



THE
POWER
TO KNOW.

SAS/STAT[®] 13.1 User's Guide

The PLM Procedure

This document is an individual chapter from *SAS/STAT® 13.1 User's Guide*.

The correct bibliographic citation for the complete manual is as follows: SAS Institute Inc. 2013. *SAS/STAT® 13.1 User's Guide*. Cary, NC: SAS Institute Inc.

Copyright © 2013, SAS Institute Inc., Cary, NC, USA

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

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

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

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

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

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513-2414.

December 2013

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

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

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



Gain Greater Insight into Your SAS[®] Software with SAS Books.

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



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



Chapter 73

The PLM Procedure

Contents

Overview: PLM Procedure	6188
Basic Features	6188
PROC PLM Contrasted with Other SAS Procedures	6189
Getting Started: PLM Procedure	6190
Syntax: PLM Procedure	6197
PROC PLM Statement	6198
CODE Statement	6202
EFFECTPLOT Statement	6202
ESTIMATE Statement	6203
FILTER Statement	6204
LSMEANS Statement	6206
LSMESTIMATE Statement	6207
SCORE Statement	6208
SHOW Statement	6211
SLICE Statement	6213
TEST Statement	6213
WHERE Statement	6213
Details: PLM Procedure	6215
BY Processing and the PLM Procedure	6215
Analysis Based on Posterior Estimates	6215
Scoring Data Sets for Zero-Inflated Models	6217
User-Defined Formats and the PLM Procedure	6217
ODS Table Names	6218
ODS Graphics	6219
Examples: PLM Procedure	6220
Example 73.1: Scoring with PROC PLM	6220
Example 73.2: Working with Item Stores	6221
Example 73.3: Group Comparisons in an Ordinal Model	6224
Example 73.4: Posterior Inference for Binomial Data	6226
Example 73.5: BY-Group Processing	6231
Example 73.6: Comparing Multiple B-Splines	6236
Example 73.7: Linear Inference with Arbitrary Estimates	6242
References	6245

Overview: PLM Procedure

The PLM procedure performs postfitting statistical analyses for the contents of a SAS item store that was previously created with the STORE statement in some other SAS/STAT procedure. An item store is a special SAS-defined binary file format used to store and restore information with a hierarchical structure.

The statements available in the PLM procedure are designed to reveal the contents of the source item store via the Output Delivery System (ODS) and to perform postfitting tasks such as the following:

- testing hypotheses
- computing confidence intervals
- producing prediction plots
- scoring a new data set

The use of item stores and PROC PLM enables you to separate common postprocessing tasks, such as testing for treatment differences and predicting new observations under a fitted model, from the process of model building and fitting. A numerically expensive model fitting technique can be applied once to produce a source item store. The PLM procedure can then be called multiple times and the results of the fitted model analyzed without incurring the model fitting expenditure again.

The PLM procedure offers the most advanced postprocessing techniques available in SAS/STAT software. These techniques include step-down multiplicity adjustments for p -values, F tests with order restrictions, analysis of means (ANOM), and sampling-based linear inference based on Bayes posterior estimates.

The following procedures support the STORE statement for the generation of item stores that can be processed with the PLM procedure: GENMOD, GLIMMIX, GLM, GLMSELECT, LIFEREG, LOGISTIC, MIXED, ORTHOREG, PHREG, PROBIT, SURVEYLOGISTIC, SURVEYPHREG, and SURVEYREG. The RELIABILITY procedure in SAS/QC software also supports the STORE statement. For details about the STORE statement, see the section “[STORE Statement](#)” on page 508 of Chapter 19, “[Shared Concepts and Topics](#).”

Basic Features

The PLM procedure, unlike most SAS/STAT procedures, does not operate primarily on an input data set. Instead, the procedure requires you to specify an item store with the [RESTORE=](#) option in the [PROC PLM](#) statement. The item store contains the necessary information and context about the statistical model that was fit when the store was created. SAS data sets are used only to provide input information in some circumstances. For example, when scoring a data set or when computing least squares means with specially defined population margins. In other words, instead of reading raw data and fitting a model, the PLM procedure reads the results of a model having been fit.

In order to interact with the item store and to reveal its contents, the PLM procedure supports the [SHOW](#) statement which converts item store information into standard ODS tables for viewing and further processing.

The PLM procedure is sensitive to the contents of the item store. For example, if a BAYES statement was in effect when the item store was created, the posterior parameter estimates are saved to the item store so

that the PLM procedure can perform postprocessing tasks by taking the posterior distribution of estimable functions into account. As another example, for item stores that are generated by a mixed model procedure using the Satterthwaite or Kenward-Roger (Kenward and Roger 1997) degrees-of-freedom method, these methods continue to be available when the item store contents are processed with the PLM procedure.

Because the PLM procedure does not read data and does not fit a model, the processing time of this procedure is usually considerably less than the processing time of the procedure that generates the item store.

PROC PLM Contrasted with Other SAS Procedures

In contrast to other analytic procedures in SAS/STAT software, the PLM procedure does not use an input data set. Instead, it retrieves information from an item store.

Some of the statements in the PLM procedure are also available as postprocessing statements in other procedures. Table 73.1 lists SAS/STAT procedures that support the same postprocessing statements as PROC PLM does.

Table 73.1 SAS/STAT Procedures with Postprocessing Statements Similar to PROC PLM

	EFFECTPLOT	ESTIMATE	LSMEANS	LSMESTIMATE	SLICE	TEST
GENMOD	✓	✓*	✓	✓	✓	
GLIMMIX		✓*	✓*	✓*	✓	
GLM		✓*	✓*			✓*
LIFEREG	✓	✓	✓	✓	✓	✓
LOGISTIC	✓	✓	✓	✓	✓	✓*
MIXED		✓*	✓*	✓	✓	
ORTHOREG	✓	✓	✓	✓	✓	✓
PHREG		✓	✓	✓	✓	✓*
PROBIT	✓	✓	✓	✓	✓	✓
SURVEYLOGISTIC		✓	✓	✓	✓	✓*
SURVEYPHREG		✓	✓	✓	✓	✓
SURVEYREG		✓	✓	✓	✓	✓

Table entries marked with ✓ indicate procedures that support statements with the same functionality as in PROC PLM. Those entries marked with ✓* indicate procedures that support statements with same names but different syntaxes from PROC PLM. You can find the most comprehensive set of features for these statements in the PLM procedure. For example, the LSMEANS statement is available in all of the listed procedures. For example, the ESTIMATE statement available in the GENMOD, GLIMMIX, GLM and MIXED procedures does not support all options that PROC PLM supports, such as multiple rows and multiplicity adjustments.

The WHERE statement in other procedures enables you to conditionally select a subset of the observations from the input data set so that the procedure processes only the observations that meet the specified conditions. Since the PLM procedure does not use an input data set, the WHERE statement in the PLM procedure has different functionality. If the item store contains information about BY groups—that is, a BY statement was in effect when the item store was created—you can use the WHERE statement to select specific BY groups for the analysis. You can also use the FILTER statement in the PLM procedure to filter results from the ODS output and output data sets.

Getting Started: PLM Procedure

The following DATA step creates a data set from a randomized block experiment with a factorial treatment structure of factors A and B:

```
data BlockDesign;
  input block a b y @@;
  datalines;
  1 1 1 56 1 1 2 41
  1 2 1 50 1 2 2 36
  1 3 1 39 1 3 2 35
  2 1 1 30 2 1 2 25
  2 2 1 36 2 2 2 28
  2 3 1 33 2 3 2 30
  3 1 1 32 3 1 2 24
  3 2 1 31 3 2 2 27
  3 3 1 15 3 3 2 19
  4 1 1 30 4 1 2 25
  4 2 1 35 4 2 2 30
  4 3 1 17 4 3 2 18
  ;
```

The GLM procedure is used in the following statements to fit the model and to create a source item store for the PLM procedure:

```
proc glm data=BlockDesign;
  class block a b;
  model y = block a b a*b / solution;
  store sasuser.BlockAnalysis / label='PLM: Getting Started';
run;
```

The CLASS statement identifies the variables Block, A, and B as classification variables. The MODEL statement specifies the response variable and the model effects. The block effect models the design effect, and the a, b, and a*b effects model the factorial treatment structure. The STORE statement requests that the context and results of this analysis be saved to an item store named sasuser.BlockAnalysis. Because the SASUSER library is specified as the library name of the item store, the store will be available after the SAS session completes. The optional label in the STORE statement identifies the store in subsequent analyses with the PLM procedure.

Note that having BlockDesign as the name of the output store would not create a conflict with the input data set name, because data sets and item stores are saved as files of different types.

Figure 73.1 displays the results from the GLM procedure. The “Class Level Information” table shows the number of levels and their values for the three classification variables. The “Parameter Estimates” table shows the estimates and their standard errors along with *t* tests.

Figure 73.1 Class Variable Information, Fit Statistics, and Parameter Estimates

The GLM Procedure					
Class Level Information					
Class		Levels	Values		
block		4	1	2	3 4
a		3	1	2	3
b		2	1	2	
R-Square		Coeff Var	Root MSE		y Mean
0.848966		15.05578	4.654747		30.91667
Parameter		Estimate	Standard Error	t Value	Pr > t
Intercept		20.41666667 B	2.85043856	7.16	<.0001
block	1	17.00000000 B	2.68741925	6.33	<.0001
block	2	4.50000000 B	2.68741925	1.67	0.1148
block	3	-1.16666667 B	2.68741925	-0.43	0.6704
block	4	0.00000000 B	.	.	.
a	1	3.25000000 B	3.29140294	0.99	0.3391
a	2	4.75000000 B	3.29140294	1.44	0.1695
a	3	0.00000000 B	.	.	.
b	1	0.50000000 B	3.29140294	0.15	0.8813
b	2	0.00000000 B	.	.	.
a*b	1 1	7.75000000 B	4.65474668	1.66	0.1167
a*b	1 2	0.00000000 B	.	.	.
a*b	2 1	7.25000000 B	4.65474668	1.56	0.1402
a*b	2 2	0.00000000 B	.	.	.
a*b	3 1	0.00000000 B	.	.	.
a*b	3 2	0.00000000 B	.	.	.

The following statements invoke the PLM procedure and use sasuser.BlockAnalysis as the source item store:

```
proc plm restore=sasuser.BlockAnalysis;
run;
```

These statements produce [Figure 73.2](#). The “Store Information” table displays information that is gleaned from the source item store. For example, the store was created by the GLM procedure at the indicated time and date, and the input data set for the analysis was WORK.BLOCKDESIGN. The label used earlier in the STORE statement of the GLM procedure also appears as a descriptor in [Figure 73.2](#).

Figure 73.2 Default Information

The PLM Procedure	
Store Information	
Item Store	SASUSER.BLOCKANALYSIS
Label	PLM: Getting Started
Data Set Created From	WORK.BLOCKDESIGN
Created By	PROC GLM
Date Created	24OCT13:13:54:05
Response Variable	y
Class Variables	block a b
Model Effects	Intercept block a b a*b

Class Level Information		
Class	Levels	Values
block	4	1 2 3 4
a	3	1 2 3
b	2	1 2

The “Store Information” table also echoes partial information about the variables and model effects that are used in the analysis. The “Class Level Information” table is produced by the PLM procedure by default whenever the model contains effects that depend on CLASS variables.

The following statements request a display of the fit statistics and the parameter estimates from the source item store and a test of the treatment main effects and their interactions:

```
proc plm restore=sasuser.BlockAnalysis;
  show fit parms;
  test a b a*b;
run;
```

The statements produce [Figure 73.3](#). Notice that the estimates and standard errors in the “Parameter Estimates” table agree with the results displayed earlier by the GLM procedure, except for small differences in formatting.

Figure 73.3 Fit Statistics, Parameter Estimates, and Tests of Effects

The PLM Procedure		
Fit Statistics		
MSE	21.66667	
Error df	15	

Figure 73.3 continued

Parameter Estimates					
Effect	block	a	b	Estimate	Standard Error
Intercept				20.4167	2.8504
block	1			17.0000	2.6874
block	2			4.5000	2.6874
block	3			-1.1667	2.6874
block	4			0	.
a		1		3.2500	3.2914
a		2		4.7500	3.2914
a		3		0	.
b			1	0.5000	3.2914
b			2	0	.
a*b		1	1	7.7500	4.6547
a*b		1	2	0	.
a*b		2	1	7.2500	4.6547
a*b		2	2	0	.
a*b		3	1	0	.
a*b		3	2	0	.

Type III Tests of Model Effects				
Effect	Num DF	Den DF	F Value	Pr > F
a	2	15	7.54	0.0054
b	1	15	8.38	0.0111
a*b	2	15	1.74	0.2097

Since the main effects, but not the interaction are significant in this experiment, the subsequent analysis focuses on the main effects, in particular on the effect of variable A.

The following statements request the least squares means of the A effect along with their pairwise differences:

```
proc plm restore=sasuser.BlockAnalysis seed=3;
  lsmeans      a      / diff;
  lsmestimate a -1 1,
              1 1 -2 / uppertailed ftest;
run;
```

The **LSMESTIMATE** statement tests two linear combinations of the A least squares means: equality of the first two levels and whether the sum of the first two level effects equals twice the effect of the third level. The **FTEST** option in the **LSMESTIMATE** statement requests a joint *F* test for this two-row contrast. The **UPPERTAILED** option requests that the *F* test also be carried out under one-sided order restrictions. Since *F* tests under order restrictions (chi-bar-square statistic) require a simulation-based approach for the calculation of *p*-values, the random number stream is initialized with a seed value through the **SEED=** option in the **PROC PLM** statement.

The results of the **LSMEANS** and the **LSMESTIMATE** statement are shown in Figure 73.4.

Figure 73.4 LS-Means Related Inference for A Effect

The PLM Procedure							
a Least Squares Means							
a		Estimate	Standard Error	DF	t Value	Pr > t	
1		32.8750	1.6457	15	19.98	<.0001	
2		34.1250	1.6457	15	20.74	<.0001	
3		25.7500	1.6457	15	15.65	<.0001	
Differences of a Least Squares Means							
a	_a	Estimate	Standard Error	DF	t Value	Pr > t	
1	2	-1.2500	2.3274	15	-0.54	0.5991	
1	3	7.1250	2.3274	15	3.06	0.0079	
2	3	8.3750	2.3274	15	3.60	0.0026	
Least Squares Means Estimates							
Effect	Label	Estimate	Standard Error	DF	t Value	Tails	Pr > t
a	Row 1	1.2500	2.3274	15	0.54	Upper	0.2995
a	Row 2	15.5000	4.0311	15	3.85	Upper	0.0008
F Test for Least Squares Means Estimates							
Effect	Num DF	Den DF	F Value	Pr > F	ChiBar Sq Value	Pr > ChiBarSq	
a	2	15	7.54	0.0054	15.07	0.0001	

The least squares means for the three levels of variable A are 32.875, 34.125, and 25.75. The differences between the third level and the first and second levels are statistically significant at the 5% level (p -values of 0.0079 and 0.0026, respectively). There is no significant difference between the first two levels. The first row of the “Least Squares Means Estimates” table also displays the difference between the first two levels of factor A. Although the (absolute value of the) estimate and its standard error are identical to those in the “Differences of a Least Squares Means” table, the p -values do not agree because one-sided tests were requested in the LSMESTIMATE statement.

The “F Test” table in Figure 73.4 shows the two degree-of-freedom test for the linear combinations of the LS-means. The F value of 7.54 with p -value of 0.0054 represents the usual (two-sided) F test. Under the one-sided right-tailed order restriction imposed by the UPPERTAILED option, the ChiBarSq value of 15.07 represents the observed value of the chi-bar-square statistic of Silvapulle and Sen (2004). The associated p -value of 0.0001 was obtained by simulation.

