decision Tree Leafwise Sum of Squares Error

The sum of squares error (SSE) is treated similarly to the misclassification rate. Each observation is assigned per-target posterior probabilities P_τ^λ from the training data set. These are the predictions for the purpose of the SSE.

The observations at leaf λ are then grouped by the observations' target levels. Because each observation in the group has the same actual target level, Φ, and because all observations on the same node are assigned the same posterior probabilities, P_{τ}^{λ}, the per-observation SSE equation is identical:

$$\begin{aligned}\mathrm{SSE}_{\Phi}^{\lambda} &= \sum_{\omega \in \Phi} \left[\sum_{\tau \neq \tau_{\pi}^{\Phi}} \left(P_{\tau}^{\lambda}\right)^2 + \left(1 - P_{\tau_{\pi}^{\Phi}}^{\lambda}\right)^2 \right] \\ &= N_{\Phi}^{\lambda} \left[\sum_{\tau \neq \tau_{\pi}^{\Phi}} \left(P_{\tau}^{\lambda}\right)^2 + \left(1 - P_{\tau_{\pi}^{\Phi}}^{\lambda}\right)^2 \right]\end{aligned}$$

Here, the posterior probabilities P_{τ}^{λ} are from the training set, and the counts N_{τ}^{λ} are from whichever data set is being examined.

Thus, the SSE equation for the leaf can be rephrased in terms of a further summation over the target levels Φ:

$$\mathrm{SSE}_{\lambda} = \sum_{\Phi} N_{\Phi}^{\lambda} \left[\sum_{\tau \neq \tau_{\pi}^{\Phi}} \left(P_{\tau}^{\lambda}\right)^2 + \left(1 - P_{\tau_{\pi}^{\Phi}}^{\lambda}\right)^2 \right]$$

So the SSE for the entire tree is then

$$\mathrm{SSE} = \sum_{\lambda} \sum_{\Phi} N_{\Phi}^{\lambda} \left[\sum_{\tau \neq \tau_{\pi}^{\Phi}} \left(P_{\tau}^{\lambda}\right)^2 + \left(1 - P_{\tau_{\pi}^{\Phi}}^{\lambda}\right)^2 \right]$$

Substituting the counts from the training set back in and using ν to denote training set counts, this becomes

$$\begin{aligned}\mathrm{SSE} &= \sum_{\lambda} \sum_{\Phi} N_{\Phi}^{\lambda} \left[\sum_{\tau \neq \tau_{\pi}^{\Phi}} \left(\frac{\nu_{\tau}^{\lambda}}{\nu_{\lambda}}\right)^2 + \left(1 - \frac{\nu_{\tau_{\pi}^{\Phi}}^{\lambda}}{\nu_{\lambda}}\right)^2 \right] \\ &= \sum_{\lambda} \sum_{\Phi} N_{\Phi}^{\lambda} \left[\sum_{\tau \neq \tau_{\pi}^{\Phi}} \left(\frac{\nu_{\tau}^{\lambda}}{\nu_{\lambda}}\right)^2 + 1 - 2\frac{\nu_{\tau_{\pi}^{\Phi}}^{\lambda}}{\nu_{\lambda}} + \left(\frac{\nu_{\tau_{\pi}^{\Phi}}^{\lambda}}{\nu_{\lambda}}\right)^2 \right] \\ &= \sum_{\lambda} \sum_{\Phi} N_{\Phi}^{\lambda} \left[\sum_{\tau} \left(\frac{\nu_{\tau}^{\lambda}}{\nu_{\lambda}}\right)^2 + 1 - 2\frac{\nu_{\tau_{\pi}^{\Phi}}^{\lambda}}{\nu_{\lambda}} + \right] \\ &= \sum_{\lambda} N_{\lambda} \sum_{\Phi} \frac{N_{\Phi}^{\lambda}}{N_{\lambda}} \left[\sum_{\tau} \left(\frac{\nu_{\tau}^{\lambda}}{\nu_{\lambda}}\right)^2 + 1 - 2\frac{\nu_{\tau_{\pi}^{\Phi}}^{\lambda}}{\nu_{\lambda}} \right] \\ &= \sum_{\lambda} N_{\lambda} \left[1 + \sum_{\tau} \left(\frac{\nu_{\tau}^{\lambda}}{\nu_{\lambda}}\right)^2 - 2 \sum_{\Phi} \frac{N_{\Phi}^{\lambda}}{N_{\lambda}} \frac{\nu_{\tau_{\pi}^{\Phi}}^{\lambda}}{\nu_{\lambda}} \right]\end{aligned}$$

In the rightmost inner summation, τ_{π}^{Φ} is simply Φ, the target level being summed over. This gives the final equivalent forms

$$\text{SSE} = \sum_{\lambda} N_{\lambda} \left[1 + \sum_{\tau} \left(\frac{\nu_{\tau}^{\lambda}}{\nu_{\lambda}} \right)^2 - 2 \sum_{\Phi} \frac{N_{\Phi}^{\lambda}}{N_{\lambda}} \frac{\nu_{\Phi}^{\lambda}}{\nu_{\lambda}} \right]$$

$$\text{SSE} = \sum_{\lambda} N_{\lambda} \left[1 + \sum_{\tau} \left(P_{\tau}^{\lambda} \right)^2 - 2 \sum_{\Phi} V_{\Phi}^{\lambda} P_{\Phi}^{\lambda} \right]$$

where ν and P are again counts and fraction, respectively, from the training set, and N and V are counts and fraction, respectively, from the validation set. (For example, N_{Φ}^{λ} is the number of observations on leaf λ that have target Φ.)

If there is no validation set, the training set is used instead, and the equations simplify to the following (because Φ is merely an index over target levels and can be renamed τ):

$$\text{SSE} = \sum_{\lambda} N_{\lambda} \left[1 - \sum_{\tau} \left(\frac{\nu_{\tau}^{\lambda}}{\nu_{\lambda}} \right)^2 \right]$$

$$\text{SSE} = \sum_{\lambda} N_{\lambda} \left[1 - \sum_{\tau} \left(P_{\tau}^{\lambda} \right)^2 \right]$$

Decision Tree Leafwise Average Square Error

Because the average square error (ASE) is simply the SSE divided by the number of predictions (there are N_{τ}, the number of target levels, predictions per observation), this becomes

$$\text{ASE} = \sum_{\lambda} \frac{N_{\lambda}}{N_{\tau} N_0} \left[1 + \sum_{\tau} \left(\frac{\nu_{\tau}^{\lambda}}{\nu_{\lambda}} \right)^2 - 2 \sum_{\Phi} \frac{N_{\Phi}^{\lambda}}{N_{\lambda}} \frac{\nu_{\Phi}^{\lambda}}{\nu_{\lambda}} \right]$$

$$\text{ASE} = \sum_{\lambda} \frac{N_{\lambda}}{N_{\tau} N_0} \left[1 + \sum_{\tau} \left(P_{\tau}^{\lambda} \right)^2 - 2 \sum_{\Phi} V_{\Phi}^{\lambda} P_{\Phi}^{\lambda} \right]$$

Or, if only the training set is used:

$$\text{SSE} = \sum_{\lambda} \frac{N_{\lambda}}{N_{\tau} N_0} \left[1 - \sum_{\tau} \left(\frac{\nu_{\tau}^{\lambda}}{\nu_{\lambda}} \right)^2 \right]$$

$$\text{SSE} = \sum_{\lambda} \frac{N_{\lambda}}{N_{\tau} N_0} \left[1 - \sum_{\tau} \left(P_{\tau}^{\lambda} \right)^2 \right]$$

Regression Tree Per-Observation Methods

Unlike a decision tree, regression trees model a continuous target. The predictions produced by a decision tree are based on the average of the partitioned space (the observations at the leaves).

As with decision trees, predictions come from the training set. The predicted value at leaf λ is

$$\hat{y}_{\lambda}^{T} = \frac{1}{N_{\lambda}} \sum_{i \in \lambda} y_i^T$$

where the summation is over the observations i within leaf λ, and y_i^T is the value of the target variable at observation i within the training set.

Regression Tree Observationwise Sum of Squares Error

The observationwise sum of squares error (SSE) is the sum of squares of the difference between the observation's value and the value that is predicted for that observation, $\hat{y}_{\omega}^{T}$, which is equal to $\hat{y}_{\lambda}$ where λ is the leaf to which that observation has been assigned, as described in the equation in the preceding section. The SSE is then simply

$$\text{SSE} = \sum_{\omega} \left(y_{\omega} - \hat{y}_{\omega}^{T} \right)^2$$

Regression Tree Observationwise Average Square Error

The observationwise average square error (ASE) is simply the average of the observationwise square differences and is identical to the per-observation ASE:

$$\text{ASE} = \frac{1}{N_0} \sum_{\omega} \left(y_{\omega} - \hat{y}_{\omega}^{T} \right)^2$$

Regression Tree Per-Leaf Methods

Regression Tree Leafwise Sum of Squares Error

The leafwise sum of squares error (SSE) is related to the variance within that leaf. The training set SSE is identical to the sum of the variances within the leaves:

$$\text{SSE} = \sum_{\lambda}^{\Lambda} \sum_{i \in \lambda} \left(y_i - \hat{y}_{\lambda}^{T} \right)^2$$

Regression Tree Leafwise Average Square Error

The average square error (ASE) is simply the sum of weighted per-leaf average SSE's. That is, the ASE is identical to the per-observation ASE:

$$\text{ASE} = \sum_{\lambda}^{\Lambda} \frac{N_{\lambda}}{N_0} \frac{1}{N_{\lambda}} \sum_{i \in \lambda} \left(y_i - \hat{y}_{\lambda}^{T} \right)^2$$

$$\text{ASE} = \frac{1}{N_0} \sum_{\lambda}^{\Lambda} \sum_{i \in \lambda} \left(y_i - \hat{y}_{\lambda}^{T} \right)^2$$

Variable Importance

Variable importance is based on how the variables are used in the pruned tree. Four metrics are used: count, surrogate count, SSE, and relative importance. The count-based variable importance simply counts the number of times in the entire tree that a given variable is used in a split. Similarly, the surrogate count counts the number of times a variable is used in a surrogate splitting rule. The SSE and relative importance are calculated from the training set. They are also calculated again from the validation set if one exists.

The SSE-based variable importance is based on the nodes in which the variable is used in a split. For each variable, the change of the SSE that results from the split is found. The change is

$$\Delta_\eta = \mathrm{SSE}_\eta - \sum_\lambda \mathrm{SSE}_\lambda^\eta$$

where η denotes the node. SSE_η is then the SSE if the node is treated as a leaf, and $\mathrm{SSE}_\lambda^\eta$ is the SSE of the node after it has been split. If the change in SSE is negative (which is possible when you use the validation set), then the change is set to *0*.

If surrogate rules are present, they are also credited with a portion of the reduction in SSE. The credit is related to the agreement; that is, the fraction of nonmissing observations in which the primary and surrogate splitting rules send to the same children is expressed by

$$\Delta_\eta = \kappa \left(\mathrm{SSE}_\eta - \sum_\lambda \mathrm{SSE}_\lambda^\eta \right)$$

where κ is the agreement,

$$\kappa_\eta = \sum_c^\eta \frac{N_{cc}}{N_\eta}$$

where c is a child of node η, N_η is the number of nonmissing observations, and N_{cc} is the number of observations that were assigned to c by both the primary rule and the surrogate rule.

The SSE-based importance is then

$$\mathrm{SSEIMPORT}_{\mathrm{variable}} = \sqrt{\sum_\eta^{\mathrm{variablenodes}} \Delta_\eta}$$

The relative importance metric is based on the SSE of each variable. The maximum SSE variable importance is found. Then all the variables are assigned a relative importance, which is simply

$$\mathrm{IMPORT}_{\mathrm{variable}} = \frac{\mathrm{SSEIMPORT}_{\mathrm{variable}}}{\mathrm{SSEIMPORT}_{\mathrm{max}}}$$

Outputs

Performance Information

The "Performance Information" table is created by default. It displays information about the execution mode. For single-machine mode, the table displays the number of threads used. For distributed mode, the table displays the grid mode (symmetric or asymmetric), the number of compute nodes, and the number of threads per node.

Output Data Sets

SCORE Data Set

Table 15.3 shows the variables that are contained in an example data set that the SCORE statement produces.

Table 15.3 Example SCORE Statement Data Set Variables

Variable	Target Type	Description
VAR	Either	Target variable
LEAF	Either	Leaf number to which this observation is assigned
NODE	Either	Node number to which this observation is assigned
P_VARLEV	Nominal	Proportion of training set at this leaf that has target VAR = LEV
V_VARLEV	Nominal	Proportion of validation set at this leaf that has target VAR = LEV
P_VAR	Interval	Average value of target VAR in the training set
V_VAR	Interval	Average value of target VAR in the validation set

IMPORTANCE= Data Set

The variable importance data set contains the importance of the input variables in creating the pruned decision tree. PROC HPSPLIT outputs two simple count-based importance metrics and two variable importance metrics that are based on the sum of squares error. In addition, it outputs the number of observations that are used in the training and validation sets, the number of observations that have a missing value, and the number of observations that have a missing target. Table 15.4 shows the variables contained in the data set when you specify the IMPORTANCE= option in the OUTPUT statement. In addition to the variables listed below, a variable that contains the importance for each input variable is included.

Table 15.4 Variable Importance Data Set Variables

Variable	Description
CRITERION	Criterion used to generate the tree
ITYPE	Importance type (Count, NSURROGATES, SSE, VSSE, IMPORT, or VIMPORT)
OBSMISS	Number of observations that have a missing value
OBSTMISS	Number of observations that have a missing target
OBSUSED	Number of observations that were used to build the tree (training set)
OBSVALID	Number of observations in the validation set
TREENUM	Tree number (always 1)

NODESTATS= Data Set

The data set specified in the NODESTATS= option in the OUTPUT statement can be used to visualize the tree. Table 15.5 shows the variables in this data set.

