SAS/STAT® 12.3
User's Guide
High-Performance Procedures

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2013. *SAS/STAT® 12.3 User's Guide: High-Performance Procedures*. Cary, NC: SAS Institute Inc.

SAS/STAT® 12.3 User's Guide: High-Performance Procedures

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

July 2013

Contents

Chapter 1. Introduction 1
Chapter 2. Shared Concepts and Topics 5
Chapter 3. Shared Statistical Concepts 39
Chapter 4. The HPGENSELECT Procedure 71
Chapter 5. The HPLOGISTIC Procedure 127
Chapter 6. The HPLMIXED Procedure 185
Chapter 7. The HPNLMOD Procedure 227
Chapter 8. The HPREG Procedure 263
Chapter 9. The HPSPLIT Procedure 309

Subject Index **345**

Syntax Index **353**

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.

HPGENSELECT	Gordon Johnston
HPLMIXED	Tianlin Wang, Biruk Gebramariam
HPLOGISTIC	Robert E. Derr, Oliver Schabenberger
HPNLMOD	Marc Kessler, Oliver Schabenberger
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, Oliver Schabenberger, Charles Shorb, Tom P. Weber
Numerical routines	Georges H. Guirguis

The following people contribute with their leadership and support: Chris Bailey, Tanya Balan, David Pope, Oliver Schabenberger, Renee Sciortino.

Testing

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
Introduction

Contents

Overview of SAS/STAT High-Performance Procedures **1**
About This Book **1**
  Chapter Organization 1
  Typographical Conventions 2
  Options Used in Examples 2
Online Documentation **3**
SAS Technical Support Services **3**

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 all these 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.

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

Chapter 2, “Shared Concepts and Topics,” describes the modes in which SAS/STAT high-performance procedures can execute.

Chapter 3, “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

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

```
options linesize=80 pagesize=500 nonumber nodate;
```

The HTMLBLUE style is used to create the HTML output and graphs that appear in the online documentation. A style template controls stylistic elements such as colors, fonts, and presentation attributes. The style template is specified in the ODS HTML statement as follows:

```
ods html style=HTMLBlue;
```

If you run the examples, your output might be slightly different, because of the SAS System options you use 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 2
Shared Concepts and Topics

Contents

Overview **5**
Processing Modes **6**
Single-Machine Mode 6
Distributed Mode 6
Symmetric and Asymmetric Distributed Modes 7
Controlling the Execution Mode with Environment Variables and Performance Statement Options 7
Determining Single-Machine Mode or Distributed Mode 9
Alongside-the-Database Execution **13**
Alongside-LASR Distributed Execution **16**
Running High-Performance Analytical Procedures Alongside a SAS LASR Analytic Server in Distributed Mode **17**
Starting a SAS LASR Analytic Server Instance 17
Associating a SAS Libref with the SAS LASR Analytic Server Instance 18
Running a High-Performance Analytical Procedure Alongside the SAS LASR Analytic Server Instance 18
Terminating a SAS LASR Analytic Server Instance 19
Alongside-LASR Distributed Execution on a Subset of the Appliance Nodes **19**
Running High-Performance Analytical Procedures in Asymmetric Mode **19**
Running in Symmetric Mode 20
Running in Asymmetric Mode on One Appliance 21
Running in Asymmetric Mode on Distinct Appliances 22
Alongside-HDFS Execution **25**
Alongside-HDFS Execution by Using the SASHDAT Engine 25
Alongside-HDFS Execution by Using the Hadoop Engine 27
Output Data Sets **31**
Working with Formats **32**
PERFORMANCE Statement **34**

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

Distributed mode has several variations:

- Client-data (or local-data) mode: The input data for the analytic task are not stored on the appliance or cluster but are distributed to the distributed computing environment by the SAS High-Performance Analytics infrastructure when the procedure runs.
- Alongside-the-database mode: The data are stored in the distributed database and are read from the DBMS in parallel into a high-performance analytical procedure that runs on the database appliance.

- Alongside-HDFS mode: The data are stored in the Hadoop Distributed File System (HDFS) and are read in parallel from the HDFS. This mode is available if you install the SAS High-Performance Deployment of Hadoop on the appliance or when you configure a Cloudera 4 Hadoop deployment on the appliance to operate with the SAS High-Performance Analytics infrastructure. For more information about installing the SAS High-Performance Deployment of Hadoop, see the *SAS High-Performance Analytics Infrastructure: Installation and Configuration Guide*.
- Alongside-LASR mode: The data are loaded from a SAS LASR Analytic Server that runs on the appliance.

Symmetric and Asymmetric Distributed Modes

SAS high-performance analytical procedures can run alongside the database or alongside HDFS in asymmetric mode. The primary reason for providing the asymmetric mode is to enable you to manage and house data on one appliance (the data appliance) and to run the high-performance analytical procedure on a second appliance (the computing appliance). You can also run in asymmetric mode on a single appliance that functions as both the data appliance and the computing appliance. This enables you to run alongside the database or alongside HDFS, where computations are done on a different set of nodes from the nodes that contain the data. The following subsections provide more details.

Symmetric Mode

When SAS high-performance analytical procedures run in symmetric distributed mode, the data appliance and the computing appliance must be the same appliance. Both the SAS Embedded Process and the high-performance analytical procedures execute in a SAS process that runs on the same hardware where the DBMS process executes. This is called symmetric mode because the number of nodes on which the DBMS executes is the same as the number of nodes on which the high-performance analytical procedures execute. The initial data movement from the DBMS to the high-performance analytical procedure does not cross node boundaries.

Asymmetric Mode

When SAS high-performance analytical procedures run in asymmetric distributed mode, the data appliance and computing appliance are usually distinct appliances. The high-performance analytical procedures execute in a SAS process that runs on the computing appliance. The DBMS and a SAS Embedded Process run on the data appliance. Data are requested by a SAS data feeder that runs on the computing appliance and communicates with the SAS Embedded Process on the data appliance. The SAS Embedded Process transfers the data in parallel to the SAS data feeder that runs on each of the nodes of the computing appliance. This is called asymmetric mode because the number of nodes on the data appliance does not need to be the same as the number of nodes on the computing appliance.

Controlling the Execution Mode with Environment Variables and Performance Statement Options

You control the execution mode by using environment variables or by specifying options in the PERFORMANCE statement 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.
- *data server* identifies the database server on Teradata appliances as defined in the *hosts* file on the client. This data server is the same entry that you usually specify in the SERVER= entry of a LIBNAME statement for Teradata. For more information about specifying LIBNAME statements for Teradata and other engines, see the DBMS-specific section of *SAS/ACCESS for Relational Databases: Reference* for your engine.
- *grid mode* specifies whether the high-performance analytical procedures execute in symmetric or asymmetric mode. Valid values for this variable are `'sym'` for symmetric mode and `'asym'` for asymmetric mode. The default is symmetric mode.

You can set an environment variable directly from the SAS program by using the OPTION SET= command. For example, the following statements define three variables for a Teradata appliance (the grid mode is the default symmetric mode):

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

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

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

The following statements define three variables that are needed to run asymmetrically on a computing appliance.

```
option set=GRIDHOST      ="compute_appliance.sas.com";
option set=GRIDINSTALLLOC="/opt/TKGrid";
option set=GRIDMODE      ="asym";
```

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

```
performance host      ="compute_appliance.sas.com"
            install   ="/opt/TKGrid"
            gridmode  ="asym"
```

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

Specifying a data server is necessary only on Teradata systems when you do not explicitly set the *gridmode* environment variable or specify the GRIDMODE= option in the PERFORMANCE statement. The data server specification depends on the entries in the (client) *hosts* file. The file specifies the server (suffixed by *cop* and a number) and an IP address. For example:

```
myservercop1  33.44.55.66
```

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

The following sets of SAS statements are functionally equivalent:

```
proc 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 2.1 shows the results from the analysis.

Figure 2.1 Results from Logistic Regression in Single-Machine Mode

```
                        The HPLOGISTIC Procedure

                        Performance Information

                  Execution Mode       Single-Machine
                  Number of Threads    4


                           Model Information

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

Figure 2.1 *continued*

```
                           Parameter Estimates

                                   Standard
     Parameter    Estimate            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 on the client.
```

The following statements use 10 nodes (in distributed mode) to analyze the data on the appliance; results appear in Figure 2.2:

```
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 2.2 Results from Logistic Regression in Distributed Mode

```
                         The HPLOGISTIC Procedure

                          Performance Information

               Host Node                        hpa.sas.com
               Execution Mode                   Distributed
               Grid Mode                        Symmetric
               Number of Compute Nodes          10
               Number of Threads per Node       24
```

Figure 2.2 *continued*

```
                              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

                               Standard
      Parameter    Estimate       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 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 2.3 shows the “Performance Information” table. The numeric results are not reproduced here, but they agree with the previous analyses, which are shown in Figure 2.1 and Figure 2.2.

Figure 2.3 Single-Machine Mode Despite Host Specification

```
                    The HPLOGISTIC Procedure

                    Performance Information

              Execution Mode       Single-Machine
              Number of Threads    4
```

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 on the client.
```

In the analysis shown previously in Figure 2.2, 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.

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 co-located with the nodes of the DBMS. Data then pass from the DBMS to the analytic process on each node. Instead of moving across the network and possibly back to the client machine, the data pass locally between the processes on each node of the appliance.

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

In general, when you have a large amount of input data, you can achieve the best performance from 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="hpa.sas.com";
libname applianc greenplm
        server  ="hpa.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 the 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 2.4 shows the results from this analysis. The "Performance Information" table shows that the execution was in distributed mode. In this case the execution was alongside the Greenplum database. The numeric results agree with the previous analyses, which are shown in Figure 2.1 and Figure 2.2.

Figure 2.4 Alongside-the-Database Execution on Greenplum

```
                          The HPLOGISTIC Procedure

                          Performance Information

                 Host Node                       hpa.sas.com
                 Execution Mode                  Distributed
                 Grid Mode                       Symmetric
                 Number of Compute Nodes         8
                 Number of Threads per Node      24


                             Model Information

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

Figure 2.4 *continued*

```
                         Parameter Estimates

                                  Standard
     Parameter     Estimate          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
```

When high-performance analytical procedures execute symmetrically alongside the database, any nonzero specification of the NODES= option in the PERFORMANCE statement is ignored. If the data are read alongside the database, the number of compute nodes is determined by the layout of the database and cannot be modified. In this example, the appliance contains 16 nodes. (See the "Performance Information" table.)

However, when high-performance analytical procedures execute asymmetrically alongside the database, the number of compute nodes that you specify in the PERFORMANCE statement can differ from the number of nodes across which the data are partitioned. For an example, see the section "Running High-Performance Analytical Procedures in Asymmetric Mode" on page 19.

Alongside-LASR Distributed Execution

You can execute high-performance analytical procedures in distributed mode alongside a SAS LASR Analytic Server. When high-performance analytical procedures execute in this mode, the data are preloaded in distributed form in memory that is managed by a LASR Analytic Server. The data on the nodes of the appliance are accessed in parallel in the process that runs the LASR Analytic Server, and they are transferred to the process where the high-performance analytical procedure runs. In general, each high-performance analytical procedure copies the data to memory that persists only while that procedure executes. Hence, when a high-performance analytical procedure runs alongside a LASR Analytic Server, both the high-performance analytical procedure and the LASR Analytic Server have a copy of the subset of the data that is used by the high-performance analytical procedure. The advantage of running high-performance analytical procedures alongside a LASR Analytic Server (as opposed to running alongside a DBMS table or alongside HDFS) is that the initial transfer of data from the LASR Analytic Server to the high-performance analytical procedure is a memory-to-memory operation that is faster than the disk-to-memory operation when the procedure runs alongside a DBMS or HDFS. When the cost of preloading a table into a LASR Analytic Server is amortized by multiple uses of these data in separate runs of high-performance analytical procedures, using the LASR Analytic Server can result in improved performance.

Running 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=12345
          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: Administration 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 19.

Figure 2.5 shows the "Performance Information" table, which shows that the LASR procedure executes in distributed mode on 16 nodes.

Figure 2.5 Performance Information

```
                    The LASR Procedure

                  Performance Information

           Host Node                  hpa.sas.com
           Execution Mode             Distributed
           Grid Mode                  Symmetric
           Number of Compute Nodes    8
```

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=12345;
```

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: Administration 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 2.6.

Figure 2.6 Performance Information

```
                 Performance Information

       Host Node                       hpa.sas.com
       Execution Mode                  Distributed
       Grid Mode                       Symmetric
       Number of Compute Nodes         8
       Number of Threads per Node      24
```

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 alongside 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=12345;
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 "Symmetric and Asymmetric Distributed Modes" on page 7.

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;
```

Because you explicitly specified the GRIDMODE= option, you do not need to also specify the DATASERVER= option in the PERFORMANCE statement. Figure 2.7 shows the results of this analysis.

Figure 2.7 Alongside-the-Database Execution in Symmetric Mode on Teradata

```
                         The HPLOGISTIC Procedure

                         Performance Information

          Host Node                    data_appliance.sas.com
          Execution Mode               Distributed
          Grid Mode                    Symmetric
          Number of Compute Nodes      24
          Number of Threads per Node   24
```

Figure 2.7 *continued*

```
                           Model Information

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


                          Parameter Estimates

                                    Standard
     Parameter    Estimate         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

You can switch to running the HPLOGISTIC procedure in asymmetric mode by specifying the GRID-MODE=ASYM option in the PERFORMANCE statement as follows:

```
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    = asym;
run;
```

Figure 2.8 shows the “Performance Information” table.

Figure 2.8 Alongside Teradata Execution in Asymmetric Mode

```
                      The HPLOGISTIC Procedure

                      Performance Information

        Host Node                      data_appliance.sas.com
        Execution Mode                 Distributed
        Grid Mode                      Asymmetric
        Number of Compute Nodes        10
        Number of Threads per Node     24
```

You can see that now the grid mode is asymmetric. Furthermore, the NODES=10 option that you specified in the PERFORMANCE statement is honored. The data are moved in parallel from the 24 nodes on which the data are stored to the 10 nodes on which the execution occurs. 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 computing appliance that has 15 compute nodes, and it is a different appliance 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"
               gridmode    = asym;
run;
```

Figure 2.9 shows the “Performance Information” table.

Figure 2.9 Asymmetric Mode with Distinct Data and Computing Appliances

```
                    The HPLOGISTIC Procedure

                    Performance Information

          Host Node                       compute_appliance.sas.com
          Execution Mode                  Distributed
          Grid Mode                       Asymmetric
          Number of Compute Nodes         15
          Number of Threads per Node      24
```

PROC HPLOGISTIC ran on the 15 nodes of the computing appliance, even though the data are 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 2.1 and Figure 2.2.

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=54321
          data=dataLib.simData
          path="/tmp/";
   performance host     ="compute_appliance.sas.com"
               gridmode = asym;
run;

libname MyLasr sasiola tag="dataLib" port=54321 host="compute_appliance.sas.com" ;
```

Figure 2.10 show the “Performance Information” table.

Figure 2.10 PROC LASR Running in Asymmetric Mode

```
                       The LASR Procedure

                    Performance Information

            Host Node                    compute_appliance.sas.com
            Execution Mode               Distributed
            Grid Mode                    Asymmetric
            Number of Compute Nodes      15
```

PROC LASR ran in asymmetric mode on the computing appliance, which has 15 compute nodes. In this mode, the data are loaded in parallel from the 24 data appliance nodes to the 15 compute nodes on the computing appliance. By default, all the nodes on the computing appliance are used. You can use the NODES= option in the PERFORMANCE statement to run the LASR Analytic Server on a subset of the nodes on the computing appliance. If you omit the GRIDMODE=ASYM option from the PERFORMANCE statement, PROC LASR still runs successfully but much less efficiently. The Teradata access engine transfers the simData data set to a temporary table on the client, and the High-Performance Analytics infrastructure then transfers these data from the temporary table on the client to the grid nodes on the computing appliance.

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. Because these procedures run on the same computing appliance where the LASR Analytic Server is running, it is best to run these procedures in symmetric mode, which is the default or can be explicitly specified in the GRIDMODE=SYM option in the PERFORMANCE statement. The following statements provide an example. The OUTPUT statement creates an output data set that is held in memory by the LASR Analytic Server. The data appliance has no role in executing these 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 54321. 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. In order for this output to be directly written in parallel from the nodes of the computing appliance to the nodes of the data appliance, you need to run the HPLOGISTIC procedure in asymmetric mode by specifying the GRIDMODE=ASYM option in the PERFORMANCE statement as follows:

```
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"
               gridmode  = asym;
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, it is important to run the high-performance analytical procedures in asymmetric mode so that the Read and Write operations take place in parallel without any movement of data to and from the SAS client. If you omit running the preceding PROC HPLOGISTIC step in asymmetric mode, then the output data set would be created much less efficiently: the output data would be moved sequentially to a temporary table on the client, after which the Teradata access engine sequentially would write this table to the data appliance.

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=54321;
    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 25.

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 co-located with the nodes of the cluster. Data then pass from HDFS to the analytic process on each node of the cluster.

Before you can run a procedure alongside HDFS, you must distribute the data to the cluster. The following statements use the SASHDAT engine to distribute 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: Administration 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.

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

run;
```

Figure 2.11 shows the "Performance Information" table. You see that the procedure ran in distributed mode. The numeric results shown in Figure 2.12 agree with the previous analyses shown in Figure 2.1, Figure 2.2, and Figure 2.4.

Figure 2.11 Alongside-HDFS Execution Performance Information

```
                    Performance Information

          Host Node                         hpa.sas.com
          Execution Mode                    Distributed
          Grid Mode                         Symmetric
          Number of Compute Nodes           206
          Number of Threads per Node        8
```

Figure 2.12 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
```

Figure 2.12 *continued*

```
                         Parameter Estimates

                                   Standard
   Parameter      Estimate            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 XML shows the contents of the file demo.xml that is used in this example:

```
<configuration>
<property>
  <name>fs.default.name</name>
  <value>hdfs://hpa.sas.com:8020</value>
</property>
<property>
  <name>mapred.job.tracker</name>
  <value>hpa.sas.com:8021</value>
</property>
<property>
    <name>dfs.replication</name>
```

```
        <value>1</value>
    </property>
    <property>
        <name>dfs.block.size</name>
        <value>33554432</value>
    </property>
    </configuration>
```

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 analytics 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. However, whenever you use the Hadoop engine, you must execute the analysis in asymmetric mode to cause the execution to occur alongside HDFS.

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

Figure 2.13 shows the "Performance Information" table. You see that the procedure ran asymmetrically in distributed mode. The numeric results shown in Figure 2.14 agree with the previous analyses.

Figure 2.13 Alongside-HDFS Execution by Using the Hadoop Engine

```
                          The HPLOGISTIC Procedure

                          Performance Information

              Host Node                      compute_appliance.sas.com
              Execution Mode                 Distributed
              Grid Mode                      Asymmetric
              Number of Compute Nodes        15
              Number of Threads per Node     24
```

Figure 2.14 Alongside-HDFS Execution by Using the Hadoop Engine

```
                              Model Information

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


                             Parameter Estimates

                                 Standard
             Parameter    Estimate        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
- JavaScript Object Notation (JSON)
- 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 analytics 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 analytics 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 2.15 shows the results of this analysis. You see that the procedure ran asymmetrically in distributed mode. The metadata that you created by using the HDMD procedure have been used successfully in executing this analysis.

Figure 2.15 Alongside-HDFS Execution with CSV Data

```
                        The HPLOGISTIC Procedure

                        Performance Information

        Host Node                        compute_appliance.sas.com
        Execution Mode                   Distributed
        Grid Mode                        Asymmetric
        Number of Compute Nodes          15
        Number of Threads per Node       24


                           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

                                   Standard
   Parameter     Estimate             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
- 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.

DATASERVER=*"name"*

specifies the name of the server on Teradata systems as defined through the *hosts* file and as used in the LIBNAME statement for Teradata. For example, assume that the *hosts* file defines the server for Teradata as follows:

```
myservercop1  33.44.55.66
```

Then a LIBNAME specification would be as follows:

```
libname TDLib teradata server=myserver user= password= database= ;
```

A PERFORMANCE statement to induce running alongside the Teradata server would specify the following:

```
performance dataserver="myserver";
```

The DATASERVER= option is not required if you specify the GRIDMODE=option in the PERFORMANCE statement or if you set the GRIDMODE environment variable.

Specifying the DATASERVER= option overrides the GRIDDATASERVER environment variable.

DETAILS

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

GRIDHOST=*"name"*

HOST=*"name"*

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

GRIDMODE=SYM | ASYM

MODE=SYM | ASYM

specifies whether the high-performance analytical procedure runs in symmetric (SYM) mode or asymmetric (ASYM) mode. The default is GRIDMODE=SYM. For more information about these modes, see the section "Symmetric and Asymmetric Distributed Modes" on page 7.

If this option is specified, it overrides the GRIDMODE environment variable.

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 16 and the *SAS LASR Analytic Server: Administration 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 execute 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 3
Shared Statistical Concepts

Contents

Common Features of SAS High-Performance Statistical Procedures **40**
Syntax Common to SAS High-Performance Statistical Procedures **40**
    CLASS Statement 40
    FREQ Statement 44
    ID Statement 44
    SELECTION Statement 45
    VAR Statement 50
    WEIGHT Statement 50
Levelization of Classification Variables **50**
Specification and Parameterization of Model Effects **52**
    Effect Operators 53
        Bar and At Sign Operators 54
        Colon, Dash, and Double Dash Operators 55
    GLM Parameterization of Classification Variables and Effects 55
        Intercept 56
        Regression Effects 56
        Main Effects 56
        Interaction Effects 57
        Nested Effects 58
        Continuous-Nesting-Class Effects 58
        Continuous-by-Class Effects 59
        General Effects 59
    Reference Parameterization 60
Model Selection **61**
    Methods 61
        Full Model Fitted 61
        Forward Selection 61
        Backward Elimination 63
        Stepwise Selection 64
        Forward-Swap Selection 66
        Least Angle Regression 67
        Lasso Selection 67
        Adaptive Lasso Selection 68
References **69**

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 50.

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*:

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 52.

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 61.

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 only 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 3.1 lists the applicable *method-options* for each of these methods.

Table 3.1 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 3.1, 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.

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 3.2. 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 3.2 apply two different formats to the variable X.

Table 3.2 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
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 3.2 leads to the level assignment in Table 3.3.

Table 3.3 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 3.4 shows how values of the ORDER= option are interpreted.

Table 3.4 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
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 3.5 displays the results of levelizing the values in Table 3.2 when the MISSING option is in effect.

Table 3.5 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 **X** and **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 procedure.

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 50. 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 53.

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 55 and "Reference Parameterization" on page 60.

Effect Operators

Table 3.6 summarizes the operators that are available for selecting and constructing effects. These operators are discussed in the following sections.

Table 3.6 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 3.7 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 3.7 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

Effect	Example	Description
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 3.8 shows some examples of MODEL statements that use various types of effects.

Table 3.8 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 3.9 is an example where β_0 denotes the intercept and A and B are classification variables that have two and three levels, respectively.

Table 3.9 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 3.10).

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 3.10 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 3.11).

Table 3.11 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 3.12).

Table 3.12 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 3.13 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 3.12.

Table 3.13 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 3.14.

Table 3.14 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 61.

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 61.

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:

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

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 61. 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:

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

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,* Second Edition, New York: Springer-Verlag.

Efron, B., Hastie, T., Johnstone, I., 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., Presnell, B., and Turlach, B. (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 4

The HPGENSELECT Procedure

Contents

Overview: HPGENSELECT Procedure **72**

PROC HPGENSELECT Features 72

PROC HPGENSELECT Contrasted with PROC GENMOD 73

Getting Started: HPGENSELECT Procedure **73**

Syntax: HPGENSELECT Procedure **79**

PROC HPGENSELECT Statement 79

CLASS Statement 84

CODE Statement 85

FREQ Statement 85

ID Statement 86

MODEL Statement 86

OUTPUT Statement 92

PERFORMANCE Statement 94

SELECTION Statement 94

WEIGHT Statement 95

ZEROMODEL Statement 96

Details: HPGENSELECT Procedure **97**

Missing Values 97

Exponential Family Distributions 97

Response Distributions 98

Response Probability Distribution Functions 99

Log-Likelihood Functions 102

Computational Method: Multithreading 106

Choosing an Optimization Algorithm 107

First- or Second-Order Algorithms 107

Algorithm Descriptions 109

Displayed Output 111

ODS Table Names 115

Examples: HPGENSELECT Procedure **116**

Example 4.1: Model Selection 116

Example 4.2: Modeling Binomial Data 119

Example 4.3: Tweedie Model 124

References **126**

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 6 in Chapter 2, "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 4.1 through Figure 4.8.

The “Performance Information” table in Figure 4.1 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 4.1 Performance Information

```
                    The HPGENSELECT Procedure

                     Performance Information

               Execution Mode       Single-Machine
               Number of Threads    4
```

Figure 4.2 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 4.2 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 4.3.

Figure 4.3 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 4.4 displays the “Number of Observations” table. All 100 observations in the data set are used in the analysis.

Figure 4.4 Number of Observations

```
Number of Observations Read             100
Number of Observations Used             100
```

Figure 4.5 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 **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 4.5 Dimensions in Poisson Regression

```
                 Dimensions

        Number of Effects              6
        Number of Parameters          16
        Columns in X                  21
```

Figure 4.6 displays the final convergence status of the Newton-Raphson algorithm. The GCONV= relative convergence criterion is satisfied.

Figure 4.6 Convergence Status

```
Convergence criterion (GCONV=1E-8) satisfied.
```

The "Fit Statistics" table is shown in Figure 4.7. 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 4.7 Fit Statistics

```
                  Fit Statistics

     -2 Log Likelihood             290.16169
     AIC  (smaller is better)      322.16169
     AICC (smaller is better)      328.71590
     BIC  (smaller is better)      363.84441
     Pearson Chi-Square             77.76937
     Pearson Chi-Square/DF           0.92583
```

The “Parameter Estimates” table in Figure 4.8 shows that many parameters have fairly large *p*-values, indicating that one or more of the model effects might not be necessary.

Figure 4.8 Parameter Estimates

```
                         Parameter Estimates

                                             Standard
Parameter    DF        Estimate                 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* > **;**
> **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 4.1 summarizes the available options in the PROC HPGENSELECT statement by function. The options are then described fully in alphabetical order.