Now suppose that you are interested in analyzing the relationship of the interaction cell means. (Typically this would not be the case in this example since the $a*b$ interaction is not significant; see [Figure 73.3](#).) The **SLICE** statement in the following PROC PLM run produces an F test of equality and all pair-wise differences of the interaction means for the subset (partition) where variable B is at level '1'. With ODS Graphics enabled, the pairwise differences are displayed in a diffogram.

```
ods graphics on;
proc plm restore=sasuser.BlockAnalysis;
  slice a*b / sliceby(b='1') diff;
run;
ods graphics off;
```

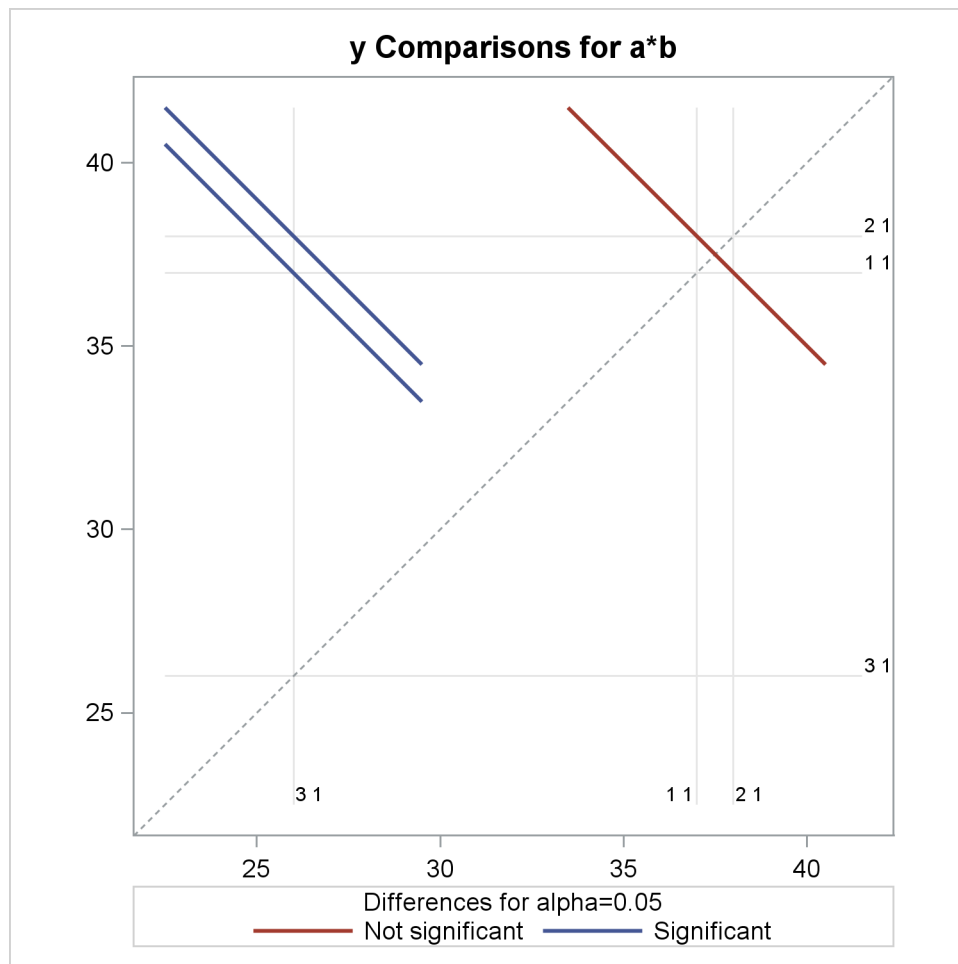
The results are shown in [Figure 73.5](#). Since variable A has three levels, the test of equality of the A means at level '1' of B is a two-degree comparison. This comparison is statistically significant (p -value equals 0.0040). You can conclude that the three levels of A are not the same for the first level of B.

Figure 73.5 Results from Analyzing an Interaction Partition

The PLM Procedure							
F Test for a*b Least Squares Means Slice							
Slice	Num DF	Den DF	F Value	Pr > F			
b 1	2	15	8.18	0.0040			
Simple Differences of a*b Least Squares Means							
Slice	a	_a	Estimate	Standard Error	DF	t Value	Pr > t
b 1	1	2	-1.0000	3.2914	15	-0.30	0.7654
b 1	1	3	11.0000	3.2914	15	3.34	0.0045
b 1	2	3	12.0000	3.2914	15	3.65	0.0024

The table of “Simple Differences” was produced by the DIFF option in the **SLICE** statement. As is the case with the marginal comparisons in [Figure 73.4](#), there are significant differences against the third level of A if variable B is held fixed at '1'.

[Figure 73.6](#) shows the diffogram that displays the three pairwise least squares mean differences and their significance. Each line segment corresponds to a comparison. It centers at the least squares means in the pair with its length corresponding to the projected width of a confidence interval for the difference. If the variable B is held fixed at '1', both the first two levels are significantly different from the third level, but the difference between the first and the second level is not significant.

Figure 73.6 LS-Means Difference Diffogram

Syntax: PLM Procedure

The following statements are available in the PLM procedure:

```

PROC PLM RESTORE=item-store-specification < options > ;
CODE < options > ;
EFFECTPLOT < plot-type < (plot-definition-options) > > < / options > ;
ESTIMATE < 'label' > estimate-specification < (divisor=n) >
    < , ... < 'label' > estimate-specification < (divisor=n) > > < / options > ;
FILTER expression ;
LSMEANS < model-effects > < / options > ;
LSMESTIMATE model-effect < 'label' > values < divisor=n >
    < , ... < 'label' > values < divisor=n > > < / options > ;
SCORE DATA=SAS-data-set < OUT=SAS-data-set >
    < keyword<=name> > ...
    < keyword<=name> > < / options > ;
SHOW options ;
SLICE model-effect < / options > ;
TEST < model-effects > < / options > ;
WHERE expression ;

```

With the exception of the **PROC PLM** statement and the **FILTER** statement, any statement can appear multiple times and in any order. The default order in which the statements are processed by the PLM procedure depends on the specification in the item store and can be modified with the **STMTORDER=** option in the **PROC PLM** statement.

In contrast to many other SAS/STAT modeling procedures, the PLM procedure does not have common modeling statements such as the **CLASS** and **MODEL** statements. This is because the information about classification variables and model effects is contained in the source item store that is passed to the procedure in the **PROC PLM** statement. All subsequent statements are checked for consistency with the stored model. For example, the statement

```
lsmeans c / diff;
```

is detected as not valid unless one of the following conditions was true at the time when the source store was created:

- The effect C was used in the model.
- C was specified in the **CLASS** statement.
- The **CLASS** variables in the model had a GLM parameterization.

The **FILTER**, **SCORE**, **SHOW**, and **WHERE** statements are described in full after the **PROC PLM** statement in alphabetical order. The **CODE**, **EFFECTPLOT**, **ESTIMATE**, **LSMEANS**, **LSMESTIMATE**, **SLICE**, and **TEST** statements are also used by many other procedures. Summary descriptions of functionality and syntax for these statements are also given after the **PROC PLM** statement in alphabetical order, but full documentation about them is available in Chapter 19, “Shared Concepts and Topics.”

PROC PLM Statement

PROC PLM RESTORE=*item-store-specification* < options > ;

The PROC PLM statement invokes the PLM procedure. The **RESTORE=** option with *item-store-specification* is required. Table 73.2 summarizes the *options* available in the PROC PLM statement.

Table 73.2 PROC PLM Statement Options

Option	Description
Basic Options	
RESTORE=	Specifies the source item store for processing
SEED=	Specifies the random number seed
STMTORDER=	Affects the order in which statements are grouped during processing
FORMAT=	Specifies how the PLM procedure handles user-defined formats
WHEREFORMAT	Specifies the constants (literals) in terms of the formatted values of the BY variables
Computational Options	
ALPHA=	Specifies the nominal significance level
DDFMETHOD=	Specifies the method for determining denominator degrees of freedom
PERCENTILES=	Supplies a list of percentiles for the construction of HPD intervals
Displayed Output	
MAXLEN=	Determines the maximum length of informational strings
NOCLPRINT	Suppresses the display of the “Class Level Information” table
NOINFO	Suppresses the display of the “Store Information” table
NOPRINT	Suppresses tabular and graphical output
PLOT	Controls the plots produced through ODS Graphics
Singularity Tolerances	
ESTEPS=	Specifies the tolerance value used in determining the estimability of linear functions
SINGCHOL=	Tunes the singularity criterion in Cholesky decompositions
SINGRES=	Sets the tolerance for which the residual variance or scale parameter is considered to be zero
SINGULAR=	Tunes the general singularity criterion
ZETA=	Tunes the sensitivity in forming Type III functions

You can specify the following *options*:

ALPHA= α

specifies the nominal significance level for multiplicity corrections and for the construction of confidence intervals. The value of α must be between 0 and 1. The default is the value specified in the source item store, or 0.05 if the item store does not provide a value. The confidence level based on α is $1 - \alpha$.

DDFMETHOD=RESIDUAL | RES | ERROR

DDFMETHOD=NONE

DDFMETHOD=KENROG | KR | KENWARDROGER

DDFMETHOD=SATTERTH | SAT | SATTERTHWAITE

specifies the method for determining denominator degrees of freedom for tests and confidence intervals. The default degree-of-freedom method is determined by the contents of the item store. You can override the default to some extent with the DDFMETHOD= option.

If you choose DDFMETHOD=NONE, then infinite denominator degrees of freedom are assumed for tests and confidence intervals. This essentially produces z tests and intervals instead of t tests and intervals and chi-square tests instead of F tests.

The KENWARDROGER and SATTERTHWAITE methods require that the source item store contain information about these methods. This information is currently available for item stores that were created with the MIXED or GLIMMIX procedures when the appropriate DDFM= option was in effect.

ESTEPS= ϵ

specifies the tolerance value used in determining the estimability of linear functions. The default value is determined by the contents of the source item store; it is usually 1E-4.

FORMAT=NOLOAD | RELOAD

specifies how the PLM procedure handles user-defined formats, which are not permanent. When the item store is created, user-defined formats are stored. When the PLM procedure opens an item store, it uses this option as follows. If FORMAT=RELOAD (the default), the stored formats are loaded again from the item store and formats that already exist in your SAS session are replaced by the reloaded formats. If FORMAT=NOLOAD, stored formats are not loaded from the item store and existing formats are not replaced.

With FORMAT=NOLOAD, you prevent the PLM procedure from reloading the format from the item store. As a consequence, PLM statements might fail if a format was present at the item store creation and is not available in your SAS session. Also, if you modify the format that was used in the item store creation and use FORMAT=NOLOAD, you might obtain unexpected results because levels of classification variables are remapped.

The “Class Level Information” table always displays the formatted values of classification variables that were used in fitting the model, regardless of the FORMAT= option. For more details about using formats with the PLM procedure, see [“User-Defined Formats and the PLM Procedure”](#) on page 6217.

MAXLEN= n

determines the maximum length of informational strings in the “Store Information” table. This table displays, for example, lists of classification or BY variables and lists of model effects. The value of n determines the truncation length for these strings. The minimum and maximum values for n are 20 and 256, respectively. The default is $n = 100$.

NOCLPRINT<= $number$ >

suppresses the display of the “Class Level Information” table if you do not specify $number$. If you specify $number$, only levels with totals that are less than $number$ are listed in the table. The PLM procedure produces the “Class Level Information” table by default when the model contains effects that depend on classification variables.

NOINFO

suppresses the display of the “Store Information” table.

NOPRINT

suppresses the generation of tabular and graphical output. When the NOPRINT option is in effect, ODS tables are also not produced.

PERCENTILES=*value-list***PERCENTILE=***value-list*

supplies a list of percentiles for the construction of highest posterior density (HPD) intervals when the PLM procedure performs a sampling-based analysis (for example, when processing an item store that contains posterior parameter estimates from a Bayesian analysis). The default set of percentiles depends on the contents of the source item store; it is typically PERCENTILES=25, 50, 75. The entries in *value-list* must be strictly between 0 and 100.

PLOTS <(*global-plot-option*)> <=*specific-plot-options*>

controls the plots produced through ODS Graphics. ODS Graphics must be enabled before plots can be requested. For example:

```
ods graphics on;

proc plm plots=all;
    lsmeans a/diff;
run;

ods graphics off;
```

For more information about enabling and disabling ODS Graphics, see the section “[Enabling and Disabling ODS Graphics](#)” on page 606 in Chapter 21, “[Statistical Graphics Using ODS](#).”

Global Plot Option

The following *global-plot-option* applies to all plots produced by [PROC PLM](#).

UNPACKPANEL**UNPACK**

suppresses paneling. (By default, multiple plots can appear in some output panels.) Specify UNPACK to display each plot separately.

Specific Plot Options

You can specify the following *specific-plot-options*:

ALL

requests that all the appropriate plots be produced.

NONE

suppresses all plots.

SEED=number

specifies the random number seed for analyses that depend on a random number stream. You can also specify the random number seed through some PLM statements (for example, through the SEED= options in the ESTIMATE, LSMEANS, and LSMESTIMATE statements). However, note that there is only a single random number stream per procedure run. Specifying the SEED= option in the PROC PLM statement initializes the stream for all subsequent statements. If you do not specify a random number seed, the source item store might supply one for you. If a seed is in effect when the PLM procedure opens the source store, the “Store Information” table displays its value.

If the random number seed is less than or equal to zero, the seed is generated from reading the time of day from the computer clock and a log message indicates the chosen seed value.

SINGCHOL=number

tunes the singularity criterion in Cholesky decompositions. The default value depends on the contents of the source item store. The default value is typically 1E4 times the machine epsilon; this product is approximately 1E–12 on most computers.

SINGRES=number

sets the tolerance for which the residual variance or scale parameter is considered to be zero. The default value depends on the contents of the source item store. The default value is typically 1E4 times the machine epsilon; this product is approximately 1E–12 on most computers.

SINGULAR=number

tunes the general singularity criterion applied by the PLM procedure in divisions and inversions. The default value used by the PLM procedure depends on the contents of the item store. The default value is typically 1E4 times the machine epsilon; this product is approximately 1E–12 on most computers.

RESTORE=item-store-specification

specifies the source item store for processing. This option is required because, in contrast to SAS data sets, there is no default item store. An *item-store-specification* consists of a one- or two-level name as with SAS data sets. As with data sets, the default library association of an item store is with the WORK library, and any stores created in this library are deleted when the SAS session concludes.

STMTORDER=SYNTAX | GROUP**STMT=SYNTAX | GROUP**

affects the order in which statements are grouped during processing. The default behavior depends on the contents of the source item store and can be modified with the STMTORDER= option. If STMTORDER=SYNTAX is in effect, the statements are processed in the order in which they appear. Note that this precludes the hierarchical grouping of ODS objects. If STMTORDER=GROUP is in effect, the statements are processed in groups and in the following order: **SHOW**, **TEST**, **LSMEANS**, **SLICE**, **LSMESTIMATE**, **ESTIMATE**, **SCORE**, **EFFECTPLOT**, and **CODE**.

WHEREFORMAT

specifies that the constants (literals) specified in **WHERE** expressions for group selection are in terms of the formatted values of the BY variables. By default, WHERE expressions are specified in terms of the unformatted (raw) values of the BY variables, as in the SAS DATA step.

ZETA=number

tunes the sensitivity in forming Type III functions. Any element in the estimable function basis with an absolute value less than *number* is set to 0. The default depends on the contents of the source item store; it usually is 1E–8.

CODE Statement

CODE < options > ;

The CODE statement writes SAS DATA step code for computing predicted values of the fitted model either to a file or to a catalog entry. This code can then be included in a DATA step to score new data.

Table 73.3 summarizes the *options* available in the CODE statement.

Table 73.3 CODE Statement Options

Option	Description
CATALOG=	Names the catalog entry where the generated code is saved
DUMMIES	Retains the dummy variables in the data set
ERROR	Computes the error function
FILE=	Names the file where the generated code is saved
FORMAT=	Specifies the numeric format for the regression coefficients
GROUP=	Specifies the group identifier for array names and statement labels
IMPUTE	Imputes predicted values for observations with missing or invalid covariates
LINESIZE=	Specifies the line size of the generated code
LOOKUP=	Specifies the algorithm for looking up CLASS levels
RESIDUAL	Computes residuals

For details about the syntax of the CODE statement, see the section “CODE Statement” on page 395 in Chapter 19, “Shared Concepts and Topics.”

EFFECTPLOT Statement

EFFECTPLOT < plot-type < (plot-definition-options) > > < / options > ;

The EFFECTPLOT statement produces a display of the fitted model and provides options for changing and enhancing the displays. Table 73.4 describes the available *plot-types* and their *plot-definition-options*.

Table 73.4 Plot-Types and Plot-Definition-Options

Plot-Type and Description	Plot-Definition-Options
BOX Displays a box plot of continuous response data at each level of a CLASS effect, with predicted values superimposed and connected by a line. This is an alternative to the INTERACTION plot-type.	PLOTBY= variable or CLASS effect X= CLASS variable or effect
CONTOUR Displays a contour plot of predicted values against two continuous covariates	PLOTBY= variable or CLASS effect X= continuous variable Y= continuous variable
FIT Displays a curve of predicted values versus a continuous variable	PLOTBY= variable or CLASS effect X= continuous variable
INTERACTION Displays a plot of predicted values (possibly with error bars) versus the levels of a CLASS effect. The predicted values are connected with lines and can be grouped by the levels of another CLASS effect.	PLOTBY= variable or CLASS effect SLICEBY= variable or CLASS effect X= CLASS variable or effect
MOSAIC Displays a mosaic plot of predicted values by using up to three CLASS effects	PLOTBY= variable or CLASS effect X= CLASS effects
SLICEFIT Displays a curve of predicted values versus a continuous variable, grouped by the levels of a CLASS effect	PLOTBY= variable or CLASS effect SLICEBY= variable or CLASS effect X= continuous variable

For full details about the syntax and options of the EFFECTPLOT statement, see the section “**EFFECTPLOT Statement**” on page 416 in Chapter 19, “**Shared Concepts and Topics**.”