Table 15.5 NODESTATS= Data Set Variables

Variable	Target Type	Description
ALLTEXT	Either	Text that describes the split
CRITERION	Either	Which of the three criteria was used
DECISION	Either	Values of the parent variable's split to get to this node
DEPTH	Either	Depth of the node
ID	Either	Node number
LEAF	Either	Leaf number
LINKWIDTH	Either	Fraction of all training observations going to this node
N	Either	Number of training observations at this node
NVALID	Either	Number of validation observations at this node
PARENT	Either	Parent's node number
PREDICTEDVALUE	Either	Value of target predicted at this node
SPLITVAR	Either	Variable used in the split
TREENUM	Either	Tree number (always 1)
P_VARLEV	Nominal	Proportion of training set at this leaf that has target VAR = LEV
V_VARLEV	Nominal	Proportion of validation set at this leaf that has target VAR = LEV
P_VAR	Interval	Average value of target VAR in the training set
V_VAR	Interval	Average value of target VAR in the validation set

GROWTHSUBTREE= and PRUNESUBTREE= Data Sets

During tree growth and pruning, the number of leaves at each growth or pruning iteration and other metrics are output to data sets that are specified in the GROWTHSUBTREE= and PRUNESUBTREE= options, respectively.

The growth and pruning data sets are identical, except that:

- The growth data set reflects statistics of the tree during growth. The pruning data set reflects statistics of the tree during pruning.
- The statistics of the growth data set are always computed from the training subset. The statistics of the pruning data set are computed from the validation subset if one is available. Otherwise, the statistics of the pruning data set are computed from the training subset.

Table 15.6 GROWTHSUBTREE= and PRUNESUBTREE= Data Set Variables

Variable	Target Type	Description
ITERATION	Either	Iteration number
NLEAVES	Either	Number of leaves
TREENUM	Either	Tree number (always 1)
ASE	Either	Training set: average square error
ENTROPY	Nominal	Training set: entropy
GINI	Nominal	Training set: Gini
MISC	Nominal	Training set: misclassification rate
SSE	Either	Training set: sum of squares error
VASE	Either	Validation set: average square error
VENTROPY	Nominal	Validation set: entropy
VGINI	Nominal	Validation set: Gini
VMISC	Nominal	Validation set: misclassification rate
VSSE	Either	Validation set: sum of squares error
ASSESS	Either	Subtree assessment value
		Ratio of slopes or change in errors
FINALTREE	Either	Chosen subtree if 1
CHOSENID	Either	Intermediate number of node selected for pruning
NODECHANGE	Either	Change in node selection metric by pruning node

Examples: HPSPLIT Procedure

Example 15.1: Creating a Node Rules Description of a Tree

This example creates a tree model and saves a node rules representation of the model in a file. It uses the mortgage application data set HMEQ in the Sample Library, which is described in the Getting Started example in section "Getting Started: HPSPLIT Procedure" on page 595.

The following statements create the tree model:

```
proc hpsplit data=sampsio.hmeq maxdepth=7 maxbranch=2;
   target BAD;
   input DELINQ DEROG JOB NINQ REASON / level=nom;
   input CLAGE CLNO DEBTINC LOAN MORTDUE VALUE YOJ  / level=int;
   criterion entropy;
   prune misc / N <= 6;
   partition fraction(validate=0.2);
   rules file='hpsplhme2-rules.txt';
   score out=scored2;
run;
```

The target variable (BAD) and input variables that are specified for the tree model are the same as in the Getting Started example. The criteria for growing and pruning the tree are also the same, except that pruning is stopped when the number of leaves is less than or equal to 6.

The RULES statement specifies a file named *hpsplhme2-rules.txt*, to which the node rules description of the model is saved. The following listing of this file shows that each leaf of the tree (labeled as a NODE) is numbered and described:

```
*-------------------------------------------------------------*
NODE = 2
*-------------------------------------------------------------*
(DELINQ IS ONE OF 5, 6, 7, 8, 10, 11, 12, 13)
AND (DELINQ IS ONE OF 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13)
    PREDICTED VALUE IS 1
    PREDICTED 1 = 0.9403( 63/67)
    PREDICTED 0 = 0.0597( 4/67)
*-------------------------------------------------------------*
NODE = 7
*-------------------------------------------------------------*
MISSING(DEROG) OR (DEROG IS ONE OF 0, 1, 2, 7, 10)
AND MISSING(NINQ) OR (NINQ IS ONE OF 0, 1, 2, 3, 7, 9, 11, 12, 13, 14)
AND MISSING(DELINQ) OR (DELINQ IS ONE OF 1, 2, 3, 4)
AND (DELINQ IS ONE OF 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13)
    PREDICTED VALUE IS 0
    PREDICTED 1 = 0.3283( 263/801)
    PREDICTED 0 = 0.6717( 538/801)
*-------------------------------------------------------------*
NODE = 6
*-------------------------------------------------------------*
(DEROG IS ONE OF 3, 4, 5, 6, 8, 9)
AND MISSING(NINQ) OR (NINQ IS ONE OF 0, 1, 2, 3, 7, 9, 11, 12, 13, 14)
AND MISSING(DELINQ) OR (DELINQ IS ONE OF 1, 2, 3, 4)
AND (DELINQ IS ONE OF 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13)
    PREDICTED VALUE IS 1
    PREDICTED 1 = 0.9032( 28/31)
    PREDICTED 0 = 0.09677( 3/31)
*-------------------------------------------------------------*
NODE = 9
*-------------------------------------------------------------*
MISSING(DEBTINC) OR (DEBTINC <45.137782)
AND MISSING(DELINQ) OR (DELINQ IS ONE OF 0, 15)
    PREDICTED VALUE IS 0
    PREDICTED 1 = 0.1286( 488/3795)
    PREDICTED 0 = 0.8714( 3307/3795)
```

The preceding listing includes each leaf of the tree, along with the proportion of training set observations that are in the region that is represented by the respective leaf. The predicted value is shown for each leaf, along with the fraction of that leaf's observations that is in each of the target levels. The nodes are not numbered consecutively because the non-leaf nodes are not included.

The splits that lead to each leaf are shown above the predicted value and fractions. The same variable can be involved in more than one split. For example, the leaf labeled "NODE = 2" covers the region where DELINQ is between 1 and 8 or between 10 and 13, and also the region where DELINQ is between 5 and 8 or between 10 and 13. In other words, the variable DELINQ is split twice in succession. By preserving multiple splits of the same variable rather than merging them, the rules description makes it possible to traverse the splits from

the bottom of the tree to the top. At this leaf (node 2), the predicted value for BAD is 1 because the majority of the observations (94%) have value 0.

The SCORE statement saves scores for the observations in a SAS data set named SCORED2. Output 15.1.1 lists the first 10 observations of SCORED2.

Output 15.1.1 Scored Input Data Set (HMEQ)

Obs	BAD	_NODE_	_LEAF_	P_BAD1	P_BAD0	V_BAD1	V_BAD0
1	1	9	4	0.50962	0.49038	0.44828	0.55172
2	1	4	1	0.36740	0.63260	0.34973	0.65027
3	1	9	4	0.50962	0.49038	0.44828	0.55172
4	1	9	4	0.50962	0.49038	0.44828	0.55172
5	0	9	4	0.50962	0.49038	0.44828	0.55172
6	1	9	4	0.50962	0.49038	0.44828	0.55172
7	1	3	0	0.70485	0.29515	0.65672	0.34328
8	1	9	4	0.50962	0.49038	0.44828	0.55172
9	1	4	1	0.36740	0.63260	0.34973	0.65027
10	1	9	4	0.50962	0.49038	0.44828	0.55172

The variables _LEAF_ and _NODE_ show the leaf to which the observation was assigned. The variables P_BAD0 and P_BAD1 are the proportions of observations in the training set that have BAD=1 and BAD=0 for that leaf. The variables V_BAD0 and V_BAD1 are the proportions of observations in the validation set that have BAD=1 and BAD=0 for that leaf. For information about the variables in the scored data set, see the section "Outputs" on page 626.

Example 15.2: Assessing Variable Importance

During the manufacture of a semiconductor device, the levels of temperature, atomic composition, and other parameters are vital to ensuring that the final device is usable. This example creates a decision tree model for the performance of finished devices.

The following statements create a data set named MBE_DATA, which contains measurements for 20 devices:

```
data mbe_data;
   label gtemp  = 'Growth Temperature of Substrate';
   label atemp  = 'Anneal Temperature';
   label rot    = 'Rotation Speed';
   label dopant = 'Dopant Atom';
   label usable = 'Experiment Could be Performed';

   input gtemp atemp rot dopant $ 38-40 usable $ 46-54;
   datalines;
   384.614    633.172    1.01933      C       Unusable
   363.874    512.942    0.72057      C       Unusable
   397.395    671.179    0.90419      C       Unusable
   389.962    653.940    1.01417      C       Unusable
   387.763    612.545    1.00417      C       Unusable
   394.206    617.021    1.07188      Si      Usable
   387.135    616.035    0.94740      Si      Usable
```

```
428.783    745.345    0.99087     Si      Unusable
399.365    600.932    1.23307     Si      Unusable
455.502    648.821    1.01703     Si      Unusable
387.362    697.589    1.01623     Ge      Usable
408.872    640.406    0.94543     Ge      Usable
407.734    628.196    1.05137     Ge      Usable
417.343    612.328    1.03960     Ge      Usable
482.539    669.392    0.84249     Ge      Unusable
367.116    564.246    0.99642     Sn      Unusable
398.594    733.839    1.08744     Sn      Unusable
378.032    619.561    1.06137     Sn      Usable
357.544    606.871    0.85205     Sn      Unusable
384.578    635.858    1.12215     Sn      Unusable
;
```

The variables GTEMP and ATEMP are temperatures, ROT is a rotation speed, and DOPANT is the atom that is used during device growth. The variable USABLE indicates whether the device is usable.

The following statements create the tree model:

```
proc hpsplit data=mbe_data maxdepth=1;
   target usable;
   input gtemp atemp rot dopant;
   output importance=import;
   prune none;
run;
```

There is only one INPUT statement because all of the numeric variables are interval inputs.

The MAXDEPTH=1 option specifies that the tree is to stop splitting when the maximum specified depth of one is reached. In other words, PROC HPSPLIT tries to split the data by each input variable and then chooses the best variable on which to split the data. The split that is chosen divides the data into higher and lower incidences of the target variable (USABLE). The PRUNE statement suppresses pruning because there is only one split.

The OUTPUT statement saves information about variable importance in a data set named IMPORT. The following statements list the relevant observation in IMPORT:

```
proc print data=import(where=(itype='Import'));
run;
```

The result of these statements is provided in Output 15.2.1.

Output 15.2.1 Variable Importance of the One-Split Decision Tree

Obs	_TREENUM_	_CRITERION_	_OBSMISS_	_OBSUSED_	_OBSVALID_	_OBSTMISS_	ITYPE	gtemp	atemp	rot	dopant
3	1	Entropy	0	20	0	0	Import	0	0	0	1

The dopant atom is the most important consideration in determining the usability of the sample because the input DOPANT is used in the one-split decision tree (the other input variables are not used at all.)

References

Agresti, A. and Coull, B. A. (1998), "Approximate Is Better Than 'Exact' for Interval Estimation of Binomial Proportions," *American Statistician*, 52, 119–126.

Blyth, C. R. and Still, H. A. (1983), "Binomial Confidence Intervals," *Journal of the American Statistical Association*, 78, 108–116.

Breiman, L., Friedman, J., Olshen, R. A., and Stone, C. J. (1984), *Classification and Regression Trees*, Belmont, CA: Wadsworth.

Friedman, J. H. (1977), "A Recursive Partitioning Decision Rule for Nonparametric Classification," *IEEE Transactions on Computers*, 26, 404–408.

Hastie, T. J., Tibshirani, R. J., and Friedman, J. H. (2001), *The Elements of Statistical Learning*, New York: Springer-Verlag.

Kass, G. V. (1980), "An Exploratory Technique for Investigating Large Quantities of Categorical Data," *Applied Statistics*, 29, 119–127.

Quinlan, R. J. (1993), *C4.5: Programs for Machine Learning*, San Francisco: Morgan Kaufmann.

Rokach, L. and Maimon, O. (2008), *Data Mining with Decision Trees: Theory and Applications*, volume 69 of *Series in Machine Perception and Artificial Intelligence*, London: World Scientific.

Soman, K. P., Diwakar, S., and Ajay, V. (2010), *Insight into Data Mining: Theory and Practice*, New Delhi: PHI Learning.

Utgoff, P. E. and Clouse, J. A. (1996), *A Kolmogorov-Smirnov Metric for Decision Tree Induction*, Technical Report 96-3, University of Massachusetts, Amherst.

Wilson, E. B. (1927), "Probable Inference, the Law of Succession, and Statistical Inference," *Journal of the American Statistical Association*, 22, 209–212.