Table 4.1 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
Output Options	
CORR	Displays the “Parameter Estimates Correlation Matrix” table
COV	Displays the “Parameter Estimates Covariance Matrix” table
ITDETAILS	Displays the “Iteration History” table
ITSELECT	Displays the “Iteration History” table when 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

Table 4.1 *continued*

Option	Description
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:

$$\max_j |g_j(\boldsymbol{\psi}^{(k)})| \le r$$

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–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 6; for information about the alongside-the-database model, see the section "Alongside-the-Database Execution" on page 13.

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:

$$\frac{|f(\boldsymbol{\psi}^{(k)}) - f(\boldsymbol{\psi}^{(k-1)})|}{|f(\boldsymbol{\psi}^{(k-1)})|} \le r$$

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 32 in Chapter 2, "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:

$$\frac{\mathbf{g}(\boldsymbol{\psi}^{(k)})'[\mathbf{H}^{(k)}]^{-1}\mathbf{g}(\boldsymbol{\psi}^{(k)})}{|f(\boldsymbol{\psi}^{(k)})|} \le r$$

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:

$$\frac{\| \mathbf{g}(\boldsymbol{\psi}^{(k)}) \|_2^2 \quad \| \mathbf{s}(\boldsymbol{\psi}^{(k)}) \|_2}{\| \mathbf{g}(\boldsymbol{\psi}^{(k)}) - \mathbf{g}(\boldsymbol{\psi}^{(k-1)}) \|_2 \; |f(\boldsymbol{\psi}^{(k)})|} \le r$$

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.

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 107.

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 40 of Chapter 3, "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 enables you to write 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 4.2 summarizes the *options* available in the CODE statement.

Table 4.2 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 more information about the syntax of the CODE statement, see the section “CODE Statement” (Chapter 19, *SAS/STAT User’s Guide*).

The HPGENSELECT procedure supports the IMPUTE option only for multinomial, binomial, and binary distributions.

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 44 in Chapter 3, “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 52 of Chapter 3, “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 4.3 summarizes these options.

Table 4.3 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

Table 4.3 *continued*

Option	Description
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 4.4 shows the interpretation of the ORDER= option.

Table 4.4 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 4.5.

Table 4.5 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.

P=
specifies 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 4.6. For the binary and multinomial distributions, only the link functions shown in Table 4.6 are available. For the other distributions, you can use any link function shown in Table 4.7 by specifying the LINK= option. Other commonly used link functions for each distribution are shown in Table 4.6.

Table 4.6 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 4.7. Default and commonly used link functions for the available distributions are shown in Table 4.6.

Table 4.7 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 | ADJPEARS | STDRESCHI

requests the Pearson residual, adjusted to have unit variance. The adjusted Pearson residual is defined for the ith 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 ith observation is defined as the ith 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 that has $w_{ei} = \frac{w_i}{\phi \mathrm{V}(\mu_i)(g'(\mu_i))^2}$ as the ith diagonal, and w_i is a prior weight specified by 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.

LINP | XBETA
requests the linear predictor $\eta = \mathbf{x}'\boldsymbol{\beta}$.

LOWER
requests a lower confidence limit for the predicted value. This statistic is not computed for generalized logit multinomial models or zero-modified models.

PEARSON | PEARS | RESCHI
requests the Pearson residual, $\frac{y-\mu}{\sqrt{\mathrm{V}(\mu)}}$, 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 100 and the zero-inflated models defined in the sections "Zero-Inflated Poisson Distribution" on page 102 and "Zero-Inflated Negative Binomial Distribution" on page 102, the distribution variance is used in place of $\mathrm{V}(\mu)$.

This statistic is not computed for multinomial models.

PREDICTED | PRED | P
requests predicted values for the response variable.

PZERO
requests zero-inflation probabilities for zero-inflated models.

RESIDUAL | RESID | R
requests the raw residual, $y - \mu$, where μ is the estimate of the predicted mean. This statistic is not computed for multinomial models.

UPPER
requests an upper confidence limit for the predicted value. This statistic is not computed for generalized logit multinomial models or zero-modified models.

ZBETA
requests the linear predictor for the zeros model in zero-modified models: $\kappa = \mathbf{z}'\boldsymbol{\gamma}$.

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.

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 34 of Chapter 2, "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 45 in Chapter 3, "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:

$$\text{AIC} = -2l + 2p$$

$$\text{AICC} = \begin{cases} -2l + 2pf/(f-p-1) & \text{when } f > p+2 \\ -2l + 2p(p+2) & \text{otherwise} \end{cases}$$

$$\text{BIC} = -2l + p\log(f)$$

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 4.8.

Table 4.8 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 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 99. The expressions for the log-likelihood functions of these distributions are given in the section "Log-Likelihood Functions" on page 102.

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 4.8.

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 101. Evaluation of the Tweedie log-likelihood for model fitting is performed numerically as described in Dunn and Smyth (2005, 2008).

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

$$\begin{aligned}
\eta_i &= \mathbf{x}_i'\boldsymbol{\beta} \\
\kappa_i &= \mathbf{z}_i'\boldsymbol{\gamma} \\
\lambda_i(\boldsymbol{\beta}) &= g^{-1}(\eta_i) \\
\omega_i(\boldsymbol{\gamma}) &= h^{-1}(\kappa_i) \\
l(\mu_i(\boldsymbol{\beta}), \omega_i(\boldsymbol{\gamma}); y_i, w_i) &= \begin{cases} \log[\omega_i + (1-\omega_i)(1+\frac{k}{w_i}\lambda)^{-\frac{1}{k}}] & y_i = 0 \\[2ex] \log(1-\omega_i) + y_i \log\left(\frac{k\lambda}{w_i}\right) \\ -(y_i + \frac{w_i}{k})\log\left(1+\frac{k\lambda}{w_i}\right) \\ +\log\left(\frac{\Gamma(y_i+\frac{w_i}{k})}{\Gamma(y_i+1)\Gamma(\frac{w_i}{k})}\right) & y_i > 0 \end{cases}
\end{aligned}$$

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

$$\begin{aligned}
\eta_i &= \mathbf{x}_i'\boldsymbol{\beta} \\
\kappa_i &= \mathbf{z}_i'\boldsymbol{\gamma} \\
\lambda_i(\boldsymbol{\beta}) &= g^{-1}(\eta_i) \\
\omega_i(\boldsymbol{\gamma}) &= h^{-1}(\kappa_i) \\
l(\mu_i(\boldsymbol{\beta}), \omega_i(\boldsymbol{\gamma}); y_i, w_i) &= \begin{cases} w_i \log[\omega_i + (1-\omega_i)\exp(-\lambda_i)] & y_i = 0 \\[2ex] w_i[\log(1-\omega_i) + y_i\log(\lambda_i) - \lambda_i - \log(y_i!)] & y_i > 0 \end{cases}
\end{aligned}$$

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 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 4.9 shows which derivatives are required for each optimization technique.

Table 4.9 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 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 4.9.

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); 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 $\mathrm{s}^{(k)}$ as the linear combination of the steepest descent or ascent search direction $\mathrm{s}_1^{(k)}$ and a quasi-Newton search direction $\mathrm{s}_2^{(k)}$:

$$\mathrm{s}^{(k)} = \alpha_1 \mathrm{s}_1^{(k)} + \alpha_2 \mathrm{s}_2^{(k)}$$

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 98.

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.

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.

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.

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.

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 4.10.

Table 4.10 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 ITSELECT
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
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

Table 4.10 *continued*

Table Name	Description	Required Statement / Option
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 4.1: Model Selection

The following HPGENSELECT statements examine the same data that is used in the section "Getting Started: HPGENSELECT Procedure" on page 73, 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 73. It contains 100 observations on a count response variable (Y), a continuous variable (Total) to be used in Example 4.3, 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 4.1.1 through Output 4.1.3.

The "Selection Information" table in Output 4.1.1 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 4.1.1 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 4.1.2 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 4.1.2 Selection Summary Information

```
                         The HPGENSELECT Procedure

                             Selection Summary

                             Effect         Number          p
                     Step    Entered    Effects In      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 4.1.3 Selection Details

```
                              Selection Details

                       Effects                     Pr >
Step   Description     In Model   Chi-Square     ChiSq      -2 LogL          AIC

   0   Initial Model          1                            350.193      352.193
   1   C2 entered             2      25.7340    <.0001     324.611      332.611
   2   C5 entered             3      23.0291    <.0001     303.580      317.580
   3   C1 entered             4       7.8328    0.0496     295.263      315.263

                              Selection Details

                       Step         AICC           BIC

                          0      352.234       354.798
                          1      333.032       343.032
                          2      318.798       335.817
                          3      317.735       341.315
```

Output 4.1.4 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 4.7. 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 4.1.4 Fit Statistics

```
                               Fit Statistics

                     -2 Log Likelihood             295.26316
                     AIC  (smaller is better)      315.26316
                     AICC  (smaller is better)     317.73507
                     BIC  (smaller is better)      341.31486
                     Pearson Chi-Square             85.06563
                     Pearson Chi-Square/DF           0.94517
```

The parameter estimates of the selected model are given in Output 4.1.5. 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 4.1.5 Parameter Estimates

```
                              Parameter Estimates

                                               Standard
Parameter     DF          Estimate                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 4.2: 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 4.2.1 shows that the procedure executes in single-machine mode.

Output 4.2.1 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 4.2.2). 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 4.2.2 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 4.2.2 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 4.2.3 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 4.2.3 Dimensions in Binomial Logistic Regression

```
                    Dimensions

          Number of Effects             4
          Number of Parameters          4
          Columns in X                  4
```

Output 4.2.4 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 4.2.4 Fit Statistics

```
                     Fit Statistics

       -2 Log Likelihood              27.95689
       AIC (smaller is better)        35.95689
       AICC (smaller is better)       38.81403
       BIC (smaller is better)        39.73464
       Pearson Chi-Square             13.43503
       Pearson Chi-Square/DF           0.89567
```

The "Parameter Estimates" table in Output 4.2.5 displays the estimates and standard errors of the model effects.

Output 4.2.5 Parameter Estimates

```
                         Parameter Estimates

                                          Standard
Parameter     DF        Estimate            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 4.2.6 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 4.2.6 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 4.2.7 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 4.2.7 Model Information in Binary Model

```
                    The HPGENSELECT Procedure

                     Performance Information

               Execution Mode       Single-Machine
               Number of Threads    4


                        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            Total
                         Value     Y    Frequency

                             1     0          375
                             2     1           12

               You are modeling the probability that Y='1'.
```

Output 4.2.8 displays the parameter estimates. These results match those in Output 4.2.5.

Output 4.2.8 Parameter Estimates

```
                          Parameter Estimates

                                            Standard
   Parameter     DF         Estimate           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 4.3: Tweedie Model

The following HPGENSELECT statements examine the data set getStarted used in the section "Getting Started: HPGENSELECT Procedure" on page 73, 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 "Parameter Estimates" table in Output 4.3.1 shows the resulting regression model parameter estimates, the estimated Tweedie dispersion parameter, and the estimated Tweedie power.

Output 4.3.1 Parameter Estimates

```
                          The HPGENSELECT Procedure

                             Parameter Estimates

                                            Standard
Parameter      DF         Estimate             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 $\mu = g^{-1}(\mathrm{x}'\hat{\boldsymbol{\beta}})$ 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 4.3.2.

```
data Scores;
   set ScoringData;
   %inc 'ScoringParameters.txt';
;
proc print data=Scores;
run;
```

Output 4.3.2 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.

Hurvich, C. M. and Tsai, C.-L. (1989), "Regression and Time Series Model Selection in Small Samples," *Biometrika*, 76, 297–307.

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 5
The HPLOGISTIC Procedure

Contents

Overview: HPLOGISTIC Procedure **128**

PROC HPLOGISTIC Features 128

PROC HPLOGISTIC Contrasted with Other SAS Procedures 129

Getting Started: HPLOGISTIC Procedure **129**

Binary Logistic Regression 129

Syntax: HPLOGISTIC Procedure **136**

PROC HPLOGISTIC Statement 136

BY Statement 141

CODE Statement 142

CLASS Statement 142

FREQ Statement 143

ID Statement 143

MODEL Statement 143

OUTPUT Statement 148

PERFORMANCE Statement 150

SELECTION Statement 150

WEIGHT Statement 151

Details: HPLOGISTIC Procedure **152**

Missing Values 152

Response Distributions 152

Log-Likelihood Functions 153

Existence of Maximum Likelihood Estimates 154

Generalized Coefficient of Determination 155

The Hosmer-Lemeshow Goodness-of-Fit Test 156

Computational Method: Multithreading 157

Choosing an Optimization Algorithm 158

First- or Second-Order Algorithms 158

Algorithm Descriptions 159

Displayed Output 161

ODS Table Names 165

Examples: HPLOGISTIC Procedure **166**

Example 5.1: Model Selection 166

Example 5.2: Modeling Binomial Data 170

Example 5.3: Ordinal Logistic Regression 175

Example 5.4: Conditional Logistic Regression for Matched Pairs Data 178

References **182**

Overview: HPLOGISTIC Procedure

The HPLOGISTIC procedure is a high-performance 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 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

- 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 6 in Chapter 2, "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 40. 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 5.1 through Figure 5.11.

The “Performance Information” table in Figure 5.1 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 5.1 Performance Information

```
                         The HPLOGISTIC Procedure

                         Performance Information

                   Execution Mode       Single-Machine
                   Number of Threads    4
```

Figure 5.2 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 5.2 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 5.3.

Figure 5.3 Class Level Information

```
                         Class Level Information

                 Class     Levels     Values

                 C             10     A B C D E F G H I J
```

Figure 5.4 displays the “Number of Observations” table. All 100 observations in the data set are used in the analysis.

Figure 5.4 Number of Observations

```
Number of Observations Read             100
Number of Observations Used             100
```

The “Response Profile” table in Figure 5.5 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 5.5 Response Profile

```
              Response Profile

            Ordered             Total
              Value    y    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 5.6 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 **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 5.6 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 5.7. The Newton-Raphson algorithm with ridging converged after four iterations, not counting the initial setup iteration.

Figure 5.7 Iteration History

```
                          Iteration History

                                  Objective                        Max
     Iteration     Evaluations     Function          Change     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 5.8 displays the final convergence status of the Newton-Raphson algorithm. The GCONV= relative convergence criterion is satisfied.

Figure 5.8 Convergence Status

```
                 Convergence criterion (GCONV=1E-8) satisfied.
```

The “Fit Statistics” table is shown in Figure 5.9. 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 5.9 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 5.10 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 5.10 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 5.11 shows that many parameters have fairly large *p*-values, indicating that one or more of the model effects might not be necessary.

Figure 5.11 Parameter Estimates

```
                             Parameter Estimates

                                      Standard
     Parameter     Estimate              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* > **;**
  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 5.1 summarizes the available options in the PROC HPLOGISTIC statement by function. The options are then described fully in alphabetical order.

Table 5.1 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

Table 5.1 *continued*

Option	Description
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
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
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 $f(\boldsymbol{\psi}^{(k)}) \le r$, 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:

$$|f(\boldsymbol{\psi}^{(k-1)}) - f(\boldsymbol{\psi}^{(k)})| \le r$$

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:

$$\max_j |g_j(\boldsymbol{\psi}^{(k)})| \le r$$

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=*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 6; for information about the alongside-the-database model, see the section "Alongside-the-Database Execution" on page 13.

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,

$$\frac{|f(\boldsymbol{\psi}^{(k)}) - f(\boldsymbol{\psi}^{(k-1)})|}{|f(\boldsymbol{\psi}^{(k-1)})|} \le r$$

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. See the section "Working with Formats" on page 32 for details about how to generate a XML stream for your formats.

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,

$$\frac{g(\boldsymbol{\psi}^{(k)})'[\mathbf{H}^{(k)}]^{-1}g(\boldsymbol{\psi}^{(k)})}{|f(\boldsymbol{\psi}^{(k)})|} \le r$$

Here, $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization, $f(\cdot)$ is the objective function, and $g(\cdot)$ is the gradient. For the CONGRA technique (where a reliable Hessian estimate $\mathbf{H}$ is not available), the following criterion is used:

$$\frac{\parallel g(\boldsymbol{\psi}^{(k)}) \parallel_2^2 \quad \parallel s(\boldsymbol{\psi}^{(k)}) \parallel_2}{\parallel g(\boldsymbol{\psi}^{(k)}) - g(\boldsymbol{\psi}^{(k-1)}) \parallel_2 |f(\boldsymbol{\psi}^{(k)})|} \le r$$

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. The ITDETAILS option is not available with model selection.

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.

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 158.

BY Statement

BY *variables* **;**

You can specify a BY statement in PROC HPLOGISTIC to obtain separate analyses of observations in groups that are defined by the BY variables. When a BY statement appears, PROC HPLOGISTIC 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 and 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 *SAS Language Reference: Concepts*. For more information about the DATASETS procedure, see *Base SAS Procedures Guide*.

CODE Statement

CODE < *options* > **;**

The CODE statement enables you to write 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 5.2 summarizes the *options* available in the CODE statement.

Table 5.2 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 more information about the syntax of the CODE statement, see the section "CODE Statement" (Chapter 19, *SAS/STAT User's 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 Analytics procedures is documented in the section "CLASS Statement" on page 40 of Chapter 3, "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'.

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 Analytics 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 Analytics 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 Analytics procedures, see the section "ID Statement" on page 44 in Chapter 3, "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 52 of Chapter 3, "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 5.3 summarizes these options.

Table 5.3 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
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
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; 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 (as above) 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 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. 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:

$$\begin{array}{lcl} \text{concordance index } C \text{ (AUC)} & = & (n_c + 0.5n_t)/n \\ \text{Somers' } D \text{ (Gini coefficient)} & = & (n_c - n_d)/n \\ \text{Goodman-Kruskal gamma} & = & (n_c - n_d)/(n_c + n_d) \\ \text{Kendall's tau-}a & = & (n_c - n_d)/(0.5N(N-1)) \end{array}$$

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 156. The concordance index, C, is an estimate of the AUC, which is the area under the receiver operating characteristic (ROC) curve.

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.

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* ≥ 1; by default, *r*=2.

A small p-value suggests that the fitted model is not an adequate model. See the section "The Hosmer-Lemeshow Goodness-of-Fit Test" on page 156 for more information.

LINK=*keyword*

specifies the link function for the model. The *keywords* and the associated link functions are shown in Table 5.4.

Table 5.4 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 154.

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.

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 155.

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 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 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, only variables named by the XBETA and PREDICTED keywords have their values computed; 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. 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. 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 DATAn convention to name 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 $\eta = \mathbf{x}'\boldsymbol{\beta}$.

PREDICTED | PRED | P

requests predicted values (predicted probabilities of events) for the response variable.

RESIDUAL | RESID | R

requests the raw residual, $y - \mu$, where μ is the estimate of the predicted event probability. This statistic is not computed for multinomial models.

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). This statistic is not computed for multinomial models.

You can specify the following *options* in the OUTPUT statement after the slash (/):

COPYVAR=*variable*

COPYVAR=(*variables***)**

transfers one or more *variables* from the input data set to the output data set.

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.

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 Analytics procedures is documented in the section "PERFORMANCE Statement" on page 34

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 defined by model selection methods. The statement is fully documented in the section "SELECTION Statement" on page 45 in Chapter 3, "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. 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 only effect-selection criterion supported by the HPLOGISTIC 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 HPLOGISTIC procedure calculates an approximate chi-square score test statistic.