ESTIMATE Statement

```
ESTIMATE <'label'> estimate-specification <(divisor=n)>
    < , ... <'label'> estimate-specification <(divisor=n)> >
    </ options> ;
```

The ESTIMATE statement provides a mechanism for obtaining custom hypothesis tests. Estimates are formed as linear estimable functions of the form $L\beta$. You can perform hypothesis tests for the estimable functions, construct confidence limits, and obtain specific nonlinear transformations.

Table 73.5 summarizes the *options* available in the ESTIMATE statement.

Table 73.5 ESTIMATE Statement Options

Option	Description
Construction and Computation of Estimable Functions	
DIVISOR=	Specifies a list of values to divide the coefficients
NOFILL	Suppresses the automatic fill-in of coefficients for higher-order effects
SINGULAR=	Tunes the estimability checking difference
Degrees of Freedom and p-values	
ADJUST=	Determines the method for multiple comparison adjustment of estimates
ALPHA= α	Determines the confidence level ($1 - \alpha$)
LOWER	Performs one-sided, lower-tailed inference
STEPPDOWN	Adjusts multiplicity-corrected p -values further in a step-down fashion
TESTVALUE=	Specifies values under the null hypothesis for tests
UPPER	Performs one-sided, upper-tailed inference
Statistical Output	
CL	Constructs confidence limits
CORR	Displays the correlation matrix of estimates
COV	Displays the covariance matrix of estimates
E	Prints the L matrix
JOINT	Produces a joint F or chi-square test for the estimable functions
PLOTS=	Requests ODS statistical graphics if the analysis is sampling-based
SEED=	Specifies the seed for computations that depend on random numbers
Generalized Linear Modeling	
CATEGORY=	Specifies how to construct estimable functions with multinomial data
EXP	Exponentiates and displays estimates
ILINK	Computes and displays estimates and standard errors on the inverse linked scale

For details about the syntax of the ESTIMATE statement, see the section “[ESTIMATE Statement](#)” on page 444 in Chapter 19, “[Shared Concepts and Topics](#).”

FILTER Statement

FILTER *expression* ;

The FILTER statement enables you to filter the results of the PLM procedure, specifically the contents of ODS tables and the output data sets. There can be at most one FILTER statement per PROC PLM run, and the filter is applied to all BY groups and to all queries generated through [WHERE](#) expressions.

A filter *expression* follows the same pattern as a *where-expression* in the [WHERE](#) statement. The expressions consist of operands and operators. For more information about specifying *where-expressions*, see the [WHERE](#) statement for the PLM procedure and *SAS Language Reference: Concepts*.

Valid *keywords* for the formation of operands in the FILTER statement are shown in [Table 73.6](#).

Table 73.6 Keywords for Filtering Results

Keyword	Description
Prob	Regular (unadjusted) p -values from t , F , or chi-square tests
ProbChi	Regular (unadjusted) p -values from chi-square tests
ProbF	Regular (unadjusted) p -values from F tests
ProbT	Regular (unadjusted) p -values from t tests
AdjP	Adjusted p -values
Estimate	Results displayed in “Estimates” column of ODS tables
Pred	Predicted values in SCORE output data sets
Resid	Residuals in SCORE output data sets.
Std	Standard errors in ODS tables and in SCORE results
Mu	Results displayed in the “Mean” column of ODS tables (this column is typically produced by the ILINK option)
tValue	The value of the usual t statistic
FValue	The value of the usual F statistic
Chisq	The value of the chi-square statistic
testStat	The value of the test statistic (a generic <i>keyword</i> for the ‘tValue’, ‘FValue’, and ‘Chisq’ tokens)
Lower	The lower confidence limit displayed in ODS tables
Upper	The upper confidence limit displayed in ODS tables
AdjLower	The adjusted lower confidence limit displayed in ODS tables
AdjUpper	The adjusted upper confidence limit displayed in ODS tables
LowerMu	The lower confidence limit for the mean displayed in ODS tables
UpperMu	The upper confidence limit for the mean displayed in ODS tables
AdjLowerMu	The adjusted lower confidence limit for the mean displayed in ODS tables
AdjUpperMu	The adjusted upper confidence limit for the mean displayed in ODS tables

When you write filtering expressions, be advised that filtering variables that are not used in the results are typically set to missing values. For example, the following statements select all results (filter nothing) because no adjusted p -values are computed:

```
proc plm restore=MyStore;
  lsmeans a / diff;
  filter adjp < 0.05;
run;
```

If the adjusted p -values are set to missing values, the condition `adjp < 0.05` is true in each case (missing values always compare smaller than the smallest nonmissing value).

See “[Example 73.6: Comparing Multiple B-Splines](#)” on page 6236 for an example of using the `FILTER` statement.

Filtering results has no effect on the item store contents that are displayed with the `SHOW` statement. However, BY-group selection with the `WHERE` statement can limit the amount of information that is displayed by the `SHOW` statements.

LSMEANS Statement

LSMEANS < *model-effects* > < / *options* > ;

The LSMEANS statement computes and compares least squares means (LS-means) of fixed effects. LS-means are *predicted population margins*—that is, they estimate the marginal means over a balanced population. In a sense, LS-means are to unbalanced designs as class and subclass arithmetic means are to balanced designs.

Table 73.7 summarizes the *options* available in the LSMEANS statement.

Table 73.7 LSMEANS Statement Options

Option	Description
Construction and Computation of LS-Means	
<code>AT</code>	Modifies the covariate value in computing LS-means
<code>BYLEVEL</code>	Computes separate margins
<code>DIFF</code>	Requests differences of LS-means
<code>OM=</code>	Specifies the weighting scheme for LS-means computation as determined by the input data set
<code>SINGULAR=</code>	Tunes estimability checking
Degrees of Freedom and <i>p</i>-values	
<code>ADJUST=</code>	Determines the method for multiple-comparison adjustment of LS-means differences
<code>ALPHA=α</code>	Determines the confidence level ($1 - \alpha$)
<code>STEPDOWN</code>	Adjusts multiple-comparison <i>p</i> -values further in a step-down fashion
Statistical Output	
<code>CL</code>	Constructs confidence limits for means and mean differences
<code>CORR</code>	Displays the correlation matrix of LS-means
<code>COV</code>	Displays the covariance matrix of LS-means
<code>E</code>	Prints the L matrix
<code>LINES</code>	Produces a “Lines” display for pairwise LS-means differences
<code>MEANS</code>	Prints the LS-means
<code>PLOTS=</code>	Requests graphs of means and mean comparisons
<code>SEED=</code>	Specifies the seed for computations that depend on random numbers

Table 73.7 *continued*

Option	Description
Generalized Linear Modeling	
EXP	Exponentiates and displays estimates of LS-means or LS-means differences
ILINK	Computes and displays estimates and standard errors of LS-means (but not differences) on the inverse linked scale
ODDSRATIO	Reports (simple) differences of least squares means in terms of odds ratios if permitted by the link function

For details about the syntax of the LSMEANS statement, see the section “[LSMEANS Statement](#)” on page 460 in Chapter 19, “[Shared Concepts and Topics](#).”

LSMESTIMATE Statement

```
LSMESTIMATE model-effect <'label'> values < divisor=n >
              < , ... <'label'> values < divisor=n > >
              < / options > ;
```

The LSMESTIMATE statement provides a mechanism for obtaining custom hypothesis tests among least squares means.

Table 73.8 summarizes the *options* available in the LSMESTIMATE statement.

Table 73.8 LSMESTIMATE Statement Options

Option	Description
Construction and Computation of LS-Means	
AT	Modifies covariate values in computing LS-means
BYLEVEL	Computes separate margins
DIVISOR=	Specifies a list of values to divide the coefficients
OM=	Specifies the weighting scheme for LS-means computation as determined by a data set
SINGULAR=	Tunes estimability checking
Degrees of Freedom and <i>p</i>-values	
ADJUST=	Determines the method for multiple-comparison adjustment of LS-means differences
ALPHA= α	Determines the confidence level ($1 - \alpha$)
LOWER	Performs one-sided, lower-tailed inference
STEPDOWN	Adjusts multiple-comparison <i>p</i> -values further in a step-down fashion
TESTVALUE=	Specifies values under the null hypothesis for tests
UPPER	Performs one-sided, upper-tailed inference

Table 73.8 *continued*

Option	Description
Statistical Output	
CL	Constructs confidence limits for means and mean differences
CORR	Displays the correlation matrix of LS-means
COV	Displays the covariance matrix of LS-means
E	Prints the L matrix
ELSM	Prints the K matrix
JOINT	Produces a joint <i>F</i> or chi-square test for the LS-means and LS-means differences
PLOTS=	Requests graphs of means and mean comparisons
SEED=	Specifies the seed for computations that depend on random numbers
Generalized Linear Modeling	
CATEGORY=	Specifies how to construct estimable functions with multinomial data
EXP	Exponentiates and displays LS-means estimates
ILINK	Computes and displays estimates and standard errors of LS-means (but not differences) on the inverse linked scale

For details about the syntax of the LSMESTIMATE statement, see the section “[LSMESTIMATE Statement](#)” on page 476 in Chapter 19, “[Shared Concepts and Topics](#).”

SCORE Statement

```
SCORE DATA=SAS-data-set <OUT=SAS-data-set>
      < keyword=<name>> ...
      < keyword=<name>> </ options> ;
```

The SCORE statement applies the contents of the source item store to compute predicted values and other observation-wise statistics for a SAS data set.

You can specify the following syntax elements in the SCORE statement before the option slash (/):

DATA=SAS-data-set

specifies the input data set for scoring. This option is required, and the data set is examined for congruity with the previously fitted (and stored) model. For example, all necessary variables to form a row of the **X** matrix must be present in the input data set and must be of the correct type and format. The following variables do not have to be present in the input data set:

- the response variable
- the *events* and *trials* variables used in the *events/trials* syntax for binomial data
- variables used in WEIGHT or FREQ statements

OUT=SAS-data-set

specifies the name of the output data set. If you do not specify an output data set with the OUT= option, the PLM procedure uses the *DATA**n* convention to name the output data set.

keyword<=name>

specifies a statistic to be included in the OUT= data set and optionally assigns the statistic the variable name *name*. Table 73.9 lists the *keywords* and the default names assigned by the PLM procedure if you do not specify a *name*.

Table 73.9 Keywords for Output Statistics

Keyword	Description	Expression	Name
PREDICTED	Linear predictor	$\hat{\eta} = \mathbf{x}\hat{\boldsymbol{\beta}}$	Predicted
STDERR	Standard deviation of linear predictor	$\sqrt{\text{Var}(\hat{\eta})}$	StdErr
RESIDUAL	Residual	$y - g^{-1}(\hat{\eta})$	Resid
LCLM	Lower confidence limit for the linear predictor		LCLM
UCLM	Upper confidence limit for the linear predictor		UCLM
LCL	Lower prediction limit for the linear predictor		LCL
UCL	Upper prediction limit for the linear predictor		UCL
PZERO	zero-inflation probability for zero-inflated models	$g_z^{-1}(\mathbf{z}\hat{\boldsymbol{\gamma}})$	PZERO

Prediction limits (LCL, UCL) are available only for statistical models that allow such limits, typically regression-type models for normally distributed data with an identity link function. Zero-inflation probability (PZERO) is available only for zero-inflated models. For details on how PROC PLM computes statistics for zero-inflated models, see “[Scoring Data Sets for Zero-Inflated Models](#)” on page 6217.

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

ALPHA=number

determines the coverage probability for two-sided confidence and prediction intervals. The coverage probability is computed as $1 - \text{number}$. The value of *number* must be between 0 and 1; the default is 0.05.

DF=number

specifies the degrees of freedom to use in the construction of prediction and confidence limits.

ILINK

requests that predicted values be inversely linked to produce predictions on the data scale. By default, predictions are produced on the linear scale where covariate effects are additive.

NOOFFSET

requests that the offset values not be added to the prediction if the offset variable is used in the fitted model.

NOUNIQUE

requests that names not be made unique in the case of naming conflicts. By default, the PLM procedure avoids naming conflicts by assigning a unique name to each output variable. If you specify the NOUNIQUE option, variables with conflicting names are not renamed. In that case, the first variable added to the output data set takes precedence.

NOVAR

requests that variables from the input data set not be added to the output data set.

OBSCAT

requests that statistics in models for multinomial data be written to the output data set only for the response level that corresponds to the observed level of the observation.

SAMPLE

requests that the sample of parameter estimates in the item store be used to form scoring statistics. This option is useful when the item store contains the results of a Bayesian analysis and a posterior sample of parameter estimates. The predicted value is then computed as the average predicted value across the posterior estimates, and the standard error measures the standard deviation of these estimates. For example, let $\hat{\beta}_1, \dots, \hat{\beta}_k$ denote the k posterior sample estimates of β , and let $\mathbf{x}_i$ denote the $\mathbf{x}$ -vector for the i th observation in the scoring data set. If the SAMPLE option is in effect, the output statistics for the predicted value, the standard error, and the residual of the i th observation are computed as

$$\begin{aligned}\eta_{ij} &= \mathbf{x}_i \hat{\beta}_j \\ \text{PRED}_i &= \bar{\eta}_i = \frac{1}{k} \sum_{j=1}^k \eta_{ij} \\ \text{STDERR}_i &= \left(\frac{1}{k-1} \sum_{j=1}^k (\eta_{ij} - \bar{\eta}_i)^2 \right)^{1/2} \\ \text{RESIDUAL}_i &= y_i - g^{-1}(\bar{\eta}_i)\end{aligned}$$

where $g^{-1}(\cdot)$ denotes the inverse link function.

If, in addition, the **ILINK** option is in effect, the calculations are as follows:

$$\begin{aligned}\eta_{ij} &= \mathbf{x}_i \hat{\beta}_j \\ \text{PRED}_i &= \frac{1}{k} \sum_{j=1}^k g^{-1}(\eta_{ij}) \\ \text{STDERR}_i &= \left(\frac{1}{k-1} \sum_{j=1}^k (g^{-1}(\eta_{ij}) - \text{PRED}_i)^2 \right)^{1/2} \\ \text{RESIDUAL}_i &= y_i - \text{PRED}_i\end{aligned}$$

The LCL and UCL statistics are not available with the SAMPLE option. When the LCLM and UCLM statistics are requested, the SAMPLE option yields the lower $100 \times \alpha/2\%$ and upper $100 \times (1 - \alpha/2)\%$

percentiles of the predicted values under the sample (posterior) distribution. When you request residuals with the SAMPLE option, the calculation depends on whether the [ILINK](#) option is specified.

SHOW Statement

SHOW *options* ;

The SHOW statement uses the Output Delivery System to display contents of the item store. This statement is useful for verifying that the contents of the item store apply to the analysis and for generating ODS tables. [Table 73.10](#) summarizes the *options* available in the SHOW statement.

Table 73.10 SHOW Statement Options

Option	Description
ALL	Displays all applicable contents
BYVAR	Displays information about the BY variables
CLASSLEVELS	Displays the “Class Level Information” table
CORRELATION	Produces the correlation matrix of the parameter estimates
COVARIANCE	Produces the covariance matrix of the parameter estimates
EFFECTS	Displays information about the constructed effects
FITSTATS	Displays the fit statistics
HESSIAN	Displays the Hessian matrix
HERMITE	Generates the Hermite matrix $\mathbf{H} = (\mathbf{X}'\mathbf{X})^{-1}(\mathbf{X}'\mathbf{X})$
PARAMETERS	Displays the parameter estimates
PROGRAM	Displays the SAS program that generated the item store
XPX	Displays the crossproduct matrix $\mathbf{X}'\mathbf{X}$
XPXI	Displays the generalized inverse of the crossproduct matrix $\mathbf{X}'\mathbf{X}$

You can specify the following *options* after the SHOW statement:

ALL | **_ALL_**

displays all applicable contents.

BYVAR | **BY**

displays information about the BY variables in the source item store. If a BY statement was present when the item store was created, the PLM procedure performs the analysis separately for each BY group.

CLASSLEVELS | **CLASS**

displays the “Class Level Information” table. This table is produced by the PLM procedure by default if the model contains effects that depend on classification variables.