Subject Index

2D geometric anisotropic structure
    HPLMIXED procedure, 305

adaptive lasso selection
    high-performance statistical procedures, SELECTION statement, 70
Akaike's information criterion
    HPLMIXED procedure, 321
Akaike's information criterion (finite sample corrected version)
    HPLMIXED procedure, 321
alpha level
    HPFMM procedure, 160, 168
    HPGENSELECT procedure, 241
    HPLMIXED procedure, 299, 304
    HPLOGISTIC procedure, 348
analysis of variance
    multivariate (HPCANDISC), 78
ANCOVA effects
    Shared Concepts, 58
ANOVA effects
    Shared Concepts, 58
ANOVA table
    HPREG procedure, 568
ANTE(1) structure
    HPLMIXED procedure, 305
antedependence structure
    HPLMIXED procedure, 305
AR(1) structure
    HPLMIXED procedure, 305
association statistics
    HPLOGISTIC procedure, 374
at sign (@) operator
    Shared Concepts, 56, 60
autoregressive moving-average structure
    HPLMIXED procedure, 305
autoregressive structure
    HPLMIXED procedure, 305

backward elimination
    high-performance statistical procedures, SELECTION statement, 66
banded Toeplitz structure
    HPLMIXED procedure, 305
bar (|) operator
    Shared Concepts, 56, 59, 60
bar (|) operator
    Shared Concepts, 59
Bayes information
    HPFMM procedure, 194
Bayesian analysis
    HPFMM procedure, 146
Bayesian fit statistics
    HPFMM procedure, 194
Bernoulli distribution
    HPFMM procedure, 161
    HPNLMOD procedure, 404
beta distribution
    HPFMM procedure, 161
beta-binomial distribution
    HPFMM procedure, 161
binary distribution
    HPFMM procedure, 161
    HPNLMOD procedure, 404
binomial distribution
    HPFMM procedure, 161
    HPNLMOD procedure, 404
BLUE
    HPLMIXED procedure, 317
BLUP
    HPLMIXED procedure, 317
boundary constraints, 403
    HPLMIXED procedure, 302, 303
bounds
    HPNLMOD procedure, 403

candidates for addition or removal
    HPGENSELECT procedure, 268
    HPLOGISTIC procedure, 372
    HPQUANTSELECT procedure, 529
    HPREG procedure, 567
canonical coefficients, 74
canonical component, 74
canonical discriminant analysis, 73
canonical variables, 74
canonical weights, 74
centering and scaling information
    HPPLS procedure, 456
class level
    HPFMM procedure, 141, 191
    HPGENSELECT procedure, 236, 267
    HPLMIXED procedure, 297, 320
    HPLOGISTIC procedure, 342, 371
    HPPLS procedure, 443, 457
    HPQUANTSELECT procedure, 509, 529
    HPREG procedure, 549, 567
CLASS statement

Shared Concepts, 52
classification table
HPLOGISTIC, 348
HPLOGISTIC procedure, 363, 374
classification variables
Shared Concepts, 52
colon (:) operator
Shared Concepts, 57
complete separation
HPLOGISTIC procedure, 359
components
HPPLS procedure, 430
compound symmetry structure
example (HPLMIXED), 312
HPLMIXED procedure, 305
computational method
HPFMM procedure, 186
HPGENSELECT procedure, 262
HPLMIXED procedure, 318
HPLOGISTIC procedure, 366
HPNLMOD procedure, 414
HPPLS procedure, 455
HPPRINCOMP procedure, 484
HPQUANTSELECT procedure, 527
HPREG procedure, 564
confidence limits
model parameters (HPFMM), 160, 169
model parameters (HPGENSELECT), 241
model parameters (HPLOGISTIC), 348
constrained analysis
HPFMM procedure, 170
constraints
boundary (HPLMIXED), 302, 303
continuous-by-class effects
Shared Concepts, 61
continuous-nesting-class effects
Shared Concepts, 61
convergence criterion
HPFMM procedure, 136, 139
HPGENSELECT procedure, 232–234
HPLMIXED procedure, 294, 295, 320
HPLOGISTIC procedure, 339, 340
HPNLMOD procedure, 399, 400
convergence status
HPFMM procedure, 192
HPGENSELECT procedure, 269
HPLMIXED procedure, 321
HPLOGISTIC procedure, 373
correlation
HPPRINCOMP procedure, 485
principal components, 483, 485
correlation matrix
HPGENSELECT procedure, 233, 271
covariance
HPPRINCOMP procedure, 485
principal components, 483, 485
covariance matrix
HPGENSELECT procedure, 233, 271
covariance parameter estimates
HPLMIXED procedure, 321
covariance structure
antedependence (HPLMIXED), 307
autoregressive (HPLMIXED), 307
autoregressive moving average (HPLMIXED), 307
banded (HPLMIXED), 308
compound symmetry (HPLMIXED), 308
equi-correlation (HPLMIXED), 308
examples (HPLMIXED), 305
factor-analytic (HPLMIXED), 308
heterogeneous autoregressive (HPLMIXED), 307
heterogeneous compound symmetry (HPLMIXED), 308
heterogeneous Toeplitz (HPLMIXED), 308
HPLMIXED procedure, 288, 305
Huynh-Feldt (HPLMIXED), 308
simple (HPLMIXED), 308
Toeplitz (HPLMIXED), 308
unstructured (HPLMIXED), 309
unstructured, correlation (HPLMIXED), 309
variance components (HPLMIXED), 309
crossed effects
Shared Concepts, 59

dash (-) operator
Shared Concepts, 57
data access information
HPPLS procedure, 456
HPPRINCOMP procedure, 485
HPREG procedure, 528, 566
default output
HPFMM procedure, 191
degrees of freedom
HPLMIXED procedure, 299
infinite (HPLMIXED), 299
method (HPLMIXED), 299
residual method (HPLMIXED), 299
diagnostic statistics
HPQUANTSELECT procedure, 522
HPREG procedure, 560
dimension information
HPLMIXED procedure, 320
dimensions
HPGENSELECT procedure, 270
HPLOGISTIC procedure, 373
HPPLS procedure, 457
HPQUANTSELECT procedure, 529
HPREG procedure, 567

direct product structure
HPLMIXED procedure, 305
discriminant analysis
canonical, 73
discriminant functions, 74
displayed output
HPCANDISC procedure, 91
HPGENSELECT procedure, 267
HPLMIXED procedure, 319
HPLOGISTIC procedure, 371
HPNLMOD procedure, 419
HPPLS procedure, 456
HPPRINCOMP procedure, 485
HPQUANTSELECT procedure, 528
HPREG procedure, 566
distributed computing
HPLMIXED procedure, 318
distribution function
HPGENSELECT procedure, 241
distribution functions
HPNLMOD procedure, 412
double dash (- -) operator
Shared Concepts, 57

effect
name length (HPFMM), 141
name length (HPGENSELECT), 236
name length (HPLMIXED), 297
name length (HPLOGISTIC), 342
name length (HPPLS), 442
name length (HPQUANTSELECT), 509
name length (HPREG), 549
eigenvalues
HPPRINCOMP procedure, 486
eigenvalues and eigenvectors
HPPRINCOMP procedure, 470, 483, 486
eigenvectors
HPPRINCOMP procedure, 486
empirical Bayes estimate
HPNLMOD procedure, 408
estimation
mixed model (HPLMIXED), 316
estimation methods
HPLMIXED procedure, 297
examples, HPFMM
binomial data, 199
cattle feeding data, 206
logistic model, binomial cluster, 199
ossification data, 199
PROBMODEL specification, 199
salmonella assay, 214
three-component mixture, 208
Weibull distribution, 208
examples, HPLMIXED
compound symmetry, G-side setup, 313
compound symmetry, R-side setup, 312
holding covariance parameters fixed, 302
specifying lower bounds, 302
split-plot design, 313
subject and no-subject formulation, 313
unstructured covariance, G-side, 304
examples, HPNLMOD procedure
boundary specification, 403
conditional model expression, 424
enzyme data, 395
gamma distribution, 413
join point, 424
plateau model, 424
predicted values, 424
segmented model, 424
starting values, data set, 407
starting values, grid, 406
exponential distribution
HPFMM procedure, 161

factor analytic structures
HPLMIXED procedure, 305
factors
HPPLS procedure, 430
finite mixture models
See HPFMM procedure, 106
first canonical variable, 74
fit criteria
HPQUANTSELECT procedure, 519
HPREG procedure, 558
fit statistics
HPFMM procedure, 192
HPGENSELECT procedure, 270
HPLOGISTIC procedure, 374
HPQUANTSELECT procedure, 530
HPREG procedure, 569
fixed effects
HPLMIXED procedure, 288
fixed-effects parameters
HPLMIXED procedure, 311
folded normal distribution
HPFMM procedure, 161
forward selection
high-performance statistical procedures, SELECTION statement, 63
forward swap selection
high-performance statistical procedures, SELECTION statement, 68
frequency variable
high-performance statistical procedures, 46
HPFMM procedure, 157
HPGENSELECT procedure, 238
HPLOGISTIC procedure, 345

HPPRINCOMP procedure, 480
HPREG procedure, 551

G matrix
HPLMIXED procedure, 288, 303, 311, 312
gamma distribution
HPFMM procedure, 161
HPNLMOD procedure, 404
Gaussian distribution
HPFMM procedure, 161
HPNLMOD procedure, 404
general distribution
HPNLMOD procedure, 404
general effects
Shared Concepts, 61
general linear covariance structure
HPLMIXED procedure, 305
generalized inverse, 318
generalized Poisson distribution
HPFMM procedure, 161
geometric distribution
HPFMM procedure, 161
global tests
HPLOGISTIC procedure, 374
gradient
HPLMIXED procedure, 320
growth curve analysis
example (HPLMIXED), 312

heavy-tailed density
finite mixture models (HPFMM), 162, 175
Hessian matrix
HPLMIXED procedure, 320
heterogeneous
AR(1) structure (HPLMIXED), 305
compound-symmetry structure (HPLMIXED), 305
covariance structure (HPLMIXED), 309
Toeplitz structure (HPLMIXED), 305
high-performance statistical procedures, SELECTION statement
adaptive lasso selection, 70
backward elimination, 66
forward selection, 63
forward swap selection, 68
LAR selection, 69
LASSO selection, 70
stepwise selection, 67
Hosmer-Lemeshow test
HPLOGISTIC procedure, 349, 366, 374
HPCANDISC procedure, 73
computational method, 86
displayed output, 91
examples, 94
fish measurement data, example, 76
general formulas, 86
Mahalanobis distance, 95
MANOVA, 78
missing values, 86
multithreading, 85, 87
multivariate analysis of variance, 78
ODS table names, 93
OUT= data set, 88
output data sets, 83, 88
OUTSTAT= data set, 89
HPFMM procedure, 106
alpha level, 160, 168
Bayes information, 194
Bayesian analysis, 146
Bayesian fit statistics, 194
Bernoulli distribution, 161
beta distribution, 161
beta-binomial distribution, 161
binary distribution, 161
binomial distribution, 161
centering and scaling, 141
class level, 141, 191
computational method, 186
confidence limits, 160, 169
constrained analysis, 170
convergence criterion, 136, 139
convergence status, 192
default output, 191
effect name length, 141
exponential distribution, 161
fit statistics, 192
folded normal distribution, 161
function-based convergence criteria, 136, 138
gamma distribution, 161
Gaussian distribution, 161
generalized Poisson distribution, 161
geometric distribution, 161
gradient-based convergence criteria, 136, 139
heavy-tailed density, 162, 175
hurdle model, 110, 116
input data sets, 137
introductory example, 110
inverse Gaussian distribution, 161
iteration details, 140
iteration history, 192
link function, 164, 169
lognormal distribution, 161
mixing probabilities, 194
model information, 191
multimodal density, 124, 125, 175
multithreading, 168, 186
negative binomial distribution, 161
normal distribution, 161

number of observations, 191
ODS Graphics, 142, 197
ODS table names, 195
offset variable, 165
optimization information, 192
optimization technique, 187
output data set, 190
overdispersion, 109, 110, 116, 118, 173–175, 197, 205, 215
parameter estimates, 193
parameterization, 185
performance information, 191
plots, 107
Poisson distribution, 161
posterior autocorrelations, 195
posterior intervals, 195
posterior summaries, 194
prior distributions, 194
random number seed, 145
residual variance tolerance, 145
response level ordering, 159
response profile, 192
response variable options, 159
restricted analysis, 170
statistical graphics, 197
t distribution, 161
Weibull distribution, 161
weighting, 172
zero-inflated model, 107, 110, 116, 117, 162, 177
HPGENSELECT procedure, 223
alpha level, 241
candidates for addition or removal, 268
class level, 236, 267
computational method, 262
confidence limits, 241
convergence criterion, 232–234
convergence status, 269
correlation matrix, 233, 271
covariance matrix, 233, 271
dimensions, 270
displayed output, 267
distribution function, 241
effect name length, 236
fit statistics, 270
function-based convergence criteria, 232–234
gradient-based convergence criteria, 233, 234
input data sets, 233
iteration history, 269
link function, 243
model information, 267
model options summary, 238
multithreading, 248, 262
number of observations, 268
ODS table names, 271
optimization stage, 270
optimization technique, 264
parameter estimates, 270
performance information, 267
response level ordering, 239
response profile, 268
response variable options, 239
selected effects, 269
selection details, 269
selection information, 268
selection reason, 269
selection summary, 268
stop reason, 269
test data, 262
user-defined formats, 234
validation data, 262
weighting, 250
XML input stream, 234
zero inflation link function, 251
zero-inflation parameter estimates, 271
HPLMIXED procedure, 286
2D geometric anisotropic structure, 305
Akaike's information criterion, 321
Akaike's information criterion (finite sample corrected version), 321
alpha level, 299, 304
ANTE(1) structure, 305
antedependence structure, 305
AR(1) structure, 305
ARMA structure, 305
autoregressive moving-average structure, 305
autoregressive structure, 305
banded Toeplitz structure, 305
BLUE, 295, 317
BLUP, 295, 317
boundary constraints, 302, 303
chi-square test, 299
class level, 297, 320
compound symmetry structure, 305, 312
computational method, 318
confidence interval, 304
confidence limits, 299, 304
convergence criterion, 294, 295, 320
convergence status, 321
covariance parameter estimates, 321
covariance structure, 288, 305
degrees of freedom, 299
dimension information, 320
direct product structure, 305
displayed output, 319
distributed computing, 318
EBLUPs, 304, 318
effect name length, 297
estimation methods, 297

