THE POWER TO KNOW.

SAS/STAT® 14.3 User's Guide

The TRANSREG Procedure

This document is an individual chapter from *SAS/STAT® 14.3 User's Guide*.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2017. *SAS/STAT® 14.3 User's Guide*. Cary, NC: SAS Institute Inc.

SAS/STAT® 14.3 User's Guide

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

September 2017

Chapter 120

The TRANSREG Procedure

Contents

Overview: TRANSREG Procedure

The TRANSREG (transformation regression) procedure fits linear models, optionally with smooth, spline, Box-Cox, and other nonlinear transformations of the variables. You can use PROC TRANSREG to fit a curve through a scatter plot or fit multiple curves, one for each level of a classification variable. You can also constrain the functions to be parallel or monotone or have the same intercept. PROC TRANSREG can be used to code experimental designs and classification variables prior to their use in other analyses.

The TRANSREG procedure fits many types of linear models, including the following:

- ordinary regression and ANOVA
- metric and nonmetric conjoint analysis (Green and Wind 1975; De Leeuw, Young, and Takane 1976)
- linear models with Box-Cox (1964) transformations of the dependent variables
- regression with a smooth (Reinsch 1967), spline (De Boor 1978; Van Rijckevorsel 1982), monotone spline (Winsberg and Ramsay 1980), or penalized B-spline (Eilers and Marx 1996) fit function
- metric and nonmetric vector and ideal point preference mapping (Carroll 1972)
- simple, multiple, and multivariate regression with variable transformations (Young, de Leeuw, and Takane 1976; Winsberg and Ramsay 1980; Breiman and Friedman 1985)
- redundancy analysis (Stewart and Love 1968) with variable transformations (Israels 1984)
- canonical correlation analysis with variable transformations (Van der Burg and de Leeuw 1983)
- response surface regression (Myers 1976; Khuri and Cornell 1987) with variable transformations

The data set can contain variables measured on nominal, ordinal, interval, and ratio scales (Siegel 1956). You can specify any mix of these variable types for the dependent and independent variables. PROC TRANSREG can do the following:

- transform nominal variables by scoring the categories to minimize squared error (Fisher 1938), or treat nominal variables as classification variables
- transform ordinal variables by monotonically scoring the ordered categories so that order is weakly preserved (adjacent categories can be merged) and squared error is minimized. Ties can be optimally untied or left tied (Kruskal 1964). Ordinal variables can also be transformed to ranks.
- transform interval and ratio scale of measurement variables linearly or nonlinearly with spline (De Boor 1978; Van Rijckevorsel 1982), monotone spline (Winsberg and Ramsay 1980), penalized B-spline (Eilers and Marx 1996), smooth (Reinsch 1967), or Box-Cox (Box and Cox 1964) transformations. In addition, logarithmic, exponential, power, logit, and inverse trigonometric sine transformations are available.

Transformations produced by the PROC TRANSREG multiple regression algorithm, requesting spline transformations, are often similar to transformations produced by the ACE smooth regression method of Breiman and Friedman (1985). However, ACE does not explicitly optimize a loss function (De Leeuw 1986), while PROC TRANSREG explicitly minimizes a squared-error criterion.

PROC TRANSREG extends the ordinary general linear model by providing optimal variable transformations that are iteratively derived. PROC TRANSREG iterates until convergence, alternating two major steps: finding least squares estimates of the model parameters given the current scoring of the data, and finding least squares estimates of the scoring parameters given the current set of model parameters. This is called the method of alternating least squares (Young 1981).

For more background on alternating least squares optimal scaling methods and transformation regression methods, see Young, de Leeuw, and Takane (1976); Winsberg and Ramsay (1980); Young (1981); Gifi (1990); Schiffman, Reynolds, and Young (1981); Van der Burg and de Leeuw (1983); Israels (1984); Breiman and Friedman (1985); Hastie and Tibshirani (1986). (These are just a few of the many relevant sources.)

Getting Started: TRANSREG Procedure

This section provides several examples that illustrate a few of the more basic features of PROC TRANSREG.

Fitting a Curve through a Scatter Plot

PROC TRANSREG can fit curves through data and detect nonlinear relationships among variables. This example uses a subset of the data from an experiment in which nitrogen oxide emissions from a single cylinder engine are measured for various combinations of fuel and equivalence ratio (Brinkman 1981). This gas data set is available from the Sashelp library. The following step creates a subset of the data for analysis:

```
title 'Gasoline and Emissions Data';

data gas;
   set sashelp.gas;
   if fuel in ('Ethanol', '82rongas', 'Gasohol');
run;
```

The next step fits a spline or curve through the data and displays the regression results. For information about splines and knots, see the sections "Smoothing Splines" on page 9962, "Linear and Nonlinear Regression Functions" on page 9949, "Simultaneously Fitting Two Regression Functions" on page 9953, and "Using Splines and Knots" on page 9932, as well as Example 120.1. The following statements produce Figure 120.1:

```
ods graphics on;

* Request a Spline Transformation of Equivalence Ratio;
proc transreg data=Gas solve ss2 plots=(transformation obp residuals);
   model identity(nox) = spline(EqRatio / nknots=4);
run;
```

The SOLVE algorithm option, or *a-option*, requests a direct solution for both the transformation and the parameter estimates. For many models, PROC TRANSREG with the SOLVE *a-option* can produce exact results without iteration. The SS2 (Type II sums of squares) *a-option* requests regression and ANOVA results. The PLOTS= option requests plots of the variable transformations, a plot of the observed values by the predicted values, and a plot of the residuals. The dependent variable NOx was specified with an IDENTITY transformation, which means that it will not be transformed, just as in ordinary regression. The independent variable EqRatio, in contrast, is transformed by using a cubic spline with four knots. The NKNOTS= option is known as a transformation option, or *t-option*. Graphical results are enabled when ODS Graphics is enabled. The results are shown in Figure 120.1 through Figure 120.5.

Figure 120.1 Iteration, ANOVA, and Regression Results

Gasoline and Emissions Data

The TRANSREG Procedure

Dependent Variable Identity(NOx)
Nitrogen Oxide

Number of Observations Read	112
Number of Observations Used	110

TRANSREG MORALS Algorithm Iteration History for Identity(NOx)

Iteration Number	Average Change	Maximum Change	R-Square	Criterion Change	Note
0	1.04965	3.46121	0.00917		
1	0.00000	0.00000	0.82429	0.81512	Converged

Algorithm converged.

The TRANSREG Procedure Hypothesis Tests for Identity(NOx)
Nitrogen Oxide

Univariate ANOVA Table Based on the Usual Degrees of Freedom

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	7	180.0951	25.72788	68.36	<.0001
Error	102	38.3891	0.37636		
Corrected Total	109	218.4842			

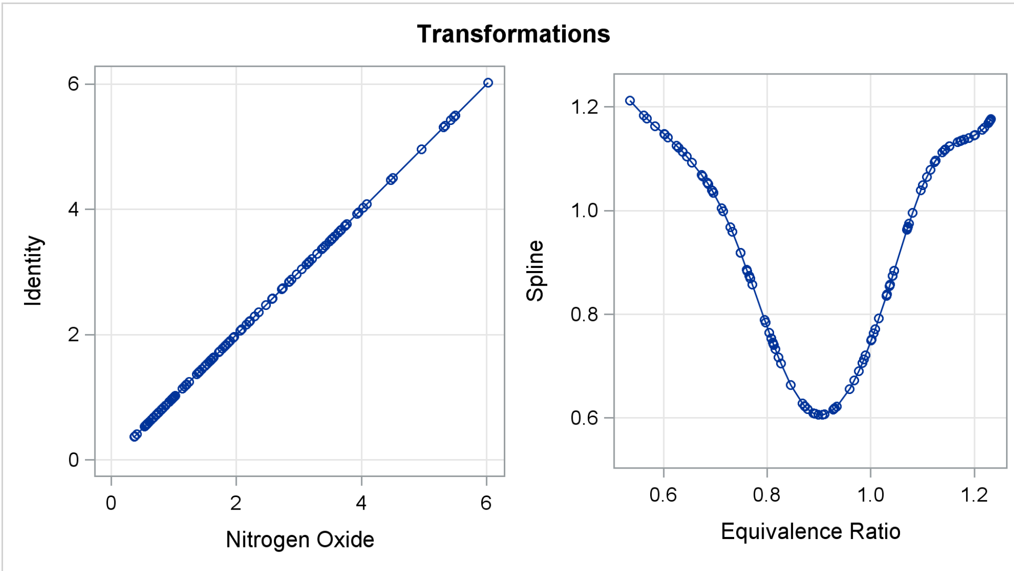
Root MSE	0.61348	**R-Square**	0.8243
Dependent Mean	2.25022	**Adj R-Sq**	0.8122
Coeff Var	27.26334		

Univariate Regression Table Based on the Usual Degrees of Freedom

Variable	DF	Coefficient	Type II Sum of Squares	Mean Square	F Value	Pr > F	Label
Intercept	1	8.3165407	324.065	324.065	861.04	<.0001	Intercept
Spline(EqRatio)	7	-6.5740158	180.095	25.728	68.36	<.0001	Equivalence Ratio

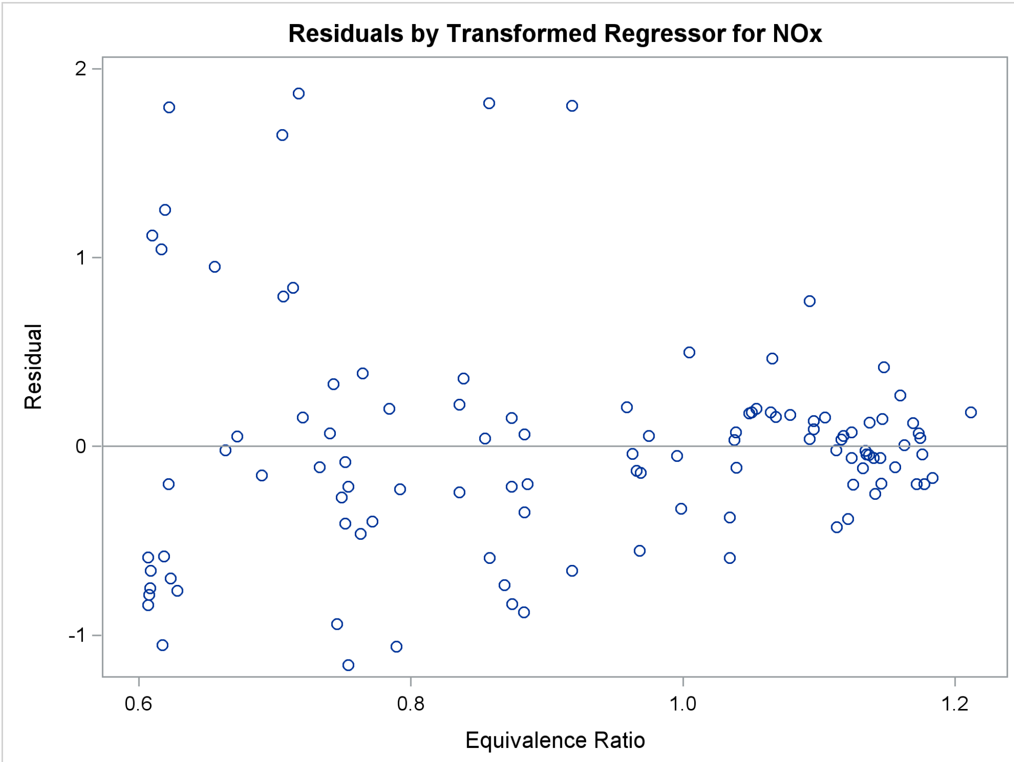
PROC TRANSREG increases the squared multiple correlation from the original value of 0.00917 to 0.82429. Iteration 0 shows the fit before the data are transformed, and iteration 1 shows the fit after the transformation, which was directly solved for in the initial iteration. The change values for iteration 0 show the change from the original EqRatio variable to the transformed EqRatio variable. For this model, no improvement on the initial solution is possible, so in iteration 1, all change values are zero. The ANOVA and regression results show that you are fitting a model with 7 model parameters, 4 knots plus a degree 3 or cubic spline. The overall model fit is identical to the test for the spline transformation, since there is only one term in the model besides the intercept, and the results are significant at the 0.0001 level. The transformations are shown next in Figure 120.2.

Figure 120.2 Transformations



The transformation plots show the identity transformation of NOx and the nonlinear spline transformation of EqRatio. These plots are requested with the PLOTS=TRANSFORMATION option. The plot on the left shows that NOx is unchanged, which is always the case with the IDENTITY transformation. In contrast, the spline transformation of EqRatio is nonlinear. It is this nonlinear transformation of EqRatio that accounts for the increase in fit that is shown in the iteration history table.

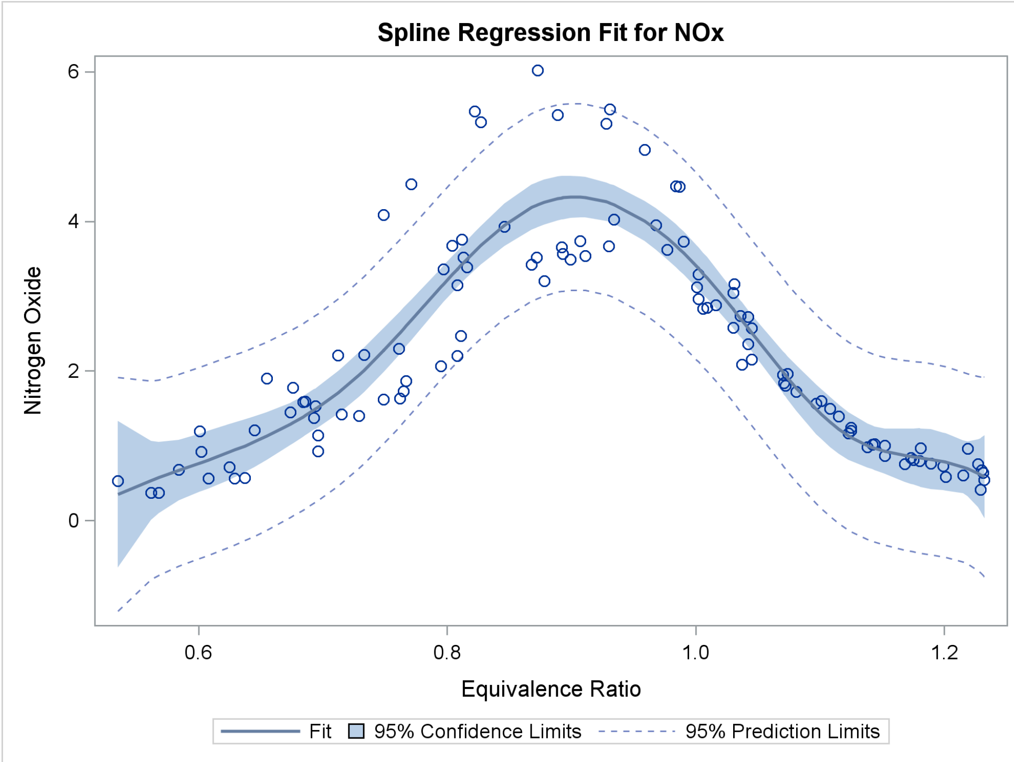
Figure 120.3 Residuals



The residuals plot in Figure 120.3 shows the residuals as a function of the transformed independent variable.

The "Spline Regression Fit" plot in Figure 120.4 displays the nonlinear regression function plotted through the original data, along with 95% confidence and prediction limits. This plot clearly shows that nitrous oxide emissions are largest in the middle range of equivalence ratio, 0.08 to 1.0, and are much lower for the extreme values of equivalence ratio, such as around 0.6 and 1.2.

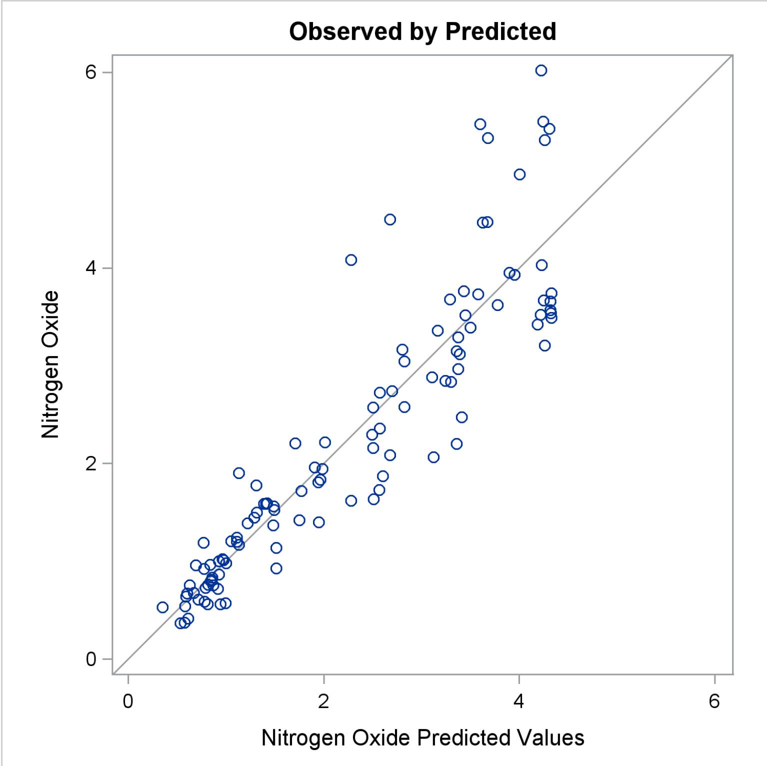
Figure 120.4 Fitting a Curve through a Scatter Plot



This plot is produced by default when ODS Graphics is enabled and when there is an IDENTITY dependent variable and one non-CLASS independent variable. The plot consists of an ordinary scatter plot of NOx plotted as a function of EqRatio. It also contains the predicted values of NOx, which are a function of the spline transformation of EqRatio (or TEqRatio shown previously), and are plotted as a function of EqRatio. Similarly, it contains confidence limits based on NOx and TEqRatio.

The "Observed by Predicted" values plot in Figure 120.5 displays the dependent variable plotted as a function of the regression predicted values along with a linear regression line, which for this plot always has a slope of 1. This plot was requested with the OBP or OBSERVEDBYPREDICTED suboption in the PLOTS= option. The residual differences between the transformed data and the regression line show how well the nonlinearly transformed data fit a linear-regression model. The residuals look mostly random; however, they are larger for larger values of NOx, suggesting that maybe this is not the optimal model. You can also see this by examining the fit of the function through the original scatter plot in Figure 120.4. Near the middle of the function, the residuals are much larger. You can refit the model, this time requesting separate functions for each type of fuel. You can request the original scatter plot, without any regression information and before the variables are transformed, by specifying the SCATTER suboption in the PLOTS= option.

Figure 120.5 Observed by Predicted



These next statements fit an additive model with separate functions for each of the different fuels. The statements produce Figure 120.6 through Figure 120.9.

```
* Separate Curves and Intercepts;
proc transreg data=Gas solve ss2 additive plots=(transformation obp);
   model identity(nox) = class(Fuel / zero=none) |
                         spline(EqRatio / nknots=4 after);
run;
```

The ADDITIVE *a-option* requests an additive model, where the regression coefficients are absorbed into the transformations, and so the final regression coefficients are all one. The specification CLASS(Fuel / ZERO=NONE) recodes fuel into a set of three binary variables, one for each of the three fuels in this data set. The vertical bar between the CLASS and SPLINE specifications request both main effects and interactions. For this model, it requests both a separate intercept and a separate spline function for each fuel. The original two variables, Fuel and EqRatio, are replaced by six variables—three binary intercept terms and three spline variables. The three spline variables are zero when their corresponding intercept binary variable is zero, and nonzero otherwise. The nonzero parts are optimally transformed by the analysis. The AFTER *t-option* specified with the SPLINE transformation specifies that the four knots should be selected independently for each of the three spline transformations, *after* EqRatio is crossed with the CLASS variable. Alternatively, and by default, the knots are chosen by examining EqRatio before it is crossed with the CLASS variable, and the same knots are used for all three transformations. The results are shown in Figure 120.6.

Figure 120.6 Iteration, ANOVA, and Regression Results

Gasoline and Emissions Data

The TRANSREG Procedure

Dependent Variable Identity(NOx)
Nitrogen Oxide

Class Level Information		
Class	**Levels**	**Values**
Fuel	3	82rongas Ethanol Gasohol

Number of Observations Read	112
Number of Observations Used	110
Implicit Intercept Model	

TRANSREG MORALS Algorithm Iteration History for Identity(NOx)

Iteration Number	Average Change	Maximum Change	R-Square	Criterion Change	Note
0	0.12476	1.13866	0.18543		
1	0.00000	0.00000	0.95870	0.77327	Converged

Algorithm converged.

Hypothesis Test Iterations Excluding Spline(Fuel82rongasEqRatio)

TRANSREG MORALS Algorithm Iteration History for Identity(NOx)

Iteration Number	Average Change	Maximum Change	R-Square	Criterion Change	Note
0	0.00000	0.00000	0.80234		
1	0.00000	0.00000	0.80234	-.00000	Converged

Algorithm converged.

Hypothesis Test Iterations Excluding Spline(FuelEthanolEqRatio)

TRANSREG MORALS Algorithm Iteration History for Identity(NOx)

Iteration Number	Average Change	Maximum Change	R-Square	Criterion Change	Note
0	0.00000	0.00000	0.48801		
1	0.00000	0.00000	0.48801	-.00000	Converged

Algorithm converged.

Figure 120.6 *continued*

Hypothesis Test Iterations Excluding Spline(FuelGasoholEqRatio)

TRANSREG MORALS Algorithm Iteration History for Identity(NOx)

Iteration Number	Average Change	Maximum Change	R-Square	Criterion Change	Note
0	0.00000	0.00000	0.80052		
1	0.00000	0.00000	0.80052	-.00000	Converged

Algorithm converged.

The TRANSREG Procedure Hypothesis Tests for Identity(NOx)
Nitrogen Oxide

Univariate ANOVA Table Based on the Usual Degrees of Freedom

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	23	209.4613	9.107012	86.80	<.0001
Error	86	9.0229	0.104918		
Corrected Total	109	218.4842			

Root MSE	0.32391	**R-Square**	0.9587
Dependent Mean	2.25022	**Adj R-Sq**	0.9477
Coeff Var	14.39461		

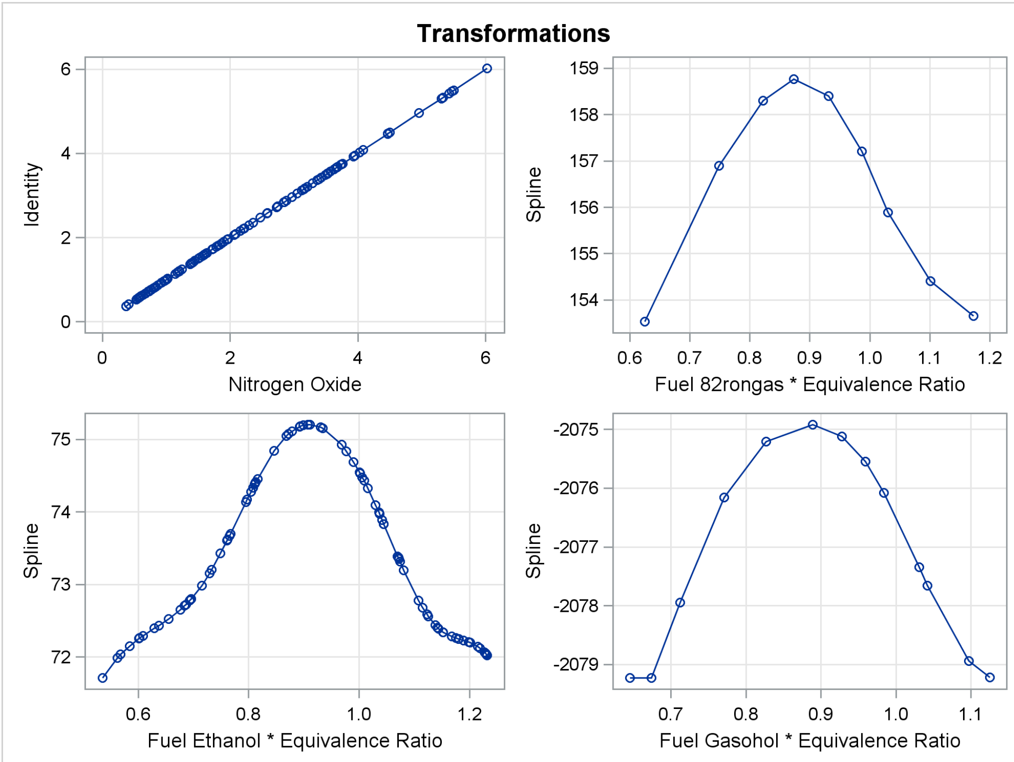
Univariate Regression Table Based on the Usual Degrees of Freedom

Variable	DF	Coefficient	Type II Sum of Squares	Mean Square	F Value	Pr > F	Label
Class.Fuel82rongas	1	1.00000000	32.634	32.6338	311.04	<.0001	Fuel 82rongas
Class.FuelEthanol	1	1.00000000	97.406	97.4058	928.40	<.0001	Fuel Ethanol
Class.FuelGasohol	1	1.00000000	34.672	34.6720	330.47	<.0001	Fuel Gasohol
Spline(Fuel82rongasEqRatio)	7	1.00000000	34.162	4.8803	46.52	<.0001	Fuel 82rongas * Equivalence Ratio
Spline(FuelEthanolEqRatio)	7	1.00000000	102.840	14.6914	140.03	<.0001	Fuel Ethanol * Equivalence Ratio
Spline(FuelGasoholEqRatio)	7	1.00000000	34.561	4.9372	47.06	<.0001	Fuel Gasohol * Equivalence Ratio

ZERO=SUM and ZERO=NONE coefficient tests are not exact when there are iterative transformations. Those tests are performed holding all transformations fixed, and so are generally liberal.

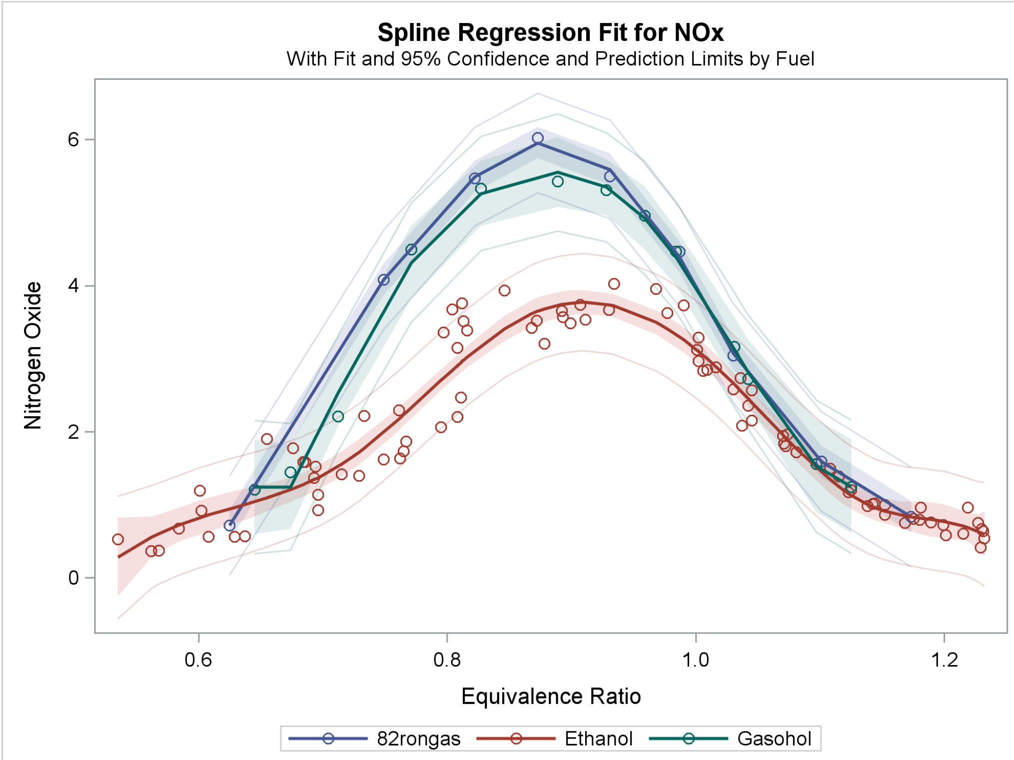
The first iteration history table in Figure 120.6 shows that PROC TRANSREG increases the squared multiple correlation from the original value of 0.18543 to 0.95870. The remaining iteration histories pertain to PROC TRANSREG's process of comparing models to test hypotheses. The important thing to look for is convergence in all of the tables.

Figure 120.7 Transformations



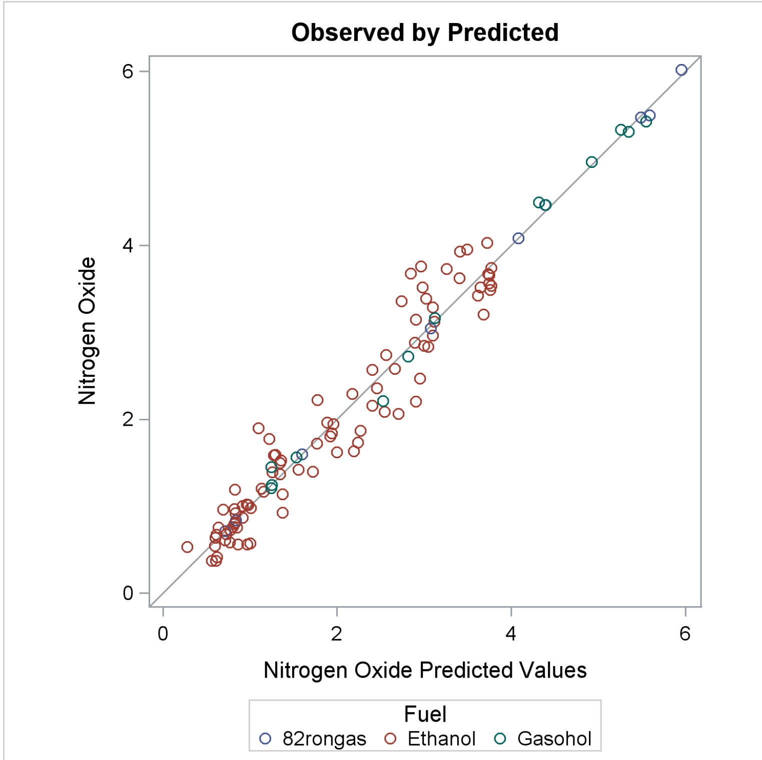
The transformations, shown in Figure 120.7, show that for all three groups, the transformation of EqRatio is approximately quadratic.

Figure 120.8 Fitting Curves through a Scatter Plot



The fit plot, shown in Figure 120.8, shows that there are in fact three distinct functions in the data. The increase in fit over the previous model comes from individually fitting each group instead of providing an aggregate fit.

Figure 120.9 Observed by Predicted



The residuals in the observed by predicted plot displayed in Figure 120.9 are much better for this analysis.

You could fit a model that is "in between" the two models shown previously. This next model provides for separate intercepts for each group, but calls for a common function. There are still three functions, one per group, but their shapes are the same, and they are equidistant or parallel. This model is requested by omitting the vertical bar so that separate intercepts are requested, but not separate curves within each group. The following statements fit the separate intercepts model and create Figure 120.10:

```
* Separate Intercepts;
proc transreg data=Gas solve ss2 additive;
   model identity(nox) = class(Fuel / zero=none)
                         spline(EqRatio / nknots=4);
run;
```

The ANOVA table and fit plot are shown in Figure 120.10.

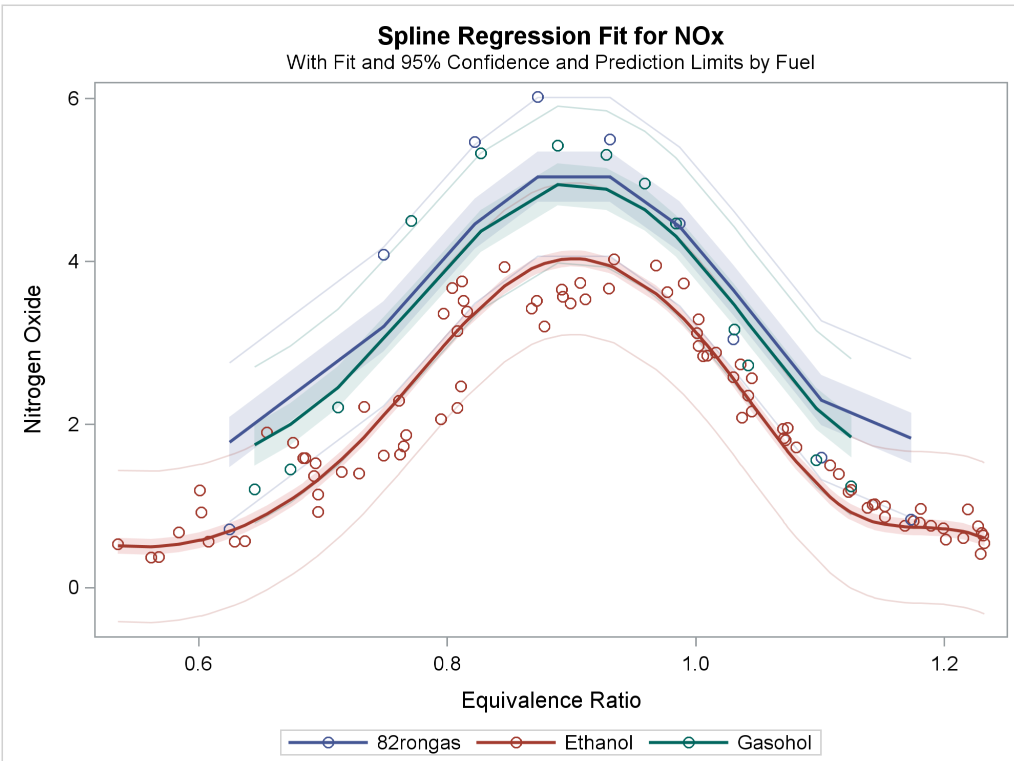
Figure 120.10 Separate Intercepts Only

Gasoline and Emissions Data

The TRANSREG Procedure

Univariate ANOVA Table Based on the Usual Degrees of Freedom					
Source	**DF**	**Sum of Squares**	**Mean Square**	**F Value**	**Pr > F**
Model	9	196.7548	21.86165	100.61	<.0001
Error	100	21.7294	0.21729		
Corrected Total	109	218.4842			

Figure 120.10 *continued*



Now, squared multiple correlation is 0.9005, which is smaller than the model with the unconstrained separate curves, but larger than the model with only one curve. Because of the restrictions on the shapes, these curves do not track the data as well as the previous model. However, this model is more parsimonious with many fewer parameters.

There are other ways to fit curves through scatter plots in PROC TRANSREG. For example, you could use smoothing splines or penalized B-splines, as is illustrated next. The following statements fit separate curves through each group by using penalized B-splines and produce Figure 120.11:

```
* Separate Curves and Intercepts with Penalized B-Splines;
proc transreg data=Gas ss2 plots=transformation lprefix=0;
   model identity(nox) = class(Fuel / zero=none) * pbspline(EqRatio);
run;
```

This example asks for a separate penalized B-spline transformation, PBSPLINE, of equivalence ratio for each type of fuel. The LPREFIX=0 *a-option* is specified in the PROC statement so that zero characters of the CLASS variable name (Fuel) are used in constructing the labels for the coded variables. The result is label components like "Ethanol" instead of the more redundant "Fuel Ethanol". The results of this analysis are shown in Figure 120.11.

Figure 120.11 Penalized B-Splines

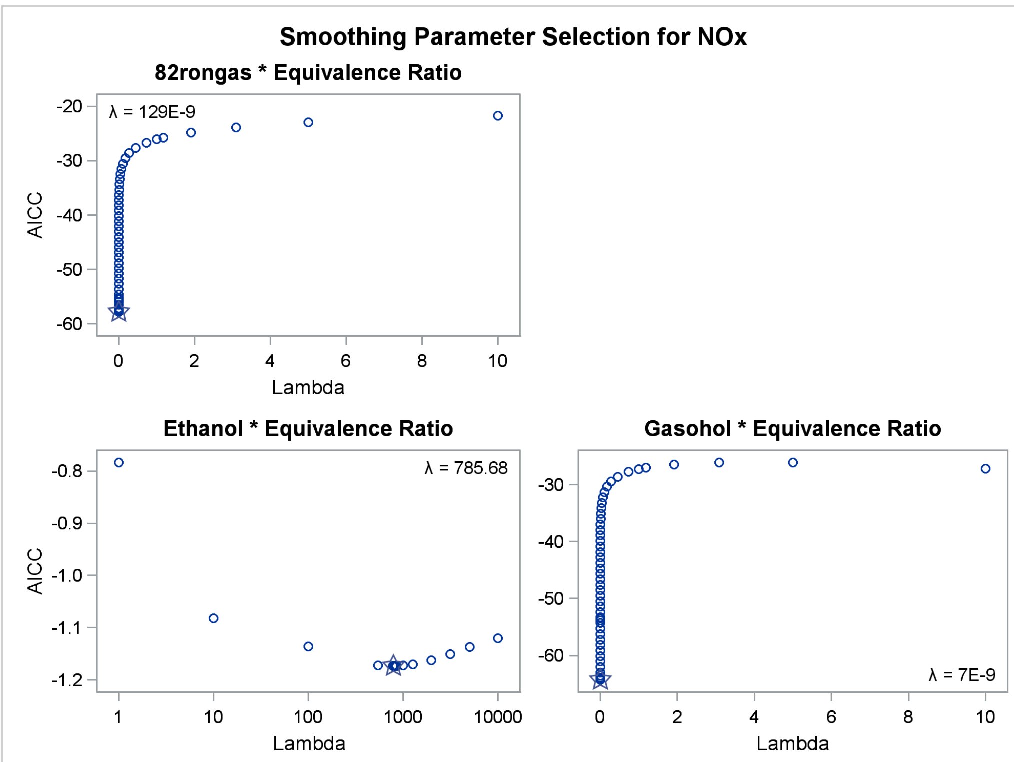


Figure 120.11 *continued*

Dependent Variable Identity(NOx)
Nitrogen Oxide

Class Level Information		
Class	**Levels**	**Values**
Fuel	3	82rongas Ethanol Gasohol

Number of Observations Read	112
Number of Observations Used	110
Implicit Intercept Model	

Figure 120.11 *continued*

TRANSREG Univariate Algorithm Iteration History for Identity(NOx)			
Iteration Number	**Average Change**	**Maximum Change**	**Note**
1	0.00000	0.00000	Converged

Algorithm converged.

The TRANSREG Procedure Hypothesis Tests for Identity(NOx) Nitrogen Oxide

Univariate ANOVA Table, Penalized B-Spline Transformation

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	33.194	211.4818	6.371106	68.97	<.0001
Error	75.806	7.0024	0.092373		
Corrected Total	109	218.4842			

Root MSE	0.30393	**R-Square**	0.9680
Dependent Mean	2.25022	**Adj R-Sq**	0.9539
Coeff Var	13.50663		

Penalized B-Spline Transformation

Variable	DF	Coefficient	Lambda	AICC	Label
Pbspline(Fuel82rongasEqRatio)	9.000	1.000	1.287E-7	-57.7841	82rongas * Equivalence Ratio
Pbspline(FuelEthanolEqRatio)	12.19	1.000	785.7	-1.1736	Ethanol * Equivalence Ratio
Pbspline(FuelGasoholEqRatio)	13.00	1.000	7.019E-9	-64.2961	Gasohol * Equivalence Ratio

Figure 120.11 *continued*

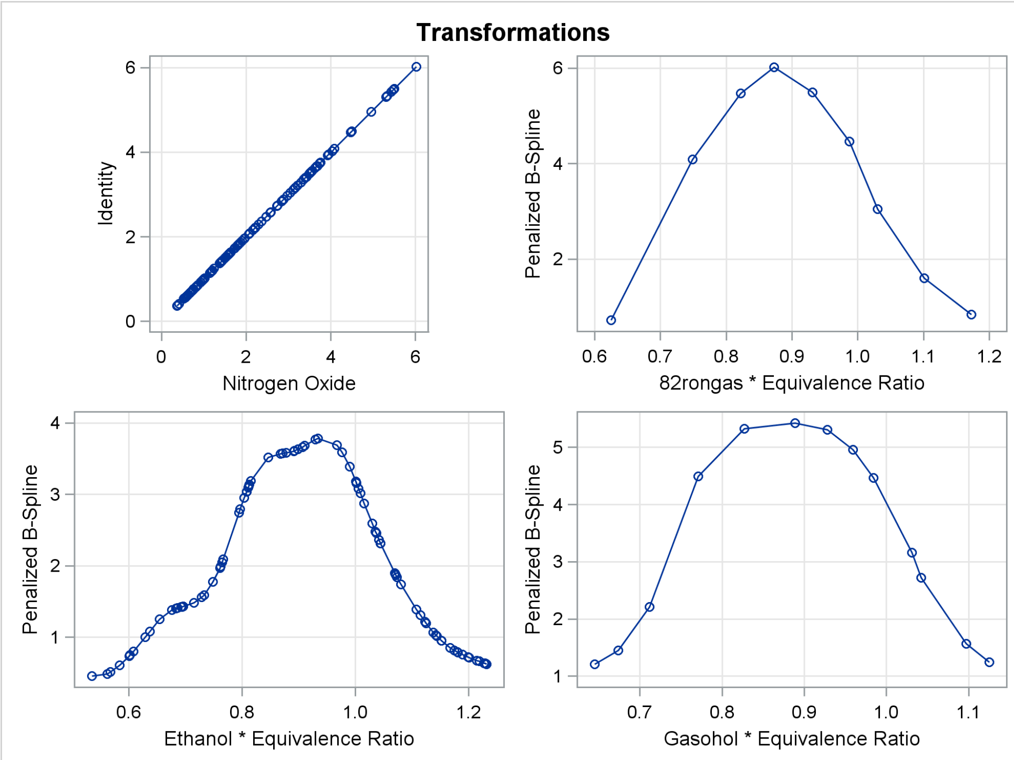
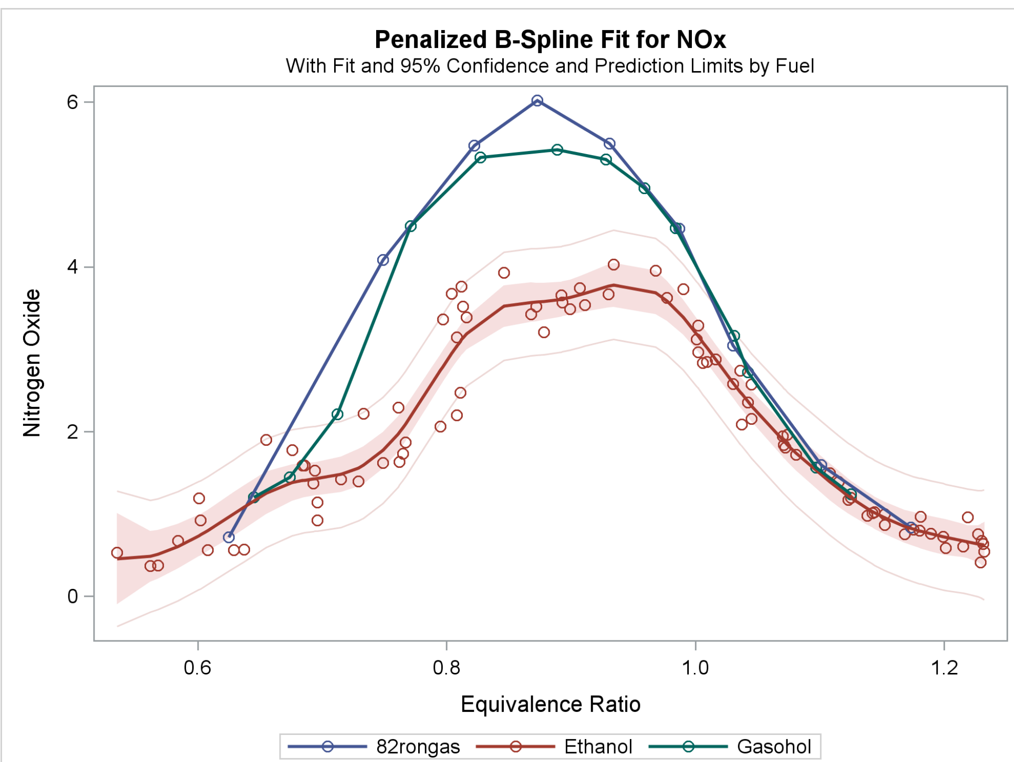


Figure 120.11 *continued*



With penalized B-splines, the degrees of freedom are based on the trace of the transformation hat matrix and are typically not integers. The first panel of plots shows AICC as a function of lambda, the smoothing parameter. The smoothing parameter is automatically chosen, and since the smoothing parameters range from essentially 0 to almost 800, it is clear that some functions are smoother than others. The plots of the criterion (AICC in this example) as a function of lambda use a linear scale for the horizontal axis when the range of lambdas is small, as in the first and third plot, and a log scale when the range is large, as in the second plot. The transformation for equivalence ratio for Ethanol required more smoothing than for the other two fuels. All three have an overall quadratic shape, but for Ethanol, the function more closely follows the smaller variations in the data. You could get similar results with SPLINE by using more knots.

For other examples of curve fitting by using PROC TRANSREG, see the sections "Smoothing Splines" on page 9962, "Linear and Nonlinear Regression Functions" on page 9949, "Simultaneously Fitting Two Regression Functions" on page 9953, and "Using Splines and Knots" on page 9932, as well as Example 120.3. These examples include cases where multiple curves are fit through scatter plots with multiple groups. Special cases include linear models with separate slopes and separate intercepts. Many constraints on the slopes, curves, and intercepts are possible.

Main-Effects ANOVA

This example shows how to use PROC TRANSREG to code and fit a main-effects ANOVA model. PROC TRANSREG has very extensive and versatile options for coding or creating so-called dummy variables. PROC TRANSREG is commonly used to code classification variables before they are used for analysis in other procedures. See the sections "Using the DESIGN Output Option" on page 10027 and "Discrete Choice Experiments: DESIGN, NORESTORE, NOZERO" on page 10031. In this example, the input data set contains the dependent variables y, factors x1 and x2, and 12 observations. PROC TRANSREG can be useful for coding even before running procedures with a CLASS statement because of its detailed options that enable you to control how the coded variable names and labels are constructed. The following statements perform a main-effects ANOVA and display the results in Figure 120.12 and Figure 120.13:

```
title 'Introductory Main-Effects ANOVA Example';

data a;
   input y x1 $ x2 $;
   datalines;
8 a a
7 a a
4 a b
3 a b
5 b a
4 b a
2 b b
1 b b
8 c a
7 c a
5 c b
2 c b
;
```

```
* Fit a main-effects ANOVA model with 1, 0, -1 coding;
proc transreg ss2;
   model identity(y) = class(x1 x2 / effects);
   output coefficients replace;
run;

* Display TRANSREG output data set;
proc print label;
   format intercept -- x2a 5.2;
run;
```

The SS2 *a-option* requests results based on Type II sums of squares. The simple ANOVA model is fit by designating y as an IDENTITY variable, which specifies no transformation. The independent variables are specified with a CLASS expansion, which replaces them with coded variables. There are $(3-1)+(2-1) = 3$ coded variables created by the CLASS specification, since the two CLASS variables have 3 and 2 different values or levels. In this case, the EFFECTS *t-option* is specified. This option requests an *effects coding* (displayed in Figure 120.13), which is also called a deviations from means or 0, 1, –1 coding. The OUTPUT statement requests an output data set with the data and coded variables. The COEFFICIENTS output option, or *o-option*, adds the parameter estimates and marginal means to the data set. The REPLACE *o-option* specifies that the transformed variables should replace the original variables in the output data set. The output data set variable names are the same as the original variable name. In an example like this, there are no nonlinear transformations; the transformed variables are the same as the original variables. The REPLACE *o-option* is used to eliminate unnecessary and redundant transformed variables from the output data set. The results of the PROC TRANSREG step are shown in Figure 120.12.