The default criterion for the CHOOSE= and STOP= options in the SELECT statement is the significance level of the score test. The following criteria can be specified:

AIC	Akaike's information criterion (Akaike 1974)
AICC	a small-sample bias corrected version of Akaike's information criterion as promoted in, for example, Hurvich and Tsai (1989) and Burnham and Anderson (1998)
BIC | SBC	Schwarz' Bayesian criterion (Schwarz 1978)
SL	the significance level of the score test (STOP= only)

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:

$$\mathrm{AIC} = -2l + 2p$$

$$\mathrm{AICC} = \begin{cases} -2l + 2pf/(f - p - 1) & \text{when } f > p + 2 \\ -2l + 2p(p + 2) & \text{otherwise} \end{cases}$$

$$\mathrm{BIC} = -2l + p\log(f)$$

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 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 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 detailed listing of all candidates at each step and their ranking in terms of the significance level 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 the approximate chi-square score 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

$$\begin{aligned}\Pr(Y = 1) &= p \\ \Pr(Y = 0) &= 1 - p\end{aligned}$$

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, since 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. 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

$$\begin{aligned} \eta_i &= \mathbf{x}_i'\boldsymbol{\beta} \\ \mu_i(\boldsymbol{\beta}) &= g^{-1}(\eta_i) \\ l(\mu_i(\boldsymbol{\beta}); y_i, w_i) &= w_i\,(y_i \log\{\mu_i\} + (n_i - y_i)\log\{1-\mu_i\}) \\ &\quad + w_i\,(\log\{\Gamma(n_i+1)\} - \log\{\Gamma(y_i+1)\} - \log\{\Gamma(n_i - y_i + 1)\}) \end{aligned}$$

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.

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 with ordered response categories, the probabilities are expressed in cumulative form, so that the last category is redundant. In generalized logit models (multinomial models with unordered categories), one category is chosen as the reference category and the linear predictor in the reference category is set to zero.

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.

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 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 that achieves an upper limit of 1, by dividing R^2 by its maximum value,

$$R^2_{\text{max}} = 1 - \{L(\mathbf{0})\}^{\frac{2}{n}}$$

If you specify the RSQUARE option in the MODEL statement, the HPLOGISTIC procedure computes R^2 and the rescaled coefficient of determination according to Nagelkerke:

$$\tilde{R}^2 = \frac{R^2}{R^2_{\text{max}}}$$

The R^2 and $\tilde{R}^2$ 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.

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 5.5 shows which derivatives are required for each optimization technique.

Table 5.5 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 5.5.

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 $\mathrm{s}^{(k)}$ as the linear combination of the steepest descent or ascent search direction $\mathrm{s}_1^{(k)}$ and a quasi-Newton search direction $\mathrm{s}_2^{(k)}$:

$$\mathrm{s}^{(k)} = \alpha_1 \mathrm{s}_1^{(k)} + \alpha_2 \mathrm{s}_2^{(k)}$$

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 152 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.

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.

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 *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.

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.

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 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.

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.

The values displayed in the "Fit Statistics" table are not based on a normalized log-likelihood function.

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

The "Partition for the Hosmer and Lemeshow Test" table displays the grouping used in the Hosmer-Lemeshow test. This table is displayed if you specify the LACKFIT option in the MODEL statement. See the section "The Hosmer-Lemeshow Goodness-of-Fit Test" on page 156 for details, and see Hosmer and Lemeshow (2000) for examples of using this partition.

Hosmer and Lemeshow Goodness-of-Fit Test

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. This table is displayed if you specify the LACKFIT option in the MODEL statement. See the section "The Hosmer-Lemeshow Goodness-of-Fit Test" on page 156 for further details.

Association Statistics

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. This table is displayed if you specify the ASSOCIATION option in the MODEL statement.

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 5.6.

Table 5.6 ODS Tables Produced by PROC HPLOGISTIC

Table Name	**Description**	**Required 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
ClassLevels	Level information from the CLASS statement	CLASS
ConvergenceStatus	Status of optimization at conclusion of optimization	Default output
Dimensions	Model dimensions	Default output
FitStatistics	Fit statistics	Default output
GlobalTests	Test of the model versus the null model	Default output
IterHistory	Iteration history	Default output or PROC HPLOGISTIC 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	Default output
NObs	Number of observations read and used, and number of events and trials, if applicable	Default output
ParameterEstimates	Solutions for the parameter estimates associated with effects in MODEL statements	Default output

Table 5.6 *continued*

Table Name	Description	Required Statement / Option
PerformanceInfo	Information about the high-performance computing environment	Default output
ResponseProfile	Response categories and category modeled in models for binary and multinomial data	Default output
SelectedEffects	List of effects selected into 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

Examples: HPLOGISTIC Procedure

Example 5.1: Model Selection

The following HPLOGISTIC statements examine the same data as in the section "Getting Started: HPLOGISTIC Procedure" on page 129, 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 5.1.1 through Output 5.1.4.

The "Selection Information" table in Output 5.1.1 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 5.1.1 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 5.1.2 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 5.1.2 Selection Summary Information

```
                            Selection Summary

                            Effect         Number         p
                    Step    Entered    Effects In     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 5.1.3 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 5.1.3 Candidate Details

```
                 Candidate Entry and Removal Details

                                        Candidate          p
          Step      Rank      Effect    For            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 5.1.4 Selection Details

```
                               Selection Details

       Effect                Number                   Pr >
Step   Entered            Effects In   Chi-Square    ChiSq      -2 LogL          AIC

   0   Initial Model               1                            123.820      125.820
   1   x8                          2       4.2986   0.0381      119.462      123.462
   2   x2                          3       4.9882   0.0255      114.396      120.396

                              Selection Details

                       Step          AICC           BIC

                          0       125.861       128.425
                          1       123.586       128.672
                          2       120.646       128.212
```

Output 5.1.5 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 5.9. 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 5.1.5 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 5.1.6. 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 5.1.6 Parameter Estimates

```
                            Parameter Estimates

                                 Standard
        Parameter    Estimate       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 5.2: 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 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;
   id Obsnum;
   output out=Out xbeta predicted=Pred;
run;
```

The “Performance Information” table in Output 5.2.1 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 5.2.1 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 5.2.2). 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 5.2.2 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 5.2.2 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 5.2.3 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 5.2.3 Iteration History and Convergence Status

```
                          Iteration History

                                 Objective                       Max
  Iteration    Evaluations        Function          Change      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 5.2.4 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 5.2.4 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 5.2.5 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 5.2.5 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 5.2.6 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 5.2.6 Null Test

```
              Testing Global Null Hypothesis: BETA=0

      Test                 Chi-Square       DF     Pr > ChiSq

      Likelihood Ratio        11.7663        3         0.0082
```

The "Parameter Estimates" table in Output 5.2.7 displays the estimates and standard errors of the model effects.

Output 5.2.7 Parameter Estimates

```
                         Parameter Estimates

                                Standard
       Parameter    Estimate       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 5.2.8 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 5.2.8 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 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 5.2.9 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 5.2.9 Model Information in Binary Model

```
                    The HPLOGISTIC Procedure

                    Performance Information

              Execution Mode        Single-Machine
              Number of Threads     4


                       Model Information

     Data Source                WORKS.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              Total
                      Value     y    Frequency

                          1     0          375
                          2     1           12

          You are modeling the probability that y='1'.
```

Output 5.2.10 displays the result for the test of the global null hypothesis and the parameter estimates. These results match those in Output 5.2.6 and Output 5.2.7.

Output 5.2.10 Null Test and Parameter Estimates

```
                Testing Global Null Hypothesis: BETA=0

        Test                    Chi-Square       DF     Pr > ChiSq

        Likelihood Ratio           11.7663        3         0.0082


                          Parameter Estimates

                                    Standard
   Parameter     Estimate          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 5.3: 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 5.3.1 through Output 5.3.3.

The "Response Profile" table in Output 5.3.1 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 5.3.1 Proportional Odds Model Regression Analysis

```
                  Multiple Response Cheese Tasting Experiment

                            The HPLOGISTIC Procedure

                            Performance Information

                      Execution Mode       Single-Machine
                      Number of Threads    4


                               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

                                      Reference
                  Class       Levels    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
```

Output 5.3.1 *continued*

```
                         Response Profile

                    Ordered     Taste         Total
                      Value     Rating    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 5.3.2 Proportional Odds Model Regression Analysis

```
                           Iteration History

                                   Objective                      Max
     Iteration    Evaluations       Function         Change    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
```

Output 5.3.2 *continued*

```
                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 5.3.3 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 5.3.3 Proportional Odds Model Regression Analysis

```
                           Parameter Estimates

             Taste                 Standard
Parameter    Rating    Estimate       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
```

Example 5.4: Conditional Logistic Regression for Matched Pairs Data

In matched pairs (*case-control*) studies, conditional logistic regression is used to investigate the relationship between an outcome of being an event (case) or a non-event (control) and a set of prognostic factors.

The following data are a subset of the data from the Los Angeles Study of the Endometrial Cancer Data in Breslow and Day (1980). There are 63 matched pairs, each consisting of a case of endometrial cancer (Outcome=1) and a control (Outcome=0). The case and corresponding control have the same ID. Two prognostic factors are included: Gall (an indicator variable for gall bladder disease) and Hyper (an indicator variable for hypertension). The goal of the case-control analysis is to determine the relative risk for gall bladder disease, controlling for the effect of hypertension.

```
data Data1;
  do ID=1 to 63;
    do Outcome = 1 to 0 by -1;
      input Gall Hyper @@;
      output;
    end;
  end;
  datalines;
0 0  0 0    0 0  0 0    0 1  0 1    0 0  1 0    1 0  0 1
0 1  0 0    1 0  0 0    1 1  0 1    0 0  0 0    0 0  0 0
1 0  0 0    0 0  0 1    1 0  0 1    1 0  1 0    1 0  0 1
0 1  0 0    0 0  1 1    0 0  1 1    0 0  0 1    0 1  0 0
0 0  1 1    0 1  0 1    0 1  0 0    0 0  0 0    0 0  0 0
0 0  0 1    1 0  0 1    0 0  0 1    1 0  0 0    0 1  0 0
0 1  0 0    0 1  0 0    0 1  0 0    0 0  0 0    1 1  1 1
0 0  0 1    0 1  0 0    0 1  0 1    0 1  0 1    0 1  0 0
0 0  0 0    0 1  1 0    0 0  0 1    0 0  0 0    1 0  0 0
0 0  0 0    1 1  0 0    0 1  0 0    0 0  0 0    0 1  0 1
0 0  0 0    0 1  0 1    0 1  0 0    0 1  0 0    1 0  0 0
0 0  0 0    1 1  1 0    0 0  0 0    0 0  0 0    1 1  0 0
1 0  1 0    0 1  0 0    1 0  0 0
;
```

When each matched set consists of one event and one non-event, the conditional likelihood is given by

$$\prod_i (1 + \exp(-(\boldsymbol{x}_{i1} - \boldsymbol{x}_{i0})'\boldsymbol{\psi})^{-1}$$

where $\boldsymbol{x}_{i1}$ and $\boldsymbol{x}_{i0}$ are vectors that represent the prognostic factors for the event and non-event, respectively, of the ith matched set. This likelihood is identical to the likelihood of fitting a logistic regression model to a set of data with constant response, where the model contains no intercept term and has explanatory variables given by $\boldsymbol{d}_i = \boldsymbol{x}_{i1} - \boldsymbol{x}_{i0}$ (Breslow 1982).

To apply this method, the following DATA step transforms each matched pair into a single observation, where the variables Gall and Hyper contain the differences between the corresponding values for the case and the control (case – control). The variable Outcome, which is used as the response variable in the logistic regression model, is given a constant value of 0 (which is the Outcome value for the control, although any constant, numeric or character, suffices).

```
data Data2;
   set Data1;
   drop id1 gall1 hyper1;
   retain id1 gall1 hyper1 0;
   if (ID = id1) then do;
      Gall=gall1-Gall; Hyper=hyper1-Hyper;
      output;
   end;
   else do;
      id1=ID; gall1=Gall; hyper1=Hyper;
   end;
run;
```

Note that there are 63 observations in the data set, one for each matched pair. Since the number of observations (n) is halved, statistics that depend on n such as R^2 will be incorrect. The variable Outcome has a constant value of 0.

In the following statements, PROC HPLOGISTIC is invoked with the NOINT option to obtain the conditional logistic model estimates. Because the option CL is specified, PROC HPLOGISTIC computes a 95% confidence interval for the parameter.

```
proc hplogistic data=Data2;
   model outcome=Gall / noint cl;
run;
```

Results from the conditional logistic analysis are shown in Output 5.4.1 through Output 5.4.3.

Output 5.4.1 shows that you are fitting a binary logistic regression where the response variable Outcome has only one level.

Output 5.4.1 Conditional Logistic Regression (Gall as Risk Factor)

```
                Multiple Response Cheese Tasting Experiment

                          The HPLOGISTIC Procedure

                          Performance Information

                    Execution Mode       Single-Machine
                    Number of Threads    4


                             Model Information

           Data Source                 WORK.DATA2
           Response Variable           Outcome
           Distribution                Binary
           Link Function               Logit
           Optimization Technique      Newton-Raphson with Ridging


                Number of Observations Read              63
                Number of Observations Used              63


                              Response Profile

                         Ordered                   Total
                           Value    Outcome    Frequency

                               1    0                 63

              You are modeling the probability that Outcome='0'.
```

Output 5.4.2 shows that the model is marginally significant (*p*=0.0550).

Output 5.4.2 Conditional Logistic Regression (Gall as Risk Factor)

```
                                Iteration History

                                   Objective                          Max
     Iteration     Evaluations      Function           Change     Gradient

             0               4    0.6662698453       .            0.015669
             1               2    0.6639330101      0.00233684    0.001351
             2               2    0.6639171997      0.00001581     6.88E-6
             3               2    0.6639171993      0.00000000    1.83E-10


                  Convergence criterion (GCONV=1E-8) satisfied.


                                  Dimensions

                     Columns in X                          1
                     Number of Effects                     1
                     Max Effect Columns                    1
                     Rank of Cross-product Matrix          1
                     Parameters in Optimization            1


                                Fit Statistics

                    -2 Log Likelihood                 83.6536
                    AIC (smaller is better)           85.6536
                    AICC (smaller is better)          85.7191
                    BIC (smaller is better)           87.7967


                     Testing Global Null Hypothesis: BETA=0

             Test                 Chi-Square       DF     Pr > ChiSq

             Likelihood Ratio         3.6830        1         0.0550
```

Note that there is no intercept term in the "Parameter Estimates" table in Output 5.4.3. The intercepts have been *conditioned out* of the analysis.

Output 5.4.3 Conditional Logistic Regression (Gall as Risk Factor)

```
                          Parameter Estimates

                             Standard
Parameter     Estimate          Error        DF     t Value     Pr > |t|      Alpha

Gall            0.9555         0.5262     Infty        1.82       0.0694       0.05

                          Parameter Estimates

                   Parameter        Lower         Upper

                   Gall          -0.07589        1.9869
```

The odds ratio estimate for Gall is $\exp(0.9555) = 2.60$, which is marginally significant (p=0.0694) and which is an estimate of the relative risk for gall bladder disease. A subject who has gall bladder disease has 2.6 times the odds of having endometrial cancer as a subject who does not have gall bladder disease. A 95% confidence interval for this relative risk, produced by exponentiating the confidence interval for the parameter, is (0.927, 7.293).

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.

Breslow, N. E. (1982), "Covariance Adjustment of Relative-Risk Estimates in Matched Studies," *Biometrics*, 38, 661–672.

Breslow, N. E. and Day, N. E. (1980), *The Analysis of Case-Control Studies*, Statistical Methods in Cancer Research, IARC Scientific Publications, vol. 1, no. 32, Lyon: International Agency for Research on Cancer.

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.

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.

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.

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.

Moré, J. J. and Sorensen, D. C. (1983), “Computing a Trust-Region Step,” *SIAM Journal on Scientific and Statistical Computing*, 4, 553–572.

Nagelkerke, N. J. D. (1991), “A Note on a General Definition of the Coefficient of Determination,” *Biometrika*, 78, 691–692.

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.

Chapter 6
The HPLMIXED Procedure

Contents

Overview: HPLMIXED Procedure **186**
PROC HPLMIXED Features 187
Notation for the Mixed Model 188
PROC HPLMIXED Contrasted with Other SAS Procedures 189
Getting Started: HPLMIXED Procedure **189**
Mixed Model Analysis of Covariance with Many Groups 189
Syntax: HPLMIXED Procedure **192**
PROC HPLMIXED Statement 193
CLASS Statement 198
MODEL Statement 198
PARMS Statement 199
PERFORMANCE Statement 201
RANDOM Statement 201
REPEATED Statement 208
Details: HPLMIXED Procedure **209**
Linear Mixed Models Theory 209
Matrix Notation 209
Formulation of the Mixed Model 210
Estimating Covariance Parameters in the Mixed Model 214
Estimating Fixed and Random Effects in the Mixed Model 215
Statistical Properties 216
Computational Method 217
Distributed Computing 217
Multithreading 217
Displayed Output 218
Performance Information 218
Model Information 218
Class Level Information 218
Dimensions 218
Number of Observations 218
Optimization Information 219
Iteration History 219
Convergence Status 220
Covariance Parameter Estimates 220
Fit Statistics 220
Timing Information 221

ODS Table Names 221
Examples: HPLMIXED Procedure **221**
Example 6.1: Computing BLUPs for a Large Number of Subjects 221
References **224**

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 MIXED 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 6 in Chapter 2, "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 221.

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 209.

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

$$\mathbf{V} = \mathbf{ZGZ}' + \mathbf{R}$$

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 6.1.

Figure 6.1 Mixed Model Analysis of Covariance

```
                          The HPLMIXED Procedure

                         Performance Information

            Host Node                        rdgrd0001.unx.sas.com
            Execution Mode                   Distributed
            Number of Compute Nodes          20
            Number of Threads per Node       8
```

Figure 6.1 *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

                                     Objective                          Max
       Iteration     Evaluations      Function          Change     Gradient

               0               2    225641.67142         .           2.135E-8


               Convergence criterion (ABSGCONV=0.00001) satisfied.
```

Figure 6.1 *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

                                 Standard
    Effect        Estimate          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* ;
  MODEL *dependent* **=** < *fixed-effects* > < / *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 6.1 summarizes the basic functions and important options of the PROC HPLMIXED statements. The syntax of each statement in Table 6.1 is described in the following sections in alphabetical order after the description of the PROC HPLMIXED statement.

Table 6.1 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
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.
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 6.2 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 6.2 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
BLUP	Computes the best linear unbiased prediction

Table 6.2 *continued*

Option	Description
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
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 $f(\boldsymbol{\psi}^{(k)}) \le r$, 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:

$$|f(\boldsymbol{\psi}^{(k-1)}) - f(\boldsymbol{\psi}^{(k)})| \le r$$

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:

$$\max_j |g_j(\boldsymbol{\psi}^{(k)})| \le r$$

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.

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,

$$\frac{|f(\boldsymbol{\psi}^{(k)}) - f(\boldsymbol{\psi}^{(k-1)})|}{|f(\boldsymbol{\psi}^{(k-1)})|} \le r$$

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,

$$\frac{\mathrm{g}(\boldsymbol{\psi}^{(k)})'[\mathbf{H}^{(k)}]^{-1}\mathrm{g}(\boldsymbol{\psi}^{(k)})}{|f(\boldsymbol{\psi}^{(k)})|} \le r$$

Here, $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization, $f(\cdot)$ is the objective function, and $\mathrm{g}(\cdot)$ is the gradient. For the CONGRA technique (where a reliable Hessian estimate $\mathbf{H}$ is not available), the following criterion is used:

$$\frac{\parallel \mathrm{g}(\boldsymbol{\psi}^{(k)}) \parallel_2^2 \quad \parallel \mathrm{s}(\boldsymbol{\psi}^{(k)}) \parallel_2}{\parallel \mathrm{g}(\boldsymbol{\psi}^{(k)}) - \mathrm{g}(\boldsymbol{\psi}^{(k-1)}) \parallel_2 |f(\boldsymbol{\psi}^{(k)})|} \le r$$

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.

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 50 of Chapter 3, "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.

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 52 of Chapter 3, "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 6.3 summarizes options in the MODEL statement. These are subsequently discussed in detail in alphabetical order.

Table 6.3 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

$$p = \Pr\{F_{l,\infty} > F_{obs}\} = \Pr\{\chi^2_l > l F_{obs}\}$$

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 209, 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.

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 **G** matrix to be positive definite in order to avoid the more computationally intensive algorithms that would be required when **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.

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 34 of Chapter 2, “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 209, 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 6.4 summarizes important options in the RANDOM statement. All options are subsequently discussed in alphabetical order.

Table 6.4 Summary of Important RANDOM Statement Options

Option	**Description**
Construction of Covariance Structure	
SUBJECT=	Identifies the subjects in the model
TYPE=	Specifies the covariance structure
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 209, 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 $\widehat{\gamma}_i - \gamma_i$, where $\widehat{\gamma}_i$ is the ith EBLUP and γ_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 **G**. Valid values for *covariance-structure* and their descriptions are listed in Table 6.5. 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 6.5 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	$\sigma^2 \rho^{\|i-j\|}$
ARH(1)	Heterogeneous AR(1)	$t+1$	$\sigma_i \sigma_j \rho^{\|i-j\|}$
ARMA(1,1)	Autoregressive moving average(1,1)	3	$\sigma^2[\gamma\rho^{\|i-j\|-1} 1(i \neq j) + 1(i=j)]$
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	$\sigma_{\|i-j\|+1}$
TOEP(q)	Banded Toeplitz	q	$\sigma_{\|i-j\|+1} 1(\|i-j\| < q)$
TOEPH	Heterogeneous TOEP	$2t-1$	$\sigma_i \sigma_j \rho_{\|i-j\|}$
TOEPH(q)	Banded heterogeneous TOEP	$t+q-1$	$\sigma_i \sigma_j \rho_{\|i-j\|} 1(\|i-j\| < q)$
UN	Unstructured	$t(t+1)/2$	σ_{ij}
UN(q)	Banded	$\frac{q}{2}(2t-q+1)$	$\sigma_{ij} 1(\|i-j\| < q)$

Table 6.5 *continued*

Structure	Description	Parms	(i,j) element
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 6.5, 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 6.6 lists some examples of the structures in Table 6.5.

Table 6.6 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	$\begin{bmatrix} \sigma^2 & \sigma_1 & \sigma_2 & \sigma_3 \\ \sigma_1 & \sigma^2 & \sigma_1 & \sigma_2 \\ \sigma_2 & \sigma_1 & \sigma^2 & \sigma_1 \\ \sigma_3 & \sigma_2 & \sigma_1 & \sigma^2 \end{bmatrix}$

Table 6.6 *continued*

Description	Structure	Example
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	$\begin{bmatrix} \sigma_1^2 & \sigma_1\sigma_2\rho & \sigma_1\sigma_3\rho & \sigma_1\sigma_4\rho \\ \sigma_2\sigma_1\rho & \sigma_2^2 & \sigma_2\sigma_3\rho & \sigma_2\sigma_4\rho \\ \sigma_3\sigma_1\rho & \sigma_3\sigma_2\rho & \sigma_3^2 & \sigma_3\sigma_4\rho \\ \sigma_4\sigma_1\rho & \sigma_4\sigma_2\rho & \sigma_4\sigma_3\rho & \sigma_4^2 \end{bmatrix}$
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 6.5, 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 6.5, 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 6.6 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 6.5, 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 $\mathbf{\Lambda\Lambda}' + \mathbf{D}$, 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 6.5, 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 **R** matrix in the mixed model. If no REPEATED statement is specified, **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 **R** that correspond to the observations with the same level. This results in a singular **R** matrix and an infinite likelihood.