factor analytic structures, 305
fitting information, 321
fixed effects, 288
fixed-effects parameters, 300, 311
function-based convergence criteria, 294, 295
G matrix, 288, 303, 311, 312
general linear covariance structure, 305
generalized inverse, 318
gradient, 320
gradient-based convergence criteria, 294, 295
grid search, 301
growth curve analysis, 312
Hessian matrix, 320
heterogeneous AR(1) structure, 305
heterogeneous compound-symmetry structure, 305
heterogeneous covariance structures, 309
heterogeneous Toeplitz structure, 305
Huynh-Feldt structure, 305
infinite degrees of freedom, 299
infinite likelihood, 309
initial values, 301
input data sets, 295
intercept effect, 300, 303
iteration history, 320
iterations, 320
Kronecker product structure, 305
linear covariance structure, 305
Matérn covariance structure, 305
matrix notation, 310
mixed model, 311
mixed model equations, 317
mixed model theory, 310
model information, 319
multi-threading, 303
multithreading, 318
Newton-Raphson algorithm, 316
number of observations, 320
ODS table names, 322
optimization information, 320
parameter constraints, 302
parameter-based convergence criteria, 298
performance information, 319
R matrix, 288, 309, 311, 312
random effects, 288, 303
random-effects parameters, 304, 311
repeated measures, 309
residual method, 299
restricted maximum likelihood (REML), 287
ridging, 316
Schwarz's Bayesian information criterion, 321
spatial anisotropic exponential structure, 305
split-plot design, 313
standard linear model, 288
statement positions, 292
subject effect, 304, 310
summary of commands, 292
table names, 322
timing, 322
Toeplitz structure, 305
unstructured correlations, 305
unstructured covariance matrix, 305
variance components, 305
variance ratios, 302
HPLOGISTIC procedure, 329
alpha level, 348
association statistics, 374
candidates for addition or removal, 372
class level, 342, 371
classification table, 348, 363, 374
complete separation, 359
computational method, 366
confidence limits, 348
convergence criterion, 339, 340
convergence status, 373
dimensions, 373
displayed output, 371
effect name length, 342
existence of MLEs, 359
fit statistics, 374
function-based convergence criteria, 339, 340
global tests, 374
gradient-based convergence criteria, 339, 340
Hosmer-Lemeshow test, 349, 366, 374
infinite parameter estimates, 359
input data sets, 339
iteration history, 373
link function, 350
model information, 371
model options summary, 346
multithreading, 354, 366
number of observations, 371
ODS table names, 375
optimization technique, 367
parameter estimates, 375
performance information, 371
quasi-complete separation, 359
response level ordering, 346
response profile, 371
response variable options, 346
selected effects, 372
selection information, 372
selection reason, 372
selection summary, 372
separation, 359
stop reason, 372
test data, 360
user-defined formats, 340

validation, 360
weighting, 356
XML input stream, 340
HPNLMOD procedure, 393
additional estimates, 403, 421
additional estimates correlation, 422
additional estimates covariance, 422
ANOVA, 421
Bernoulli distribution, 404
binary distribution, 404
binomial distribution, 404
bounds, 403
computational method, 414
convergence criterion, 399, 400
convergence status, 420
correlation, 421
covariance, 421
dimensions, 420
displayed output, 419
distribution functions, 412
empirical Bayes estimate, 408
fit statistics, 420
function-based convergence criteria, 399, 400
gamma distribution, 404
Gaussian distribution, 404
general distribution, 404
gradient-based convergence criteria, 399, 400
initial values, 405
input data sets, 400
iteration history, 420
lag functionality, 410
least squares, 412
least squares distribution, 404
linear constraints, 420
log-likelihood functions, 412
multithreading, 407, 414
negative binomial distribution, 404
normal distribution, 404
number of observations, 419
ODS table names, 422
optimization technique, 415
output data sets, 402
parameter estimates, 421
parameters, 420
performance information, 419
Poisson distribution, 404
prediction, 408
procedure task timing, 422
programming statements, 410
residual distribution, 404
restrictions, 409
segmented model example, 423
specifications, 419
starting values, 405
user-defined formats, 400, 402
XML input stream, 400, 402
HPPLS procedure, 430
algorithms, 442
centering, 454
centering and scaling information, 456
class level, 443, 457
compared to other procedures, 430
components, 430
computational method, 441, 455
data access information, 456
dimensions, 457
displayed output, 456
effect name length, 442
examples, 459
factors, 430
factors, selecting the number of, 435
input data set, 441
introductory example, 432
latent variables, 430
latent vectors, 430
missing values, 454
model details, 457
model information, 456
multithreading, 448, 455
number of observations, 457
ODS table names, 458
output data sets, 445, 456
parameter estimates, 458
partial least squares regression, 430, 431, 448
percentage of variation accounted for by extracted factors, 457
performance information, 456
predicting new observations, 438
principal components regression, 431, 449
reduced rank regression, 431, 449
scaling, 454
SIMPLS method, 449
test set validation, 452, 457, 459
timing, 458
HPPRINCOMP procedure, 470
computational method, 484
correction for means, 477
correlation, 485
covariance, 485
crime rate data, example, 471
data access information, 485
displayed output, 485
eigenvalues, 486
eigenvalues and eigenvectors, 470, 483, 486
eigenvectors, 486
examples, 487
input data set, 477
multithreading, 480, 484

number of observations, 485
number of variables, 485
ODS table names, 486
OUT= data set, 481
output data sets, 477, 481–483
OUTSTAT= data set, 482
partial correlation, 486
partial covariance, 486
performance information, 485
regression coefficients, 485
regression statistics, 485
SCORE procedure, 484
simple statistics, 485
suppressing output, 477
timing, 486
total variance, 486
weights, 481
HPQUANTSELECT procedure, 494
candidates for addition or removal, 529
class level, 509, 529
computational method, 527
diagnostic statistics, 522
dimensions, 529
displayed output, 528
effect name length, 509
fit criteria, 519
fit statistics, 530
information criteria, 519
input data sets, 508
introductory example, 498
macro variables, 524
model information, 528
multithreading, 515, 527
number of observations, 529
ODS table names, 531
output data set, 528
parameter estimates, 531
performance information, 528
quantile regression, 517
random number seed, 509
selected effects, 530
selection information, 528
selection reason, 530
selection summary, 529
significance level criteria, 520
stop reason, 529
test data, 525
timing, 531
user-defined formats, 509
validation, 525
weighting, 517
XML input stream, 509
HPREG procedure, 539
ANOVA table, 568
candidates for addition or removal, 567
class level, 549, 567
computational method, 564
data access information, 528, 566
diagnostic statistics, 560
dimensions, 567
displayed output, 566
effect name length, 549
fit criteria, 558
fit statistics, 569
input data sets, 549
introductory example, 542
model information, 566
multithreading, 555, 564
number of observations, 567
ODS table names, 570
output data set, 565
parameter estimates, 570
performance information, 566
random number seed, 549
screened effects, 567
screening, 565, 567
screening information, 566
selected effects, 568
selection information, 566
selection reason, 568
selection summary, 567
stop reason, 568
test data, 562
timing, 570
user-defined formats, 549
validation, 562
weighting, 558
XML input stream, 549
hurdle model
finite mixture models (HPFMM), 110, 116
Huynh-Feldt
structure (HPLMIXED), 305

infinite likelihood
HPLMIXED procedure, 309
infinite parameter estimates
HPLOGISTIC procedure, 359
information criteria
HPQUANTSELECT procedure, 519
initial values
HPLMIXED procedure, 301
HPNLMOD procedure, 405
interaction effects
Shared Concepts, 59
intercept
Shared Concepts, 59
inverse Gaussian distribution
HPFMM procedure, 161

iteration details
    HPFMM procedure, 140
iteration history
    HPFMM procedure, 192
    HPGENSELECT procedure, 269
    HPLMIXED procedure, 320
    HPLOGISTIC procedure, 373
iterations
    history (HPLMIXED), 320

Kronecker product structure
    HPLMIXED procedure, 305

lag functionality
    HPNLMOD procedure, 410
LAR selection
    high-performance statistical procedures, SELECTION statement, 69
LASSO selection
    high-performance statistical procedures, SELECTION statement, 70
latent variables
    HPPLS procedure, 430
latent vectors
    HPPLS procedure, 430
least squares
    HPNLMOD procedure, 412
least squares distribution
    HPNLMOD procedure, 404
levelization
    Shared Concepts, 52
linear covariance structure
    HPLMIXED procedure, 305
link function
    HPFMM procedure, 164, 169
    HPGENSELECT procedure, 243
    HPLOGISTIC procedure, 350
log-likelihood functions
    HPNLMOD procedure, 412
lognormal distribution
    HPFMM procedure, 161

macro variables
    HPQUANTSELECT procedure, 524
Mahalanobis distance, 82
    HPCANDISC procedure, 95
main effects
    Shared Concepts, 59
MANOVA
    HPCANDISC procedure, 78
Matérn covariance structure
    HPLMIXED procedure, 305
matrix
    notation, theory (HPLMIXED), 310
maximum likelihood
    estimates (HPLOGISTIC), 359
maximum likelihood estimation
    mixed model (HPLMIXED), 316
missing values
    HPCANDISC procedure, 86
    HPPLS procedure, 454
    HPPRINCOMP procedure, 481
mixed model (HPLMIXED)
    estimation, 316
    formulation, 311
    maximum likelihood estimation, 316
    notation, 288
    objective function, 320
    theory, 310
mixed model equations
    HPLMIXED procedure, 317
mixing probabilities
    HPFMM procedure, 194
mixture model (HPFMM)
    parameterization, 185
model
    information (HPFMM), 191
    information (HPGENSELECT), 267
    information (HPLOGISTIC), 371
    information (HPQUANTSELECT), 528
    information (HPREG), 566
    specification (HPNLMOD procedure, 404
model details
    HPPLS procedure, 457
model information
    HPLMIXED procedure, 319
    HPPLS procedure, 456
multi-threading
    HPLMIXED procedure, 303
multimodal density
    finite mixture models (HPFMM), 124, 125, 175
multithreading
    HPCANDISC procedure, 85
    HPFMM procedure, 168, 186
    HPGENSELECT procedure, 248, 262
    HPLMIXED procedure, 318
    HPLOGISTIC procedure, 354, 366
    HPNLMOD procedure, 407, 414
    HPPLS procedure, 448, 455
    HPPRINCOMP procedure, 480, 484
    HPQUANTSELECT procedure, 515, 527
    HPREG procedure, 555, 564
multivariate analysis of variance
    HPCANDISC procedure, 78

negative binomial distribution
    HPFMM procedure, 161
    HPNLMOD procedure, 404
nested effects

Shared Concepts, 60
nested versus crossed effects
Shared Concepts, 60
Newton-Raphson algorithm
HPLMIXED procedure, 316
normal distribution
HPFMM procedure, 161
HPNLMOD procedure, 404
number of observations
HPFMM procedure, 191
HPGENSELECT procedure, 268
HPLMIXED procedure, 320
HPLOGISTIC procedure, 371
HPPLS procedure, 457
HPPRINCOMP procedure, 485
HPQUANTSELECT procedure, 529
HPREG procedure, 567
number of variables
HPPRINCOMP procedure, 485

objective function
mixed model (HPLMIXED), 320
ODS Graphics
HPFMM procedure, 142, 197
ODS table names
HPCANDISC procedure, 93
HPPLS procedure, 458
HPPRINCOMP procedure, 486
offset variable
HPFMM procedure, 165
optimization information
HPFMM procedure, 192
HPLMIXED procedure, 320
optimization stage
HPGENSELECT procedure, 270
optimization technique
HPFMM procedure, 187
HPGENSELECT procedure, 264
HPLOGISTIC procedure, 367
HPNLMOD procedure, 415
options summary
BAYES statement, 147
MODEL statement (HPFMM), 158
MODEL statement (HPLMIXED), 299
PROC HPFMM statement, 134
PROC HPGENSELECT statement, 231
PROC HPLMIXED statement, 293
PROC HPLOGISTIC statement, 338
PROC HPNLMOD statement, 398
PROC HPQUANTSELECT statement, 508
PROC HPREG statement, 548
RANDOM statement (HPLMIXED), 303
REPEATED statement (HPLMIXED), 309
ordering
of class levels (Shared Concepts), 53
OUT= data set
HPCANDISC procedure, 88
HPPRINCOMP procedure, 481
output data set
HPFMM procedure, 190
HPQUANTSELECT procedure, 528
HPREG procedure, 565
output data sets
HPPLS procedure, 445, 456
OUTSTAT= data set
HPCANDISC procedure, 89
HPPRINCOMP procedure, 482
overdispersion
finite mixture models (HPFMM), 109, 110, 116, 118, 173–175, 197, 205, 215
overlap of data points
HPLOGISTIC procedure, 359

parameter constraints
HPLMIXED procedure, 302
parameter estimates
HPFMM procedure, 193
HPGENSELECT procedure, 270
HPLOGISTIC procedure, 375
HPPLS procedure, 458
HPQUANTSELECT procedure, 531
HPREG procedure, 570
parameterization
HPFMM procedure, 185
mixture model (HPFMM), 185
Shared Concepts, 55
partial correlation
HPPRINCOMP procedure, 486
principal components, 486
partial covariance
HPPRINCOMP procedure, 486
principal components, 486
partial least squares, 430, 431, 448
percentage of variation accounted for by extracted factors
HPPLS procedure, 457
performance information
HPFMM procedure, 191
HPGENSELECT procedure, 267
HPLMIXED procedure, 319
HPLOGISTIC procedure, 371
HPPLS procedure, 456
HPPRINCOMP procedure, 485
HPQUANTSELECT procedure, 528
HPREG procedure, 566
plots
finite mixture models (HPFMM), 107
Poisson distribution

HPFMM procedure, 161
HPNLMOD procedure, 404
polynomial effects
Shared Concepts, 59
posterior autocorrelations
HPFMM procedure, 195
posterior intervals
HPFMM procedure, 195
posterior summaries
HPFMM procedure, 194
prediction
HPNLMOD procedure, 408
principal components
interpreting eigenvalues, 474
partialing out variables, 480
regression (HPPLS), 431, 449
rotating, 484
using weights, 481
prior distributions
HPFMM procedure, 194
probability distributions
HPFMM procedure, 161
programming statements
HPNLMOD procedure, 410

quantile regression
HPQUANTSELECT procedure, 517
quasi-complete separation
HPLOGISTIC procedure, 359