Figure 120.12 ANOVA Example Output from PROC TRANSREG

Introductory Main-Effects ANOVA Example

The TRANSREG Procedure

Dependent Variable Identity(y)

Class Level Information

Class	Levels	Values
x1	3	a b c
x2	2	a b

Number of Observations Read	12
Number of Observations Used	12

Figure 120.12 *continued*

The TRANSREG Procedure Hypothesis Tests for Identity(y)

Univariate ANOVA Table Based on the Usual Degrees of Freedom					
Source	**DF**	**Sum of Squares**	**Mean Square**	**F Value**	**Pr > F**
Model	3	57.00000	19.00000	19.83	0.0005
Error	8	7.66667	0.95833		
Corrected Total	11	64.66667			

Root MSE	0.97895	**R-Square**	0.8814
Dependent Mean	4.66667	**Adj R-Sq**	0.8370
Coeff Var	20.97739		

Univariate Regression Table Based on the Usual Degrees of Freedom							
Variable	**DF**	**Coefficient**	**Type II Sum of Squares**	**Mean Square**	**F Value**	**Pr > F**	**Label**
Intercept	1	4.6666667	261.333	261.333	272.70	<.0001	Intercept
Class.x1a	1	0.8333333	4.167	4.167	4.35	0.0705	x1 a
Class.x1b	1	-1.6666667	16.667	16.667	17.39	0.0031	x1 b
Class.x2a	1	1.8333333	40.333	40.333	42.09	0.0002	x2 a

Figure 120.12 shows the ANOVA results, fit statistics, and regression tables. The output data set, with the coded design, parameter estimates and means, is shown in Figure 120.13. For more information about PROC TRANSREG for ANOVA and other codings, see the section "ANOVA Codings" on page 9969.

Figure 120.13 Output Data Set from PROC TRANSREG

Introductory Main-Effects ANOVA Example

Obs	_TYPE_	_NAME_	y	Intercept	x1 a	x1 b	x2 a	x1	x2
1	SCORE	ROW1	8	1.00	1.00	0.00	1.00	a	a
2	SCORE	ROW2	7	1.00	1.00	0.00	1.00	a	a
3	SCORE	ROW3	4	1.00	1.00	0.00	-1.00	a	b
4	SCORE	ROW4	3	1.00	1.00	0.00	-1.00	a	b
5	SCORE	ROW5	5	1.00	0.00	1.00	1.00	b	a
6	SCORE	ROW6	4	1.00	0.00	1.00	1.00	b	a
7	SCORE	ROW7	2	1.00	0.00	1.00	-1.00	b	b
8	SCORE	ROW8	1	1.00	0.00	1.00	-1.00	b	b
9	SCORE	ROW9	8	1.00	-1.00	-1.00	1.00	c	a
10	SCORE	ROW10	7	1.00	-1.00	-1.00	1.00	c	a
11	SCORE	ROW11	5	1.00	-1.00	-1.00	-1.00	c	b
12	SCORE	ROW12	2	1.00	-1.00	-1.00	-1.00	c	b
13	M COEFFI	y	.	4.67	0.83	-1.67	1.83		
14	MEAN	y	.	.	5.50	3.00	6.50		

The output data set has three kinds of observations, identified by values of _TYPE_ as follows:

- When _TYPE_='SCORE', the observation contains the following information about the dependent and independent variables:
  - y is the original dependent variable.
  - x1 and x2 are the independent classification variables, and the Intercept through x2 a columns contain the main-effects design matrix that PROC TRANSREG creates. The variable names are Intercept, x1a, x1b, and x2a. Their labels are shown in the listing.
- When _TYPE_='M COEFFI', the observation contains coefficients of the final linear model (parameter estimates).
- When _TYPE_='MEAN', the observation contains the marginal means.

The observations with _TYPE_='SCORE' form the score or data partition of the output data set, and the observations with _TYPE_='M COEFFI' and _TYPE_='MEAN' form the output statistics partition of the output data set.

Syntax: TRANSREG Procedure

The following statements are available in the TRANSREG procedure:

PROC TRANSREG < **DATA=***SAS-data-set*>
< **PLOTS=(***plot-requests***)** >
< **OUTTEST=***SAS-data-set*> < *a-options*> < *o-options*> **;**
MODEL < *transform***(***dependents* < / *t-options*>**)** >
< *transform***(***dependents* < / *t-options*>**)**. . . **=** >
*transform***(***independents* < / *t-options*>**)**
< *transform***(***independents* < / *t-options*>**)**. . . > < / *a-options*> **;**
OUTPUT < **OUT=***SAS-data-set*> < *o-options*> **;**
ID *variables* **;**
FREQ *variable* **;**
WEIGHT *variable* **;**
BY *variables* **;**

To use PROC TRANSREG, you need both the PROC TRANSREG and MODEL statements. To produce an OUT= output data set, the OUTPUT statement is required. PROC TRANSREG enables you to specify the same options in more than one statement. All of the MODEL statement *a-options* (algorithm options) and all of the OUTPUT statement *o-options* (output options) can also be specified in the PROC TRANSREG statement. You can abbreviate all *a-options*, *o-options*, and *t-options* (transformation options) to their first three letters. This is a special feature of PROC TRANSREG and is not generally true of other SAS/STAT procedures. See Table 120.1 for a list of options available in the PROC TRANSREG statement.

The PROC TRANSREG statement starts the TRANSREG procedure. Optionally, this statement identifies an input and an OUTTEST= data set, specifies the algorithm and other computational details, requests displayed output, and controls the contents of the OUT= data set (which is created with the OUTPUT statement). The

DATA= and OUTTEST= options can appear only in the PROC TRANSREG statement. All *a-options* and *o-options* are described in the sections on either the MODEL or OUTPUT statement, in which these options can also be specified.

The rest of this section provides detailed syntax information for each of the preceding statements, beginning with the PROC TRANSREG statement. The remaining statements are described in alphabetical order.

PROC TRANSREG Statement

PROC TRANSREG < **DATA=***SAS-data-set*>
< **PLOTS=(***plot-requests***)** >
< **OUTTEST=***SAS-data-set*> < *a-options*> < *o-options*> **;**

The PROC TRANSREG statement invokes the TRANSREG procedure. Optionally, this statement identifies an input and an OUTTEST= data set, specifies the algorithm and other computational details, requests displayed output, and controls the contents of the OUT= data set (which is created with the OUTPUT statement). The DATA=, OUTTEST=, and PLOTS= options can appear only in the PROC TRANSREG statement. Table 120.1 summarizes the options available in the PROC TRANSREG statement. The *a-options* are also available in the MODEL statement, and the *o-options* are also available in the OUTPUT statement.

Table 120.1 Options Available in the PROC TRANSREG Statement

Option	Description
Data Set Options (PROC Statement)	
DATA=	Specifies input SAS data set
OUTTEST=	Specifies output test statistics data set
ODS Graphics (PROC Statement)	
PLOTS=	Specifies ODS Graphics selection
Input Control (PROC or MODEL)	
REITERATE	Restarts the iterations
TYPE=	Specifies input observation type
Method and Iterations (PROC or MODEL)	
CCONVERGE=	Specifies minimum criterion change
CONVERGE=	Specifies minimum data change
MAXITER=	Specifies maximum number of iterations
METHOD=	Specifies iterative algorithm
NCAN=	Specifies number of canonical variables
NSR	Specifies no restrictions on smoothing models
SINGULAR=	Specifies singularity criterion
SOLVE	Attempts direct solution instead of iteration
Missing Data Handling (PROC or MODEL)	
INDIVIDUAL	Fits each model individually (METHOD=MORALS)
MONOTONE=	Includes monotone special missing values
NOMISS	Excludes observations with missing values

Table 120.1 *continued*

Option	Description
UNTIE=	Unties special missing values
Intercept and CLASS Variables (PROC or MODEL)	
CPREFIX=	Specifies CLASS coded variable name prefix
LPREFIX=	Specifies CLASS coded variable label prefix
NOINT	Specifies no intercept or centering
ORDER=	Specifies order of CLASS variable levels
REFERENCE=	Controls output of reference levels
SEPARATORS=	Controls CLASS coded variable label separators
Control Displayed Output (PROC or MODEL)	
ALPHA=	Specifies confidence limits alpha
CL	Displays parameter estimate confidence limits
DETAIL	Displays model specification details
HISTORY	Displays iteration histories
NOPRINT	Suppresses displayed output
PBOXCOXTABLE	Prints the Box-Cox log likelihood table
RSQUARE	Displays the R square
SHORT	Suppresses the iteration histories
SS2	Displays regression results
TEST	Displays ANOVA table
TSUFFIX=	Shortens transformed variable labels
UTILITIES	Displays conjoint part-worth utilities
Standardization (PROC or MODEL)	
ADDITIVE	Fits additive model
NOZEROCONSTANT	Does not zero constant variables
TSTANDARD=	Specifies transformation standardization
Predicted Values, Residuals, Scores (PROC or OUTPUT)	
CANONICAL	Outputs canonical scores
CLI	Outputs individual confidence limits
CLM	Outputs mean confidence limits
DESIGN=	Specifies design matrix coding
DREPLACE	Replaces dependent variables
IREPLACE	Replaces independent variables
LEVERAGE	Outputs leverage
NORESTOREMISSING	Does not restore missing values
NOSCORES	Suppresses output of scores
PREDICTED	Outputs predicted values
REDUNDANCY=	Outputs redundancy variables
REPLACE	Replaces all variables
RESIDUALS	Outputs residuals
Output Data Set Coefficients (PROC or OUTPUT)	
COEFFICIENTS	Outputs coefficients

Table 120.1 *continued*

Option	Description
COORDINATES=	Outputs ideal point coordinates
MEANS	Outputs marginal means
MREDUNDANCY	Outputs redundancy analysis coefficients
Output Data Set Variable Name Prefixes (PROC or OUTPUT)	
ADPREFIX=	Specifies dependent variable approximations
AIPREFIX=	Specifies independent variable approximations
CDPREFIX=	Specifies canonical dependent variables
CILPREFIX=	Specifies conservative individual lower CL
CIPREFIX=	Specifies canonical independent variables
CIUPREFIX=	Specifies conservative-individual-upper CL
CMLPREFIX=	Specifies conservative-mean-lower CL
CMUPREFIX=	Specifies conservative-mean-upper CL
DEPENDENT=	Specifies METHOD=MORALS untransformed dependent
LILPREFIX=	Specifies liberal-individual-lower CL
LIUPREFIX=	Specifies liberal-individual-upper CL
LMLPREFIX=	Specifies liberal-mean-lower CL
LMUPREFIX=	Specifies liberal-mean-upper CL
PPREFIX=	Specifies predicted values
RDPREFIX=	Specifies residuals
RPREFIX=	Specifies redundancy variables
TDPREFIX=	Specifies transformed dependents
TIPREFIX=	Specifies transformed independents
Macros Variables (PROC or OUTPUT)	
MACRO	Creates macro variables
Other Options (PROC or OUTPUT)	
APPROXIMATIONS	Outputs dependent and independent approximations
CCC	Outputs canonical correlation coefficients
CEC	Outputs canonical elliptical point coordinates
CPC	Outputs canonical point coordinates
CQC	Outputs canonical quadratic point coordinates
DAPPROXIMATIONS	Outputs approximations to transformed dependents
IAPPROXIMATIONS	Outputs approximations to transformed independents
MEC	Outputs elliptical point coordinates
MPC	Outputs point coordinates
MQC	Outputs quadratic point coordinates
MRC	Outputs multiple regression coefficients

DATA=*SAS-data-set*

specifies the SAS data set to be analyzed. If you do not specify the DATA= option, PROC TRANSREG uses the most recently created SAS data set. The data set must be an ordinary SAS data set; it cannot be a special TYPE= data set.

OUTTEST=*SAS-data-set*

specifies an output data set to contain hypothesis tests results. When you specify the OUTTEST= option, the data set contains ANOVA results. When you specify the SS2 *a-option*, regression tables are also output. When you specify the UTILITIES *o-option*, conjoint analysis part-worth utilities are also output. For more information about the OUTTEST= data set, see the section "OUTTEST= Output Data Set" on page 10012.

PLOTS **<(***global-plot-options***)>** **<=** *plot-request* <**(***options***)**>**>**

PLOTS **<(***global-plot-options***)>** **<= (***plot-request* <**(***options***)**> **<...** *plot-request* <**(***options***)**>**>) >**

controls the plots produced through ODS Graphics. When you specify only one plot request, you can omit the parentheses around the plot request. Here are some examples:

```
plots=none
plots=(residuals transformation)
plots(unpack)=boxcox
plots(unpack)=(transformation boxcox(p=0))
plots=(residuals(unpack) transformation(dep unp) boxcox(t rmse))
```

ODS Graphics must be enabled before plots can be requested. For example:

```
ods graphics on;

proc transreg plots=all;
   model identity(y) = pbspline(x);
run;

ods graphics off;
```

For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" on page 615 in Chapter 21, "Statistical Graphics Using ODS."

If ODS Graphics is enabled, but you do not specify the PLOTS= option, then PROC TRANSREG produces a default set of plots. The fit, scatter, residual, and observed-by-predicted plots are available with METHOD=MORALS and also with METHOD=UNIVARIATE when there is only one dependent variable. When no method is specified and there is more than one dependent variable, and when regression plots are requested, the default method is set to METHOD=MORALS. When there is more than one dependent variable, when METHOD= is not specified, or when METHOD=MORALS is specified and PLOTS=ALL is specified, the plots that are produced might be different from those you would see with METHOD=UNIVARIATE and PLOTS=ALL. Certain plots appear by default when ODS Graphics is enabled and certain combinations of options are specified. The Box-Cox $F = t^2$ and log-likelihood plots appear when a BOXCOX dependent variable transformation is specified. The regression fit plot appears for models with a single dependent variable that is not transformed (for example, IDENTITY(y)), a single quantitative independent variable that might or might not be transformed, and at most one CLASS independent variable. Preference mapping plots appear when the COORDINATES *o-option* is used.

The global plot options include the following:

INTERPOLATE

INT

uses observations that are excluded from the analysis for interpolation in the fit and transformation plots. By default, observations with zero weight are excluded from all plots. These include observations with a zero, negative, or missing weight or frequency and observations excluded due to missing and invalid values. You can specify PLOTS(INTERPOLATE)=(*plot-requests*) to include some of these observations in the plots. You might want to use this option, for example, with sparse data sets to show smoother functions over the range of the data (see the section "The PLOTS(INTERPOLATE) Option" on page 10036). Observations with missing values in CLASS variables are excluded from the plots even when PLOTS(INTERPOLATE) is specified.

ONLY

ONL

suppresses the default plots. Only plots specifically requested are displayed.

UNPACKPANEL

UNPACK

UNP

suppresses paneling. By default, multiple plots can appear in some output panels. Specify UNPACKPANEL to get each plot in a separate panel. You can specify PLOTS(UNPACKPANEL) to unpack the default plots. You can also specify UNPACKPANEL as a suboption with TRANSFORMATION, RESIDUALS, PBSPLINE, and BOXCOX.

The plot requests include the following:

ALL

produces all appropriate plots. You can specify other options with ALL; for example, to request all plots and unpack only the residuals, specify PLOTS=(ALL RES(UNP)).

BOXCOX < (*options*) >

BOX < (*options*) >

requests a display of the results of the Box-Cox transformation. These results are displayed by default when there is a Box-Cox transformation. The BOXCOX plot request has the following options:

P=*n*

adds t or $F = t^2$ curves to the legend for the functions where $p(t) < n$, where t is the t statistic corresponding to the optimal lambda. You can specify P=0 to suppress the legend and P=1 to see all curves in the legend. The default value comes from the BOXCOX(variable / ALPHA=p) specification, which by default is 0.05.

RMSE

RMS

plots the root mean square error as a function of lambda.

T

plots t statistics rather than $F = t^2$ statistics.

UNPACKPANEL

UNPACK

UNP

plots the t or $F = t^2$ and log-likelihood plots in separate panels.

FIT < (*options*) >

requests a regression fit plot. This plot is produced by default whenever it is appropriate. It is produced when the dependent variable is specified with the IDENTITY *transform*, and when there is one quantitative independent variable (for example, IDENTITY for linear fit, SPLINE or one of the other transformations for a nonlinear fit, or PSPLINE) and at most one CLASS variable. When there is a CLASS variable, separate fits are produced within levels based on your model. You would specify the FIT plot request only to specify a FIT option or with the ONLY global plot option. The FIT plot request has the following options:

FORMULA

FOR

displays the fit function as an equation in regression fit plots. This option is valid when a fit plot is produced and either an IDENTITY *transform* or a PSPLINE *expansion* with degree less than ten and no knots is specified for a single independent variable. When this option is specified, you can output the formula to a data set by using the `ods output formula=`*SAS-data-set* statement. This is the formula, complete with Unicode specifications for polynomials, that is used in the fit plot template to make the formula.

NOCLM

suppresses the confidence limits in regression fit plots.

NOCLI

suppresses the individual prediction limits in regression fit plots.

NOOBS

suppresses the observations showing only the fit function and optionally the confidence and prediction limits.

NONE

suppresses all plots.

OBSERVEDBYPREDICTED

OBP

OBS

plots the transformed dependent variable as a function of the regression predicted values.

PBSPLINE < (UNPACKPANEL) >

PBS < (UNPACK) >

requests the penalized B-spline criterion plots. You would specify the PBSPLINE plot request only to specify a PBSPLINE option or with the ONLY global plot option. The PBSPLINE plot request has the following option:

UNPACKPANEL

UNPACK

UNP

plots each criterion plot in a separate panel.

PREFMAP

PRE

plots ideal point or vector preference mapping results when either two IDENTITY or two POINT independent variables are specified along with the COORDINATES option.

RESIDUALS < **(***options***)** >

RES < **(***options***)** >

plots the residuals as a function of each of the transformed independent variables, except coded CLASS variables. The RESIDUALS plot request has the following options:

CLASS

CLA

plots the residuals as a function of each of the transformed independent variables, including coded CLASS variables. Note that the ALL plot request, which you use to request all plots, specifies the RESIDUALS plot request without the CLASS option.

UNPACKPANEL

UNPACK

UNP

plots the residuals in separate plots, not several per panel.

SMOOTH

SMO

adds a LOESS smooth function to the residuals plots.

SCATTER

SCA

plots the scatter plot of observed data, before the transformations, for models with a single quantitative dependent variable, a single quantitative independent variable, and at most one CLASS independent variable.

TRANSFORMATION < **(***options***)** >

TRA < **(***options***)** >

plots the variable transformations. The TRANSFORMATION plot request has the following options:

DEPENDENTS

DEP

plots only the dependent variable transformations.

INDEPENDENTS

IND

plots only the independent variable transformations.

UNPACKPANEL

UNPACK

UNP

plots the transformations in separate plots, not several per panel.

BY Statement

BY *variables* **;**

You can specify a BY statement with PROC TRANSREG to obtain separate analyses of observations in groups that are defined by the BY variables. When a BY statement appears, the procedure expects the input data set to be sorted in order of the BY variables. If you specify more than one BY statement, only the last one specified is used.

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

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

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

FREQ Statement

FREQ *variable* **;**

If one variable in the input data set represents the frequency of occurrence for other values in the observation, specify the variable's name in a FREQ statement. PROC TRANSREG then treats the data set as if each observation appeared n times, where n is the value of the FREQ variable for the observation. Noninteger values of the FREQ variable are truncated to the largest integer less than the FREQ value. The observation is used in the analysis only if the value of the FREQ statement variable is greater than or equal to 1.

ID Statement

ID *variables* **;**

The ID statement includes additional character or numeric variables in the OUT= data set. The variables must be contained in the input data set. The first variable is used to label points in PREFMAP plots. These variables are also used in some plots as tip variables.

MODEL Statement

MODEL < *transform*(*dependents* < / *t-options* >) >
< *transform*(*dependents* < / *t-options* >)... **=** >
transform(*independents* < / *t-options* >)
< *transform*(*independents* < / *t-options* >)... > < / *a-options* > **;**

The MODEL statement specifies the dependent and independent variables (*dependents* and *independents*, respectively) and specifies the transformation (*transform*) to apply to each variable. Only one MODEL statement can appear in PROC TRANSREG. The *t-options* are transformation options, and the *a-options* are algorithm options. The *t-options* provide details for the transformation; these depend on the *transform* chosen. The *t-options* are listed after a slash in the parentheses that enclose the variable list (either *dependents* or *independents*). The *a-options* control the algorithm used, details of iteration, details of how the intercept and coded variables are generated, and displayed output details. The *a-options* are listed after the entire model specification (the *dependents*, *independents*, transformations, and *t-options*) and after a slash. You can also specify the algorithm options in the PROC TRANSREG statement. When you specify the DESIGN *o-option*, *dependents* and an equal sign are not required. The operators *, |, and @ from the GLM procedure are available for interactions with the CLASS expansion and the IDENTITY transformation. They are used as follows:

```
   Class(a * b ...
         c | d ...
         e | f ... @ n)
Identity(a * b ...
         c | d ...
         e | f ... @ n)
```

In addition, transformations and spline expansions can be crossed with classification variables as follows:

```
transform(var) * class(group)
transform(var) | class(group)
```

See the section "Types of Effects" on page 3773 in Chapter 48, "The GLM Procedure," for a description of the @, *, and | operators and see the section "Model Statement Usage" on page 9920 for information about how to use these operators in PROC TRANSREG. Note that nesting is not implemented in PROC TRANSREG.

The next three sections discuss the transformations available (*transforms*) (see the section "Families of Transformations" on page 9884), the transformation options (*t-options*) (see the section "Transformation Options (t-options)" on page 9890), and the algorithm options (*a-options*) (see the section "Algorithm Options (a-options)" on page 9901).

Families of Transformations

In the MODEL statement, *transform* specifies a transformation in one of the following five families:

Variable expansions	preprocess the specified variables, replacing them with more variables.
Nonoptimal transformations	preprocess the specified variables, replacing each one with a single new nonoptimal, nonlinear transformation.
Nonlinear fit transformations	preprocess the specified variable, replacing it with a smooth transformation, fitting one or more nonlinear functions through a scatter plot.
Optimal transformations	replace the specified variables with new, iteratively derived optimal transformation variables that fit the specified model better than the original variable (except for contrived cases where the transformation fits the model exactly as well as the original variable).
Other transformations	are the IDENTITY and SSPLINE transformations. These do not fit into the preceding categories.

The transformations and expansions listed in Table 120.2 are available in the MODEL statement.

Table 120.2 Transformation Families

Transformation	Description
Variable Expansions	
BSPLINE	B-spline basis
CLASS	set of coded variables
EPOINT	elliptical response surface
POINT	circular response surface & PREFMAP
PSPLINE	piecewise polynomial basis
QPOINT	quadratic response surface
Nonoptimal Transformations	
ARSIN	inverse trigonometric sine
EXP	exponential
LOG	logarithm
LOGIT	logit
POWER	raises variables to specified power
RANK	transforms to ranks
Nonlinear Fit Transformations	
BOXCOX	Box-Cox
PBSPLINE	penalized B-splines
SMOOTH	noniterative smoothing spline
Optimal Transformations	
LINEAR	linear
MONOTONE	monotonic, ties preserved
MSPLINE	monotonic B-spline
OPSCORE	optimal scoring
SPLINE	B-spline

Table 120.2 *continued*

Transformation	Description
UNTIE	monotonic, ties not preserved
Other Transformations	
IDENTITY	identity, no transformation
SSPLINE	iterative smoothing spline

You can use any transformation with either dependent or independent variables (except the SMOOTH and PBSPLINE transformations, which can be used only with independent variables, and BOXCOX, which can be used only with dependent variables). However, the variable expansions are usually more appropriate for independent variables.

The *transform* is followed by a variable (or list of variables) enclosed in parentheses. Here is an example:

```
model log(y) = class(x);
```

This example finds a LOG transformation of y and performs a CLASS expansion of x. Optionally, depending on the *transform*, the parentheses can also contain *t-options*, which follow the variables and a slash. Here is an example:

```
model identity(y) = spline(x1 x2 / nknots=3);
```

The preceding statement finds SPLINE transformations of x1 and x2. The NKNOTS= *t-option* used with the SPLINE transformation specifies three knots. The **identity(y)** transformation specifies that y is not to be transformed.

The rest of this section provides syntax details for members of the five families of transformations listed at the beginning of this section. The *t-options* are discussed in the section "Transformation Options (t-options)" on page 9890.

Variable Expansions

PROC TRANSREG performs variable expansions before iteration begins. Variable expansions expand the original variables into a typically larger set of new variables. The original variables are those that are listed in parentheses after *transform*, and they are sometimes referred to by the name of the *transform*. For example, in CLASS(x1 x2), x1 and x2 are sometimes referred to as CLASS expansion variables or simply CLASS variables, and the expanded variables are referred to as coded or sometimes "dummy" variables. Similarly, in POINT(Dim1 Dim2), Dim1 and Dim2 are sometimes referred to as POINT variables.

The resulting variables are not transformed by the iterative algorithms after the initial preprocessing. Observations with missing values for these types of variables are excluded from the analysis.

The POINT, EPOINT, and QPOINT variable expansions are used in preference mapping analyses (also called PREFMAP, external unfolding, ideal point regression) (Carroll 1972) and for response surface regressions. These three expansions create circular, elliptical, and quadratic response or preference surfaces (see the section "Point Models" on page 9993 and Example 120.6). The CLASS variable expansion is used for main-effects ANOVA.

The following list provides syntax and details for the variable expansion *transforms*.

BSPLINE

BSP

expands each variable to a B-spline basis. You can specify the DEGREE=, KNOTS=, NKNOTS=, and EVENLY= *t-options* with the BSPLINE expansion. When DEGREE=n (3 by default) with k knots (0 by default), $n + k + 1$ variables are created. In addition, the original variable appears in the OUT= data set before the ID variables. For example, **bspline(x)** expands x into x_0 x_1 x_2 x_3 and outputs x as well. The x_: variables contain the B-spline basis vectors (which are the same basis vectors that the SPLINE and MSPLINE transformations use internally). The columns of the BSPLINE expansion sum to a column of ones, so an implicit intercept model is fit when the BSPLINE expansion is specified. If you specify the BSPLINE expansion for more than one variable, the model is less than full rank. Variables specified in a BSPLINE expansion must be numeric, and they are typically continuous. See the sections "SPLINE and MSPLINE Transformations" on page 9999 and "SPLINE, BSPLINE, and PSPLINE Comparisons" on page 10001 for more information about B-splines.

CLASS

CLA

expands the variables to a set of coded or "dummy" variables. PROC TRANSREG uses the values of the formatted variables to determine class membership. The specification **class(x1 x2)** fits a simple main-effects model, **class(x1 | x2)** fits a main-effects and interactions model, and **class(x1|x2|x3|x4@2 x1*x2*x3)** fits a model with all main effects, all two-way interactions, and one three-way interaction. Variables specified with the CLASS expansion can be either character or numeric; numeric variables should be discrete. See the section "ANOVA Codings" on page 9969 for more information about CLASS variables. See the section "Model Statement Usage" on page 9920 for information about how to use the operators @, *, and | in PROC TRANSREG.

EPOINT

EPO

expands the variables for an elliptical response surface regression or for an elliptical ideal point regression. Specify the COORDINATES *o-option* to output PREFMAP ideal elliptical point model coordinates to the OUT= data set. Each axis of the ellipse (or ellipsoid) is oriented in the same direction as one of the variables. The EPOINT expansion creates a new variable for each original variable. The value of each new variable is the square of each observed value for the corresponding original variable. The regression analysis then uses both sets of variables (original and squared). Variables specified with the EPOINT expansion must be numeric, and they are typically continuous. See the section "Point Models" on page 9993 and Example 120.6 for more information about point models.

POINT

POI

expands the variables for a circular response surface regression or for a circular ideal point regression. Specify the COORDINATES *o-option* to output PREFMAP ideal point model coordinates to the OUT= data set. The POINT expansion creates a new variable having a value for each observation that is the sum of squares of all the POINT variables. This new variable is added to the set of variables and is used in the regression analysis. For more information about ideal point regression, see Carroll (1972). Variables specified with the POINT expansion must be numeric, and they are typically continuous. See the section "Point Models" on page 9993 and Example 120.6 for more information about point models.

PSPLINE

PSP

expands each variable to a piecewise polynomial basis. You can specify the DEGREE=, KNOTS=, NKNOTS=, and EVENLY *t-options* with PSPLINE. When DEGREE=n (3 by default) with k knots (0 by default), $n + k$ variables are created. In addition, the original variable appears in the OUT= data set before the ID variables. For example, `pspline(x / nknots=1)` expands x into x_1 x_2 x_3 x_4 and outputs x as well. Unlike BSPLINE, an intercept is not implicit in the columns of PSPLINE. Variables specified with the PSPLINE expansion must be numeric, and they are typically continuous. See the sections "SPLINE, BSPLINE, and PSPLINE Comparisons" on page 10001 and "Using Splines and Knots" on page 9932 for more information about splines. Also see Smith (1979) for a good introduction to piecewise polynomial splines.

QPOINT

QPO

expands the variables for a quadratic response surface regression or for a quadratic ideal point regression. Specify the COORDINATES *o-option* to output PREFMAP quadratic ideal point model coordinates to the OUT= data set. For m QPOINT variables, $m(m + 1)/2$ new variables are created containing the squares and crossproducts of the original variables. The regression analysis uses both sets (original and crossed). Variables specified with the QPOINT expansion must be numeric, and they are typically continuous. See the section "Point Models" on page 9993 and Example 120.6 for more information about point models.

Nonoptimal Transformations

The nonoptimal transformations, like the variable expansions, are computed before the iterative algorithm begins. Nonoptimal transformations create a single new transformed variable that replaces the original variable. The new variable is not transformed by the subsequent iterative algorithms (except for a possible linear transformation with missing value estimation). The following list provides syntax and details for nonoptimal variable transformations.

ARSIN

ARS

finds an inverse trigonometric sine transformation. Variables specified in the ARSIN *transform* must be numeric and in the interval ($-1.0 \le x \le 1.0$), and they are typically continuous.

EXP

exponentiates variables (x is transformed to a^x). To specify the value of a, use the PARAMETER= *t-option*. By default, a is the mathematical constant $e = 2.718\ldots$. Variables specified with the EXP *transform* must be numeric, and they are typically continuous.

LOG

transforms variables to logarithms (x is transformed to $\log_a(x)$). To specify the base of the logarithm, use the PARAMETER= *t-option*. The default is a natural logarithm with base $e = 2.718\ldots$. Variables specified with the LOG *transform* must be numeric and positive, and they are typically continuous.

LOGIT

finds a logit transformation on the variables. The logit of x is $\log(x/(1 - x))$. Unlike other transformations, LOGIT does not have a three-letter abbreviation. Variables specified with the LOGIT *transform* must be numeric and in the interval ($0.0 < x < 1.0$), and they are typically continuous.

POWER

POW

raises variables to a specified power (x is transformed to x^a). You must specify the power parameter a by specifying the PARAMETER= *t-option* following the variables. Here is an example:

```
power(variable / parameter=number)
```

You can use POWER for squaring variables (PARAMETER=2), reciprocal transformations (PARAMETER=–1), square roots (PARAMETER=0.5), and so on. Variables specified with the POWER *transform* must be numeric, and they are typically continuous.

RANK

RAN

transforms variables to ranks. Ranks are averaged within ties. The smallest input value is assigned the smallest rank. Variables specified in the RANK *transform* must be numeric.

Nonlinear Fit Transformations

Nonlinear fit transformations, like nonoptimal transformations, are computed before the iterative algorithm begins. Nonlinear fit transformations create a single new transformed variable that replaces the original variable and provides one or more smooth functions through a scatter plot. The new variable is not transformed by the subsequent iterative algorithms. The nonlinear fit transformations, unlike the nonoptimal transformations, use information in the other variables in the model to find the transformations. The nonlinear fit transformations, unlike the optimal transformations, do not minimize a squared-error criterion. The following list provides syntax and details for nonoptimal variable transformations.

BOXCOX

BOX

finds a Box-Cox (1964) transformation of the specified variables. The BOXCOX transformation can be used only with dependent variables. The ALPHA=, CLL=, CONVENIENT, GEOMETRICMEAN, LAMBDA=, and PARAMETER= *t-options* can be used with the BOXCOX transformation. Variables specified in the BOXCOX *transform* must be numeric, and they are typically continuous. See the section "Box-Cox Transformations" on page 9923 and Example 120.2 for more information about Box-Cox transformations.

PBSPLINE

PBS

is a noniterative penalized B-spline transformation (Eilers and Marx 1996). The PBSPLINE transformation can be used only with independent variables. By default with PBSPLINE, a cubic spline is fit with 100 evenly spaced knots, three evenly spaced exterior knots, and a difference matrix of order three (DEGREE=3 NKNOTS=100 EVENLY=3 PARAMETER=3). Variables specified in the PBSPLINE *transform* must be numeric, and they are typically continuous. See the section "Penalized B-Splines" on page 9959 and Example 120.3 for more information about penalized B-splines.

SMOOTH

SMO

is a noniterative smoothing spline transformation (Reinsch 1967). You can specify the smoothing parameter with either the SM= or the PARAMETER= *t-option*. The default smoothing parameter is SM=0. The SMOOTH transformation can be used only with independent variables. Variables specified with the SMOOTH *transform* must be numeric, and they are typically continuous. See the sections "Smoothing Splines" on page 9962 and "Smoothing Splines Changes and Enhancements" on page 9966 for more information about smoothing splines.

Optimal Transformations

Optimal transformations are iteratively derived. Missing values for these types of variables can be optimally estimated (see the section "Missing Values" on page 9987). The following list provides syntax and details for optimal transformations.

LINEAR

LIN

finds an optimal linear transformation of each variable. For variables with no missing values, the transformed variable is the same as the original variable. For variables with missing values, the transformed nonmissing values have a different scale and origin than the original values. Variables specified in the LINEAR *transform* must be numeric. See the section "OPSCORE, MONOTONE, UNTIE, and LINEAR Transformations" on page 9997 for more information about optimal scaling.

MONOTONE

MON

finds a monotonic transformation of each variable, with the restriction that ties are preserved. The Kruskal (1964) secondary least squares monotonic transformation is used. This transformation weakly preserves order and category membership (ties). Variables specified with the MONOTONE *transform* must be numeric, and they are typically discrete. See the section "OPSCORE, MONOTONE, UNTIE, and LINEAR Transformations" on page 9997 for more information about optimal scaling.

MSPLINE

MSP

finds a monotonically increasing B-spline transformation with monotonic coefficients (De Boor 1978; De Leeuw 1986) of each variable. You can specify the DEGREE=, KNOTS=, NKNOTS=, and EVENLY= *t-options* with MSPLINE. By default, PROC TRANSREG fits a quadratic spline with no knots. Variables specified with the MSPLINE *transform* must be numeric, and they are typically continuous. See the section "SPLINE and MSPLINE Transformations" on page 9999 for more information about monotone splines.

OPSCORE

OPS

finds an optimal scoring of each variable. The OPSCORE transformation assigns scores to each class (level) of the variable. The Fisher (1938) optimal scoring method is used. Variables specified with the OPSCORE *transform* can be either character or numeric; numeric variables should be discrete. See the sections "Character OPSCORE Variables" on page 9992 and "OPSCORE, MONOTONE, UNTIE, and LINEAR Transformations" on page 9997 for more information about optimal scaling.

SPLINE

SPL

finds a B-spline transformation (De Boor 1978) of each variable. By default, PROC TRANSREG fits a cubic spline with no knots. You can specify the DEGREE=, KNOTS=, NKNOTS=, and EVENLY= *t-options* with SPLINE. Variables specified with the SPLINE *transform* must be numeric, and they are typically continuous. See the sections "SPLINE and MSPLINE Transformations" on page 9999, "Specifying the Number of Knots" on page 10000, and "SPLINE, BSPLINE, and PSPLINE Comparisons" on page 10001, and "Using Splines and Knots" on page 9932 for more information about splines.

UNTIE

UNT

finds a monotonic transformation of each variable without the restriction that ties are preserved. PROC TRANSREG uses the Kruskal (1964) primary least squares monotonic transformation method. This transformation weakly preserves order but not category membership (it might untie some previously tied values). Variables specified with the UNTIE *transform* must be numeric, and they are typically discrete. See the section "OPSCORE, MONOTONE, UNTIE, and LINEAR Transformations" on page 9997 for more information about optimal scaling.

Other Transformations

IDENTITY

IDE

specifies variables that are not changed by the iterations. Typically, the IDENTITY transformation is used with a simple variable list, such as **`identity(x1-x5)`**. However, you can also specify interaction terms. For example, **`identity(x1 | x2)`** creates x1, x2, and the product x1*x2; and **`identity(x1 | x2 | x3)`** creates x1, x2, x1*x2, x3, x1*x3, x2*x3, and x1*x2*x3. See the section "Model Statement Usage" on page 9920 for information about how to use the operators @, *, and | in PROC TRANSREG. Variables specified in the IDENTITY *transform* must be numeric.

The IDENTITY transformation is used for variables when no transformation and no missing data estimation are desired. However, the REFLECT *t-option*, the ADDITIVE *a-option*, and the TSTANDARD=Z, and TSTANDARD=CENTER options can linearly transform all variables, including IDENTITY variables, after the iterations. Observations with missing values in IDENTITY variables are excluded from the analysis, and no optimal scores are computed for missing values in IDENTITY variables.

SSPLINE

SSP

finds an iterative smoothing spline transformation of each variable. The SSPLINE transformation does not generally minimize squared error. You can specify the smoothing parameter with either the SM= *t-option* or the PARAMETER= *t-option*. The default smoothing parameter is SM=0. Variables specified with the SSPLINE *transform* must be numeric, and they are typically continuous.

Transformation Options (t-options)

If you use a nonoptimal, nonlinear fit, optimal, or other transformation, you can use *t-options*, which specify additional details of the transformation. The *t-options* are specified within the parentheses that enclose variables and are listed after a slash. You can use *t-options* with both the dependent and the independent variables. Here is an example of using just one *t-option*:

```
proc transreg;
   model identity(y)=spline(x / nknots=3);
   output;
run;
```

The preceding statements find an optimal variable transformation (SPLINE) of the independent variable, and they use a *t-option* to specify the number of knots (NKNOTS=). The following is a more complex example:

```
proc transreg;
   model mspline(y / nknots=3)=class(x1 x2 / effects);
   output;
run;
```

These statements find a monotone spline transformation (MSPLINE with three knots) of the dependent variable and perform a CLASS expansion with effects coding of the independents.

Table 120.3 summarizes the *t-options* available in the MODEL statement.

Table 120.3 Transformation Options

Option	Description
Nonoptimal Transformation	
ORIGINAL	Uses original mean and variance
Parameter Specification	
PARAMETER=	Specifies miscellaneous parameters
SM=	Specifies smoothing parameter
Penalized B-Spline	
AIC	Uses Akaike's information criterion
AICC	Uses corrected AIC
CV	Uses cross validation criterion
GCV	Uses generalized cross validation criterion
LAMBDA=	Specifies smoothing parameter list or range
RANGE	Specifies a LAMBDA= range, not a list
SBC	Uses Schwarz's Bayesian criterion
Spline	
DEGREE=	Specifies the degree of the spline
EVENLY=	Spaces the knots evenly
EXKNOTS=	Specifies exterior knots
KNOTS=	Specifies the interior knots or break points
NKNOTS=	Creates n knots
CLASS Variable	
CPREFIX=	Specifies CLASS coded variable name prefix
DEVIATIONS	Specifies a deviations-from-means coding
EFFECTS	Specifies a deviations-from-means coding
LPREFIX=	Specifies CLASS coded variable label prefix
ORDER=	Specifies order of CLASS variable levels
ORTHOGONAL	Specifies an orthogonal-contrast coding

Table 120.3 *continued*

Option	Description
SEPARATORS=	Specifies CLASS coded variable label separators
STANDORTH	Specifies a standardized-orthogonal coding
ZERO=	Controls reference levels
Box-Cox	
ALPHA=	Specifies confidence interval alpha
CLL=	Specifies convenient lambda list
CONVENIENT	Uses a convenient lambda
GEOMETRICMEAN	Scales transformation using geometric mean
LAMBDA=	Specifies power parameter list
Other t-options	
AFTER	Specifies operations occur after the expansion
CENTER	Specifies center before the analysis begins
NAME=	Renames variables
REFLECT	Reflects the variable around the mean
TSTANDARD=	Specifies transformation standardization
Z	Standardizes before the analysis begins

The following sections discuss the *t-options* available for nonoptimal, nonlinear fit, optimal, and other transformations.

Nonoptimal Transformation t-options

ORIGINAL

ORI

matches the variable's final mean and variance to the mean and variance of the original variable. By default, the mean and variance are based on the transformed values. The ORIGINAL *t-option* is available for all of the nonoptimal transformations.

Parameter t-options

PARAMETER=*number*

PAR=*number*

specifies the transformation parameter. The PARAMETER= *t-option* is available for the BOXCOX, EXP, LOG, POWER, SMOOTH, SSPLINE, and PBSPLINE transformations. For BOXCOX, the parameter is the value to add to each value of the variable before a Box-Cox transformation. For EXP, the parameter is the value to be exponentiated; for LOG, the parameter is the base value; and for POWER, the parameter is the power. For SMOOTH and SSPLINE, the parameter is the raw smoothing parameter. (See the SM= option for an alternative way to specify the smoothing parameter.) The default for the PARAMETER= *t-option* for the BOXCOX transformation is 0 and for the LOG and EXP transformations is $e = 2.718\ldots$. The default parameter for SMOOTH and SSPLINE is computed from SM=0. For the POWER transformation, you must specify the PARAMETER= *t-option*; there is no default. For PBSPLINE, the parameter is the order of the difference matrix, which provides some control over the smoothness of the transformation. The default order parameter with PBSPLINE is the maximum of the DEGREE= *t-option*, and 1. With PBSPLINE, the default is DEGREE=3 and PARAMETER=3, which works well for most problems.

SM=*n*

specifies a smoothing parameter in the range 0 to 100, just like PROC GPLOT uses. For example, SM=50 in PROC TRANSREG is equivalent to I=SM50 in the SYMBOL statement with PROC GPLOT. You can specify the SM= *t-option* only with the SMOOTH and SSPLINE transformations. The smoothness of the function increases as the value of the smoothing parameter increases. By default, SM=0.

Spline t-options

The following *t-options* are available with the SPLINE, MSPLINE and PBSPLINE transformations and with the PSPLINE and BSPLINE expansions.

DEGREE=*n*

DEG=*n*

specifies the degree of the spline transformation. The degree must be a nonnegative integer. The defaults are DEGREE=3 for SPLINE, PSPLINE, and BSPLINE variables and DEGREE=2 for MSPLINE variables.

The polynomial degree should be a small integer, usually 0, 1, 2, or 3. Larger values are rarely useful. If you have any doubt as to what degree to specify, use the default.

EVENLY< =*n*>

EVE< =*n*>

is used with the NKNOTS= *t-option* to space the knots evenly. The differences between adjacent knots are constant.

If you specify NKNOTS=*k* and EVENLY, *k* knots are created at

$$\text{minimum} + i((\text{maximum} - \text{minimum})/(k + 1))$$

for $i = 1, \ldots, k$. Here is an example:

```
spline(x / nknots=2 evenly)
```

When the variable x has a minimum of 4 and a maximum of 10, then the two interior knots are 6 and 8. Without the EVENLY *t-option*, the NKNOTS= *t-option* places knots at percentiles, so the knots are not evenly spaced. By default for the BSPLINE expansion and the SPLINE and MSPLINE transformations, the smaller exterior knots are all the same and all just a little smaller than the minimum. Similarly, by default, the larger exterior knots are all the same and all just a little larger than the maximum. However, if you specify EVENLY=*n*, then the *n* exterior knots are evenly spaced as well. The number of exterior knots must be greater than or equal to the degree. You can specify values larger than the degree when you want to interpolate slightly beyond the range or your data. The exterior knots must be less than the minimum or greater than the maximum; hence the knots across all sets are not precisely equally spaced. For example, with data ranging from 0 to 10, and with EVENLY=3 and NKNOTS=4, the first exterior knots are –4.000000000001, –2.000000000001, and –0.000000000001, the interior knots are 2, 4, 6, and 8, and the second exterior knots are 10.000000000001, 12.000000000001, and 14.000000000001.

With the BSPLINE and PSPLINE expansions and the SPLINE and MSPLINE transformations, evenly spaced knots are not the default. With the PBSPLINE transformation, evenly spaced interior and exterior knots are the default. If you want unevenly spaced knots with PBSPLINE, you must use the KNOTS= *t-option*.

EXKNOTS=*number-list*

EXK=*number-list*

specifies exterior knots for SPLINE and MSPLINE transformations and BSPLINE expansions. Usually, this *t-option* is not needed; PROC TRANSREG automatically picks suitable exterior knots. The only time you need to use this option is when you want to ensure that the exact same basis is used for different splines, such as when you apply coefficients from one spline transformation to a variable in a different data set (see the section "Scoring Spline Variables" on page 9944).

Specify one or two values. If the minimum EXKNOTS= value is less than the minimum data value, it is used as the exterior knot. If the maximum EXKNOTS= value is greater than the maximum data value, it is used as the exterior knot. Otherwise these values are ignored. When EXKNOTS= is specified with the CENTER or Z *t-options*, the knots apply to the original variable, not to the centered or standardized variable.

The B-spline transformations and expansions use a knot list consisting of exterior knots (values just smaller than the minimum), the specified (interior) knots, and exterior knots (values just larger than the minimum). You can use the DETAIL *a-option* to see all of these knots. If you use different exterior knots, you get different but equivalent B-spline bases. You can specify exterior knots in either the KNOTS= or EXKNOTS= *t-options*; however, for the BSPLINE expansion, the KNOTS= *t-option* creates extra all-zero basis columns, whereas the EXKNOTS= *t-option* gives you the correct basis. See the EVENLY= *t-option* for an alternative way to specify exterior knots.

KNOTS=*number-list* **|** *n* **TO** *m* **BY** *p*

KNO=*number-list* **|** *n* **TO** *m* **BY** *p*

specifies the interior knots or break points. By default, there are no knots. The first time you specify a value in the knot list, it indicates a discontinuity in the nth (from DEGREE=n) derivative of the transformation function at the value of the knot. The second mention of a value indicates a discontinuity in the $(n-1)$ derivative of the transformation function at the value of the knot. Knots can be repeated any number of times for decreasing smoothness at the break points, but the values in the knot list can never decrease.

You cannot use the KNOTS= *t-option* with the NKNOTS= *t-option*. You should keep the number of knots small (see the section "Specifying the Number of Knots" on page 10000).

NKNOTS=*n*

NKN=*n*

creates *n* knots, the first at the $100/(n+1)$th percentile, the second at the $200/(n+1)$th percentile, and so on. Knots are always placed at data values; there is no interpolation. For example, if NKNOTS=3, knots are placed at the 25th percentile, the median, and the 75th percentile. You can use the EVENLY= *t-option* along with NKNOTS= to get evenly spaced knots. By default, with the BSPLINE and PSPLINE expansions and the SPLINE and MSPLINE transformations, NKNOTS=0. By default, with the PBSPLINE transformation, NKNOTS=100.

The value specified for the NKNOTS= *t-option* must be $\geq$ 0.

You cannot use the NKNOTS= *t-option* with the KNOTS= *t-option*.

You should keep the number of knots small (see the section "Specifying the Number of Knots" on page 10000).

Penalized B-Spline t-options

The following *t-options* are available with the PBSPLINE transformation.

AIC

specifies that the procedure should select the smoothing parameter, λ, that minimizes the (Akaike 1973) information criterion (AIC). By default, the (AICC) criterion is minimized.