Table 6.7 summarizes important options in the REPEATED statement. All options are subsequently discussed in alphabetical order.

Table 6.7 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 **R** with identical blocks. When the SUBJECT= effect consists entirely of classification variables, the blocks of **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 **R** matrix with dimensions equal to the number of observations.

TYPE=*covariance-structure*

specifies the covariance structure of the **R** matrix. The SUBJECT= option defines the blocks of **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 Brown and Prescott (1999).

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:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \ldots & x_{1p} \\ x_{21} & x_{22} & \ldots & x_{2p} \\ \vdots & \vdots & & \vdots \\ x_{n1} & x_{n2} & \ldots & x_{np} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

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 **Z** and **G** matrices and let the **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 **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 **R** matrix: TYPE=CS specifies the compound symmetry structure, and SUBJECT=INDIV specifies the blocks of **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 **Z** matrix has $3s$ rows and s columns, and **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 **X**, **Z**, and **G** have the following form:

$$
\mathbf{X} = \begin{bmatrix}
1 & 1 & & & 1 & & 1 & & & & & \\
1 & 1 & & & & 1 & & 1 & & & & \\
1 & & 1 & & 1 & & & & 1 & & & \\
1 & & 1 & & & 1 & & & & 1 & & \\
1 & & & 1 & 1 & & & & & & 1 & \\
1 & & & 1 & & 1 & & & & & & 1 \\
\vdots & & \vdots & & \vdots & & & & & \vdots & & \\
1 & 1 & & & 1 & & 1 & & & & & \\
1 & 1 & & & & 1 & & 1 & & & & \\
1 & & 1 & & 1 & & & & 1 & & & \\
1 & & 1 & & & 1 & & & & 1 & & \\
1 & & & 1 & 1 & & & & & & 1 & \\
1 & & & 1 & & 1 & & & & & & 1
\end{bmatrix}
$$

$$
\mathbf{Z} = \begin{bmatrix}
1 & & & & 1 & & & & & & & & & & & \\
1 & & & & 1 & & & & & & & & & & & \\
1 & & & & & 1 & & & & & & & & & & \\
1 & & & & & 1 & & & & & & & & & & \\
1 & & & & & & 1 & & & & & & & & & \\
1 & & & & & & 1 & & & & & & & & & \\
 & 1 & & & & & & 1 & & & & & & & & \\
 & 1 & & & & & & 1 & & & & & & & & \\
 & 1 & & & & & & & 1 & & & & & & & \\
 & 1 & & & & & & & 1 & & & & & & & \\
 & 1 & & & & & & & & 1 & & & & & & \\
 & 1 & & & & & & & & 1 & & & & & & \\
 & & 1 & & & & & & & & 1 & & & & & \\
 & & 1 & & & & & & & & 1 & & & & & \\
 & & 1 & & & & & & & & & 1 & & & & \\
 & & 1 & & & & & & & & & 1 & & & & \\
 & & 1 & & & & & & & & & & 1 & & & \\
 & & 1 & & & & & & & & & & 1 & & & \\
 & & & 1 & & & & & & & & & & 1 & & \\
 & & & 1 & & & & & & & & & & 1 & & \\
 & & & 1 & & & & & & & & & & & 1 & \\
 & & & 1 & & & & & & & & & & & 1 & \\
 & & & 1 & & & & & & & & & & & & 1 \\
 & & & 1 & & & & & & & & & & & & 1
\end{bmatrix}
$$

$$
\mathbf{G} = \begin{bmatrix}
\sigma_B^2 & & & & & & & \\
 & \sigma_B^2 & & & & & & \\
 & & \sigma_B^2 & & & & & \\
 & & & \sigma_B^2 & & & & \\
 & & & & \sigma_{AB}^2 & & & \\
 & & & & & \sigma_{AB}^2 & & \\
 & & & & & & \ddots & \\
 & & & & & & & \sigma_{AB}^2
\end{bmatrix}
$$

where σ_B^2 is the variance component for Block and σ_{AB}^2 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;
```

$$\mathbf{Z} = \begin{bmatrix}
1 & 1 & & & & & & & & & & & & & & \\
1 & 1 & & & & & & & & & & & & & & \\
1 & & 1 & & & & & & & & & & & & & \\
1 & & 1 & & & & & & & & & & & & & \\
1 & & & 1 & & & & & & & & & & & & \\
1 & & & 1 & & & & & & & & & & & & \\
 & & & & 1 & 1 & & & & & & & & & & \\
 & & & & 1 & 1 & & & & & & & & & & \\
 & & & & 1 & & 1 & & & & & & & & & \\
 & & & & 1 & & 1 & & & & & & & & & \\
 & & & & 1 & & & 1 & & & & & & & & \\
 & & & & 1 & & & 1 & & & & & & & & \\
 & & & & & & & & 1 & 1 & & & & & & \\
 & & & & & & & & 1 & 1 & & & & & & \\
 & & & & & & & & 1 & & 1 & & & & & \\
 & & & & & & & & 1 & & 1 & & & & & \\
 & & & & & & & & 1 & & & 1 & & & & \\
 & & & & & & & & 1 & & & 1 & & & & \\
 & & & & & & & & & & & & 1 & 1 & & \\
 & & & & & & & & & & & & 1 & 1 & & \\
 & & & & & & & & & & & & 1 & & 1 & \\
 & & & & & & & & & & & & 1 & & 1 & \\
 & & & & & & & & & & & & 1 & & & 1 \\
 & & & & & & & & & & & & 1 & & & 1
\end{bmatrix}$$

$$\mathbf{G} = \begin{bmatrix}
\sigma_B^2 & & & & & & & & \\
 & \sigma_{AB}^2 & & & & & & & \\
 & & \sigma_{AB}^2 & & & & & & \\
 & & & \sigma_{AB}^2 & & & & & \\
 & & & & \ddots & & & & \\
 & & & & & \sigma_B^2 & & & \\
 & & & & & & \sigma_{AB}^2 & & \\
 & & & & & & & \sigma_{AB}^2 & \\
 & & & & & & & & \sigma_{AB}^2
\end{bmatrix}$$

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 $\boldsymbol{\gamma}$, **G**, and **R** . Least squares is no longer the best method. Generalized least squares (GLS) is more appropriate, minimizing

$$(\mathrm{y} - \mathbf{X}\boldsymbol{\beta})'\mathrm{V}^{-1}(\mathrm{y} - \mathbf{X}\boldsymbol{\beta})$$

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 **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

$$\begin{aligned} \widehat{\mathbf{C}}_{11} &= (\mathbf{X}'\widehat{\mathbf{V}}^{-1}\mathbf{X})^{-} \\ \widehat{\mathbf{C}}_{21} &= -\widehat{\mathbf{G}}\mathbf{Z}'\widehat{\mathbf{V}}^{-1}\mathbf{X}\widehat{\mathbf{C}}_{11} \\ \widehat{\mathbf{C}}_{22} &= (\mathbf{Z}'\widehat{\mathbf{R}}^{-1}\mathbf{Z} + \widehat{\mathbf{G}}^{-1})^{-1} - \widehat{\mathbf{C}}_{21}\mathbf{X}'\widehat{\mathbf{V}}^{-1}\mathbf{Z}\widehat{\mathbf{G}} \end{aligned}$$

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 and Watts 1988). 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 **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 **G** and **R**. (See the section "Estimating Covariance Parameters in the Mixed Model" on page 214.) 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 214. If the log likelihood is an extremely large number, then PROC HPLMIXED has deemed the estimated **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 6.8.

Table 6.8 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 6.9.

Table 6.9 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
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 6.1: 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

$$
\begin{aligned}
\text{Var(Center)} &= 1.7564 \\
\text{Cov(Patient)} &= \begin{bmatrix} 11.4555 & 3.6883 & 4.5951 \\ 3.6883 & 11.2071 & 3.6311 \\ 4.5951 & 3.6311 & 12.1050 \end{bmatrix}
\end{aligned}
$$

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 6.1.1. 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 6.1.1 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 Transaction on Automatic Control*, AC–19, 716–723.

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.

Brown, H. and Prescott, R. (1999), *Applied Mixed Models in Medicine*, New York: John Wiley & Sons.

Carlin, B. P. and Louis, T. A. (1996), *Bayes and Empirical Bayes Methods for Data Analysis,* London: Chapman and 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*, Ser. B., 39, 1–38.

Fai, A. H. T. and Cornelius, P. L. (1996), "Approximate *F*-tests of Multiple Degree of Freedom Hypotheses in Generalized Least Squares Analyses of Unbalanced Split-Plot Experiments," *Journal of Statistical Computation and Simulation,* 54, 363–378.

Fuller, W. A. (1976), *Introduction to Statistical Time Series,* New York: John Wiley & Sons.

Giesbrecht, F. G. and Burns, J. C. (1985), "Two-Stage Analysis Based on a Mixed Model: Large-sample Asymptotic Theory and Small-Sample Simulation Results," *Biometrics*, 41, 477–486.

Hartley, H. O. and Rao, J. N. K. (1967), "Maximum-Likelihood Estimation for the Mixed Analysis of Variance Model," *Biometrika*, 54, 93–108.

Hurvich, C. M. and Tsai, C.-L. (1989), "Regression and Time Series Model Selection in Small Samples," *Biometrika*, 76, 297–307.

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," *Proceedings of the Statistical Computing Section*, American Statistical Association, New Orleans, 41–49.

Harville, D. A. (1990), "BLUP (Best Linear Unbiased Prediction), and Beyond," in *Advances in Statistical Methods for Genetic Improvement of Livestock*, Springer-Verlag, 239–276.

Henderson, C. R. (1984), *Applications of Linear Models in Animal Breeding*, University of Guelph.

Henderson, C. R. (1990), "Statistical Method in Animal Improvement: Historical Overview," in *Advances in Statistical Methods for Genetic Improvement of Livestock*, New York: Springer-Verlag, 1–14.

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., and Stram, D. (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,* Second 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(9), 2683–2699.

McLean, R. A. and Sanders, W. L. (1988), "Approximating Degrees of Freedom for Standard Errors in Mixed Linear Models," *Proceedings of the Statistical Computing Section*, American Statistical Association, New Orleans, 50–59.

McLean, R. A., Sanders, W. L., and Stroup, W. W. (1991), "A Unified Approach to Mixed Linear Models," *The American Statistician*, 45, 54–64.

Milliken, G. A. and Johnson, D. E. (1992), *Analysis of Messy Data, Volume 1: Designed Experiments*, New York: Chapman and 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.

Schluchter, M. D. and Elashoff, J. D. (1990), "Small-Sample Adjustments to Tests with Unbalanced Repeated Measures Assuming Several Covariance Structures," *Journal of Statistical Computation and Simulation*, 37, 69–87.

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*, Ames: Iowa State University Press.

Steel, R. G. D., Torrie, J. H., and Dickey D. (1997), *Principles and Procedures of Statistics: A Biometrical Approach*, Third Edition, New York: McGraw-Hill, Inc.

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*, Second Edition, New York: McGraw-Hill, Inc.

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(6), 1294–1310.

Chapter 7
The HPNLMOD Procedure

Contents

Overview: HPNLMOD Procedure **227**

PROC HPNLMOD Features 228

PROC HPNLMOD Contrasted with the NLIN and NLMIXED Procedures 228

Getting Started: HPNLMOD Procedure **229**

Least Squares Model 229

Binomial Model 230

Syntax: HPNLMOD Procedure **231**

PROC HPNLMOD Statement 232

BOUNDS Statement 237

BY Statement 237

ESTIMATE Statement 238

MODEL Statement 238

PARAMETERS Statement 239

PERFORMANCE Statement 241

PREDICT Statement 242

RESTRICT Statement 243

Programming Statements 243

Details: HPNLMOD Procedure **245**

Least Squares Estimation 245

Built-In Log-Likelihood Functions 246

Computational Method 248

Choosing an Optimization Algorithm 249

Displayed Output 252

ODS Table Names 255

Examples: HPNLMOD Procedure **256**

Example 7.1: Segmented Model 256

References **260**

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 6 in Chapter 2, “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 7.1. The table displays the degrees of freedom, sums of squares, and mean squares along with the model F test.

Figure 7.1 Nonlinear Least Squares Analysis of Variance

```
                          The HPNLMOD Procedure

                          Analysis of Variance

                                      Sum of        Mean                 Approx
Source                       DF      Squares      Square    F Value    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 7.2 displays the parameter estimates, standard errors, *t* statistics, and 95% confidence intervals for θ_1 and θ_2.

Figure 7.2 Parameter Estimates and Approximate 95% Confidence Intervals

```
                           Parameter Estimates

                         Standard                     Approx     Approximate 95%
Parameter    Estimate       Error    DF   t Value    Pr > |t|    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 63.3 (*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 229. 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 7.3.

Figure 7.3 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 229 are displayed in Figure 7.4.

Figure 7.4 Parameter Estimates and Approximate 95% Confidence Intervals

```
                           Parameter Estimates

                        Standard                        Approx      Approximate 95%
  Parameter   Estimate     Error    DF   t Value   Pr > |t|    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 7.1 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 7.1 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 6; for more information about the alongside-the-database model, see the section "Alongside-the-Database Execution" on page 13.

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 $r = 10^{-\text{FDIGITS}}$, 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 32.

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:

$$\frac{\| \mathrm{g}(\psi^{(k)}) \|_2^2 \quad \| \mathrm{s}(\psi^{(k)}) \|_2}{\| \mathrm{g}(\psi^{(k)}) - \mathrm{g}(\psi^{(k-1)}) \|_2 \, |f(\psi^{(k)})|} \leq r$$

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 31.

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 32.

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 in PROC HPNLMOD to obtain separate analyses of observations in groups that are defined by the BY variables. When a BY statement appears, PROC HPNLMOD 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 and 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 *SAS Language Reference: Concepts*. For more information about the DATASETS procedure, see *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 246 .

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 240.

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 34 of Chapter 2, "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 31.

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, informaton 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:

$Y \sim \mathrm{normal}(m, v)$

$$\begin{aligned} l(m, v; y) &= -\frac{1}{2}\left(\log\{2\pi\} + \frac{(y-m)^2}{v} + \log\{v\}\right) \\ \mathrm{E}[Y] &= m \\ \mathrm{Var}[Y] &= v \\ v &> 0 \end{aligned}$$

$Y \sim \mathrm{binary}(p)$

$$\begin{aligned} l_1(p; y) &= \begin{cases} y \ \log\{p\} & y > 0 \\ 0 & \text{otherwise} \end{cases} \\ l_2(p; y) &= \begin{cases} (1-y) \ \log\{1-p\} & y < 1 \\ 0 & \text{otherwise} \end{cases} \\ l(p; y) &= l_1(p; y) + l_2(p; y) \\ \mathrm{E}[Y] &= p \\ \mathrm{Var}[Y] &= p\,(1-p) \\ 0 < p &< 1 \end{aligned}$$

$Y \sim \mathrm{binomial}(n, p)$

$$\begin{aligned} l_c &= \log\{\Gamma(n+1)\} - \log\{\Gamma(y+1)\} - \log\{\Gamma(n-y+1)\} \\ l_1(n, p; y) &= \begin{cases} y \ \log\{p\} & y > 0 \\ 0 & \text{otherwise} \end{cases} \\ l_2(n, p; y) &= \begin{cases} (n-y) \ \log\{1-p\} & n-y > 0 \\ 0 & \text{otherwise} \end{cases} \\ l(n, p; y) &= l_c + l_1(n, p; y) + l_2(n, p; y) \\ \mathrm{E}[Y] &= n\,p \\ \mathrm{Var}[Y] &= n\,p\,(1-p) \\ 0 < p &< 1 \end{aligned}$$

$Y \sim \text{gamma}(a, b)$

$$\begin{aligned} l(a,b;y) &= -a\log\{b\} - \log\{\Gamma(a)\} + (a-1)\log\{y\} - y/b \\ \text{E}[Y] &= ab \\ \text{Var}[Y] &= ab^2 \\ a &> 0 \\ b &> 0 \end{aligned}$$

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 parallelizing 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 7.3 shows which derivatives are required for each optimization technique.

Table 7.3 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 mutli-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 7.3.

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 $\mathrm{s}^{(k)}$ as the linear combination of the steepest descent or ascent search direction $\mathrm{s}_1^{(k)}$ and a quasi-Newton search direction $\mathrm{s}_2^{(k)}$:

$$\mathrm{s}^{(k)} = \alpha_1 \mathrm{s}_1^{(k)} + \alpha_2 \mathrm{s}_2^{(k)}$$

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:

$$\text{AIC} = -2l + 2p$$

$$\text{AICC} = \begin{cases} -2l + 2pn/(n-p-1) & f > p+2 \\ -2l + 2p(p+2) & \text{otherwise} \end{cases}$$

$$\text{BIC} = -2l + p\log(f)$$

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 7.4.

Table 7.4 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

Table 7.4 *continued*

Table Name	Description	Required Statement and Option
ECorrB	Additional estimates' correlation matrix	ECORR option in the PROC HPNLMOD statement
ECovB	Additional estimates' covariance matrix	ECOV option in the PROC HPNLMOD statement
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 7.1: 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

$$\begin{aligned} x_0 &= -\beta/2\gamma \\ c &= \alpha - \beta^2/4\gamma \end{aligned}$$

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 7.1.1 through Output 7.1.3. The iterative optimization converges after six iterations (Output 7.1.1). Output 7.1.2 shows the estimated parameters. Output 7.1.3 indicates that the join point is 12.7477 and the plateau value is 0.7775.

Output 7.1.1 Nonlinear Least Squares Iterative Phase

```
                        Quadratic Model with Plateau

                           The HPNLMOD Procedure

                             Iteration History

                                  Objective                         Max
     Iteration     Evaluations     Function           Change     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 7.1.2 Least Squares Analysis for the Quadratic Model

```
                           Analysis of Variance

                                   Sum of         Mean                 Approx
   Source                   DF    Squares       Square     F Value    Pr > F

   Model                     2     0.1769       0.0884      114.22    <.0001
   Error                    13     0.0101     0.000774
   Corrected Total          15     0.1869


                            Parameter Estimates

                           Standard                    Approx     Approximate 95%
   Parameter   Estimate       Error      DF  t Value   Pr > |t|    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
```

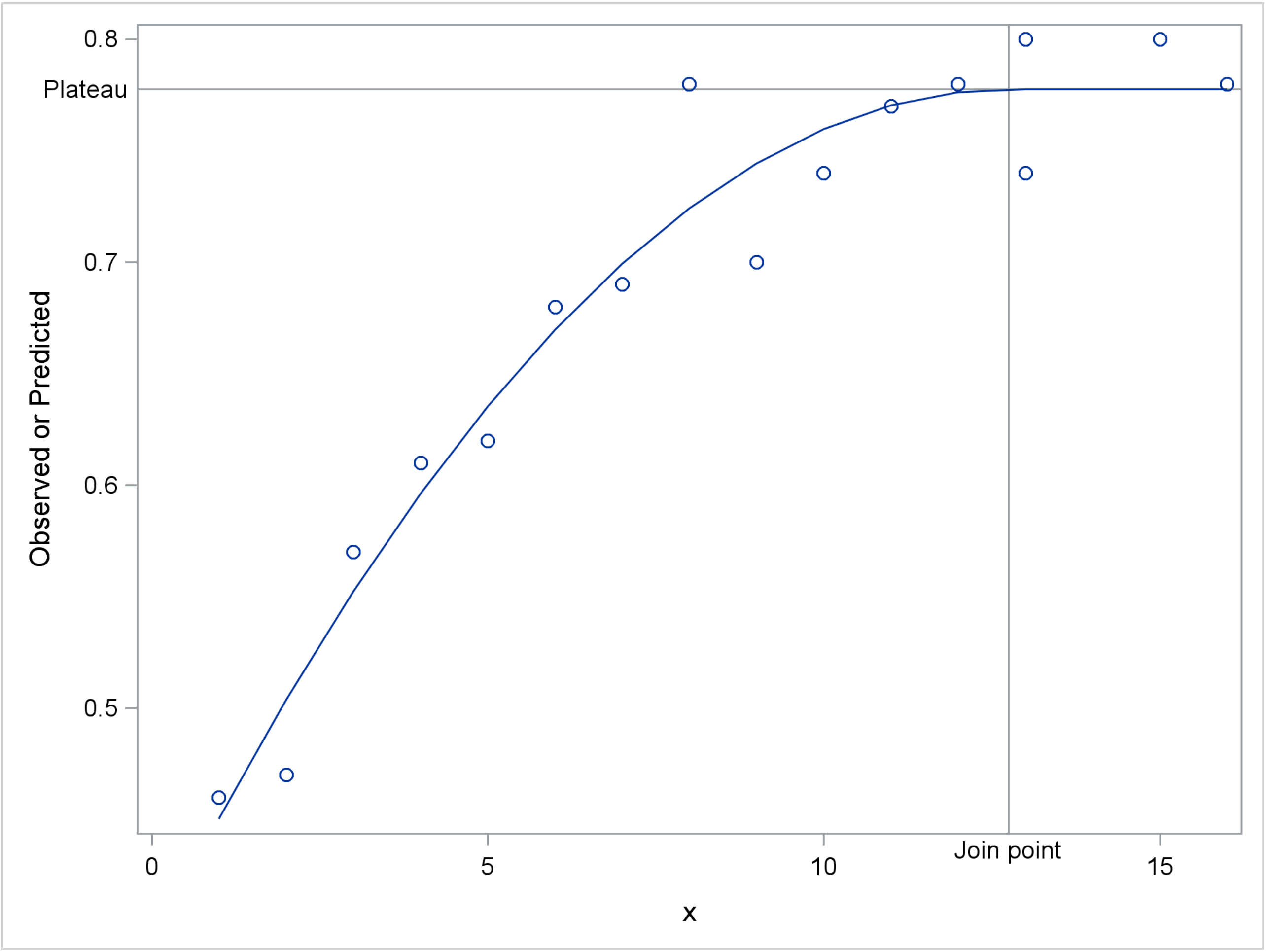
Output 7.1.3 Additional Estimates for the Quadratic Model

```
                              Additional Estimates

                                Standard                            Approx
Label                Estimate      Error       DF     t Value     Pr > |t|      Alpha

join point            12.7477     1.2781        1        9.97       <.0001       0.05
plateau value c        0.7775     0.0123        1       63.11       <.0001       0.05

                             Additional Estimates

                    Label               Approximate Confidence Limits

                    join point             9.9864     15.5089
                    plateau value c        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 7.1.4):

```
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 7.1.4 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 8
The HPREG Procedure

Contents

Overview: HPREG Procedure **264**

PROC HPREG Features 264

PROC HPREG Contrasted with Other SAS Procedures 265

Getting Started: HPREG Procedure **266**

Syntax: HPREG Procedure **272**

PROC HPREG Statement 273

BY Statement 274

CODE Statement 275

CLASS Statement 275

FREQ Statement 276

ID Statement 276

MODEL Statement 276

PARTITION Statement 278

PERFORMANCE Statement 278

SELECTION Statement 278

OUTPUT Statement 279

WEIGHT Statement 281

Details: HPREG Procedure **282**

Criteria Used in Model Selection 282

Diagnostic Statistics 283

Classification Variables and the SPLIT Option 284

Using Validation and Test Data 286

Computational Method 287

Output Data Set 288

Displayed Output 289

ODS Table Names 293

Examples: HPREG Procedure **294**

Example 8.1: Model Selection with Validation 294

Example 8.2: Backward Selection in Single-Machine and Distributed Modes 300

Example 8.3: Forward-Swap Selection 304

References **307**

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 61 in Chapter 3, "Shared Statistical Concepts."

FORWARD	The forward selection method starts with no effects in the model and adds effects.
BACKWARD	The backward elimination method starts with all effects in the model and deletes effects.
STEPWISE	The stepwise regression method is similar to the FORWARD method except that effects already in the model do not necessarily stay there.
FORWARDSWAP	The forward swap selection method is a modification of forward selection where before any addition step, all pairwise swaps of effects in and out of the current model that improve the selection criterion are made. When the selection criterion is R square, this method coincides with the MAXR method in the REG procedure in SAS/STAT software.
LAR	The least angle regression method, 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	The lasso method 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 6 in Chapter 2, "Shared Concepts and Topics."

PROC HPREG 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 40 in Chapter 3, "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 data set contains salary and performance information for Major League Baseball players (excluding pitchers) who played at least one game in both the 1986 and 1987 seasons. The salaries are for the 1987 season (*Sports Illustrated,* April 20, 1987) and the performance measures are from 1986 (Collier Books, *The 1987 Baseball Encyclopedia Update*).