CORRELATION | **CORR** | **CORRB**

produces the correlation matrix of the parameter estimates. If the source item store contains a posterior sample of parameter estimates, the computed matrix is the correlation matrix of the sample covariance matrix.

COVARIANCE | COV | COVB

produces the covariance matrix of the parameter estimates. If the source item store contains a posterior sample of parameter estimates, the PLM procedure computes the empirical sample covariance matrix from the posterior estimates. You can convert this matrix into a sample correlation matrix with the CORRELATION option in the SHOW statement.

EFFECTS

displays information about the constructed effects in the model. Constructed effects are those that were created with the EFFECT statement in the procedure run that generated the source item store.

FITSTATS | FIT

displays the fit statistics from the item store.

HESSIAN | HESS

displays the Hessian matrix.

HERMITE | HERM

generates the Hermite matrix $\mathbf{H} = (\mathbf{X}'\mathbf{X})^{-}(\mathbf{X}'\mathbf{X})$. The PLM procedure chooses a reflexive, g_2 -inverse for the generalized inverse of the crossproduct matrix $\mathbf{X}'\mathbf{X}$. See “[Important Linear Algebra Concepts](#)” on page 44 in Chapter 3, “[Introduction to Statistical Modeling with SAS/STAT Software](#),” for information about generalized inverses and the sweep operator.

PARAMETERS<=n>**PARMS<=n>**

displays the parameter estimates. The structure of the display depends on whether a posterior sample of parameter estimates is available in the source item store. If such a sample is present, up to the first 20 parameter vectors are shown in wide format. You can modify this number with the n argument.

If no posterior sample is present, the single vector of parameter estimates is shown in narrow format. If the store contains information about the covariance matrix of the parameter estimates, then standard errors are added.

PROGRAM<(WIDTH=n)>**PROG<(WIDTH=n)>**

displays the SAS program that generated the item store, provided that this was stored at store generation time. The program does not include comments, titles, or some other global statements. The optional width parameter n determines the display width of the source code.

XPX | CROSSPRODUCT

displays the crossproduct matrix $\mathbf{X}'\mathbf{X}$.

XPXI

displays the generalized inverse of the crossproduct matrix $\mathbf{X}'\mathbf{X}$. The PLM procedure obtains a reflexive g_2 -inverse by sweeping. See “[Important Linear Algebra Concepts](#)” on page 44 in Chapter 3, “[Introduction to Statistical Modeling with SAS/STAT Software](#),” for information about generalized inverses and the sweep operator.

SLICE Statement

SLICE *model-effect* < / *options* > ;

The SLICE statement provides a general mechanism for performing a partitioned analysis of the LS-means for an interaction. This analysis is also known as an analysis of simple effects.

The SLICE statement uses the same *options* as the LSMEANS statement, which are summarized in [Table 19.21](#). For details about the syntax of the SLICE statement, see the section “[SLICE Statement](#)” on page 505 in Chapter 19, “[Shared Concepts and Topics](#).”

TEST Statement

TEST < *model-effects* > < / *options* > ;

The TEST statement enables you to perform *F* tests for model effects that test Type I, Type II, or Type III hypotheses. See Chapter 15, “[The Four Types of Estimable Functions](#),” for details about the construction of Type I, II, and III estimable functions.

[Table 73.11](#) summarizes the *options* available in the TEST statement.

Table 73.11 TEST Statement Options

Option	Description
CHISQ	Requests chi-square tests
DDF=	Specifies denominator degrees of freedom for fixed effects
E	Requests Type I, Type II, and Type III coefficients
E1	Requests Type I coefficients
E2	Requests Type II coefficients
E3	Requests Type III coefficients
HTYPE=	Indicates the type of hypothesis test to perform
INTERCEPT	Adds a row that corresponds to the overall intercept

For details about the syntax of the TEST statement, see the section “[TEST Statement](#)” on page 509 in Chapter 19, “[Shared Concepts and Topics](#).”

WHERE Statement

WHERE *expression* ;

You can use the WHERE statement in the PLM procedure when the item store contains BY-variable information and you want to apply the PROC PLM statements to only a subset of the BY groups.

A WHERE expression is a type of SAS expression that defines a condition. In the DATA step and in procedures that use SAS data sets as the input source, the WHERE expression is used to select observations for inclusion in the DATA step or in the analysis. In the PLM procedure, which does not accept a SAS data set but rather takes an item store that was created by a qualifying SAS/STAT procedure, the WHERE statement

is also used to specify conditions. The conditional selection does not apply to observations in PROC PLM, however. Instead, you use the WHERE statement in the PLM procedure to select a subset of BY groups from the item store to which to apply the PROC PLM statements.

The general syntax of the WHERE statement is

```
WHERE operand < operator operand > < AND | OR operand < operator operand > . . . > ;
```

where

operand is something to be operated on. The operand can be the name of a BY variable in the item store, a SAS function, a constant, or a predefined name to identify columns in result tables.

operator is a symbol that requests a comparison, logical operation, or arithmetic calculation. All SAS expression operators are valid for a WHERE expression.

For more details about how to specify general WHERE expressions, see *SAS Language Reference: Concepts*. Notice that the **FILTER** statement accepts similar expressions that are specified in terms of predefined *keywords*. Expressions in the WHERE statement of the PLM procedure are written in terms of BY variables.

There is no limit to the number of WHERE statements in the PLM procedure. When you specify multiple WHERE statements, the statements are *not* cumulative. Each WHERE statement is executed separately. You can think of each selection WHERE statement as one analytic query to the item store: the WHERE statement defines the query, and the PLM procedure is the querying engine. For example, suppose that the item store contains results for the numeric BY variables A and B. The following statements define two separate queries of the item store:

```
WHERE a = 4;
WHERE (b < 3) and (a > 4);
```

The PLM procedure first applies the requested analysis to all BY groups where a equals 4 (irrespective of the value of variable b). The analysis is then repeated for all BY groups where b is less than 3 and a is greater than 4.

Group selection with WHERE statements is possible only if the item store contains BY variables. You can use the **BYVAR** option in the **SHOW** statement to display the BY variables in the item store.

Note that WHERE expressions in the SAS DATA step and in many procedures are specified in terms of the unformatted values of data set variables, even if a format was applied to the variable. If you specify the **WHEREFORMAT** option in the **PROC PLM** statement, the PLM procedure evaluates WHERE expressions for BY variables in terms of the formatted values. For example, assume that the following format was applied to the variable tx when the item store was created:

```
proc format;
  value bf 1 = 'Control'
           2 = 'Treated';
run;
```

Then the following two PROC PLM runs are equivalent:

```
proc plm restore=MyStore;
  show parms;
  where b = 2;
run;
```



```
proc plm restore=MyStore whereformat;
  show parms;
  where b = 'Treated';
run;
```

Details: PLM Procedure

BY Processing and the PLM Procedure

When a BY statement is in effect for the analysis that creates an item store, the information about BY variables and BY-group-specific modeling results are transferred to the item store. In this case, the PLM procedure automatically assumes a processing mode for the item store that is akin to BY processing, with the PLM statements being applied in turn for each of the BY groups. Also, you can then obtain a table of BY groups with the **BYVAR** option in the **SHOW** statement. The “Source Information” table also displays the variable names of the BY variables if BY groups are present. The **WHERE** statement can be used to restrict the analysis to specific BY groups that meet the conditions of the **WHERE** expression.

See [Example 73.4](#) for an example that uses BY-group-specific information in the source item store.

As with procedures that operate on input data sets, the BY variable information is added automatically to any output data sets and ODS tables produced by the PLM procedure.

When you score a data set with the **SCORE** statement and the item store contains BY variables, three situations can arise:

- None of the BY variables are present in the scoring data set. In this situation the results of the BY groups in the item store are applied in turn to the entire scoring data set. For example, if the scoring data set contains 50 observations and no BY-variable information, the number of observations in the output data set of the **SCORE** statement equals 50 times the number of BY groups.
- The scoring data set contains only a part of the BY variables, or the variables have different type or format. The PLM procedure does not process such an incompatible scoring data set.
- All BY variables are in the scoring data set in the same type and format as when the item store was created. The BY-group-specific results are applied to each observation in the scoring data set. The scoring data set does not have to be sorted or grouped by the BY variables. However, it is computationally more efficient if the scoring data set is arranged by groups of the BY variables.

Analysis Based on Posterior Estimates

If an item store is saved from a Bayesian analysis (by PROC GENMOD or PROC PHREG or PROC LIFEREG), then PROC PLM can perform sampling-based inference based on Bayes posterior estimates that are saved in the item store. For example, the following statements request a Bayesian analysis and save the results to an item store named `sasuser.gmd`. For the Bayesian analysis, the random number generator seed is set to 1. By default, a noninformative distribution is set as the prior distribution for the regression coefficients and the posterior sample size is 10,000.

```

proc genmod data=gs;
  class a b;
  model y = a b;
  bayes seed=1;
  store sasuser.gmd / label='Bayesian Analysis';
run;

```

When the PLM procedure opens the item store `sasuser.gmd`, it detects that the results were saved from a Bayesian analysis. The posterior sample of regression coefficient estimates are then loaded to perform statistical inference tasks.

The majority of postprocessing tasks involve inference based on an estimable linear function $\mathbf{L}\hat{\boldsymbol{\beta}}$, which often requires its mean and variance. When the standard frequentist analyses are performed, the mean and variance have explicit forms because the parameter estimate $\hat{\boldsymbol{\beta}}$ is analytically tractable. However, explicit forms are not usually available when Bayesian models are fitted. Instead, empirical means and variance-covariance matrices for the estimable function are constructed from the posterior sample.

Let $\hat{\boldsymbol{\beta}}_i, i = 1, \dots, N_p$ denote the N_p vectors of posterior sample estimates of $\boldsymbol{\beta}$ saved in `sasuser.gmd`. Use these vectors to construct the posterior sample of estimable functions $\mathbf{L}\hat{\boldsymbol{\beta}}_i$. The posterior mean of the estimable function is thus

$$\overline{\mathbf{L}\hat{\boldsymbol{\beta}}} = \frac{1}{N_p} \sum_{i=1}^{N_p} \mathbf{L}\hat{\boldsymbol{\beta}}_i$$

and the posterior variance of the estimable function is

$$\mathbf{V}(\mathbf{L}\hat{\boldsymbol{\beta}}) = \frac{1}{N_p - 1} \sum_{i=1}^{N_p} (\mathbf{L}\hat{\boldsymbol{\beta}}_i - \overline{\mathbf{L}\hat{\boldsymbol{\beta}}})^2$$

Sometimes statistical inference on a transformation of $\mathbf{L}\hat{\boldsymbol{\beta}}$ is requested. For example, the EXP option for the ESTIMATE and LSMESTIMATE statements requests analysis based on $\exp(\mathbf{L}\hat{\boldsymbol{\beta}})$, exponentiation of the estimable function. If this type of analysis is requested, the posterior sample of transformed estimable functions is constructed by transforming each of the estimable function evaluated at the posterior sample: $f(\mathbf{L}\hat{\boldsymbol{\beta}}_i), i = 1, \dots, N_p$. The posterior mean and variance for $f(\mathbf{L}\hat{\boldsymbol{\beta}})$ are then computed from the constructed sample to make the inference:

$$\overline{f(\mathbf{L}\hat{\boldsymbol{\beta}})} = \frac{1}{N_p} \sum_{i=1}^{N_p} f(\mathbf{L}\hat{\boldsymbol{\beta}}_i)$$

$$\mathbf{V}(f(\mathbf{L}\hat{\boldsymbol{\beta}})) = \frac{1}{N_p - 1} \sum_{i=1}^{N_p} (f(\mathbf{L}\hat{\boldsymbol{\beta}}_i) - \overline{f(\mathbf{L}\hat{\boldsymbol{\beta}})})^2$$

After obtaining the posterior mean and variance, the PLM procedure proceeds to perform statistical inference based on them.

Scoring Data Sets for Zero-Inflated Models

The PLM procedure can score new observations for zero-inflated models with the **SCORE** statement. If you specify the **ILINK** option, the computed statistics are for estimated counts.

In the following formula, $\mathbf{x}$ is the design row for covariates that correspond to the Poisson or negative binomial component, $\hat{\boldsymbol{\beta}}$ is the column vector of the fitted regression parameters; $\mathbf{z}$ is the design row for covariates that correspond to the zero inflation component, $\hat{\boldsymbol{\gamma}}$ is the column vector of the fitted regression parameters; g and g^{-1} are the link and inverse link functions for the Poisson or negative binomial component, g_z and g_z^{-1} are the link and inverse link functions for the zero inflation component; Φ is the standard normal cumulative distribution function and α is the nominal significance level. Let

$$\begin{aligned}\mu &= g^{-1}(\eta) &= g^{-1}(\mathbf{x}\hat{\boldsymbol{\beta}}) \\ \mu_z &= g_z^{-1}(\eta_z) &= g_z^{-1}(\mathbf{z}\hat{\boldsymbol{\gamma}}) \\ v_1 &= \left(\frac{d\mu}{d\eta}\right)^2 \mathbf{x}\hat{\mathbf{V}}_{\boldsymbol{\beta},\boldsymbol{\beta}}\mathbf{x}' \\ v_2 &= \left(\frac{d\mu_z}{d\eta_z}\right)^2 \mathbf{z}\hat{\mathbf{V}}_{\boldsymbol{\gamma},\boldsymbol{\gamma}}\mathbf{z}' \\ v_{12} &= -\left(\frac{d\mu}{d\eta}\right)\left(\frac{d\mu_z}{d\eta_z}\right)\mathbf{x}\hat{\mathbf{V}}_{\boldsymbol{\beta},\boldsymbol{\gamma}}\mathbf{z}'\end{aligned}$$

The formula for statistics in the **SCORE** statement for zero-inflated models are listed as follows.

$$\begin{aligned}\text{PZERO} &= \omega = \mu_z \\ \text{PRED/ILINK} &= p_c = \mu(1 - \omega) \\ \text{STD/ILINK} &= s_c = \sqrt{p_c^2 v_2 + (1 - \omega)^2 v_1 + v_1 v_2 + 2p_c(1 - \omega)v_{12} + v_{12}^2} \\ \text{UCLM/ILINK} &= u_c = p_c \exp(\Phi^{-1}(1 - \alpha/2)s_c/p_c) \\ \text{LCLM/ILINK} &= l_c = p_c / \exp(\Phi^{-1}(1 - \alpha/2)s_c/p_c) \\ \text{PRED} &= p_l = g(p_c) \\ \text{STD} &= s_l = g(s_c)/\left(\frac{d\mu}{d\eta}\right) \\ \text{UCLM} &= u_l = g(u_c) \\ \text{LCLM} &= l_l = g(l_c)\end{aligned}$$

User-Defined Formats and the PLM Procedure

The PLM procedure does not support a **FORMAT** statement because it operates without an input data set, and also because changing the format properties of variables could alter the interpretation of parameter estimates, thus creating a dissonance with variable properties in effect when the item store was created. Instead, user-defined formats that are applied to classification variables when the item store is created are saved to the store and are by default reloaded by the PLM procedure. When the PLM procedure loads a format, notes are issued to the log.

You can change the load behavior for formats with the **FORMAT=** option in the **PROC PLM** statement.

User-defined formats do not need to be supplied in a new SAS session. However, when a user-defined format with the same name as a stored format exists and the default **FORMAT=RELOAD** option is in effect, the format definition loaded from the item store replaces the format currently in effect.

In the following statements, the format **AFORM** is created and applied to the variable **a** in the **PROC GLM** step. This format definition is transferred to the item store **sasuser.glm** through the **STORE** statement.

```

proc format;
  value aform 1='One' 2='Two' 3='Three';
run;
proc glm data=sp;
  format a aform.;
  class block a b;
  model y = block a b x;
  store sasuser.glm;
  weight x;
run;

```

The following statements replace the format definition of the AFORM format. The PLM step then reloads the AFORM format (from the item store) and thereby restores its original state.

```

proc format;
  value aform 1='Un' 2='Deux' 3='Trois';
run;
proc plm restore=sasuser.glm;
  show class;
  score data=sp out=plmout lcl lclm ucl uclm;
run;

```

The following notes, issued by the PLM procedure, inform you that the procedure loaded the format, the format already existed, and the existing format was replaced:

```

NOTE: The format AFORM was loaded from item store SASUSER.GLM.
NOTE: Format AFORM is already on the library.
NOTE: Format AFORM has been output.

```

After the PROC PLM run, the definition that is in effect for the format AFORM corresponds to the following SAS statements:

```

proc format;
  value aform 1='One' 2='Two' 3='Three';
run;

```

ODS Table Names

PROC PLM assigns a name to each table it creates. You can use these names to refer to the table when you use the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in [Table 73.12](#). For more information about ODS, see Chapter 20, “Using the Output Delivery System.”