AICC

specifies that the procedure should select the smoothing parameter, λ, that minimizes the corrected Akaike information criterion (Hurvich, Simonoff, and Tsai 1998). This is the default criterion unless the AIC, CV, GCV, or SBC *t-option* is specified.

CV

specifies that the procedure should select the smoothing parameter, λ, that minimizes the cross validation criterion (CV). By default, the (AICC) criterion is minimized.

GCV

specifies that the procedure should select the smoothing parameter, λ, that minimizes the generalized cross validation criterion (Craven and Wahba 1979). By default, the (AICC) criterion is minimized.

LAMBDA=*number-list*

LAM=*number-list*

specifies a list of penalized B-spline smoothing parameters. By default, PROC TRANSREG considers lambdas in the range 0 to 1E6. Alternatively, you can specify the RANGE *t-option* with LAMBDA=, such as LAMBDA=1E3 1E5 RANGE, to only consider lambdas in a narrower range. Note that the algorithm might not actually evaluate the criterion at the minimum and maximum if it does not have to. In particular, it avoids evaluating the criterion at LAMBDA=0 (no smoothing) unless it is the only LAMBDA= value specified. You can also specify a list of lambdas, such as LAMBDA=1 TO 10, and the procedure selects the best lambda from the list. In all cases, the lambda that minimizes the specified criterion (or AICC by default) is chosen.

RANGE

RAN

specifies that the LAMBDA= *t-option* specifies two lambdas that define a range of values, from which an optimal lambda is selected. By default, PROC TRANSREG considers lambdas in the range 0 to 1E6.

SBC

specifies that the procedure should select the smoothing parameter, λ, that minimizes Schwarz's Bayesian criterion (Schwarz 1978; Judge et al. 1980). By default, the (AICC) criterion is minimized.

Class Variable t-options

CPREFIX=*n* | *number-list*

CPR=*n* | *number-list*

specifies the number of first characters of a CLASS expansion variable's name to use in constructing names for coded variables. When you specify CPREFIX= as an *a-option* or an *o-option*, it specifies the default for all CLASS variables. When you specify CPREFIX= as a *t-option*, it overrides the default only for selected variables. A different CPREFIX= value can be specified for each CLASS variable by specifying the CPREFIX=number-list *t-option*, like the ZERO=*formatted-value t-option*.

DEVIATIONS

DEV

requests a deviations-from-means coding of CLASS variables. The coded design matrix has values of 0, 1, and –1 for reference levels. This coding is referred to as "deviations-from-means," "effects," "center-point," or "full-rank" coding. For example, here is the coding for two-, three-, four-, and five-level factors:

	Number of Levels									
	Two	Three		Four			Five			
a	1	1	0	1	0	0	1	0	0	0
b	-1	0	1	0	1	0	0	1	0	0
c		-1	-1	0	0	1	0	0	1	0
d				-1	-1	-1	0	0	0	1
e							-1	-1	-1	-1

EFFECTS

EFF

See the DEVIATIONS *t-option*.

LPREFIX=*n* | *number-list*

LPR=*n* | *number-list*

specifies the number of first characters of a CLASS expansion variable's label (or name if no label is specified) to use in constructing labels for the coded variables. When you specify LPREFIX= as an *a-option* or an *o-option*, it specifies the default for all CLASS variables. When you specify LPREFIX= as a *t-option*, it overrides the default only for selected variables. A different LPREFIX= value can be specified for each CLASS variable by specifying the LPREFIX=number-list *t-option*, like the ZERO=*formatted-value t-option*.

ORDER=DATA | FREQ | FORMATTED | INTERNAL

ORD=DAT | FRE | FOR | INT

specifies the order in which the CLASS variable levels are to be reported. The default is ORDER=INTERNAL. For ORDER=FORMATTED and ORDER=INTERNAL, the sort order is machine dependent. When you specify ORDER= as an *a-option* or an *o-option*, it specifies the default ordering for all CLASS variables. When you specify ORDER= as a *t-option*, it overrides the default ordering only for selected variables. You can specify a different ORDER= value for each CLASS specification.

ORTHOGONAL

ORT

requests an orthogonal-contrast coding of CLASS variables. For example, here is the orthogonal-contrast coding for two-, three-, four-, and five-level factors:

	Number of Levels									
	Two	Three		Four			Five			
a	1	1	-1	1	-1	-1	1	-1	-1	-1
b	-1	0	2	0	2	-1	0	2	-1	-1
c		-1	-1	0	0	3	0	0	3	-1
d				-1	-1	-1	0	0	0	4
e							-1	-1	-1	-1

The sum of the coded values within each column is zero, all columns within a factor are orthogonal, and the ith column represents a contrast between the ith level and the combination of all preceding levels and the last level. The $\mathbf{X}$ matrix is orthogonal and $\mathbf{X}'\mathbf{X}$ is diagonal with this coding only if the experimental design is orthogonal.

SEPARATORS='*string-1*' <'*string-2*'>

SEP='*string-1*' <'*string-2*'>

specifies separators for creating CLASS expansion variable labels. By default, SEPARATORS=' ' ' * ' ("blank" and "blank asterisk blank"). When you specify SEPARATORS= as an *a-option* or an *o-option*, it specifies the default separators for all CLASS variables. When you specify SEPARATORS= as a *t-option*, it overrides the default only for selected variables. You can specify a different SEPARATORS= value for each CLASS specification.

STANDORTH

STA

ORTHEFFECT

requests a standardized-orthogonal coding of CLASS variables. For example, here is the standardized-orthogonal coding for two-, three-, four-, and five-level factors:

	Number of Levels									
	Two	Three		Four			Five			
a	1	1.22	-0.71	1.41	-0.82	-0.58	1.58	-0.91	-0.65	-0.50
b	-1	0.00	1.41	0.00	1.63	-0.58	0.00	1.83	-0.65	-0.50
c		-1.22	-0.71	0.00	0.00	1.73	0.00	0.00	1.94	-0.50
d				-1.41	-0.82	-0.58	0.00	0.00	0.00	2.00
e							-1.58	-0.91	-0.65	-0.50

The sum of the coded values within each column is zero, the sum of squares of the coded values within each column is equal to the number of levels, all columns within a factor are orthogonal, and the ith column represents a contrast between the ith level and the combination of all preceding levels and the last level. The $\mathbf{X}$ matrix is orthogonal and $\mathbf{X}'\mathbf{X}$ is diagonal ($\mathbf{X}'\mathbf{X} = n\mathbf{I}$, the number of observations times an identity matrix) with this coding only if the experimental design is orthogonal.

ZERO=FIRST | LAST | NONE | SUM

ZER=FIR | LAS | NON | SUM

ZERO='*formatted-value*' <'*formatted-value*' ... >

is used with CLASS variables. The default is ZERO=LAST.

The specification CLASS(variable / ZERO=FIRST) sets to missing the coded variable for the first of the sorted categories, implying a zero coefficient for that category.

The specification CLASS(variable / ZERO=LAST) sets to missing the coded variable for the last of the sorted categories, implying a zero coefficient for that category.

The specification CLASS(variable / ZERO='*formatted-value*') sets to missing the coded variable for the category with a formatted value that matches '*formatted-value*', implying a zero coefficient for that category. With ZERO=*formatted-value*, the first formatted value applies to the first variable in the specification, the second formatted value applies to the next variable that was not previously mentioned, and so on. For example, `class(a a*b b b*c c / zero='x' 'y' 'z')` specifies that the reference level for a is 'x', for b is 'y', and for c is 'z'. With ZERO='*formatted-value*', the procedure first looks for exact matches between the formatted values and the specified value. If none are found, leading blanks are stripped from both and the values are compared again. If zero or two or more matches are found, warnings are issued.

The specifications ZERO=FIRST, ZERO=LAST, and ZERO='*formatted-value*' are used for reference cell models. The Intercept parameter estimate is the marginal mean for the reference cell, and the other marginal means are obtained by adding the intercept to the coded variable coefficients.

The specification CLASS(variable / ZERO=NONE) sets to missing none of the coded variables. The columns of the expansion sum to a column of ones, so an implicit intercept model is fit. If you specify ZERO=NONE for more than one variable, the model is less than full rank. In the model `model identity(y) = class(x / zero=none)`, the coefficients are cell means.

The specification CLASS(variable / ZERO=SUM) sets to missing none of the coded variables, and the coefficients for the coded variables created from the variable sum to 0. This creates a less-than-full-rank model, but the coefficients are uniquely determined due to the sum-to-zero constraint.

In the presence of iterative transformations, hypothesis tests for ZERO=NONE and ZERO=SUM levels are not exact; they are liberal because a model with an explicit intercept is fit inside the iterations. There is no provision for adjusting the transformations while setting to 0 a parameter that is redundant given the explicit intercept and the other parameters.

Box-Cox t-options

The following *t-options* are available only with the BOXCOX transformation of the dependent variable (see the section "Box-Cox Transformations" on page 9923 and Example 120.2).

ALPHA=*p*

ALP=*p*

specifies the Box-Cox alpha for the confidence interval for the power parameter. By default, ALPHA=0.05.

CLL=*number-list*

specifies the Box-Cox convenient lambda list. The default is CLL=1 0 0.5 –1 –0.5 2 –2 3 –3. By default, a linear transformation ($\lambda = 1$) is preferred over log ($\lambda = 0$), square root ($\lambda = 0.5$), inverse ($\lambda = -1$), inverse square root ($\lambda = -0.5$), quadratic ($\lambda = 2$), inverse quadratic ($\lambda = -2$), cubic ($\lambda = 3$), and inverse cubic ($\lambda = -3$) transformations. A value in the CLL= list is considered only if it is also in the LAMBDA= list.[1] When the confidence interval for the power parameter includes one of the parameters in the CLL= list, PROC TRANSREG reports it. In addition, if you specify the CONVENIENT *t-option*, then PROC TRANSREG uses the first convenient power parameter in the CLL=*number-list* that is in the confidence interval. For example, if the optimal power parameter is 0.25 and 0.0 is in the confidence interval but 1.0 is not, then the convenient power parameter is 0.0.

[1] This is a new feature in SAS/STAT 14.1. If you want other values from the CLL= list to be considered as in previous releases, you must specify those values in the LAMBDA= *t-option*.

CONVENIENT

CON

requests that PROC TRANSREG use a power parameter from the CLL= *t-option* for the final transformation instead of using the optimal LAMBDA= *t-option* value if a CLL= power parameter is in both the confidence interval and the LAMBDA=*number-list*. For more information, see the CLL= *t-option*.

GEOMETRICMEAN

GEO

divides the Box-Cox transformation by $\dot{y}^{\lambda-1}$, where $\dot{y}$ is the geometric mean of the variable to be transformed. This form of the Box-Cox transformation essentially converts the transformation back to original units, and hence it permits direct comparison of the residual sums of squares for models with different power parameters.

LAMBDA=*number-list*

LAM=*number-list*

specifies a list of Box-Cox power parameters. By default, LAMBDA=–3 TO 3 BY 0.25. PROC TRANSREG evaluates the log likelihood for each power parameter in the list and chooses the best one. However, when the CONVENIENT *t-option* is specified, PROC TRANSREG can choose a convenient parameter from the confidence interval instead of choosing the optimal power parameter from the LAMBDA= list. For example, if the optimal power parameter is 0.25 and 0.0 is in the confidence interval but 1.0 is not, then the convenient power parameter 0.0 (log transformation) is chosen instead of the more optimal parameter 0.25. You can specify the convenient power parameters by using the CLL= *t-option*.

Other t-options

AFTER

AFT

requests that certain operations occur after the expansion. This *t-option* affects the NKNOTS= *t-option* when the SPLINE or MSPLINE transformation is crossed with a CLASS specification. For example, if the original spline variable (1 2 3 4 5 6 7 8 9) is expanded into the three variables (1 2 3 0 0 0 0 0 0), (0 0 0 4 5 6 0 0 0), and (0 0 0 0 0 0 7 8 9), then, by default, NKNOTS=1 would use the overall median of 5 as the knot for all three variables. When you specify the AFTER *t-option*, the knots for the three variables are 2, 5, and 8. Note that the structural zeros are ignored when the internal knot list is created, but they are not ignored for the exterior knots.

You can also specify the AFTER *t-option* with the RANK, SMOOTH, and PBSPLINE transformations. The following specifications compute ranks and smooth transformations within groups, after crossing, ignoring the structural zeros:

```
class(x / zero=none) | rank(z / after)
class(x / zero=none) | smooth(z / after)
```

CENTER

CEN

centers the variables before the analysis begins (in contrast to the TSTANDARD=CENTER option, which centers after the analysis ends). The CENTER *t-option* can be used instead of running PROC STANDARD before PROC TRANSREG (see the section "Centering" on page 10032). When the KNOTS= *t-option* is specified with CENTER, the knots apply to the original variable, not to the centered variable. PROC TRANSREG centers the knots.

NAME=(*variable-list*)

NAM=(*variable-list*)

renames variables as they are used in the MODEL statement. This *t-option* lets you use a variable more than once.

For example, if x is a character variable, then the following step stores both the original character variable x and a numeric variable xc that contains category numbers in the OUT= data set:

```
proc transreg data=a;
   model identity(y) = opscore(x / name=(xc));
   output;
   id x;
run;
```

With the CLASS and IDENTITY transformations, which can contain interaction effects, the first name applies to the first variable in the specification, the second name applies to the next variable that was not previously mentioned, and so on. For example, `identity(a a * b b b * c c / name=(g h i))` specifies that the new name for a is g, for b is h, and for c is i. The same assignment is used for the (not useful) specification `identity(a a b b c c / name=(g h i))`. For all *transforms* other than CLASS and IDENTITY (all those in which interactions are not supported), repeated variables are not handled specially. For example, `spline(a a b b c c / name=(a g b h c i))` creates six variables: a copy of a named a, another copy of a named g, a copy of b named b, another copy of b named h, a copy of c named c, and another copy of c named i.

REFLECT

REF

reflects the transformation

$$y = -(y - \bar{y}) + \bar{y}$$

after the iterations are completed and before the final standardization and results calculations. This *t-option* is particularly useful with the dependent variable in a conjoint analysis. When the dependent variable consists of ranks with the most preferred combination assigned 1.0, the REFLECT *t-option* reflects the transformation so that positive utilities mean high preference. (See Example 120.4.)

TSTANDARD=CENTER | NOMISS | ORIGINAL | Z

TST=CEN | NOM | ORI | Z

specifies the standardization of the transformed variables for the hypothesis tests and in the OUT= data set (see the section “Centering” on page 10032). By default, TSTANDARD=ORIGINAL. When you specify TSTANDARD= as an *a-option* or an *o-option*, it determines the default standardization for all variables. When you specify TSTANDARD= as a *t-option*, it overrides the default standardization only for selected variables. You can specify a different TSTANDARD= value for each transformation. For example, to perform a redundancy analysis with standardized dependent variables, specify the following:

```
model identity(y1-y4 / tstandard=z) = identity(x1-x10);
```

Z

centers and standardizes the variables to variance one before the analysis begins (in contrast to the TSTANDARD=Z option, which standardizes after the analysis ends). The Z *t-option* can be used instead of running PROC STANDARD before PROC TRANSREG (see the section "Centering" on page 10032). When the KNOTS= *t-option* is specified with Z, the knots apply to the original variable, not to the standardized variable. PROC TRANSREG standardizes the knots.

Algorithm Options (a-options)

This section discusses the options that can appear in the PROC TRANSREG or MODEL statement as *a-options*. They are listed after the entire model specification and after a slash. Here is an example:

```
proc transreg;
   model spline(y / nknots=3)=log(x1 x2 / parameter=2)
         / nomiss maxiter=50;
   output;
run;
```

In the preceding statements, NOMISS and MAXITER= are *a-options*. (SPLINE and LOG are *transforms*, and NKNOTS= and PARAMETER= are *t-options*.) The statements find a spline transformation with 3 knots on y and a base 2 logarithmic transformation on x1 and x2. The NOMISS *a-option* excludes all observations with missing values, and the MAXITER= *a-option* specifies the maximum number of iterations.

Table 120.4 summarizes the *a-options* available in the PROC TRANSREG or MODEL statement.

Table 120.4 Options Available in the PROC TRANSREG or MODEL Statement

Option	Description
Input Control	
REITERATE	Restarts iterations
TYPE=	Specifies input observation type
Method and Iterations	
CCONVERGE=	Specifies minimum criterion change
CONVERGE=	Specifies minimum data change
MAXITER=	Specifies maximum number of iterations
METHOD=	Specifies iterative algorithm
NCAN=	Specifies number of canonical variables
NSR	Specifies no restrictions on smoothing models
SINGULAR=	Specifies singularity criterion
SOLVE	Attempts direct solution instead of iteration
Missing Data Handling	
INDIVIDUAL	Fits each model individually (METHOD=MORALS)
MONOTONE=	Includes monotone special missing values
NOMISS	Excludes observations with missing values
UNTIE=	Unties special missing values
Intercept and CLASS Variables	
CPREFIX=	Specifies CLASS coded variable name prefix

Table 120.4 *continued*

Option	Description
LPREFIX=	Specifies CLASS coded variable label prefix
NOINT	Specifies no intercept or centering
ORDER=	Specifies order of CLASS variable levels
REFERENCE=	Controls output of reference levels
SEPARATORS=	Specifies CLASS coded variable label separators
Control Displayed Output	
ALPHA=	Specifies confidence limits alpha
CL	Displays parameter estimate confidence limits
DETAIL	Displays model specification details
HISTORY	Displays iteration histories
NOPRINT	Suppresses displayed output
PBOXCOXTABLE	Prints the Box-Cox log likelihood table
RSQUARE	Displays the R square
SHORT	Suppresses the iteration histories
SS2	Displays regression results
TEST	Displays ANOVA table
TSUFFIX=	Shortens transformed variable labels
UTILITIES	Displays conjoint part-worth utilities
Standardization	
ADDITIVE	Fits additive model
NOZEROCONSTANT	Does not zero constant variables
TSTANDARD=	Specifies transformation standardization

The following list provides details about these *a-options*. The *a-options* are available in the PROC TRANSREG or MODEL statement.

ADDITIVE

ADD

creates an additive model by multiplying the values of each independent variable (after the TSTANDARD= standardization) by that variable's corresponding multiple regression coefficient. This process scales the independent variables so that the predicted-values variable for the final dependent variable is simply the sum of the final independent variables. An additive model is a univariate multiple regression model. As a result, the ADDITIVE *a-option* is not valid if METHOD=CANALS, or if METHOD=REDUNDANCY or METHOD=UNIVARIATE with more than one dependent variable.

ALPHA=*number*

ALP=*number*

specifies the level of significance for all of the confidence limits. By default, ALPHA=0.05.

CCONVERGE=*n*

CCO=*n*

specifies the minimum change in the criterion being optimized (squared multiple correlation for METHOD=MORALS and METHOD=UNIVARIATE, average squared multiple correlation for

METHOD=REDUNDANCY, average squared canonical correlation for METHOD=CANALS) that is required to continue iterating. By default, CCONVERGE=0.0.

CL

requests confidence limits on the parameter estimates in the displayed output.

CONVERGE=*n*

CON=*n*

specifies the minimum average absolute change in standardized variable scores that is required to continue iterating. By default, CONVERGE=0.00001. Average change is computed over only those variables that can be transformed by the iterations; that is, all LINEAR, OPSCORE, MONOTONE, UNTIE, SPLINE, MSPLINE, and SSPLINE variables and nonoptimal transformation variables with missing values.

CPREFIX=*n*

CPR=*n*

specifies the number of first characters of a CLASS expansion variable's name to use in constructing names for coded variables. Coded variable names are constructed from the first *n* characters of the CLASS expansion variable's name and the first $32 - n$ characters of the formatted CLASS expansion variable's value. For example, if the variable ClassVariable has values 1, 2, and 3, then, by default, the coded variables are named ClassVariable1, ClassVariable2, and ClassVariable3. However, with CPREFIX=5, the coded variables are named Class1, Class2, and Class3. When CPREFIX=0, coded variable names are created entirely from the CLASS expansion variable's formatted values. Valid values range from –1 to 31, where –1 indicates the default calculation and 0 to 31 are the number of prefix characters to use. The default, –1, sets *n* to 32 – min(32, max(2, *fl*)), where *fl* is the format length. When you specify CPREFIX= as an *a-option* or an *o-option*, it specifies the default for all CLASS variables. When you specify CPREFIX= as a *t-option*, it overrides the default only for selected variables.

DETAIL

DET

reports on details of the model specification. For example, it reports the knots and coefficients for splines, reference levels for CLASS variables, Box-Cox results, the smoothing parameter, and so on. The DETAIL option can take two optional suboptions, NOCOEFFICIENTS and NOKNOTS (or NOC and NOK). To suppress knots from the details listing, specify DETAIL(NOKNOTS). To suppress coefficients from the details listing, specify DETAIL(NOCOEFFICIENTS). To suppress both knots and coefficients from the details listing, specify DETAIL(NOKNOTS NOCOEFFICIENTS).

SOLVE

SOL

DUMMY

DUM

provides a canonical initialization. When there are no monotonicity constraints, when there is at most one canonical variable in each set, and when there is enough available memory, PROC TRANSREG (with the SOLVE *a-option*) can usually directly solve for the optimal solution in only one iteration. The initialization iteration is number 0, which is slower and uses more memory than other iterations. However, for some models, specifying the SOLVE *a-option* can greatly decrease the amount of time required to find the optimal transformations. During iteration 0, each variable is replaced by an expanded variable and the model is fit to the larger, expanded set of variables. For example, an OPSCORE variable is expanded into coded (or "dummy") variables, as if CLASS were specified, and a SPLINE variable is expanded into a B-spline basis, as if BSPLINE were specified. Then for each expanded variable, the results of iteration zero are constructed by multiplying the expanded basis times the $\boldsymbol{\beta}$ subvector to get the optimal transformation. This *a-option* can be useful even in models where a direct solution is not possible, because it provides good initial transformations of all the variables.

HISTORY

HIS

displays the iteration histories even when the NOPRINT *a-option* is specified.

INDIVIDUAL

IND

fits each model for each dependent variable individually. This means, for example, that when INDIVIDUAL is specified, missing values in one dependent variable will not cause that observation to be deleted for the other models with the other dependent variables. In contrast, by default, missing values in any variable in any model can cause the observation to be deleted for all models. The INDIVIDUAL *a-option* can be specified only with METHOD=MORALS.

This *a-option* also affects the order of the output. By default, the number of observations table is printed once at the beginning of the output. With INDIVIDUAL, a number of observations table appears for each model.

LPREFIX=*n*

LPR=*n*

specifies the number of first characters of a CLASS expansion variable's label (or name if no label is specified) to use in constructing labels for coded variables. Coded variable labels are constructed from the first *n* characters of the CLASS expansion variable's name and the first 127 – *n* characters of the formatted CLASS expansion variable's value. Valid values range from –1 to 127. Values of 0 to 127 specify the number of name or label characters to use. The default is –1, which specifies that PROC TRANSREG should pick a value depending on the length of the prefix and the formatted class value. When you specify LPREFIX= as an *a-option* or an *o-option*, it determines the default for all CLASS variables. When you specify LPREFIX= as a *t-option*, it overrides the default only for selected variables.

MAXITER=*n*

MAX=*n*

specifies the maximum number of iterations (see the section "Controlling the Number of Iterations" on page 9989). By default, MAXITER=30. You can specify MAXITER=0 to save time when no transformations are requested.

METHOD=CANALS | MORALS | REDUNDANCY | UNIVARIATE

MET=CAN | MOR | RED | UNI

specifies the iterative algorithm. By default, METHOD=UNIVARIATE, unless you specify options that cannot be handled by the UNIVARIATE algorithm. Specifically, the default is METHOD=MORALS for the following situations:

- if you specify LINEAR, OPSCORE, MONOTONE, UNTIE, SPLINE, MSPLINE, or SSPLINE transformations for the independent variables

- if you specify the ADDITIVE *a-option* with more than one dependent variable
- if you specify the IAPPROXIMATIONS *o-option*
- if you specify the INDIVIDUAL *a-option*
- if ODS Graphics is enabled, regression plots are produced, and there is more than one dependent variable

CANALS specifies canonical correlation with alternating least squares. This jointly transforms all dependent and independent variables to maximize the average of the first n squared canonical correlations, where n is the value of the NCAN= *a-option*.

MORALS specifies multiple optimal regression with alternating least squares. This transforms each dependent variable, along with the set of independent variables, to maximize the squared multiple correlation.

REDUNDANCY jointly transforms all dependent and independent variables to maximize the average of the squared multiple correlations (see the section "Redundancy Analysis" on page 9994).

UNIVARIATE transforms each dependent variable to maximize the squared multiple correlation, while the independent variables are not transformed.

MONOTONE=*two-letters*

MON=*two-letters*

specifies the first and last special missing value in the list of those special missing values to be estimated with within-variable order and category constraints. By default, there are no order constraints on missing value estimates. The *two-letters* value must consist of two letters in alphabetical order. For example, MONOTONE=DF means that the estimate of .D must be less than or equal to the estimate of .E, which must be less than or equal to the estimate of .F; no order constraints are placed on estimates of ._, .A through .C, and .G through .Z. For details, see the section "Missing Values" on page 9987.

NCAN=*n*

NCA=*n*

specifies the number of canonical variables to use in the METHOD=CANALS algorithm. By default, NCAN=1. The value of the NCAN= *a-option* must be $\geq$ 1.

When canonical coefficients and coordinates are included in the OUT= data set, the NCAN= *a-option* also controls the number of rows of the canonical coefficient matrices in the data set. If you specify an NCAN= value larger than the minimum of the number of dependent variables and the number of independent variables, PROC TRANSREG displays a warning and sets the NCAN= *a-option* to the maximum value.

NOINT

NOI

omits the intercept from the OUT= data set and suppresses centering of data. You cannot specify the NOINT *a-option* with iterative transformations since there is no provision for optimal scaling without an intercept. The NOINT *a-option* can be specified only when there is no implicit intercept and when all of the data in a BY group absolutely will not change during the iterations.

NOMISS

NOM

excludes all observations with missing values from the analysis, but does not exclude them from the OUT= data set. If you omit the NOMISS *a-option*, PROC TRANSREG simultaneously computes the optimal transformations of the nonmissing values and estimates the missing values that minimize squared error. For details, see the section "Missing Values" on page 9987.

Casewise deletion of observations with missing values occurs when the NOMISS *a-option* is specified, when there are missing values in expansions, when there are missing values in METHOD=UNIVARIATE independent variables, when there are weights less than or equal to 0, or when there are frequencies less than 1. Excluded observations are output with a blank value for the _TYPE_ variable, and they have a weight of 0. They do not contribute to the analysis but are scored and transformed as *supplementary* or passive observations.

See the section "Passive Observations" on page 9993 for more information about excluded observations.

NOPRINT

NOP

suppresses the display of all output unless you specify the HISTORY *a-option*. The NOPRINT *a-option* without the HISTORY *a-option* disables the Output Delivery System (ODS), including ODS Graphics, for the duration of the procedure run. The NOPRINT *a-option* with the HISTORY *a-option* disables all output except the iteration history, again including ODS Graphics, for the duration of the procedure run. For more information, see Chapter 20, "Using the Output Delivery System."

NOZEROCONSTANT

NOZERO

NOZ

specifies that constant variables are expected and should not be zeroed. By default, constant variables are zeroed. This option is useful when PROC TRANSREG is used to code experimental designs for discrete choice models (see the section "Discrete Choice Experiments: DESIGN, NORESTORE, NOZERO" on page 10031). When these designs are very large, it might be more efficient to use the DESIGN=*n* *a-option*. It might be that attributes are constant within a block of *n* observations, so you need to specify the NOZEROCONSTANT *a-option* to get the correct results. You can specify this option in the PROC TRANSREG, MODEL, and OUTPUT statements.

NSR

specifies that no restrictions are placed on the use of SMOOTH and SSPLINE and the ordinary least squares is used to find the coefficients and predicted values. By default, only certain types of models can be specified with SMOOTH and ordinary least squares is not used to find the coefficients and predicted values. See the section "Smoothing Splines Changes and Enhancements" on page 9966 for more information about the NSR option and smooth transformations.

ORDER=DATA | FREQ | FORMATTED | INTERNAL

ORD=DAT | FRE | FOR | INT

specifies the order in which the CLASS variable levels are to be reported. The default is ORDER=INTERNAL. For ORDER=FORMATTED and ORDER=INTERNAL, the sort order is machine dependent. When you specify ORDER= as an *a-option* or an *o-option*, it determines the default ordering for all CLASS variables. When you specify ORDER= as a *t-option*, it overrides the default ordering only for selected variables.

DATA	sorts by order of appearance in the input data set.
FORMATTED	sorts by formatted value.
FREQ	sorts by descending frequency count; levels with the most observations appear first.
INTERNAL	sorts by unformatted value.

PBOXCOXTABLE

PBO

prints the Box-Cox table with the log likelihood displayed as a function of lambda. The important information in this table is displayed in the Box-Cox plot, so when ODS Graphics is enabled and the plot is produced, the table is not produced by default. When ODS Graphics is not enabled or when the plot is not produced, the table is produced by default. Specify the PBOXCOXTABLE option if you want to see the table in addition to the plot.

REFERENCE=NONE | MISSING | ZERO

REF=NON | MIS | ZER

specifies how reference levels of CLASS variables are to be treated. The options are REFERENCE=NONE, the default, in which reference levels are suppressed; REFERENCE=MISSING, in which reference levels are displayed and output with missing values; and REFERENCE=ZERO, in which reference levels are displayed and output with zeros. You can specify the REFERENCE= option in the PROC TRANSREG, MODEL, or OUTPUT statement, and you can specify it independently for the OUT= data set and the displayed output. When you specify it in only one statement, it sets the option for both the displayed output and the OUT= data set.

REITERATE

REI

enables PROC TRANSREG to use previous transformations as starting points. The REITERATE *a-option* affects only variables that are iteratively transformed (specified as LINEAR, OPSCORE, MONOTONE, UNTIE, SPLINE, MSPLINE, and SSPLINE). For iterative transformations, the REITERATE *a-option* requests a search in the input data set for a variable that consists of the value of the TDPREFIX= or TIPREFIX= *o-option* followed by the original variable name. If such a variable is found, it is used to provide the initial values for the first iteration. The final transformation is a member of the transformation family defined by the original variable, not the transformation family defined by the initialization variable. See the section "Using the REITERATE Algorithm Option" on page 9990 for more information about the REITERATE option.

RSQUARE

RSQ

prints a table with only the model R square.

SEPARATORS='*string-1***' < '***string-2***' >**

SEP='*string-1***' < '***string-2***' >**

specifies separators for creating CLASS expansion variable labels. By default, SEPARATORS=' ' ' * ' ("blank" and "blank asterisk blank"). The first value is used to separate variable names and values in interactions. The second value is used to separate interaction components. For example, the label for the coded variable for the A=1 and B=2 cell is, by default, 'A 1 * B 2'. If SEPARATORS='=' 'x' is specified, then the label is 'A=1xB=2'. When you specify SEPARATORS= as an *a-option* or an *o-option*, it determines the default separators for all CLASS variables. When you specify SEPARATORS= as a *t-option*, it overrides the default only for selected variables.

SHORT

SHO

suppresses the iteration histories.

SINGULAR=*n*

SIN=*n*

specifies the largest value within rounding error of zero. By default, SINGULAR=1E–12. PROC TRANSREG uses the value of the SINGULAR= *a-option* for checking $1 - \mathrm{R}^2$ when constructing full-rank matrices of predictor variables, checking denominators before dividing, and so on. PROC TRANSREG computes the regression coefficients by sweeping with rational pivoting.

SS2

produces a regression table based on Type II sums of squares. Tests of the contribution of each transformation to the overall model are displayed and output to the OUTTEST= data set when you specify the OUTTEST= option. When you specify the SS2 *a-option*, the TEST *a-option* is automatically specified for you. See the section "Hypothesis Tests" on page 10002 for more information about the TEST and SS2 options. You can suppress the variable labels in the regression tables by specifying the NOLABEL option in the OPTIONS statement.

TEST

TES

generates an ANOVA table. PROC TRANSREG tests the null hypothesis that the vector of scoring coefficients for all of the transformations is zero. See the section "Hypothesis Tests" on page 10002 for more information about the TEST option.

TSUFFIX=*n*

TSU=*n*

specifies the number of characters in "Transformation" to append to variable labels for transformed variables. By default, all characters are used.

TSTANDARD=CENTER | NOMISS | ORIGINAL | Z

TST=CEN | NOM | ORI | Z

specifies the standardization of the transformed variables for the hypothesis tests and in the OUT= data set. By default, TSTANDARD=ORIGINAL. When you specify TSTANDARD= as an *a-option* or an *o-option*, it determines the default standardization for all variables. When you specify TSTANDARD= as a *t-option*, it overrides the default standardization only for selected variables.

CENTER centers the output variables to mean zero, but the variances are the same as the variances of the input variables.

NOMISS sets the means and variances of the transformed variables in the OUT= data set, computed over all output values that correspond to nonmissing values in the input data set, to the means and variances computed from the nonmissing observations of the original variables. The TSTANDARD=NOMISS specification is useful with missing data. When a variable is linearly transformed, the final variable contains the original nonmissing values and the missing value estimates. In other words, the nonmissing values are unchanged. If your data have no missing values, TSTANDARD=NOMISS and TSTANDARD=ORIGINAL produce the same results.

ORIGINAL sets the means and variances of the transformed variables to the means and variances of the original variables. This is the default.

Z standardizes the variables to mean zero, variance one.

The final standardization is affected by other options. If you also specify the ADDITIVE *a-option*, the TSTANDARD= option specifies an intermediate step in computing the final means and variances. The final independent variables, along with their means and standard deviations, are scaled by the regression coefficients, creating an additive model with all coefficients equal to one.

For nonoptimal variable transformations, the means and variances of the original variables are actually the means and variances of the nonlinearly transformed variables, unless you specify the ORIGINAL nonoptimal *t-option* in the MODEL statement. For example, if a variable x with no missing values is specified as LOG, then, by default, the final transformation of x is simply the log of x, not the log of x standardized to the mean of x and variance of x.

TYPE='*text***'|***name*

TYP='*text***'|***name*

specifies the valid value for the _TYPE_ variable in the input data set. If PROC TRANSREG finds an input _TYPE_ variable, it uses only observations with a _TYPE_ value that matches the TYPE= value. This enables a PROC TRANSREG OUT= data set containing coefficients to be used as input to PROC TRANSREG without requiring a WHERE statement to exclude the coefficients. If a _TYPE_ variable is not in the data set, all observations are used. The default is TYPE='SCORE', so if you do not specify the TYPE= *a-option*, only observations with _TYPE_='SCORE' are used. Do not confuse this *a-option* with the data set TYPE= option. The DATA= data set must be an ordinary SAS data set.

PROC TRANSREG displays a note when it reads observations with blank values of _TYPE_, but it does not automatically exclude those observations. Data sets created by the TRANSREG and PRINQUAL procedures have blank _TYPE_ values for those observations that were excluded from the analysis due to nonpositive weights, nonpositive frequencies, or missing data. When these observations are read again, they are excluded for the same reason that they were excluded from their original analysis, not because their _TYPE_ value is blank.

UNTIE=*two-letters*

UNT=*two-letters*

specifies the first and last special missing values in the list of those special missing values that are to be estimated with within-variable order constraints but no category constraints. The *two-letters* value must consist of two letters in alphabetical order. By default, there are category constraints but no order constraints on special missing value estimates. For details, see the sections "Missing Values" on page 9987 and "Optimal Scaling" on page 9997.

UTILITIES

UTI

produces a table of the part-worth utilities from a conjoint analysis. Utilities, their standard errors, and the relative importance of each factor are displayed and output to the OUTTEST= data set when you specify the OUTTEST= option. When you specify the UTILITIES *a-option*, the TEST *a-option* is automatically specified for you. See Example 120.4 and Example 120.5 for more information about conjoint analysis.

OUTPUT Statement

OUTPUT OUT=*SAS-data-set < o-options >* **;**

The OUTPUT statement creates a new SAS data set that contains coefficients, marginal means, and information about the original and transformed variables. The information about original and transformed variables composes the score partition of the data set; observations have _TYPE_='SCORE'. The coefficients and marginal means compose the coefficient partition of the data set; observations have _TYPE_='M COEFFI' or _TYPE_='MEAN'. Other values of _TYPE_ are possible; for details, see "_TYPE_ and _NAME_ Variables" later in this chapter. For details about data set structure, see the section "Output Data Set" on page 10004. To specify the name of the output data set, use the OUT= option.

OUT=*SAS-data-set*

specifies the output data set for the data, transformed data, predicted values, residuals, scores, coefficients, and so on. When you use an OUTPUT statement but do not use the OUT= specification, PROC TRANSREG creates a data set and uses the DATA*n* convention. If you want to create a SAS data set in a permanent library, you must specify a two-level name. For more information about permanent libraries and SAS data sets, see *SAS Language Reference: Concepts*.

To control the contents of the data set and variable names, use one or more of the *o-options*. You can also specify these options in the PROC TRANSREG statement.

Output Options (o-options)

Table 120.5 summarizes the options available in the OUTPUT statement. These options include the OUT= option and all of the *o-options*. Many of the statistics created in the OUTPUT statement are exactly the same as statistics created by PROC REG. More details are given in the sections "Predicted and Residual Values" on page 8213, "Model Fit and Diagnostic Statistics" on page 8220 in Chapter 100, "The REG Procedure," and Chapter 4, "Introduction to Regression Procedures."

Table 120.5 Options Available in the OUTPUT Statement

Option	Description
Identify output data set	
OUT=	Outputs data set
Predicted Values, Residuals, Scores	
CANONICAL	Outputs canonical scores
CLI	Outputs individual confidence limits
CLM	Outputs mean confidence limits
DESIGN=	Specifies design matrix coding
DREPLACE	Replaces dependent variables
IREPLACE	Replaces independent variables
LEVERAGE	Outputs leverage
NORESTOREMISSING	Does not restore missing values
NOSCORES	Suppresses output of scores
PREDICTED	Outputs predicted values
REDUNDANCY=	Outputs redundancy variables
REPLACE	Replaces all variables
RESIDUALS	Outputs residuals

Table 120.5 *continued*

Option	Description
Output Data Set Coefficients	
COEFFICIENTS	Outputs coefficients
COORDINATES=	Outputs ideal point coordinates
MEANS	Outputs marginal means
MREDUNDANCY	Outputs redundancy analysis coefficients
Output Data Set Variable Name Prefixes	
ADPREFIX=	Specifies dependent variable approximations
AIPREFIX=	Specifies independent variable approximations
CDPREFIX=	Specifies canonical dependent variables
CILPREFIX=	Specifies conservative individual lower CL
CIPREFIX=	Specifies canonical independent variables
CIUPREFIX=	Specifies conservative-individual-upper CL
CMLPREFIX=	Specifies conservative-mean-lower CL
CMUPREFIX=	Specifies conservative-mean-upper CL
DEPENDENT=	Specifies METHOD=MORALS untransformed dependent
LILPREFIX=	Specifies liberal-individual-lower CL
LIUPREFIX=	Specifies liberal-individual-upper CL
LMLPREFIX=	Specifies liberal-mean-lower CL
LMUPREFIX=	Specifies liberal-mean-upper CL
RDPREFIX=	Specifies residuals
PPREFIX=	Specifies predicted values
RPREFIX=	Specifies redundancy variables
TDPREFIX=	Specifies transformed dependents
TIPREFIX=	Specifies transformed independents
Macros Variables	
MACRO	Creates macro variables
Other Options	
APPROXIMATIONS	Outputs dependent and independent approximations
CCC	Outputs canonical correlation coefficients
CEC	Outputs canonical elliptical point coordinates
CPC	Outputs canonical point coordinates
CQC	Outputs canonical quadratic point coordinates
DAPPROXIMATIONS	Outputs approximations to transformed dependents
IAPPROXIMATIONS	Outputs approximations to transformed independents
MEC	Outputs elliptical point coordinates
MPC	Outputs point coordinates
MQC	Outputs quadratic point coordinates
MRC	Outputs multiple regression coefficients

For the coefficients partition, the COEFFICIENTS, COORDINATES, and MEANS *o-options* provide the coefficients that are appropriate for your model. For more explicit control of the coefficient partition, use the options that control details and prefixes. The following list provides details about these options.

ADPREFIX=*name*

ADP=*name*

specifies a prefix for naming the dependent variable predicted values. The default is ADPREFIX=P when you specify the PREDICTED *o-option*; otherwise, it is ADPREFIX=A. When you specify the ADPREFIX= *o-option*, the PREDICTED *o-option* is automatically specified for you. The ADPREFIX= *o-option* is the same as the PPREFIX= *o-option*.

AIPREFIX=*name*

AIP=*name*

specifies a prefix for naming the independent variable approximations. The default is AIPREFIX=A. When you specify the AIPREFIX= *o-option*, the IAPPROXIMATIONS *o-option* is automatically specified for you.

APPROXIMATIONS

APPROX

APP

is equivalent to specifying both the DAPPROXIMATIONS and the IAPPROXIMATIONS *o-options*. If you specify METHOD=UNIVARIATE, then the APPROXIMATIONS *o-option* specifies only the DAPPROXIMATIONS *o-option*.

CANONICAL

CAN

outputs canonical variables to the OUT= data set. When you specify METHOD=CANALS, the CANONICAL *o-option* is automatically specified for you. The CDPREFIX= *o-option* specifies a prefix for naming the dependent canonical variables (default Cand), and the CIPREFIX= *o-option* specifies a prefix for naming the independent canonical variables (default Cani).

CCC

outputs canonical correlation coefficients to the OUT= data set.

CDPREFIX=*name*

CDP=*name*

provides a prefix for naming the canonical dependent variables. The default is CDPREFIX=Cand. When you specify the CDPREFIX= *o-option*, the CANONICAL *o-option* is automatically specified for you.

CEC

outputs canonical elliptical point model coordinates to the OUT= data set.

CILPREFIX=*name*

CIL=*name*

specifies a prefix for naming the conservative-individual-lower confidence limits. The default prefix is CIL. When you specify the CILPREFIX= *o-option*, the CLI *o-option* is automatically specified for you.

CIPREFIX=*name*

CIP=*name*

provides a prefix for naming the canonical independent variables. The default is CIPREFIX=Cani. When you specify the CIPREFIX= *o-option*, the CANONICAL *o-option* is automatically specified for you.

CIUPREFIX=*name*

CIU=*name*

specifies a prefix for naming the conservative-individual-upper confidence limits. The default prefix is CIU. When you specify the CIUPREFIX= *o-option*, the CLI *o-option* is automatically specified for you.

CLI

outputs individual confidence limits to the OUT= data set. The names of the confidence limits variables are constructed from the original dependent variable names and the prefixes specified in the following *o-options*: LILPREFIX= (default LIL for liberal individual lower), CILPREFIX= (default CIL for conservative individual lower), LIUPREFIX= (default LIU for liberal individual upper), and CIUPREFIX= (default CIU for conservative individual upper). When there are no monotonicity constraints, the liberal and conservative limits are the same.

CLM

outputs mean confidence limits to the OUT= data set. The names of the confidence limits variables are constructed from the original dependent variable names and the prefixes specified in the following *o-options*: LMLPREFIX= (default LML for liberal mean lower), CMLPREFIX= (default CML for conservative mean lower), LMUPREFIX= (default LMU for liberal mean upper), and CMUPREFIX= (default CMU for conservative mean upper). When there are no monotonicity constraints, the liberal and conservative limits are the same.

CMLPREFIX=*name*

CML=*name*

specifies a prefix for naming the conservative-mean-lower confidence limits. The default prefix is CML. When you specify the CMLPREFIX= *o-option*, the CLM *o-option* is automatically specified for you.

CMUPREFIX=*name*

CMU=*name*

specifies a prefix for naming the conservative-mean-upper confidence limits. The default prefix is CMU. When you specify the CMUPREFIX= *o-option*, the CLM *o-option* is automatically specified for you.

COEFFICIENTS

COE

outputs either multiple regression coefficients or raw canonical coefficients to the OUT= data set. If you specify METHOD=CANALS (in the MODEL or PROC TRANSREG statement), then the COEFFICIENTS *o-option* outputs the first *n* canonical variables, where *n* is the value of the NCAN= *a-option* (specified in the MODEL or PROC TRANSREG statement). Otherwise, the COEFFICIENTS *o-option* includes multiple regression coefficients in the OUT= data set. In addition, when you specify the CLASS expansion for any independent variable, the COEFFICIENTS *o-option* also outputs marginal means.

COORDINATES< =*n*>

COO< =*n*>

outputs either ideal point or vector model coordinates for preference mapping to the OUT= data set. When METHOD=CANALS, these coordinates are computed from canonical coefficients; otherwise, the coordinates are computed from multiple regression coefficients. For details, see the section "Point Models" on page 9993.

When ODS Graphics is enabled and vector model coordinates are requested, a plot is produced with points for each row and vectors for each column. If the vectors are plotted based on the actual computed coordinates, then often the vectors are short. A better graphical display is produced when the vectors are stretched. The absolute lengths of each vector can optionally be changed by specifying COORDINATES=*n*. Then the vector coordinates are all multiplied by *n*. Usually, *n* is a value such as 2, 2.5, or 3. The default is 2.5. Specify COORDINATES=1 if you want to see the vectors without any stretching. The relative lengths of the different vectors are important and interpretable, and these are preserved by the stretching.

CPC

outputs canonical point model coordinates to the OUT= data set.

CQC

outputs canonical quadratic point model coordinates to the OUT= data set.

DAPPROXIMATIONS

DAP

outputs the approximations of the transformed dependent variables to the OUT= data set. These are the target values for the optimal transformations. With METHOD=UNIVARIATE and METHOD=MORALS, the dependent variable approximations are the ordinary predicted values from the linear model. The names of the approximation variables are constructed from the ADPREFIX= *o-option* (default A) and the original dependent variable names. For ordinary predicted values, use the PREDICTED *o-option* instead of the DAPPROXIMATIONS *o-option*, since the PREDICTED *o-option* uses a more relevant prefix ("P" instead of "A") and a more relevant variable label suffix ("Predicted Values" instead of "Approximations").

DESIGN< =*n*>

DES< =*n*>

specifies that your primary goal is design matrix coding, not analysis. Specifying the DESIGN *o-option* makes the procedure run faster. The DESIGN *o-option* sets the default method to UNIVARIATE and the default MAXITER= value to zero. It suppresses computing the regression coefficients, unless they are needed for some other option. Furthermore, when the DESIGN *o-option* is specified, the MODEL statement is not required to have an equal sign. When no MODEL statement equal sign is specified, all variables are considered independent variables, all options that require dependent variables are ignored, and the IREPLACE *o-option* is automatically specified for you.

You can use DESIGN=*n* for coding very large data sets, where *n* is the number of observations to code at one time. For example, to code a data set with a large number of observations, you can specify DESIGN=100 or DESIGN=1000 to process the data set in blocks of 100 or 1000 observations. If you specify the DESIGN *o-option* rather than DESIGN=*n*, PROC TRANSREG tries to process all observations at once, which might not work with very large data sets. Specify the NOZEROCONSTANT *a-option* with DESIGN=*n* to ensure that constant variables within blocks are not zeroed. See the sections "Using the DESIGN Output Option" on page 10027 and "Discrete Choice Experiments: DESIGN, NORESTORE, NOZERO" on page 10031 for more information about the DESIGN option.

DEPENDENT=*name*

DEP=*name*

specifies the untransformed dependent variable for OUT= data sets with METHOD=MORALS when there is more than one dependent variable. The default is DEPENDENT=_DEPEND_.

DREPLACE

DRE

replaces the original dependent variables with the transformed dependent variables in the OUT= data set. The names of the transformed variables in the OUT= data set correspond to the names of the original dependent variables in the input data set. By default, both the original dependent variables and the transformed dependent variables (with names constructed from the TDPREFIX= (default T) *o-option* and the original dependent variable names) are included in the OUT= data set.