```
data baseball;
  length name $ 18;
  length team $ 12;
  input name $ 1-18 nAtBat nHits nHome nRuns nRBI nBB
        yrMajor crAtBat crHits crHome crRuns crRbi crBB
        league $ division $ team $ position $ nOuts nAssts
        nError salary;
  label name="Player's Name"
     nAtBat="Times at Bat in 1986"
     nHits="Hits in 1986"
     nHome="Home Runs in 1986"
     nRuns="Runs in 1986"
     nRBI="RBIs in 1986"
     nBB="Walks in 1986"
     yrMajor="Years in the Major Leagues"
     crAtBat="Career times at bat"
     crHits="Career Hits"
     crHome="Career Home Runs"
     crRuns="Career Runs"
     crRbi="Career RBIs"
```

```
      crBB="Career Walks"
      league="League at the end of 1986"
      division="Division at the end of 1986"
      team="Team at the end of 1986"
      position="Position(s) in 1986"
      nOuts="Put Outs in 1986"
      nAssts="Assists in 1986"
      nError="Errors in 1986"
      salary="1987 Salary in $ Thousands";
      logSalary = log(Salary);
   datalines;
Allanson, Andy       293    66     1    30    29    14
                       1   293    66     1    30    29    14
                    American East Cleveland C 446 33 20 .
Ashby, Alan          315    81     7    24    38    39
                      14  3449   835    69   321   414   375
                    National West Houston C 632 43 10 475
Davis, Alan          479   130    18    66    72    76

   ... more lines ...

Wilson, Willie       631   170     9    77    44    31
                      11  4908  1457    30   775   357   249
                    American West KansasCity CF 408 4 3 1000
;
```

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=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 8.1 through Figure 8.5.

Figure 8.1 Performance, Model, and Selection Information

```
                         The HPREG Procedure

                       Performance Information

                  Execution Mode       Single-Machine
                  Number of Threads    4
```

Figure 8.1 *continued*

```
                      Model Information

           Data Source                WORK.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 8.1 displays the "Performance 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 "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 8.2 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 8.2 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
```

Figure 8.2 *continued*

```
                    Dimensions

        Number of Effects              19
        Number of Parameters           21
```

The “Stepwise Selection Summary” table in Figure 8.3 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 8.3 Selection Summary Table

```
                  The HPREG Procedure

                   Selection Summary

                Effect     Effect        Number
         Step   Entered    Removed   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 8.4 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 8.4 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 8.5 give details of the selected model.

Figure 8.5 Details of the Selected Model

```
                              Analysis of Variance

                                       Sum of            Mean
Source                     DF         Squares          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

                                                   Standard
   Parameter     DF         Estimate                  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=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 8.6 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 8.6 Parameter Estimates with Additional Statistics

```
                              The HPREG Procedure
                                Selected Model

                              Parameter Estimates

                                  Standard                              Variance
Parameter    DF       Estimate       Error    t Value    Pr > |t|      Inflation

Intercept     1       4.013911    0.111290      36.07      <.0001              0
nHits         1       0.007929    0.000994       7.98      <.0001        1.49642
nBB           1       0.007280    0.002049       3.55      0.0005        1.52109
yrMajor       1       0.100663    0.007551      13.33      <.0001        1.02488

                              Parameter Estimates

                      Parameter      95% Confidence Limits

                      Intercept       3.79476       4.23306
                      nHits           0.00597       0.00989
                      nBB             0.00325       0.01131
                      yrMajor         0.08579       0.11553
```

By default, High-Performance Analytics 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 8.7 First 5 Observations of the baseballOut Data Set

```
                            predicted
Obs  name                   LogSalary  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 *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* ;
  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 8.1 summarizes the options in the PROC HPREG statement by function.

Table 8.1 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 6 about the various execution modes and the section "Alongside-the-Database Execution" on page 13 about the alongside-the-database model. Both sections are in Chapter 2, "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 32 in Chapter 2, "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 in PROC HPREG to obtain separate analyses of observations in groups that are defined by the BY variables. When a BY statement appears, PROC HPREG 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 and 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).

BY statement processing is not supported when the HPREG 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 *SAS Language Reference: Concepts*. For more information about the DATASETS procedure, see *Base SAS Procedures Guide*.

CODE Statement

CODE < *options* > **;**

The CODE statement enables you to write 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 8.2 summarizes the *options* available in the CODE statement.

Table 8.2 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 more information about the syntax of the CODE statement, see the section "CODE Statement" (Chapter 19, *SAS/STAT User's 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 Analytics procedures is documented in the section "CLASS Statement" on page 40 of Chapter 3, "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'.

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 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 documentation on the common ID statement in SAS High-Performance Analytics procedures, see the section "ID Statement" on page 44 in Chapter 3, "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 52, of Chapter 3, "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 MODELstatement 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 MODELstatement 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.

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 34.

SELECTION Statement

SELECTION < *options* > **;**

The SELECTION statement performs variable selection. The statement is fully documented in the section "SELECTION Statement" on page 45 in Chapter 3, "Shared Statistical Concepts."

The HPREG procedure supports the following variable selection methods in the METHOD= option in the SELECTION statement:

NONE	No model selection.
FORWARD	The forward selection method starts with no effects in the model and adds effects.
BACKWARD	The backward elimination method starts with all effects in the model and deletes effects.
STEPWISE	The stepwise regression method is similar to the FORWARD method except that effects already in the model do not necessarily stay there.
FORWARDSWAP	The forward-swap selection method 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	The least angle regression method, 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. If the model contains classification variables, then these classification variables are split. See the SPLIT option in the CLASS statement for details.
LASSO	The lasso method adds and deletes parameters based on a version of ordinary least squares where the sum of the absolute regression coefficients is constrained. If the model contains classification variables, then these classification variables are split. See the SPLIT option in the CLASS statement for details.

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.

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 8.3:

Table 8.3 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 8.4 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 283 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 8.4 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, $x_i(\mathbf{X}'\mathbf{X})^{-1}x_i'$
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
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

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

$$F = \frac{(\mathrm{RSS}_{p-k} - \mathrm{RSS}_p)/k}{\mathrm{RSS}_p/(n-p-k)}$$

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 8.5 provides formulas and definitions for these fit statistics.

Table 8.5 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	$\frac{\mathrm{SSE}}{n}$
MSE	$\frac{\mathrm{SSE}}{n-p}$
R^2	$1-\frac{\mathrm{SSE}}{\mathrm{SST}}$
ADJRSQ	$1-\frac{(n-1)(1-R^2)}{n-p}$
AIC	$n\ln\left(\frac{\mathrm{SSE}}{n}\right)+2p$
AICC	$1+\ln\left(\frac{\mathrm{SSE}}{n}\right)+\frac{2(p+1)}{n-p-2}$
CP (C_p)	$\frac{\mathrm{SSE}}{\hat{\sigma}^2}+2p-n$
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	$\sqrt{\mathrm{MSE}}$
SBC	$n\ln\left(\frac{\mathrm{SSE}}{n}\right)+p\ln(n)$

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 8.6 contains the diagnostic statistics and their formulas. Each statistic is computed for each observation.

Table 8.6 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	$\widehat{Y}_i - t_{\frac{\alpha}{2}}\mathrm{STDI}$
LCLM	$\widehat{Y}_i - t_{\frac{\alpha}{2}}\mathrm{STDP}$
UCL	$\widehat{Y}_i + t_{\frac{\alpha}{2}}\mathrm{STDI}$
UCLM	$\widehat{Y}_i + t_{\frac{\alpha}{2}}\mathrm{STDP}$
STUDENT	$\frac{r_i}{\mathrm{STDR}_i}$
RSTUDENT	$\frac{r_i}{\hat{\sigma}_{(i)}\sqrt{1-h_i}}$
COOKD	$\frac{1}{p}\mathrm{STUDENT}^2\frac{\mathrm{STDP}^2}{\mathrm{STDR}^2}$
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 8.8 is produced by default whenever you specify a CLASS statement.

Figure 8.8 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 8.9 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 8.9 Parameter Estimates

```
                              Parameter Estimates

                                              Standard
Parameter     DF          Estimate               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 a both 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 Analytics 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 31 in Chapter 2, “Shared Concepts and Topics.”

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 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, 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.

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 8.5 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 8.7.

Table 8.7 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
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 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	SELECTION DETAILS

Examples: HPREG Procedure

Example 8.1: 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 8.1.1 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 8.1.1 Oracle ASE Values by Role

```
                    Oracle
          Role        ASE

          TEST     25.5784
          TRAIN    25.4008
          VAL      25.8993
```

The ASE values shown Output 8.1.1 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 8.1.2 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 8.1.2 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 8.1.3 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 8.1.4 provide this information.

Output 8.1.3 Selection Summary

```
                              The HPREG Procedure

                                Selection Summary

                        Effect        Number    Validation          p
               Step     Entered    Effects In          ASE      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 8.1.4 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 8.1.5 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 8.1.1.

Output 8.1.5 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 8.1.6 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 8.1.6 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 8.1.7 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 8.1.7 Part of the Parameter Estimates Table for the Selected Model

```
                              Parameter Estimates

                                        Standardized      Standard
Parameter          DF        Estimate       Estimate         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 8.1.8 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 8.1.8 Dimensions with c1 Split

```
                              The HPREG Procedure

                                   Dimensions

                    Number of Effects                       278
                    Number of Effects after Splits          439
                    Number of Parameters                    661
```

Output 8.1.9 shows the selected effects. You can see that as anticipated the selected model now depends on only levels 3 and 7 of c1.

Output 8.1.9 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 8.1.10.

Output 8.1.10 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 8.1.10 with the oracle values in Output 8.1.1 and the values for the model without splitting c1 in Output 8.1.5, 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 8.2: 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 6 in Chapter 2, "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 8.2.1 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 8.2.1 Performance Information

```
                           The HPREG Procedure

                        Performance Information

                    Execution Mode          Single-Machine
                    Number of Threads       4
```

Output 8.2.2 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 8.2.2 Parameter Estimates for the Selected Model

```
                          Parameter Estimates

                                             Standard
Parameter     DF          Estimate              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 8.2.3 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 8.2.3 Timing

```
                     Procedure Task Timing

       Task                                  Seconds     Percent

       Reading and Levelizing Data              0.69       4.18%
       Loading Design Matrix                    0.31       1.90%
       Computing Moments                        0.08       0.48%
       Computing Cross Products Matrix         12.62      76.98%
       Performing Model Selection               2.70      16.46%
```

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 6 in Chapter 2, “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 8.2.4 indicates that the calculations were performed in a distributed environment that uses 10 nodes, each of which uses eight threads.

Output 8.2.4 Performance Information in Distributed Mode

```
                         Performance Information

         Host Node                     << your grid host >>
         Install Location              << your grid install location >>
         Execution Mode                Distributed
         Grid Mode                     Symmetric
         Number of Compute Nodes       10
         Number of Threads per Node    8
```

Another indication of distributed execution is the following message issued by all High-Performance Analytics procedures in the SAS Log:

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

Output 8.2.5 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 performing the model selection.

Output 8.2.5 Timing

```
                          Procedure Task Timing

            Task                                  Seconds     Percent

            Distributing Data                        0.93      18.68%
            Reading and Levelizing Data              0.03       0.66%
            Loading Design Matrix                    0.01       0.22%
            Computing Moments                        0.01       0.14%
            Computing Cross Products Matrix          1.19      23.82%
            Performing Model Selection               2.48      49.60%
            Waiting on Client                        0.34       6.88%
```

Example 8.3: 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 8.3.1.

Output 8.3.1 Selection Summary

```
                              The HPREG Procedure

                               Selection Summary

             Effect     Effect        Number                       Model
     Step    Entered    Removed   Effects In          SBC      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 8.3.2 shows the "Candidates" for step 2. By default, only the best 10 swaps are displayed.

Output 8.3.2 Swap Candidates at Step 2

```
                       Best 10 Candidates

                   Effect      Effect
           Rank    Dropped     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 8.3.3 provides details of the model at step 3.

Output 8.3.3 Model Details at Step 3

```
                           Analysis of Variance

                                      Sum of            Mean
Source                     DF        Squares          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 8.3.3 *continued*

```
                           Parameter Estimates

                                           Standard
Parameter    DF         Estimate              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.

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.

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.

Time Inc. (1987), "What They Make," *Sports Illustrated*, April, 54–81.

Chapter 9
The HPSPLIT Procedure

Contents

Overview: HPSPLIT Procedure **310**
PROC HPSPLIT Features 310
Getting Started: HPSPLIT Procedure **311**
Syntax: HPSPLIT Procedure **314**
PROC HPSPLIT Statement 314
CODE Statement 316
CRITERION Statement 316
ID Statement 317
INPUT Statement 317
OUTPUT Statement 318
PARTITION Statement 319
PERFORMANCE Statement 320
PRUNE Statement 320
RULES Statement 322
SCORE Statement 322
TARGET Statement 322
Details: HPSPLIT Procedure **322**
Building a Tree 322
Interval Input Binning Details 323
Input Variable Splitting and Selection 324
Pruning 325
Memory Considerations 326
Handling Missing Values 326
Handling Unknown Levels in Scoring 327
Splitting Criteria 327
Pruning Criteria 328
Subtree Statistics 329
Variable Importance 335
Outputs 336
Examples: HPSPLIT Procedure **338**
Example 9.1: Creating an English Rules Description of a Tree 338
Example 9.2: Assessing Variable Importance 340
References **342**

Overview: HPSPLIT Procedure

The HPSPLIT procedure is a high-performance utility procedure that creates a decision tree model and saves results in output data sets and files for use in SAS Enterprise Miner.

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
  - provides the entropy, Gini, and FastCHAID methods for tree growth
  - provides multiple statistical metrics for tree pruning
  - provides C4.5-style pruning
  - partitions the input data set into training and validation sets
- 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 English 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 6 in Chapter 2, "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 9.1 describes the variables in HMEQ.

Table 9.1 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 9.1 shows a partial listing of HMEQ.

Figure 9.1 Partial Listing of the HMEQ Data

```
            M                                                  D
            O             R                  D                 E
            R      V      E                D E    C            B
       L    T      A      A                E L    L    N  C    T
O  B   O    D      L      S      J      Y  R I    A    I  L    I
b  A   A    U      U      O      O      O  O N    G    N  N    N
s  D   N    E      E      N      B      J  G Q    E    Q  O    C

 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, 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=sashelp.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 sashelp.hmeq;
  %include 'hpsplhme-code.sas';
run;
```

A partial listing of SCORED is shown in Figure 9.2.

Figure 9.2 Partial Listing of the Scored HMEQ Data

```
Obs BAD LOAN MORTDUE  VALUE REASON   JOB    YOJ DEROG DELINQ  CLAGE  NINQ

  1  1  1100  25860   39025 HomeImp Other  10.5   0      0    94.367   1
  2  1  1300  70053   68400 HomeImp Other   7.0   0      2   121.833   0
  3  1  1500  13500   16700 HomeImp Other   4.0   0      0   149.467   1
  4  1  1500      .       .                  .    .      .       .     .
  5  0  1700  97800  112000 HomeImp Office  3.0   0      0    93.333   0
  6  1  1700  30548   40320 HomeImp Other   9.0   0      0   101.466   1
  7  1  1800  48649   57037 HomeImp Other   5.0   3      2    77.100   1
  8  1  1800  28502   43034 HomeImp Other  11.0   0      0    88.766   0
  9  1  2000  32700   46740 HomeImp Other   3.0   0      2   216.933   1
 10  1  2000      .   62250 HomeImp Sales  16.0   0      0   115.800   0

Obs CLNO  DEBTINC  _NODE_  _LEAF_  _WARN_   P_BAD1   P_BAD0   V_BAD1   V_BAD0

  1   9      .       16       7            0.17391  0.82609  0.18808  0.81192
  2  14      .       13       6            0.29969  0.70031  0.32450  0.67550
  3  10      .       16       7            0.17391  0.82609  0.18808  0.81192
  4   .      .       16       7            0.17391  0.82609  0.18808  0.81192
  5  14      .       16       7            0.17391  0.82609  0.18808  0.81192
  6   8   37.1136    16       7            0.17391  0.82609  0.18808  0.81192
  7  17      .        6       2            0.93939  0.06061  0.87500  0.12500
  8   8   36.8849    16       7            0.17391  0.82609  0.18808  0.81192
  9  12      .       13       6            0.29969  0.70031  0.32450  0.67550
 10  13      .       16       7            0.17391  0.82609  0.18808  0.81192
```

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 336

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. 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 9.2 summarizes the options in the PROC HPSPLIT statement.

Table 9.2 PROC HPSPLIT Statement Options

Option	Description
Basic Options	
DATA=	Specifies the input data set
EVENT=	Specifies the formatted value of the target event
INTERVALBINS=	Specifies the number of bins for interval variables
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
FastCHAID Options	
ALPHA=	Specifies the maximum p-value for a split to be considered
BONFERRNOI	Enables the Bonferroni adjustment to after-split p-values
MINDIST=	Specifies the minimum Kolmogorov-Smirnov distance

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 6 about the various execution modes and the section "Alongside-the-Database Execution" on page 13 about the alongside-the-database model. Both sections are in Chapter 2, "Shared Concepts and Topics."

EVENT=*value*

specifies a formatted *value* of the target level variable to use for sorting nominal input levels when you use the FastCHAID criterion or when PROC HPSPLIT uses the fastest splitting category. Ties are broken in internal order. For example, if you are looking for defects in a data set where a target value of 'D' indicates a defect, specify EVENT='D'.

See section "Input Variable Splitting and Selection" on page 324 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 322.

The default is 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.

The default is 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. Targets that have fewer observations than *number* receive the missing value assignment for that split.

The default is MINCATSIZE=0. That is, it is disabled by default.

ALPHA=*number*

specifies the maximum p-value for a split to be considered if you specify FastCHAID in the CRITERION statement. Otherwise, this option is ignored.

The default is ALPHA=0.3.

BONFERRONI

enables the Bonferroni adjustment to the p-value of each variable's split after the split has been determined by Kolmogorov-Smirnov distance if you specify FastCHAID in the CRITERION statement. Otherwise, this option is ignored.

By default, there is no Bonferroni adjustment.

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.

The default is MINDIST=0.01.

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.

You can set the *criterion* to one of the following:

ENTROPY

uses the gain in information (decrease in entropy) to split each variable and then to determine the split.

This is the default *criterion*.

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.

NOTE: The FASTCHAID criterion is experimental in this release.

GINI
: uses the decrease in Gini statistic to split each variable and then to determine the split.

You can also specify the following *options*:

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.

  The default is 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.

  The default is 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*> < / *subtreestat-option*> **;**

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.

You can specify the following *subtreestat-option*:

SUBTREESTATS=(*metric* **<** *metric* **... >)**

specifies the statistical metrics to write to the subtree data sets. The iteration number, number of leaves, and tree number are always provided.

You can specify one or more of the following *metric*s.

ENTROPY	calculates the entropy of the subtree.
GINI	calculates the Gini statistic of the subtree.
ASE	calculates the average square error of the subtree.
MISC	calculates the misclassification rate of the subtree.
SSE	calculates the sum of squares error of the subtree.
ALL	enables all the statistics.

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 is SEED=3054.

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***', VALID='***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 VALID= 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', VALID='0');
```

The TRAIN= and VALID= 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 34 of Chapter 2, "Shared Concepts and Topics."

PRUNE Statement

PRUNE C45 < / *value* > **;**

PRUNE NONE ;

PRUNE *by-metric* < / *until-metric operator value* > **;**

The PRUNE statement controls pruning. It has three different syntaxes: one for C4.5-style pruning, one for no pruning, and one for pruning by using a specified metric.

The default pruning method is entropy. The following PRUNE statement example is equivalent to having no PRUNE statement:

```
prune entropy;
```

The preceding statement is also equivalent to the following statement:

```
prune entropy / entropy >= 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.

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 any of the following values for *by-metric*:

ASE	chooses the leaf that has the smallest change in the average square error.
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 any of the following values for *until-metric*:

ASE	stops pruning when the per-leaf change in average square error rate is *operator value* times the per-leaf change in the ASE of pruning the whole initial tree to a leaf.
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.
N	stops pruning when the number of leaves is *operator value*.

You can specify any of the following values for *operator*:

<=	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

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 < / option >* **;**

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 *option*:

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.