R matrix
HPLMIXED procedure, 288, 309, 311, 312
random effects
HPLMIXED procedure, 288, 303
random number seed
HPFMM procedure, 145
random-effects parameters
HPLMIXED procedure, 311
reduced rank regression, 431
HPPLS procedure, 449
regression
partial least squares (PROC HPPLS), 431, 448
principal components (PROC HPPLS), 431, 449
reduced rank (PROC HPPLS), 431, 449
regression coefficients
HPPRINCOMP procedure, 485
regression effects
Shared Concepts, 58, 59
regression statistics
HPPRINCOMP procedure, 485
repeated measures
HPLMIXED procedure, 309
residual distribution
HPNLMOD procedure, 404
residual maximum likelihood (REML)
HPLMIXED procedure, 316
residuals
and partial correlation (HPPRINCOMP), 482
partial correlation (HPPRINCOMP), 480
response level ordering
HPFMM procedure, 159
HPGENSELECT procedure, 239
HPLOGISTIC procedure, 346
response profile
HPFMM procedure, 192
HPGENSELECT procedure, 268
HPLOGISTIC procedure, 371
response variable options
HPFMM procedure, 159
HPGENSELECT procedure, 239
HPLOGISTIC procedure, 346
restricted analysis
HPFMM procedure, 170
restricted maximum likelihood
HPLMIXED procedure, 287
restricted maximum likelihood (REML)
HPLMIXED procedure, 316
restrictions
HPNLMOD procedure, 409
reverse response level ordering
HPFMM procedure, 159
HPGENSELECT procedure, 239
HPLOGISTIC procedure, 346
ridging
HPLMIXED procedure, 316
rotating principal components, 484

Schwarz's Bayesian information criterion
HPLMIXED procedure, 321
SCORE procedure
HPPRINCOMP procedure, 484
screened effects
HPREG procedure, 567
screening
HPREG procedure, 565, 567
screening information
HPREG procedure, 566
selected effects
HPGENSELECT procedure, 269
HPLOGISTIC procedure, 372
HPQUANTSELECT procedure, 530
HPREG procedure, 568
selection details
HPGENSELECT procedure, 269
selection information
HPGENSELECT procedure, 268
HPLOGISTIC procedure, 372
HPQUANTSELECT procedure, 528

HPREG procedure, 566
selection reason
HPGENSELECT procedure, 269
HPLOGISTIC procedure, 372
HPQUANTSELECT procedure, 530
HPREG procedure, 568
selection summary
HPGENSELECT procedure, 268
HPLOGISTIC procedure, 372
HPQUANTSELECT procedure, 529
HPREG procedure, 567
separation
HPLOGISTIC procedure, 359
Shared Concepts
ANCOVA effects, 58
ANOVA effects, 58
at sign (@) operator, 56, 60
bar (|) operator, 56, 59, 60
bar (|) operator, 59
CLASS statement, 52
classification variables, 52
colon (:) operator, 57
continuous-by-class effects, 61
continuous-nesting-class effects, 61
crossed effects, 59
dash (-) operator, 57
double dash (- -) operator, 57
general effects, 61
interaction effects, 59
intercept, 59
levelization, 52
main effects, 59
missing values, class variables, 54
nested effects, 60
nested versus crossed effects, 60
ORDER= option, 53
ordering of class levels, 53
parameterization, 55
polynomial effects, 59
regression effects, 58, 59
singular parameterization, 59
sort order of class levels, 53
significance level criteria
HPQUANTSELECT procedure, 520
simple statistics
HPPRINCOMP procedure, 485
SIMPLS method
HPPLS procedure, 449
singular parameterization
Shared Concepts, 59
sort order
of class levels (Shared Concepts), 53
spatial anisotropic exponential structure
HPLMIXED procedure, 305
split-plot design
HPLMIXED procedure, 313
standard linear model
HPLMIXED procedure, 288
starting values
HPNLMOD procedure, 405
statistical graphics
HPFMM procedure, 197
stepwise selection
high-performance statistical procedures, SELECTION statement, 67
stop reason
HPGENSELECT procedure, 269
HPLOGISTIC procedure, 372
HPQUANTSELECT procedure, 529
HPREG procedure, 568
subject effect
HPLMIXED procedure, 304, 310
summary of commands
HPLMIXED procedure, 292

t distribution
HPFMM procedure, 161
table names
HPLMIXED procedure, 322
test data
HPGENSELECT procedure, 262
HPLOGISTIC procedure, 360
HPQUANTSELECT procedure, 525
HPREG procedure, 562
test set validation
HPPLS procedure, 452, 457
timing
HPLMIXED procedure, 322
HPPLS procedure, 458
HPPRINCOMP procedure, 486
HPQUANTSELECT procedure, 531
HPREG procedure, 570
Toeplitz structure
HPLMIXED procedure, 305
total variance
HPPRINCOMP procedure, 486

unstructured correlations
HPLMIXED procedure, 305
unstructured covariance matrix
HPLMIXED procedure, 305

validation
HPLOGISTIC procedure, 360
HPQUANTSELECT procedure, 525
HPREG procedure, 562
validation data
HPGENSELECT procedure, 262
variance components

HPLMIXED procedure, 305
variance ratios
HPLMIXED procedure, 302

Weibull distribution
HPFMM procedure, 161
weighting
HPFMM procedure, 172
HPGENSELECT procedure, 250
HPLOGISTIC procedure, 356
HPQUANTSELECT procedure, 517
HPREG procedure, 558

zero inflation link function
HPGENSELECT procedure, 251
zero-inflated model
finite mixture models (HPFMM), 107, 110, 116, 117, 162, 177
zero-inflation parameter estimates
HPGENSELECT procedure, 271

Syntax Index

ABSCONV option
PROC HPFMM statement, 136
PROC HPGENSELECT statement, 232
PROC HPLMIXED statement, 294
PROC HPLOGISTIC statement, 339
PROC HPNLMOD statement, 399
ABSFCONV option
PROC HPFMM statement, 136
PROC HPGENSELECT statement, 233
PROC HPLMIXED statement, 294
PROC HPLOGISTIC statement, 339
PROC HPNLMOD statement, 399
ABSGCONV option
PROC HPFMM statement, 136
PROC HPGENSELECT statement, 233
PROC HPLMIXED statement, 294
PROC HPLOGISTIC statement, 339
PROC HPNLMOD statement, 399
ABSGTOL option
PROC HPFMM statement, 136
PROC HPGENSELECT statement, 233
PROC HPLMIXED statement, 294
PROC HPLOGISTIC statement, 339
PROC HPNLMOD statement, 399
ABSOLUTE option
PROC HPLMIXED statement, 320
ABSTOL option
PROC HPFMM statement, 136
PROC HPGENSELECT statement, 232
PROC HPLMIXED statement, 294
PROC HPLOGISTIC statement, 339
PROC HPNLMOD statement, 399
ADAPTIVE option
SELECTION statement (high-performance statistical procedures), 48
ALGORITHM= option
PROC HPPLS statement, METHOD=PLS option, 442
ALL option
PROC HPCANDISC statement, 82
ALLSTATS option
OUTPUT statement (HPFMM), 167
OUTPUT statement (HPLMIXED), 301
ALPHA= option
ESTIMATE statement (HPNLMOD), 404
MODEL statement (HPFMM), 160
MODEL statement (HPGENSELECT), 241
MODEL statement (HPLOGISTIC), 348
OUTPUT statement (HPGENSELECT), 247
PREDICT statement (HPNLMOD), 408
PROBMODEL statement (HPFMM, 168
PROC HPGENSELECT statement, 233
PROC HPLOGISTIC statement, 339
PROC HPNLMOD statement, 399
PROC HPQUANTSELECT statement, 508
PROC HPREG statement, 549
RANDOM statement (HPLMIXED), 304
ANOVA option
PROC HPCANDISC statement, 82
ASSOCIATION option
MODEL statement (HPLOGISTIC), 348

BAYES statement
HPFMM procedure, 146
BCORR option
PROC HPCANDISC statement, 82
BCOV option
PROC HPCANDISC statement, 82
BETAPRIORPARMS option
BAYES statement (HPFMM), 147
BLUP option
PROC HPLMIXED statement, 295
BOUNDS statement
HPNLMOD procedure, 403
BSSCP option
PROC HPCANDISC statement, 82
BY statement
HPCANDISC procedure, 84
HPFMM procedure, 156
HPLOGISTIC procedure, 343
HPNLMOD procedure, 403
HPPLS procedure, 443
HPPRINCOMP procedure, 479
HPQUANTSELECT procedure, 509
HPREG procedure, 549

CENSCALE option
PROC HPPLS statement, 441
CHOOSE= option
MODEL statement (high-performance statistical procedures), 48
CINFO option
PROC HPFMM statement, 137
CL option
MODEL statement (HPFMM), 160
MODEL statement (HPGENSELECT), 241
MODEL statement (HPLMIXED), 299

MODEL statement (HPLOGISTIC), 348
PROBMODEL statement (HPFMM), 169
RANDOM statement (HPLMIXED), 304
CLASS statement
high-performance statistical procedures, 42
HPCANDISC procedure, 85
HPFMM procedure, 156
HPGENSELECT procedure, 237
HPLMIXED procedure, 298, 320
HPLOGISTIC procedure, 344
HPPLS procedure, 444
HPQUANTSELECT procedure, 510
HPREG procedure, 550
CLB option
MODEL statement (HPQUANTSELECT), 511
MODEL statement (HPREG), 552
CODE statement
HPGENSELECT procedure, 237
HPGENSELECT procedure, 237
HPLOGISTIC procedure, 344
HPPRINCOMP procedure, 479
HPQUANTSELECT procedure, 510
HPREG procedure, 550
COMMIT= option
PERFORMANCE statement (high-performance analytical procedures), 36
COMPETITIVE option
SELECTION statement (high-performance statistical procedures), 48
COMPINFO option
PROC HPFMM statement, 137
COMPONENTINFO option
PROC HPFMM statement, 137
CONVF option
PROC HPLMIXED statement, 320
CONVG option
PROC HPLMIXED statement, 320
CONVH option
PROC HPLMIXED statement, 320
COPYVAR= option
OUTPUT statement (HPLOGISTIC), 352
OUTPUT statement (HPQUANTSELECT), 513
OUTPUT statement (HPREG), 553
CORR option
PROC HPFMM statement, 137
PROC HPGENSELECT statement, 233
PROC HPNLMOD statement, 399
COV option
PROC HPFMM statement, 137
PROC HPGENSELECT statement, 233
PROC HPNLMOD statement, 399
PROC HPPRINCOMP statement, 476
COVARIANCE option
PROC HPPRINCOMP statement, 476
COVI option
PROC HPFMM statement, 137
CRIT= option
PROC HPFMM statement, 137
CRITERION statement
HPSPLIT procedure, 601
CRITERION= option
PROC HPFMM statement, 137
SELECTION statement (high-performance statistical procedures), 48
CTABLE option
MODEL statement (HPLOGISTIC), 348
CUTPOINT= option
MODEL statement (HPLOGISTIC), 349
CVTEST option
PROC HPPLS statement, 441

DATA= option
OUTPUT statement (HPGENSELECT), 245
OUTPUT statement (HPLOGISTIC), 352
OUTPUT statement (HPPLS), 445
OUTPUT statement (HPQUANTSELECT), 513
OUTPUT statement (HPREG), 553
PROC HPCANDISC statement, 82
PROC HPFMM statement, 137
PROC HPGENSELECT statement, 233
PROC HPLMIXED statement, 295
PROC HPLOGISTIC statement, 339
PROC HPNLMOD statement, 400
PROC HPPLS statement, 441
PROC HPPRINCOMP statement, 477
PROC HPQUANTSELECT statement, 508
PROC HPREG statement, 549
DDFM= option
MODEL statement (HPLMIXED), 299
MODEL statement (HPLOGISTIC), 349
DESCENDING option
CLASS statement (high-performance statistical procedures), 42
MODEL statement, 159
MODEL statement (HPGENSELECT), 239
MODEL statement (HPLOGISTIC), 346
DETAILS option
PERFORMANCE statement (high-performance analytical procedures), 37
PROC HPPLS statement, 441
DETAILS= option
SELECTION statement (high-performance statistical procedures), 50
DF= option
ESTIMATE statement (HPNLMOD), 404
PREDICT statement (HPNLMOD), 408
PROC HPNLMOD statement, 400
DIAGNOSTICS option

BAYES statement (HPFMM), 148
DISPERSION= option
MODEL statement (HPGENSELECT), 241
DIST= option
MODEL statement (HPFMM), 161
DISTANCE option
PROC HPCANDISC statement, 82
DISTRIBUTION= option
MODEL statement (HPFMM), 161
MODEL statement (HPGENSELECT), 241
DIVISOR= option
RESTRICT statement (HPFMM), 172

ECORR option
PROC HPNLMOD statement, 400
ECOV option
PROC HPNLMOD statement, 400
EPSILON= option
PROC HPPLS statement, METHOD=PLS option, 442
EQCONS= option
PARMS statement (HPLMIXED), 302
EQUATE= option
MODEL statement (HPFMM), 163
ESTIMATE statement
HPNLMOD procedure, 404
ESTIMATE= option
BAYES statement (HPFMM), 150
EVENT= option
MODEL statement, 159
EXCLUDE= option
PROC HPFMM statement, 137
EXCLUSION= option
PROC HPFMM statement, 137

FAST option
SELECTION statement (high-performance statistical procedures), 49
FCONV option
PROC HPFMM statement, 138
PROC HPGENSELECT statement, 234
PROC HPLMIXED statement, 295
PROC HPLOGISTIC statement, 340
PROC HPNLMOD statement, 400
FCONV2 option
PROC HPFMM statement, 138
FITDETAILS option
PROC HPFMM statement, 139
FMTLIBXML= option
PROC HPGENSELECT statement, 234
PROC HPLOGISTIC statement, 340
PROC HPNLMOD statement, 400
PROC HPQUANTSELECT statement, 509
PROC HPREG statement, 549
FRACTION option
HPGENSELECT procedure, PARTITION statement, 248
HPLOGISTIC procedure, PARTITION statement, 354
HPPLS procedure, PARTITION statement, 447
HPQUANTSELECT procedure, PARTITION statement, 515
HPREG procedure, PARTITION statement, 555
HPSPLIT procedure, PARTITION statement, 604
FREQ statement
high-performance statistical procedures, 46
HPCANDISC procedure, 85
HPFMM procedure, 157
HPGENSELECT procedure, 238
HPLOGISTIC procedure, 345
HPPRINCOMP procedure, 480
HPREG procedure, 551
FTOL option
PROC HPFMM statement, 138
PROC HPGENSELECT statement, 234
PROC HPLMIXED statement, 295
PROC HPLOGISTIC statement, 340
PROC HPNLMOD statement, 400
FTOL2 option
PROC HPFMM statement, 138