IAPPROXIMATIONS

IAP

outputs the approximations of the transformed independent variables to the OUT= data set. These are the target values for the optimal transformations. The names of the approximation variables are constructed from the AIPREFIX= *o-option* (default A) and the original independent variable names. When you specify the AIPREFIX= *o-option*, the IAPPROXIMATIONS *o-option* is automatically specified for you. The IAPPROXIMATIONS *o-option* is not valid when METHOD=UNIVARIATE.

IREPLACE

IRE

replaces the original independent variables with the transformed independent variables in the OUT= data set. The names of the transformed variables in the OUT= data set correspond to the names of the original independent variables in the input data set. By default, both the original independent variables and the transformed independent variables (with names constructed from the TIPREFIX= *o-option* (default T) and the original independent variable names) are included in the OUT= data set.

LEVERAGE< =*name* **>**

LEV< =*name* **>**

creates a variable with the specified name in the OUT= data set that contains leverages. Specifying the LEVERAGE *o-option* is equivalent to specifying LEVERAGE=Leverage.

LILPREFIX=*name*

LIL=*name*

specifies a prefix for naming the liberal-individual-lower confidence limits. The default prefix is LIL. When you specify the LILPREFIX= *o-option*, the CLI *o-option* is automatically specified for you.

LIUPREFIX=*name*

LIU=*name*

specifies a prefix for naming the liberal-individual-upper confidence limits. The default prefix is LIU. When you specify the LIUPREFIX= *o-option*, the CLI *o-option* is automatically specified for you.

LMLPREFIX=*name*
LML=*name*
specifies a prefix for naming the liberal-mean-lower confidence limits. The default prefix is LML. When you specify the LMLPREFIX= *o-option*, the CLM *o-option* is automatically specified for you.

LMUPREFIX=*name*
LMU=*name*
specifies a prefix for naming the liberal-mean-upper confidence limits. The default prefix is LMU. When you specify the LMUPREFIX= *o-option*, the CLM *o-option* is automatically specified for you.

MACRO(*keyword=name. . .***)**
MAC(*keyword=name. . .***)**
creates macro variables. Most of the options available within the MACRO *o-option* are rarely needed. By default, PROC TRANSREG creates a macro variable named _TrgInd with a complete list of independent variables created by the procedure. When PROC TRANSREG is being used for design matrix creation prior to running a procedure without a CLASS statement, this macro provides a convenient way to use the results from PROC TRANSREG. For example, a PROC LOGISTIC step that uses a design matrix coded by PROC TRANSREG can use the following MODEL statement:

```
model y=&_trgind;
```

PROC TRANSREG, also by default, creates a macro variable named _TrgIndN, which contains the number of variables in the _TrgInd list. These macro variables can be used in an ARRAY statement as follows:

```
array indvars[&_trgindn] &_trgind;
```

See the sections "Using the DESIGN Output Option" on page 10027 and "Discrete Choice Experiments: DESIGN, NORESTORE, NOZERO" on page 10031 for examples of using the default macro variables.

The available *keywords* are as follows.

DN=*name* specifies the name of a macro variable that contains the number of dependent variables. By default, a macro variable named _TrgDepN is created. This is the number of variables in the DL= list and the number of macro variables created by the DV= and DE= specifications.

IN=*name* specifies the name of a macro variable that contains the number of independent variables. By default, a macro variable named _TrgIndN is created. This is the number of variables in the IL= list and the number of macro variables created by the IV= and IE= specifications.

DL=*name* specifies the name of a macro variable that contains the list of the dependent variables. By default, a macro variable named _TrgDep is created. These are the variable names of the final transformed variables in the OUT= data set. For example, if there are three dependent variables, y1–y3, then _TrgDep contains, by default, Ty1 Ty2 Ty3 (or y1 y2 y3 if you specify the REPLACE *o-option*).

IL=*name* specifies the name of a macro variable that contains the list of the independent variables. By default, a macro variable named _TrgInd is created. These are the variable names of the final transformed variables in the OUT= data set. For example, if there are three independent variables, x1–x3, then _TrgInd contains, by default, Tx1 Tx2 Tx3 (or x1 x2 x3 if you specify the REPLACE *o-option*).

DV=*prefix* specifies a prefix for creating a list of macro variables, each of which contains one dependent variable name. For example, if there are three dependent variables, y1–y3, and you specify **macro(dv=Dep)**, then three macro variables, Dep1, Dep2, and Dep3, are created, containing Ty1, Ty2, and Ty3, respectively (or y1, y2, and y3 if you specify the REPLACE *o-option*). By default, no list is created.

IV=*prefix* specifies a prefix for creating a list of macro variables, each of which contains one independent variable name. For example, if there are three independent variables, x1–x3, and you specify **macro(iv=Ind)**, then three macro variables, Ind1, Ind2, and Ind3, are created, containing Tx1, Tx2, and TX3, respectively (or x1, x2, and x3 if you specify the REPLACE *o-option*). By default, no list is created.

DE=*prefix* specifies a prefix for creating a list of macro variables, each of which contains one dependent variable effect. This list shows the origin of each model term. Each effect consists of two or more parts, and each part consists of a value in 32 columns followed by a blank. For example, if you specify **macro(de=d)**, then a macro variable d1 is created for **identity(y)**. The d1 macro variable is shown next, wrapped onto two lines:

```
4                                TY
IDENTITY                         Y
```

The first part is the number of parts (4), the second part is the transformed variable name, the third part is the transformation, and the last part is the input variable name. By default, no list is created.

IE=*prefix* specifies a prefix for creating a list of macro variables, each of which contains one independent variable effect. This list shows the origin of each model term. Each effect consists of two or more parts, and each part consists of a value in 32 columns followed by a blank. For example, if you specify **macro(ie=I)**, then three macro variables, I1, I2, and I3, are created for **class(x1 | x2)** when both x1 and x2 have values of 1 and 2. These macro variables are shown next, with extra white space removed:

```
5     Tx11      CLASS     x1   1
5     Tx21      CLASS     x2   1
8     Tx11x21   CLASS     x1   1       CLASS     x2   1
```

For CLASS variables, the formatted level appears after the variable name. The first two effects are the main effects, and the last is the interaction term. By default, no list is created.

MEANS
MEA
outputs marginal means for CLASS variable expansions to the OUT= data set.

MEC
outputs multiple regression elliptical point model coordinates to the OUT= data set.

MPC
outputs multiple regression point model coordinates to the OUT= data set.

MQC
outputs multiple regression quadratic point model coordinates to the OUT= data set.

MRC
outputs multiple regression coefficients to the OUT= data set.

MREDUNDANCY
MRE
outputs multiple redundancy analysis coefficients to the OUT= data set.

NORESTOREMISSING
NORESTORE
NOR
specifies that missing values should not be restored when the OUT= data set is created. By default, the coded CLASS variable contains a row of missing values for observations in which the CLASS variable is missing. When you specify the NORESTOREMISSING *o-option*, these observations contain a row of zeros instead. This is useful when PROC TRANSREG is used to code experimental designs for discrete choice models and there is a constant alternative indicated by a missing value.

NOSCORES
NOS
excludes original variables, transformed variables, predicted values, residuals, and scores from the OUT= data set. You can use the NOSCORES *o-option* with various other options to create an OUT= data set that contains only a coefficient partition (for example, a data set consisting entirely of coefficients and coordinates).

PREDICTED
PRE
P
outputs predicted values, which for METHOD=UNIVARIATE and METHOD=MORALS are the ordinary predicted values from the linear model, to the OUT= data set. The names of the predicted values' variables are constructed from the PPREFIX= *o-option* (default P) and the original dependent variable names. When you specify the PPREFIX= *o-option*, the PREDICTED *o-option* is automatically specified for you.

PPREFIX=*name*
PDPREFIX=*name*
PDP=*name*
specifies a prefix for naming the dependent variable predicted values. The default is PPREFIX=P when you specify the PREDICTED *o-option*; otherwise, it is PPREFIX=A. When you specify the PPREFIX= *o-option*, the PREDICTED *o-option* is automatically specified for you. The PPREFIX= *o-option* is the same as the ADPREFIX= *o-option*.

RDPREFIX=*name*

RDP=*name*

specifies a prefix for naming the residual (dependent) variables to the OUT= data set. The default is RDPREFIX=R. When you specify the RDPREFIX= *o-option*, the RESIDUALS *o-option* is automatically specified for you.

REDUNDANCY< =STANDARDIZE | UNSTANDARDIZE >

RED< =STA | UNS >

outputs redundancy variables to the OUT= data set, either standardized or unstandardized. Specifying the REDUNDANCY *o-option* is the same as specifying REDUNDANCY=STANDARDIZE. The results of the REDUNDANCY *o-option* depends on the TSTANDARD= option. You must specify TSTANDARD=Z to get results based on standardized data. The TSTANDARD= option controls how the data that go into the redundancy analysis are scaled, and REDUNDANCY=STANDARDIZE|UNSTANDARDIZE controls how the redundancy variables are scaled. The REDUNDANCY *o-option* is automatically specified for you when you specify the METHOD=REDUNDANCY *a-option*. The RPREFIX= *o-option* specifies a prefix (default Red) for naming the redundancy variables.

REFERENCE=NONE | MISSING | ZERO

REF=NON | MIS | ZER

specifies how reference levels of CLASS variables are to be treated. The options are REFERENCE=NONE, the default, in which reference levels are suppressed; REFERENCE=MISSING, in which reference levels are displayed and output with missing values; and REFERENCE=ZERO, in which reference levels are displayed and output with zeros. You can specify the REFERENCE= option in the PROC TRANSREG, MODEL, or OUTPUT statement, and you can specify it independently for the OUT= data set and the displayed output. When you specify it in only one statement, it sets the option for both the displayed output and the OUT= data set.

REPLACE

REP

is equivalent to specifying both the DREPLACE and the IREPLACE *o-options*.

RESIDUALS

RES

R

outputs the differences between the transformed dependent variables and their predicted values. The names of the residual variables are constructed from the RDPREFIX= *o-option* (default R) and the original dependent variable names.

RPREFIX=*name*

RPR=*name*

provides a prefix for naming the redundancy variables. The default is RPREFIX=Red. When you specify the RPREFIX= *o-option*, the REDUNDANCY *o-option* is automatically specified for you.

TDPREFIX=*name*

TDP=*name*

specifies a prefix for naming the transformed dependent variables. By default, TDPREFIX=T. The TDPREFIX= *o-option* is ignored when you specify the DREPLACE *o-option*.

TIPREFIX=*name*

TIP=*name*

specifies a prefix for naming the transformed independent variables. By default, TIPREFIX=T. The TIPREFIX= *o-option* is ignored when you specify the IREPLACE *o-option*.

WEIGHT Statement

WEIGHT *variable* **;**

When you use a WEIGHT statement, a weighted residual sum of squares is minimized. The WEIGHT statement has no effect on degrees of freedom or number of observations, but the weights affect most other calculations. The observation is used in the analysis only if the value of the WEIGHT statement variable is greater than 0.

Details: TRANSREG Procedure

Model Statement Usage

MODEL < *transform*(*dependents* < / *t-options* >) >
< *transform*(*dependents* < / *t-options* >)... = >
transform(*independents* < / *t-options* >)
< *transform*(*independents* < / *t-options* >)... > < / *a-options* > ;

Here are some examples of model statements:

- linear regression

```
model identity(y) = identity(x);
```

- a linear model with a nonlinear regression function

```
model identity(y) = spline(x / nknots=5);
```

- multiple regression

```
model identity(y) = identity(x1-x5);
```

- multiple regression with nonlinear transformations

```
model spline(y / nknots=3) = spline(x1-x5 / nknots=3);
```

- multiple regression with nonlinear but monotone transformations

```
model mspline(y / nknots=3) = mspline(x1-x5 / nknots=3);
```

- multivariate multiple regression

```
model identity(y1-y4) = identity(x1-x5);
```

- canonical correlation

```
model identity(y1-y4) = identity(x1-x5) / method=canals;
```

- redundancy analysis

```
model identity(y1-y4) = identity(x1-x5) / method=redundancy;
```

- preference mapping, vector model (Carroll 1972)

```
model identity(Attrib1-Attrib3) = identity(Dim1-Dim2);
```

- preference mapping, ideal point model (Carroll 1972)

```
model identity(Attrib1-Attrib3) = point(Dim1-Dim2);
```

- preference mapping, ideal point model, elliptical (Carroll 1972)

```
model identity(Attrib1-Attrib3) = epoint(Dim1-Dim2);
```

- preference mapping, ideal point model, quadratic (Carroll 1972)

```
model identity(Attrib1-Attrib3) = qpoint(Dim1-Dim2);
```

- metric conjoint analysis

```
model identity(Subj1-Subj50) = class(a b c d e f / zero=sum);
```

- nonmetric conjoint analysis

```
model monotone(Subj1-Subj50) = class(a b c d e f / zero=sum);
```

- main effects, two-way interaction

```
model identity(y) = class(a|b);
```

- less-than-full-rank model—main effects and two-way interaction are constrained to sum to zero

```
model identity(y) = class(a|b / zero=sum);
```

- main effects and all two-way interactions

```
model identity(y) = class(a|b|c@2);
```

- main effects and all two- and three-way interactions

```
model identity(y) = class(a|b|c);
```

- main effects and only the b*c two-way interaction

```
model identity(y) = class(a b c b*c);
```

- seven main effects, three two-way interactions

```
model identity(y) = class(a b c d e f g a*b a*c a*d);
```

- deviations-from-means (effects or $(1, 0, -1)$) coding, with an a reference level of '1' and a b reference level of '2'

```
model identity(y) = class(a|b / deviations zero='1' '2');
```

- cell-means coding (implicit intercept)

```
model identity(y) = class(a*b / zero=none);
```

- reference cell model

```
model identity(y) = class(a|b / zero='1' '1');
```

- reference line with change in line parameters

```
model identity(y) = class(a) | identity(x);
```

- reference curve with change in curve parameters

```
model identity(y) = class(a) | spline(x);
```

- separate curves and intercepts

```
model identity(y) = class(a / zero=none) | spline(x);
```

- quantitative effects with interaction

```
model identity(y) = identity(x1 | x2);
```

- separate quantitative effects with interaction within each cell

```
model identity(y) = class(a * b / zero=none) | identity(x1 | x2);
```

Box-Cox Transformations

Box-Cox (1964) transformations are used to find potentially nonlinear transformations of a dependent variable. The Box-Cox transformation has the form

$$\begin{array}{ll} (y^{\lambda} - 1)/\lambda & \lambda \neq 0 \\ \log(y) & \lambda = 0 \end{array}$$

This family of transformations of the positive dependent variable y is controlled by the parameter λ. Transformations linearly related to square root, inverse, quadratic, cubic, and so on are all special cases. The limit as λ approaches 0 is the log transformation. More generally, Box-Cox transformations of the following form can be fit:

$$\begin{array}{ll} ((y + c)^{\lambda} - 1)/(\lambda g) & \lambda \neq 0 \\ \log(y + c)/g & \lambda = 0 \end{array}$$

By default, $c = 0$. The parameter c can be used to rescale y so that it is strictly positive. By default, $g = 1$. Alternatively, g can be $\dot{y}^{\lambda-1}$, where $\dot{y}$ is the geometric mean of y.

The BOXCOX transformation in PROC TRANSREG can be used to perform a Box-Cox transformation of the dependent variable. You can specify a list of power parameters by using the LAMBDA= *t-option*. By default, LAMBDA=–3 TO 3 BY 0.25. The procedure chooses the optimal power parameter by using a maximum likelihood criterion (Draper and Smith 1981, pp. 225–226). You can specify the PARAMETER=*c* transformation option when you want to shift the values of y, usually to avoid negatives. To divide by $\dot{y}^{\lambda-1}$, specify the GEOMETRICMEAN *t-option*.

Here are three examples of using the LAMBDA= *t-option*:

```
model BoxCox(y / lambda=0) = identity(x1-x5);
model BoxCox(y / lambda=-2 to 2 by 0.1) = identity(x1-x5);
model BoxCox(y) = identity(x1-x5);
```

Here is the first example:

```
model BoxCox(y / lambda=0) = identity(x1-x5);
```

LAMBDA=0 specifies a Box-Cox transformation with a power parameter of 0. Since a single value of 0 was specified for LAMBDA=, there is no difference between the following models:

```
model BoxCox(y / lambda=0) = identity(x1-x5);
model log(y) = identity(x1-x5);
```

Here is the second example:

```
model BoxCox(y / lambda=-2 to 2 by 0.1) = identity(x1-x5);
```

LAMBDA= specifies a list of power parameters. PROC TRANSREG tries each power parameter in the list and picks the best transformation. A maximum likelihood approach (Draper and Smith 1981, pp. 225–226) is used. With Box-Cox transformations, PROC TRANSREG finds the transformation before the usual iterations begin. Note that this is quite different from PROC TRANSREG's usual approach of iteratively finding optimal transformations with ordinary and alternating least squares. It is analogous to SMOOTH and PBSPLINE, which also find transformations before the iterations begin based on a criterion other than least squares.

Here is the third example:

```
model BoxCox(y) = identity(x1-x5);
```

The default LAMBDA= list of –3 TO 3 BY 0.25 is used.

The procedure prints the optimal power parameter, a confidence interval on the power parameter (based on the ALPHA= *t-option*), a "convenient" power parameter (selected from the CLL= *t-option* list), and the log likelihood for each power parameter tried (see Example 120.2).

To illustrate how Box-Cox transformations work, data were generated from the model

$$y = e^{x+\epsilon}$$

where $\epsilon \sim \mathrm{N}(0, 1)$. The transformed data can be fit with a linear model

$$\log(y) = x + \epsilon$$

The following statements produce Figure 120.14 through Figure 120.15:

```
title 'Basic Box-Cox Example';

data x;
   do x = 1 to 8 by 0.025;
      y = exp(x + normal(7));
      output;
   end;
run;

ods graphics on;

title2 'Default Options';

proc transreg data=x test;
   model BoxCox(y) = identity(x);
run;
```

Figure 120.14 Basic Box-Cox Example, Default Output

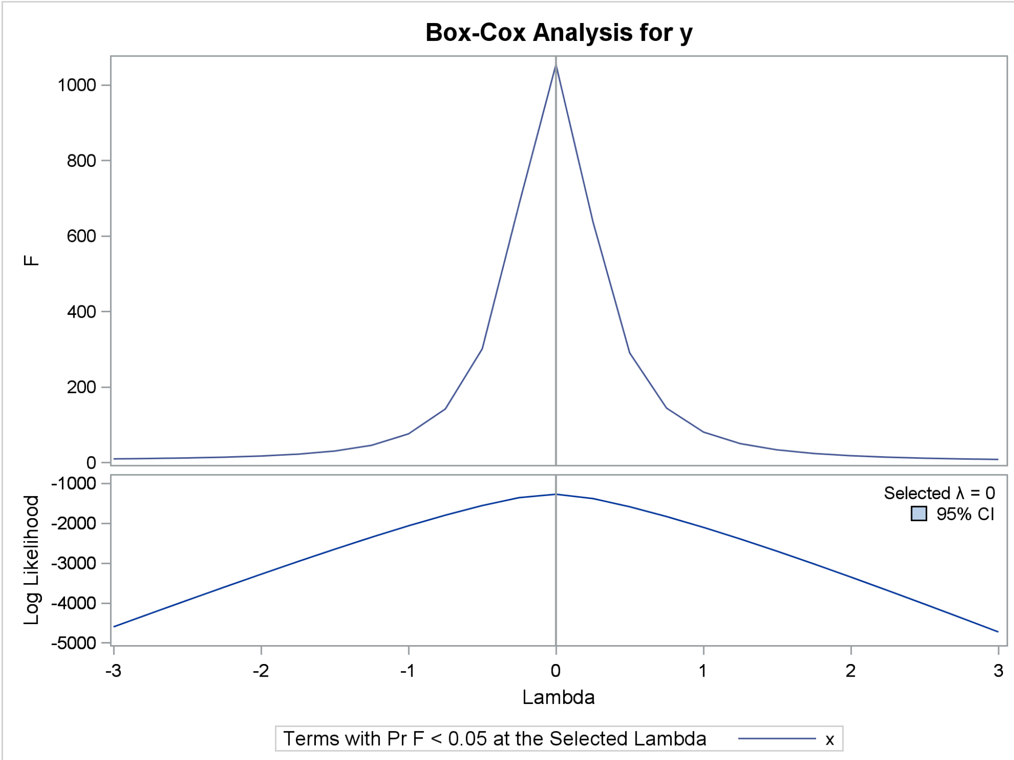


Figure 120.14 shows that PROC TRANSREG correctly selects the log transformation $\lambda = 0$, with a narrow confidence interval. The $F = t^2$ plot shows that F is at its largest in the vicinity of the optimal Box-Cox transformation.

The rest of the output, which contains the ANOVA results, is shown in Figure 120.15.

Figure 120.15 Basic Box-Cox Example, Default Output

Dependent Variable BoxCox(y)

Number of Observations Read	281
Number of Observations Used	281

The TRANSREG Procedure Hypothesis Tests for BoxCox(y)

Univariate ANOVA Table Based on the Usual Degrees of Freedom

Source	DF	Sum of Squares	Mean Square	F Value	Liberal p
Model	1	1145.884	1145.884	1053.66	>= <.0001
Error	279	303.421	1.088		
Corrected Total	280	1449.305			

The above statistics are not adjusted for the fact that the dependent variable was transformed and so are generally liberal.

Figure 120.15 *continued*

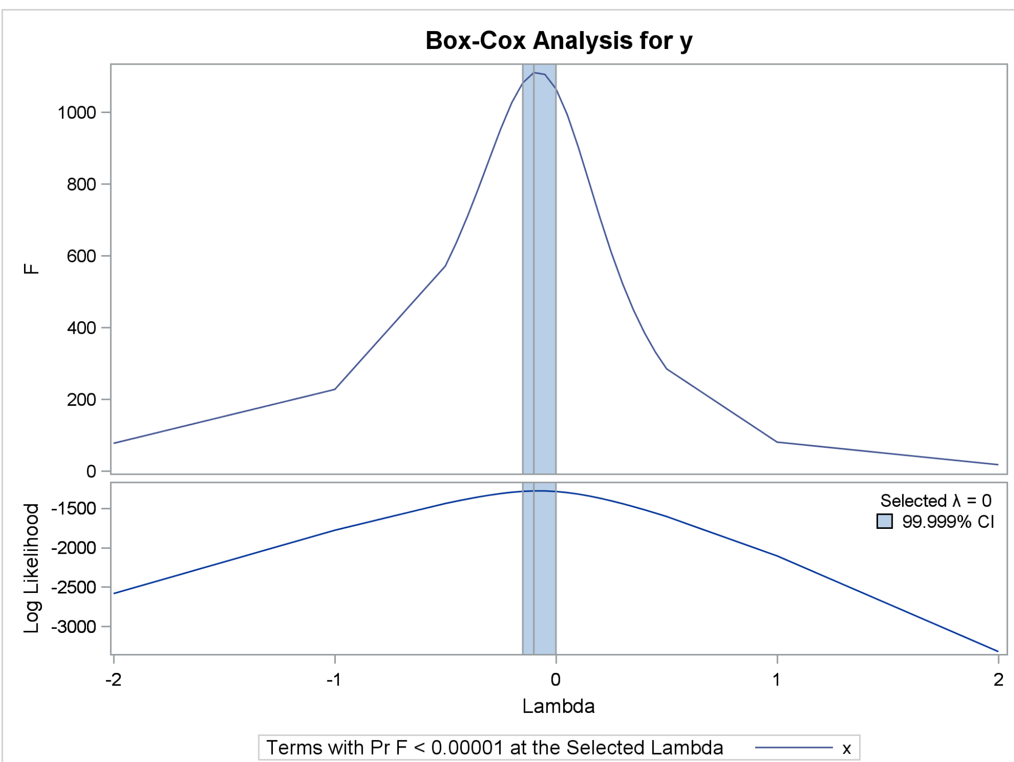
Root MSE	1.04285	**R-Square**	0.7906
Dependent Mean	4.49653	**Adj R-Sq**	0.7899
Coeff Var	23.19225	**Lambda**	0.0000

This next example uses several options. The LAMBDA= *t-option* specifies power parameters sparsely from –2 to –0.5 and 0.5 to 2 just to get the general shape of the log-likelihood function in that region. Between –0.5 and 0.5, more power parameters are tried. The CONVENIENT *t-option* is specified so that if a power parameter like $\lambda = 1$ or $\lambda = 0$ is found in the confidence interval, it is used instead of the optimal power parameter. PARAMETER=2 is specified to add 2 to each *y* before performing the transformations. ALPHA=0.00001 specifies a wide confidence interval.

These next statements perform the Box-Cox analysis and produce Figure 120.16 and Figure 120.17:

```
title2 'Several Options Demonstrated';

proc transreg data=x ss2 details
              plots=(transformation(dependent) scatter
                    observedbypredicted);
   model BoxCox(y / lambda=-2 -1 -0.5 to 0.5 by 0.05 1 2
                    convenient parameter=2 alpha=0.00001) =
         identity(x);
run;
```

Figure 120.16 Basic Box-Cox Example, Several Options Demonstrated



The results in Figure 120.16 and Figure 120.17 show that the optimal power parameter is –0.1, but 0 is in the confidence interval, and hence a log transformation is chosen. The actual Box-Cox transformation, the original scatter plot, and observed by predicted values plot are shown in Figure 120.17.

Figure 120.17 Basic Box-Cox Example, Several Options Demonstrated

Dependent Variable BoxCox(y)

Number of Observations Read	281
Number of Observations Used	281

Model Statement Specification Details

Type	DF	Variable	Description	Value
Dep	1	BoxCox(y)	Lambda Used	0
			Lambda	-0.1
			Log Likelihood	-1280.1
			Conv. Lambda	0
			Conv. Lambda LL	-1287.7
			CI Limit	-1289.9
			Alpha	0.00001
			Parameter	2
			Options	Convenient Lambda Used
Ind	1	Identity(x)	DF	1

The TRANSREG Procedure Hypothesis Tests for BoxCox(y)

Univariate ANOVA Table Based on the Usual Degrees of Freedom

Source	DF	Sum of Squares	Mean Square	F Value	Liberal p
Model	1	999.438	999.4381	1064.82	>= <.0001
Error	279	261.868	0.9386		
Corrected Total	280	1261.306			

The above statistics are not adjusted for the fact that the dependent variable was transformed and so are generally liberal.

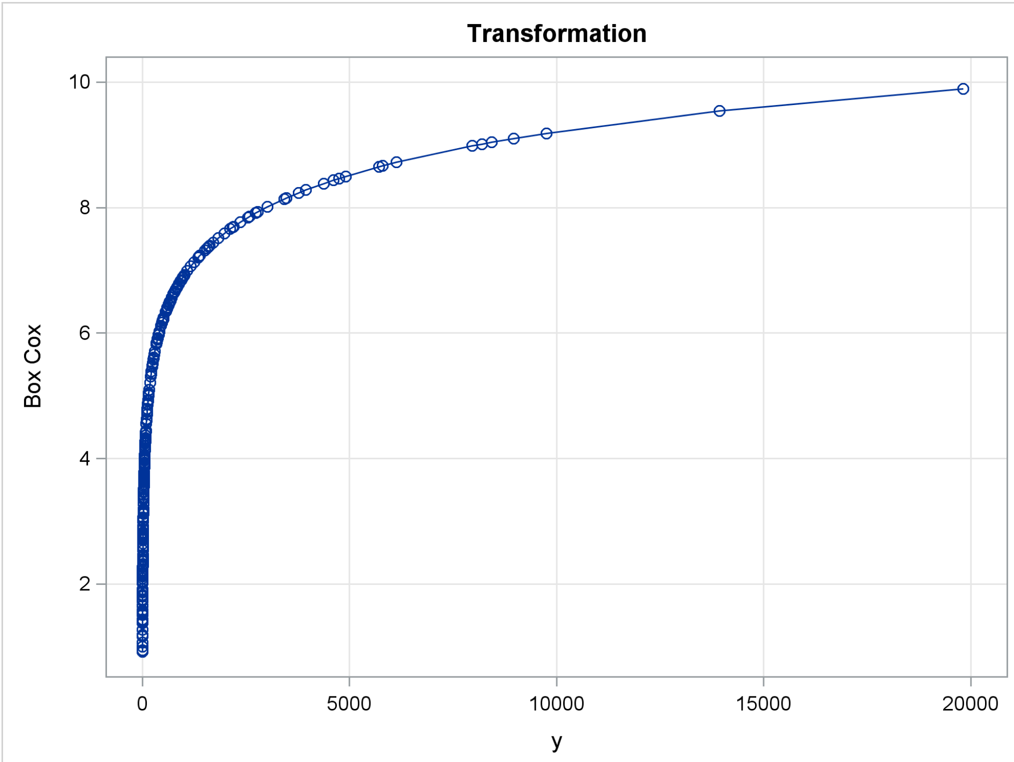
Root MSE	0.96881	**R-Square**	0.7924
Dependent Mean	4.61429	**Adj R-Sq**	0.7916
Coeff Var	20.99591	**Lambda**	0.0000

Univariate Regression Table Based on the Usual Degrees of Freedom

Variable	DF	Coefficient	Type II Sum of Squares	Mean Square	F Value	Liberal p
Intercept	1	0.42939328	8.746	8.746	9.32	>= 0.0025
Identity(x)	1	0.92997620	999.438	999.438	1064.82	>= <.0001

The above statistics are not adjusted for the fact that the dependent variable was transformed and so are generally liberal.

Figure 120.17 *continued*



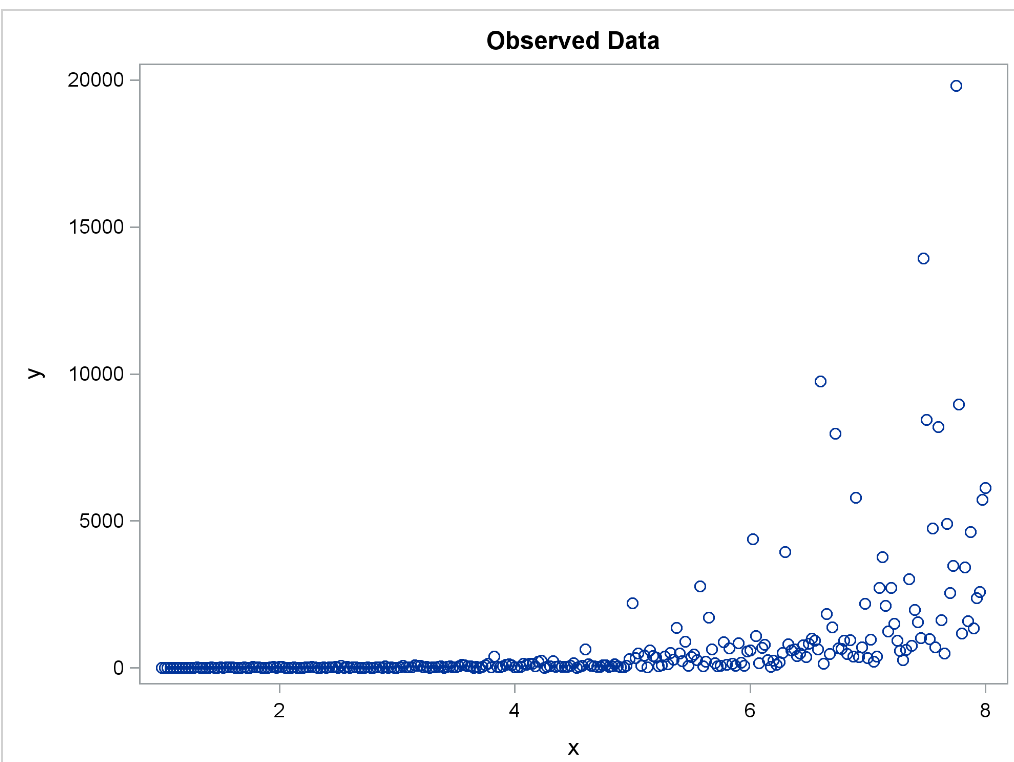
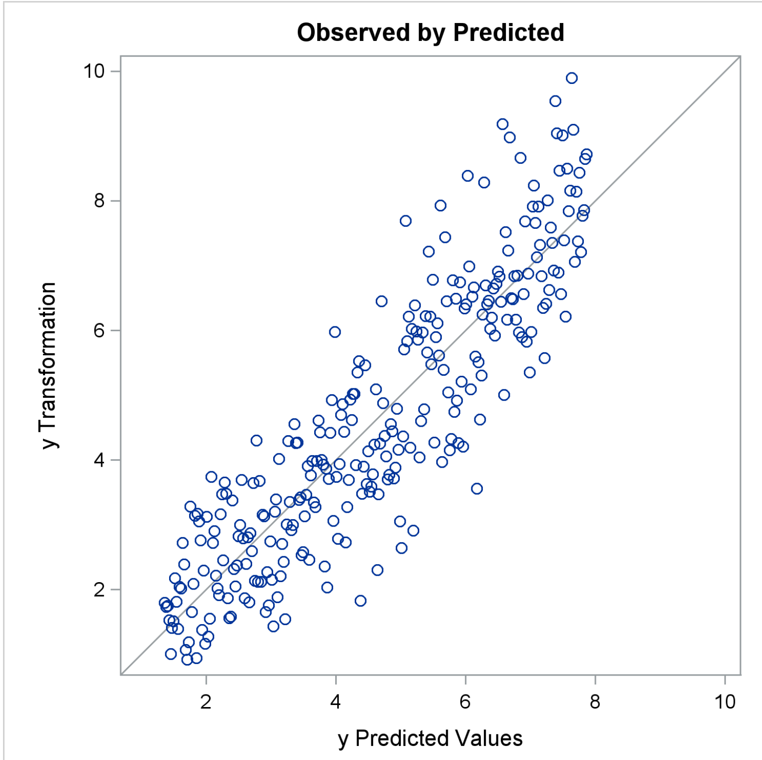


Figure 120.17 *continued*



The next example shows how to find a Box-Cox transformation without an independent variable. This seeks to normalize the univariate histogram. This example generates 500 random observations from a lognormal distribution. In addition, a constant variable z is created that is all zero. This is because PROC TRANSREG requires some independent variable to be specified, even if it is constant. Two options are specified in the PROC TRANSREG statement. MAXITER=0 is specified because the Box-Cox transformation is performed before any iterations are begun. No iterations are needed since no other work is required. The NOZEROCONSTANT *a-option* (which can be abbreviated NOZ) is specified so that PROC TRANSREG does not print any warnings when it encounters the constant independent variable. The MODEL statement asks for a Box-Cox transformation of y and an IDENTITY transformation (which does nothing) of the constant variable z. Finally, PROC UNIVARIATE is run to show a histogram of the original variable y, and the Box-Cox transformation, Ty. The following statements fit the univariate Box-Cox model and produce Figure 120.18:

```
title 'Univariate Box-Cox';

data x;
   call streaminit(17);
   z = 0;
   do i = 1 to 500;
      y = rand('lognormal');
      output;
   end;
run;

proc transreg maxiter=0 nozeroconstant;
   model BoxCox(y) = identity(z);
   output;
run;

proc univariate noprint;
   histogram y ty;
run;
```

The PROC TRANSREG results in Figure 120.18 show that zero is chosen for lambda, so a log transformation is chosen. The first histogram shows that the original data are skewed, but a log transformation makes the data appear much more nearly normal.

Figure 120.18 Box-Cox with No Independent Variable

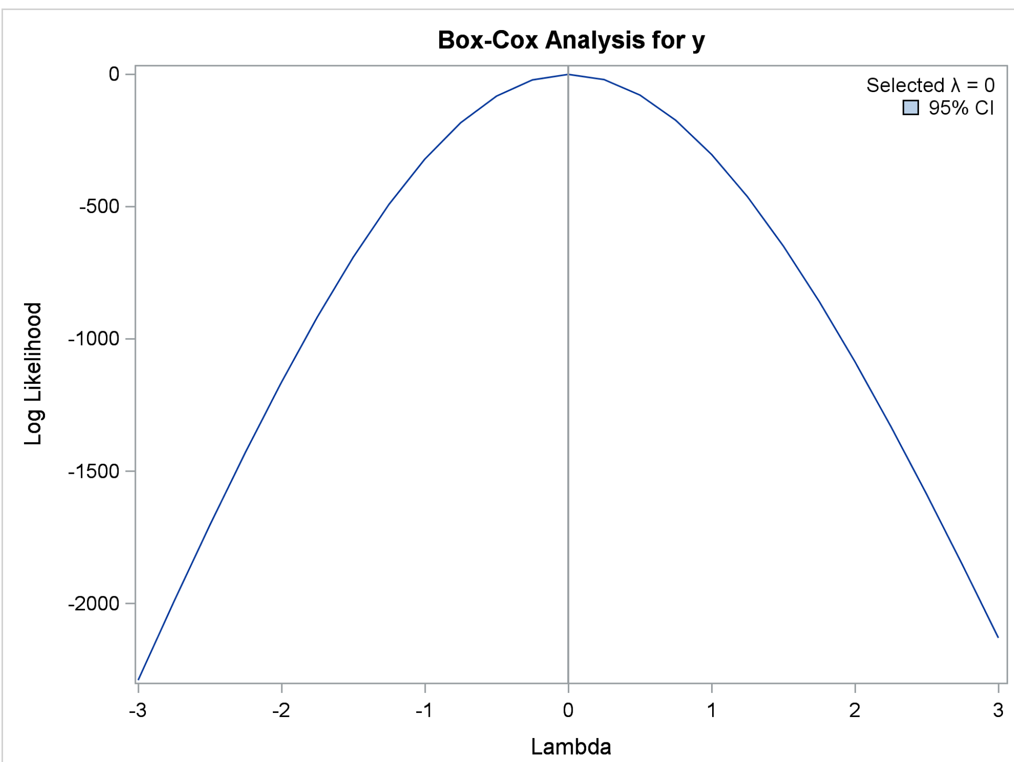
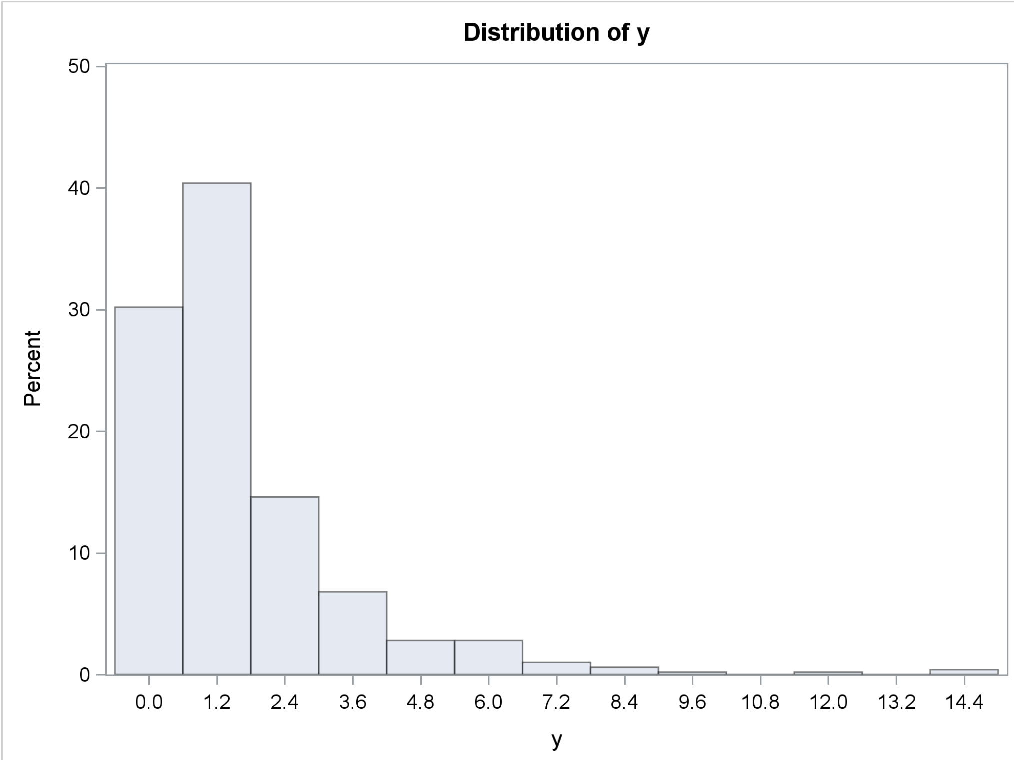
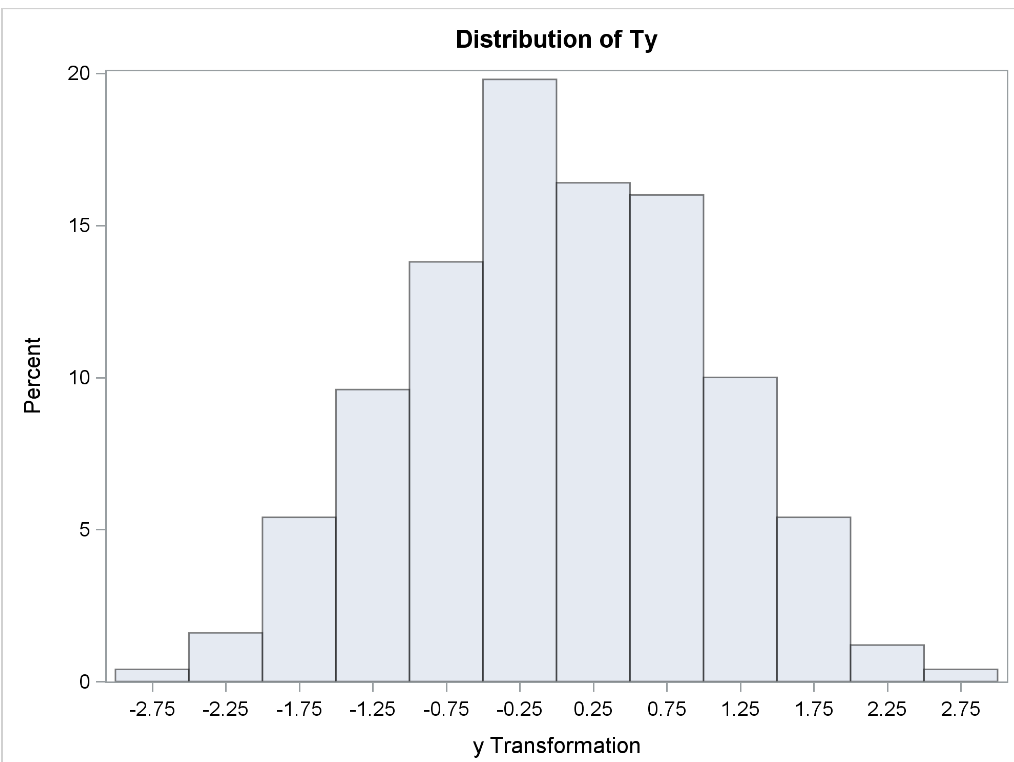


Figure 120.18 *continued*





Using Splines and Knots

This section illustrates some properties of splines. *Splines* are curves, and they are usually required to be continuous and smooth. Splines are usually defined as piecewise polynomials of degree n with function values and first $n - 1$ derivatives that agree at the points where they join. The abscissa or X-axis values of the join points are called *knots*. The term "spline" is also used for polynomials (splines with no knots) and piecewise polynomials with more than one discontinuous derivative. Splines with no knots are generally smoother than splines with knots, which are generally smoother than splines with multiple discontinuous derivatives. Splines with few knots are generally smoother than splines with many knots; however, increasing the number of knots usually increases the fit of the spline function to the data. Knots give the curve freedom to bend to more closely follow the data. See Smith (1979) for an excellent introduction to splines.

In this section, an artificial data set is created with a variable y that is a discontinuous function of x. (See Figure 120.20.) Notice that the function has four unconnected parts, each of which is a curve. Notice too that there is an overall quadratic trend—that is, ignoring the shapes of the individual curves, at first the y values tend to decrease as x increases, then y values tend to increase. While these artificial data are clearly not realistic, their distinct pattern helps illustrate how splines work. The following statements create the data set, fit a simple linear regression model, and produce Figure 120.19 through Figure 120.20:

```
title 'An Illustration of Splines and Knots';

* Create in y a discontinuous function of x.;

data a;
   x = -0.000001;
   do i = 0 to 199;
      if mod(i, 50) = 0 then do;
         c = ((x / 2) - 5)**2;
         if i = 150 then c = c + 5;
         y = c;
      end;
      x = x + 0.1;
      y = y - sin(x - c);
      output;
   end;
run;

ods graphics on;

title2 'A Linear Regression Fit';
proc transreg data=a plots=scatter rsquare;
   model identity(y) = identity(x);
run;
```

The R square for the linear regression is 0.1006. The linear fit results in Figure 120.19 show the predicted values of y given x. It can clearly be seen in Figure 120.19 that the linear regression model is not appropriate for these data.

Figure 120.19 A Linear Regression Fit

An Illustration of Splines and Knots
A Linear Regression Fit

The TRANSREG Procedure

The TRANSREG Procedure Hypothesis Tests for Identity(y)

R-Square	0.1006

Figure 120.19 *continued*

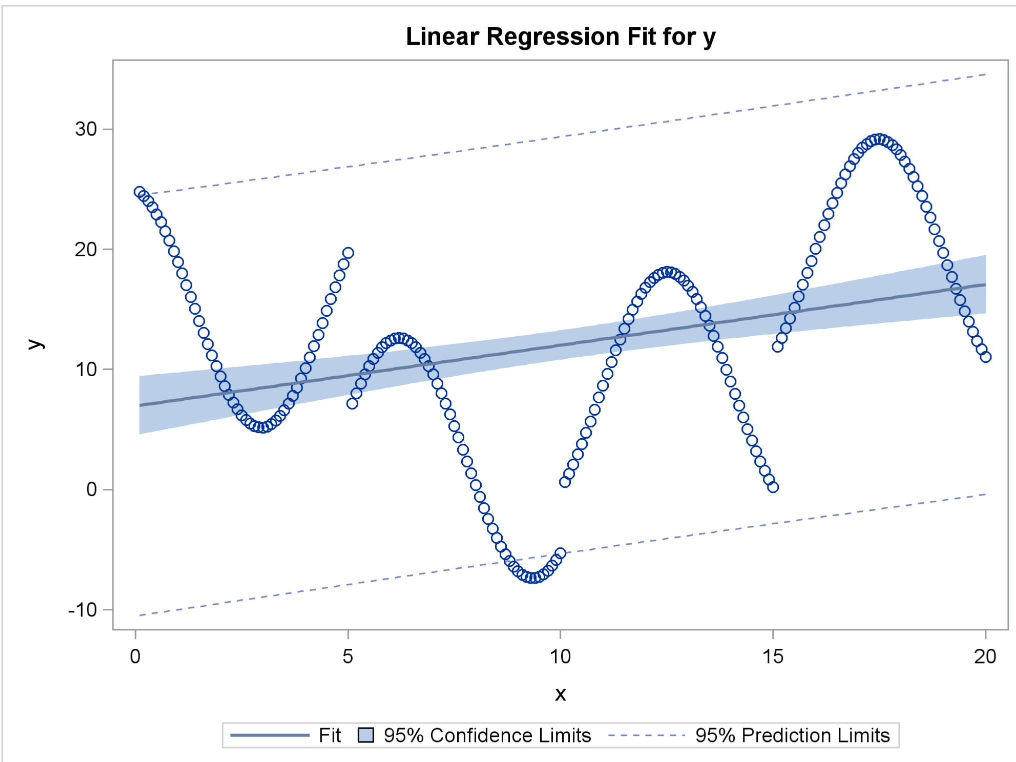
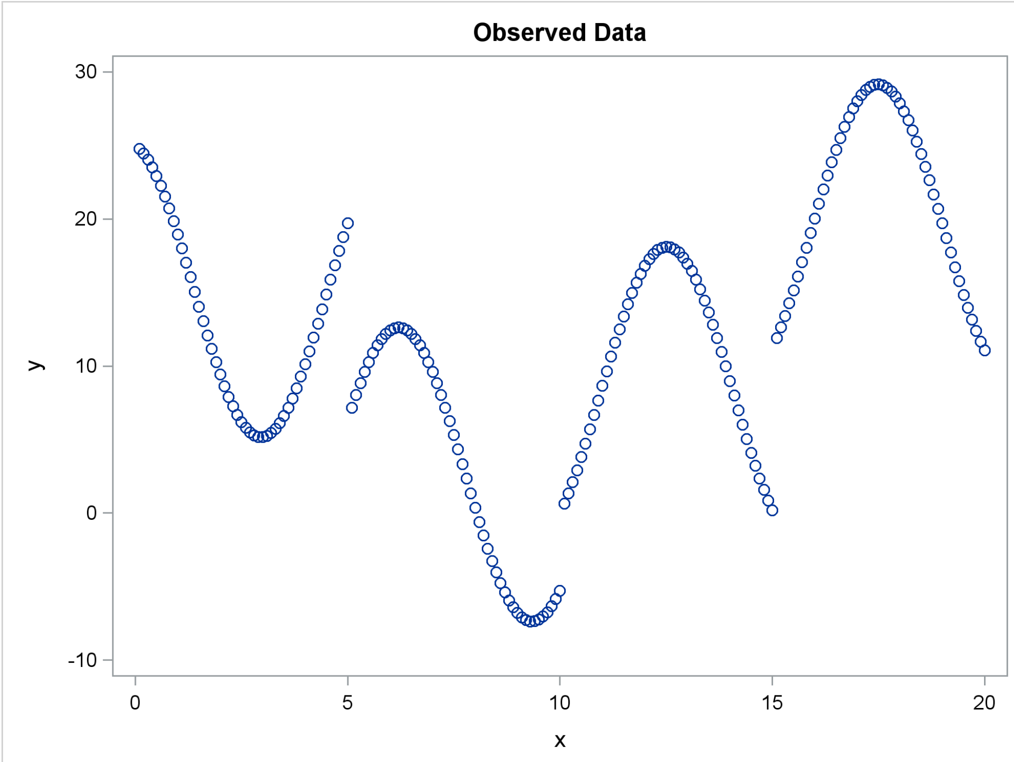


Figure 120.20 The Original Scatter Plot



The next PROC TRANSREG step finds a degree-two spline transformation with no knots, which is a quadratic polynomial. The spline is a weighted sum of a single constant, a single straight line, and a single quadratic curve. The following statements perform the quadratic analysis and produce Figure 120.21:

```
title2 'A Quadratic Polynomial Fit';

proc transreg data=A;
   model identity(y)=spline(x / degree=2);
run;
```

The R square in Figure 120.21 increases from 0.10061, which is the linear fit value from before, to 0.40720. The plot shows that the quadratic regression function does not fit any of the individual curves well, but it does follow the overall trend in the data. Since the overall trend is quadratic, if you were to fit a degree-three spline with no knots (not shown) would increase R square by only a small amount.