Each of the EFFECTPLOT, ESTIMATE, LSMEANS, LSMESTIMATE, and SLICE statements also creates tables, which are not listed in [Table 73.12](#). For information about these tables, see the corresponding sections of Chapter 19, “Shared Concepts and Topics.”

Table 73.12 ODS Tables Produced by PROC PLM

Table Name	Description	Required Option
ByVarInfo	Information about BY variables in source item store (if present)	SHOW BYVAR
ClassLevels	Level information from the CLASS statement	Default output when model effects depend on CLASS variables
Corr	Correlation matrix of parameter estimates	SHOW CORR
Cov	Covariance matrix of parameter estimates	SHOW COV
FitStatistics	Fit statistics	SHOW FIT
Hessian	Hessian matrix	SHOW HESSIAN
Hermite	Hermite matrix	SHOW HERMITE
ParameterEstimates	Parameter estimates	SHOW PARMS
ParameterSample	Sampled (posterior) parameter estimates	SHOW PARMS
Program	Originating source code	SHOW PROGRAM
StoreInfo	Information about source item store	Default
XpX	$\mathbf{X}'\mathbf{X}$ matrix	SHOW XPX
XpXI	$(\mathbf{X}'\mathbf{X})^{-1}$ matrix	SHOW XPXI

ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 21, “[Statistical Graphics Using ODS](#).”

Before you create graphs, ODS Graphics must be enabled (for example, by specifying the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section “[Enabling and Disabling ODS Graphics](#)” on page 606 in Chapter 21, “[Statistical Graphics Using ODS](#).”

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section “[A Primer on ODS Statistical Graphics](#)” on page 605 in Chapter 21, “[Statistical Graphics Using ODS](#).”

When ODS Graphics is enabled, then each of the EFFECTPLOT, ESTIMATE, LSMEANS, LSMESTIMATE, and SLICE statements can produce plots associated with their analyses. For information about these plots, see the corresponding sections of Chapter 19, “[Shared Concepts and Topics](#).”

Examples: PLM Procedure

Example 73.1: Scoring with PROC PLM

Logistic regression with model selection is often used to extract useful information and build interpretable models for classification problems with many variables. This example demonstrates how you can use PROC LOGISTIC to build a spline model on a simulated data set and how you can later use the fitted model to classify new observations.

The following DATA step creates a data set named SimuData, which contains 5,000 observations and 100 continuous variables:

```
%let nObs    = 5000;
%let nVars   = 100;
data SimuData;
    array x{&nVars};
    do obsNum=1 to &nObs;
        do j=1 to &nVars;
            x{j}=ranuni(1);
        end;

        linp = 10 + 11*x1 - 10*sqrt(x2) + 2/x3 - 8*exp(x4) + 7*x5*x5
              - 6*x6**1.5 + 5*log(x7) - 4*sin(3.14*x8) + 3*x9 - 2*x10;
        TrueProb = 1/(1+exp(-linp));

        if ranuni(1) < TrueProb then y=1;
                                   else y=0;

        output;
    end;
run;
```

The response is binary based on the inversely transformed logit values. The true logit is a function of only 10 of the 100 variables, including nonlinear transformations of seven variables, as follows:

$$\text{logit}(p) = 10 + 11x_1 - 10\sqrt{x_2} + \frac{2}{x_3} - 8\exp(x_4) + 7x_5^2 - 6x_6^{1.5} + 5\log(x_7) - 4\sin(3.14x_8) + 3x_9 - 2x_{10}$$

Now suppose the true model is not known. With some exploratory data analysis, you determine that the dependency of the logit on some variables is nonlinear. Therefore, you decide to use splines to model this nonlinear dependence. Also, you want to use stepwise regression to remove unimportant variable transformations. The following statements perform the task:

```
proc logistic data=SimuData;
    effect splines = spline(x1-x&nVars/separate);
    model y = splines/selection=stepwise;
    store sasuser.SimuModel;
run;
```

By default, PROC LOGISTIC models the probability that $y = 0$. The EFFECT statement requests an effect named splines constructed by all predictors in the data. The SEPARATE option specifies that the

spline basis for each variable be treated as a separate set so that model selection applies to each individual set. The `SELECTION=STEPWISE` specifies the stepwise regression as the model selection technique. The `STORE` statement requests that the fitted model be saved to an item store `sasuser.SimuModel`. See “[Example 73.2: Working with Item Stores](#)” on page 6221 for an example with more details about working with item stores.

The spline effect for each predictor produces seven columns in the design matrix, making stepwise regression computationally intensive. For example, a typical Pentium 4 workstation takes around ten minutes to run the preceding statements. Real data sets for classification can be much larger. See examples at UCI Machine Learning Repository (Asuncion and Newman 2007). If new observations about which you want to make predictions are available at model fitting time, you can add the `SCORE` statement in the `LOGISTIC` procedure. Consider the case in which observations to predict become available after fitting the model. With `PROC PLM`, you do not have to repeat the computationally intensive model-fitting processes multiple times. You can use the `SCORE` statement in the `PLM` procedure to score new observations based on the item store `sasuser.SimuModel` that was created during the initial model building. For example, to compute the probability of $y = 0$ for one new observation with all predictor values equal to 0.15 in the data set `test`, you can use the following statements:

```
data test;
  array x{&nVars};
  do j=1 to &nVars;
    x{j}=0.15;
  end;
  drop j;
  output;
run;

proc plm restore=sasuser.SimuModel;
  score data=test out=testout predicted / ilink;
run;
```

The `ILINK` option in the `SCORE` statement requests that predicted values be inversely transformed to the response scale. In this case, it is the predicted probability of $y = 0$. [Output 73.1.1](#) shows the predicted probability for the new observation.

Output 73.1.1 Predicted Probability for One New Observation

Obs	Predicted
1	0.56649

Example 73.2: Working with Item Stores

This example demonstrates how procedures save statistical analysis context and results into item stores and how you can use `PROC PLM` to make post hoc inference based on saved item stores. The data are taken from McCullagh and Nelder (1989) and concern the effects on taste of various cheese additives. Four cheese additives were tested, and 52 response ratings for each additive were obtained. The response was measured on a scale of nine categories that range from strong dislike (1) to excellent taste (9). The following program

saves the data in the data set `Cheese`. The variable `y` contains the taste rating, the variable `Additive` contains cheese additive types, and the variable `freq` contains the frequencies 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 `y` is a categorical variable that contains nine ordered levels. You can use PROC LOGISTIC to fit an ordinal model to investigate the effects of the cheese additive types on taste ratings. Suppose you also want to save the ordinal model into an item store so that you can make statistical inference later. You can use the following statements to perform the tasks:

```
proc logistic data=cheese;
  freq freq;
  class additive y / param=glm;
  model y=additive;
  store sasuser.cheese;
  title 'Ordinal Model on Cheese Additives';
run;
```

By default, PROC LOGISTIC uses the cumulative logit model for the ordered categorical response. The STORE statement saves the fitted model to a SAS item store named `sasuser.cheese`. The name is a two-level SAS name of the form `libname.membername`. If `libname` is not specified in the STORE statement, the fitted results are saved in `work.membername` and the item store is deleted after the current SAS session ends. With this example, the fitted model is saved to an item store named `sasuser.cheese` in the `Sasuser` library. It is not deleted after the current SAS session ends. You can use PROC PLM to restore the results later.

The following statements use PROC PLM to load the saved model context and results by specifying `RESTORE=` with the target item store `sasuser.cheese`. Then they use two SHOW statements to display separate information saved in the item store. The first SHOW statement with the PROGRAM option displays the program that was used to generate the item store `sasuser.cheese`. The second SHOW statement with the PARMS option displays parameter estimates and associated statistics of the fitted ordinal model.

```
proc plm restore=sasuser.cheese;
  show program;
  show parms;
run;
```

Output 73.2.1 displays the program that generated the item store `sasuser.cheese`. Except for the title information, it matches the original program.

Output 73.2.1 Program Information from sasuser.cheese

```

Ordinal Model on Cheese Additives

The PLM Procedure

SAS Program Information

proc logistic data=cheese;
  freq freq;
  class additive y / param=glm;
  model y=additive;
  store sasuser.cheese;
run;

```

Output 73.2.2 displays estimates of the intercept terms and covariates and associated statistics. The intercept terms correspond to eight cumulative logits defined on taste ratings; that is, the i th intercept for i th logit is

$$\log\left(\frac{\sum_{j \leq i} p_j}{1 - \sum_{j \leq i} p_j}\right)$$

Output 73.2.2 Parameter Estimates of the Ordinal Model

Parameter Estimates			
Parameter	Taste Rating	Estimate	Standard Error
Intercept	1	-7.0801	0.5624
Intercept	2	-6.0249	0.4755
Intercept	3	-4.9254	0.4272
Intercept	4	-3.8568	0.3902
Intercept	5	-2.5205	0.3431
Intercept	6	-1.5685	0.3086
Intercept	7	-0.06688	0.2658
Intercept	8	1.4930	0.3310
Additive 1		1.6128	0.3778
Additive 2		4.9645	0.4741
Additive 3		3.3227	0.4251
Additive 4		0	.

You can perform various statistical inference tasks from a saved item store, as long as the task is applicable under the model context. For example, you can perform group comparisons between different cheese additive types. See the next example for details.

Example 73.3: Group Comparisons in an Ordinal Model

This example continues the study of the effects on taste of various cheese additives. You have finished fitting an ordinal logistic model and saved it to an item store named `sasuser.cheese` in the previous example. Suppose you want to make comparisons between any pair of cheese additives. You can conduct the analysis by using the `ESTIMATE` statement and constructing an appropriate `L` matrix, or by using the `LSMEANS` statement to compute least squares means differences. For an ordinal logistic model with the cumulative logit link, the least squares means are predicted population margins of the cumulative logits. The following statements compute and display differences between least squares means of cheese additive:

```
ods graphics on;
proc plm restore=sasuser.cheese;
    lsmeans additive / cl diff oddsratio plot=diff;
run;
ods graphics off;
```

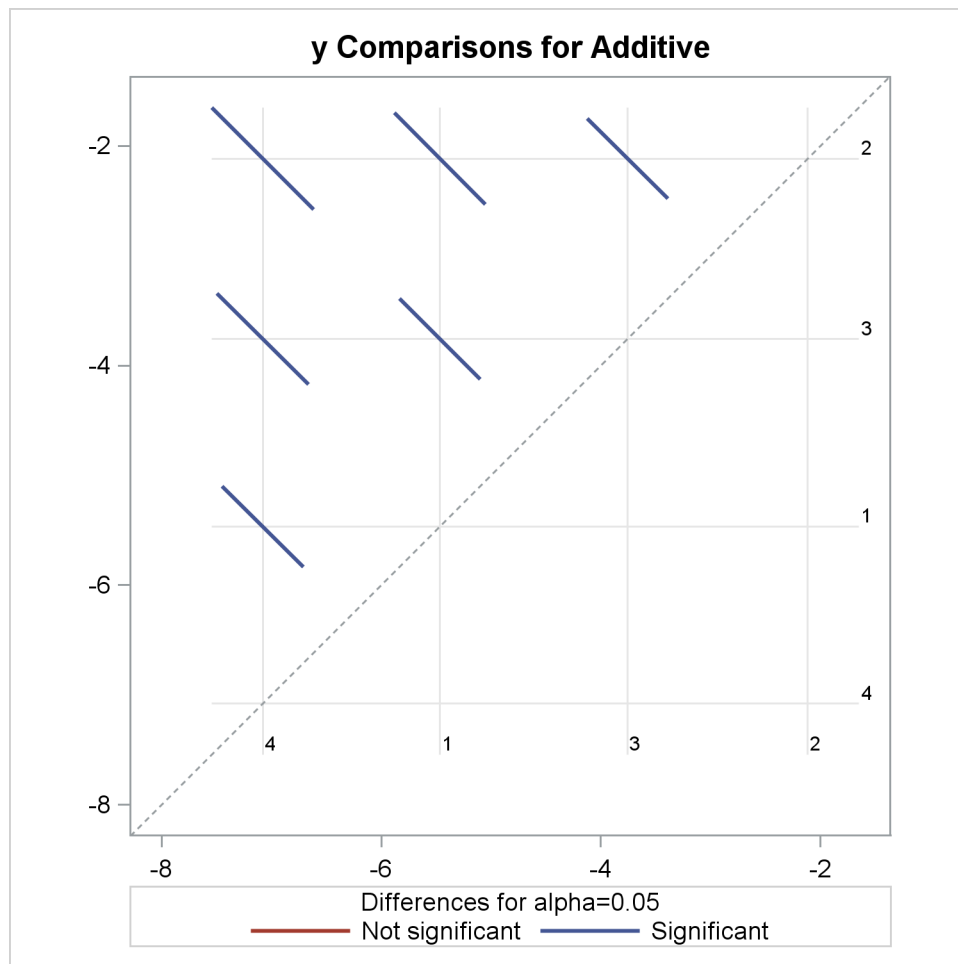
The `LSMEANS` statement contains four options. The `DIFF` option requests least squares means differences for cheese additives. Since the fitted model is an ordinal logistic model with the cumulative logit link, the least squares means differences represent log cumulative odds ratios. The `ODDSRATIO` option requests exponentiation of the LS-means differences which produces cumulative odds ratios. The `CL` option constructs confidence limits for the LS-means differences. When ODS Graphics is enabled, the `PLOTS=DIFF` option requests a display of all pairwise least squares means differences and their significance.

Output 73.3.1 displays the LS-means differences. The reported log odds ratios indicate the relative difference among the cheese additives. A negative log odds ratio indicates that the first category (displayed in the `Additive` column) having a lower taste rating is less likely than the second category (displayed in the `_Additive` column) having a lower taste rating. For example, the log odds ratio between cheese additive 1 and 2 is -3.3517 and the corresponding odds ratio is 0.035 . This means the odds of cheese additive 1 receiving a poor rating is 0.035 times the odds of cheese additive 2 receiving a poor rating. In addition to the highly significant p -value (< 0.0001), the confidence limits for both the log odds ratio and the odds ratio indicate that you can reject the null hypothesis that the odds of cheese additive 1 having a lower taste rating is the same as that of cheese additive 2 having a lower rating. Similarly, the odds of cheese additive 2 having a lower rating is 143.241 (with 95% confidence limits $(56.558, 362.777)$) times the odds of cheese additive 4 having a lower rating. With the same logic, you can conclude that the preference order for the four cheese types from the most favorable to the least favorable is: 4, 1, 3 and 2.

Output 73.3.1 LS-Means Differences of Additive

Ordinal Model on Cheese Additives						
The PLM Procedure						
Differences of Additive Least Squares Means						
Additive	_Additive	Estimate	Standard Error	z Value	Pr > z	Alpha
1	2	-3.3517	0.4235	-7.91	<.0001	0.05
1	3	-1.7098	0.3731	-4.58	<.0001	0.05
1	4	1.6128	0.3778	4.27	<.0001	0.05
2	3	1.6419	0.3738	4.39	<.0001	0.05
2	4	4.9645	0.4741	10.47	<.0001	0.05
3	4	3.3227	0.4251	7.82	<.0001	0.05
Differences of Additive Least Squares Means						
Additive	_Additive	Lower	Upper	Odds Ratio	Lower Confidence Limit for Odds Ratio	Upper Confidence Limit for Odds Ratio
1	2	-4.1818	-2.5216	0.035	0.015	0.080
1	3	-2.4410	-0.9787	0.181	0.087	0.376
1	4	0.8724	2.3532	5.017	2.393	10.520
2	3	0.9092	2.3746	5.165	2.482	10.746
2	4	4.0353	5.8938	143.241	56.558	362.777
3	4	2.4895	4.1558	27.734	12.055	63.805

Output 73.3.2 displays the DiffPlot. This shows that all pairs of LS-means differences, equivalent to log odds ratios in this case, are significant at the level of $\alpha = 0.05$. This means that the preference between any pair of the four cheese additive types are statistically significantly different.