GCONV option
PROC HPFMM statement, 139
PROC HPGENSELECT statement, 234
PROC HPLMIXED statement, 295
PROC HPLOGISTIC statement, 340
PROC HPNLMOD statement, 400
GRIDHOST= option
PERFORMANCE statement (high-performance analytical procedures), 37
GRIDMODE= option
PERFORMANCE statement (high-performance analytical procedures), 37
GRIDTIMEOUT= option
PERFORMANCE statement (high-performance analytical procedures), 37
GTOL option
PROC HPFMM statement, 139
PROC HPGENSELECT statement, 234
PROC HPLMIXED statement, 295
PROC HPLOGISTIC statement, 340
PROC HPNLMOD statement, 400

HESSIAN option
PROC HPFMM statement, 139
HIERARCHY= option
SELECTION statement (high-performance statistical procedures), 51

high-performance analytical procedures, PERFORMANCE statement, 36
COMMIT= option, 36
DETAILS option, 37
GRIDHOST= option, 37
GRIDMODE= option, 37
GRIDTIMEOUT= option, 37
HOST= option, 37
INSTALL= option, 37
INSTALLLOC= option, 37
LASR= option, 38
LASRSERVER= option, 38
MODE= option, 37
NNODES= option, 38
NODES= option, 38
NTHREADS= option, 39
THREADS= option, 39
TIMEOUT= option, 37
high-performance statistical procedures
FREQ statement, 46
ID statement, 46
VAR statement, 52
WEIGHT statement, 52
high-performance statistical procedures, CLASS statement, 42
DESCENDING option, 42
MISSING option, 45
ORDER= option, 43
PARAM= option, 45
REF= option, 43
TRUNCATE= option, 46
high-performance statistical procedures, FREQ statement, 46
high-performance statistical procedures, ID statement, 46
high-performance statistical procedures, SELECTION statement, 47
ADAPTIVE option, 48
CHOOSE= option, 48
COMPETITIVE option, 48
CRITERION= option, 48
DETAILS= option, 50
HIERARCHY= option, 51
LSCOEFFS option, 49
MAXEFFECTS= option, 49
MAXSTEP= option, 49
METHOD= option, 47
MINEFFECTS= option, 49
SCREEN= option, 51
SELECT= option, 49
SELECTION= option, 51
SLE= option, 49
SLENTRY= option, 49
SLS= option, 50
SLSTAY= option, 50
STOP= option, 50
STOPHORIZON= option, 51
high-performance statistical procedures, VAR statement, 52
high-performance statistical procedures, WEIGHT statement, 52
HOLD= option
PARMS statement (HPLMIXED), 302
HOST= option
PERFORMANCE statement (high-performance analytical procedures), 37
HPCANDISC procedure
PERFORMANCE statement, 85
PROC HPCANDISC statement, 81
syntax, 80
HPCANDISC procedure, BY statement, 84
HPCANDISC procedure, CLASS statement, 85
HPCANDISC procedure, FREQ statement, 85
HPCANDISC procedure, ID statement, 85
HPCANDISC procedure, PERFORMANCE statement, 85
HPCANDISC procedure, PROC HPCANDISC statement, 81
ALL option, 82
ANOVA option, 82
BCORR option, 82
BCOV option, 82
BSSCP option, 82
DATA= option, 82
DISTANCE option, 82
MAHALANOBIS option, 82
NCAN= option, 82
NOPRINT option, 82
OUT= option, 83
OUTSTAT= option, 83
PCORR option, 83
PCOV option, 83
PREFIX= option, 83
PSSCP option, 83
SHORT option, 83
SIMPLE option, 83
SINGULAR= option, 83
STDMEAN option, 84
TCORR option, 84
TCOV option, 84
TSSCP option, 84
WCORR option, 84
WCOV option, 84
WSSCP option, 84
HPCANDISC procedure, VAR statement, 86
HPCANDISC procedure, WEIGHT statement, 86
HPFMM procedure, 134
BAYES statement, 146

FREQ statement, 157
ID statement, 157
MODEL statement, 157
OUTPUT statement, 165
PERFORMANCE statement, 168
PROBMODEL statement, 168
PROC HPFMM statement, 134
RESTRICT statement, 170
syntax, 134
WEIGHT statement, 172
HPFMM procedure, BAYES statement, 146
BETAPRIORPARMS option, 147
DIAGNOSTICS option, 148
ESTIMATE= option, 150
INITIAL= option, 151
METROPOLIS option, 151
MIXPRIORPARMS option, 150
MUPRIORPARMS option, 151
NBI= option, 152
NMC= option, 152
OUTPOST= option, 152
PHIPRIORPARMS option, 152
PRIOROPTIONS option, 153
PRIOROPTS option, 153
STATISTICS option, 154
SUMMARIES option, 154
THIN= option, 155
THINNING= option, 155
TIMEINC= option, 155
HPFMM procedure, BY statement, 156
HPFMM procedure, CLASS statement, 156
UPCASE option, 156
HPFMM procedure, FREQ statement, 157
HPFMM procedure, ID statement, 157
HPFMM procedure, MODEL statement, 157
ALPHA= option, 160
CL option, 160
DESCENDING option, 159
DIST= option, 161
DISTRIBUTION= option, 161
EQUATE= option, 163
EVENT= option, 159
K= option, 163
KMAX= option, 163
KMIN= option, 164
KRESTART option, 164
LABEL= option, 164
LINK= option, 164
NOINT option, 164
NUMBER= option, 163
OFFSET= option, 165
ORDER= option, 159
PARAMETERS option, 165
PARMS option, 165
REFERENCE= option, 160
HPFMM procedure, OUTPUT statement, 165
ALLSTATS option, 167
keyword= option, 165
OUT= option, 165
PREDTYPE option, 167
HPFMM procedure, PERFORMANCE statement, 168
HPFMM procedure, PROBMODEL statement, 168
ALPHA= option, 168
CL option, 169
LINK= option, 169
NOINT option, 169
PARAMETERS option, 169
PARMS option, 169
HPFMM procedure, PROC HPFMM statement, 134
ABSCONV option, 136
ABSFCONV option, 136
ABSFTOL option, 136
ABSGCONV option, 136
ABSGTOL option, 136
ABSTOL option, 136
CINFO option, 137
COMPINFO option, 137
COMPONENTINFO option, 137
CORR option, 137
COV option, 137
COVI option, 137
CRIT= option, 137
CRITERION= option, 137
DATA= option, 137
EXCLUDE= option, 137
EXCLUSION= option, 137
FCONV option, 138
FCONV2 option, 138
FITDETAILS option, 139
FTOL option, 138
FTOL2 option, 138
GCONV option, 139
GTOL option, 139
HESSIAN option, 139
INVALIDLOGL= option, 140
ITDETAILS option, 140
MAXFUNC= option, 140
MAXITER= option, 140
MAXTIME= option, 140
MEMBERSHIP= option, 141
NAMELEN= option, 141
NOCENTER option, 141
NOCLPRINT option, 141
NOITPRINT option, 141
NOPRINT option, 141
PARMSTYLE= option, 141
PARTIAL= option, 141
PLOTS option, 142

SEED= option, 145
SINGCHOL= option, 145
SINGULAR= option, 145
TECHNIQUE= option, 145
ZEROPROB= option, 146
HPFMM procedure, RESTRICT statement, 170
DIVISOR= option, 172
HPFMM procedure, WEIGHT statement, 172
HPGENSELECT procedure, 231
CLASS statement, 237
CODE statement, 237
FREQ statement, 238
ID statement, 238
MODEL statement, 238
OUTPUT statement, 245
PARTITION statement, 248
PERFORMANCE statement, 248
PROC HPGENSELECT statement, 231
SELECTION statement, 249
syntax, 231
WEIGHT statement, 250
ZEROMODEL statement, 250
HPGENSELECT procedure, CLASS statement, 237
UPCASE option, 237
HPGENSELECT procedure, CODE statement, 237
HPGENSELECT procedure, FREQ statement, 238
HPGENSELECT procedure, ID statement, 238
HPGENSELECT procedure, MODEL statement, 238
ALPHA= option, 241
CL option, 241
DESCENDING option, 239
DISPERSION= option, 241
DISTRIBUTION= option, 241
INCLUDE option, 243
INITIALPHI= option, 243
LINK= option, 243
NOCENTER option, 244
NOINT option, 244
OFFSET= option, 244
ORDER= option, 240
SAMPLEFRAC= option, 244
START option, 244
HPGENSELECT procedure, OUTPUT statement, 245
ALPHA= option, 247
DATA= option, 245
keyword= option, 245
OBSCAT option, 247
OUT= option, 245
HPGENSELECT procedure, PARTITION statement, 248
FRACTION option, 248
ROLEVAR= option, 248
HPGENSELECT procedure, PERFORMANCE statement, 248
HPGENSELECT procedure, PROC HPGENSELECT statement, 231
ABSCONV option, 232
ABSFCONV option, 233
ABSFTOL option, 233
ABSGCONV option, 233
ABSGTOL option, 233
ABSTOL option, 232
ALPHA= option, 233
CORR option, 233
COV option, 233
DATA= option, 233
FCONV option, 234
FMTLIBXML= option, 234
FTOL option, 234
GCONV option, 234
GTOL option, 234
ITDETAILS option, 234
ITSELECT option, 235
ITSUMMARY option, 235
MAXFUNC= option, 235
MAXITER= option, 235
MAXTIME= option, 235
NAMELEN= option, 236
NOCLPRINT option, 236
NOPRINT option, 236
NORMALIZE= option, 236
NOSTDERR option, 236
SINGCHOL= option, 236
SINGSWEEP= option, 236
SINGULAR= option, 236
TECHNIQUE= option, 236
HPGENSELECT procedure, SELECTION statement, 249
HPGENSELECT procedure, WEIGHT statement, 250
HPGENSELECT procedure, ZEROMODEL statement, 250
INCLUDE option, 251
START option, 251
HPGENSELECT procedure, CODE statement, 237
HPLMIXED procedure, 292
ID statement, 298
OUTPUT statement, 300
PERFORMANCE statement, 303
PROC HPLMIXED statement, 293
syntax, 292
HPLMIXED procedure, CLASS statement, 298, 320
HPLMIXED procedure, ID statement, 298
HPLMIXED procedure, MODEL statement, 299
CL option, 299
DDFM= option, 299
NOINT option, 300
SOLUTION option, 300
HPLMIXED procedure, OUTPUT statement, 300

ALLSTATS option, 301
OUT= option, 300
HPLMIXED procedure, PARMS statement, 301
EQCONS= option, 302
HOLD= option, 302
LOWERB= option, 302
NOITER option, 302
PARMSDATA= option, 302
PDATA= option, 302
UPPERB= option, 303
HPLMIXED procedure, PERFORMANCE statement, 303
HPLMIXED procedure, PROC HPLMIXED statement
ABSCONV option, 294
ABSFCONV option, 294
ABSFTOL option, 294
ABSGCONV option, 294
ABSGTOL option, 294
ABSOLUTE option, 320
ABSTOL option, 294
BLUP option, 295
CONVF option, 320
CONVG option, 320
CONVH option, 320
DATA= option, 295
FCONV option, 295
FTOL option, 295
GCONV option, 295
GTOL option, 295
MAXCLPRINT= option, 296
MAXFUNC= option, 296
MAXITER= option, 296, 297
MAXTIME= option, 296
METHOD= option, 297
NAMELEN= option, 297
NOCLPRINT option, 297
NOPRINT option, 297
RANKS option, 297
SINGCHOL= option, 297
SINGSWEEP= option, 297
SINGULAR= option, 297
TECHNIQUE= option, 297
XCONV option, 298
XTOL option, 298
HPLMIXED procedure, RANDOM statement, 303
ALPHA= option, 304
CL option, 304
SOLUTION option, 304
SUBJECT= option, 304
TYPE= option, 304
HPLMIXED procedure, REPEATED statement, 309
SUBJECT= option, 310
TYPE= option, 310
HPLMIXEDC procedure, PROC HPLMIXED statement, 293
HPLOGISTIC procedure, 337
CLASS statement, 344
FREQ statement, 345
ID statement, 345
MODEL statement, 345
OUTPUT statement, 352
PARTITION statement, 354
PERFORMANCE statement, 354
PROC HPLOGISTIC statement, 338
SELECTION statement, 355
syntax, 337
WEIGHT statement, 356
HPLOGISTIC procedure, BY statement, 343
HPLOGISTIC procedure, CLASS statement, 344
UPCASE option, 344
HPLOGISTIC procedure, FREQ statement, 345
HPLOGISTIC procedure, ID statement, 345
HPLOGISTIC procedure, MODEL statement, 345
ALPHA= option, 348
ASSOCIATION option, 348
CL option, 348
CTABLE option, 348
CUTPOINT= option, 349
DDFM= option, 349
DESCENDING option, 346
INCLUDE option, 349
LACKFIT option, 349
LINK= option, 350
NOCHECK option, 350
NOINT option, 350
OFFSET= option, 350
ORDER= option, 347
OUTROC= option, 348
PEVENT= option, 350
PRIOR= option, 350
RSQUARE option, 351
START option, 351
HPLOGISTIC procedure, OUTPUT statement, 352
COPYVAR= option, 352
DATA= option, 352
keyword= option, 352
OBSCAT option, 353
OUT= option, 352
HPLOGISTIC procedure, PARTITION statement, 354
FRACTION option, 354
ROLEVAR= option, 354
HPLOGISTIC procedure, PERFORMANCE statement, 354
HPLOGISTIC procedure, PROC HPLOGISTIC statement, 338
ABSCONV option, 339
ABSFCONV option, 339