Figure 120.21 A Quadratic Polynomial Fit

An Illustration of Splines and Knots
A Quadratic Polynomial Fit

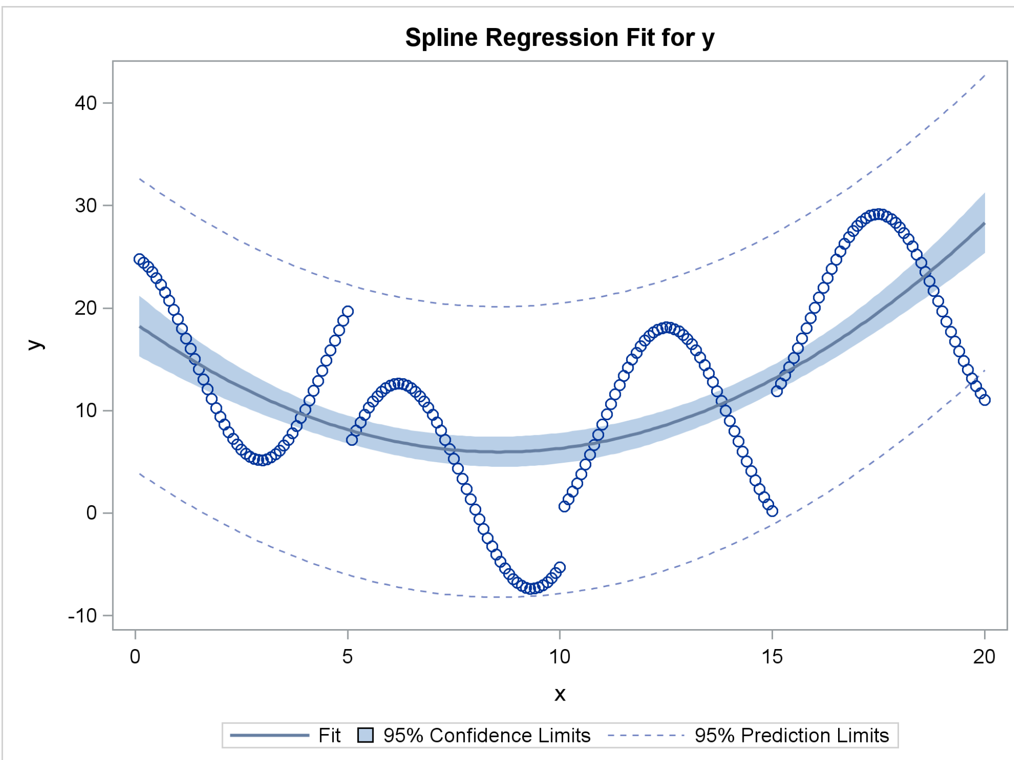
The TRANSREG Procedure

TRANSREG MORALS Algorithm Iteration History for Identity(y)

Iteration Number	Average Change	Maximum Change	R-Square	Criterion Change	Note
1	0.82127	2.77121	0.10061		
2	0.00000	0.00000	0.40720	0.30659	Converged

Algorithm converged.

Figure 120.21 *continued*



The next step uses the default degree of three, for a piecewise cubic polynomial, and requests knots at the known break points, x=5, 10, and 15. This requests a spline that is continuous, has continuous first and second derivatives, and has a third derivative that is discontinuous at 5, 10, and 15. The spline is a weighted sum of a single constant, a single straight line, a single quadratic curve, a cubic curve for the portion of x less than 5, a different cubic curve for the portion of x between 5 and 10, a different cubic curve for the portion of x between 10 and 15, and another cubic curve for the portion of x greater than 15. The following statements fit the spline model and produce Figure 120.22:

```
title2 'A Cubic Spline Fit with Knots at X=5, 10, 15';

proc transreg data=a;
   model identity(y) = spline(x / knots=5 10 15);
run;
```

The new R square in Figure 120.22 is 0.61730. The plot shows that the spline is less smooth than the quadratic polynomial and follows the data more closely than the quadratic polynomial.

Figure 120.22 A Cubic Spline Fit

An Illustration of Splines and Knots
A Cubic Spline Fit with Knots at X=5, 10, 15

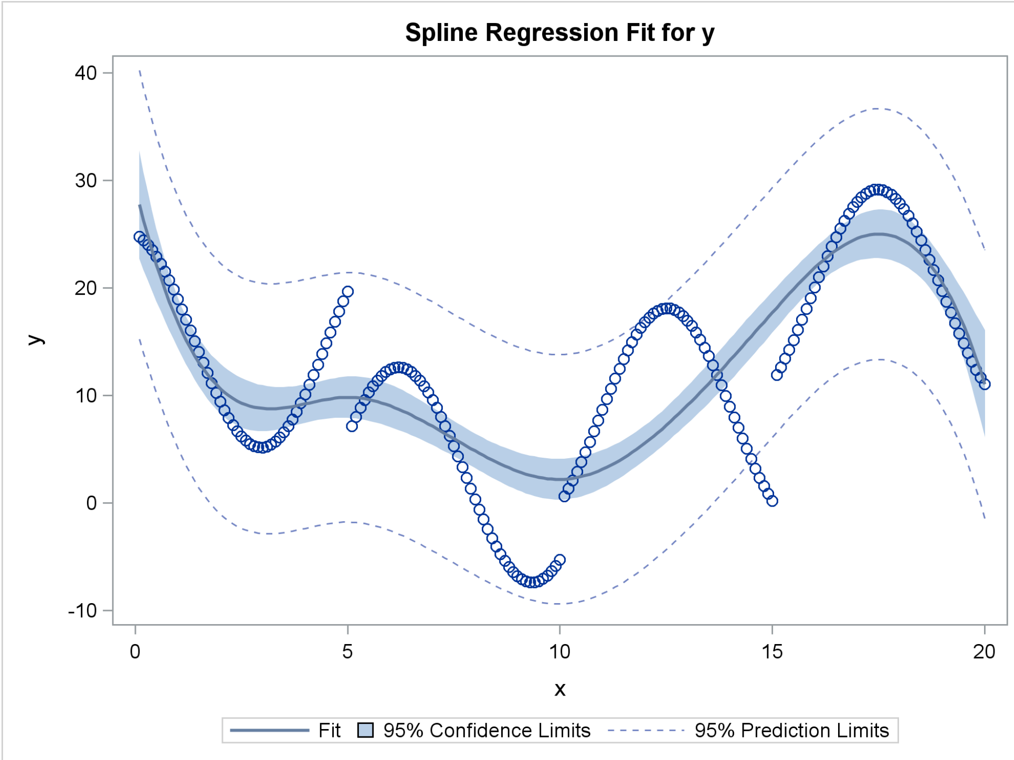
The TRANSREG Procedure

TRANSREG MORALS Algorithm Iteration History for Identity(y)

Iteration Number	Average Change	Maximum Change	R-Square	Criterion Change	Note
1	0.85367	3.88449	0.10061		
2	0.00000	0.00000	0.61730	0.51670	Converged

Algorithm converged.

Figure 120.22 *continued*



The same model could be fit with a DATA step and PROC REG, as follows:

```
data b;             /* A is the data set used by transreg */
   set a(keep=x y);
   x1=x;                         /* x                       */
   x2=x**2;                      /* x squared               */
   x3=x**3;                      /* x cubed                 */
   x4=(x> 5)*((x-5)**3);         /* change in x**3 after  5 */
   x5=(x>10)*((x-10)**3);        /* change in x**3 after 10 */
   x6=(x>15)*((x-15)**3);        /* change in x**3 after 15 */
run;

proc reg;
   model y=x1-x6;
run; quit;
```

The output from these previous statements is not displayed. The assignment statements and comments show how you can construct terms that can be used to fit the same model.

In the next step, each knot is repeated three times, so the first, second, and third derivatives are discontinuous at x=5, 10, and 15, but the spline is continuous at the knots. The spline is a weighted sum of the following:

- a single constant
- a line for the portion of x less than 5
- a quadratic curve for the portion of x less than 5
- a cubic curve for the portion of x less than 5
- a different line for the portion of x between 5 and 10
- a different quadratic curve for the portion of x between 5 and 10
- a different cubic curve for the portion of x between 5 and 10
- a different line for the portion of x between 10 and 15
- a different quadratic curve for the portion of x between 10 and 15
- a different cubic curve for the portion of x between 10 and 15
- another line for the portion of x greater than 15
- another quadratic curve for the portion of x greater than 15
- another cubic curve for the portion of x greater than 15

The spline is continuous since there is not a separate constant or separate intercept in the formula for the spline for each knot. The following statements perform this analysis and produce Figure 120.23:

```
title3 'First - Third Derivatives Discontinuous at X=5, 10, 15';

proc transreg data=a;
   model identity(y) = spline(x / knots=5 5 5 10 10 10 15 15 15);
run;
```

Now the R square in Figure 120.23 is 0.95542, and the spline closely follows the data, except at the knots.

Figure 120.23 Spline with Discontinuous Derivatives

An Illustration of Splines and Knots
A Cubic Spline Fit with Knots at X=5, 10, 15
First - Third Derivatives Discontinuous at X=5, 10, 15

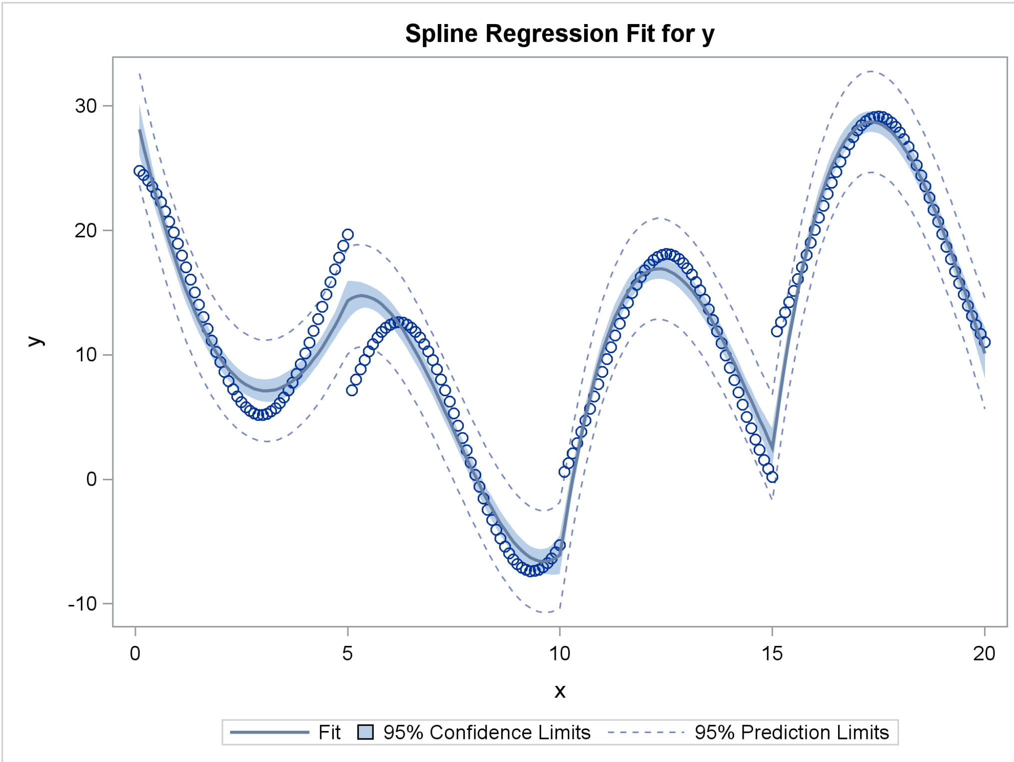
The TRANSREG Procedure

TRANSREG MORALS Algorithm Iteration History for Identity(y)

Iteration Number	Average Change	Maximum Change	R-Square	Criterion Change	Note
1	0.92492	3.50038	0.10061		
2	0.00000	0.00000	0.95542	0.85481	Converged

Algorithm converged.

Figure 120.23 *continued*



The same model could be fit with a DATA step and PROC REG, as follows:

```
data b;
   set a(keep=x y);
   x1=x;                          /* x                      */
   x2=x**2;                       /* x squared              */
   x3=x**3;                       /* x cubed                */
   x4=(x>5)    * (x- 5);          /* change in x    after  5 */
   x5=(x>10)   * (x-10);          /* change in x    after 10 */
   x6=(x>15)   * (x-15);          /* change in x    after 15 */
   x7=(x>5)    * ((x-5)**2);      /* change in x**2 after  5 */
   x8=(x>10)   * ((x-10)**2);     /* change in x**2 after 10 */
   x9=(x>15)   * ((x-15)**2);     /* change in x**2 after 15 */
   x10=(x>5)   * ((x-5)**3);      /* change in x**3 after  5 */
   x11=(x>10)  * ((x-10)**3);     /* change in x**3 after 10 */
   x12=(x>15)  * ((x-15)**3);     /* change in x**3 after 15 */
run;

proc reg;
   model y=x1-x12;
run; quit;
```

The output from these previous statements is not displayed. The assignment statements and comments show how you can construct terms that can be used to fit the same model.

Each knot is repeated four times in the next step. Now the spline function is discontinuous at the knots, and it can follow the data more closely. The following statements perform this analysis and produce Figure 120.24:

```
title3 'Discontinuous Function and Derivatives';

proc transreg data=a;
   model identity(y) = spline(x / knots=5 5 5 5 10 10 10 10
                                        15 15 15 15);
run;
```

Now the R square in Figure 120.24 is 0.99254. In this step, each separate curve is approximated by a cubic polynomial (with no knots within the separate polynomials). (Note, however, that the separate functions are connected in the plot, because PROC TRANSREG cannot currently produce separate functions for a model like this. Usually, you would use a CLASS variable to get separate functions.)

Figure 120.24 Discontinuous Spline Fit

An Illustration of Splines and Knots
A Cubic Spline Fit with Knots at X=5, 10, 15
Discontinuous Function and Derivatives

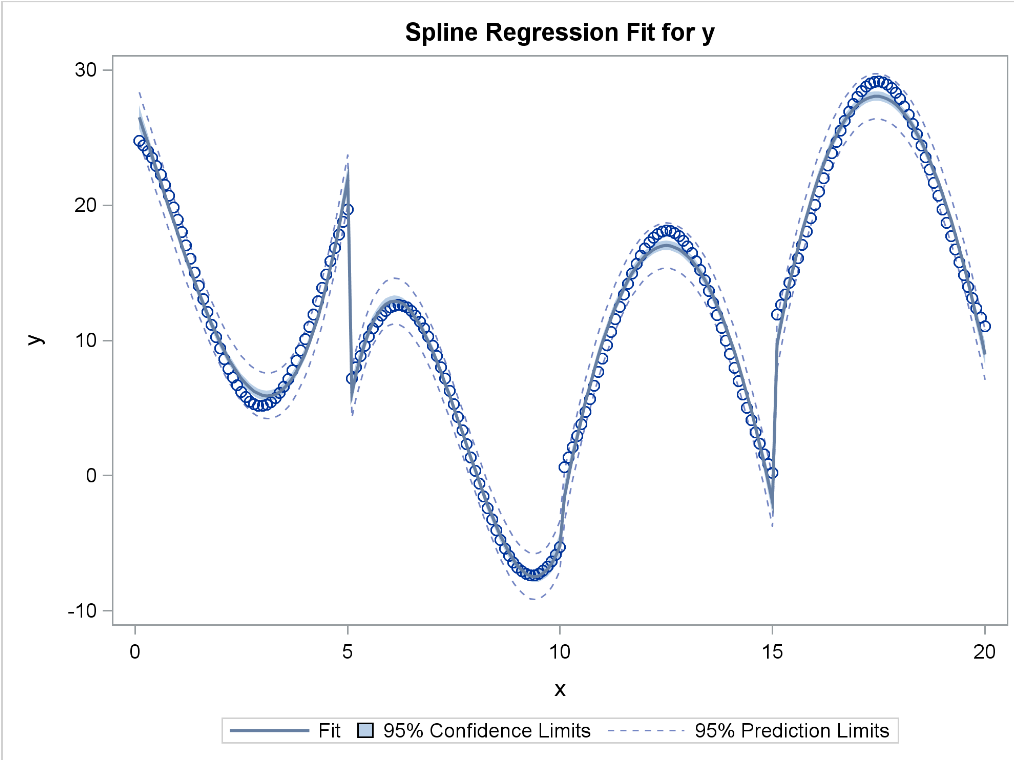
The TRANSREG Procedure

TRANSREG MORALS Algorithm Iteration History for Identity(y)

Iteration Number	Average Change	Maximum Change	R-Square	Criterion Change	Note
1	0.90271	3.29184	0.10061		
2	0.00000	0.00000	0.99254	0.89193	Converged

Algorithm converged.

Figure 120.24 *continued*



To solve this problem with a DATA step and PROC REG, you would need to create all of the variables in the preceding DATA step (the B data set for the piecewise polynomial with discontinuous third derivatives), plus the following three variables:

```
x13=(x >  5);   /* intercept change after  5 */
x14=(x > 10);   /* intercept change after 10 */
x15=(x > 15);   /* intercept change after 15 */
```

The next two examples use the NKNOTS= *t-option* to specify the number of knots but not their location. NKNOTS=4 places knots at the quintiles, whereas NKNOTS=9 places knots at the deciles. The spline and its first two derivatives are continuous. The following statements produce Figure 120.25 and Figure 120.26:

```
title3 'Four Knots';

proc transreg data=a;
   model identity(y) = spline(x / nknots=4);
run;

title3 'Nine Knots';

proc transreg data=a;
   model identity(y) = spline(x / nknots=9);
run;
```

The R-square values displayed in Figure 120.25 and Figure 120.26 are 0.74450 and 0.95256, respectively. Even though the knots are not optimally placed, the spline can closely follow the data with NKNOTS=9.

Figure 120.25 Spline Fit with Knots at the Quintiles

An Illustration of Splines and Knots
A Cubic Spline Fit with Knots at X=5, 10, 15
Four Knots

The TRANSREG Procedure

TRANSREG MORALS Algorithm Iteration History for Identity(y)

Iteration Number	Average Change	Maximum Change	R-Square	Criterion Change	Note
1	0.90305	4.46027	0.10061		
2	0.00000	0.00000	0.74450	0.64389	Converged

Algorithm converged.

Figure 120.25 *continued*

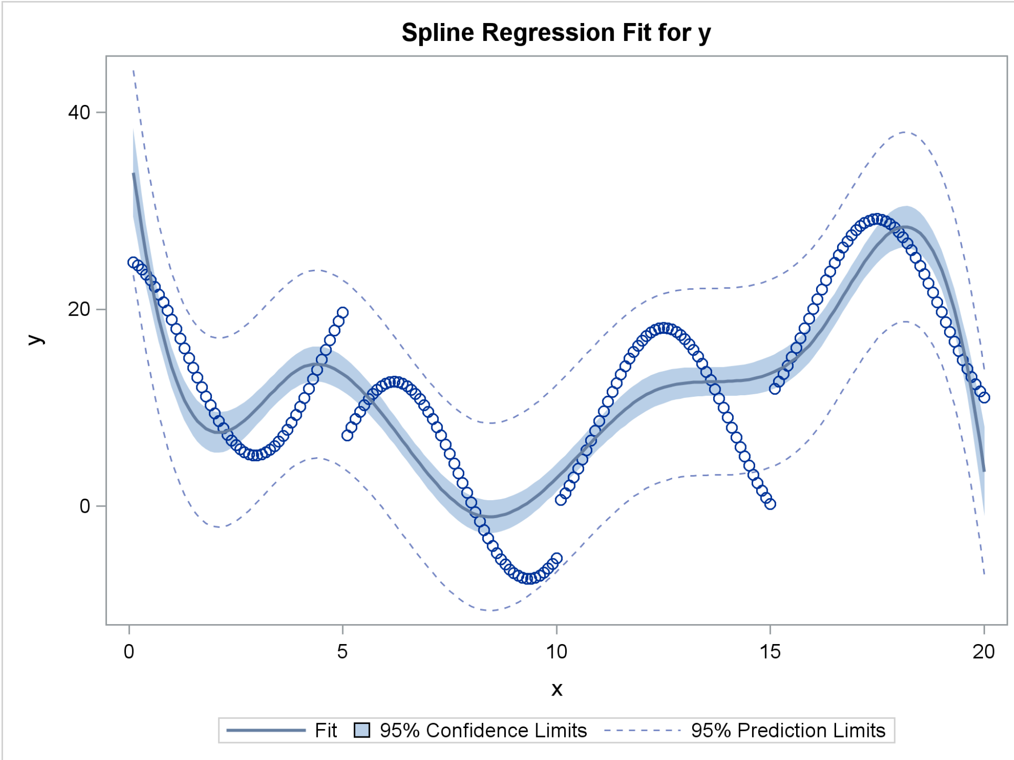


Figure 120.26 Spline Fit with Knots at the Deciles

An Illustration of Splines and Knots
A Cubic Spline Fit with Knots at X=5, 10, 15
Nine Knots

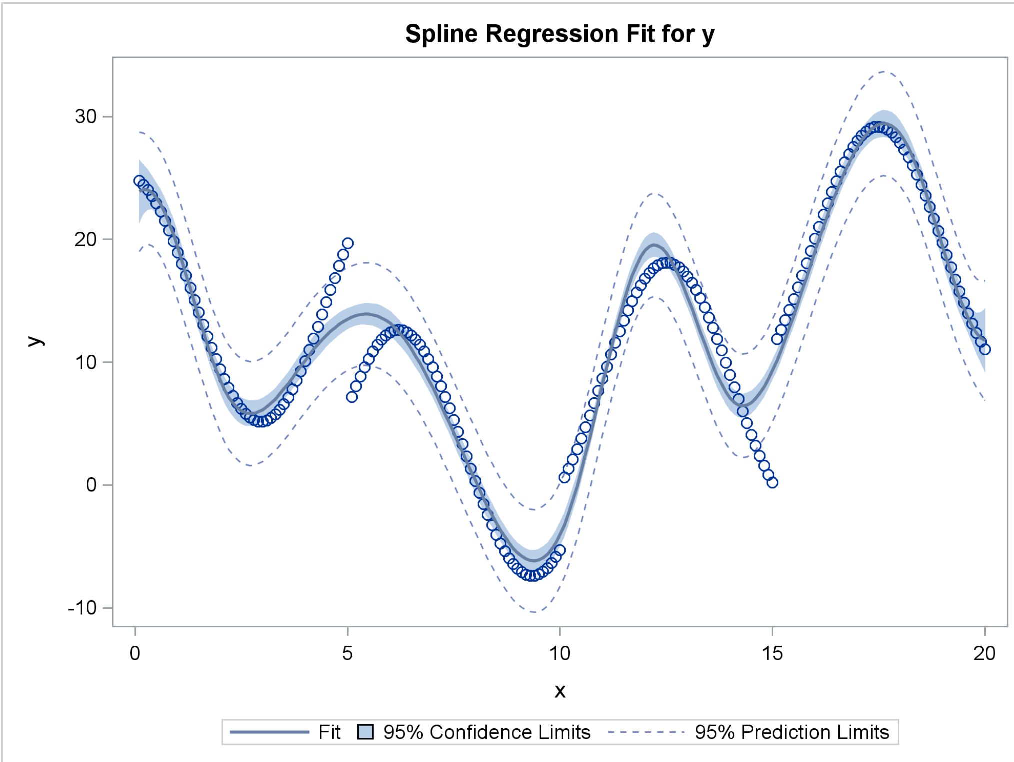
The TRANSREG Procedure

TRANSREG MORALS Algorithm Iteration History for Identity(y)

Iteration Number	Average Change	Maximum Change	R-Square	Criterion Change	Note
1	0.94832	3.03488	0.10061		
2	0.00000	0.00000	0.95256	0.85196	Converged

Algorithm converged.

Figure 120.26 *continued*



Scoring Spline Variables

This section shows you how to find spline transformations of variables in one data set and apply the same transformations to variables in another data set. This is illustrated with artificial data. In these data sets, the variable y is approximately a linear function of nonlinear transformations of the variables x, w, and z. The model is fit using data set X, and those results are used to score data set Z. The following statements create the two data sets:

```
title 'An Illustration of Splines and Knots';
title2 'Scoring Spline Variables';

data x;
   do i = 1 to 5000;
      w = normal(7);
      x = normal(7);
      z = normal(7);
      y = w * w + log(5 + x) + sin(z) + normal(7);
      output;
   end;
run;
```

```
data z;
   do i = 1 to 5000;
      w = normal(1);
      x = normal(1);
      z = normal(1);
      y = w * w + log(5 + x) + sin(z) + normal(1);
      output;
   end;
run;
```

First, you run PROC TRANSREG to fit the transformation regression model asking for spline transformations of the three independent variables. You must use the EXKNOTS= *t-option*, because you need to use the same knots, both interior and exterior, with both data sets. By default, the exterior knots will be different if the minima and maxima are different in the two data sets, so you get the wrong results if you do not specify the EXKNOTS= *t-option* with values less than the minima and greater than the maxima of the six x, y, and w variables. If the ranges in all three pairs were different, you would need separate spline transformation for each variable with different knot and exterior knot specifications. The following statements fit the spline model:

```
proc transreg data=x solve details ss2;
   ods output splinecoef=c;
   model identity(y) = spline(w x z / knots=-1.5 to 1.5 by 0.5
                                       exknots=-5 5);
   output out=d;
run;
```

The results of this step are not displayed. The nonprinting SplineCoef table is output to a SAS data set. This data set contains the coefficients that were used to get the spline transformations and can be used to transform variables in other data sets. These coefficients are also in the details table. However, in the SplineCoef table, they are in a form directly suitable for use with PROC SCORE.

The next step reads the second input data set, Z, and generates an output data set with the B-spline basis for each of the variables:

```
proc transreg data=z design;
   model bspl(w x z / knots=-1.5 to 1.5 by 0.5 exknots=-5 5);
   output out=b;
run;
```

Note that the same interior and exterior knots are used in both of the previous steps. The next three steps score the B-spline bases created in the previous step by using the coefficients generated in the first PROC TRANSREG step. PROC SCORE is run once for each SPLINE variable in the statements that follow:

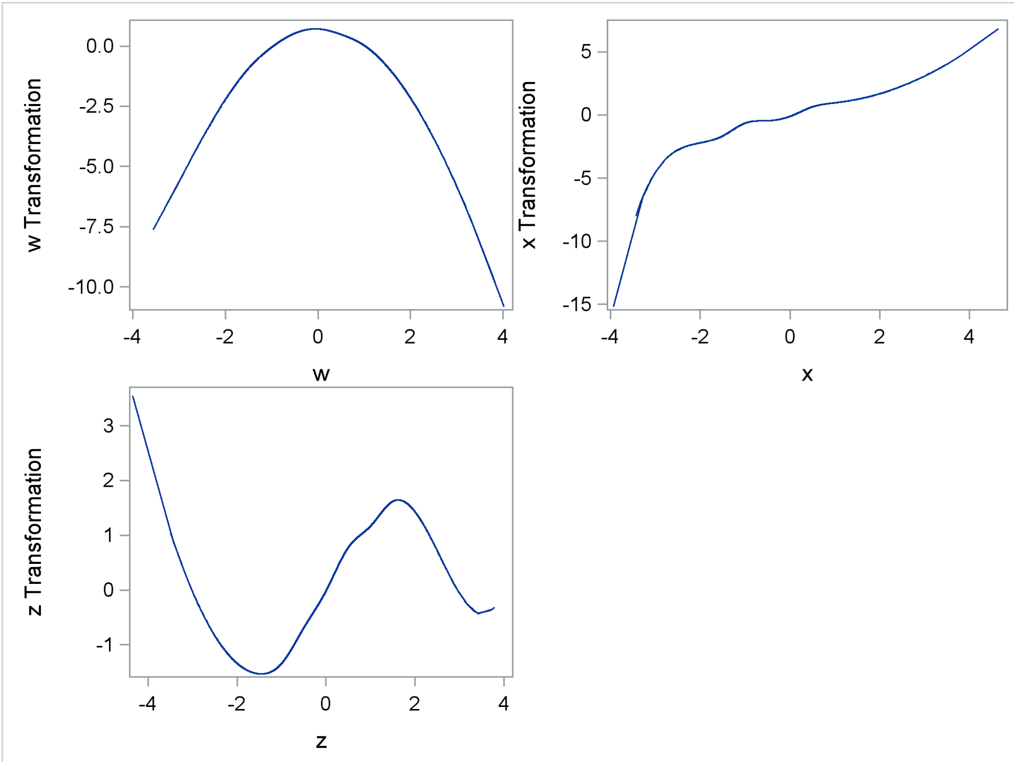
```
proc score data=b score=c out=o1(rename=(spline=bw w=nw));
   var w:;
run;

proc score data=b score=c out=o2(rename=(spline=bx x=nx));
   var x:;
run;

proc score data=b score=c out=o3(rename=(spline=bz z=nz));
   var z:;
run;
```

The following steps merge the three transformations with the original data and plot the results:

```
data all;
   merge d(keep=w x z tw tx tz) o1(keep=nw bw)
         o2(keep=nx bx) o3(keep=nz bz);
run;

proc template;
   define statgraph twobytwo;
      begingraph;
         layout lattice / rows=2 columns=2;
            layout overlay;
               seriesplot y=tw x=w  / connectorder=xaxis;
               seriesplot y=bw x=nw / connectorder=xaxis;
            endlayout;
            layout overlay;
               seriesplot y=tx x=x  / connectorder=xaxis;
               seriesplot y=bx x=nx / connectorder=xaxis;
            endlayout;
            layout overlay;
               seriesplot y=tz x=z  / connectorder=xaxis;
               seriesplot y=bz x=nz / connectorder=xaxis;
            endlayout;
         endlayout;
      endgraph;
   end;
run;

proc sgrender data=all template=twobytwo;
run;
```

The plots in Figure 120.27 show that the two transformations for each variable, original and scored, are the same function. The two functions in each plot are on top of each other and are indistinguishable. Furthermore, PROC TRANSREG found the functional forms that were used to generate the data: quadratic for w, log for x, and sine for z.

Figure 120.27 Scoring Spline Variables Example



The next statements show how to run PROC TRANSREG, output the interior and exterior knots to an output data set with ODS, extract the knots, and use them in a DATA step to re-create the B-spline basis that PROC TRANSREG makes. In practice, you would never need to use a DATA step to make the B-spline basis since PROC TRANSREG does it automatically. The following statements show how you could do it yourself:

```
data x;
   input x @@;
   datalines;
1 2 3 4 5 6 7 8 9 10
;

ods output details=d;
proc transreg details design;
   model bspline(x / nkn=3);
   output out=y;
run;
```

```
%let k = 0;
data d;
   set d;
   length d $ 20;
   retain d ' ';
   if description ne ' ' then d = description;
   if d = 'Degree' then call symput('d', compress(formattedvalue));
   if d = 'Number of Knots'
      then call symput('k', compress(formattedvalue));
   if index(d, 'Knots') and not index(d, 'Number');
   keep d numericvalue;
run;

%let nkn = %eval(&d * 2 + &k); /* total number of knots   */
%let nb  = %eval(&d + 1 + &k); /* number of cols in basis */

proc transpose data=d out=k(drop=_name_) prefix=Knot;
run;

proc print; format k: 20.16;
run;

data b(keep=x:);
   if _n_ = 1 then set k; /* read knots from transreg */
   array k[&nkn] knot1-knot&nkn;        /* knots      */
   array b[&nb] x_0 - x_%eval(&nb - 1); /* basis      */
   array w[%eval(2 * &d)];              /* work       */
   set x;
   do i = 1 to &nb; b[i] = 0; end;

   * find the index of first knot greater than current data value;
   do ki = 1 to &nkn while(k[ki] le x); end;
   kki = ki - &d - 1;

   * make the basis;
   b[1 + kki] = 1;
   do j = 1 to &d;
      w[&d + j] = k[ki + j - 1] - x;
      w[j] = x - k[ki - j];
      s = 0;
      do i = 1 to j;
         t = w[&d + i] + w[j + 1 - i];
         if t ne 0.0 then t = b[i + kki] / t;
         b[i + kki] = s + w[&d + i] * t;
         s = w[j + 1 - i] * t;
      end;
      b[j + 1 + kki] = s;
   end;
run;

proc compare data=y(keep=x:) compare=b
   criterion=1e-12 note nosummary;
   title3 "should be no differences";
run;
```

The output from these steps is not shown. There are several things to note about the DATA step. It produces the same basis as PROC TRANSREG only because it uses exactly the same interior and exterior knots. The exterior knots (0.999999999999 and 10.000000000001) are just slightly smaller than 1 (the minimum in x) and just slightly greater than 10 (the maximum in x). Both exterior knots appear in the list three times, because a cubic (degree 3) polynomial was requested. The complete knot list is: 0.999999999999 0.999999999999 0.999999999999 3 6 8 10.000000000001 10.000000000001 10.000000000001. The exterior knots do not have any particular interpretation, but they are needed by the algorithm to construct the proper basis. The construction method computes differences between each value and the nearby knots. The algorithm that makes the B-spline basis is not very obvious, particularly compared to the polynomial spline basis. However, the B-spline basis is much better behaved numerically than a polynomial-spline basis, so that is why it is used.

Linear and Nonlinear Regression Functions

This section shows how to use PROC TRANSREG in simple regression (one dependent variable and one independent variable) to find the optimal regression line, a nonlinear but monotone regression function, and a nonlinear and nonmonotone regression function. To find a linear regression function, specify the IDENTITY transformation of the independent variable. For a monotone curve, specify the MSPLINE transformation of the independent variable. To relax the monotonicity constraint, specify the SPLINE transformation. You can get more flexibility in spline functions by specifying knots. The more knots you specify, the more freedom the function has to follow minor variations in the data. This example uses artificial data. While these artificial data are clearly not realistic, their distinct pattern helps illustrate how splines work. The following statements generate the data and produce Figure 120.28 through Figure 120.31:

```
title 'Linear and Nonlinear Regression Functions';

* Generate an Artificial Nonlinear Scatter Plot;
data a;
   do i = 1 to 500;
      x = i / 2.5;
      y = -((x/50)-1.5)**2 + sin(x/8) + sqrt(x)/5 + 2*log(x) + cos(x);
      x = x / 21;
      if y > 2 then output;
   end;
run;

ods graphics on;
ods select fitplot(persist);

title2 'Linear Regression';

proc transreg data=a;
   model identity(y)=identity(x);
run;

title2 'A Monotone Regression Function';
```

```
proc transreg data=a;
   model identity(y)=mspline(x / nknots=9);
run;

title2 'A Nonlinear Regression Function';

proc transreg data=a;
   model identity(y)=spline(x / nknots=9);
run;

title2 'A Nonlinear Regression Function, 100 Knots';

proc transreg data=a;
   model identity(y)=spline(x / nknots=100);
run;

ods select all;
```

Figure 120.28 Linear Regression

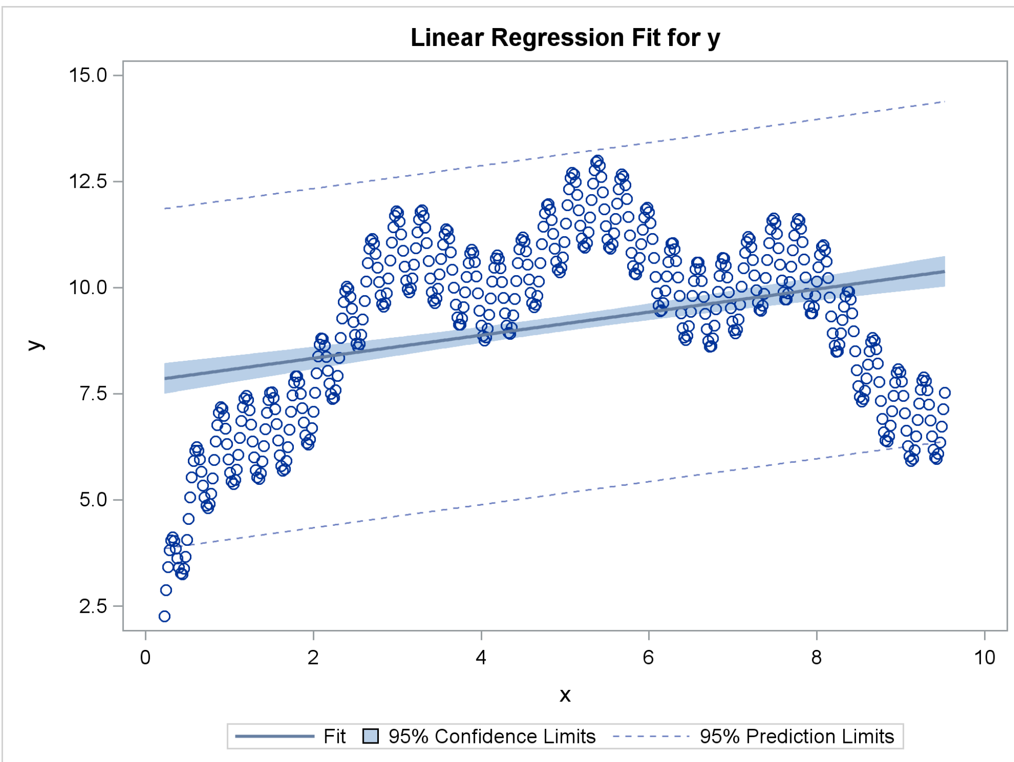


Figure 120.29 A Monotone Regression Function

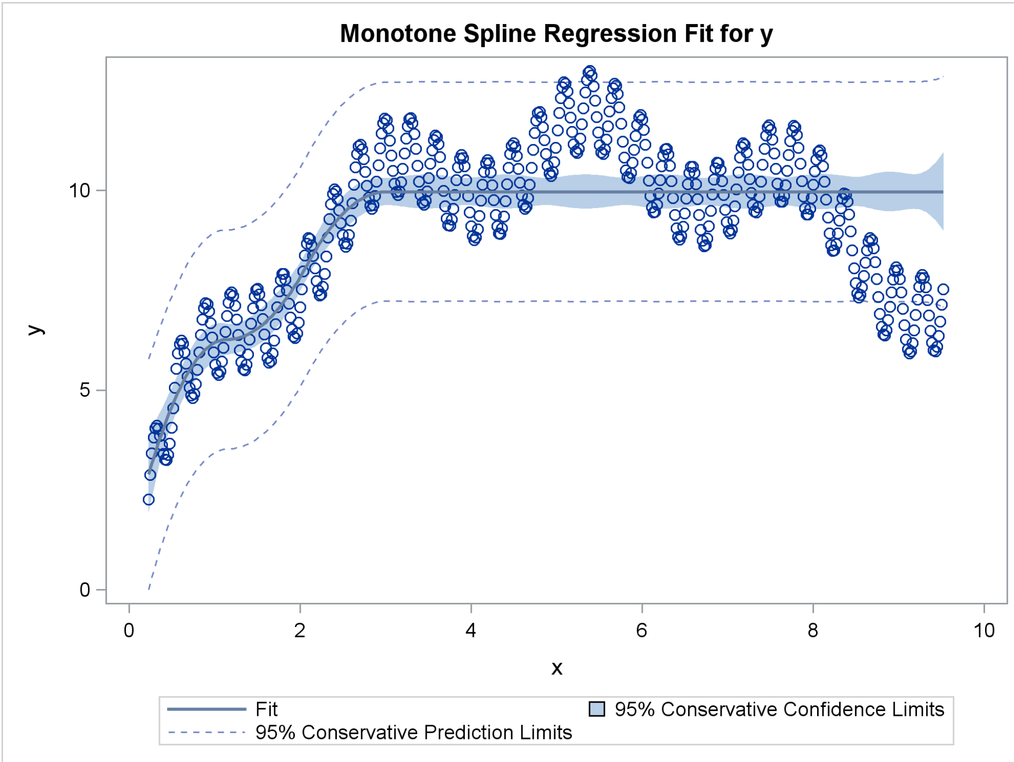


Figure 120.30 A Nonlinear Regression Function

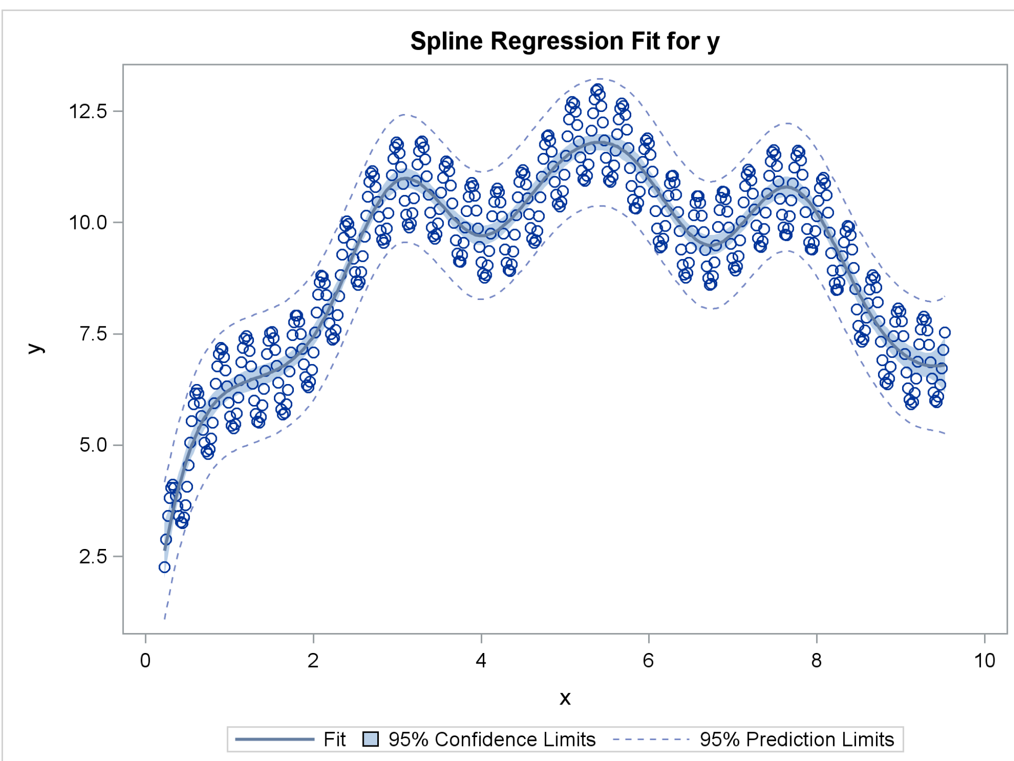
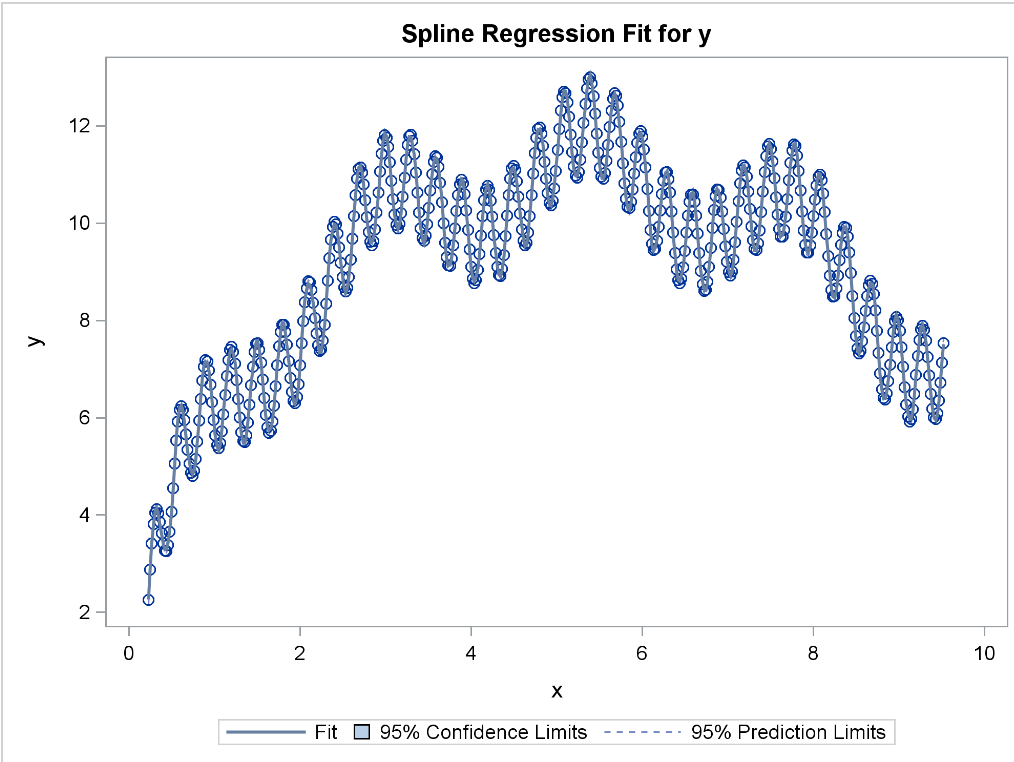


Figure 120.31 A Less-Smooth Nonlinear Regression Function



The squared correlation is only 0.15 for the linear regression in Figure 120.28. Clearly, a simple linear regression model is not appropriate for these data. By relaxing the constraints placed on the regression line, the proportion of variance accounted for increases from 0.15 (linear) to 0.61 (monotone in Figure 120.29) to 0.90 (nonlinear but smooth in Figure 120.30) to almost 1.0 with 100 knots (nonlinear and not very smooth in Figure 120.28). Relaxing the linearity constraint permits the regression function to bend and more closely follow the right portion of the scatter plot. Relaxing the monotonicity constraint permits the regression function to follow the periodic portion of the left side of the plot more closely. The nonlinear MSPLINE transformation is a quadratic spline with knots at the deciles. The first nonlinear nonmonotonic SPLINE transformation is a cubic spline with knots at the deciles.