Output 73.3.2 LS-Means Plot of Pairwise Differences

Example 73.4: Posterior Inference for Binomial Data

This example demonstrates how you can use PROC PLM to perform posterior inference from a Bayesian analysis. The data for this example are taken from Weisberg (1985) and concern the effect of small electrical currents on farm animals. The ultimate goal of the experiment was to understand the effects of high-voltage power lines on livestock and to better protect farm animals. Seven cows and six shock intensities were used in two experiments. In one experiment, each cow was given 30 electrical shocks with five at each shock intensity in random order. The number of shock responses was recorded for each cow at each shock level. The experiment was then repeated to investigate whether the response diminished due to fatigue of cows, or due to learning. So each cow received a total of 60 shocks. For the following analysis, the cow difference is ignored. The following DATA step lists the data where the variable `current` represents the shock level, the variable `response` represents the number of shock responses, the variable `trial` represents the total number of trials at each shock level, and the variable `experiment` represents the experiment number (1 for the initial experiment and 2 for the repeated one):

```

data cow;
    input current response trial experiment;
    datalines;
0  0 35 1
0  0 35 2
1  6 35 1
1  3 35 2
2 13 35 1
2  8 35 2
3 26 35 1
3 21 35 2
4 33 35 1
4 27 35 2
5 34 35 1
5 29 35 2
;

```

Suppose you are interested in modeling the distribution of the shock response based on the level of the current and the experiment number. You can use the GENMOD procedure to fit a frequentist logistic model for the data. However, if you have some prior information about parameter estimates, you can fit a Bayesian logistic regression model to take this prior information into account. In this case, suppose you believe the logit of response has a positive association with the shock level but you are uncertain about the ranges of other regression coefficients. To incorporate this prior information in the regression model, you can use the following statements:

```

data prior;
    input _type_$ current;
    datalines;
mean 100
var   50
;

proc genmod data=cow;
    class experiment;
    bayes coeffprior=normal(input=prior) seed=1;
    model response/trial = current|experiment / dist=binomial;
    store cowgmd;
    title 'Bayesian Logistic Model on Cow';
run;

```

The DATA step creates a data set prior that specifies the prior distribution for the variable current, which in this case is a normal distribution with mean 100 and variance 50. This reflects a rough belief in a positive coefficient in a moderate range for current. The prior distribution parameters are not specified for experiment and the interaction between experiment and current, and so PROC GENMOD assigns a default prior for them, which is a normal distribution with mean 0 and variance 1E6.

The BAYES statement in PROC GENMOD specifies that the regression coefficients follow a normal distribution with mean and variance specified in the input data set named prior. It also specifies 1 as the seed for the random number generator in the simulation of the posterior sample. The MODEL statement requests a logistic regression model with a logit link. The STORE statement requests that the fitted results be saved into an item store named cowgmd.

The convergence diagnostics in the output of PROC GENMOD indicate that the Markov chain has converged. [Output 73.4.1](#) displays summaries on the posterior sample of the regression coefficients. The posterior mean for the intercept is -3.5857 with a 95% HPD interval $(-4.5226, -2.6303)$. The posterior mean of the coefficient for current is 1.1893 with a 95% HPD interval $(0.8950, 1.4946)$, which indicates a positive association between the logit of response and the shock level. Further investigation about whether shock reaction was different between two experiment is warranted.

Output 73.4.1 Posterior Summaries on the Bayesian Logistic Model

Bayesian Logistic Model on Cow						
The GENMOD Procedure						
Bayesian Analysis						
Posterior Summaries						
Parameter	N	Mean	Standard Deviation	Percentiles		
				25%	50%	75%
Intercept	10000	-3.6047	0.4906	-3.9281	-3.5842	-3.2679
current	10000	1.1966	0.1561	1.0873	1.1927	1.2996
experiment1	10000	0.0350	0.7014	-0.4206	0.0347	0.4987
experiment1current	10000	0.3574	0.2503	0.1906	0.3520	0.5235
Posterior Intervals						
Parameter	Alpha	Equal-Tail Interval		HPD Interval		
Intercept	0.050	-4.6233	-2.7074	-4.5611	-2.6581	
current	0.050	0.9073	1.5152	0.9028	1.5064	
experiment1	0.050	-1.3651	1.4004	-1.2995	1.4370	
experiment1current	0.050	-0.1293	0.8580	-0.1287	0.8580	

Bayesian model fitting typically involves a large amount of simulation. Using the item store and PROC PLM, you do not need to refit the model to perform further posterior inference. Suppose you want to determine whether the shock reaction for the current level is different between the two experiments. You can use PROC PLM with the ESTIMATE statement in the following statements:

```
proc plm restore=cowgmd;
  estimate
    'Diff at current 0' experiment 1 -1 current*experiment [1, 0 1] [-1, 0 2],
    'Diff at current 1' experiment 1 -1 current*experiment [1, 1 1] [-1, 1 2],
    'Diff at current 2' experiment 1 -1 current*experiment [1, 2 1] [-1, 2 2],
    'Diff at current 3' experiment 1 -1 current*experiment [1, 3 1] [-1, 3 2],
    'Diff at current 4' experiment 1 -1 current*experiment [1, 4 1] [-1, 4 2],
    'Diff at current 5' experiment 1 -1 current*experiment [1, 5 1] [-1, 5 2]
  / exp cl;
run;
```

Each line in the ESTIMATE statement compares the fits between the two groups at each current level. The nonpositional syntax is used for the interaction effect current*experiment. For example, the first line requests coefficient 1 for the interaction effect at current level 0 for the initial experiment, and coefficient -1 for the

effect at current level 0 for the repeated experiment. The two terms are then added to derive the difference. For more details about the nonpositional syntax, see “[Positional and Nonpositional Syntax for Coefficients in Linear Functions](#)” on page 455 in Chapter 19, “[Shared Concepts and Topics](#).”

The EXP option exponentiates log odds ratios to produce odds ratios. The CL option requests that confidence limits be constructed for both log odds ratios and odds ratios. [Output 73.4.2](#) lists the posterior sample estimates for differences between experiments at different current levels.

Output 73.4.2 Comparisons between Experiments at Different Current Levels

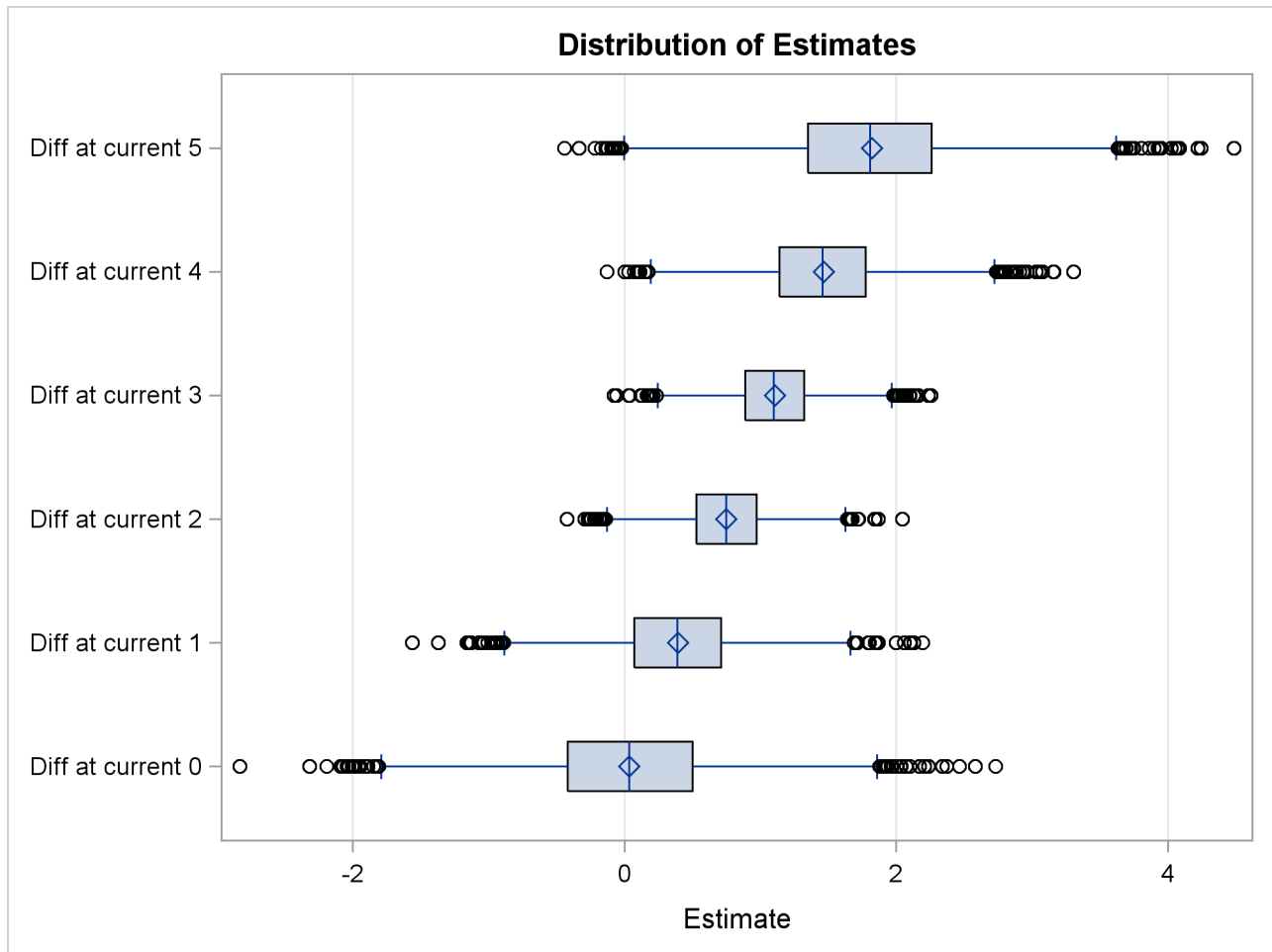
Bayesian Logistic Model on Cow						
The PLM Procedure						
Sample Estimates						
Label	N	Estimate	Standard Deviation	-----Percentiles-----		
				25th	50th	75th
Diff at current 0	10000	0.03500	0.7014	-0.4206	0.0347	0.4987
Diff at current 1	10000	0.3924	0.4865	0.0700	0.3884	0.7096
Diff at current 2	10000	0.7498	0.3266	0.5283	0.7468	0.9719
Diff at current 3	10000	1.1072	0.3194	0.8901	1.1001	1.3220
Diff at current 4	10000	1.4646	0.4718	1.1387	1.4559	1.7756
Diff at current 5	10000	1.8219	0.6844	1.3508	1.8079	2.2601
Sample Estimates						
Label	Alpha	Lower HPD	Upper HPD	Exponentiated	Standard Deviation of Exponentiated	
Diff at current 0	0.05	-1.2995	1.4370	1.3258	1.080632	
Diff at current 1	0.05	-0.5287	1.3599	1.6672	0.873185	
Diff at current 2	0.05	0.1300	1.3827	2.2328	0.753450	
Diff at current 3	0.05	0.4772	1.7182	3.1863	1.068673	
Diff at current 4	0.05	0.5369	2.3694	4.8511	2.562732	
Diff at current 5	0.05	0.4881	3.1505	7.8941	6.668085	
Sample Estimates						
Label	-----Percentiles for Exponentiated-----			Lower HPD of Exponentiated	Upper HPD of Exponentiated	
	25th	50th	75th			
Diff at current 0	0.6566	1.0353	1.6466	0.1263	3.3077	
Diff at current 1	1.0725	1.4746	2.0331	0.3809	3.3011	
Diff at current 2	1.6961	2.1101	2.6429	1.0081	3.7135	
Diff at current 3	2.4353	3.0045	3.7509	1.3990	5.1655	
Diff at current 4	3.1227	4.2885	5.9040	1.3083	9.6902	
Diff at current 5	3.8606	6.0976	9.5838	0.9081	19.4638	

The sample statistics are constructed from the posterior sample saved in the item store `cowgmd`. From the output, the odds of a cow showing shock reaction at level 0 in the initial experiment is 1.2811 (with a 95% HPD interval (0.07387, 3.1001)) times the odds in the repeated experiment. The HPD interval for the odds ratio is constructed based on the mean and variance of the sample of the exponentiated log odds ratios, instead of based on the exponentiated mean and variance of the posterior sample of log odds ratios. The HPD interval suggests that there is not much evidence that the cows responded differently at current level 0 between the two experiments. Similar conclusions can be drawn for current level 1, 2, and 5. However, there is strong evidence that cows responded differently at current level 3 and 4 between the two experiments. The possible explanation is that, if the current level is so small that cows could hardly feel it or the current level is so strong that cows could hardly bear it, cows would respond consistently in the two experiment. If the current level is moderate, cows might get used to it and their response diminished in the repeated experiment.

You can visualize the distribution of the posterior sample of log odds ratios by specifying the `PLOTS=` option in the `ESTIMATE` statement. In the following statements, ODS Graphics is enabled by the `ODS GRAPHICS ON` statement, the `PLOTS=BOXPLOT` option requests a box plot of posterior distribution of log odds ratios. The suboption `ORIENT=HORIZONTAL` specifies a horizontal orientation of the boxes.

```
ods graphics on;
proc plm restore=cowgmd;
  estimate
    'Diff at current 0' experiment 1 -1 current*experiment [1, 0 1] [-1, 0 2],
    'Diff at current 1' experiment 1 -1 current*experiment [1, 1 1] [-1, 1 2],
    'Diff at current 2' experiment 1 -1 current*experiment [1, 2 1] [-1, 2 2],
    'Diff at current 3' experiment 1 -1 current*experiment [1, 3 1] [-1, 3 2],
    'Diff at current 4' experiment 1 -1 current*experiment [1, 4 1] [-1, 4 2],
    'Diff at current 5' experiment 1 -1 current*experiment [1, 5 1] [-1, 5 2]
    / plots=boxplot(orient=horizontal);
run;
ods graphics off;
```

Output 73.4.3 displays the box plot of the posterior sample of log odds ratios. The two boxes for differences at current level 3 and 4 show that the corresponding log odds ratios are significantly larger than the reference value $x = 0$. This indicates that there is obvious evidence that the probability of cow response is larger in the initial experiment than in the repeated one at the two current levels. The other four boxes show that the corresponding log odds ratios are not significantly different from 0, which suggests that there is no obvious reaction difference at current level 0, 1, 2, and 5 between the two experiments.

Output 73.4.3 Box Plot of Difference between Two Experiments

Example 73.5: BY-Group Processing

This example uses a data set on a study of the analgesic effects of treatments on elderly patients with neuralgia. The purpose of this example is to show how PROC PLM behaves under different situations when BY-group processing is present. Two test treatments and a placebo are compared to test whether the patient reported pain or not. For each patient, the information of age, gender, and the duration of complaint before the treatment began were recorded. The following DATA step creates the data set named Neuralgia:

```

Data Neuralgia;
  input Treatment $ Sex $ Age Duration Pain $ @@;
  datalines;
P F 68 1 No B M 74 16 No P F 67 30 No
P M 66 26 Yes B F 67 28 No B F 77 16 No
A F 71 12 No B F 72 50 No B F 76 9 Yes
A M 71 17 Yes A F 63 27 No A F 69 18 Yes
B F 66 12 No A M 62 42 No P F 64 1 Yes
A F 64 17 No P M 74 4 No A F 72 25 No
P M 70 1 Yes B M 66 19 No B M 59 29 No
A F 64 30 No A M 70 28 No A M 69 1 No

```

```

B F 78 1 No P M 83 1 Yes B F 69 42 No
B M 75 30 Yes P M 77 29 Yes P F 79 20 Yes
A M 70 12 No A F 69 12 No B F 65 14 No
B M 70 1 No B M 67 23 No A M 76 25 Yes
P M 78 12 Yes B M 77 1 Yes B F 69 24 No
P M 66 4 Yes P F 65 29 No P M 60 26 Yes
A M 78 15 Yes B M 75 21 Yes A F 67 11 No
P F 72 27 No P F 70 13 Yes A M 75 6 Yes
B F 65 7 No P F 68 27 Yes P M 68 11 Yes
P M 67 17 Yes B M 70 22 No A M 65 15 No
P F 67 1 Yes A M 67 10 No P F 72 11 Yes
A F 74 1 No B M 80 21 Yes A F 69 3 No
;

```

The data set contains five variables. Treatment is a classification variable that has three levels: A and B represent the two test treatments, and P represents the placebo treatment. Sex is a classification variable that indicates each patient's gender. Age is a continuous variable that indicates the age in years of each patient when a treatment began. Duration is a continuous variable that indicates the duration of complaint in months. The last variable Pain is the response variable with two levels: 'Yes' if pain was reported, 'No' if no pain was reported.

Suppose there is some preliminary belief that the dependency of pain on the explanatory variables is different for male and female patients, leading to separate models between genders. You believe there might be redundant information for predicting the probability of Pain. Thus, you want to perform model selection to eliminate unnecessary effects. You can use the following statements:

```

proc sort data=Neuralgia;
  by sex;
run;

proc logistic data=Neuralgia;
  class Treatment / param=glm;
  model pain = Treatment Age Duration / selection=backward;
  by sex;
  store painmodel;
  title 'Logistic Model on Neuralgia';
run;

```

PROC SORT is called to sort the data by variable Sex. The LOGISTIC procedure is then called to fit the probability of no pain. Three variables are specified for the full model: Treatment, Age, and Duration. Backward elimination is used as the model selection method. The BY statement fits separate models for male and female patients. Finally, the STORE statement specifies that the fitted results be saved to an item store named painmodel.