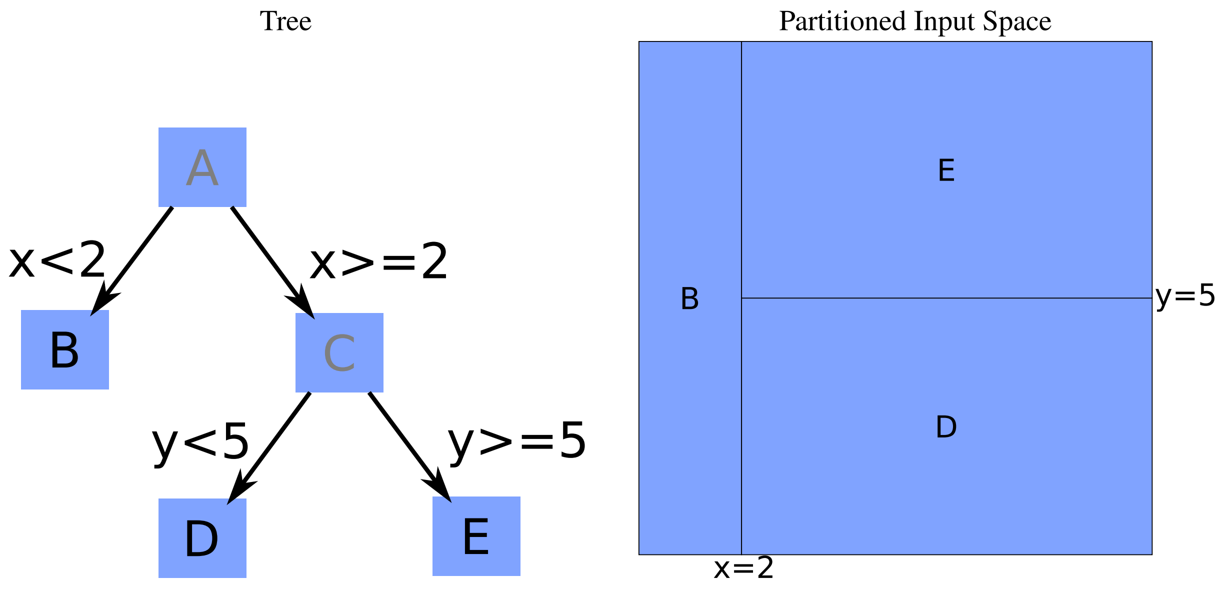
Details: HPSPLIT Procedure

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 9.3 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 9.3 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

$$\left[\frac{vmax - vmin}{intervalbins} bin + vmin, \frac{vmax - vmin}{intervalbins}(bin + 1) + vmin\right)$$

where *vmax* and *vmin* 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:

- entropy
- FastCHAID
- Gini

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.

Selecting the Split Variable

After it finds the split for each variable, the splitter uses the *criterion* from the CRITERION statement to choose the best split variable to use for the final tree node. The entropy and Gini criteria use the named metric to guide the decision.

FastCHAID uses the *p*-value of the two-way table of target-child counts of the proposed split. The ALPHA= option in the PROC HPSPLIT statement (default of 0.3) is 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 causes the *p*-value of the split (which was determined by Kolmogorov-Smirnov distance) to be adjusted using the Bonferroni adjustment.

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 into a node 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 the node to prune back to a leaf. The smallest global increase (or largest decrease) in the specified metric is the criterion used to choose the node to prune. 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 with the lowest global change (smallest increase or largest decrease) in the Gini statistic when the node is made into a leaf, and it terminates 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 had an entropy E_0 and N_0 leaves and trimming away the whole tree to a stump had an entropy of E_s (and, because it is a stump, just one leaf), 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 E_n and N_n leaves and has an entropy of E_λ if it were replaced by 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

$$\frac{\left.\frac{\Delta E}{\Delta N}\right|_n}{\left.\frac{\Delta E}{\Delta N}\right|_0} \geq 1$$

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. 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. After the splitter has determined the per-variable split, the observations that have a missing value in that variable are assigned to the leaf that has the largest number of observations.

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.

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 variables 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 less than the minimum to the first bin and a value 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."

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, if a validation partition is present, pruning statistics are calculated from that.

Entropy Pruning Criterion

When you prune by entropy, the entropy is calculated as though the *entire data set* were a single leaf partitioned into the final number of leaves. Thus it can be expected that the pruning 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 with the node preserved and the node pruned back to a leaf.

Gini Pruning Criterion

As with entropy, the change in Gini statistic is calculated based on the change in the global Gini statistic. The equations are otherwise unchanged.

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

$$\mathrm{MISC}_\lambda = \sum_{\tau_i \neq \tau_P} \frac{N^\lambda_{\tau_i}}{N_\lambda}$$

For all the leaves in the tree, it is

$$\mathrm{MISC} = \sum_\lambda \frac{N_\lambda}{N_0} \sum_{\tau_i \neq \tau_P} \frac{N^\lambda_{\tau_i}}{N_\lambda}$$

The predicted target level is always based on the training data set.

Average Square Error Pruning Criterion

The average square error (ASE) is based on the sum of squares error (SSE). You would expect, for a perfect assignment, 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.

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.

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 that 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 per-observation statistics are calculated separately for each. 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 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).

Observationwise Entropy Statistic

The entropy at each observation is calculated from the posterior probabilities:

$$\text{Entropy} = -\sum_\omega \frac{1}{N_0} \sum_\tau P_\tau^\omega \log_2\left(P_\tau^\omega\right)$$

Observationwise Gini Statistic

Like the entropy, the Gini statistic is also calculated from the posterior probabilities:

$$\text{Gini} = \sum_\omega \frac{1}{N_0} \sum_\tau P_\tau^\omega \left(1 - P_\tau^\omega\right)$$

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:

$$\delta_{\tau_P^\omega \tau_\pi^\omega} = \begin{cases} 1 & \text{if } \tau_p^\omega = \tau_\pi^\omega \\ 0 & \text{otherwise} \end{cases}$$

Or, phrased slightly differently, the misclassification rate is the fraction of incorrectly predicted observations:

$$\text{MISC} = \frac{1}{N_0} \sum_{\omega} \begin{cases} 0 & \text{if } \tau_p^\omega = \tau_\pi^\omega \\ 1 & \text{otherwise} \end{cases}$$

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, with τ_π^ω once again being the actual target level for observation ω:

$$\text{SSE} = \sum_{\omega} \left[\sum_{\tau \neq \tau_\pi^\omega} \left(P_\tau^\omega\right)^2 + \left(1 - P_{\tau_\pi^\omega}^\omega\right)^2 \right]$$

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):

$$\text{ASE} = \frac{1}{N_\tau N_0} \sum_{\omega} \left[\sum_{\tau \neq \tau_\pi^\omega} \left(P_\tau^\omega\right)^2 + \left(1 - P_{\tau_\pi^\omega}\right)^2 \right]$$

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, the proportions of observations are examined 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.

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

$$\text{Entropy} = -\sum_{\lambda} \frac{N_\lambda}{N_0} \sum_{\tau} \frac{N_\tau^\lambda}{N_\lambda} \log_2 \left(\frac{N_\tau^\lambda}{N_\lambda} \right)$$

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

$$\text{Gini} = \sum_{\lambda} \frac{N_\lambda}{N_0} \sum_{\tau} \frac{N_\tau^\lambda}{N_\lambda} \left(1 - \frac{N_\tau^\lambda}{N_\lambda} \right)$$

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. The misprediction rate of a single leaf, similar to the misprediction rate of an entire data set, 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

$$\begin{aligned}\mathrm{MISC} &= \sum \frac{N_\lambda}{N_0} \sum_{\tau \neq \tau_\pi} P_\tau^\lambda \\ &= \sum \frac{N_\lambda}{N_0} \sum_{\tau \neq \tau_\pi} \frac{N_\tau^\lambda}{N_\lambda}\end{aligned}$$

where again P_τ^λ is V_τ^λ for the validation set.

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 themselves 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

$$\begin{aligned}
\text{SSE} &= \sum_{\lambda} \sum_{\Phi} N_{\Phi}^{\lambda} \left[\sum_{\tau \neq \tau_{\pi}^{\Phi}} \left(\frac{\nu_{\tau}^{\lambda}}{\nu_{\lambda}} \right)^{2} + \left(1 - \frac{\nu_{\tau_{\pi}^{\Phi}}^{\lambda}}{\nu_{\lambda}} \right)^{2} \right] \\
&= \sum_{\lambda} \sum_{\Phi} N_{\Phi}^{\lambda} \left[\sum_{\tau \neq \tau_{\pi}^{\Phi}} \left(\frac{\nu_{\tau}^{\lambda}}{\nu_{\lambda}} \right)^{2} + 1 - 2\frac{\nu_{\tau_{\pi}^{\Phi}}^{\lambda}}{\nu_{\lambda}} + \left(\frac{\nu_{\tau_{\pi}^{\Phi}}^{\lambda}}{\nu_{\lambda}} \right)^{2} \right] \\
&= \sum_{\lambda} \sum_{\Phi} N_{\Phi}^{\lambda} \left[\sum_{\tau} \left(\frac{\nu_{\tau}^{\lambda}}{\nu_{\lambda}} \right)^{2} + 1 - 2\frac{\nu_{\tau_{\pi}^{\Phi}}^{\lambda}}{\nu_{\lambda}} + \right] \\
&= \sum_{\lambda} N_{\lambda} \sum_{\Phi} \frac{N_{\Phi}^{\lambda}}{N_{\lambda}} \left[\sum_{\tau} \left(\frac{\nu_{\tau}^{\lambda}}{\nu_{\lambda}} \right)^{2} + 1 - 2\frac{\nu_{\tau_{\pi}^{\Phi}}^{\lambda}}{\nu_{\lambda}} \right] \\
&= \sum_{\lambda} N_{\lambda} \left[1 + \sum_{\tau} \left(\frac{\nu_{\tau}^{\lambda}}{\nu_{\lambda}} \right)^{2} - 2 \sum_{\Phi} \frac{N_{\Phi}^{\lambda}}{N_{\lambda}} \frac{\nu_{\tau_{\pi}^{\Phi}}^{\lambda}}{\nu_{\lambda}} \right]
\end{aligned}$$

Now, in that 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 with 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):

$$\text{SSE} = \sum_{\lambda} N_{\lambda} \left[1 - \sum_{\tau} \left(\frac{\nu_{\tau}^{\lambda}}{\nu_{\lambda}} \right)^{2} \right]$$

$$\text{SSE} = \sum_{\lambda} N_{\lambda} \left[1 - \sum_{\tau} \left(P_{\tau}^{\lambda} \right)^{2} \right]$$

Leafwise Average Square Error

Because 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:

$$\text{SSE} = \sum_{\lambda} \frac{N_\lambda}{N_\tau N_0} \left[1 - \sum_{\tau} \left(\frac{\nu_\tau^\lambda}{\nu_\lambda} \right)^2 \right]$$

$$\text{SSE} = \sum_{\lambda} \frac{N_\lambda}{N_\tau N_0} \left[1 - \sum_{\tau} \left(P_\tau^\lambda \right)^2 \right]$$

Variable Importance

Variable importance is calculated based on how the variables are used in the finished tree. Three metrics are used: 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. The SSE and relative importance are calculated from the training set. They are also calculated again from the validation set if one exists. These are reported as “VSSE” and “VIMPORT.”

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

$$\Delta_\eta = \text{SSE}_\eta - \sum_{\lambda} \text{SSE}_\lambda^\eta$$

where η denotes the node. SSE_η is then the SSE if the node is treated as a leaf, and SSE_λ^η 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*.

The SSE-based importance is then

$$\text{SSEIMPORT}_{\text{variable}} = \sqrt{\sum_{\eta}^{\text{variable nodes}} \Delta_\eta}$$

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

$$\text{IMPORT}_{\text{variable}} = \frac{\text{SSEIMPORT}_{\text{variable}}}{\text{SSEIMPORT}_{\text{max}}}$$

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 9.3 shows the variables that are contained in an example data set that the SCORE statement produces. In this data set, the variable BAD is the target and has values 0 and 1.

Table 9.3 Example SCORE Statement Data Set Variables

Variable	Description
BAD	Target variable
LEAF	Leaf number to which this observation is assigned
NODE	Node number to which this observation is assigned
P_BAD0	Proportion of training set at this leaf that has BAD = 0
P_BAD1	Proportion of training set at this leaf that has BAD = 1
V_BAD0	Proportion of validation set at this leaf that has BAD = 0
V_BAD1	Proportion of validation set at this leaf that has BAD = 1

IMPORTANCE= Data Set

The variable importance data set contains the importance of the input variables in creating the pruned decision tree. A simple count-based importance metric and two variable importance metrics that are based on the sum of squares error-based are output. In addition, 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 are output. Table 9.4 shows the variables contained in the data set that the OUTPUT statement produces using the IMPORTANCE= option. In addition to the variables listed below, a variable containing the importance for each input variable is included.

Table 9.4 Variable Importance Data Set Variables

Variable	Description
TREENUM	Tree number (always 1)
CRITERION	Criterion used to generate the tree
ITYPE	Importance type (“Count”, “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 used to build the tree (training set)
OBSVALID	Number of observations in the validation set

NODESTATS= Data Set

The data set specified in the NODESTATS= option in the OUTPUT statement can be used to visualize the tree. Table 9.5 shows the variables in this data set.

Table 9.5 NODESTATS= Data Set Variables

Variable	**Description**
ALLTEXT	Text that describes the split
CRITERION	Which of the three criteria was used
DECISION	Values of the parent variable's split to get to this node
DEPTH	Depth of the node
ID	Node number
LINKWIDTH	Fraction of all training observations going to this node
N	Number of training observations at this node
NVALID	Number of validation observations at this node
PARENT	Parent's node number
PREDICTEDVALUE	Value of target predicted at this node
P_BAD0	Proportion of training observations that have BAD=0
P_BAD1	Proportion of training observations that have BAD=1
SPLITVAR	Variable used in the split
TREENUM	Tree number (always 1)
V_BAD0	Proportion of validation observations that have BAD=0
V_BAD1	Proportion of validation observations that have BAD=1

GROWTHSUBTREE= and PRUNESUBTREE= Data Sets

During tree growth and pruning, the number of leaves at each growth or pruning iteration is output in addition to other, optional metrics.

The GROWTHSUBTREE= and PRUNESUBTREE= 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 from the training subset. The statistics of the pruning data set are from the validation subset if one is available. Otherwise, the statistics of the pruning data set are from the training subset.

Table 9.6 GROWTHSUBTREE= and PRUNESUBTREE= Data Set Variables

Variable	Description
ITERATION	Iteration number
NLEAVES	Number of leaves
TREENUM	Tree number (always 1)
ASE	Training set: average square error
ENTROPY	Training set: entropy
GINI	Training set: Gini
MISC	Training set: misclassification rate
SSE	Training set: sum of squares error
VASE	Validation set: average square error
VENTROPY	Validation set: entropy
VGINI	Validation set: Gini
VMISC	Validation set: misclassification rate
VSSE	Validation set: sum of squares error

Examples: HPSPLIT Procedure

Example 9.1: Creating an English Rules Description of a Tree

This example creates a tree model and saves an English 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 311.

The following statements create the tree model.

```
proc hpsplit data=sashelp.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 English rules description of the model is saved. A 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, 15
AND DELINQ IS ONE OF 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 15
    PREDICTED VALUE IS 1
    PREDICTED 1 = 0.9342( 71/76)
    PREDICTED 0 = 0.06579( 5/76)
*------------------------------------------------------------*
NODE = 4
*------------------------------------------------------------*
NINQ IS ONE OF 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 17
AND DELINQ IS ONE OF MISSING, 1, 2, 3, 4
AND DELINQ IS ONE OF 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 15
    PREDICTED VALUE IS 1
    PREDICTED 1 = 0.8714( 61/70)
    PREDICTED 0 = 0.1286( 9/70)
*------------------------------------------------------------*
NODE = 5
*------------------------------------------------------------*
NINQ IS ONE OF MISSING, 0, 1, 2, 3, 7
AND DELINQ IS ONE OF MISSING, 1, 2, 3, 4
AND DELINQ IS ONE OF 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 15
    PREDICTED VALUE IS 0
    PREDICTED 1 = 0.3682( 306/831)
    PREDICTED 0 = 0.6318( 525/831)
*------------------------------------------------------------*
NODE = 8
*------------------------------------------------------------*
DEBTINC IS MISSING OR DEBTINC <45.137782
AND CLAGE IS MISSING OR CLAGE <186.91737
AND DELINQ IS ONE OF MISSING, 0
    PREDICTED VALUE IS 0
    PREDICTED 1 = 0.1739( 392/2254)
    PREDICTED 0 = 0.8261( 1862/2254)
*------------------------------------------------------------*
NODE = 9
*------------------------------------------------------------*
DEBTINC >=45.137782
AND CLAGE IS MISSING OR CLAGE <186.91737
AND DELINQ IS ONE OF MISSING, 0
    PREDICTED VALUE IS 1
    PREDICTED 1 = 1( 37/37)
    PREDICTED 0 = 0( 0/37)
*------------------------------------------------------------*
NODE = 10
*------------------------------------------------------------*
CLAGE >=186.91737
AND DELINQ IS ONE OF MISSING, 0
    PREDICTED VALUE IS 0
    PREDICTED 1 = 0.0589( 90/1528)
    PREDICTED 0 = 0.9411( 1438/1528)
```

The 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 in the order 1, 2, 3, and so on, 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 instance, the leaf labeled "NODE = 2" covers the region where DELINQ is between 1 and 8, between 10 and 13, or is equal to 15, and the region where DELINQ is also between 5 and 8, between 10 and 13, or is equal to 15. 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. For the leaf labeled "NODE=10", the data set was first split on DELINQ and the subset with DELINQ=0 or where the value for DELINQ was missing were split again on CLAGE, with those observations having CLAGE >=186.91737 going to node 10. At this leaf (node 10), the predicted value for BAD is 0 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 9.1.1 lists the first ten observations of SCORED2.

Output 9.1.1 Scored Input Data Set (HMEQ)

```
Obs    BAD    _NODE_    _LEAF_     P_BAD1     P_BAD0     V_BAD1     V_BAD0

  1     1        8         3      0.17391    0.82609    0.18808    0.81192
  2     1        5         2      0.36823    0.63177    0.36126    0.63874
  3     1        8         3      0.17391    0.82609    0.18808    0.81192
  4     1        8         3      0.17391    0.82609    0.18808    0.81192
  5     0        8         3      0.17391    0.82609    0.18808    0.81192
  6     1        8         3      0.17391    0.82609    0.18808    0.81192
  7     1        5         2      0.36823    0.63177    0.36126    0.63874
  8     1        8         3      0.17391    0.82609    0.18808    0.81192
  9     1        5         2      0.36823    0.63177    0.36126    0.63874
 10     1        8         3      0.17391    0.82609    0.18808    0.81192
```

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 336.

Example 9.2: 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 $ 35-37 usable $ 43-51;
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
;
run;
```

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 9.2.1.

Output 9.2.1 Variable Importance of the One-Split Decision Tree

```
             C
             R              O     O
       T     I      O    O  B     B
       R     T      B    B  S     S                         d
       E     E      S    S  V     T      I     g    a        o
       E     R      M    U  A     M      T     t    t        p
  O    N     I      I    S  L     I      Y     e    e    r   a
  b    U     O      S    E  I     S      P     m    m    o   n
  s    M     N      S    D  D     S      E     p    p    t   t

  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.

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, 203

adaptive lasso selection
    high-performance statistical procedures,
        SELECTION statement, 68
Akaike's information criterion
    MIXED procedure, 220
Akaike's information criterion (finite sample corrected
        version)
    MIXED procedure, 220
alpha level
    HPGENSELECT procedure, 89
    HPLMIXED procedure, 199, 202
    HPLOGISTIC procedure, 146
ANCOVA effects
    Shared Concepts, 56
ANOVA effects
    Shared Concepts, 56
ANOVA table
    HPREG procedure, 291
ANTE(1) structure
    HPLMIXED procedure, 203
antedependence structure
    HPLMIXED procedure, 203
AR(1) structure
    HPLMIXED procedure, 203
association statistics
    HPLOGISTIC procedure, 165
at sign (@) operator
    Shared Concepts, 54, 57
autoregressive moving-average structure
    HPLMIXED procedure, 203
autoregressive structure
    HPLMIXED procedure, 203

backward elimination
    high-performance statistical procedures,
        SELECTION statement, 63
banded Toeplitz structure
    HPLMIXED procedure, 203
bar (|) operator
    Shared Concepts, 54, 56, 57
bar (|) operator
    Shared Concepts, 56
Bernoulli distribution
    HPNLMOD procedure, 238
binary distribution
    HPNLMOD procedure, 238
binomial distribution
    HPNLMOD procedure, 238
BLUE
    HPLMIXED procedure, 216
BLUP
    HPLMIXED procedure, 216
boundary constraints, 237
    HPLMIXED procedure, 200, 201
bounds
    HPNLMOD procedure, 237

candidates for addition or removal
    HPGENSELECT procedure, 112
    HPREG procedure, 290
class level
    HPGENSELECT procedure, 83, 111
    HPLMIXED procedure, 197
    HPLOGISTIC procedure, 140, 162
    HPREG procedure, 274, 289
    MIXED procedure, 218
CLASS statement
    Shared Concepts, 50
classification variables
    Shared Concepts, 50
colon (:) operator
    Shared Concepts, 55
complete separation
    HPLOGISTIC procedure, 155
compound symmetry structure
    example (HPLMIXED), 210
    HPLMIXED procedure, 203
computational method
    HPGENSELECT procedure, 106
    HPLMIXED procedure, 217
    HPLOGISTIC procedure, 157
    HPNLMOD procedure, 248
    HPREG procedure, 287
confidence limits
    model parameters (HPGENSELECT), 89
    model parameters (HPLOGISTIC), 146
constraints
    boundary (HPLMIXED), 200, 201
continuous-by-class effects
    Shared Concepts, 59
continuous-nesting-class effects
    Shared Concepts, 58
convergence criterion
    HPGENSELECT procedure, 80–82

HPLMIXED procedure, 194, 195
HPLOGISTIC procedure, 137–139
HPNLMOD procedure, 233, 234
MIXED procedure, 219
convergence status
HPGENSELECT procedure, 113
HPLOGISTIC procedure, 163
MIXED procedure, 220
correlation matrix
HPGENSELECT procedure, 81, 114
covariance matrix
HPGENSELECT procedure, 81, 114
covariance parameter estimates
MIXED procedure, 220
covariance structure
antedependence (HPLMIXED), 206
autoregressive (HPLMIXED), 206
autoregressive moving average (HPLMIXED), 206
banded (HPLMIXED), 207
compound symmetry (HPLMIXED), 206
equi-correlation (HPLMIXED), 206
examples (HPLMIXED), 204
factor-analytic (HPLMIXED), 206
heterogeneous autoregressive (HPLMIXED), 206
heterogeneous compound symmetry (HPLMIXED), 206
heterogeneous Toeplitz (HPLMIXED), 207
HPLMIXED procedure, 188, 203
Huynh-Feldt (HPLMIXED), 207
simple (HPLMIXED), 207
Toeplitz (HPLMIXED), 207
unstructured (HPLMIXED), 207
unstructured, correlation (HPLMIXED), 207
variance components (HPLMIXED), 207
crossed effects
Shared Concepts, 57

dash (-) operator
Shared Concepts, 55
degrees of freedom
HPLMIXED procedure, 199
infinite (HPLMIXED), 199
method (HPLMIXED), 199
residual method (HPLMIXED), 199
diagnostic statistics
HPREG procedure, 283
dimension information
MIXED procedure, 218
dimensions
HPGENSELECT procedure, 114
HPLOGISTIC procedure, 164
HPREG procedure, 289
direct product structure
HPLMIXED procedure, 203
displayed output
HPGENSELECT procedure, 111
HPLMIXED procedure, 218
HPLOGISTIC procedure, 161
HPNLMOD procedure, 252
HPREG procedure, 289
distributed computing
HPLMIXED procedure, 217
distribution function
HPGENSELECT procedure, 89
distribution functions
HPNLMOD procedure, 246
double dash (- -) operator
Shared Concepts, 55

effect
name length (HPGENSELECT), 83
name length (HPLMIXED), 197
name length (HPLOGISTIC), 140
name length (HPREG), 274
empirical Bayes estimate
HPNLMOD procedure, 242
estimation
mixed model (HPLMIXED), 214
estimation methods
HPLMIXED procedure, 196
examples, HPLMIXED
compound symmetry, G-side setup, 211
compound symmetry, R-side setup, 211
holding covariance parameters fixed, 200
specifying lower bounds, 200
split-plot design, 212
subject and no-subject formulation, 211
unstructured covariance, G-side, 203
examples, HPNLMOD procedure
boundary specification, 237
conditional model expression, 257
enzyme data, 229
gamma distribution, 247
join point, 257
plateau model, 257
predicted values, 257
segmented model, 257
starting values, data set, 241
starting values, grid, 240

factor analytic structures
HPLMIXED procedure, 203
fit criteria
HPREG procedure, 282
fit statistics
HPGENSELECT procedure, 114
HPLOGISTIC procedure, 164