Different knots and different degrees would produce slightly different results. The two nonlinear regression functions could be closely approximated by simpler piecewise linear regression functions. The monotone function could be approximated by a two-piece line with a single knot at the elbow. The first nonmonotone function could be approximated by a six-piece function with knots at the five elbows.

With this type of problem (one dependent variable with no missing values that is not transformed and one independent variable that is nonlinearly transformed), PROC TRANSREG always iterates exactly twice (although only one iteration is necessary). The first iteration reports the R square for the linear regression line and finds the optimal transformation of x. Since the data change in the first iteration, a second iteration is performed, which reports the R square for the final nonlinear regression function, and zero data change. The predicted values, which are a linear function of the optimal transformation of x, contain the Y coordinates for the nonlinear regression function. The variance of the predicted values divided by the variance of y is the R square for the fit of the nonlinear regression function. When x is monotonically transformed, the transformation of x is always monotonically increasing, but the predicted values increase if the correlation is positive and decrease for negative correlations.

Simultaneously Fitting Two Regression Functions

One application of ordinary multiple regression is fitting two or more regression lines through a single scatter plot. With PROC TRANSREG, this application can easily be generalized to fit separate or parallel curves. To illustrate, consider a data set with two groups and a group membership variable g that has the value 1 for one group and 2 for the other group. The data set also has a continuous independent variable x and a continuous dependent variable y. When g is crossed with x, the variables g1x and g2x both have a large partition of zeros. For this reason, the KNOTS= *t-option* is specified instead of the NKNOTS= *t-option*. (The latter would put a number of knots in the partition of zeros.) The following example generates an artificial data set with two curves. While these artificial data are clearly not realistic, their distinct pattern helps illustrate how fitting simultaneous regression functions works. The following statements generate data and show how PROC TRANSREG fits lines, curves, and monotone curves through a scatter plot:

```
title 'Separate Curves, Separate Intercepts';

data a;
   do x = -2 to 3 by 0.025;
      g = 1;
      y = 8*(x*x + 2*cos(x*6)) + 15*normal(7654321);
      output;
      g = 2;
      y = 4*(-x*x + 4*sin(x*4)) - 40 + 15*normal(7654321);
      output;
   end;
run;

ods graphics on;
ods select fitplot(persist);

title 'Parallel Lines, Separate Intercepts';

proc transreg data=a solve;
   model identity(y)=class(g) identity(x);
run;

title 'Parallel Monotone Curves, Separate Intercepts';

proc transreg data=a;
   model identity(y)=class(g) mspline(x / knots=-1.5 to 2.5 by 0.5);
run;

title 'Parallel Curves, Separate Intercepts';

proc transreg data=a solve;
   model identity(y)=class(g) spline(x / knots=-1.5 to 2.5 by 0.5);
run;

title 'Separate Slopes, Same Intercept';
```

```
proc transreg data=a;
   model identity(y)=class(g / zero=none) * identity(x);
run;

title 'Separate Monotone Curves, Same Intercept';

proc transreg data=a;
   model identity(y) = class(g / zero=none) *
                       mspline(x / knots=-1.5 to 2.5 by 0.5);
run;

title 'Separate Curves, Same Intercept';

proc transreg data=a solve;
   model identity(y) = class(g / zero=none) *
                       spline(x / knots=-1.5 to 2.5 by 0.5);
run;

title 'Separate Slopes, Separate Intercepts';

proc transreg data=a;
   model identity(y) = class(g / zero=none) | identity(x);
run;

title 'Separate Monotone Curves, Separate Intercepts';

proc transreg data=a;
   model identity(y) = class(g / zero=none) |
                       mspline(x / knots=-1.5 to 2.5 by 0.5);
run;

title 'Separate Curves, Separate Intercepts';

proc transreg data=a solve;
   model identity(y) = class(g / zero=none) |
                       spline(x / knots=-1.5 to 2.5 by 0.5);
run;
ods select all;
```

The previous statements produce Figure 120.32 through Figure 120.40. Only the fit plots are generated and displayed.

Figure 120.32 Parallel Lines, Separate Intercepts

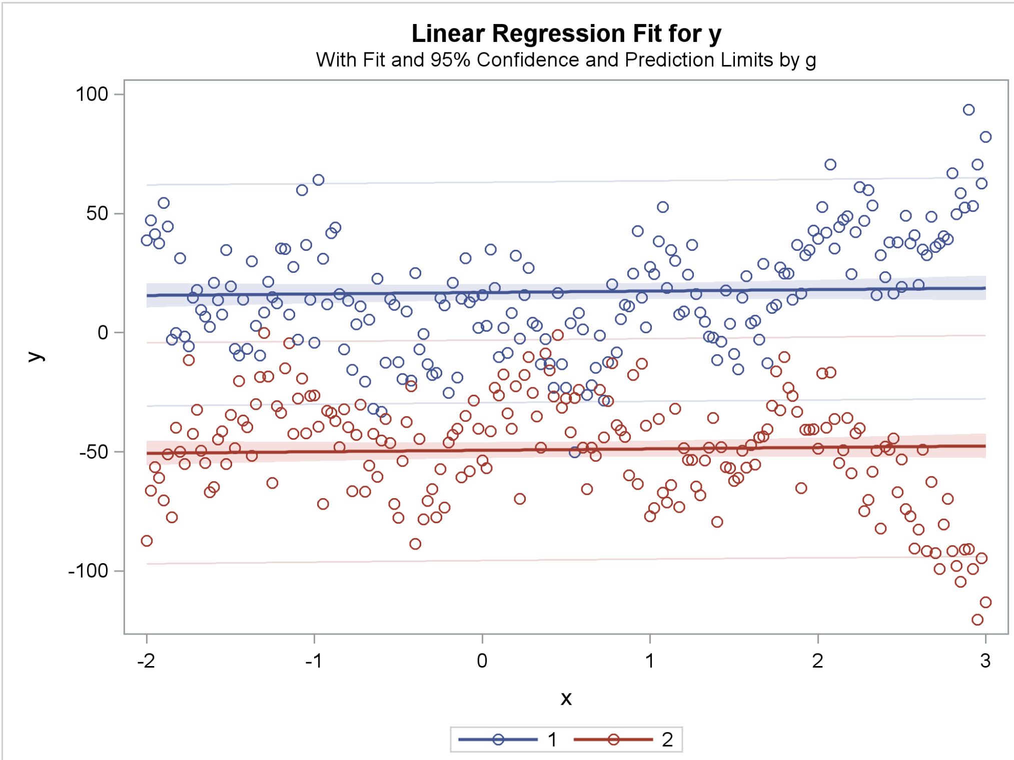


Figure 120.33 Parallel Monotone Curves, Separate Intercepts

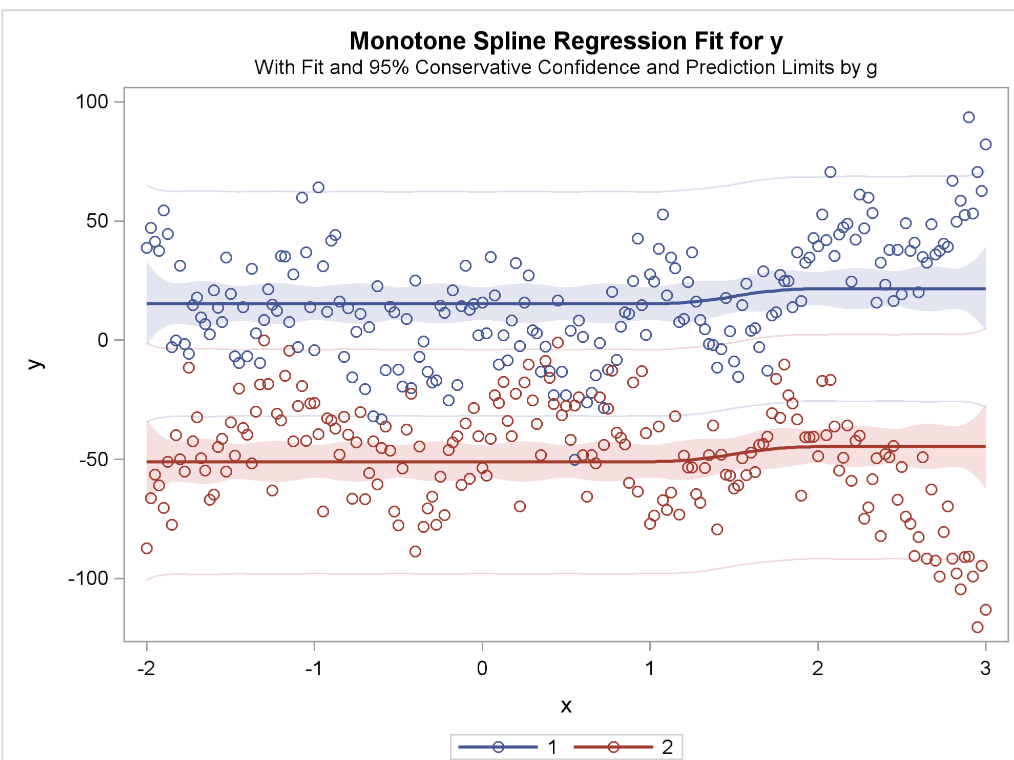


Figure 120.34 Parallel Curves, Separate Intercepts

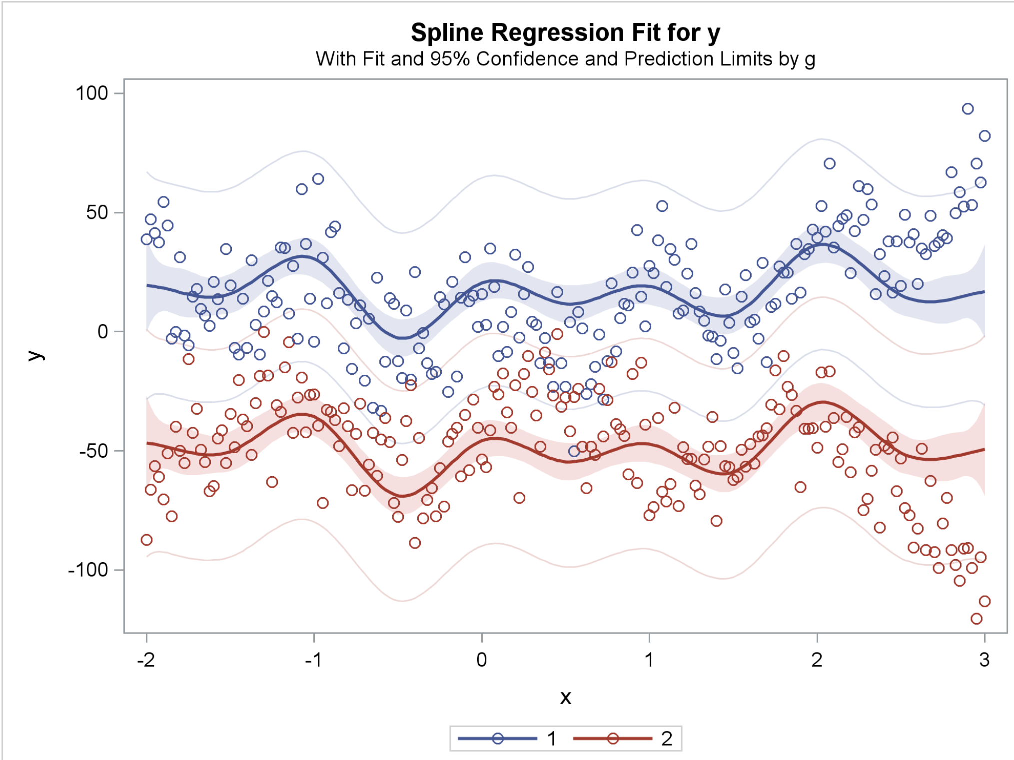


Figure 120.35 Separate Slopes, Same Intercept

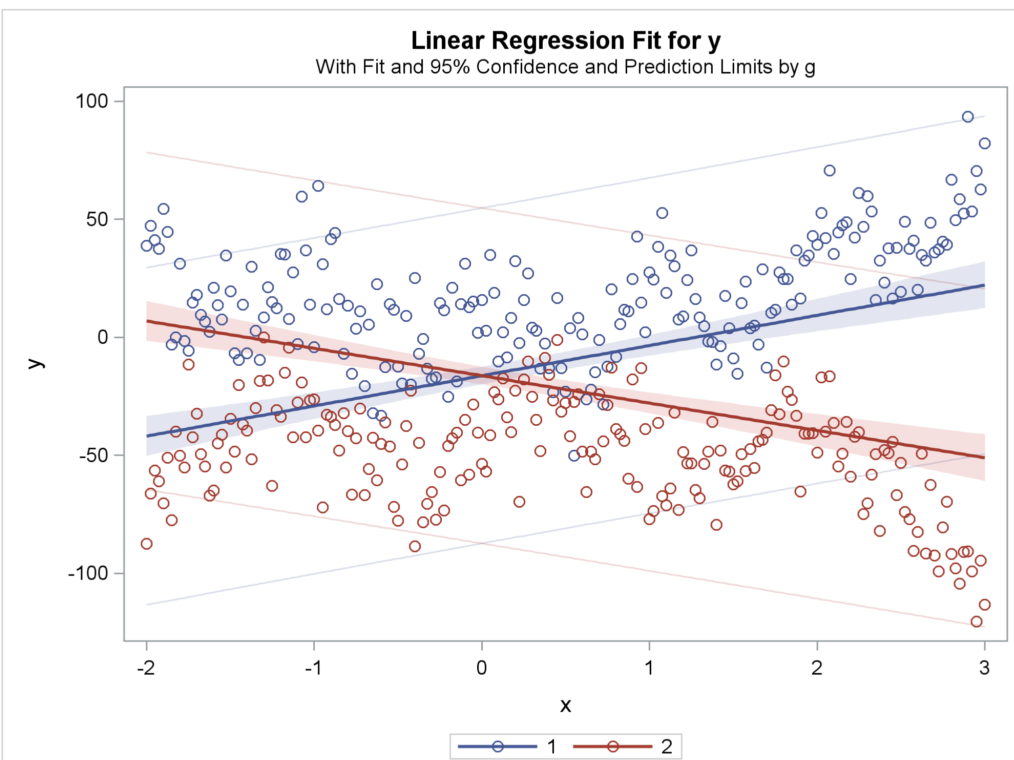


Figure 120.36 Separate Monotone Curves, Same Intercept

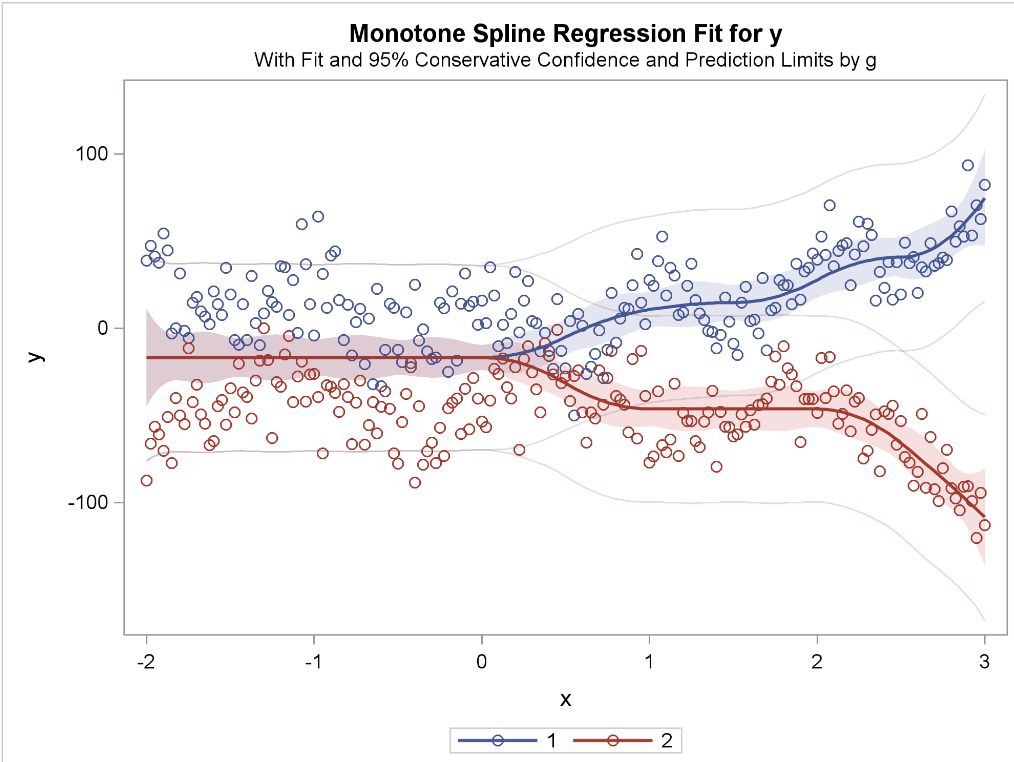


Figure 120.37 Separate Curves, Same Intercept

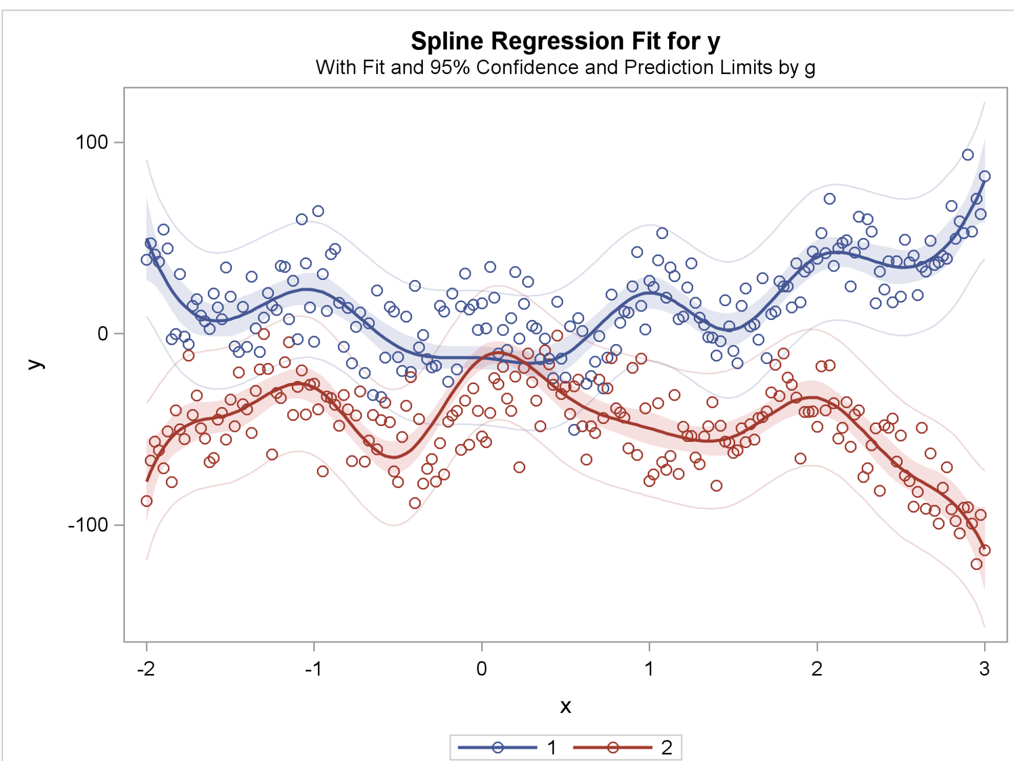


Figure 120.38 Separate Slopes, Separate Intercepts

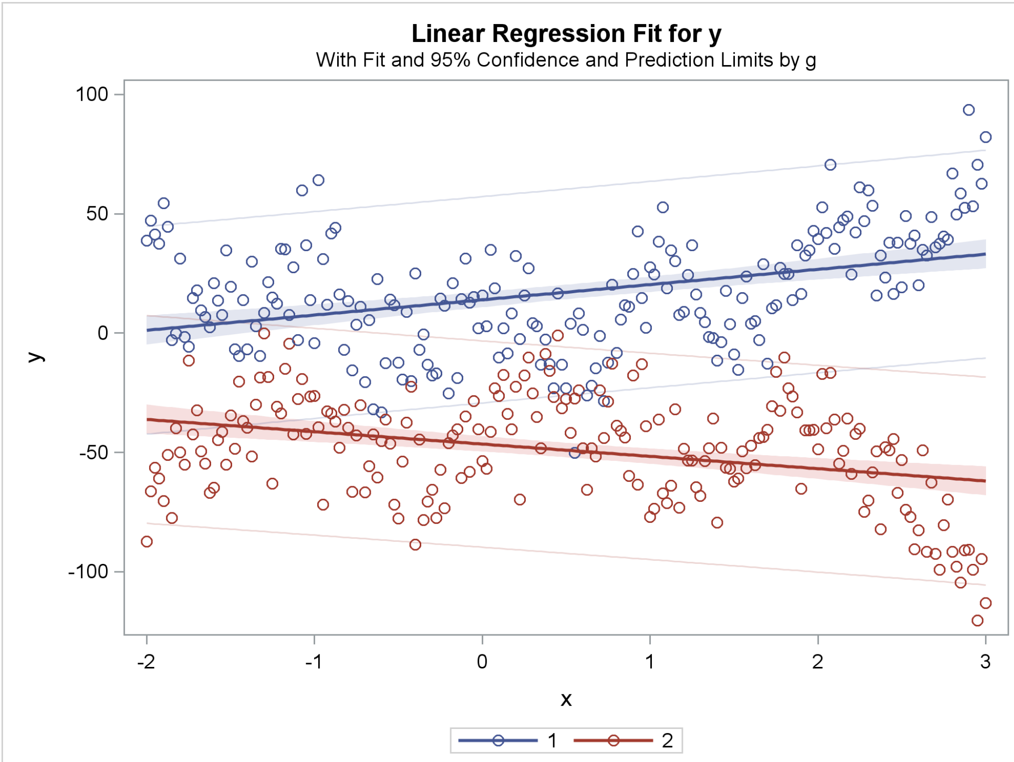


Figure 120.39 Separate Monotone Curves, Separate Intercepts

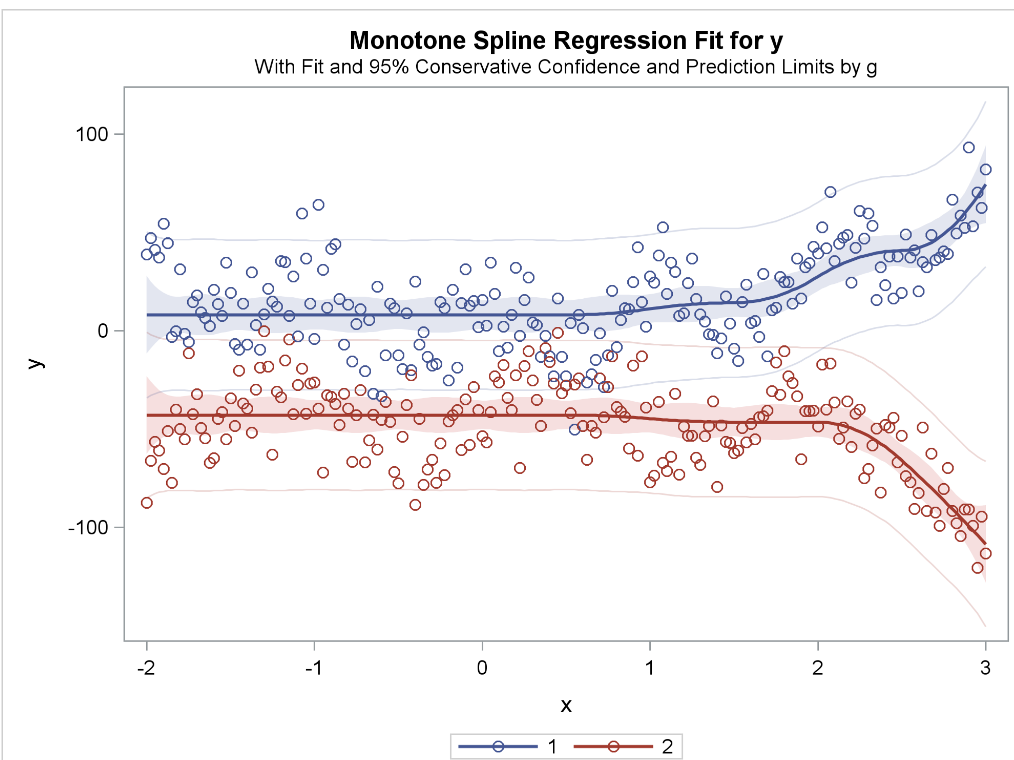
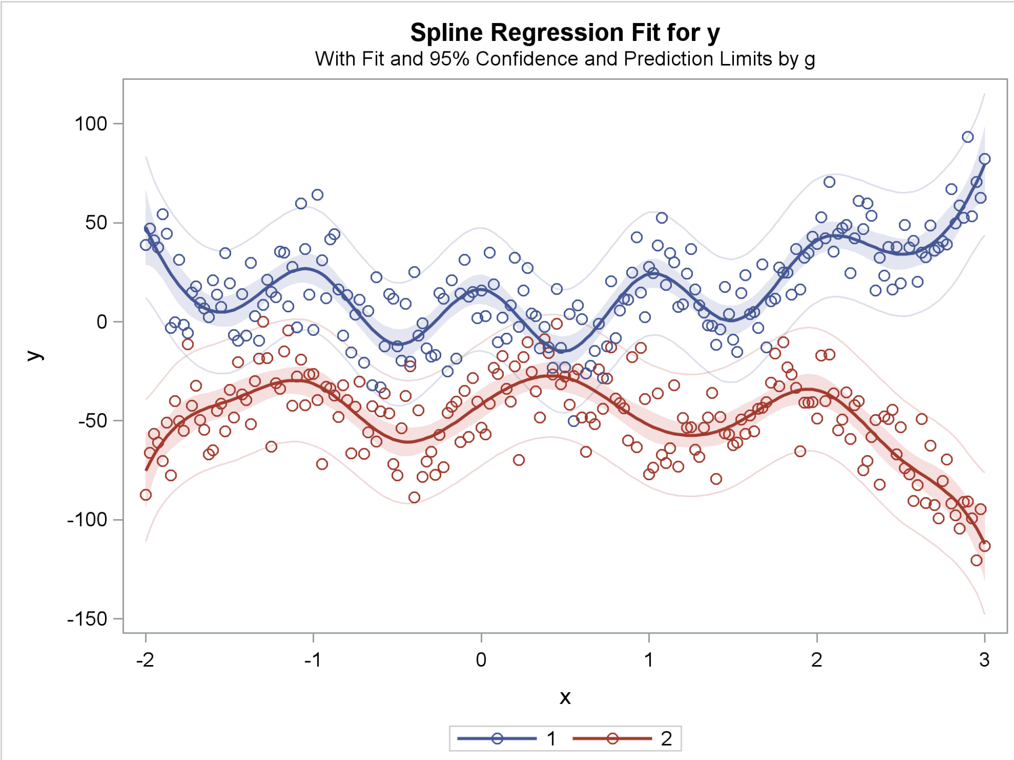


Figure 120.40 Separate Curves, Separate Intercepts



Penalized B-Splines

You can use penalized B-splines (Eilers and Marx 1996) to fit a smooth curve through a scatter plot with an automatic selection of the smoothing parameter. See Example 120.3 for an example. With penalized B-splines, you can find a transformation that minimizes any of the following criteria: CV, GCV, AIC, AICC, or SBC. These criteria are all functions of λ. For many problems, all of these criteria produce nearly identical results. However, for some problems, the choice of criterion can have a large effect. When the default results are not satisfactory, try the other criteria. Information criteria such as AIC and AICC are defined in different ways in the statistical literature, and these differences can be seen in different SAS procedures. Typically, the definitions differ only by a positive (additive or multiplicative) constant, so they are equivalent, and each of the definitions of the same criterion produces the same selection of λ. The definitions that PROC TRANSREG uses match the definitions that PROC REG uses. The penalized B-spline matrices, statistics, and criteria are defined as follows:

n	number of observations
$\mathbf{y}$	dependent variable
$\mathbf{W}$	diagonal matrix of observation weights
w_i	weight for the ith observation
$\mathbf{B}$	B-spline basis for the independent variable
λ	nonnegative smoothing parameter
$\mathbf{D}$	difference matrix, penalizes lack of smoothness
$\mathbf{H} = \mathbf{B}(\mathbf{B}'\mathbf{W}\mathbf{B} + \lambda\mathbf{D}'\mathbf{D})^{-1}\mathbf{B}'\mathbf{W}$	hat matrix
h_{ii}	ith diagonal element of $\mathbf{H}$

$\hat{\mathbf{y}} = \mathbf{H}\mathbf{y}$	penalized B-spline transformation of $\mathbf{y}$
$\text{SSE} = \sum_{i=1}^{n} w_i (y_i - \hat{y}_i)^2$	error sum of squares
$t = \sum_{i=1}^{n} h_{ii}$	trace of $\mathbf{H}$
$\sum_{i=1}^{n} w_i \left(\frac{y_i - \hat{y}_i}{1 - h_{ii}}\right)^2$	CV - cross validation criterion
$\sum_{i=1}^{n} w_i \left(\frac{y_i - \hat{y}_i}{n - t}\right)^2$	GCV - generalized cross validation criterion
$n \log(\text{SSE}/n) + 2t$	AIC - Akaike's information criterion
$1 + \log(\text{SSE}/n) + \frac{2(t+1)}{n-t-2}$	AICC - corrected AIC (default)
$n \log(\text{SSE}/n) + t \log(n)$	SBC - Schwarz's Bayesian criterion

For more information about constructing the B-spline basis, see "SPLINE and MSPLINE Transformations" on page 9999 and the section "Using Splines and Knots" on page 9932. The nonzero elements of $\mathbf{D}$, order 1 are (1 –1), order 2 are (1 –2 1), order 3 (the default) are (1 –3 3 –1), order 4 are (1 –4 6 –4 1), and so on. The nonzero elements for each order are made from the nonzero elements from the preceding order by subtraction: $\mathbf{d}'_{i+1} = (\mathbf{d}'_i \;\; 0) - (0 \;\; \mathbf{d}'_i)$. Within an order, the first nonzero element of row i is in column i—that is, each row of $\mathbf{D}$ is made from the preceding row by shifting the nonzero elements to the right one position. For example, with k = 4 knots, order o = 3, and degree d = 3, $\mathbf{D}$ is the $((d+1+k-o) \times (d+1+k))$ matrix:

$$\begin{bmatrix} 1 & -3 & 3 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -3 & 3 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -3 & 3 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -3 & 3 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -3 & 3 & -1 \end{bmatrix}$$

$$\text{where} \quad \begin{array}{rrrrr} & 1 & -1 & 0 & \\ - & 0 & 1 & -1 & \\ \hline & 1 & -2 & 1 & 0 \\ - & 0 & 1 & -2 & 1 \\ \hline & 1 & -3 & 3 & -1 \end{array}$$

The trace of the hat matrix, $t = \sum_{i=1}^{n} h_{ii}$, provides an estimate of the number of parameters needed to find the transformation and is used in *df* calculations. Note, however, that in some cases, particularly with error-free or nearly error-free data, this value can be *much* larger than you might expect. You might be able to directly create a function by using SPLINE or BSPLINE with many fewer parameters that fits essentially just as well as the penalized B-spline function.

By default with PBSPLINE, a cubic spline is fit with 100 evenly spaced knots, three evenly spaced exterior knots, and a difference matrix of order three. Options are specified as follows: PBSPLINE(x / DEGREE=3 NKNOTS=100 EVENLY=3 PARAMETER=3). By default, PROC TRANSREG searches for an optimal lambda in the range 0 to 1E6 by using parabolic interpolation and Brent's (Brent 1973; Press et al. 1989) method. Alternatively, you can specify a lambda range or a list of lambdas by using the LAMBDA= option. Be aware, however, LAMBDA=0 and values near zero might cause numerical problems including floating point errors. Also be aware that larger lambdas might cause numerical problems—for example, the error sum of squares for the model, $\Sigma(y - \hat{y})^2$, might be greater than the total sum of squares, $\Sigma(y - \bar{y})^2$—implying that the model with the transformation fits less well than simply predicting by using the mean. When this happens, you will see this message: ERROR: Degenerate transformation with PBSPLINE.

You can fit a single curve through a scatter plot (y $\times$ x) as follows:

```
model identity(y) = pbspline(x);
```

Alternatively, you can fit multiple curves through a scatter plot, one for each level of Group, as follows:

```
model identity(y) = class(group / zero=none) * pbspline(x);
```

There are several options for how the smoothing parameter, λ, is chosen. Usually, you do not specify the smoothing parameter, λ, and you let PROC TRANSREG choose λ for you by minimizing one of the information or cross validation criteria. By default, PROC TRANSREG first considers ranges defined by $\lambda = 0$ and $\lambda = 1, 10, 100, 1000, 10,000, 100,000, 1,000,000$. If it finds a range that includes the minimum, it stops and does not consider larger λ values. Then it performs further searches in that range. For example, if the initial evaluations at $\lambda = 1$ and $\lambda = 10$ show that there is at least a local minimum in the range 0 to 10, then larger values are not considered. Note that the zero smoothing case, $\lambda = 0$, provides a boundary on the range even though the criterion is not evaluated at $\lambda = 0$. The criterion is not evaluated at $\lambda = 0$ unless LAMBDA=0 is the only value specified. Also note that the default approach is not the same as specifying the options LAMBDA=0 1E6 RANGE. When a range of values is specified, along with the RANGE *t-option*, PROC TRANSREG does not try to find smaller ranges based on powers of 10.

PROC TRANSREG avoids evaluating the criterion for LAMBDA= values at or near zero unless you force it to consider them. This is because zero smoothing is rarely interesting and the results are numerically unstable. Values of λ at or near zero often result in predicted values that are far outside the range of the data, particularly with interpolation and x values that do not appear in the data set. Also, zero smoothing is prone to numerical problems including floating point errors. This is particularly true when there is a small number of observations, a large number of knots, a high degree, or a perfect or near perfect fit. If you force PROC TRANSREG to evaluate the criterion at or near $\lambda = 0$, you can easily get bad results.

Note that when some observations appear more than once, such as when you have the kind of data where you can use a FREQ statement, then you should consider directly specifying lambda based on a preliminary analysis, ignoring the frequencies. Alternatively, specify a range of λ values, such as LAMBDA=0.1 1E6 RANGE, that steers λ away from values near zero. With the default lambda list, a cross validation criterion does not perform well in choosing a smoothing parameter with replicated data. Leaving one observation out of the computations changes the frequency for that observation from one positive integer to the next smaller positive integer, so in some sense, the point corresponding to that observation is never really left out of any computations. The resulting fit will be undersmoothed unless you specify a larger λ.

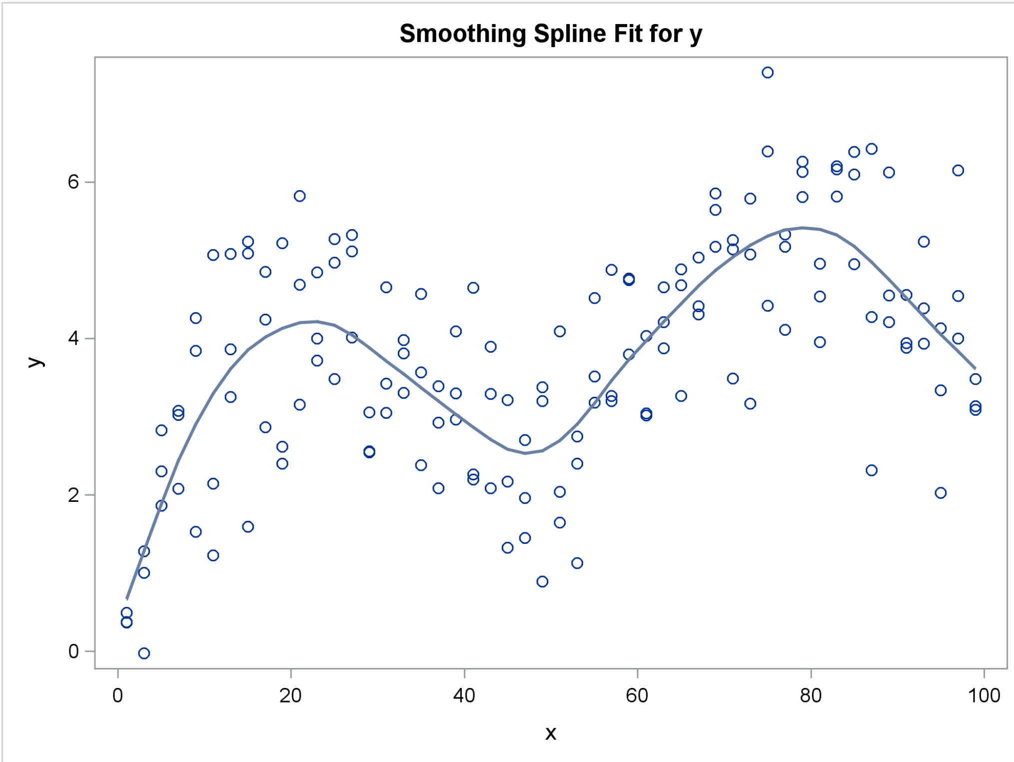
Smoothing Splines

You can use PROC TRANSREG to plot and output to a SAS data set the same smoothing spline function that the GPLOT procedure creates. You request a smoothing spline transformation by specifying SMOOTH in the MODEL statement. The smoothing parameter can be specified with either the SM= or the PARAMETER= *o-option*. The results are saved in the independent variable transformation (for example, Tx, when the independent variable is x) and the predicted values variable (for example, Py, when the dependent variable is y).

You can display the smoothing spline by using PROC TRANSREG and ODS Graphics (as shown in Figure 120.41). The following statements produce Figure 120.41:

```
title h=1.5 'Smoothing Splines';

ods graphics on;

data x;
   do x = 1 to 100 by 2;
      do rep = 1 to 3;
         y = log(x) + sin(x / 10) + normal(7);
         output;
      end;
   end;
run;

proc transreg;
   model identity(y) = smooth(x / sm=50);
   output p;
run;
```

Figure 120.41 Smoothing Spline Displayed with ODS Graphics



You can also use PROC GPLOT to verify that the two procedures produce the same results. The PROC GPLOT plot request **y * x = 1** displays the data as stars. The specification **y * x = 2** with I=SM50 requests the smooth curve through the scatter plot. It is overlaid with **Py * x = 3**, which displays with large dots the smooth function created by PROC TRANSREG. The results of the following step are not displayed:

```
proc gplot;
   axis1 minor=none label=(angle=90 rotate=0);
   axis2 minor=none;
   symbol1 color=blue v=circle i=none;   /* data              */
   symbol2 color=blue v=none   i=sm50;   /* gplot's smooth    */
   symbol3 color=red  v=dot    i=none;   /* transreg's smooth */
   plot y*x=1 y*x=2 py*x=3 / overlay haxis=axis2 vaxis=axis1 frame;
run; quit;
```

You can plot multiple nonlinear functions, one for each of several groups as defined by the levels of a CLASS variable. When you cross a SMOOTH variable with a CLASS variable, specify ZERO=NONE with the CLASS expansion. The following statements create artificial data and produce Figure 120.42:

```
title2 'Two Groups';

data x;
   do x = 1 to 100;
      Group = 1;
      do rep = 1 to 3;
         y = log(x) + sin(x / 10) + normal(7);
         output;
```

```
         end;
         group = 2;
         do rep = 1 to 3;
            y = -log(x) + cos(x / 10) + normal(7);
            output;
         end;
      end;
   run;

   proc transreg ss2 data=x;
      model identity(y) = class(group / zero=none) *
                          smooth(x / sm=50);
      output p;
   run;
```

The ANOVA table in Figure 120.42 shows the overall model fit. The degrees of freedom are based on the trace of the transformation hat matrix, and are typically not integers. The "Smooth Transformation" table reports the degrees of freedom for each term, which includes an intercept for each group; the regression coefficients, which are always 1 with smoothing splines; the 0 to 100 smoothing parameter (like the one PROC GPLOT uses); the actual computed smoothing parameter; and the name and label for each term.

Figure 120.42 Smoothing Spline Example 2

Smoothing Splines
Two Groups

The TRANSREG Procedure

Dependent Variable Identity(y)

Class Level Information		
Class	**Levels**	**Values**
Group	2	1 2

Number of Observations Read	600
Number of Observations Used	600
Implicit Intercept Model	

The TRANSREG Procedure Hypothesis Tests for Identity(y)

Univariate ANOVA Table, Smooth Transformation

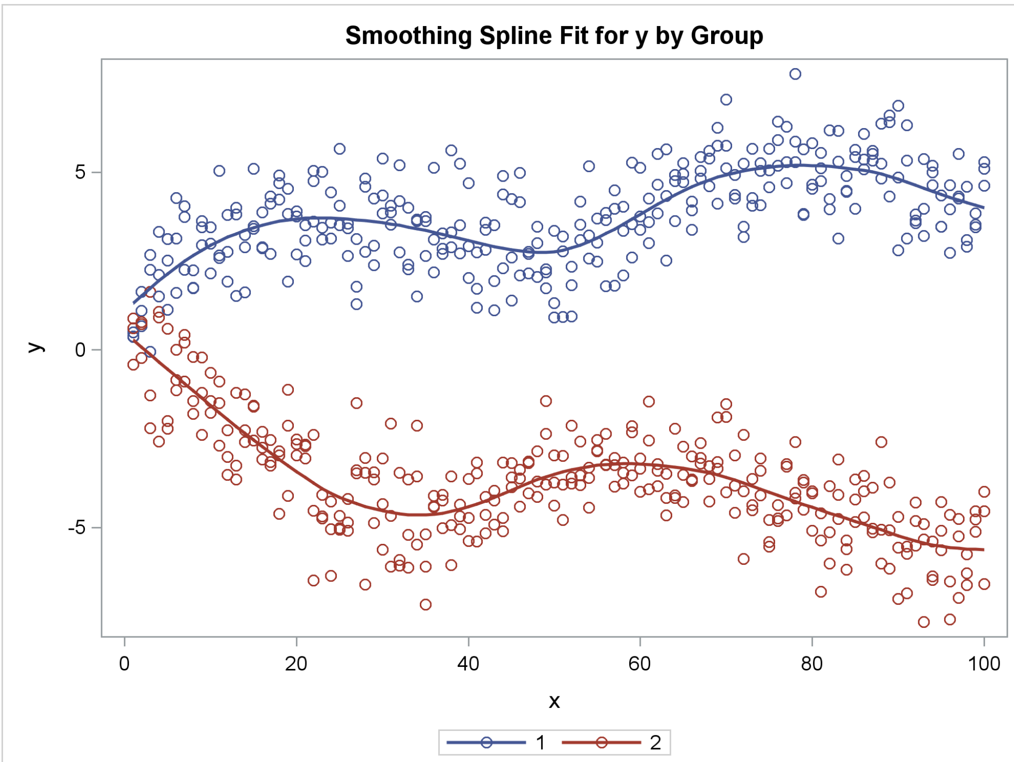
Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	16.794	9195.493	547.5365	562.03	<.0001
Error	582.21	567.195	0.9742		
Corrected Total	599	9762.688			

Root MSE	0.98702	**R-Square**	0.9419
Dependent Mean	0.03651	**Adj R-Sq**	0.9402
Coeff Var	2703.13908		

Figure 120.42 *continued*

Smooth Transformation					
Variable	**DF**	**Coefficient**	**SM**	**Parameter**	**Label**
Smooth(Group1x)	8.8971	1.000	50	2405.265	Group 1 * x
Smooth(Group2x)	8.8971	1.000	50	2405.265	Group 2 * x

Figure 120.42 *continued*



The SMOOTH transformation is valid only with independent variables. Typically, it is used only, as in the two preceding examples, in models with a single dependent variable, a single independent variable, and optionally, a single classification variable that is crossed with the independent variable. The various standardization options such as TSTANDARD=, CENTER, Z, and REFLECT are by default not permitted when the SMOOTH transformation is part of the model.

The SMOOTH transformation can also be used in other ways, but only when you specify the NSR *a-option*. (See the section "Smoothing Splines Changes and Enhancements" on page 9966.) When you specify the NSR *a-option*, and there are multiple independent variables designated as SMOOTH, PROC TRANSREG tries to smooth the *i*th independent variable by using the *i*th dependent variable as a target. When there are more independent variables than dependent variables, the last dependent variable is reused as often as is necessary. For example, consider the following statements:

```
proc transreg nsr;
   model identity(y1-y3) = smooth(x1-x5);
run;
```

Smoothing is based on the pairs (y1, x1), (y2, x2), (y3, x3), (y3, x4), and (y3, x5).

The SMOOTH transformation is a noniterative transformation. The smoothing of each variable occurs before the iterations begin. In contrast, SSPLINE provides an iterative smoothing spline transformation. It does not generally minimize squared error; hence, divergence is possible with SSPLINE.

Smoothing Splines Changes and Enhancements

How the results of the transformation are processed in PROC TRANSREG has changed with SAS 9.2. In particular, some aspects of the syntax along the coefficients and predicted values have changed. The new behavior was required to make the smoothing splines work properly with ODS Graphics and to make SMOOTH work consistently with the new PBSPLINE (penalized B-spline; see the section "Penalized B-Splines" on page 9959) capabilities. However, you can use the new NSR *a-option*, if you want the old functionality. Here are two typical uses of the SMOOTH transformation:

```
proc transreg;
   model identity(y) = smooth(x / sm=50);
   output p;
run;

proc transreg;
   model identity(y) = class(group / zero=none) * smooth(x / sm=50);
   output p;
run;
```

For the first model, the variable x is smoothly transformed by using a smoothing parameter of SM=50, and the results are stored in the transformed variable Tx. The second model has two groups of observations corresponding to group=1 and Group=2. Separate curves are fit through each group. The results for the first group are stored in the transformed variable TGroup1x, and the results for the second group are stored in the transformed variable TGroup2x. The predicted values are stored in Py. In the first case, Py = Tx, and in the second case, Py = TGroup1x + TGroup2x. These represent the two standard usages of the SMOOTH transformation, and you can use ODS Graphics to display fit plots with a single or multiple smooth functions. For the first model, which is the most typical usage, the syntax has not changed, nor has the transformed variable. For the second model, the syntax has slightly changed, but the transformed variables have not. The details of the syntax changes are discussed later in this section. The primary change involves what happens after the SMOOTH transformation is found. Now, by default, ordinary least squares (OLS) is no longer used to find the coefficients when there are smooth transformations, and in the iteration history table the OLS R square is no longer produced.

Here is some background for the change. The first three of the four models shown next have much in common:

```
model identity(y) = smooth(x / sm=50);
model identity(y) = rank(x);
model identity(y) = log(x);
model identity(y) = spline(x);
```

Before SAS 9.2, the SMOOTH, RANK, and LOG transformations all requested that PROC TRANSREG preprocess the data, nonlinearly transforming x before using OLS to fit a model to the preprocessed results. All of these first three transformations of x are nonoptimal in the sense that none of them is based in any way on the OLS regression model that follows the preprocessing of the data. In contrast, the fourth model requests a spline transformation. In this model, both the nonlinear transformation and the final regression model seek to minimize the same OLS criterion. Some PROC TRANSREG transformations, such as SPLINE, MSPLINE, OPSCORE, MONOTONE, and so on, seek to minimize squared error, whereas others, such as SMOOTH, LOG, EXP, and RANK, do not. For the latter, the data are simply preprocessed before analysis. There is a philosophical difference, however, between SMOOTH and the nonoptimal transformations. The SMOOTH and PBSPLINE transformations use the dependent variable and a model (but not OLS) to compute the transformation, whereas LOG, EXP, RANK, and the other nonoptimal transformations do not. A log transformation, for example, would be the same, regardless of context, whereas the SMOOTH and PBSPLINE transformations depend on the model.