[Output 73.5.1](#) lists parameter estimates from the two models after backward elimination is performed. From the model for female patients, Treatment is the only factor that affects the probability of no pain, and Treatment A and B have the same positive effect in predicting the probability of no pain. From the model for male patients, both Treatment and Age are included in the selected model. Treatment A and B have different positive effects, while Age has a negative effect in predicting the probability of no pain.

Output 73.5.1 Parameter Estimates for Male and Female Patients

Logistic Model on Neuralgia					
----- Sex=F -----					
The LOGISTIC Procedure					
Analysis of Maximum Likelihood Estimates					
Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	1	-0.4055	0.6455	0.3946	0.5299
Treatment A	1	2.6027	1.2360	4.4339	0.0352
Treatment B	1	2.6027	1.2360	4.4339	0.0352
Treatment P	0	0	.	.	.
----- Sex=M -----					
Analysis of Maximum Likelihood Estimates					
Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	1	20.6178	9.1638	5.0621	0.0245
Treatment A	1	3.9982	1.7333	5.3208	0.0211
Treatment B	1	4.5556	1.9252	5.5993	0.0180
Treatment P	0	0	.	.	.
Age	1	-0.3416	0.1408	5.8869	0.0153

Now the fitted models are saved to the item store `painmodel`. Suppose you want to use it to score several new observations. The following DATA steps create three data sets for scoring:

```
data score1;
  input Treatment $ Sex $ Age;
  datalines;
A F 20
B F 30
P F 40
A M 20
B M 30
P M 40
;

data score2;
  set score1(drop=sex);
run;

data score3;
  set score2(drop=Age);
run;
```

The first score data set `score1` contains six observations and all the variables that are specified in the full model. The second score data set `score2` is a duplicate of `score1` except that `Sex` is dropped. The third score data set `score3` is a duplicate of `score2` except that `Age` is dropped. You can use the following statements to score the three data sets:

```
proc plm restore=painmodel;
  score data=score1 out=score1out predicted;
  score data=score2 out=score2out predicted;
  score data=score3 out=score3out predicted;
run;
```

[Output 73.5.2](#) lists the store information that PROC PLM reads from the item store `painmodel`. The “Model Effects” entry lists all three variables that are specified in the full model before the BY-group processing.

Output 73.5.2 Item Store Information for `painmodel`

Logistic Model on Neuralgia	
The PLM Procedure	
Store Information	
Item Store	WORK.PAINMODEL
Data Set Created From	WORK.NEURALGIA
Created By	PROC LOGISTIC
Date Created	24OCT13:13:58:28
By Variable	Sex
Response Variable	Pain
Link Function	Logit
Distribution	Binary
Class Variables	Treatment Pain
Model Effects	Intercept Treatment Age Duration

With the three `SCORE` statements, three data sets are thus produced: `score1out`, `score2out`, and `score3out`. They contain the linear predictors in addition to all original variables. The data set `score1out` contains the values shown in [Output 73.5.3](#).

Output 73.5.3 Values of Data Set `score1out`

Logistic Model on Neuralgia				
Obs	Treatment	Sex	Age	Predicted
1	A	F	20	2.1972
2	B	F	30	2.1972
3	P	F	40	-0.4055
4	A	M	20	17.7850
5	B	M	30	14.9269
6	P	M	40	6.9557

Linear predictors are computed for all six observations. Because the BY variable Sex is available in score1, PROC PLM uses separate models to score observations of male and female patients. So an observation with the same Treatment and Age has different linear predictors for different genders.

The data set score2out contains the values shown in [Output 73.5.4](#).

Output 73.5.4 Values of Data Set score2out

Logistic Model on Neuralgia				
Obs	Sex	Treatment	Age	Predicted
1	F	A	20	2.1972
2	F	B	30	2.1972
3	F	P	40	-0.4055
4	F	A	20	2.1972
5	F	B	30	2.1972
6	F	P	40	-0.4055
7	M	A	20	17.7850
8	M	B	30	14.9269
9	M	P	40	6.9557
10	M	A	20	17.7850
11	M	B	30	14.9269
12	M	P	40	6.9557

The second score data set score2 does not contain the BY variable Sex. PROC PLM continues to score the full data set two times. Each time the scoring is based on the fitted model for each corresponding BY-group. In the output data set, Sex is added at the first column as the BY-group indicator. The first six entries correspond to the model for female patients, and the next six entries correspond to the model for male patients. Age is not included in the first model, and Treatment A and B have the same parameter estimates, so observations 1, 2, 4, and 5 have the same linear predicted value.

The data set score3out contains the values shown in [Output 73.5.5](#).

Output 73.5.5 Values of Data Set score3out

Logistic Model on Neuralgia			
Obs	Sex	Treatment	Predicted
1	F	A	2.19722
2	F	B	2.19722
3	F	P	-0.40547
4	F	A	2.19722
5	F	B	2.19722
6	F	P	-0.40547
7	M	A	.
8	M	B	.
9	M	P	.
10	M	A	.
11	M	B	.
12	M	P	.

The third score data set `score3` does not contain the BY variable `Sex`. PROC PLM scores the full data twice with separate models. Furthermore, it does not contain the variable `Age`, which is a selected variable for predicting the probability of no pain for male patients. Thus, PROC PLM computes linear predictor values for `score3` by using the first model for female patients, and sets the linear predictor to missing when using the second model for male patients to score the data set.

Example 73.6: Comparing Multiple B-Splines

This example conducts an analysis similar to Example 15 in Chapter 43.33, “[Examples: GLIMMIX Procedure](#).” It uses simulated data to perform multiple comparisons among predicted values in a model with group-specific trends that are modeled through regression splines. The estimable functions are formed using nonpositional syntax with constructed effects. Consider the data in the following DATA step. Each of the 100 observations for the continuous response variable `y` is associated with one of two groups.

```
data spline;
  input group y @@;
  x = _n_;
  datalines;
1    -.020 1    0.199 2    -1.36 1    -.026
2    -.397 1    0.065 2    -.861 1    0.251
1    0.253 2    -.460 2    0.195 2    -.108
1    0.379 1    0.971 1    0.712 2    0.811
2    0.574 2    0.755 1    0.316 2    0.961
2    1.088 2    0.607 2    0.959 1    0.653
1    0.629 2    1.237 2    0.734 2    0.299
2    1.002 2    1.201 1    1.520 1    1.105
1    1.329 1    1.580 2    1.098 1    1.613
2    1.052 2    1.108 2    1.257 2    2.005
2    1.726 2    1.179 2    1.338 1    1.707
2    2.105 2    1.828 2    1.368 1    2.252
1    1.984 2    1.867 1    2.771 1    2.052
2    1.522 2    2.200 1    2.562 1    2.517
1    2.769 1    2.534 2    1.969 1    2.460
1    2.873 1    2.678 1    3.135 2    1.705
1    2.893 1    3.023 1    3.050 2    2.273
2    2.549 1    2.836 2    2.375 2    1.841
1    3.727 1    3.806 1    3.269 1    3.533
1    2.948 2    1.954 2    2.326 2    2.017
1    3.744 2    2.431 2    2.040 1    3.995
2    1.996 2    2.028 2    2.321 2    2.479
2    2.337 1    4.516 2    2.326 2    2.144
2    2.474 2    2.221 1    4.867 2    2.453
1    5.253 2    3.024 2    2.403 1    5.498
;
```

The following statements fit a model with separate trends for the two groups; the trends are modeled as B-splines.

```

proc orthoreg data=spline;
  class group;
  effect spl = spline(x);
  model y = group spl*group / noint;
  store ortho_spline;
  title 'B-splines Comparisons';
run;

```

Results from this analysis are shown in [Output 73.6.1](#). The “Parameter Estimates” table shows the estimates for the spline coefficients in the two groups.

Output 73.6.1 Results for Group-Specific Spline Model

B-splines Comparisons					
The ORTHOREG Procedure					
Dependent Variable: y					
Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	13	153.0175561	11.770581238	160.11	<.0001
Error	86	6.3223804119	0.0735160513		
Corrected Total	99	159.33993651			
		Root MSE	0.2711384357		
		R-Square	0.9603214326		
Parameter	DF	Parameter Estimate	Standard Error	t Value	Pr > t
(group='1')	1	9.70265463962039	3.1341899987	3.10	0.0026
(group='2')	1	6.30619220563569	2.6299147768	2.40	0.0187
spl*group 1 1	1	-11.1786451718041	3.7008097395	-3.02	0.0033
spl*group 1 2	1	-20.1946092746139	3.9765046236	-5.08	<.0001
spl*group 2 1	1	-9.53273697995301	3.2575832048	-2.93	0.0044
spl*group 2 2	1	-5.85652496534967	2.7906116773	-2.10	0.0388
spl*group 3 1	1	-8.96118371893294	3.0717508806	-2.92	0.0045
spl*group 3 2	1	-5.55671605245205	2.5716715573	-2.16	0.0335
spl*group 4 1	1	-7.26153231478755	3.243690314	-2.24	0.0278
spl*group 4 2	1	-4.36778889738236	2.7246809593	-1.60	0.1126
spl*group 5 1	1	-6.44615256510896	2.9616955361	-2.18	0.0323
spl*group 5 2	1	-4.03801618914902	2.4588839125	-1.64	0.1042
spl*group 6 1	1	-4.63816959094139	3.7094636319	-1.25	0.2146
spl*group 6 2	1	-4.30290104395061	3.0478540171	-1.41	0.1616
spl*group 7 1	0	0	.	.	.
spl*group 7 2	0	0	.	.	.

By default, the ORTHOREG procedure constructs B-splines with seven knots. Since B-spline coefficients satisfy a sum-to-one constraint and since the model contains group-specific intercepts, the last spline coefficient for each group is redundant and is set to 0.

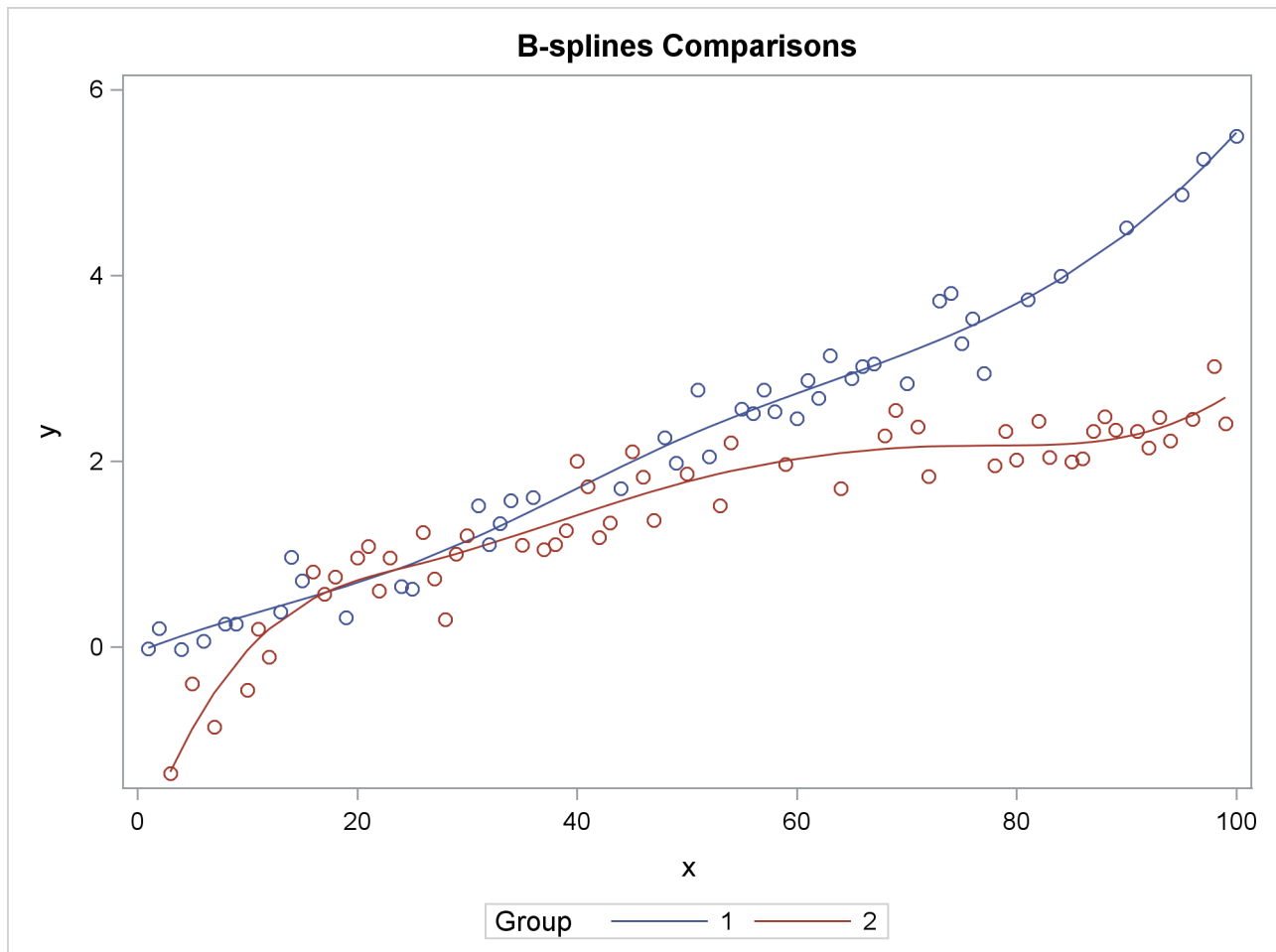
The following statements make a prediction for the input data set by using the SCORE statement with PROC PLM and graph the observed and predicted values in the two groups:

```
proc plm restore=ortho_spline;
  score data=spline out=ortho_pred predicted=p;
run;

proc sgplot data=ortho_pred;
  series y=p x=x / group=group name="fit";
  scatter y=y x=x / group=group;
  keylegend "fit" / title="Group";
run;
```

The prediction plot in [Output 73.6.2](#) suggests that there is some separation of the group trends for small values of x and for values that exceed about $x = 40$.

Output 73.6.2 Observed Data and Predicted Values by Group



In order to determine the range on which the trends separate significantly, the PLM procedure is executed in the following statements with an [ESTIMATE](#) statement that applies group comparisons at a number of values for the spline variable x :


```

%macro GroupDiff;
  %do x=0 %to 75 %by 5;
    "Diff at x=&x" group 1 -1 group*spl [1,1 &x] [-1,2 &x],
  %end;
  'Diff at x=80' group 1 -1 group*spl [1,1 80] [-1,2 80]
%mend;

proc plm restore=ortho_spline;
  show effects;
  estimate %GroupDiff / adjust=simulate seed=1 stepdown;
run;

```

For example, the following ESTIMATE statement compares the trends between the two groups at $x = 25$:

```
estimate 'Diff at x=25' group 1 -1 group*spl [1,1 25] [-1,2 25];
```

The nonpositional syntax is used for the `group*spl` effect. For example, the specification `[-1, 2 25]` requests that the spline be computed at $x = 25$ for the second level of variable `group`. The resulting coefficients are added to the L vector for the estimate after being multiplied with -1 .

Because comparisons are made at a large number of values for x , a multiplicity correction is in order to adjust the p -values to reflect familywise error control. Simulated p -values with step-down adjustment are used here.

Output 73.6.3 displays the “Store Information” for the item store and information about the spline effect (the result of the `SHOW` statement).

Output 73.6.3 Spline Details

B-splines Comparisons	
The PLM Procedure	
Store Information	
Item Store	WORK.ORTHO_SPLINE
Data Set Created From	WORK.SPLINE
Created By	PROC ORTHOREG
Date Created	24OCT13:13:58:36
Response Variable	y
Class Variable	group
Constructed Effect	spl
Model Effects	group spl*group

Output 73.6.3 *continued*

B-splines Comparisons		
The PLM Procedure		
Knots for Spline Effect spl		
Knot Number	Boundary	x
1	*	-48.50000
2	*	-23.75000
3	*	1.00000
4		25.75000
5		50.50000
6		75.25000
7	*	100.00000
8	*	124.75000
9	*	149.50000

B-splines Comparisons		
The PLM Procedure		
Basis Details for Spline Effect spl		
Column	-----Support-----	Support Knots
1	-48.50000 25.75000	1-4
2	-48.50000 50.50000	1-5
3	-23.75000 75.25000	2-6
4	1.00000 100.00000	3-7
5	25.75000 124.75000	4-8
6	50.50000 149.50000	5-9
7	75.25000 149.50000	6-9

Output 73.6.4 displays the results from the **ESTIMATE** statement.