HPREG procedure, 291
fixed effects
HPLMIXED procedure, 188
fixed-effects parameters
HPLMIXED procedure, 210
forward selection
high-performance statistical procedures, SELECTION statement, 61
forward swap selection
high-performance statistical procedures, SELECTION statement, 66
frequency variable
high-performance statistical procedures, 44
HPGENSELECT procedure, 85
HPLOGISTIC procedure, 143
HPREG procedure, 276

G matrix
HPLMIXED procedure, 188, 201, 210
gamma distribution
HPNLMOD procedure, 238
Gaussian distribution
HPNLMOD procedure, 238
general distribution
HPNLMOD procedure, 238
general effects
Shared Concepts, 59
general linear covariance structure
HPLMIXED procedure, 203
generalized inverse, 216
global tests
HPLOGISTIC procedure, 164
gradient
MIXED procedure, 219
growth curve analysis
example (HPLMIXED), 210

Hessian matrix
HPLMIXED procedure, 219
heterogeneous
AR(1) structure (HPLMIXED), 203
compound-symmetry structure (HPLMIXED), 203
covariance structure (HPLMIXED), 207
Toeplitz structure (HPLMIXED), 203
high-performance statistical procedures, SELECTION statement
adaptive lasso selection, 68
backward elimination, 63
forward selection, 61
forward swap selection, 66
LAR selection, 67
LASSO selection, 67
stepwise selection, 64
Hosmer-Lemeshow test
HPLOGISTIC procedure, 147, 156, 157, 164
HPGENSELECT procedure, 71
alpha level, 89
candidates for addition or removal, 112
class level, 83, 111
computational method, 106
confidence limits, 89
convergence criterion, 80–82
convergence status, 113
correlation matrix, 81, 114
covariance matrix, 81, 114
dimensions, 114
displayed output, 111
distribution function, 89
effect name length, 83
fit statistics, 114
function-based convergence criteria, 80, 81
gradient-based convergence criteria, 81, 82
input data sets, 81
iteration history, 113
link function, 90
model information, 111
model options summary, 86
multithreading, 94, 106
number of observations, 111
ODS table names, 115
optimization technique, 107
parameter estimates, 114
performance information, 111
response level ordering, 87
response profile, 112
response variable options, 87
selected effects, 113
selection details, 112
selection information, 112
selection reason, 113
selection summary, 112
stop reason, 113
user-defined formats, 82
weighting, 95
XML input stream, 82
zero inflation link function, 96
zero-inflation parameter estimates, 115
HPLMIXED procedure, 186
2D geometric anisotropic structure, 203
alpha level, 199, 202
ANTE(1) structure, 203
antedependence structure, 203
AR(1) structure, 203
ARMA structure, 203
autoregressive moving-average structure, 203
autoregressive structure, 203
banded Toeplitz structure, 203

BLUE, 195, 216
BLUP, 195, 216
boundary constraints, 200, 201
chi-square test, 199
class level, 197
compound symmetry structure, 203, 210
computational method, 217
confidence interval, 202
confidence limits, 199, 202
convergence criterion, 194, 195
covariance structure, 188, 203, 204
degrees of freedom, 199
direct product structure, 203
displayed output, 218
distributed computing, 217
EBLUPs, 202, 217
effect name length, 197
estimation methods, 196
factor analytic structures, 203
fixed effects, 188
fixed-effects parameters, 199, 210
function-based convergence criteria, 194, 195
G matrix, 188, 201, 210
general linear covariance structure, 203
generalized inverse, 216
gradient-based convergence criteria, 194, 195
grid search, 199
growth curve analysis, 210
Hessian matrix, 219
heterogeneous AR(1) structure, 203
heterogeneous compound-symmetry structure, 203
heterogeneous covariance structures, 207
heterogeneous Toeplitz structure, 203
Huynh-Feldt structure, 203
infinite degrees of freedom, 199
infinite likelihood, 208
initial values, 199
input data sets, 195
intercept effect, 199, 202
Kronecker product structure, 203
linear covariance structure, 203
Matérn covariance structure, 203
matrix notation, 209
mixed model, 210
mixed model equations, 215
mixed model theory, 209
multi-threading, 201
multithreading, 217
Newton-Raphson algorithm, 215
ODS table names, 221
optimization information, 219
parameter constraints, 200
parameter-based convergence criteria, 197
performance information, 218
R matrix, 188, 208, 210
random effects, 188, 201
random-effects parameters, 202, 210
repeated measures, 208
residual method, 199
restricted maximum likelihood (REML), 187
ridging, 215
spatial anisotropic exponential structure, 203
split-plot design, 212
standard linear model, 188
statement positions, 192
subject effect, 202, 208
summary of commands, 193
table names, 221
timing, 221
Toeplitz structure, 203
unstructured correlations, 203
unstructured covariance matrix, 203
variance components, 203
variance ratios, 200
HPLOGISTIC procedure, 127
alpha level, 146
association statistics, 165
class level, 140, 162
complete separation, 155
computational method, 157
confidence limits, 146
convergence criterion, 137–139
convergence status, 163
dimensions, 164
displayed output, 161
effect name length, 140
existence of MLEs, 154
fit statistics, 164
function-based convergence criteria, 137, 138
global tests, 164
gradient-based convergence criteria, 138, 139
Hosmer-Lemeshow test, 147, 156, 157, 164
infinite parameter estimates, 154
input data sets, 138
iteration history, 163
link function, 147
model information, 162
model options summary, 144
multithreading, 150, 157
number of observations, 162
ODS table names, 165
optimization technique, 158
parameter estimates, 165
performance information, 161
quasi-complete separation, 155
response level ordering, 144
response profile, 162

response variable options, 144
selected effects, 163
selection information, 162
selection reason, 163
selection summary, 162
separation, 154
stop reason, 163
user-defined formats, 138
weighting, 151
XML input stream, 138
HPNLMOD procedure, 227
additional estimates, 237, 254
additional estimates correlation, 255
additional estimates covariance, 255
ANOVA, 254
Bernoulli distribution, 238
binary distribution, 238
binomial distribution, 238
bounds, 237
computational method, 248
convergence criterion, 233, 234
convergence status, 253
correlation, 255
covariance, 254
dimensions, 253
displayed output, 252
distribution functions, 246
empirical Bayes estimate, 242
fit statistics, 254
function-based convergence criteria, 233, 234
gamma distribution, 238
Gaussian distribution, 238
general distribution, 238
gradient-based convergence criteria, 233, 234
initial values, 239
input data sets, 233
iteration history, 253
lag functionality, 244
least squares, 245
least squares distribution, 238
linear constraints, 253
log-likelihood functions, 246
multithreading, 241, 248
negative binomial distribution, 238
normal distribution, 238
number of observations, 253
ODS table names, 255
optimization technique, 249
output data sets, 236
parameter estimates, 254
parameters, 253
performance information, 252
Poisson distribution, 238
prediction, 242
procedure task timing, 255
programming statements, 243
residual distribution, 238
restrictions, 243
segmented model example, 256
specifications, 252
starting values, 239
user-defined formats, 234, 236
XML input stream, 234, 236
HPREG procedure, 263
ANOVA table, 291
candidates for addition or removal, 290
class level, 274, 289
computational method, 287
diagnostic statistics, 283
dimensions, 289
displayed output, 289
effect name length, 274
fit criteria, 282
fit statistics, 291
input data sets, 273
introductory example, 266
model information, 289
multithreading, 278, 287
number of observations, 289
ODS table names, 293
output data set, 288
parameter estimates, 292
performance information, 289
random number seed, 274
selected effects, 291
selection information, 289
selection reason, 290
selection summary, 290
stop reason, 290
test data, 286
timing, 292
user-defined formats, 273
validation, 286
weighting, 281
XML input stream, 273
Huynh-Feldt
structure (HPLMIXED), 203

infinite likelihood
HPLMIXED procedure, 208
infinite parameter estimates
HPLOGISTIC procedure, 154
initial values
HPLMIXED procedure, 199
HPNLMOD procedure, 239
interaction effects
Shared Concepts, 57
intercept

- Shared Concepts, 56

iteration history
- HPGENSELECT procedure, 113
- HPLOGISTIC procedure, 163
- MIXED procedure, 219

iterations
- history (MIXED), 219

Kronecker product structure
- HPLMIXED procedure, 203

lag functionality
- HPNLMOD procedure, 244

LAR selection
- high-performance statistical procedures, SELECTION statement, 67

LASSO selection
- high-performance statistical procedures, SELECTION statement, 67

least squares
- HPNLMOD procedure, 245

least squares distribution
- HPNLMOD procedure, 238

levelization
- Shared Concepts, 50

linear covariance structure
- HPLMIXED procedure, 203

link function
- HPGENSELECT procedure, 90
- HPLOGISTIC procedure, 147

log-likelihood functions
- HPNLMOD procedure, 246

main effects
- Shared Concepts, 56

Matérn covariance structure
- HPLMIXED procedure, 203

matrix
- notation, theory (HPLMIXED), 209

maximum likelihood
- estimates (HPLOGISTIC), 154

maximum likelihood estimation
- mixed model (HPLMIXED), 215

mixed model (HPLMIXED)
- estimation, 214
- formulation, 210
- maximum likelihood estimation, 215
- notation, 188
- theory, 209

mixed model (MIXED)
- objective function, 219

mixed model equations
- HPLMIXED procedure, 215

MIXED procedure
- Akaike's information criterion, 220
- Akaike's information criterion (finite sample corrected version), 220
- class level, 218
- convergence criterion, 219
- convergence status, 220
- covariance parameter estimates, 220
- dimension information, 218
- fitting information, 220
- gradient, 219
- iteration history, 219
- iterations, 219
- model information, 218
- number of observations, 218
- Schwarz's Bayesian information criterion, 220

model
- information (HPGENSELECT), 111
- information (HPLOGISTIC), 162
- information (HPREG), 289
- specification (HPNLMOD procedure, 238

model information
- MIXED procedure, 218

multi-threading
- HPLMIXED procedure, 201

multithreading
- HPGENSELECT procedure, 94, 106
- HPLMIXED procedure, 217
- HPLOGISTIC procedure, 150, 157
- HPNLMOD procedure, 241, 248
- HPREG procedure, 278, 287

negative binomial distribution
- HPNLMOD procedure, 238

nested effects
- Shared Concepts, 58

nested versus crossed effects
- Shared Concepts, 58

Newton-Raphson algorithm
- HPLMIXED procedure, 215

normal distribution
- HPNLMOD procedure, 238

number of observations
- HPGENSELECT procedure, 111
- HPLOGISTIC procedure, 162
- HPREG procedure, 289
- MIXED procedure, 218

objective function
- mixed model (MIXED), 219

optimization information
- HPLMIXED procedure, 219

optimization technique
- HPGENSELECT procedure, 107
- HPLOGISTIC procedure, 158
- HPNLMOD procedure, 249

options summary
  MODEL statement (HPLMIXED), 198
  PROC HPGENSELECT statement, 79
  PROC HPLMIXED statement, 193
  PROC HPLOGISTIC statement, 136
  PROC HPNLMOD statement, 232
  PROC HPREG statement, 273
  RANDOM statement (HPLMIXED), 202
  REPEATED statement (HPLMIXED), 208
ordering
  of class levels (Shared Concepts), 51
output data set
  HPREG procedure, 288
overlap of data points
  HPLOGISTIC procedure, 155

parameter constraints
  HPLMIXED procedure, 200
parameter estimates
  HPGENSELECT procedure, 114
  HPLOGISTIC procedure, 165
  HPREG procedure, 292
parameterization
  Shared Concepts, 52
performance information
  HPGENSELECT procedure, 111
  HPLMIXED procedure, 218
  HPLOGISTIC procedure, 161
  HPREG procedure, 289
Poisson distribution
  HPNLMOD procedure, 238
polynomial effects
  Shared Concepts, 56
prediction
  HPNLMOD procedure, 242
programming statements
  HPNLMOD procedure, 243

quasi-complete separation
  HPLOGISTIC procedure, 155

R matrix
  HPLMIXED procedure, 188, 208, 210
random effects
  HPLMIXED procedure, 188, 201
random-effects parameters
  HPLMIXED procedure, 210
regression effects
  Shared Concepts, 56
repeated measures
  HPLMIXED procedure, 208
residual distribution
  HPNLMOD procedure, 238
residual maximum likelihood (REML)
  HPLMIXED procedure, 215
response level ordering
  HPGENSELECT procedure, 87
  HPLOGISTIC procedure, 144
response profile
  HPGENSELECT procedure, 112
  HPLOGISTIC procedure, 162
response variable options
  HPGENSELECT procedure, 87
  HPLOGISTIC procedure, 144
restricted maximum likelihood
  HPLMIXED procedure, 187
restricted maximum likelihood (REML)
  HPLMIXED procedure, 215
restrictions
  HPNLMOD procedure, 243
reverse response level ordering
  HPGENSELECT procedure, 87
  HPLOGISTIC procedure, 144
ridging
  HPLMIXED procedure, 215

Schwarz's Bayesian information criterion
  MIXED procedure, 220
selected effects
  HPGENSELECT procedure, 113
  HPLOGISTIC procedure, 163
  HPREG procedure, 291
selection details
  HPGENSELECT procedure, 112
selection information
  HPGENSELECT procedure, 112
  HPLOGISTIC procedure, 162
  HPREG procedure, 289
selection reason
  HPGENSELECT procedure, 113
  HPLOGISTIC procedure, 163
  HPREG procedure, 290
selection summary
  HPGENSELECT procedure, 112
  HPLOGISTIC procedure, 162
  HPREG procedure, 290
separation
  HPLOGISTIC procedure, 154
Shared Concepts
  ANCOVA effects, 56
  ANOVA effects, 56
  at sign (@) operator, 54, 57
  bar (|) operator, 54, 56, 57
  bar (|) operator, 56
  CLASS statement, 50
  classification variables, 50
  colon (:) operator, 55
  continuous-by-class effects, 59
  continuous-nesting-class effects, 58

crossed effects, 57
dash (-) operator, 55
double dash (- -) operator, 55
general effects, 59
interaction effects, 57
intercept, 56
levelization, 50
main effects, 56
missing values, class variables, 52
nested effects, 58
nested versus crossed effects, 58
ORDER= option, 51
ordering of class levels, 51
parameterization, 52
polynomial effects, 56
regression effects, 56
singular parameterization, 57
sort order of class levels, 51
singular parameterization
Shared Concepts, 57
sort order
of class levels (Shared Concepts), 51
spatial anisotropic exponential structure
HPLMIXED procedure, 203
split-plot design
HPLMIXED procedure, 212
standard linear model
HPLMIXED procedure, 188
starting values
HPNLMOD procedure, 239
stepwise selection
high-performance statistical procedures,
SELECTION statement, 64
stop reason
HPGENSELECT procedure, 113
HPLOGISTIC procedure, 163
HPREG procedure, 290
subject effect
HPLMIXED procedure, 202, 208
summary of commands
HPLMIXED procedure, 193

table names
HPLMIXED procedure, 221
test data
HPREG procedure, 286
timing
HPLMIXED procedure, 221
HPREG procedure, 292
Toeplitz structure
HPLMIXED procedure, 203

unstructured correlations
HPLMIXED procedure, 203
unstructured covariance matrix
HPLMIXED procedure, 203

validation
HPREG procedure, 286
variance components
HPLMIXED procedure, 203
variance ratios
HPLMIXED procedure, 200

weighting
HPGENSELECT procedure, 95
HPLOGISTIC procedure, 151
HPREG procedure, 281

zero inflation link function
HPGENSELECT procedure, 96
zero-inflation parameter estimates
HPGENSELECT procedure, 115

Syntax Index

ABSCONV option
- PROC HPGENSELECT statement, 80
- PROC HPLMIXED statement, 194
- PROC HPLOGISTIC statement, 137
- PROC HPNLMOD statement, 233

ABSFCONV option
- PROC HPGENSELECT statement, 80
- PROC HPLMIXED statement, 194
- PROC HPLOGISTIC statement, 137
- PROC HPNLMOD statement, 233

ABSGCONV option
- PROC HPGENSELECT statement, 81
- PROC HPLMIXED statement, 194
- PROC HPLOGISTIC statement, 138
- PROC HPNLMOD statement, 233

ABSGTOL option
- PROC HPGENSELECT statement, 81
- PROC HPLMIXED statement, 194
- PROC HPLOGISTIC statement, 138
- PROC HPNLMOD statement, 233

ABSOLUTE option
- PROC HPLMIXED statement, 219

ABSTOL option
- PROC HPGENSELECT statement, 80
- PROC HPLMIXED statement, 194
- PROC HPLOGISTIC statement, 137
- PROC HPNLMOD statement, 233

ADAPTIVE option
- SELECTION statement (high-performance statistical procedures), 46

ALPHA= option
- ESTIMATE statement (HPNLMOD), 238
- MODEL statement (HPGENSELECT), 89
- MODEL statement (HPLOGISTIC), 146
- OUTPUT statement (HPGENSELECT), 93
- PREDICT statement (HPNLMOD), 242
- PROC HPGENSELECT statement, 81
- PROC HPLOGISTIC statement, 138
- PROC HPNLMOD statement, 233
- PROC HPREG statement, 273
- RANDOM statement (HPLMIXED), 202

ASSOCIATION option
- MODEL statement (HPLOGISTIC), 146

BLUP option
- PROC HPLMIXED statement, 195

BOUNDS statement
- HPNLMOD procedure, 237

BY statement
- HPLOGISTIC procedure, 141
- HPNLMOD procedure, 237
- HPREG procedure, 274

CHOOSE= option
- MODEL statement (high-performance statistical procedures), 46

CL option
- MODEL statement (HPGENSELECT), 89
- MODEL statement (HPLMIXED), 199
- MODEL statement (HPLOGISTIC), 146
- RANDOM statement (HPLMIXED), 202

CLASS statement
- high-performance statistical procedures, 40
- HPGENSELECT procedure, 84
- HPLMIXED procedure, 198
- HPLOGISTIC procedure, 142
- HPREG procedure, 275
- MIXED procedure, 218

CLB option
- MODEL statement (HPREG), 276

CODE statement
- HPGENSELECT procedure, 85
- HPGENSELECT procedure, 85
- HPLOGISTIC procedure, 142
- HPREG procedure, 275

COMMIT= option
- PERFORMANCE statement (high-performance analytical procedures), 34

COMPETITIVE option
- SELECTION statement (high-performance statistical procedures), 46

CONVF option
- PROC HPLMIXED statement, 219

CONVG option
- PROC HPLMIXED statement, 219

CONVH option
- PROC HPLMIXED statement, 219

COPYVAR= option
- OUTPUT statement (HPLOGISTIC), 149
- OUTPUT statement (HPREG), 280

CORR option
- PROC HPGENSELECT statement, 81
- PROC HPNLMOD statement, 233

COV option
- PROC HPGENSELECT statement, 81
- PROC HPNLMOD statement, 233

CRITERION statement
HPSPLIT procedure, 316
CRITERION= option
SELECTION statement (high-performance statistical procedures), 46

DATA= option
OUTPUT statement (HPGENSELECT), 92
OUTPUT statement (HPLOGISTIC), 149
OUTPUT statement (HPREG), 280
PROC HPGENSELECT statement, 81
PROC HPLMIXED statement, 195
PROC HPLOGISTIC statement, 138
PROC HPNLMOD statement, 233
PROC HPREG statement, 273
DATASERVER= option
PERFORMANCE statement (high-performance analytical procedures), 34
DDFM= option
MODEL statement (HPLMIXED), 199
MODEL statement (HPLOGISTIC), 146
DESCENDING option
CLASS statement (high-performance statistical procedures), 40
MODEL statement (HPGENSELECT), 87
MODEL statement (HPLOGISTIC), 144
DETAILS option
PERFORMANCE statement (high-performance analytical procedures), 35
DETAILS= option
SELECTION statement (high-performance statistical procedures), 48
DF= option
ESTIMATE statement (HPNLMOD), 238
PREDICT statement (HPNLMOD), 242
PROC HPNLMOD statement, 234
DISPERSION= option
MODEL statement (HPGENSELECT), 89
DISTRIBUTION= option
MODEL statement (HPGENSELECT), 89

ECORR option
PROC HPNLMOD statement, 234
ECOV option
PROC HPNLMOD statement, 234
EQCONS= option
PARMS statement (HPLMIXED), 200
ESTIMATE statement
HPNLMOD procedure, 238

FAST option
SELECTION statement (high-performance statistical procedures), 47
FCONV option
PROC HPGENSELECT statement, 81
PROC HPLMIXED statement, 195
PROC HPLOGISTIC statement, 138
PROC HPNLMOD statement, 234
FMTLIBXML= option
PROC HPGENSELECT statement, 82
PROC HPLOGISTIC statement, 138
PROC HPNLMOD statement, 234
PROC HPREG statement, 273
FRACTION option
HPREG procedure, PARTITION statement, 278
HPSPLIT procedure, PARTITION statement, 319
FREQ statement
high-performance statistical procedures, 44
HPGENSELECT procedure, 85
HPLOGISTIC procedure, 143
HPREG procedure, 276
FTOL option
PROC HPGENSELECT statement, 81
PROC HPLMIXED statement, 195
PROC HPLOGISTIC statement, 138
PROC HPNLMOD statement, 234

GCONV option
PROC HPGENSELECT statement, 82
PROC HPLMIXED statement, 195
PROC HPLOGISTIC statement, 139
PROC HPNLMOD statement, 234
GRIDHOST= option
PERFORMANCE statement (high-performance analytical procedures), 35
GRIDMODE= option
PERFORMANCE statement (high-performance analytical procedures), 35
GRIDTIMEOUT= option
PERFORMANCE statement (high-performance analytical procedures), 35
GTOL option
PROC HPGENSELECT statement, 82
PROC HPLMIXED statement, 195
PROC HPLOGISTIC statement, 139
PROC HPNLMOD statement, 234

HIERARCHY= option
SELECTION statement (high-performance statistical procedures), 49
high-performance analytical procedures, PERFORMANCE statement, 34
COMMIT= option, 34
DATASERVER= option, 34
DETAILS option, 35
GRIDHOST= option, 35
GRIDMODE= option, 35
GRIDTIMEOUT= option, 35
HOST= option, 35