ABSFTOL option, 339
ABSGCONV option, 339
ABSGTOL option, 339
ABSTOL option, 339
ALPHA= option, 339
DATA= option, 339
FCONV option, 340
FMTLIBXML= option, 340
FTOL option, 340
GCONV option, 340
GTOL option, 340
INEST= option, 340
ITDETAILS option, 341
ITSELECT option, 341
MAXFUNC= option, 341
MAXITER= option, 341, 342
MAXTIME= option, 342
NAMELEN= option, 342
NOCLPRINT option, 342
NOITPRINT option, 342
NOPRINT option, 342
NORMALIZE= option, 342
NOSTDERR option, 342
OUTEST option, 342
SINGCHOL= option, 343
SINGSWEEP= option, 343
SINGULAR= option, 343
TECHNIQUE= option, 343
HPLOGISTIC procedure, SELECTION statement, 355
HPLOGISTIC procedure, WEIGHT statement, 356
HPLOGISTIC procedure, CODE statement, 344
HPNLMOD procedure, 397
PERFORMANCE statement, 407
PROC HPNLMOD statement, 398
syntax, 397
HPNLMOD procedure, BOUNDS statement, 403
HPNLMOD procedure, BY statement, 403
HPNLMOD procedure, ESTIMATE statement, 404
ALPHA= option, 404
DF= option, 404
HPNLMOD procedure, MODEL statement, 404
HPNLMOD procedure, PARAMETERS statement, 405
HPNLMOD procedure, PERFORMANCE statement, 407
HPNLMOD procedure, PREDICT statement, 408
ALPHA= option, 408
DF= option, 408
LOWER= option, 408
PRED= option, 408
PROBT= option, 408
STDERR= option, 408
TVALUE= option, 408
UPPER= option, 408
HPNLMOD procedure, PROC HPNLMOD statement, 398
ABSCONV option, 399
ABSFCONV option, 399
ABSFTOL option, 399
ABSGCONV option, 399
ABSGTOL option, 399
ABSTOL option, 399
ALPHA= option, 399
CORR option, 399
COV option, 399
DATA= option, 400
DF= option, 400
ECORR option, 400
ECOV option, 400
FCONV option, 400
FMTLIBXML= option, 400
FTOL option, 400
GCONV option, 400
GTOL option, 400
MAXFUNC= option, 401
MAXITER= option, 401
MAXTIME= option, 401
NOITPRINT option, 402
NOPRINT option, 402
OUT= option, 402
SINGULAR= option, 402
TECHNIQUE= option, 402
XMLFORMAT= option, 402
HPNLMOD procedure, RESTRICT statement, 409
HPPLS procedure
syntax, 439
HPPLS procedure, BY statement, 443
HPPLS procedure, CLASS statement, 444
UPCASE option, 444
HPPLS procedure, ID statement, 444
HPPLS procedure, MODEL statement, 444
INTERCEPT option, 444
SOLUTION option, 444
HPPLS procedure, OUTPUT statement, 445
DATA= option, 445
keyword option, 445
OUT= option, 445
HPPLS procedure, PARTITION statement, 447
FRACTION option, 447
ROLEVAR= option, 447
HPPLS procedure, PERFORMANCE statement, 448
HPPLS procedure, PROC HPPLS statement, 440
CENSCALE option, 441
CVTEST option, 441
DATA= option, 441
DETAILS option, 441
METHOD= option, 441
NAMELEN= option, 442

NFAC= option, 442
NOCENTER option, 442
NOCLPRINT option, 443
NOCVSTDIZE option, 443
NOPRINT option, 443
NOSCALE option, 443
VARSS option, 443
HPPLS procedure, PROC HPPLS statement, CVTEST option
NSAMP= option, 441
PVAL= option, 441
SEED= option, 441
STAT= option, 441
HPPLS procedure, PROC HPPLS statement, METHOD=PLS option
ALGORITHM= option, 442
EPSILON= option, 442
MAXITER= option, 442
HPPRINCOMP procedure
PERFORMANCE statement, 480
PROC HPPRINCOMP statement, 476
syntax, 475
HPPRINCOMP procedure, BY statement, 479
HPPRINCOMP procedure, CODE statement, 479
HPPRINCOMP procedure, FREQ statement, 480
HPPRINCOMP procedure, ID statement, 480
HPPRINCOMP procedure, PARTIAL statement, 480
HPPRINCOMP procedure, PERFORMANCE statement, 480
HPPRINCOMP procedure, PROC HPPRINCOMP statement, 476
COV option, 476
COVARIANCE option, 476
DATA= option, 477
N= option, 477
NOINT option, 477
NOPRINT option, 477
OUT= option, 477
OUTSTAT= option, 477
PARPREFIX= option, 478
PPREFIX= option, 478
PREFIX= option, 478
RPREFIX= option, 478
SING= option, 478
SINGULAR= option, 478
STANDARD option, 478
STD option, 478
VARDEF= option, 478
HPPRINCOMP procedure, VAR statement, 481
HPPRINCOMP procedure, WEIGHT statement, 481
HPQUANTSELECT procedure
ID statement, 511
MODEL statement, 511
OUTPUT statement, 513
PARTITION statement, 515
PERFORMANCE statement, 515
PROC HPQUANTSELECT statement, 508
WEIGHT statement, 517
HPQUANTSELECT procedure, BY statement, 509
HPQUANTSELECT procedure, CLASS statement, 510
UPCASE option, 510
HPQUANTSELECT procedure, ID statement, 511
HPQUANTSELECT procedure, MODEL statement, 511
CLB option, 511
INCLUDE option, 511
NOINT option, 511
ORDERSELECT option, 512
QUANTILES= option, 512
SPARSITY option, 512
START option, 512
STB option, 512
TOL option, 512
VIF option, 512
HPQUANTSELECT procedure, OUTPUT statement, 513
COPYVAR= option, 513
DATA= option, 513
keyword= option, 513
OUT= option, 513
HPQUANTSELECT procedure, PARTITION statement, 515
FRACTION option, 515
ROLEVAR= option, 515
HPQUANTSELECT procedure, PERFORMANCE statement, 515
HPQUANTSELECT procedure, PROC HPQUANTSELECT statement, 508
ALPHA= option, 508
DATA= option, 508
FMTLIBXML= option, 509
MAXMACRO= option, 509
NAMELEN= option, 509
NOCLPRINT option, 509
NOPRINT option, 509
SEED= option, 509
HPQUANTSELECT procedure, SELECTION statement, 516
TEST= option, 516
HPQUANTSELECT procedure, WEIGHT statement, 517
HPQUANTSELECT procedure, CODE statement, 510
HPREG procedure
FREQ statement, 551
ID statement, 551
MODEL statement, 551
OUTPUT statement, 553

PARTITION statement, 555
PERFORMANCE statement, 555
PROC HPREG statement, 548
WEIGHT statement, 558
HPREG procedure, BY statement, 549
HPREG procedure, CLASS statement, 550
SPLIT option, 44
UPCASE option, 550
HPREG procedure, ID statement, 551
HPREG procedure, MODEL statement, 551
CLB option, 552
INCLUDE option, 552
NOINT option, 552
ORDERSELECT option, 552
START option, 552
STB option, 552
TOL option, 552
VIF option, 553
HPREG procedure, OUTPUT statement, 553
COPYVAR= option, 553
DATA= option, 553
keyword= option, 553
OUT= option, 553
HPREG procedure, PARTITION statement, 555
FRACTION option, 555
ROLEVAR= option, 555
HPREG procedure, PERFORMANCE statement, 555
HPREG procedure, PROC HPREG statement, 548
ALPHA= option, 549
DATA= option, 549
FMTLIBXML= option, 549
NAMELEN= option, 549
NOCLPRINT option, 549
NOPRINT option, 549
SEED= option, 549
HPREG procedure, SELECTION statement, 556
SCREEN option, 556
HPREG procedure, WEIGHT statement, 558
HPREG procedures, FREQ statement, 551
HPREG procedure, CODE statement, 550
HPSPLIT procedure
CRITERION statement, 601
ID statement, 603
INPUT statement, 603
OUTPUT statement, 604
PARTITION statement, 604
PERFORMANCE statement, 605
PROC HPSPLIT statement, 598
PRUNE statement, 606
RULES statement, 601, 608
SCORE statement, 608
TARGET statement, 609
HPSPLIT procedure, CRITERION statement, 601
HPSPLIT procedure, ID statement, 603
HPSPLIT procedure, INPUT statement, 603
HPSPLIT procedure, OUTPUT statement, 604
HPSPLIT procedure, PARTITION statement, 604
FRACTION option, 604
ROLEVAR= option, 605
HPSPLIT procedure, PERFORMANCE statement, 605
HPSPLIT procedure, PROC HPSPLIT statement, 598
HPSPLIT procedure, PRUNE statement, 606
HPSPLIT procedure, RULES statement, 601, 608
HPSPLIT procedure, SCORE statement, 608
HPSPLIT procedure, TARGET statement, 609

ID statement
high-performance statistical procedures, 46
HPCANDISC procedure, 85
HPFMM procedure, 157
HPGENSELECT procedure, 238
HPLMIXED procedure, 298
HPLOGISTIC procedure, 345
HPPLS procedure, 444
HPPRINCOMP procedure, 480
HPQUANTSELECT procedure, 511
HPREG procedure, 551
HPSPLIT procedure, 603
INCLUDE option
MODEL statement (HPGENSELECT), 243
MODEL statement (HPLOGISTIC), 349
MODEL statement (HPQUANTSELECT), 511
MODEL statement (HPREG), 552
ZEROMODEL statement (HPGENSELECT), 251
INEST= option
PROC HPLOGISTIC statement, 340
INITIAL= option
BAYES statement (HPFMM), 151
INITIALPHI= option
MODEL statement (HPGENSELECT), 243
INPUT statement
HPSPLIT procedure, 603
INSTALL= option
PERFORMANCE statement (high-performance analytical procedures), 37
INSTALLLOC= option
PERFORMANCE statement (high-performance analytical procedures), 37
INTERCEPT option
MODEL statement (HPPLS), 444
INVALIDLOGL= option
PROC HPFMM statement, 140
ITDETAILS option
PROC HPFMM statement, 140
PROC HPGENSELECT statement, 234, 235
PROC HPLOGISTIC statement, 341
ITSELECT option
PROC HPGENSELECT statement, 235

PROC HPLOGISTIC statement, 341

K= option
MODEL statement (HPFMM), 163
keyword option
OUTPUT statement (HPPLS), 445
keyword= option
OUTPUT statement (HPFMM), 165
OUTPUT statement (HPGENSELECT), 245
OUTPUT statement (HPLOGISTIC), 352
OUTPUT statement (HPQUANTSELECT), 513
OUTPUT statement (HPREG), 553
KMAX= option
MODEL statement (HPFMM), 163
KMIN= option
MODEL statement (HPFMM), 164
KRESTART option
MODEL statement (HPFMM), 164

LABEL= option
MODEL statement (HPFMM), 164
LACKFIT option
MODEL statement (HPLOGISTIC), 349
LASR= option
PERFORMANCE statement (high-performance analytical procedures), 38
LASRSERVER= option
PERFORMANCE statement (high-performance analytical procedures), 38
LINK= option
MODEL statement (HPFMM), 164
MODEL statement (HPGENSELECT), 243
MODEL statement (HPLOGISTIC), 350
PROBMODEL statement (HPFMM), 169
LOWER= option
PREDICT statement (HPNLMOD), 408
LOWERB= option
PARMS statement (HPLMIXED), 302
LSCOEFFS option
SELECTION statement (high-performance statistical procedures), 49

MAHALANOBIS option
PROC HPCANDISC statement, 82
MAXCLPRINT= option
PROC HPLMIXED statement, 296
MAXEFFECTS= option
SELECTION statement (high-performance statistical procedures), 49
MAXFUNC= option
PROC HPFMM statement, 140
PROC HPGENSELECT statement, 235
PROC HPLMIXED statement, 296
PROC HPLOGISTIC statement, 341
PROC HPNLMOD statement, 401
MAXITER= option
PROC HPFMM statement, 140
PROC HPGENSELECT statement, 235
PROC HPLMIXED statement, 296, 297
PROC HPLOGISTIC statement, 341, 342
PROC HPNLMOD statement, 401
PROC HPPLS statement, METHOD=PLS option, 442
MAXMACRO= option
PROC HPQUANTSELECT statement, 509
MAXSTEPS= option
SELECTION statement (high-performance statistical procedures), 49
MAXTIME= option
PROC HPFMM statement, 140
PROC HPGENSELECT statement, 235
PROC HPLMIXED statement, 296
PROC HPLOGISTIC statement, 342
PROC HPNLMOD statement, 401
MEMBERSHIP= option
PROC HPFMM statement, 141
METHOD= option
PROC HPLMIXED statement, 297
PROC HPPLS statement, 441
SELECTION statement (high-performance statistical procedures), 47
METROPOLIS option
BAYES statement (HPFMM), 151
MINEFFECTS= option
SELECTION statement (high-performance statistical procedures), 49
MISSING option
CLASS statement (high-performance statistical procedures), 45
MIXPRIORPARMS option
BAYES statement (HPFMM), 150
MODE= option
PERFORMANCE statement (high-performance analytical procedures), 37
MODEL statement
HPFMM procedure, 157
HPGENSELECT procedure, 238
HPLMIXED procedure, 299
HPLOGISTIC procedure, 345
HPNLMOD procedure, 404
HPPLS procedure, 444
HPQUANTSELECT procedure, 511
HPREG procedure, 551
MUPRIORPARMS option
BAYES statement (HPFMM), 151