Output 73.6.4 Estimate Results with Multiplicity Correction

Estimates Adjustment for Multiplicity: Holm-Simulated						
Label	Estimate	Standard Error	DF	t Value	Pr > t	Adj P
Diff at x=0	12.4124	4.2130	86	2.95	0.0041	0.0206
Diff at x=5	1.0376	0.1759	86	5.90	<.0001	<.0001
Diff at x=10	0.3778	0.1540	86	2.45	0.0162	0.0545
Diff at x=15	0.05822	0.1481	86	0.39	0.6952	0.9101
Diff at x=20	-0.02602	0.1243	86	-0.21	0.8346	0.9565
Diff at x=25	0.02014	0.1312	86	0.15	0.8783	0.9565
Diff at x=30	0.1023	0.1378	86	0.74	0.4600	0.7418
Diff at x=35	0.1924	0.1236	86	1.56	0.1231	0.2925
Diff at x=40	0.2883	0.1114	86	2.59	0.0113	0.0450
Diff at x=45	0.3877	0.1195	86	3.24	0.0017	0.0096
Diff at x=50	0.4885	0.1308	86	3.74	0.0003	0.0024
Diff at x=55	0.5903	0.1231	86	4.79	<.0001	<.0001
Diff at x=60	0.7031	0.1125	86	6.25	<.0001	<.0001
Diff at x=65	0.8401	0.1203	86	6.99	<.0001	<.0001
Diff at x=70	1.0147	0.1348	86	7.52	<.0001	<.0001
Diff at x=75	1.2400	0.1326	86	9.35	<.0001	<.0001
Diff at x=80	1.5237	0.1281	86	11.89	<.0001	<.0001

Notice that the “Store Information” in [Output 73.6.3](#) displays the classification variables (from the CLASS statement in PROC ORTHOREG), the constructed effects (from the EFFECT statement in PROC ORTHOREG), and the model effects (from the MODEL statement in PROC ORTHOREG). [Output 73.6.4](#) shows that at the 5% significance level the trends are significantly different for $x \leq 10$ and for $x \geq 40$. Between those values you cannot reject the hypothesis of trend congruity.

To see this effect more clearly, you can filter the results by adding the a filtering statement to the previous PROC PLM step:

```
proc plm restore=ortho_spline;
  estimate %GroupDiff / adjust=simulate seed=1 stepdown;
  filter adjp > 0.05;
run;
```

This produces [Output 73.6.5](#), which displays the subset of the results in [Output 73.6.4](#) that meets the condition in the [FILTER](#) expression.

Output 73.6.5 Filtered Estimate Results

B-splines Comparisons						
The PLM Procedure						
Estimates						
Adjustment for Multiplicity: Holm-Simulated						
Label	Estimate	Standard Error	DF	t Value	Pr > t	Adj P
Diff at x=10	0.3778	0.1540	86	2.45	0.0162	0.0545
Diff at x=15	0.05822	0.1481	86	0.39	0.6952	0.9101
Diff at x=20	-0.02602	0.1243	86	-0.21	0.8346	0.9565
Diff at x=25	0.02014	0.1312	86	0.15	0.8783	0.9565
Diff at x=30	0.1023	0.1378	86	0.74	0.4600	0.7418
Diff at x=35	0.1924	0.1236	86	1.56	0.1231	0.2925

Example 73.7: Linear Inference with Arbitrary Estimates

Suppose that you have calculated a vector of parameter estimates of dimension $(p \times 1)$ and its associated variance-covariance matrix by some statistical method. You might want to use these results to perform linear inference, or to score a data set and calculate predicted values and their standard errors.

The following DATA steps create two SAS data sets. The first, called `parms`, contains six estimates that represent two uncorrelated groups. The data set `cov` contains the covariance matrix of the estimates. The lack of correlation between the two sets of three parameters is evident in the block-diagonal structure of the covariance matrix.

```

data parms;
    length name $6;
    input Name$ Value;
    datalines;
alpha1 -3.5671
beta1   0.4421
gamma1 -2.6230
alpha2 -3.0111
beta2   0.3977
gamma2 -2.4442
;

data cov;
    input Parm row col1-col6;
    datalines;
1 1  0.007462 -0.005222  0.010234  0.000000  0.000000  0.000000
1 2 -0.005222  0.048197 -0.010590  0.000000  0.000000  0.000000
1 3  0.010234 -0.010590  0.215999  0.000000  0.000000  0.000000
1 4  0.000000  0.000000  0.000000  0.031261 -0.009096  0.015785
1 5  0.000000  0.000000  0.000000 -0.009096  0.039487 -0.019996
1 6  0.000000  0.000000  0.000000  0.015785 -0.019996  0.126172
;

```

Suppose that you are interested in testing whether the parameters are homogeneous across groups—that is, whether $\alpha_1 = \alpha_2$, $\beta_1 = \beta_2$, $\gamma_1 = \gamma_2$. You are interested in testing the hypothesis jointly and separately with multiplicity adjustment.

To use the PLM procedure, you first need to create an item store that contains the necessary information as if the preceding parameter vector and covariance matrix were the result of a statistical modeling procedure. The following statements use the GLIMMIX procedure to create such an item store, by fitting a saturated linear model with the data set that contains the parameter estimates serving as the input data set:

```

proc glimmix data=parms order=data;
    class Name;
    model Value = Name / noint ddfm=none s;
    random _residual_ / type=lin(1) ldata=cov v;
    parms (1) / noiter;
    store ArtificialModel;
    title 'Linear Inference';
run;

```

The RANDOM statement is used to form the covariance structure for the estimates. The PARMS statement prevents iterative updates of the covariance parameters. The resulting marginal covariance matrix of the “data” is thus identical to the covariance matrix in the data set cov. The ORDER=DATA option in the PROC GLIMMIX statement is used to arrange the levels of the classification variable Name in the order in which they appear in the data set so that the order of the parameters matches that of the covariance matrix.

The results of this analysis are shown in [Output 73.7.1](#). Note that the parameter estimates are identical to the values passed in the input data set and their standard errors equal the square root of the diagonal elements of the cov data set.

Output 73.7.1 “Fitted” Parameter Estimates and Covariance Matrix

Linear Inference						
The GLIMMIX Procedure						
Estimated V Matrix for Subject 1						
Row	Col1	Col2	Col3	Col4	Col5	Col6
1	0.007462	-0.00522	0.01023			
2	-0.00522	0.04820	-0.01059			
3	0.01023	-0.01059	0.2160			
4				0.03126	-0.00910	0.01579
5				-0.00910	0.03949	-0.02000
6				0.01579	-0.02000	0.1262
Solutions for Fixed Effects						
Effect	name	Estimate	Standard Error	DF	t Value	Pr > t
name	alpha1	-3.5671	0.08638	Infty	-41.29	<.0001
name	beta1	0.4421	0.2195	Infty	2.01	0.0440
name	gamma1	-2.6230	0.4648	Infty	-5.64	<.0001
name	alpha2	-3.0111	0.1768	Infty	-17.03	<.0001
name	beta2	0.3977	0.1987	Infty	2.00	0.0454
name	gamma2	-2.4442	0.3552	Infty	-6.88	<.0001

There are other ways to fit a saturated model with the GLIMMIX procedure. For example, you can use the TYPE=UN covariance structure in the RANDOM statement with a properly prepared input data set for the PDATA= option in the PARMS statement. See Example 17 in Chapter 43.33, “[Examples: GLIMMIX Procedure](#),” for details.

Once the item store exists, you can apply the linear inference capabilities of the PLM procedure. For example, the **ESTIMATE** statement in the following statements test the hypothesis of parameter homogeneity across groups:

```
proc plm restore=ArtificialModel;
  estimate
    'alpha1 = alpha2' Name 1 0 0 -1 0 0,
    'beta1 = beta2 ' Name 0 1 0 0 -1 0,
    'gamma1 = gamma2' Name 0 0 1 0 0 -1 /
    adjust=bon stepdown ftest(label='Homogeneity');
run;
```

Output 73.7.2 Results from the PLM Procedure

Linear Inference						
The PLM Procedure						
Estimates						
Adjustment for Multiplicity: Holm						
Label	Estimate	Standard Error	DF	t Value	Pr > t	Adj P
alpha1 = alpha2	-0.5560	0.1968	Infty	-2.83	0.0047	0.0142
beta1 = beta2	0.04440	0.2961	Infty	0.15	0.8808	1.0000
gamma1 = gamma2	-0.1788	0.5850	Infty	-0.31	0.7599	1.0000
F Test for Estimates						
Label	Num DF	Den DF	F Value	Pr > F		
Homogeneity	3	Infty	2.79	0.0389		

The F test in [Output 73.7.2](#) shows that the joint test of homogeneity is rejected. The individual tests with familywise control of the Type I error show that the overall difference is due to a significant change in the α parameters. The hypothesis of homogeneity across the two groups cannot be rejected for the β and γ parameters.

References

- Asuncion, A. and Newman, D. J. (2007), "UCI Machine Learning Repository," <http://archive.ics.uci.edu/ml/>.
- Kenward, M. G. and Roger, J. H. (1997), "Small Sample Inference for Fixed Effects from Restricted Maximum Likelihood," *Biometrics*, 53, 983–997.
- McCullagh, P. and Nelder, J. A. (1989), *Generalized Linear Models*, 2nd Edition, London: Chapman & Hall.
- Silvapulle, M. J. and Sen, P. K. (2004), *Constrained Statistical Inference: Order, Inequality, and Shape Constraints*, New York: John Wiley & Sons.
- Weisberg, S. (1985), *Applied Linear Regression*, 2nd Edition, New York: John Wiley & Sons.

Subject Index

alpha level

PLM procedure, [6209](#)

degrees of freedom

PLM procedure, [6209](#)

options summary

ESTIMATE statement, [6203](#)

PLM procedure

alpha level, [6209](#)

BY processing, [6215](#)

common postprocessing statements, [6189](#)

degrees of freedom, [6209](#)

filter PLM results, [6204](#)

item store, [6188](#)

least squares means, [6211](#)

ODS graph names, [6219](#)

ODS Graphics, [6200](#)

ODS table names, [6218](#)

posterior inference, [6215](#)

scoring statistics, [6210](#)

Scoring Zero-Inflated Models, [6217](#)

user-defined formats, [6217](#)

scoring statistics

PLM procedure, [6210](#)

Syntax Index

- ALL option
 - SHOW statement (PLM), [6211](#)
- ALPHA= option
 - PROC PLM statement (PLM), [6198](#)
 - SCORE statement (PLM), [6209](#)
- BYVAR option
 - SHOW statement (PLM), [6211](#)
- CLASS option
 - SHOW statement (PLM), [6211](#)
- CODE statement
 - PLM procedure, [6202](#)
- CORRELATION option
 - SHOW statement (PLM), [6211](#)
- COVARIANCE option
 - SHOW statement (PLM), [6212](#)
- DATA= option
 - SCORE statement (PLM), [6208](#)
- DDFMETHOD= option
 - PROC PLM statement (PLM), [6199](#)
- DF= option
 - SCORE statement (PLM), [6209](#)
- EFFECTPLOT statement
 - PLM procedure, [6202](#)
- EFFECTS option
 - SHOW statement (PLM), [6212](#)
- ESTEPS= option
 - PROC PLM statement (PLM), [6199](#)
- ESTIMATE statement
 - PLM procedure, [6203](#)
- FILTER statement
 - PLM procedure, [6204](#)
- FITSTATS option
 - SHOW statement (PLM), [6212](#)
- FORMAT= option
 - PROC PLM statement (PLM), [6199](#)
- HERMITE option
 - SHOW statement (PLM), [6212](#)
- HESSIAN option
 - SHOW statement (PLM), [6212](#)
- ILINK option
 - SCORE statement (PLM), [6209](#)
- LSMEANS statement
 - PLM procedure, [6206](#)
- LSMESTIMATE statement
 - PLM procedure, [6207](#)
- MAXLEN= option
 - PROC PLM statement (PLM), [6199](#)
- NOCLPRINT option
 - PROC PLM statement (PLM), [6199](#)
- NOINFO option
 - PROC PLM statement (PLM), [6200](#)
- NOOFFSET option
 - SCORE statement (PLM), [6209](#)
- NOPRINT option
 - PROC PLM statement (PLM), [6200](#)
- NOUNIQUE option
 - SCORE statement (PLM), [6210](#)
- NOVAR option
 - SCORE statement (PLM), [6210](#)
- OBSCAT option
 - SCORE statement (PLM), [6210](#)
- OUT= option
 - SCORE statement (PLM), [6209](#)
- PARAMETERS option
 - SHOW statement (PLM), [6212](#)
- PERCENTILES= option
 - PROC PLM statement (PLM), [6200](#)
- PLM procedure, [6197](#)
 - FILTER statement, [6204](#)
 - PROC PLM statement, [6198](#)
 - SHOW statement, [6211](#)
 - syntax, [6197](#)
 - WHERE statement, [6213](#)
- PLM procedure, FILTER statement, [6204](#)
- PLM procedure, PROC PLM statement, [6198](#)
 - ALPHA= option, [6198](#)
 - DDFMETHOD= option, [6199](#)
 - ESTEPS= option, [6199](#)
 - FORMAT= option, [6199](#)
 - MAXLEN= option, [6199](#)
 - NOCLPRINT option, [6199](#)
 - NOINFO option, [6200](#)
 - PERCENTILES= option, [6200](#)
 - PLOT option, [6200](#)
 - PLOTS option, [6200](#)
 - RESTORE= option, [6201](#)
 - SEED= option, [6201](#)

- SINGCHOL= option, [6201](#)
- SINGRES= option, [6201](#)
- SINGULAR= option, [6201](#)
- STMTORDER= option, [6201](#)
- WHEREFORMAT option, [6201](#)
- ZETA= option, [6202](#)
- PLM procedure, SCORE statement
 - ALPHA= option, [6209](#)
 - DATA= option, [6208](#)
 - DF= option, [6209](#)
 - ILINK option, [6209](#)
 - NOOFFSET option, [6209](#)
 - NOUNIQUE option, [6210](#)
 - NOVAR option, [6210](#)
 - OBSCAT option, [6210](#)
 - OUT= option, [6209](#)
 - SAMPLE option, [6210](#)
- PLM procedure, SHOW statement, [6211](#)
 - ALL option, [6211](#)
 - BYVAR option, [6211](#)
 - CLASS option, [6211](#)
 - CORREATION option, [6211](#)
 - COVARIANCE option, [6212](#)
 - EFFECTS option, [6212](#)
 - FITSTATS option, [6212](#)
 - HERMITE option, [6212](#)
 - HESSIAN option, [6212](#)
 - PARAMETERS option, [6212](#)
 - PROGRAM option, [6212](#)
 - XPX option, [6212](#)
 - XPXI option, [6212](#)
- PLM procedure, WHERE statement, [6213](#)
- PLM procedure, CODE statement, [6202](#)
- PLM procedure, EFFECTPLOT statement, [6202](#)
- PLM procedure, ESTIMATE statement, [6203](#)
- PLM procedure, LSMEANS statement, [6206](#)
- PLM procedure, LSMESTIMATE statement, [6207](#)
- PLM procedure, SLICE statement, [6213](#)
- PLM procedure, TEST statement, [6213](#)
- PLOT option
 - PROC PLM statement, [6200](#)
- PLOTS option
 - PROC PLM statement, [6200](#)
- PROC PLM statement, *see* PLM procedure
 - PLM procedure, [6198](#)
- PROGRAM option
 - SHOW statement (PLM), [6212](#)
- RESTORE= option
 - PROC PLM statement (PLM), [6201](#)
- SAMPLE option
 - SCORE statement (PLM), [6210](#)
- SEED= option
 - PROC PLM statement (PLM), [6201](#)
- SHOW statement
 - PLM procedure, [6211](#)
- SINGCHOL= option
 - PROC PLM statement (PLM), [6201](#)
- SINGRES= option
 - PROC PLM statement (PLM), [6201](#)
- SINGULAR= option
 - PROC PLM statement (PLM), [6201](#)
- SLICE statement
 - PLM procedure, [6213](#)
- STMTORDER= option
 - PROC PLM statement (PLM), [6201](#)
- TEST statement
 - PLM procedure, [6213](#)
- WHERE statement
 - PLM procedure, [6213](#)
- WHEREFORMAT option
 - PROC PLM statement (PLM), [6201](#)
- XPX option
 - SHOW statement (PLM), [6212](#)
- XPXPI option
 - SHOW statement (PLM), [6212](#)
- ZETA= option
 - PROC PLM statement (PLM), [6202](#)