The principal change to SMOOTH in PROC TRANSREG with SAS 9.2 involves making PROC TRANSREG aware of the underlying smoothing spline model. This makes SMOOTH and PBSPLINE perform similarly, and less like LOG, EXP, RANK, and the other nonoptimal transformations. Previously, if you specified SMOOTH and then examined the regression coefficients, you would probably get an intercept very close to but not exactly 0, and the remaining coefficients would be very close to but not exactly 1. This is because PROC TRANSREG was using OLS to find the coefficients. This has changed. Now, PROC TRANSREG recognizes that the SMOOTH transformation has an implicit intercept (see the section "Implicit and Explicit Intercepts" on page 9992); hence there is no separate intercept. Furthermore, now the other parameters are exactly 1, which are the correct parameters for the non-OLS smoothing spline model. Hence, the predicted values are now the sum of the transformed variables. When there is no CLASS variable, the predicted values exactly match the transformed variable. The SMOOTH transformation is no longer a form of preprocessing; it now changes the nature of the model from OLS to a true smoothing-spline model. If you still want the old behavior, preprocessing and then OLS, you can get the old default functionality by specifying the NSR *a-option*.

The new, default functionality assumes that you either want to fit a smooth function through the data or fit separate functions, one for each level of a CLASS variable. It also recognizes the smoothing-spline model as a model with an implicit intercept. For these reasons, the syntax for models with a CLASS variable has slightly changed, as is shown next:

```
proc transreg nsr; /* old */
   model identity(y) = class(group / zero=none) |
                       smooth(x / after sm=50);
   output p;
run;

proc transreg; /* new */
   model identity(y) = class(group / zero=none) *
                       smooth(x / sm=50);
   output p;
run;
```

Previously, the AFTER *t-option* was required when you wanted to fit separate and independent functions within each group. This *t-option* specifies that PROC TRANSREG should find the smoothing spline transformations *after* it crosses the independent variable with the CLASS variable. Previously, by default, PROC TRANSREG found an overall smooth transformation and then crossed it with the CLASS variable, which is probably not what you want. You can still specify the AFTER *t-option*, but now it is assumed with CLASS * SMOOTH. If you specify AFTER without the NSR *a-option*, PROC TRANSREG suppresses the note that AFTER is assumed. It does not affect the model. If you do not want AFTER to be in effect by default, you must specify the NSR *a-option*. Also previously, you typically needed to specify the vertical bar instead of the asterisk to cross the CLASS and SMOOTH variables. The difference is that the bar adds both crossed variables and separate group intercepts to the model, whereas the asterisk adds only the crossed variables to the model. Since the SMOOTH transformation is now recognized as providing an implicit intercept, you should use the asterisk and not the vertical bar.

The default behavior of the SMOOTH transformation needed to change for several reasons. SMOOTH was originally provided as nothing more than a way to get PROC GPLOT's smoothing splines into an output data set in the transformed variables. However, with new enhancements to PROC TRANSREG such as ODS Graphics and PBSPLINE, the old method for SMOOTH did not fit well. The old method produced predicted values that were not the correct values to plot in order to show the smoothing spline fit. Now, with this change, ODS Graphics can always plot the predicted values. PBSPLINE and SMOOTH are similar in spirit, and for both, OLS results are not truly appropriate. Before SAS 9.2, PROC TRANSREG fit linear models, linear models with nonlinearly preprocessed variables, and linear models with optimal nonlinear transformations that minimized squared error. Now it also has the ability to fit non-OLS models for scatter plot smoothing.

One aspect of the SMOOTH transformation has unconditionally changed with SAS 9.2. Previously, PROC TRANSREG did not evaluate the effective degrees of freedom by examining the trace of the transformation hat matrix. It simply used the number of categories in the *df* calculations, which for continuous variables is the number of observations. This made it impossible to get a sensible ANOVA test for the overall fit. With SAS 9.2, the degrees of freedom are always based on the trace. This *df* change also affects the SSPLINE transformation, which finds a smooth transformation by using the same algorithm as SMOOTH. The difference is that the SMOOTH transformation occurs once, as an analysis preprocessing step, whereas SSPLINE transformations occur iteratively and in the body of the alternating least squares algorithm.

Iteration History Changes and Enhancements

With SAS 9.2, PROC TRANSREG no longer always prints an iteration history table by default, and in some cases, the table it prints is not the same as it was previously. This change is due to the increasing use of PROC

TRANSREG with transformations that are not based on alternating least squares. Here is some background for the change. PROC TRANSREG's processing can be divided into three steps. In the first step, the data are read and certain transformations, such as SMOOTH, PBSPLINE, BOXCOX, RANK, LOG and the other nonoptimal transformations, are performed. These transformations are not based on OLS. In the second step, the alternating least squares iterations are performed according to METHOD=UNIVARIATE, MORALS, REDUNDANCY, or CANALS. It is in the second step that the alternating least squares transformations (SPLINE, MSPLINE, MONOTONE, OPSCORE, LINEAR, and UNTIE) are iteratively found. In the third step, the results are displayed. In some cases, the results are appropriately based on using the method of OLS applied to the optimally transformed variables. In other cases, such as with smoothing splines and penalized B-splines, OLS-based results are not appropriate. Furthermore, for many of these types of models, nothing changes in the iterations, so the computations needed to realize that nothing changes are not needed, nor is the iteration history table.

With SAS 9.2, the iteration history is not printed for models where it is known that nothing will change in the iterations. Suppose the NOMISS option is specified or there are no missing data. If METHOD=UNIVARIATE, if there are no iterative transformations (SPLINE, MSPLINE, MONOTONE, OPSCORE, LINEAR, and UNTIE), and if the MAXITER= option is not specified, then by default, an iteration history table is not produced. If you want to see an iteration history, there are many things you can do, such as specifying MAXITER=, changing the method to MORALS, or changing IDENTITY to LINEAR.

With models with smoothing splines or penalized B-splines, the iteration history will not contain an R square. This is because the iterations are based on the method of alternating least squares, but the smoothing splines and penalized B-splines are not based on a least squares model. Hence, an ordinary R square in the iterations, based on a computed intercept, which is typically not exactly zero, and a computed slope, which is typically not exactly one, will not be exactly the same as the correct R square, which is based on an intercept and slope of zero and one. The final reported results include the correct R square in the fit statistics table after the ANOVA table. If you want to see only the correct R square from the results, without the iteration history, you can specify the new RSQUARE option.

ANOVA Codings

This section illustrates several different codings of classification variables and hence several different ways of fitting two-way ANOVA models to some data. Each example fits an ANOVA model, displays the ANOVA table and parameter estimates, and displays the coded design matrix. Note throughout that the ANOVA tables and R squares are identical for all of the models, showing that the codings are equivalent. For each model, the parameter estimates are stated as a function of the cell means. The formulas are appropriate for a design such as this one, which is balanced and orthogonal (every level and every pair of levels occurs equally often). They will not work with unequal frequencies. Since this data set has $3 \times 2 = 6$ cells, the full-rank codings all have six parameters. The following statements create the input data set, and display it in Figure 120.43:

```
title 'Two-Way ANOVA Models';

data x;
   input a b @@;
   do i = 1 to 2; input y @@; output; end;
   drop i;
   datalines;
1 1   16 14          1 2   15 13
2 1    1  9          2 2   12 20
```

```
3 1   14  8         3 2   18 20
;

proc print label;
run;
```

Figure 120.43 Input Data Set

Two-Way ANOVA Models

Obs	a	b	y
1	1	1	16
2	1	1	14
3	1	2	15
4	1	2	13
5	2	1	1
6	2	1	9
7	2	2	12
8	2	2	20
9	3	1	14
10	3	1	8
11	3	2	18
12	3	2	20

The following statements fit a cell-means model and produce Figure 120.44 and Figure 120.45:

```
proc transreg data=x ss2 short;
   title2 'Cell-Means Model';
   model identity(y) = class(a * b / zero=none);
   output replace;
run;

proc print label;
run;
```

Figure 120.44 Cell-Means Model

Two-Way ANOVA Models
Cell-Means Model

The TRANSREG Procedure

Dependent Variable Identity(y)

Class Level Information		
Class	**Levels**	**Values**
a	3	1 2 3
b	2	1 2

Number of Observations Read	12
Number of Observations Used	12
Implicit Intercept Model	

Figure 120.44 *continued*

The TRANSREG Procedure Hypothesis Tests for Identity(y)

Univariate ANOVA Table Based on the Usual Degrees of Freedom

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	5	234.6667	46.93333	3.20	0.0946
Error	6	88.0000	14.66667		
Corrected Total	11	322.6667			

Root MSE	3.82971	**R-Square**	0.7273
Dependent Mean	13.33333	**Adj R-Sq**	0.5000
Coeff Var	28.72281		

Univariate Regression Table Based on the Usual Degrees of Freedom

Variable	DF	Coefficient	Type II Sum of Squares	Mean Square	F Value	Pr > F	Label
Class.a1b1	1	15.0000000	450.000	450.000	30.68	0.0015	a 1 * b 1
Class.a1b2	1	14.0000000	392.000	392.000	26.73	0.0021	a 1 * b 2
Class.a2b1	1	5.0000000	50.000	50.000	3.41	0.1144	a 2 * b 1
Class.a2b2	1	16.0000000	512.000	512.000	34.91	0.0010	a 2 * b 2
Class.a3b1	1	11.0000000	242.000	242.000	16.50	0.0066	a 3 * b 1
Class.a3b2	1	19.0000000	722.000	722.000	49.23	0.0004	a 3 * b 2

The parameter estimates are

$$
\begin{aligned}
\hat{\mu}_{11} &= \overline{y}_{11} = 15 \\
\hat{\mu}_{12} &= \overline{y}_{12} = 14 \\
\hat{\mu}_{21} &= \overline{y}_{21} = 5 \\
\hat{\mu}_{22} &= \overline{y}_{22} = 16 \\
\hat{\mu}_{31} &= \overline{y}_{31} = 11 \\
\hat{\mu}_{32} &= \overline{y}_{32} = 19
\end{aligned}
$$

Figure 120.45 Cell-Means Model, Design Matrix

Two-Way ANOVA Models
Cell-Means Model

Obs	_TYPE_	_NAME_	y	Intercept	a 1 * b 1	a 1 * b 2	a 2 * b 1	a 2 * b 2	a 3 * b 1	a 3 * b 2	a	b
1	SCORE	ROW1	16	.	1	0	0	0	0	0	1	1
2	SCORE	ROW2	14	.	1	0	0	0	0	0	1	1
3	SCORE	ROW3	15	.	0	1	0	0	0	0	1	2
4	SCORE	ROW4	13	.	0	1	0	0	0	0	1	2
5	SCORE	ROW5	1	.	0	0	1	0	0	0	2	1
6	SCORE	ROW6	9	.	0	0	1	0	0	0	2	1
7	SCORE	ROW7	12	.	0	0	0	1	0	0	2	2
8	SCORE	ROW8	20	.	0	0	0	1	0	0	2	2
9	SCORE	ROW9	14	.	0	0	0	0	1	0	3	1
10	SCORE	ROW10	8	.	0	0	0	0	1	0	3	1
11	SCORE	ROW11	18	.	0	0	0	0	0	1	3	2
12	SCORE	ROW12	20	.	0	0	0	0	0	1	3	2

The next model is a reference cell model, and the default reference cell is the last cell, which in this case is the (3,2) cell. The following statements fit a reference cell model and produce Figure 120.46 and Figure 120.47:

```
proc transreg data=x ss2 short;
   title2 'Reference Cell Model, (3,2) Reference Cell';
   model identity(y) = class(a | b);
   output replace;
run;

proc print label;
run;
```

Figure 120.46 Reference Cell Model, (3,2) Reference Cell

Two-Way ANOVA Models
Reference Cell Model, (3,2) Reference Cell

The TRANSREG Procedure

Dependent Variable Identity(y)

Class Level Information

Class	Levels	Values
a	3	1 2 3
b	2	1 2

Number of Observations Read	12
Number of Observations Used	12

Figure 120.46 *continued*

The TRANSREG Procedure Hypothesis Tests for Identity(y)

Univariate ANOVA Table Based on the Usual Degrees of Freedom					
Source	**DF**	**Sum of Squares**	**Mean Square**	**F Value**	**Pr > F**
Model	5	234.6667	46.93333	3.20	0.0946
Error	6	88.0000	14.66667		
Corrected Total	11	322.6667			

Root MSE	3.82971	**R-Square**	0.7273
Dependent Mean	13.33333	**Adj R-Sq**	0.5000
Coeff Var	28.72281		

Univariate Regression Table Based on the Usual Degrees of Freedom							
Variable	**DF**	**Coefficient**	**Type II Sum of Squares**	**Mean Square**	**F Value**	**Pr > F**	**Label**
Intercept	1	19.0000000	722.000	722.000	49.23	0.0004	Intercept
Class.a1	1	-5.0000000	25.000	25.000	1.70	0.2395	a 1
Class.a2	1	-3.0000000	9.000	9.000	0.61	0.4632	a 2
Class.b1	1	-8.0000000	64.000	64.000	4.36	0.0817	b 1
Class.a1b1	1	9.0000000	40.500	40.500	2.76	0.1476	a 1 * b 1
Class.a2b1	1	-3.0000000	4.500	4.500	0.31	0.5997	a 2 * b 1

The parameter estimates are

$$
\begin{aligned}
\hat{\mu}_{32} &= \overline{y}_{32} = 19 \\
\hat{\alpha}_1 &= \overline{y}_{12} - \overline{y}_{32} = 14 - 19 = -5 \\
\hat{\alpha}_2 &= \overline{y}_{22} - \overline{y}_{32} = 16 - 19 = -3 \\
\hat{\beta}_1 &= \overline{y}_{31} - \overline{y}_{32} = 11 - 19 = -8 \\
\hat{\gamma}_{11} &= \overline{y}_{11} - (\hat{\mu}_{32} + \hat{\alpha}_1 + \hat{\beta}_1) = 15 - (19 + -5 + -8) = 9 \\
\hat{\gamma}_{21} &= \overline{y}_{21} - (\hat{\mu}_{32} + \hat{\alpha}_2 + \hat{\beta}_1) = 5 - (19 + -3 + -8) = -3
\end{aligned}
$$

Figure 120.47 Reference Cell Model, (3,2) Reference Cell, Design Matrix

Two-Way ANOVA Models
Reference Cell Model, (3,2) Reference Cell

Obs	_TYPE_	_NAME_	y	Intercept	a 1	a 2	b 1	a 1 * b 1	a 2 * b 1	a	b
1	SCORE	ROW1	16	1	1	0	1	1	0	1	1
2	SCORE	ROW2	14	1	1	0	1	1	0	1	1
3	SCORE	ROW3	15	1	1	0	0	0	0	1	2
4	SCORE	ROW4	13	1	1	0	0	0	0	1	2
5	SCORE	ROW5	1	1	0	1	1	0	1	2	1
6	SCORE	ROW6	9	1	0	1	1	0	1	2	1
7	SCORE	ROW7	12	1	0	1	0	0	0	2	2
8	SCORE	ROW8	20	1	0	1	0	0	0	2	2
9	SCORE	ROW9	14	1	0	0	1	0	0	3	1
10	SCORE	ROW10	8	1	0	0	1	0	0	3	1
11	SCORE	ROW11	18	1	0	0	0	0	0	3	2
12	SCORE	ROW12	20	1	0	0	0	0	0	3	2

The next model is a deviations-from-means model. This coding is also called effects coding. The default reference cell is the last cell (3,2). The following statements produce Figure 120.48 and Figure 120.49:

```
proc transreg data=x ss2 short;
   title2 'Deviations from Means, (3,2) Reference Cell';
   model identity(y) = class(a | b / deviations);
   output replace;
run;

proc print label;
run;
```

Figure 120.48 Deviations-from-Means Model, (3,2) Reference Cell

Two-Way ANOVA Models
Deviations from Means, (3,2) Reference Cell

The TRANSREG Procedure

Dependent Variable Identity(y)

Class Level Information		
Class	**Levels**	**Values**
a	3	1 2 3
b	2	1 2

Number of Observations Read	12
Number of Observations Used	12

Figure 120.48 *continued*

The TRANSREG Procedure Hypothesis Tests for Identity(y)

Univariate ANOVA Table Based on the Usual Degrees of Freedom					
Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	5	234.6667	46.93333	3.20	0.0946
Error	6	88.0000	14.66667		
Corrected Total	11	322.6667			

Root MSE	3.82971	**R-Square**	0.7273
Dependent Mean	13.33333	**Adj R-Sq**	0.5000
Coeff Var	28.72281		

Univariate Regression Table Based on the Usual Degrees of Freedom							
Variable	DF	Coefficient	Type II Sum of Squares	Mean Square	F Value	Pr > F	Label
Intercept	1	13.3333333	2133.33	2133.33	145.45	<.0001	Intercept
Class.a1	1	1.1666667	8.17	8.17	0.56	0.4837	a 1
Class.a2	1	-2.8333333	48.17	48.17	3.28	0.1199	a 2
Class.b1	1	-3.0000000	108.00	108.00	7.36	0.0349	b 1
Class.a1b1	1	3.5000000	73.50	73.50	5.01	0.0665	a 1 * b 1
Class.a2b1	1	-2.5000000	37.50	37.50	2.56	0.1609	a 2 * b 1

The parameter estimates are

$$
\begin{aligned}
\hat{\mu} &= \overline{y} = 13.3333 \\
\hat{\alpha}_1 &= (\overline{y}_{11} + \overline{y}_{12})/2 - \overline{y} = (15 + 14)/2 - 13.3333 = 1.1667 \\
\hat{\alpha}_2 &= (\overline{y}_{21} + \overline{y}_{22})/2 - \overline{y} = (5 + 16)/2 - 13.3333 = -2.8333 \\
\hat{\beta}_1 &= (\overline{y}_{11} + \overline{y}_{21} + \overline{y}_{31})/3 - \overline{y} = (15 + 5 + 11)/3 - 13.3333 = -3 \\
\hat{\gamma}_{11} &= \overline{y}_{11} - (\overline{y} + \hat{\alpha}_1 + \hat{\beta}_1) = 15 - (13.3333 + 1.1667 + -3) = 3.5 \\
\hat{\gamma}_{21} &= \overline{y}_{21} - (\overline{y} + \hat{\alpha}_2 + \hat{\beta}_1) = 5 - (13.3333 + -2.8333 + -3) = -2.5
\end{aligned}
$$

Figure 120.49 Deviations-from-Means Model, (3,2) Reference Cell, Design Matrix

Two-Way ANOVA Models
Deviations from Means, (3,2) Reference Cell

Obs	_TYPE_	_NAME_	y	Intercept	a 1	a 2	b 1	a 1*b 1	a 2*b 1	a	b
1	SCORE	ROW1	16	1	1	0	1	1	0	1	1
2	SCORE	ROW2	14	1	1	0	1	1	0	1	1
3	SCORE	ROW3	15	1	1	0	-1	-1	0	1	2
4	SCORE	ROW4	13	1	1	0	-1	-1	0	1	2
5	SCORE	ROW5	1	1	0	1	1	0	1	2	1
6	SCORE	ROW6	9	1	0	1	1	0	1	2	1
7	SCORE	ROW7	12	1	0	1	-1	0	-1	2	2
8	SCORE	ROW8	20	1	0	1	-1	0	-1	2	2
9	SCORE	ROW9	14	1	-1	-1	1	-1	-1	3	1
10	SCORE	ROW10	8	1	-1	-1	1	-1	-1	3	1
11	SCORE	ROW11	18	1	-1	-1	-1	1	1	3	2
12	SCORE	ROW12	20	1	-1	-1	-1	1	1	3	2

The next model is a less-than-full-rank model. The parameter estimates are constrained to sum to zero within each effect. The following statements produce Figure 120.50 and Figure 120.51:

```
proc transreg data=x ss2 short;
   title2 'Less-Than-Full-Rank Model';
   model identity(y) = class(a | b / zero=sum);
   output replace;
run;

proc print label;
run;
```

Figure 120.50 Less-Than-Full-Rank Model

Two-Way ANOVA Models
Less-Than-Full-Rank Model

The TRANSREG Procedure

Dependent Variable Identity(y)

Class Level Information		
Class	**Levels**	**Values**
a	3	1 2 3
b	2	1 2

Number of Observations Read	12
Number of Observations Used	12

Figure 120.50 *continued*

The TRANSREG Procedure Hypothesis Tests for Identity(y)

Univariate ANOVA Table Based on the Usual Degrees of Freedom

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	5	234.6667	46.93333	3.20	0.0946
Error	6	88.0000	14.66667		
Corrected Total	11	322.6667			

Root MSE	3.82971	**R-Square**	0.7273
Dependent Mean	13.33333	**Adj R-Sq**	0.5000
Coeff Var	28.72281		

Univariate Regression Table Based on the Usual Degrees of Freedom

Variable	DF	Coefficient	Type II Sum of Squares	Mean Square	F Value	Pr > F	Label
Intercept	1	13.3333333	2133.33	2133.33	145.45	<.0001	Intercept
Class.a1	1	1.1666667	8.17	8.17	0.56	0.4837	a 1
Class.a2	1	-2.8333333	48.17	48.17	3.28	0.1199	a 2
Class.a3	1	1.6666667	16.67	16.67	1.14	0.3274	a 3
Class.b1	1	-3.0000000	108.00	108.00	7.36	0.0349	b 1
Class.b2	1	3.0000000	108.00	108.00	7.36	0.0349	b 2
Class.a1b1	1	3.5000000	73.50	73.50	5.01	0.0665	a 1 * b 1
Class.a1b2	1	-3.5000000	73.50	73.50	5.01	0.0665	a 1 * b 2
Class.a2b1	1	-2.5000000	37.50	37.50	2.56	0.1609	a 2 * b 1
Class.a2b2	1	2.5000000	37.50	37.50	2.56	0.1609	a 2 * b 2
Class.a3b1	1	-1.0000000	6.00	6.00	0.41	0.5461	a 3 * b 1
Class.a3b2	1	1.0000000	6.00	6.00	0.41	0.5461	a 3 * b 2

The sum of the regression table DF's, minus one for the intercept, will be greater than the model df when there are ZERO=SUM constraints.

The parameter estimates are

$$\begin{aligned}
\hat{\mu} &= \overline{y} = 13.3333 \\
\hat{\alpha}_1 &= (\overline{y}_{11} + \overline{y}_{12})/2 - \overline{y} = (15 + 14)/2 - 13.3333 = 1.1667 \\
\hat{\alpha}_2 &= (\overline{y}_{21} + \overline{y}_{22})/2 - \overline{y} = (5 + 16)/2 - 13.3333 = -2.8333 \\
\hat{\alpha}_3 &= (\overline{y}_{31} + \overline{y}_{32})/2 - \overline{y} = (11 + 19)/2 - 13.3333 = 1.6667 \\
\hat{\beta}_1 &= (\overline{y}_{11} + \overline{y}_{21} + \overline{y}_{31})/3 - \overline{y} = (15 + 5 + 11)/3 - 13.3333 = -3 \\
\hat{\beta}_2 &= (\overline{y}_{12} + \overline{y}_{22} + \overline{y}_{32})/3 - \overline{y} = (14 + 16 + 19)/3 - 13.3333 = 3 \\
\hat{\gamma}_{11} &= \overline{y}_{11} - (\overline{y} + \hat{\alpha}_1 + \hat{\beta}_1) = 15 - (13.3333 + 1.1667 + -3) = 3.5 \\
\hat{\gamma}_{12} &= \overline{y}_{12} - (\overline{y} + \hat{\alpha}_1 + \hat{\beta}_2) = 14 - (13.3333 + 1.1667 + 3) = -3.5 \\
\hat{\gamma}_{21} &= \overline{y}_{21} - (\overline{y} + \hat{\alpha}_2 + \hat{\beta}_1) = 5 - (13.3333 + -2.8333 + -3) = -2.5 \\
\hat{\gamma}_{22} &= \overline{y}_{22} - (\overline{y} + \hat{\alpha}_2 + \hat{\beta}_2) = 16 - (13.3333 + -2.8333 + 3) = 2.5 \\
\hat{\gamma}_{31} &= \overline{y}_{31} - (\overline{y} + \hat{\alpha}_3 + \hat{\beta}_1) = 11 - (13.3333 + 1.6667 + -3) = -1 \\
\hat{\gamma}_{32} &= \overline{y}_{32} - (\overline{y} + \hat{\alpha}_3 + \hat{\beta}_2) = 19 - (13.3333 + 1.6667 + 3) = 1
\end{aligned}$$

The constraints are

$$\alpha_1 + \alpha_2 + \alpha_3 \equiv \beta_1 + \beta_2 \equiv 0$$

$$\gamma_{11} + \gamma_{12} \equiv \gamma_{21} + \gamma_{22} \equiv \gamma_{31} + \gamma_{32} \equiv \gamma_{11} + \gamma_{21} + \gamma_{31} \equiv \gamma_{12} + \gamma_{22} + \gamma_{32} \equiv 0$$

Only four of the five interaction constraints are needed. The fifth constraint is implied by the other four. (Given a 2×3 table with four marginal sum-to-zero constraints, you can freely fill in only two cells. The values in the other four cells are determined from the first two cells and the constraints.) A full-rank model has six estimable parameters. This less-than-full-rank model has one parameter for the intercept, two for the first main effect (plus one more as determined by the first constraint), one for the second main effect (plus one more as determined by the second constraint), and two for the interactions (plus four more as determined by the next four constraints). Six of the twelve parameters are determined given the other six and the constraints. Notice that $\hat{\mu}, \hat{\alpha}_1, \hat{\alpha}_2, \hat{\beta}_1, \hat{\gamma}_{11}$, and $\hat{\gamma}_{21}$ match the corresponding estimates from the effects coding.

Figure 120.51 Less-Than-Full-Rank Model, Design Matrix

Two-Way ANOVA Models
Less-Than-Full-Rank Model

Obs	_TYPE_	_NAME_	y	Intercept	a 1	a 2	a 3	b 1	b 2	a 1 * b 1	a 1 * b 2	a 2 * b 1	a 2 * b 2	a 3 * b 1	a 3 * b 2	a	b
1	SCORE	ROW1	16	1	1	0	0	1	0	1	0	0	0	0	0	1	1
2	SCORE	ROW2	14	1	1	0	0	1	0	1	0	0	0	0	0	1	1
3	SCORE	ROW3	15	1	1	0	0	0	1	0	1	0	0	0	0	1	2
4	SCORE	ROW4	13	1	1	0	0	0	1	0	1	0	0	0	0	1	2
5	SCORE	ROW5	1	1	0	1	0	1	0	0	0	1	0	0	0	2	1
6	SCORE	ROW6	9	1	0	1	0	1	0	0	0	1	0	0	0	2	1
7	SCORE	ROW7	12	1	0	1	0	0	1	0	0	0	1	0	0	2	2
8	SCORE	ROW8	20	1	0	1	0	0	1	0	0	0	1	0	0	2	2
9	SCORE	ROW9	14	1	0	0	1	1	0	0	0	0	0	1	0	3	1
10	SCORE	ROW10	8	1	0	0	1	1	0	0	0	0	0	1	0	3	1
11	SCORE	ROW11	18	1	0	0	1	0	1	0	0	0	0	0	1	3	2
12	SCORE	ROW12	20	1	0	0	1	0	1	0	0	0	0	0	1	3	2

The next model is a reference cell model, but this time the reference cell is the first cell (1,1). The following statements produce Figure 120.52 and Figure 120.53:

```
proc transreg data=x ss2 short;
   title2 'Reference Cell Model, (1,1) Reference Cell';
   model identity(y) = class(a | b / zero=first);
   output replace;
run;

proc print label;
run;
```

Figure 120.52 Reference Cell Model, (1,1) Reference Cell

Two-Way ANOVA Models
Reference Cell Model, (1,1) Reference Cell

The TRANSREG Procedure

Dependent Variable Identity(y)

Class Level Information		
Class	**Levels**	**Values**
a	3	1 2 3
b	2	1 2

Number of Observations Read	12
Number of Observations Used	12

Figure 120.52 *continued*

The TRANSREG Procedure Hypothesis Tests for Identity(y)

Univariate ANOVA Table Based on the Usual Degrees of Freedom					
Source	**DF**	**Sum of Squares**	**Mean Square**	**F Value**	**Pr > F**
Model	5	234.6667	46.93333	3.20	0.0946
Error	6	88.0000	14.66667		
Corrected Total	11	322.6667			

Root MSE	3.82971	**R-Square**	0.7273
Dependent Mean	13.33333	**Adj R-Sq**	0.5000
Coeff Var	28.72281		

Univariate Regression Table Based on the Usual Degrees of Freedom							
Variable	**DF**	**Coefficient**	**Type II Sum of Squares**	**Mean Square**	**F Value**	**Pr > F**	**Label**
Intercept	1	15.000000	450.000	450.000	30.68	0.0015	Intercept
Class.a2	1	-10.000000	100.000	100.000	6.82	0.0401	a 2
Class.a3	1	-4.000000	16.000	16.000	1.09	0.3365	a 3
Class.b2	1	-1.000000	1.000	1.000	0.07	0.8027	b 2
Class.a2b2	1	12.000000	72.000	72.000	4.91	0.0686	a 2 * b 2
Class.a3b2	1	9.000000	40.500	40.500	2.76	0.1476	a 3 * b 2

The parameter estimates are

$$
\begin{aligned}
\hat{\mu}_{11} &= \overline{y}_{11} = 15 \\
\hat{\alpha}_2 &= \overline{y}_{21} - \overline{y}_{11} = 5 - 15 = -10 \\
\hat{\alpha}_3 &= \overline{y}_{31} - \overline{y}_{11} = 11 - 15 = -4 \\
\hat{\beta}_2 &= \overline{y}_{12} - \overline{y}_{11} = 14 - 15 = -1 \\
\hat{\gamma}_{22} &= \overline{y}_{22} - (\hat{\mu}_{11} + \hat{\alpha}_2 + \hat{\beta}_2) = 16 - (15 + -10 + -1) = 12 \\
\hat{\gamma}_{32} &= \overline{y}_{32} - (\hat{\mu}_{11} + \hat{\alpha}_3 + \hat{\beta}_2) = 19 - (15 + -4 + -1) = 9
\end{aligned}
$$

Figure 120.53 Reference Cell Model, (1,1) Reference Cell, Design Matrix

Two-Way ANOVA Models
Reference Cell Model, (1,1) Reference Cell

Obs	_TYPE_	_NAME_	y	Intercept	a 2	a 3	b 2	a 2 * b 2	a 3 * b 2	a	b
1	SCORE	ROW1	16	1	0	0	0	0	0	1	1
2	SCORE	ROW2	14	1	0	0	0	0	0	1	1
3	SCORE	ROW3	15	1	0	0	1	0	0	1	2
4	SCORE	ROW4	13	1	0	0	1	0	0	1	2
5	SCORE	ROW5	1	1	1	0	0	0	0	2	1
6	SCORE	ROW6	9	1	1	0	0	0	0	2	1
7	SCORE	ROW7	12	1	1	0	1	1	0	2	2
8	SCORE	ROW8	20	1	1	0	1	1	0	2	2
9	SCORE	ROW9	14	1	0	1	0	0	0	3	1
10	SCORE	ROW10	8	1	0	1	0	0	0	3	1
11	SCORE	ROW11	18	1	0	1	1	0	1	3	2
12	SCORE	ROW12	20	1	0	1	1	0	1	3	2

The next model is a deviations-from-means model, but this time the reference cell is the first cell (1,1). This coding is also called effects coding. The following statements produce Figure 120.54 and Figure 120.55:

```
proc transreg data=x ss2 short;
   title2 'Deviations from Means, (1,1) Reference Cell';
   model identity(y) = class(a | b / deviations zero=first);
   output replace;
run;

proc print label;
run;
```

Figure 120.54 Deviations-from-Means Model, (1,1) Reference Cell

Two-Way ANOVA Models
Deviations from Means, (1,1) Reference Cell

The TRANSREG Procedure

Dependent Variable Identity(y)

Class Level Information		
Class	**Levels**	**Values**
a	3	1 2 3
b	2	1 2

Number of Observations Read	12
Number of Observations Used	12

Figure 120.54 *continued*

The TRANSREG Procedure Hypothesis Tests for Identity(y)

Univariate ANOVA Table Based on the Usual Degrees of Freedom

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	5	234.6667	46.93333	3.20	0.0946
Error	6	88.0000	14.66667		
Corrected Total	11	322.6667			

Root MSE	3.82971	**R-Square**	0.7273
Dependent Mean	13.33333	**Adj R-Sq**	0.5000
Coeff Var	28.72281		

Univariate Regression Table Based on the Usual Degrees of Freedom

Variable	DF	Coefficient	Type II Sum of Squares	Mean Square	F Value	Pr > F	Label
Intercept	1	13.3333333	2133.33	2133.33	145.45	<.0001	Intercept
Class.a2	1	-2.8333333	48.17	48.17	3.28	0.1199	a 2
Class.a3	1	1.6666667	16.67	16.67	1.14	0.3274	a 3
Class.b2	1	3.0000000	108.00	108.00	7.36	0.0349	b 2
Class.a2b2	1	2.5000000	37.50	37.50	2.56	0.1609	a 2 * b 2
Class.a3b2	1	1.0000000	6.00	6.00	0.41	0.5461	a 3 * b 2

The parameter estimates are

$$
\begin{aligned}
\hat{\mu} &= \overline{y} = 13.3333 \\
\hat{\alpha}_2 &= (\overline{y}_{21} + \overline{y}_{22})/2 - \overline{y} = (5 + 16)/2 - 13.3333 = -2.8333 \\
\hat{\alpha}_3 &= (\overline{y}_{31} + \overline{y}_{32})/2 - \overline{y} = (11 + 19)/2 - 13.3333 = 1.6667 \\
\hat{\beta}_2 &= (\overline{y}_{12} + \overline{y}_{22} + \overline{y}_{32})/3 - \overline{y} = (14 + 16 + 19)/3 - 13.3333 = 3 \\
\hat{\gamma}_{22} &= \overline{y}_{22} - (\overline{y} + \hat{\alpha}_2 + \hat{\beta}_2) = 16 - (13.3333 + -2.8333 + 3) = 2.5 \\
\hat{\gamma}_{32} &= \overline{y}_{32} - (\overline{y} + \hat{\alpha}_3 + \hat{\beta}_2) = 19 - (13.3333 + 1.6667 + 3) = 1
\end{aligned}
$$

Notice that all of the parameter estimates match the corresponding estimates from the less-than-full-rank coding.

Figure 120.55 Deviations-from-Means Model, (1,1) Reference Cell, Design Matrix

Two-Way ANOVA Models
Deviations from Means, (1,1) Reference Cell

Obs	_TYPE_	_NAME_	y	Intercept	a 2	a 3	b 2	a 2 * b 2	a 3 * b 2	a	b
1	SCORE	ROW1	16	1	-1	-1	-1	1	1	1	1
2	SCORE	ROW2	14	1	-1	-1	-1	1	1	1	1
3	SCORE	ROW3	15	1	-1	-1	1	-1	-1	1	2
4	SCORE	ROW4	13	1	-1	-1	1	-1	-1	1	2
5	SCORE	ROW5	1	1	1	0	-1	-1	0	2	1
6	SCORE	ROW6	9	1	1	0	-1	-1	0	2	1
7	SCORE	ROW7	12	1	1	0	1	1	0	2	2
8	SCORE	ROW8	20	1	1	0	1	1	0	2	2
9	SCORE	ROW9	14	1	0	1	-1	0	-1	3	1
10	SCORE	ROW10	8	1	0	1	-1	0	-1	3	1
11	SCORE	ROW11	18	1	0	1	1	0	1	3	2
12	SCORE	ROW12	20	1	0	1	1	0	1	3	2

The following statements fit a model with an orthogonal-contrast coding and produce Figure 120.56 and Figure 120.57:

```
proc transreg data=x ss2 short;
   title2 'Orthogonal Contrast Coding';
   model identity(y) = class(a | b / orthogonal);
   output replace;
run;

proc print label;
run;
```

Figure 120.56 Orthogonal-Contrast Coding

Two-Way ANOVA Models
Orthogonal Contrast Coding

The TRANSREG Procedure

Dependent Variable Identity(y)

Class Level Information		
Class	**Levels**	**Values**
a	3	1 2 3
b	2	1 2

Number of Observations Read	12
Number of Observations Used	12

Figure 120.56 *continued*

The TRANSREG Procedure Hypothesis Tests for Identity(y)

Univariate ANOVA Table Based on the Usual Degrees of Freedom

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	5	234.6667	46.93333	3.20	0.0946
Error	6	88.0000	14.66667		
Corrected Total	11	322.6667			

Root MSE	3.82971	**R-Square**	0.7273
Dependent Mean	13.33333	**Adj R-Sq**	0.5000
Coeff Var	28.72281		

Univariate Regression Table Based on the Usual Degrees of Freedom

Variable	DF	Coefficient	Type II Sum of Squares	Mean Square	F Value	Pr > F	Label
Intercept	1	13.3333333	2133.33	2133.33	145.45	<.0001	Intercept
Class.a1	1	-0.2500000	0.50	0.50	0.03	0.8596	a 1
Class.a2	1	-1.4166667	48.17	48.17	3.28	0.1199	a 2
Class.b1	1	-3.0000000	108.00	108.00	7.36	0.0349	b 1
Class.a1b1	1	2.2500000	40.50	40.50	2.76	0.1476	a 1 * b 1
Class.a2b1	1	-1.2500000	37.50	37.50	2.56	0.1609	a 2 * b 1

The parameter estimates are

$$\begin{aligned}
\hat{\mu} &= \overline{y} = 13.3333 \\
\hat{\alpha}_1 &= ((\overline{y}_{11} + \overline{y}_{12}) - (\overline{y}_{31} + \overline{y}_{32}))/4 = ((15 + 14) - (11 + 19))/4 = -0.25 \\
\hat{\alpha}_2 &= ((\overline{y}_{21} + \overline{y}_{22}) - (\overline{y}_{11} + \overline{y}_{12} + \overline{y}_{31} + \overline{y}_{32})/2)/6 \\
&= ((5 + 16) - (15 + 14 + 11 + 19)/2)/6 = -1.417 \\
\hat{\beta}_1 &= ((\overline{y}_{11} + \overline{y}_{21} + \overline{y}_{31}) - (\overline{y}_{12} + \overline{y}_{22} + \overline{y}_{32}))/6 \\
&= ((15 + 5 + 11) - (14 + 16 + 19))/6 = -3 \\
\hat{\gamma}_{11} &= (\overline{y}_{11} - \overline{y}_{12} - \overline{y}_{31} + \overline{y}_{32})/4 = (15 - 14 - 11 + 19)/4 = 2.25 \\
\hat{\gamma}_{21} &= ((-\overline{y}_{11} + \overline{y}_{12} - \overline{y}_{31} + \overline{y}_{32})/2 + (\overline{y}_{21} - \overline{y}_{22}))/6 \\
&= ((-15 + 14 - 11 + 19)/2 + (5 - 16))/6 = -1.25
\end{aligned}$$

Figure 120.57 Orthogonal-Contrast Coding, Design Matrix

Two-Way ANOVA Models
Orthogonal Contrast Coding

Obs	_TYPE_	_NAME_	y	Intercept	a 1	a 2	b 1	a 1 * b 1	a 2 * b 1	a	b
1	SCORE	ROW1	16	1	1	-1	1	1	-1	1	1
2	SCORE	ROW2	14	1	1	-1	1	1	-1	1	1
3	SCORE	ROW3	15	1	1	-1	-1	-1	1	1	2
4	SCORE	ROW4	13	1	1	-1	-1	-1	1	1	2
5	SCORE	ROW5	1	1	0	2	1	0	2	2	1
6	SCORE	ROW6	9	1	0	2	1	0	2	2	1
7	SCORE	ROW7	12	1	0	2	-1	0	-2	2	2
8	SCORE	ROW8	20	1	0	2	-1	0	-2	2	2
9	SCORE	ROW9	14	1	-1	-1	1	-1	-1	3	1
10	SCORE	ROW10	8	1	-1	-1	1	-1	-1	3	1
11	SCORE	ROW11	18	1	-1	-1	-1	1	1	3	2
12	SCORE	ROW12	20	1	-1	-1	-1	1	1	3	2

The following statements fit a model with a standardized-orthogonal coding and produce Figure 120.58 and Figure 120.59:

```
proc transreg data=x ss2 short;
   title2 'Standardized-Orthogonal Coding';
   model identity(y) = class(a | b / standorth);
   output replace;
run;

proc print label;
run;
```

Figure 120.58 Standardized-Orthogonal Coding

Two-Way ANOVA Models
Standardized-Orthogonal Coding

The TRANSREG Procedure

Dependent Variable Identity(y)

Class Level Information		
Class	**Levels**	**Values**
a	3	1 2 3
b	2	1 2

Number of Observations Read	12
Number of Observations Used	12

Figure 120.58 *continued*

The TRANSREG Procedure Hypothesis Tests for Identity(y)

Univariate ANOVA Table Based on the Usual Degrees of Freedom

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	5	234.6667	46.93333	3.20	0.0946
Error	6	88.0000	14.66667		
Corrected Total	11	322.6667			

Root MSE	3.82971	**R-Square**	0.7273
Dependent Mean	13.33333	**Adj R-Sq**	0.5000
Coeff Var	28.72281		

Univariate Regression Table Based on the Usual Degrees of Freedom

Variable	DF	Coefficient	Type II Sum of Squares	Mean Square	F Value	Pr > F	Label
Intercept	1	13.3333333	2133.33	2133.33	145.45	<.0001	Intercept
Class.a1	1	-0.2041241	0.50	0.50	0.03	0.8596	a 1
Class.a2	1	-2.0034692	48.17	48.17	3.28	0.1199	a 2
Class.b1	1	-3.0000000	108.00	108.00	7.36	0.0349	b 1
Class.a1b1	1	1.8371173	40.50	40.50	2.76	0.1476	a 1 * b 1
Class.a2b1	1	-1.7677670	37.50	37.50	2.56	0.1609	a 2 * b 1

The parameter estimates are

$$
\begin{aligned}
\hat{\mu} &= \overline{y} = 13.3333 \\
\hat{\alpha}_1 &= (((\overline{y}_{11} + \overline{y}_{12}) - (\overline{y}_{31} + \overline{y}_{32}))/4) \times \sqrt{2/3} \\
&= (((15 + 14) - (11 + 19))/4) \times \sqrt{2/3} = -0.2041 \\
\hat{\alpha}_2 &= (((\overline{y}_{21} + \overline{y}_{22}) - (\overline{y}_{11} + \overline{y}_{12} + \overline{y}_{31} + \overline{y}_{32})/2)/6) \times \sqrt{6/3} \\
&= (((5 + 16) - (15 + 14 + 11 + 19)/2)/6) \times \sqrt{6/3} = -2.0035 \\
\hat{\beta}_1 &= (((\overline{y}_{11} + \overline{y}_{21} + \overline{y}_{31}) - (\overline{y}_{12} + \overline{y}_{22} + \overline{y}_{32}))/6) \times \sqrt{2/2} \\
&= (((15 + 5 + 11) - (14 + 16 + 19))/6) \times \sqrt{2/2} = -3 \\
\hat{\gamma}_{11} &= ((\overline{y}_{11} - \overline{y}_{12} - \overline{y}_{31} + \overline{y}_{32})/4) \times \sqrt{2/3} \times \sqrt{2/2} \\
&= ((15 - 14 - 11 + 19)/4) \times \sqrt{2/3} \times \sqrt{2/2} = 1.8371 \\
\hat{\gamma}_{21} &= (((-\overline{y}_{11} + \overline{y}_{12} - \overline{y}_{31} + \overline{y}_{32})/2 + (\overline{y}_{21} - \overline{y}_{22}))/6) \times \sqrt{6/3} \times \sqrt{2/2} \\
&= (((-15 + 14 - 11 + 19)/2 + (5 - 16))/6) \times \sqrt{6/3} \times \sqrt{2/2} = -1.7678
\end{aligned}
$$

The numerators in the square roots are sums of squares of the coded values for the unstandardized-orthogonal codings, and the denominators are the numbers of levels. These terms convert the estimates from the orthogonal contrast coding to the standardized-orthogonal coding. The term $\sqrt{2/2}$, which is 1 and could be dropped, is included in the preceding formulas to show the general pattern. Notice the regression tables for the orthogonal-contrast coding and the standardized-orthogonal coding. Some of the coefficients are different, but the rest of the table is the same since the coded variables for the two models differ only by a constant.

Figure 120.59 Standardized-Orthogonal Coding, Design Matrix

Two-Way ANOVA Models
Standardized-Orthogonal Coding

Obs	_TYPE_	_NAME_	y	Intercept	a 1	a 2	b 1	a 1 * b 1	a 2 * b 1	a	b
1	SCORE	ROW1	16	1	1.22474	-0.70711	1	1.22474	-0.70711	1	1
2	SCORE	ROW2	14	1	1.22474	-0.70711	1	1.22474	-0.70711	1	1
3	SCORE	ROW3	15	1	1.22474	-0.70711	-1	-1.22474	0.70711	1	2
4	SCORE	ROW4	13	1	1.22474	-0.70711	-1	-1.22474	0.70711	1	2
5	SCORE	ROW5	1	1	0.00000	1.41421	1	0.00000	1.41421	2	1
6	SCORE	ROW6	9	1	0.00000	1.41421	1	0.00000	1.41421	2	1
7	SCORE	ROW7	12	1	0.00000	1.41421	-1	0.00000	-1.41421	2	2
8	SCORE	ROW8	20	1	0.00000	1.41421	-1	0.00000	-1.41421	2	2
9	SCORE	ROW9	14	1	-1.22474	-0.70711	1	-1.22474	-0.70711	3	1
10	SCORE	ROW10	8	1	-1.22474	-0.70711	1	-1.22474	-0.70711	3	1
11	SCORE	ROW11	18	1	-1.22474	-0.70711	-1	1.22474	0.70711	3	2
12	SCORE	ROW12	20	1	-1.22474	-0.70711	-1	1.22474	0.70711	3	2

Missing Values

PROC TRANSREG can estimate missing values, with or without category or monotonicity constraints, so that the regression model fit is optimized. Several approaches to missing data handling are provided. All observations with missing values in IDENTITY, CLASS, POINT, EPOINT, QPOINT, SMOOTH, PBSPLINE, PSPLINE, and BSPLINE variables are excluded from the analysis. When METHOD=UNIVARIATE (specified in the PROC TRANSREG or MODEL statement), observations with missing values in any of the independent variables are excluded from the analysis. When you specify the NOMISS *a-option*, observations with missing values in the other analysis variables are excluded. Otherwise, missing data are estimated, and the variable means are the initial estimates.

You can specify the LINEAR, OPSCORE, MONOTONE, UNTIE, SPLINE, MSPLINE, SSPLINE, LOG, LOGIT, POWER, ARSIN, BOXCOX, RANK, and EXP transformations in any combination with nonmissing values, ordinary missing values, and special missing values, as long as the nonmissing values in each variable have positive variance. No category or order restrictions are placed on the estimates of ordinary missing values. You can force missing value estimates within a variable to be identical by using special missing values (see “DATA Step Processing” in *SAS Language Reference: Concepts*. You can specify up to 27 categories of missing values, in which within-category estimates must be the same, by coding the missing values with ._ and .A through .Z.

You can also specify an ordering of some missing value estimates. You can use the MONOTONE= *a-option* in the PROC TRANSREG or MODEL statement to indicate a range of special missing values (a subset of the list from .A to .Z) with estimates that must be weakly ordered within each variable in which they appear. For example, if MONOTONE=AI, the nine classes, .A, .B, . . . , .I, are monotonically scored and optimally scaled just as MONOTONE transformation values are scored. In this case, category but not order restrictions are placed on the missing values ._ and .J through .Z. You can also use the UNTIE= *a-option* (in the PROC

TRANSREG or MODEL statement) to indicate a range of special missing values with estimates that must be weakly ordered within each variable in which they appear but can be untied.