N= option
PROC HPPRINCOMP statement, 477
NAMELEN= option

PROC HPFMM statement, 141
PROC HPGENSELECT statement, 236
PROC HPLMIXED statement, 297
PROC HPLOGISTIC statement, 342
PROC HPPLS statement, 442
PROC HPQUANTSELECT statement, 509
PROC HPREG statement, 549
NBI= option
BAYES statement (HPFMM), 152
NCAN= option
PROC HPCANDISC statement, 82
NFAC= option
PROC HPPLS statement, 442
NMC= option
BAYES statement (HPFMM), 152
NNODES= option
PERFORMANCE statement (high-performance analytical procedures), 38
NOCENTER option
MODEL statement (HPGENSELECT), 244
PROC HPFMM statement, 141
PROC HPPLS statement, 442
NOCLPRINT option
PROC HPFMM statement, 141
PROC HPGENSELECT statement, 236
PROC HPLMIXED statement, 297
PROC HPLOGISTIC statement, 342
PROC HPPLS statement, 443
PROC HPQUANTSELECT statement, 509
PROC HPREG statement, 549
NOCVSTDIZE option
PROC HPPLS statement, 443
NODES= option
PERFORMANCE statement (high-performance analytical procedures), 38
NOINT option
MODEL statement (HPFMM), 164
MODEL statement (HPGENSELECT), 244
MODEL statement (HPLMIXED), 300
MODEL statement (HPLOGISTIC), 350
MODEL statement (HPQUANTSELECT), 511
MODEL statement (HPREG), 552
PROBMODEL statement (HPFMM), 169
PROC HPPRINCOMP statement, 477
NOITER option
PARMS statement (HPLMIXED), 302
NOITPRINT option
PROC HPFMM statement, 141
PROC HPLOGISTIC statement, 342
PROC HPNLMOD statement, 402
NOPRINT option
PROC HPCANDISC statement, 82
PROC HPFMM statement, 141
PROC HPGENSELECT statement, 236
PROC HPLMIXED statement, 297
PROC HPLOGISTIC statement, 342
PROC HPNLMOD statement, 402
PROC HPPLS statement, 443
PROC HPPRINCOMP statement, 477
PROC HPQUANTSELECT statement, 509
PROC HPREG statement, 549
NORMALIZE= option
PROC HPGENSELECT statement, 236
PROC HPLOGISTIC statement, 342
NOSCALE option
PROC HPPLS statement, 443
NOSTDERR option
PROC HPGENSELECT statement, 236
PROC HPLOGISTIC statement, 342
NSAMP= option
PROC HPPLS statement, CVTEST option, 441
NTHREADS= option
PERFORMANCE statement (high-performance analytical procedures), 39
NUMBER= option
MODEL statement (HPFMM), 163

OBSCAT option
OUTPUT statement (HPGENSELECT), 247
OUTPUT statement (HPLOGISTIC), 353
OFFSET= option
MODEL statement (HPFMM), 165
MODEL statement (HPGENSELECT), 244
MODEL statement (HPLOGISTIC), 350
ORDER= option
CLASS statement (high-performance statistical procedures), 43
CLASS statement (HPREG), 44
MODEL statement, 159
MODEL statement (HPGENSELECT), 240
MODEL statement (HPLOGISTIC), 347
ORDERSELECT option
MODEL statement (HPQUANTSELECT), 512
MODEL statement (HPREG), 552
OUT= option
OUTPUT statement (HPFMM), 165
OUTPUT statement (HPGENSELECT), 245
OUTPUT statement (HPLMIXED), 300
OUTPUT statement (HPLOGISTIC), 352
OUTPUT statement (HPPLS), 445
OUTPUT statement (HPQUANTSELECT), 513
OUTPUT statement (HPREG), 553
PROC HPCANDISC statement, 83
PROC HPNLMOD statement, 402
PROC HPPRINCOMP statement, 477
OUTEST option
PROC HPLOGISTIC statement, 342
OUTPOST= option

BAYES statement (HPFMM), 152
OUTPUT statement
HPFMM procedure, 165
HPGENSELECT procedure, 245
HPLMIXED procedure, 300
HPLOGISTIC procedure, 352
HPPLS procedure, 445
HPQUANTSELECT procedure, 513
HPREG procedure, 553
HPSPLIT procedure, 604
OUTROC= option
MODEL statement (HPLOGISTIC), 348
OUTSTAT= option
PROC HPCANDISC statement, 83
PROC HPPRINCOMP statement, 477

PARAM= option
CLASS statement (high-performance statistical procedures), 45
PARAMETERS option
MODEL statement (HPFMM), 165
PROBMODEL statement (HPFMM), 169
PARAMETERS statement
HPNLMOD procedure, 405
PARMS option
MODEL statement (HPFMM), 165
PROBMODEL statement (HPFMM), 169
PARMS statement
HPLMIXED procedure, 301
PARMSDATA= option
PARMS statement (HPLMIXED), 302
PARMSTYLE= option
PROC HPFMM statement, 141
PARPREFIX= option
PROC HPPRINCOMP statement, 478
PARTIAL statement
HPPRINCOMP procedure, 480
PARTIAL= option
PROC HPFMM statement, 141
PARTITION statement
HPGENSELECT procedure, 248
HPLOGISTIC procedure, 354
HPPLS procedure, 447
HPQUANTSELECT procedure, 515
HPREG procedure, 555
HPSPLIT procedure, 604
PCORR option
PROC HPCANDISC statement, 83
PCOV option
PROC HPCANDISC statement, 83
PDATA= option
PARMS statement (HPLMIXED), 302
PERFORMANCE statement
high-performance analytical procedures, 36
HPCANDISC procedure, 85
HPFMM procedure, 168
HPGENSELECT procedure, 248
HPLMIXED procedure, 303
HPLOGISTIC procedure, 354
HPNLMOD procedure, 407
HPPLS procedure, 448
HPPRINCOMP procedure, 480
HPQUANTSELECT procedure, 515
HPREG procedure, 555
HPSPLIT procedure, 605
PEVENT= option
MODEL statement (HPLOGISTIC), 350
PHIPRIORPARMS option
BAYES statement (HPFMM), 152
PLOTS option
PROC HPFMM statement, 142
PPREFIX= option
PROC HPPRINCOMP statement, 478
PRED= option
PREDICT statement (HPNLMOD), 408
PREDICT statement
HPNLMOD procedure, 408
PREDTYPE option
OUTPUT statement (HPFMM), 167
PREFIX= option
PROC HPCANDISC statement, 83
PROC HPPRINCOMP statement, 478
PRIOR= option
MODEL statement (HPLOGISTIC), 350
PRIOROPTIONS option
BAYES statement (HPFMM), 153
PRIOROPTS option
BAYES statement (HPFMM), 153
PROBMODEL statement
HPFMM procedure, 168
PROBT= option
PREDICT statement (HPNLMOD), 408
PROC HPCANDISC statement, *see* HPCANDISC procedure
HPCANDISC procedure, 81
PROC HPFMM procedure, PROC HPFMM statement
SINGRES= option, 145
PROC HPFMM statement
HPFMM procedure, 134
PROC HPGENSELECT statement, *see* HPGENSELECT procedure
PROC HPLMIXED statement, *see* HPLMIXED procedure
HPLMIXED procedure, 293
PROC HPLOGISTIC statement, *see* HPLOGISTIC procedure
PROC HPNLMOD statement, *see* HPNLMOD procedure

HPNLMOD procedure, 398
PROC HPPLS statement, *see* HPPLS procedure
HPPLS procedure, 440
PROC HPPRINCOMP statement, *see* HPPRINCOMP procedure
HPPRINCOMP procedure, 476
PROC HPQUANTSELECT statement, *see* HPQUANTSELECT procedure
HPQUANTSELECT procedure, 508
PROC HPREG statement, *see* HPREG procedure
HPREG procedure, 548
PROC HPSPLIT statement, *see* HPSPLIT procedure
HPSPLIT procedure, 598
PRUNE statement
HPSPLIT procedure, 606
PSSCP option
PROC HPCANDISC statement, 83
PVAL= option
PROC HPPLS statement, CVTEST option, 441

QUANTILES option
MODEL statement (HPQUANTSELECT), 512

RANDOM statement
HPLMIXED procedure, 303
RANKS option
PROC HPLMIXED statement, 297
REF= option
CLASS statement (high-performance statistical procedures), 43
REFERENCE= option
MODEL statement (HPFMM), 160
REPEATED statement
HPLMIXED procedure, 309
RESTRICT statement
HPFMM procedure, 170
HPNLMOD procedure, 409
ROLEVAR= option
HPGENSELECT procedure, PARTITION statement, 248
HPLOGISTIC procedure, PARTITION statement, 354
HPPLS procedure, PARTITION statement, 447
HPQUANTSELECT procedure, PARTITION statement, 515
HPREG procedure, PARTITION statement, 555
HPSPLIT procedure, PARTITION statement, 605
RPREFIX= option
PROC HPPRINCOMP statement, 478
RSQUARE option
MODEL statement (HPLOGISTIC), 351
RULES statement
HPSPLIT procedure, 601, 608

SAMPLEFRAC= option
MODEL statement (HPGENSELECT), 244
SCORE statement
HPSPLIT procedure, 608
SCREEN option
HPREG procedure, SELECTION statement, 556
SCREEN= option
SELECTION statement (high-performance statistical procedures), 51
SEED= option
PROC HPFMM statement, 145
PROC HPPLS statement, CVTEST option, 441
PROC HPQUANTSELECT statement, 509
PROC HPREG statement, 549
SELECT= option
SELECTION statement (high-performance statistical procedures), 49
SELECTION statement
high-performance statistical procedures, 47
HPGENSELECT procedure, 249
HPLOGISTIC procedure, 355
HPQUANTSELECT procedure, 516
HPREG procedure, 556
SELECTION= option
SELECTION statement (high-performance statistical procedures), 51
SHORT option
PROC HPCANDISC statement, 83
SIMPLE option
PROC HPCANDISC statement, 83
SING= option
PROC HPPRINCOMP statement, 478
SINGCHOL= option
PROC HPFMM statement, 145
PROC HPGENSELECT statement, 236
PROC HPLMIXED statement, 297
PROC HPLOGISTIC statement, 343
SINGRES= option
PROC HPFMM statement, 145
SINGSWEEP= option
PROC HPGENSELECT statement, 236
PROC HPLMIXED statement, 297
PROC HPLOGISTIC statement, 343
SINGULAR= option
PROC HPCANDISC statement, 83
PROC HPFMM statement, 145
PROC HPGENSELECT statement, 236
PROC HPLMIXED statement, 297
PROC HPLOGISTIC statement, 343
PROC HPNLMOD statement, 402
PROC HPPRINCOMP statement, 478
SLE= option
SELECTION statement (high-performance statistical procedures), 49
SLENTRY= option

SELECTION statement (high-performance statistical procedures), 49
SLS= option
SELECTION statement (high-performance statistical procedures), 50
SLSTAY= option
SELECTION statement (high-performance statistical procedures), 50
SOLUTION option
MODEL statement (HPLMIXED), 300
MODEL statement (HPPLS), 444
RANDOM statement (HPLMIXED), 304
SPARSITY option
MODEL statement (HPQUANTSELECT), 512
STANDARD option
PROC HPPRINCOMP statement, 478
START option
MODEL statement (HPGENSELECT), 244
MODEL statement (HPLOGISTIC), 351
MODEL statement (HPQUANTSELECT), 512
MODEL statement (HPREG), 552
ZEROMODEL statement (HPGENSELECT), 251
STAT= option
PROC HPPLS statement, CVTEST option, 441
STATISTICS option
BAYES statement (HPFMM), 154
STB option
MODEL statement (HPQUANTSELECT), 512
MODEL statement (HPREG), 552
STD option
PROC HPPRINCOMP statement, 478
STDERR= option
PREDICT statement (HPNLMOD), 408
STDMEAN option
PROC HPCANDISC statement, 84
STOP= option
SELECTION statement (high-performance statistical procedures), 50
STOPHORIZON= option
SELECTION statement (high-performance statistical procedures), 51
SUBJECT= option
RANDOM statement (HPLMIXED), 304
REPEATED statement (HPLMIXED), 310
SUMMARIES option
BAYES statement (HPFMM), 154
syntax
HPFMM procedure, 134
HPGENSELECT procedure, 231
HPLOGISTIC procedure, 337
HPNLMOD procedure, 397

TARGET statement
HPSPLIT procedure, 609
TCORR option
PROC HPCANDISC statement, 84
TCOV option
PROC HPCANDISC statement, 84
TECHNIQUE= option
PROC HPFMM statement, 145
PROC HPGENSELECT statement, 236
PROC HPLMIXED statement, 297
PROC HPLOGISTIC statement, 343
PROC HPNLMOD statement, 402
TEST= option
HPQUANTSELECT procedure, SELECTION statement, 516
THIN= option
BAYES statement (HPFMM), 155
THINNING= option
BAYES statement (HPFMM), 155
THREADS= option
PERFORMANCE statement (high-performance analytical procedures), 39
TIMEINC= option
BAYES statement (HPFMM), 155
TIMEOUT= option
PERFORMANCE statement (high-performance analytical procedures), 37
TOL option
MODEL statement (HPQUANTSELECT), 512
MODEL statement (HPREG), 552
TRUNCATE= option
CLASS statement (high-performance statistical procedures), 46
TSSCP option
PROC HPCANDISC statement, 84
TVALUE= option
PREDICT statement (HPNLMOD), 408
TYPE= option
RANDOM statement (HPLMIXED), 304
REPEATED statement (HPLMIXED), 310

UPCASE option
CLASS statement (HPFMM), 156
CLASS statement (HPGENSELECT), 237
CLASS statement (HPLOGISTIC), 344
CLASS statement (HPPLS), 444
CLASS statement (HPQUANTSELECT), 510
CLASS statement (HPREG), 550
UPPER= option
PREDICT statement (HPNLMOD), 408
UPPERB= option
PARMS statement (HPLMIXED), 303

VAR statement
high-performance statistical procedures, 52
HPCANDISC procedure, 86

HPPRINCOMP procedure, 481
VARDEF= option
PROC HPPRINCOMP statement, 478
VARSS option
PROC HPPLS statement, 443
VIF option
MODEL statement (HPQUANTSELECT), 512
MODEL statement (HPREG), 553

WCORR option
PROC HPCANDISC statement, 84
WCOV option
PROC HPCANDISC statement, 84
WEIGHT statement
high-performance statistical procedures, 52
HPCANDISC procedure, 86
HPFMM procedure, 172
HPGENSELECT procedure, 250
HPLOGISTIC procedure, 356
HPPRINCOMP procedure, 481
HPQUANTSELECT procedure, 517
HPREG procedure, 558
WSSCP option
PROC HPCANDISC statement, 84

XCONV option
PROC HPLMIXED statement, 298
XMLFORMAT= option
PROC HPNLMOD statement, 402
XTOL option
PROC HPLMIXED statement, 298

ZEROMODEL statement
HPGENSELECT procedure, 250
ZEROPROB= option
PROC HPFMM statement, 146