INSTALL= option, 35
INSTALLLOC= option, 35
LASR= option, 35
LASRSERVER= option, 35
MODE= option, 35
NNODES= option, 36
NODES= option, 36
NTHREADS= option, 37
THREADS= option, 37
TIMEOUT= option, 35

high-performance statistical procedures
FREQ statement, 44
ID statement, 44
VAR statement, 50
WEIGHT statement, 50

high-performance statistical procedures, CLASS statement, 40
DESCENDING option, 40
MISSING option, 43
ORDER= option, 41
PARAM= option, 43
REF= option, 41
TRUNCATE= option, 44

high-performance statistical procedures, FREQ statement, 44

high-performance statistical procedures, ID statement, 44

high-performance statistical procedures, SELECTION statement, 45
ADAPTIVE option, 46
CHOOSE= option, 46
COMPETITIVE option, 46
CRITERION= option, 46
DETAILS= option, 48
HIERARCHY= option, 49
LSCOEFFS option, 47
MAXEFFECTS= option, 47
MAXSTEP= option, 47
METHOD= option, 45
MINEFFECTS= option, 47
SELECT= option, 47
SELECTION= option, 49
SLE= option, 47
SLENTRY= option, 47
SLS= option, 48
SLSTAY= option, 48
STOP= option, 48
STOPHORIZON= option, 49

high-performance statistical procedures, VAR statement, 50

high-performance statistical procedures, WEIGHT statement, 50

HOLD= option
PARMS statement (HPLMIXED), 200

HOST= option
PERFORMANCE statement (high-performance analytical procedures), 35

HPGENSELECT procedure, 79
CLASS statement, 84
CODE statement, 85
FREQ statement, 85
ID statement, 86
MODEL statement, 86
OUTPUT statement, 92
PERFORMANCE statement, 94
PROC HPGENSELECT statement, 79
SELECTION statement, 94
syntax, 79
WEIGHT statement, 95
ZEROMODEL statement, 96

HPGENSELECT procedure, CLASS statement, 84
UPCASE option, 85

HPGENSELECT procedure, CODE statement, 85

HPGENSELECT procedure, FREQ statement, 85

HPGENSELECT procedure, ID statement, 86

HPGENSELECT procedure, MODEL statement, 86
ALPHA= option, 89
CL option, 89
DESCENDING option, 87
DISPERSION= option, 89
DISTRIBUTION= option, 89
INCLUDE option, 90
INITIALPHI= option, 90
LINK= option, 90
NOCENTER option, 91
NOINT option, 91
OFFSET= option, 91
ORDER= option, 88
SAMPLEFRAC= option, 91
START option, 91

HPGENSELECT procedure, OUTPUT statement, 92
ALPHA= option, 93
DATA= option, 92
keyword= option, 92
OBSCAT option, 94
OUT= option, 92

HPGENSELECT procedure, PERFORMANCE statement, 94

HPGENSELECT procedure, PROC HPGENSELECT statement, 79
ABSCONV option, 80
ABSFCONV option, 80
ABSFTOL option, 80
ABSGCONV option, 81
ABSGTOL option, 81
ABSTOL option, 80
ALPHA= option, 81
CORR option, 81

COV option, 81
DATA= option, 81
FCONV option, 81
FMTLIBXML= option, 82
FTOL option, 81
GCONV option, 82
GTOL option, 82
ITDETAILS option, 82
ITSELECT option, 82
MAXFUNC= option, 82
MAXITER= option, 83
MAXTIME= option, 83
NAMELEN= option, 83
NOCLPRINT option, 83
NOPRINT option, 83
NORMALIZE= option, 83
NOSTDERR option, 83
SINGCHOL= option, 84
SINGSWEEP= option, 84
SINGULAR= option, 84
TECHNIQUE= option, 84
HPGENSELECT procedure, SELECTION statement, 94
HPGENSELECT procedure, WEIGHT statement, 95
HPGENSELECT procedure, ZEROMODEL statement, 96
INCLUDE option, 96
START option, 96
HPGENSELECT procedure, CODE statement, 85
HPLMIXED procedure, 192
PERFORMANCE statement, 201
PROC HPLMIXED statement, 193
syntax, 192
HPLMIXED procedure, CLASS statement, 198
HPLMIXED procedure, MODEL statement, 198
CL option, 199
DDFM= option, 199
NOINT option, 199
SOLUTION option, 199
HPLMIXED procedure, PARMS statement, 199
EQCONS= option, 200
HOLD= option, 200
LOWERB= option, 200
NOITER option, 201
PARMSDATA= option, 201
PDATA= option, 201
UPPERB= option, 201
HPLMIXED procedure, PERFORMANCE statement, 201
HPLMIXED procedure, PROC HPLMIXED statement
ABSCONV option, 194
ABSFCONV option, 194
ABSFTOL option, 194
ABSGCONV option, 194
ABSGTOL option, 194
ABSTOL option, 194
BLUP option, 195
DATA= option, 195
FCONV option, 195
FTOL option, 195
GCONV option, 195
GTOL option, 195
MAXCLPRINT= option, 196
MAXFUNC= option, 196
MAXITER= option, 196
MAXTIME= option, 196
METHOD= option, 196
NAMELEN= option, 197
NOCLPRINT option, 197
NOPRINT option, 197
SINGCHOL= option, 197
SINGSWEEP= option, 197
SINGULAR= option, 197
TECHNIQUE= option, 197
XCONV option, 197
XTOL option, 197
HPLMIXED procedure, RANDOM statement, 201
ALPHA= option, 202
CL option, 202
SOLUTION option, 202
SUBJECT= option, 202
TYPE= option, 203
HPLMIXED procedure, REPEATED statement, 208
SUBJECT= option, 208
TYPE= option, 208
HPLMIXEDC procedure, PROC HPLMIXED statement, 193
HPLOGISTIC procedure, 136
CLASS statement, 142
FREQ statement, 143
ID statement, 143
MODEL statement, 143
OUTPUT statement, 148
PERFORMANCE statement, 150
PROC HPLOGISTIC statement, 136
SELECTION statement, 150
syntax, 136
WEIGHT statement, 151
HPLOGISTIC procedure, BY statement, 141
HPLOGISTIC procedure, CLASS statement, 142
UPCASE option, 143
HPLOGISTIC procedure, FREQ statement, 143
HPLOGISTIC procedure, ID statement, 143
HPLOGISTIC procedure, MODEL statement, 143
ALPHA= option, 146
ASSOCIATION option, 146
CL option, 146
DDFM= option, 146

DESCENDING option, 144
INCLUDE option, 147
LACKFIT option, 147
LINK= option, 147
NOCHECK option, 148
NOINT option, 148
OFFSET= option, 148
ORDER= option, 145
RSQUARE option, 148
START option, 148
HPLOGISTIC procedure, OUTPUT statement, 148
COPYVAR= option, 149
DATA= option, 149
keyword= option, 149
OBSCAT option, 150
OUT= option, 149
HPLOGISTIC procedure, PERFORMANCE statement, 150
HPLOGISTIC procedure, PROC HPLOGISTIC statement, 136
ABSCONV option, 137
ABSFCONV option, 137
ABSFTOL option, 137
ABSGCONV option, 138
ABSGTOL option, 138
ABSTOL option, 137
ALPHA= option, 138
DATA= option, 138
FCONV option, 138
FMTLIBXML= option, 138
FTOL option, 138
GCONV option, 139
GTOL option, 139
ITDETAILS option, 139
ITSELECT option, 139
MAXFUNC= option, 139
MAXITER= option, 139, 140
MAXTIME= option, 140
NAMELEN= option, 140
NOCLPRINT option, 140
NOITPRINT option, 140
NOPRINT option, 140
NORMALIZE= option, 140
NOSTDERR option, 140
SINGCHOL= option, 141
SINGSWEEP= option, 141
SINGULAR= option, 141
TECHNIQUE= option, 141
HPLOGISTIC procedure, SELECTION statement, 150
HPLOGISTIC procedure, WEIGHT statement, 151
HPLOGISTIC procedure, CODE statement, 142
HPNLMOD procedure, 231
PERFORMANCE statement, 241
PROC HPNLMOD statement, 232
syntax, 231
HPNLMOD procedure, BOUNDS statement, 237
HPNLMOD procedure, BY statement, 237
HPNLMOD procedure, ESTIMATE statement, 238
ALPHA= option, 238
DF= option, 238
HPNLMOD procedure, MODEL statement, 238
HPNLMOD procedure, PARAMETERS statement, 239
HPNLMOD procedure, PERFORMANCE statement, 241
HPNLMOD procedure, PREDICT statement, 242
ALPHA= option, 242
DF= option, 242
LOWER= option, 242
PRED= option, 242
PROBT= option, 242
STDERR= option, 242
TVALUE= option, 242
UPPER= option, 242
HPNLMOD procedure, PROC HPNLMOD statement, 232
ABSCONV option, 233
ABSFCONV option, 233
ABSFTOL option, 233
ABSGCONV option, 233
ABSGTOL option, 233
ABSTOL option, 233
ALPHA= option, 233
CORR option, 233
COV option, 233
DATA= option, 233
DF= option, 234
ECORR option, 234
ECOV option, 234
FCONV option, 234
FMTLIBXML= option, 234
FTOL option, 234
GCONV option, 234
GTOL option, 234
MAXFUNC= option, 235
MAXITER= option, 235
MAXTIME= option, 235
NOITPRINT option, 236
NOPRINT option, 236
OUT= option, 236
SINGULAR= option, 236
TECHNIQUE= option, 236
XMLFORMAT= option, 236
HPNLMOD procedure, RESTRICT statement, 243
HPREG procedure
FREQ statement, 276
ID statement, 276
MODEL statement, 276

OUTPUT statement, 279
PARTITION statement, 278
PERFORMANCE statement, 278
PROC HPREG statement, 273
WEIGHT statement, 281
HPREG procedure, BY statement, 274
HPREG procedure, CLASS statement, 275
SPLIT option, 42
UPCASE option, 275
HPREG procedure, ID statement, 276
HPREG procedure, MODEL statement, 276
CLB option, 276
INCLUDE option, 277
NOINT option, 277
ORDERSELECT option, 277
START option, 277
STB option, 277
TOL option, 277
VIF option, 277
HPREG procedure, OUTPUT statement, 279
COPYVAR= option, 280
DATA= option, 280
keyword= option, 280
OUT= option, 280
HPREG procedure, PARTITION statement, 278
FRACTION option, 278
ROLEVAR= option, 278
HPREG procedure, PERFORMANCE statement, 278
HPREG procedure, PROC HPREG statement, 273
ALPHA= option, 273
DATA= option, 273
FMTLIBXML= option, 273
NAMELEN= option, 274
NOCLPRINT option, 274
NOPRINT option, 274
SEED= option, 274
HPREG procedure, SELECTION statement, 278
HPREG procedure, WEIGHT statement, 281
HPREG procedures, FREQ statement, 276
HPREG procedure, CODE statement, 275
HPSPLIT procedure
CRITERION statement, 316
ID statement, 317
INPUT statement, 317
OUTPUT statement, 318
PARTITION statement, 319
PERFORMANCE statement, 320
PROC HPSPLIT statement, 314
PRUNE statement, 320
RULES statement, 316, 322
SCORE statement, 322
TARGET statement, 322
HPSPLIT procedure, CRITERION statement, 316
HPSPLIT procedure, ID statement, 317
HPSPLIT procedure, INPUT statement, 317
HPSPLIT procedure, OUTPUT statement, 318
HPSPLIT procedure, PARTITION statement, 319
FRACTION option, 319
ROLEVAR= option, 319
HPSPLIT procedure, PERFORMANCE statement, 320
HPSPLIT procedure, PROC HPSPLIT statement, 314
HPSPLIT procedure, PRUNE statement, 320
HPSPLIT procedure, RULES statement, 316, 322
HPSPLIT procedure, SCORE statement, 322
HPSPLIT procedure, TARGET statement, 322

ID statement
high-performance statistical procedures, 44
HPGENSELECT procedure, 86
HPLOGISTIC procedure, 143
HPREG procedure, 276
HPSPLIT procedure, 317
INCLUDE option
MODEL statement (HPGENSELECT), 90
MODEL statement (HPLOGISTIC), 147
MODEL statement (HPREG), 277
ZEROMODEL statement (HPGENSELECT), 96
INITIALPHI= option
MODEL statement (HPGENSELECT), 90
INPUT statement
HPSPLIT procedure, 317
INSTALL= option
PERFORMANCE statement (high-performance analytical procedures), 35
INSTALLLOC= option
PERFORMANCE statement (high-performance analytical procedures), 35
ITDETAILS option
PROC HPGENSELECT statement, 82
PROC HPLOGISTIC statement, 139
ITSELECT option
PROC HPGENSELECT statement, 82
PROC HPLOGISTIC statement, 139

keyword= option
OUTPUT statement (HPGENSELECT), 92
OUTPUT statement (HPLOGISTIC), 149
OUTPUT statement (HPREG), 280

LACKFIT option
MODEL statement (HPLOGISTIC), 147
LASR= option
PERFORMANCE statement (high-performance analytical procedures), 35
LASRSERVER= option
PERFORMANCE statement (high-performance analytical procedures), 35
LINK= option
MODEL statement (HPGENSELECT), 90

MODEL statement (HPLOGISTIC), 147
LOWER= option
PREDICT statement (HPNLMOD), 242
LOWERB= option
PARMS statement (HPLMIXED), 200
LSCOEFFS option
SELECTION statement (high-performance statistical procedures), 47

MAXCLPRINT= option
PROC HPLMIXED statement, 196
MAXEFFECTS= option
SELECTION statement (high-performance statistical procedures), 47
MAXFUNC= option
PROC HPGENSELECT statement, 82
PROC HPLMIXED statement, 196
PROC HPLOGISTIC statement, 139
PROC HPNLMOD statement, 235
MAXITER= option
PROC HPGENSELECT statement, 83
PROC HPLMIXED statement, 196
PROC HPLOGISTIC statement, 139, 140
PROC HPNLMOD statement, 235
MAXSTEPS= option
SELECTION statement (high-performance statistical procedures), 47
MAXTIME= option
PROC HPGENSELECT statement, 83
PROC HPLMIXED statement, 196
PROC HPLOGISTIC statement, 140
PROC HPNLMOD statement, 235
METHOD= option
PROC HPLMIXED statement, 196
SELECTION statement (high-performance statistical procedures), 45
MINEFFECTS= option
SELECTION statement (high-performance statistical procedures), 47
MISSING option
CLASS statement (high-performance statistical procedures), 43
MIXED procedure, CLASS statement, 218
MIXED procedure, PROC HPLMIXED statement
ABSOLUTE option, 219
CONVF option, 219
CONVG option, 219
CONVH option, 219
MODE= option
PERFORMANCE statement (high-performance analytical procedures), 35
MODEL statement
HPGENSELECT procedure, 86
HPLMIXED procedure, 198
HPLOGISTIC procedure, 143
HPNLMOD procedure, 238
HPREG procedure, 276

NAMELEN= option
PROC HPGENSELECT statement, 83
PROC HPLMIXED statement, 197
PROC HPLOGISTIC statement, 140
PROC HPREG statement, 274
NNODES= option
PERFORMANCE statement (high-performance analytical procedures), 36
NOCENTER option
MODEL statement (HPGENSELECT), 91
NOCLPRINT option
PROC HPGENSELECT statement, 83
PROC HPLMIXED statement, 197
PROC HPLOGISTIC statement, 140
PROC HPREG statement, 274
NODES= option
PERFORMANCE statement (high-performance analytical procedures), 36
NOINT option
MODEL statement (HPGENSELECT), 91
MODEL statement (HPLMIXED), 199
MODEL statement (HPLOGISTIC), 148
MODEL statement (HPREG), 277
NOITER option
PARMS statement (HPLMIXED), 201
NOITPRINT option
PROC HPLOGISTIC statement, 140
PROC HPNLMOD statement, 236
NOPRINT option
PROC HPGENSELECT statement, 83
PROC HPLMIXED statement, 197
PROC HPLOGISTIC statement, 140
PROC HPNLMOD statement, 236
PROC HPREG statement, 274
NORMALIZE= option
PROC HPGENSELECT statement, 83
PROC HPLOGISTIC statement, 140
NOSTDERR option
PROC HPGENSELECT statement, 83
PROC HPLOGISTIC statement, 140
NTHREADS= option
PERFORMANCE statement (high-performance analytical procedures), 37

OBSCAT option
OUTPUT statement (HPGENSELECT), 94
OUTPUT statement (HPLOGISTIC), 150
OFFSET= option
MODEL statement (HPGENSELECT), 91
MODEL statement (HPLOGISTIC), 148

ORDER= option
- CLASS statement (high-performance statistical procedures), 41
- CLASS statement (HPREG), 42
- MODEL statement (HPGENSELECT), 88
- MODEL statement (HPLOGISTIC), 145

ORDERSELECT option
- MODEL statement (HPREG), 277

OUT= option
- OUTPUT statement (HPGENSELECT), 92
- OUTPUT statement (HPLOGISTIC), 149
- OUTPUT statement (HPREG), 280
- PROC HPNLMOD statement, 236

OUTPUT statement
- HPGENSELECT procedure, 92
- HPLOGISTIC procedure, 148
- HPREG procedure, 279
- HPSPLIT procedure, 318

PARAM= option
- CLASS statement (high-performance statistical procedures), 43

PARAMETERS statement
- HPNLMOD procedure, 239

PARMS statement
- HPLMIXED procedure, 199

PARMSDATA= option
- PARMS statement (HPLMIXED), 201

PARTITION statement
- HPREG procedure, 278
- HPSPLIT procedure, 319

PDATA= option
- PARMS statement (HPLMIXED), 201

PERFORMANCE statement
- high-performance analytical procedures, 34
- HPGENSELECT procedure, 94
- HPLMIXED procedure, 201
- HPLOGISTIC procedure, 150
- HPNLMOD procedure, 241
- HPREG procedure, 278
- HPSPLIT procedure, 320

PRED= option
- PREDICT statement (HPNLMOD), 242

PREDICT statement
- HPNLMOD procedure, 242

PROBT= option
- PREDICT statement (HPNLMOD), 242

PROC HPGENSELECT statement, *see* HPGENSELECT procedure

PROC HPLMIXED statement, *see* HPLMIXED procedure
- HPLMIXED procedure, 193

PROC HPLOGISTIC statement, *see* HPLOGISTIC procedure

PROC HPNLMOD statement, *see* HPNLMOD procedure
- HPNLMOD procedure, 232

PROC HPREG statement, *see* HPREG procedure
- HPREG procedure, 273

PROC HPSPLIT statement, *see* HPSPLIT procedure
- HPSPLIT procedure, 314

PRUNE statement
- HPSPLIT procedure, 320

RANDOM statement
- HPLMIXED procedure, 201

REF= option
- CLASS statement (high-performance statistical procedures), 41

REPEATED statement
- HPLMIXED procedure, 208

RESTRICT statement
- HPNLMOD procedure, 243

ROLEVAR= option
- HPREG procedure, PARTITION statement, 278
- HPSPLIT procedure, PARTITION statement, 319

RSQUARE option
- MODEL statement (HPLOGISTIC), 148

RULES statement
- HPSPLIT procedure, 316, 322

SAMPLEFRAC= option
- MODEL statement (HPGENSELECT), 91

SCORE statement
- HPSPLIT procedure, 322

SEED= option
- PROC HPREG statement, 274

SELECT= option
- SELECTION statement (high-performance statistical procedures), 47

SELECTION statement
- high-performance statistical procedures, 45
- HPGENSELECT procedure, 94
- HPLOGISTIC procedure, 150
- HPREG procedure, 278

SELECTION= option
- SELECTION statement (high-performance statistical procedures), 49

SINGCHOL= option
- PROC HPGENSELECT statement, 84
- PROC HPLMIXED statement, 197
- PROC HPLOGISTIC statement, 141

SINGSWEEP= option
- PROC HPGENSELECT statement, 84
- PROC HPLMIXED statement, 197
- PROC HPLOGISTIC statement, 141

SINGULAR= option
- PROC HPGENSELECT statement, 84

PROC HPLMIXED statement, 197
PROC HPLOGISTIC statement, 141
PROC HPNLMOD statement, 236
SLE= option
SELECTION statement (high-performance statistical procedures), 47
SLENTRY= option
SELECTION statement (high-performance statistical procedures), 47
SLS= option
SELECTION statement (high-performance statistical procedures), 48
SLSTAY= option
SELECTION statement (high-performance statistical procedures), 48
SOLUTION option
MODEL statement (HPLMIXED), 199
RANDOM statement (HPLMIXED), 202
START option
MODEL statement (HPGENSELECT), 91
MODEL statement (HPLOGISTIC), 148
MODEL statement (HPREG), 277
ZEROMODEL statement (HPGENSELECT), 96
STB option
MODEL statement (HPREG), 277
STDERR= option
PREDICT statement (HPNLMOD), 242
STOP= option
SELECTION statement (high-performance statistical procedures), 48
STOPHORIZON= option
SELECTION statement (high-performance statistical procedures), 49
SUBJECT= option
RANDOM statement (HPLMIXED), 202
REPEATED statement (HPLMIXED), 208
syntax
HPGENSELECT procedure, 79
HPLOGISTIC procedure, 136
HPNLMOD procedure, 231

TARGET statement
HPSPLIT procedure, 322
TECHNIQUE= option
PROC HPGENSELECT statement, 84
PROC HPLMIXED statement, 197
PROC HPLOGISTIC statement, 141
PROC HPNLMOD statement, 236
THREADS= option
PERFORMANCE statement (high-performance analytical procedures), 37
TIMEOUT= option
PERFORMANCE statement (high-performance analytical procedures), 35
TOL option
MODEL statement (HPREG), 277
TRUNCATE= option
CLASS statement (high-performance statistical procedures), 44
TVALUE= option
PREDICT statement (HPNLMOD), 242
TYPE= option
RANDOM statement (HPLMIXED), 203
REPEATED statement (HPLMIXED), 208

UPCASE option
CLASS statement (HPGENSELECT), 85
CLASS statement (HPLOGISTIC), 143
CLASS statement (HPREG), 275
UPPER= option
PREDICT statement (HPNLMOD), 242
UPPERB= option
PARMS statement (HPLMIXED), 201

VAR statement
high-performance statistical procedures, 50
VIF option
MODEL statement (HPREG), 277

WEIGHT statement
high-performance statistical procedures, 50
HPGENSELECT procedure, 95
HPLOGISTIC procedure, 151
HPREG procedure, 281

XCONV option
PROC HPLMIXED statement, 197
XMLFORMAT= option
PROC HPNLMOD statement, 236
XTOL option
PROC HPLMIXED statement, 197

ZEROMODEL statement
HPGENSELECT procedure, 96

Your Turn

We welcome your feedback.

- If you have comments about this book, please send them to `yourturn@sas.com`. Include the full title and page numbers (if applicable).
- If you have comments about the software, please send them to `suggest@sas.com`.