The missing value estimation facilities enable you to have partitioned or mixed-type variables. For example, a variable can be considered part nominal and part ordinal. Nominal classes of otherwise ordinal variables are coded with special missing values. This feature can be useful with survey research. The class “unfamiliar with the product” in the variable “Rate your preference for ’Brand X’ on a 1 to 9 scale, or if you are unfamiliar with the product, check ’unfamiliar with the product’” is an example. You can code “unfamiliar with the product” as a special missing value, such as .A. The 1s to 9s can be monotonically transformed, while no monotonic restrictions are placed on the quantification of the “unfamiliar with the product” class.

A variable specified for a LINEAR transformation, with special missing values and ordered categorical missing values, can be part interval, part ordinal, and part nominal. A variable specified for a MONOTONE transformation can have two independent ordinal parts. A variable specified for an UNTIE transformation can have an ordered categorical part and an ordered part without category restrictions. Many other mixes are possible.

Missing Values, UNTIE, and Hypothesis Tests

PROC TRANSREG can estimate missing data and monotonically transform variables while untying tied values. Estimates of ordinary missing values (.) are all permitted to be different. Analyses with UNTIE transformations, the UNTIE= *a-option*, and ordinary missing data estimation are all prone to degeneracy problems. Consider the following example. A perfect fit is found by collapsing all observations except the one with two missing values into a single value in y and x1. The following statements produce Figure 120.60:

```
title 'Missing Data';

data x;
   input y x1 x2 @@;
   datalines;
1 3 7    8 3 9    1 8 6    . . 9    3 3 9
8 5 1    6 7 3    2 7 2    1 8 2    . 9 1
;

proc transreg solve;
   model linear(y) = linear(x1 x2);
   output;
run;

proc print;
run;
```

Figure 120.60 Missing Values Example

Missing Data

Obs	_TYPE_	_NAME_	y	Ty	Intercept	x1	x2	TIntercept	Tx1	Tx2
1	SCORE	ROW1	1	2.7680	1	3	7	1	5.1233	7
2	SCORE	ROW2	8	2.7680	1	3	9	1	5.1233	9
3	SCORE	ROW3	1	2.7680	1	8	6	1	5.1233	6
4	SCORE	ROW4	.	12.5878	1	.	9	1	12.7791	9
5	SCORE	ROW5	3	2.7680	1	3	9	1	5.1233	9
6	SCORE	ROW6	8	2.7680	1	5	1	1	5.1233	1
7	SCORE	ROW7	6	2.7680	1	7	3	1	5.1233	3
8	SCORE	ROW8	2	2.7680	1	7	2	1	5.1233	2
9	SCORE	ROW9	1	2.7680	1	8	2	1	5.1233	2
10	SCORE	ROW10	.	2.7680	1	9	1	1	5.1233	1

Generally, the use of ordinary missing data estimation, the UNTIE transformation, and the UNTIE= *a-option* should be avoided, particularly with hypothesis tests. With these options, parameters are estimated based on only a single observation, and they can exert tremendous influence over the results. Each of these parameters has one model degree of freedom associated with it, so small or zero error degrees of freedom can also be a problem.

Controlling the Number of Iterations

Several *a-options* in the PROC TRANSREG or MODEL statement control the number of iterations performed. Iteration terminates when any one of the following conditions is satisfied:

- The number of iterations equals the value of the MAXITER= *a-option*.
- The average absolute change in variable scores from one iteration to the next is less than the value of the CONVERGE= *a-option*.
- The criterion change is less than the value of the CCONVERGE= *a-option*.

You can specify negative values for either convergence *a-option* if you want to define convergence only in terms of the other option. The criterion change can become negative when the data have converged, so it is numerically impossible, within machine precision, to increase the criterion. Usually, a negative criterion change is the result of very small amounts of rounding error, since the algorithms are (usually) convergent. However, there are cases where a negative criterion change is a sign of divergence, which is not necessarily an error. When you specify an SSPLINE transformation or the REITERATE or SOLVE *a-option*, divergence is perfectly normal.

When there are no monotonicity constraints and there is only one canonical variable in each set, PROC TRANSREG (with the SOLVE *a-option*) can usually find the optimal solution in only one iteration. (There are no monotonicity constraints when none of the following is specified: MONOTONE, MSPLINE, or UNTIE transformation or the UNTIE= or MONOTONE= *a-option*. There is only one canonical variable in each set when METHOD=MORALS or METHOD=UNIVARIATE, or when METHOD=REDUNDANCY with only one dependent variable, or when METHOD=CANALS and NCAN=1.)

The initialization iteration is number 0. When there are no monotonicity constraints and there is only one canonical variable in each set, the next iteration shows no change, and iteration stops. At least two iterations (0 and 1) are performed with the SOLVE *a-option* even if nothing changes in iteration 0. The MONOTONE, MSPLINE, and UNTIE variables are not transformed by the canonical initialization. Note that divergence with the SOLVE *a-option*, particularly in the second iteration, is not an error. The initialization iteration is slower and uses more memory than other iterations. However, for many models, specifying the SOLVE *a-option* can greatly decrease the amount of time required to find the optimal transformations.

You can increase the number of iterations to ensure convergence by increasing the value of the MAXITER= *a-option* and decreasing the value of the CONVERGE= *a-option*. Since the average absolute change in standardized variable scores seldom decreases below 1E–11, you should not specify a value for the CONVERGE= *a-option* less than 1E–8 or 1E–10. Most of the data changes occur during the first few iterations, but the data can still change after 50 or even 100 iterations. You can try different combinations of values for the CONVERGE= and MAXITER= *a-options* to ensure convergence without extreme overiteration. If the data do not converge with the default specifications, try CONVERGE=1E–8 and MAXITER=50, or CONVERGE=1E–10 and MAXITER=200. Note that you can specify the REITERATE *a-option* to start iterating where the previous analysis stopped.

Using the REITERATE Algorithm Option

You can use the REITERATE *a-option* to perform additional iterations when PROC TRANSREG stops before the data have adequately converged. For example, suppose that you execute the following step:

```
proc transreg data=a;
   model mspline(y) = mspline(x1-x5);
   output out=b coefficients;
run;
```

If the transformations do not converge in the default 30 iterations, you can perform more iterations without repeating the first 30 iterations, as follows:

```
proc transreg data=b reiterate;
   model mspline(y) = mspline(x1-x5);
   output out=b coefficients;
run;
```

Note that a WHERE statement is not necessary to exclude the coefficient observations. They are automatically excluded because their _TYPE_ value is not SCORE.

You can also use the REITERATE *a-option* to specify starting values other than the original values for the transformations. Providing alternate starting points might help avoid local optima. Here are two examples:

```
proc transreg data=a;
   model rank(y) = rank(x1-x5);
   output out=b;
run;

proc transreg data=b reiterate;
   /* Use ranks as the starting point. */
   model mspline(y) = mspline(x1-x5);
   output out=c coefficients;
```

```
run;

data b;
   set a;
   array tx[6] ty tx1-tx5;
   do j = 1 to 6;
      tx[j] = normal(7);
   end;
run;

proc transreg data=b reiterate;
   /* Use a random starting point. */
   model mspline(y) = mspline(x1-x5);
   output out=c coefficients;
run;
```

Note that divergence with the REITERATE *a-option*, particularly in the second iteration, is not an error since the initial transformation is not required to be a valid member of the transformation family. When you specify the REITERATE *a-option*, the iteration does not terminate when the criterion change is negative during the first 10 iterations.

Avoiding Constant Transformations

There are times when the optimal scaling produces a constant transformed variable. This can happen with the MONOTONE, UNTIE, and MSPLINE transformations when the target is negatively correlated with the original input variable. It can happen with all transformations when the target is uncorrelated with the original input variable. When this happens, the procedure modifies the target to avoid a constant transformation. This strategy avoids certain nonoptimal solutions.

If the transformation is monotonic and a constant transformed variable results, the procedure multiplies the target by –1 and tries the optimal scaling again. If the transformation is not monotonic or if the multiplication by –1 did not help, the procedure tries using a random target. If the transformation is still constant, the previous nonconstant transformation is retained. When a constant transformation is avoided by any strategy, this message is displayed: "A constant transformation was avoided for *name*."

With extreme collinearity, small amounts of rounding error might interact with the instability of the coefficients to produce target vectors that are not positively correlated with the original scaling. If a regression coefficient for a variable is zero, the formula for the target for that variable contains a zero divide. In a multiple regression model, after many iterations, one independent variable can be scaled the same way as the current scaling of the dependent variable, so the other independent variables have coefficients of zero. When the constant transformation warning appears, you should interpret your results with extreme caution, and recheck your model.

Constant Variables

Constant and almost constant variables are zeroed and ignored. When constant variables are expected and should not be zeroed, specify the NOZEROCONSTANT *a-option*.

Character OPSCORE Variables

Character OPSCORE variables are replaced by a numeric variable containing category numbers before the iterations, and the character values are discarded. Only the first eight characters are considered in determining category membership. If you want the original character variable in the output data set, give it a different name in the OPSCORE specification (OPSCORE(x / name=(x2)) and name the original variable in the ID statement (ID x;).

Convergence and Degeneracies

When you specify the SSPLINE transformation, divergence is normal. The rest of this section assumes that you did not specify SSPLINE. For all the methods available in PROC TRANSREG, the algorithms are convergent, in terms of both the criterion being optimized and the parameters being estimated. The value of the criterion being maximized (squared multiple correlation, average squared multiple correlation, or average squared canonical correlation) can, theoretically, never decrease from one iteration to the next. The values of the parameters being solved for (the scores and weights of the transformed variables) become stable after sufficient iteration.

In practice, the criterion being maximized can decrease with overiteration. When the statistic has very nearly reached its maximum, further iterations might report a decrease in the criterion in the last few decimal places. This is a normal result of very small amounts of rounding error. By default, iteration terminates when this occurs because, by default, CCONVERGE=0.0. Specifying CCONVERGE=–1, an impossible change, turns off this check for convergence.

Even though the algorithms are convergent, they might not converge to a global optimum. Also, under extreme circumstances, the solution might degenerate. Because two points always form a straight line, the algorithms sometimes try to reach this degenerate optimum. This sometimes occurs when one observation is an ordinal outlier (when one observation has the extreme rank on all variables). The algorithm can reach an optimal solution that ties all other categories producing two points. Similar results can occur when there are many missing values. More generally, whenever there are very few constraints on the scoring of one or more points, degeneracies can be a problem. In a well-behaved analysis, the maximum data change, average data change, and criterion change all decrease at a rapid rate with each iteration. When the rate of change increases for several iterations, the solution might be degenerating.

Implicit and Explicit Intercepts

Depending on several options, the model intercept is nonzero, zero, or implicit, or there is no intercept. Ordinarily, the model contains an explicit nonzero intercept, and the Intercept variable in the OUT= data set contains ones. When TSTANDARD=CENTER or TSTANDARD=Z is specified, the model contains an explicit, zero intercept and the Intercept variable contains zeros. When METHOD=CANALS, the model is fit with centered variables and the Intercept variable is set to missing.

If you specify CLASS with ZERO=NONE or BSPLINE for one or more independent variables, and TSTANDARD=NOMISS or TSTANDARD=ORIGINAL (the default), an implicit intercept model is fit. The intercept is implicit in a set of the independent variables since there exists a set of independent variables the sum of which is a column of ones. All statistics are mean corrected. The implicit intercept is not an option; it is implied by the model. Specifying SMOOTH or PBSPLINE also implies an implicit intercept model.

With METHOD=CANALS, the Intercept variable contains the *canonical intercept* for canonical coefficients observations: $\hat{\beta}_0 = \bar{\mathbf{y}}'\hat{\boldsymbol{\alpha}} - \bar{\mathbf{x}}'\hat{\boldsymbol{\beta}}$ where $\mathbf{Y}\hat{\boldsymbol{\alpha}} \approx \mathbf{X}\hat{\boldsymbol{\beta}}$.

Passive Observations

Observations can be excluded from the analysis for several reasons; these include zero weight; zero frequency; missing values in variables designated IDENTITY, CLASS, POINT, EPOINT, QPOINT, SMOOTH, PBSPLINE, PSPLINE, or BSPLINE; and missing values with the NOMISS *a-option* specified. These observations are passive in that they do not contribute to determining transformations, R square, sums of squares, degrees of freedom, and so on. However, some information can be computed for them. For example, if no independent variable values are missing, predicted values and redundancy variable values can both be computed. Residuals can be computed for observations with a nonmissing dependent and nonmissing predicted value. Canonical variables for dependent variables can be computed when no dependent variables are missing; canonical variables for independent variables can be computed when no independent variables are missing, and so on. Passive observations in the OUT= data set have a blank value for _TYPE_.

Point Models

The expanded set of independent variables generated from the POINT, EPOINT, and QPOINT expansions can be used to perform ideal point regressions (Carroll 1972) and compute ideal point coordinates for plotting in a biplot (Gabriel 1981). The three types of ideal point coordinates can all be described as transformed coefficients. Assume that m independent variables are specified in one of the three point expansions. Let $\mathbf{b}'$ be a $1 \times m$ row vector of coefficients for these variables and one of the dependent variables. Let $\mathbf{R}$ be a matrix created from the coefficients of the extra variables. When coordinates are requested with the MPC, MEC, or MQC *o-option*, $\mathbf{b}'$ and $\mathbf{R}$ are created from multiple regression coefficients. When coordinates are requested with the CPC, CEC, or CQC *o-option*, $\mathbf{b}'$ and $\mathbf{R}$ are created from canonical coefficients.

If you specify the POINT expansion in the MODEL statement, $\mathbf{R}$ is an $m \times m$ identity matrix times the coefficient for the sums of squares (_ISSQ_) variable. If you specify the EPOINT expansion, $\mathbf{R}$ is an $m \times m$ diagonal matrix of coefficients from the squared variables. If you specify the QPOINT expansion, $\mathbf{R}$ is an $m \times m$ symmetric matrix of coefficients from the squared variables on the diagonal and crossproduct variables off the diagonal. The MPC, MEC, MQC, CPC, CEC, and CQC ideal point coordinates are defined as $-0.5\mathbf{b}'\mathbf{R}^{-1}$. When $\mathbf{R}$ is singular, the ideal point coordinates are infinitely far away and are set to missing, so you should try a simpler version of the model. The version that is simpler than the POINT model is the vector model, where no extra variables are created. In the vector model, designate all independent variables as IDENTITY. Then draw vectors from the origin to the COEFFICIENTS points.

Typically, when you request ideal point coordinates, the MODEL statement should consist of a single transformation for the dependent variables (usually IDENTITY, MONOTONE, or MSPLINE) and a single expansion for the independent variables (one of POINT, EPOINT, or QPOINT).

Redundancy Analysis

Redundancy analysis (Stewart and Love 1968) is a principal component analysis of multivariate regression predicted values. These first steps show the redundancy analysis results produced by PROC TRANSREG. The specification TSTANDARD=Z sets all variables to mean zero and variance one. METHOD=REDUNDANCY specifies redundancy analysis and outputs the redundancy variables to the OUT= data set. The MREDUNDANCY *o-option* outputs two sets of redundancy analysis coefficients to the OUT= data set.

The following statements produce Figure 120.61:

```
title 'Redundancy Analysis';

data x;
   input y1-y3 x1-x4;
   datalines;
 6  8  8 15 18 26 27
 1 12 16 18  9 20  8
 5  6 15 20 17 29 31
 6  9 15 14 10 16 22
 7  5 12 14  6 13  9
 3  6  7  2 14 26 22
 3  5  9 13 18 10 22
 6  3 11  3 15 22 29
 6  3  7 10 20 21 27
 7  5  9  8 10 12 18
;

proc transreg data=x tstandard=z method=redundancy;
   model identity(y1-y3) = identity(x1-x4);
   output out=red mredundancy replace;
run;

proc print data=red(drop=Intercept);
   format _numeric_ 4.1;
run;
```

Figure 120.61 Redundancy Analysis Example

Redundancy Analysis

Obs	_TYPE_	_NAME_	y1	y2	y3	x1	x2	x3	x4	Red1	Red2	Red3
1	SCORE	ROW1	0.5	0.6	-0.8	0.6	0.9	1.0	0.7	0.2	-0.5	-0.9
2	SCORE	ROW2	-2.0	2.1	1.5	1.1	-1.0	0.1	-1.7	1.6	-1.5	0.4
3	SCORE	ROW3	0.0	-0.1	1.2	1.4	0.7	1.5	1.2	1.0	0.8	-1.3
4	SCORE	ROW4	0.5	1.0	1.2	0.4	-0.8	-0.5	0.1	0.5	1.7	0.1
5	SCORE	ROW5	1.0	-0.4	0.3	0.4	-1.6	-1.0	-1.6	1.0	0.1	0.9
6	SCORE	ROW6	-1.0	-0.1	-1.1	-1.6	0.1	1.0	0.1	-0.8	-0.9	1.4
7	SCORE	ROW7	-1.0	-0.4	-0.6	0.2	0.9	-1.5	0.1	-1.0	-0.4	-1.3
8	SCORE	ROW8	0.5	-1.2	0.0	-1.5	0.3	0.4	1.0	-1.2	0.8	0.7
9	SCORE	ROW9	0.5	-1.2	-1.1	-0.3	1.3	0.2	0.7	-1.0	-0.9	-0.8
10	SCORE	ROW10	1.0	-0.4	-0.6	-0.6	-0.8	-1.1	-0.4	-0.4	0.8	0.7
11	M REDUND	Red1	.	.	.	0.7	-0.6	0.4	-0.1	.	.	.
12	M REDUND	Red2	.	.	.	0.3	-1.5	-0.6	1.9	.	.	.
13	M REDUND	Red3	.	.	.	-0.7	-0.7	0.3	-0.3	.	.	.
14	R REDUND	x1	.	.	.	.	.	.	.	0.8	-0.0	-0.6
15	R REDUND	x2	.	.	.	.	.	.	.	-0.6	-0.2	-0.7
16	R REDUND	x3	.	.	.	.	.	.	.	0.1	-0.2	-0.1
17	R REDUND	x4	.	.	.	.	.	.	.	-0.5	0.3	-0.5

The _TYPE_='SCORE' observations of the Red1–Red3 variables contain the redundancy variables. The nonmissing "M REDUND" values are coefficients for predicting the redundancy variables from the independent variables. The nonmissing "R REDUND" values are coefficients for predicting the independent variables from the redundancy variables.

The next steps show how to generate the same results manually. The data set is standardized, predicted values are computed, and principal components of the predicted values are computed. The following statements produce the redundancy variables, shown in Figure 120.62:

```
proc standard data=x out=std m=0 s=1;
   title2 'Manually Generate Redundancy Variables';
run;

proc reg noprint data=std;
   model y1-y3 = x1-x4;
   output out=p p=ay1-ay3;
run; quit;

proc princomp data=p cov noprint std out=p;
   var ay1-ay3;
run;

proc print data=p(keep=Prin:);
   format _numeric_ 4.1;
run;
```

Figure 120.62 Redundancy Analysis Example

Redundancy Analysis
Manually Generate Redundancy Variables

Obs	Prin1	Prin2	Prin3
1	0.2	-0.5	-0.9
2	1.6	-1.5	0.4
3	1.0	0.8	-1.3
4	0.5	1.7	0.1
5	1.0	0.1	0.9
6	-0.8	-0.9	1.4
7	-1.0	-0.4	-1.3
8	-1.2	0.8	0.7
9	-1.0	-0.9	-0.8
10	-0.4	0.8	0.7

The following statements produce the coefficients for predicting the redundancy variables from the independent variables, shown in Figure 120.63:

```
proc reg data=p outest=redcoef noprint;
   title2 'Manually Create Redundancy Coefficients';
   model Prin1-Prin3 = x1-x4;
run; quit;

proc print data=redcoef(keep=x1-x4);
   format _numeric_ 4.1;
run;
```

Figure 120.63 Redundancy Analysis Example

Redundancy Analysis
Manually Create Redundancy Coefficients

Obs	x1	x2	x3	x4
1	0.7	-0.6	0.4	-0.1
2	0.3	-1.5	-0.6	1.9
3	-0.7	-0.7	0.3	-0.3

The following statements produce the coefficients for predicting the independent variables from the redundancy variables, shown in Figure 120.64:

```
proc reg data=p outest=redcoef2 noprint;
   title2 'Manually Create Other Coefficients';
   model x1-x4 = prin1-prin3;
run; quit;

proc print data=redcoef2(keep=Prin1-Prin3);
   format _numeric_ 4.1;
run;
```

Figure 120.64 Redundancy Analysis Example

Redundancy Analysis
Manually Create Other Coefficients

Obs	Prin1	Prin2	Prin3
1	0.8	-0.0	-0.6
2	-0.6	-0.2	-0.7
3	0.1	-0.2	-0.1
4	-0.5	0.3	-0.5

Optimal Scaling

An alternating least squares optimal scaling algorithm can be divided into two major stages. The first major stage estimates the parameters of the linear model. These parameters are used to create the predicted values or target for each variable that can be transformed. Each target minimizes squared error (as explained in the discussion of the algorithms in *SAS Technical Report R-108*). The definition of the target depends on many factors, such as whether a variable is independent or dependent, which algorithm is used (for example, regression, redundancy, CANALS, or principal components), and so on. The definition of the target is independent of the transformation family you specify for the variable. However, the target values for a variable typically do not fit the prescribed transformation family for the variable. They might not have the right category structure; they might not have the right order; they might not be a linear combination of the columns of a B-spline basis; and so on.

The second major stage is optimal scaling. Optimal scaling can be defined as a possibly constrained, least squares regression problem. When you specify an optimal transformation, or when missing data are estimated for any variable, the full representation of the variable is not simply a vector; it is a matrix with more than one column. The optimal scaling phase finds the vector that is a linear combination of the columns of this matrix that is closest to the target (in terms of minimum squared error), among those that do not violate any of the constraints imposed by the transformation family. Optimal scaling methods are independent of the data analysis method that generated the target. In all cases, optimal scaling can be accomplished by creating a design matrix based on the original scaling of the variable and the transformation family specified for that variable. The optimally scaled variable is a linear combination of the columns of the design matrix. The coefficients of the linear combination are found by using (possibly constrained) least squares. Many optimal scaling problems are solved without actually constructing design and projection matrices. The next two sections describe the algorithms used by PROC TRANSREG for optimal scaling. The first section discusses optimal scaling for OPSCORE, MONOTONE, UNTIE, and LINEAR transformations, including how missing values are handled. The second section addresses SPLINE and MSPLINE transformations.

OPSCORE, MONOTONE, UNTIE, and LINEAR Transformations

Two vectors of information are needed to produce the optimally scaled variable: the initial variable scaling vector $\mathbf{x}$ and the target vector $\mathbf{y}$. For convenience, both vectors are first sorted on the values of the initial scaling vector. If you request an UNTIE transformation, the target vector is sorted within ties in the initial scaling vector. The normal SAS collating sequence for missing and nonmissing values is used. Sorting simply permits the constraints to be specified in terms of relationships among adjoining coefficients. The sorting process partitions $\mathbf{x}$ and $\mathbf{y}$ into missing and nonmissing parts $(\mathbf{x}_m'\mathbf{x}_n')'$, and $(\mathbf{y}_m'\mathbf{y}_n')'$.

Next, PROC TRANSREG determines category membership. Every ordinary missing value (.) forms a separate category. (Three ordinary missing values form three categories.) Every special missing value within the range specified in the UNTIE= *a-option* forms a separate category. (If UNTIE= BC and there are three .B and two .C missing values, five categories are formed from them.) For all other special missing values, a separate category is formed for each different value. (If there are four .A missing values, one category is formed from them.)

Each distinct nonmissing value forms a separate category for OPSCORE and MONOTONE transformations (1 1 1 2 2 3 form three categories). Each nonmissing value forms a separate category for all other transformations (1 1 1 2 2 3 form six categories). When category membership is determined, category means are computed. Here is an example:

x:	(.	.	.A	.A	.B	1	1	1	2	2	3	3	3	4)′
y:	(5	6	2	4	2	1	2	3	4	6	4	5	6	7)′
OPSCORE and MONOTONE means:	(5	6	3		2	2			5		5			7)′
other means:	(5	6	3		2	1	2	3	4	6	4	5	6	7)′

The category means are the coefficients of a category indicator design matrix. The category means are the Fisher (1938) optimal scores. For MONOTONE and UNTIE transformations, order constraints are imposed on the category means for the nonmissing partition by merging categories that are out of order. The algorithm checks upward until an order violation is found, and then averages downward until the order violation is averaged away. (The average of $\bar{x}_1$ computed from n_1 observations and $\bar{x}_2$ computed from n_2 observations is $(n_1\bar{x}_1 + n_2\bar{x}_2)/(n_1 + n_2)$.) The MONOTONE algorithm (Kruskal 1964, secondary approach to ties) for this example with means for the nonmissing values $(2\ 5\ 5\ 7)'$ would do the following checks: $2 < 5$: OK, $5 = 5$: OK, $5 < 7$: OK. The means are in the proper order, so no work is needed.

The UNTIE transformation (Kruskal 1964, primary approach to ties) uses the same algorithm on the means of the nonmissing values $(1\ 2\ 3\ 4\ 6\ 4\ 5\ 6\ 7)'$ but with different results for this example: $1 < 2$: OK, $2 < 3$: OK, $3 < 4$: OK, $4 < 6$: OK, $6 > 4$: average 6 and 4 and replace 6 and 4 by the average. The new means of the nonmissing values are $(1\ 2\ 3\ 4\ 5\ 5\ 5\ 6\ 7)'$. The check resumes: $4 < 5$: OK, $5 = 5$: OK, $5 = 5$: OK, $5 < 6$: OK, $6 < 7$: OK. If some of the special missing values are ordered, the upward-checking, downward-averaging algorithm is applied to them also, independently of the other missing and nonmissing partitions. When the means conform to any required category or order constraints, an optimally scaled vector is produced from the means. The following example results from a MONOTONE transformation:

x:	(.	.	.A	.A	.B	1	1	1	2	2	3	3	3	4)′
y:	(5	6	2	4	2	1	2	3	4	6	4	5	6	7)′
result:	(5	6	3	3	2	2	2	2	5	5	5	5	5	7)′

The upward-checking, downward-averaging algorithm is equivalent to creating a category indicator design matrix, solving for least squares coefficients with order constraints, and then computing the linear combination of design matrix columns.

For the optimal transformation LINEAR and for nonoptimal transformations, missing values are handled as just described. The nonmissing target values are regressed onto the matrix defined by the nonmissing initial scaling values and an intercept. In this example, the target vector $y_n = (1\ 2\ 3\ 4\ 6\ 4\ 5\ 6\ 7)'$ is regressed onto the design matrix

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 2 & 2 & 3 & 3 & 3 & 4 \end{bmatrix}'$$

Although only a linear transformation is performed, the effect of a linear regression optimal scaling is not eliminated by the later standardization step (unless the variable has no missing values). In the presence of missing values, the linear regression is necessary to minimize squared error.

SPLINE and MSPLINE Transformations

The missing portions of variables subjected to SPLINE or MSPLINE transformations are handled the same way as for OPSCORE, MONOTONE, UNTIE, and LINEAR transformations (see the previous section). The nonmissing partition is handled by first creating a B-spline basis of the specified degree with the specified knots for the nonmissing partition of the initial scaling vector and then regressing the target onto the basis. The optimally scaled vector is a linear combination of the B-spline basis vectors. Ordinary least squares regression coefficients are used. An algorithm for generating the B-spline basis is given in De Boor (1978, pp. 134–135). B-splines are both a computationally accurate and efficient way of constructing a basis for piecewise polynomials; however, they are not the most natural method of describing splines.

Consider an initial scaling vector $x = (1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9)'$ and a degree-three spline with interior knots at 3.5 and 6.5. The B-spline basis for the transformation is the left matrix, and the natural piecewise polynomial spline basis is the right matrix.

$$\underset{\text{B-Spline Basis}}{\begin{bmatrix} 1.000 & 0.000 & 0.000 & 0.000 & 0 & 0 \\ 0.216 & 0.608 & 0.167 & 0.009 & 0 & 0 \\ 0.008 & 0.458 & 0.461 & 0.073 & 0 & 0 \\ 0 & 0.172 & 0.585 & 0.241 & 0.001 & 0 \\ 0 & 0.037 & 0.463 & 0.463 & 0.037 & 0 \\ 0 & 0.001 & 0.241 & 0.585 & 0.172 & 0 \\ 0 & 0 & 0.073 & 0.461 & 0.458 & 0.008 \\ 0 & 0 & 0.009 & 0.167 & 0.608 & 0.216 \\ 0 & 0 & 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}} \quad \underset{\text{Piecewise Polynomial Splines}}{\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 2 & 4 & 8 & 0 & 0 \\ 1 & 3 & 9 & 27 & 0 & 0 \\ 1 & 4 & 16 & 64 & 0.125 & 0 \\ 1 & 5 & 25 & 125 & 3.375 & 0 \\ 1 & 6 & 36 & 216 & 15.625 & 0 \\ 1 & 7 & 49 & 343 & 42.875 & 0.125 \\ 1 & 8 & 64 & 512 & 91.125 & 3.375 \\ 1 & 9 & 81 & 729 & 166.375 & 15.625 \end{bmatrix}}$$

The two matrices span the same column space. The natural basis has an intercept, a linear term, a quadratic term, a cubic term, and two more terms since there are two interior knots. These terms are generated (for knot k and $\mathbf{x}$ element x) by the formula $(x-k)^3 \times I_{(x>k)}$. The indicator variable $I_{(x>k)}$ evaluates to 1.0 if x is greater than k and to 0.0 otherwise. If knot k had been repeated, there would be a $(x-k)^2 \times I_{(x>k)}$ term also. Notice that the fifth column makes no contribution to the curve before 3.5, makes zero contribution at 3.5 (the transformation is continuous), and makes an increasing contribution beyond 3.5. The same pattern of results holds for the last term with knot 6.5. The coefficient of the fifth column represents the change in the cubic portion of the curve after 3.5. The coefficient of the sixth column represents the change in the cubic portion of the curve after 6.5.

The numbers in the B-spline basis do not have a simple interpretation like the numbers in the natural piecewise polynomial basis. The B-spline basis has a diagonally banded structure. The band shifts one column to the right after every knot. The number of entries in each row that can potentially be nonzero is one greater than the degree. The elements within a row always sum to one. The B-spline basis is accurate because of the smallness of the numbers and the lack of extreme collinearity inherent in the natural polynomials. B-splines are efficient because PROC TRANSREG can take advantage of the sparseness of the B-spline basis when it accumulates crossproducts. The number of required multiplications and additions to accumulate the crossproduct matrix does not increase with the number of knots but does increase with the degree of the spline, so it is much more computationally efficient to increase the number of knots than to increase the degree of the polynomial.

MSPLINE transformations are handled like SPLINE transformations except that constraints are placed on the coefficients to ensure monotonicity. When the coefficients of the B-spline basis are monotonically increasing, the transformation is monotonically increasing. When the polynomial degree is two or less, monotone coefficient splines, integrated splines (Winsberg and Ramsay 1980), and the general class of all monotone splines are equivalent.

Specifying the Number of Knots

Keep the number of knots small (usually less than 10, although you can specify more). A degree-three spline with nine knots, one at each decile, can closely follow a large variety of curves. Each spline transformation of degree p with q knots fits a model with $p + q$ parameters. The total number of parameters should be much less than the number of observations. Usually in regression analyses, it is recommended that there be at least five or ten observations for each parameter in order to get stable results. For example, when spline transformations of degree three with nine knots are requested for six variables, the number of observations in the data set should be at least 5 or 10 times 72 (since $6 \times (3 + 9)$ is the total number of parameters). The overall model can also have a parameter for the intercept and one or more parameters for each nonspline variable in the model.

Increasing the number of knots gives the spline more freedom to bend and follow the data. Increasing the degree also gives the spline more freedom, but to a lesser extent. Specifying a large number of knots is much better than increasing the degree beyond three.

When you specify NKNOTS=q for a variable with n observations, then each of the $q + 1$ segments of the spline contains $n/(q + 1)$ observations on the average. When you specify KNOTS=number-list, make sure that there is a reasonable number of observations in each interval.

The following statements find a cubic polynomial transformation of x and no transformation of y:

```
proc transreg;
   model identity(y)=spline(x);
   output;
run;
```

The following statements find a cubic-spline transformation for x that consists of the weighted sum of a single constant, a single straight line, a quadratic curve for the portion of the variable less than 3.0, a different quadratic curve for the portion greater than 3.0 (since the 3.0 knot is repeated), and a different cubic curve for each of the intervals: (minimum to 1.5), (1.5 to 2.4), (2.4 to 3.0), (3.0 to 4.0), and (4.0 to maximum):

```
proc transreg;
   model identity(y)=spline(x / knots=1.5 2.4 3.0 3.0 4.0);
   output;
run;
```

The transformation is continuous everywhere, its first derivative is continuous everywhere, its second derivative is continuous everywhere except at 3.0, and its third derivative is continuous everywhere except at 1.5, 2.4, 3.0, and 4.0.

The following statements find a quadratic spline transformation that consists of a polynomial $\mathsf{x\_t} = b_0 + b_1\mathsf{x} + b_2\mathsf{x}^2$ for the range ($\mathsf{x} < 3.0$) and a completely different polynomial $\mathsf{x\_t} = b_3 + b_4\mathsf{x} + b_5\mathsf{x}^2$ for the range ($\mathsf{x} > 3.0$):

```
proc transreg;
   model identity(y)=spline(x / knots=3 3 3 degree=2);
   output;
run;
```

The two curves are not required to be continuous at 3.0.

The following statements categorize y into 10 intervals and find a step-function transformation:

```
proc transreg;
   model identity(y)=spline(x / degree=0 nknots=9);
   output;
run;
```

One aspect of this transformation family is unlike all other optimal transformation families. The initial scaling of the data does not fit the restrictions imposed by the transformation family. This is because the initial variable can be continuous, but a discrete step-function transformation is sought. Zero-degree spline variables are categorized before the first iteration.

The following statements find a continuous, piecewise linear transformation of x:

```
proc transreg;
   model identity(y)=spline(x / degree=1 nknots=8);
   output;
run;
```

SPLINE, BSPLINE, and PSPLINE Comparisons

SPLINE is a transformation. It takes a variable as input and produces a transformed variable as output. Internally, with SPLINE, a B-spline basis is used to find the transformation, which is a linear combination of the columns of the B-spline basis. However, with SPLINE, the basis is not made available in any output.

BSPLINE is an expansion. It takes a variable as input and produces more than one variable as output. The output variables are the same B-spline basis that is used internally by SPLINE.

PSPLINE is an expansion. It takes a variable as input and produces more than one variable as output. The difference between PSPLINE and BSPLINE is that PSPLINE produces a piecewise polynomial, whereas BSPLINE produces a B-spline. A matrix consisting of a piecewise polynomial basis and an intercept spans the same space as the B-spline matrix, but the basis vectors are quite different. The numbers in the piecewise polynomials can get quite large; the numbers in the B-spline basis range between 0 and 1. There are many more zeros in the B-spline basis.

Interchanging SPLINE, BSPLINE, and PSPLINE should have no effect on the fit of the overall model except for the fact that PSPLINE is much more prone to numerical problems. Similarly, interchanging a CLASS expansion and an OPSCORE transformation should have no effect on the fit of the overall model.

Hypothesis Tests

PROC TRANSREG has a set of options for testing hypotheses in models with a single dependent variable. The TEST *a-option* produces an ANOVA table. It tests the null hypothesis that the vector of coefficients for all of the transformations is zero. The SS2 *a-option* produces a regression table with Type II tests of the contribution of each transformation to the overall model. In some cases, exact tests are provided; in other cases, the tests are approximate, liberal, or conservative.

There are two reasons why it is typically not appropriate to test hypotheses by using the output from PROC TRANSREG as input to other procedures such as the REG procedure. First, PROC REG has no way of determining how many degrees of freedom were used for each transformation. Second, the Type II sums of squares for the tests of the individual regression coefficients are not correct for the transformation regression model because PROC REG, as it evaluates the effect of each variable, cannot change the transformations of the other variables. PROC TRANSREG uses the correct degrees of freedom and sums of squares.

In an ordinary univariate linear model, there is one parameter for each independent variable, including the intercept. In the transformation regression model, many of the "variables" are used internally in the bases for the transformations. Each basis column has one parameter or *scoring* coefficient, and each linearly independent column has one model degree of freedom associated with it. Coefficients applied to transformed variables, *model coefficients*, do not enter into the degrees-of-freedom calculations. They are byproducts of the standardizations and can be absorbed into the transformations by specifying the ADDITIVE *a-option*. The word *parameter* is reserved for model and scoring coefficients that have a degree of freedom associated with them.

For expansions, there is one model parameter for each variable created by the expansion (except for all missing CLASS columns and expansions that have an implicit intercept). Each IDENTITY variable has one model parameter. If there are m POINT variables, they expand to $m + 1$ variables and hence have $m + 1$ model parameters. For m EPOINT variables, there are $2m$ model parameters. For m QPOINT variables, there are $m(m + 3)/2$ model parameters. If a variable with m categories is designated CLASS, there are $m - 1$ parameters. For BSPLINE and PSPLINE variables of DEGREE=n with NKNOTS=k, there are $n + k$ parameters. Note that one of the $n + k + 1$ BSPLINE columns and one of the m CLASS(variable / ZERO=NONE) columns are not counted due to the implicit intercept.

There are scoring parameters for missing values in nonexcluded observations. Each ordinary missing value (.) has one scoring parameter. Each different special missing value (._ and .A through .Z) within each variable has one scoring parameter. Missing values specified in the UNTIE= and MONOTONE= options follow the rules for UNTIE and MONOTONE transformations, which are described later in this chapter.

For all nonoptimal transformations (LOG, LOGIT, ARSIN, POWER, EXP, RANK, BOXCOX), there is one parameter per variable in addition to any missing value scoring parameters.

For SPLINE, OPSCORE, and LINEAR transformations, the number of scoring parameters is the number of basis columns that are used internally to find the transformations minus 1 for the intercept. The number of scoring parameters for SPLINE variables is the same as the number of model parameters for BSPLINE and PSPLINE variables. If DEGREE=*n* and NKNOTS=*k*, there are $n + k$ scoring parameters. The number of scoring parameters for OPSCORE, SMOOTH, and SSPLINE variables is the same as the number of model parameters for CLASS variables. If there are *m* categories, there are *m* – 1 scoring parameters. There is one parameter for each LINEAR variable. For SPLINE, OPSCORE, LINEAR, MONOTONE, UNTIE, and MSPLINE transformations, missing value scoring parameters are computed as described previously with the nonoptimal transformations.

The number of scoring parameters for MONOTONE, UNTIE, and MSPLINE transformations is less precise than for SPLINE, OPSCORE, and LINEAR transformations. One way of handling a MONOTONE transformation is to treat it as if it were the same as an OPSCORE transformation. If there are *m* categories, there are *m* – 1 potential scoring parameters. However, there are typically fewer than *m* – 1 unique parameter estimates, since some of those *m* – 1 scoring parameter estimates might be tied during the optimal scaling to impose the order constraints. Imposing ties on the scoring parameter estimates is equivalent to fitting a model with fewer parameters. So there are two available scoring parameter counts: *m* – 1 and a smaller number that is determined during the analysis. Using *m* – 1 as the model degrees of freedom for MONOTONE variables (treating OPSCORE and MONOTONE transformations the same way) is *conservative*, since the MONOTONE scoring parameter estimates are more restricted than the OPSCORE scoring parameter estimates. Using the smaller count (the number of scoring parameter estimates that are different, minus 1 for the intercept) in the model degrees of freedom is *liberal*, since the data and the model together are being used to determine the number of parameters. PROC TRANSREG reports tests that use both liberal and conservative degrees of freedom to provide lower and upper bounds on the "true" *p*-values.

For the UNTIE transformation, the conservative scoring parameter count is the number of distinct observations, whereas the liberal scoring parameter count is the number of scoring parameter estimates that are different, minus 1 for the intercept. Hence, when you specify UNTIE, conservative tests have zero error degrees of freedom unless there are replicated observations.

For MSPLINE variables of DEGREE=*n* and NKNOTS=*k*, the conservative scoring parameter count is $n + k$, whereas the liberal parameter count is the number of scoring parameter estimates that are different, minus 1 for the intercept. A liberal degrees of freedom of 1 does not necessarily imply a linear transformation. It implies only that *n* plus *k* minus the number of ties imposed equals 1. An example of a one-degree-of-freedom nonlinear transformation is a two-piece linear transformation in which the slope of one piece is 0.

The number of scoring parameters is determined during each iteration. After the last iteration, enough information is available for the TEST *a-option* to produce an ANOVA table that reports the overall fit of the model. If you specify the SS2 *a-option*, further iterations are necessary to test the contribution of each transformation to the overall model.

The liberal tests do not compensate for overparameterization. For example, requesting a spline transformation with *k* knots when a linear transformation will suffice results in "liberal" tests that are actually conservative because too many degrees of freedom are being used for the transformations. To avoid this problem, use as few knots as possible.

In ordinary multiple regression, an *F* test of the null hypothesis that the coefficient for variable x_j is zero can be constructed by comparing two linear models. One model is the full model with all parameters, and the other is a reduced model that has all parameters except the parameter for variable x_j. The difference between the model sum of squares for the full model and the model sum of squares for the reduced model is the Type II sum of squares for the test of the null hypothesis that the coefficient for variable x_j is 0. The numerator of the *F* test has one degree of freedom. The mean square error for the full model is the denominator of the *F* test of variable x_j. Note that the estimates of the coefficients for the two models are not usually the same. When variable x_j is removed, the coefficients for the other variables change to compensate for the removal of x_j. In a transformation regression model, the transformations of the other variables must be permitted to change and the numerator degrees of freedom are not always ones. It is not correct to simply let the model coefficients for the transformed variables change and apply the new model coefficients to the old transformations computed with the old scoring parameter estimates. In a transformation regression model, further iteration is needed to test each transformation, because all the scoring parameter estimates for other variables must be permitted to change to test the effect of variable x_j. This can be quite time-consuming for a large model if the SOLVE *a-option* cannot be used to solve directly for the transformations.

Output Data Set

The OUT= output data set can contain a great deal of information; however, in most cases, the output data set contains a small portion of the entire range of available information.

Output Data Set Examples

This section provides three brief examples, illustrating some typical OUT= output data sets. See the section "Output Data Set Contents" on page 10009 for a complete list of the contents of the OUT= data set.

The first example shows the output data set from a two-way ANOVA model. The following statements produce Figure 120.65:

```
title 'ANOVA Output Data Set Example';

data ReferenceCell;
   input y x1 $ x2 $;
   datalines;
11  a  a
12  a  a
10  a  a
 4  a  b
 5  a  b
 3  a  b
 5  b  a
 6  b  a
 4  b  a
 2  b  b
 3  b  b
 1  b  b
;

* Fit Reference Cell Two-Way ANOVA Model;
proc transreg data=ReferenceCell;
   model identity(y) = class(x1 | x2);
   output coefficients replace predicted residuals;
run;

* Print the Results;
proc print;
run;
```

```
proc contents position;
   ods select position;
run;
```

Figure 120.65 ANOVA Example Output Data Set Contents

ANOVA Output Data Set Example

Obs	_TYPE_	_NAME_	y	Py	Ry	Intercept	x1a	x2a	x1ax2a	x1	x2
1	SCORE	ROW1	11	11	0	1	1.0	1	1	a	a
2	SCORE	ROW2	12	11	1	1	1.0	1	1	a	a
3	SCORE	ROW3	10	11	-1	1	1.0	1	1	a	a
4	SCORE	ROW4	4	4	0	1	1.0	0	0	a	b
5	SCORE	ROW5	5	4	1	1	1.0	0	0	a	b
6	SCORE	ROW6	3	4	-1	1	1.0	0	0	a	b
7	SCORE	ROW7	5	5	0	1	0.0	1	0	b	a
8	SCORE	ROW8	6	5	1	1	0.0	1	0	b	a
9	SCORE	ROW9	4	5	-1	1	0.0	1	0	b	a
10	SCORE	ROW10	2	2	0	1	0.0	0	0	b	b
11	SCORE	ROW11	3	2	1	1	0.0	0	0	b	b
12	SCORE	ROW12	1	2	-1	1	0.0	0	0	b	b
13	M COEFFI	y	.	.	.	2	2.0	3	4		
14	MEAN	y	.	.	.	.	7.5	8	11		

ANOVA Output Data Set Example

The CONTENTS Procedure

Variables in Creation Order				
#	**Variable**	**Type**	**Len**	**Label**
1	_TYPE_	Char	8	
2	_NAME_	Char	32	
3	y	Num	8	
4	Py	Num	8	y Predicted Values
5	Ry	Num	8	y Residuals
6	Intercept	Num	8	Intercept
7	x1a	Num	8	x1 a
8	x2a	Num	8	x2 a
9	x1ax2a	Num	8	x1 a * x2 a
10	x1	Char	32	
11	x2	Char	32	

The _TYPE_ variable indicates observation type: score, multiple regression coefficient (parameter estimates), and marginal means. The _NAME_ variable contains the default observation labels, “ROW1”, “ROW2”, and so on, and contains the dependent variable name (y) for the remaining observations. If you specify an ID statement, _NAME_ contains the values of the first ID variable for score observations. The y variable is the dependent variable, Py contains the predicted values, Ry contains the residuals, and the variables Intercept through x1ax2a contain the design matrix. The x1 and x2 variables are the original CLASS variables.

The next example shows the contents of the output data set from fitting a curve through a scatter plot. The following statements produce Figure 120.66:

```
title 'Output Data Set for Curve Fitting Example';

data a;
   do x = 1 to 100;
      y = log(x) + sin(x / 10) + normal(7);
      output;
   end;
run;

proc transreg;
   model identity(y) = spline(x / nknots=9);
   output predicted out=b;
run;

proc contents position;
   ods select position;
run;
```

Figure 120.66 Predicted Values Example Output Data Set Contents

Output Data Set for Curve Fitting Example

The CONTENTS Procedure

Variables in Creation Order				
#	**Variable**	**Type**	**Len**	**Label**
1	_TYPE_	Char	8	
2	_NAME_	Char	32	
3	y	Num	8	
4	Ty	Num	8	y Transformation
5	Py	Num	8	y Predicted Values
6	Intercept	Num	8	Intercept
7	x	Num	8	
8	TIntercept	Num	8	Intercept Transformation
9	Tx	Num	8	x Transformation

The OUT= data set contains _TYPE_ and _NAME_ variables. Since no coefficients or coordinates are requested, all observations are _TYPE_='SCORE'. The y variable is the original dependent variable, Ty is the transformed dependent variable, Py contains the predicted values, x is the original independent variable, and Tx is the transformed independent variable. The data set also contains an Intercept and transformed intercept TIntercept variable. (In this case, the transformed intercept is the same as the intercept. However, if you specify the TSTANDARD= and ADDITIVE options, these are not always the same.)

The following example shows the results from specifying METHOD=MORALS when there is more than one dependent variable:

```
title 'METHOD=MORALS Output Data Set Example';

data x;
   input y1 y2 x1 $ x2 $;
   datalines;
11 1 a a
10 4 b a
 5 2 a b
 5 9 b b
 4 3 c c
 3 6 b a
 1 8 a b
;

* Fit Reference Cell Two-Way ANOVA Model;
proc transreg data=x noprint solve;
   model spline(y1 y2) = opscore(x1 x2 / name=(n1 n2));
   output coefficients predicted residuals;
   id x1 x2;
run;

* Print the Results;
proc print;
run;

proc contents position;
   ods select position;
run;
```

These statements produce Figure 120.67.

Figure 120.67 METHOD=MORALS Rolled Output Data Set

METHOD=MORALS Output Data Set Example

Obs	_DEPVAR_	_TYPE_	_NAME_	_DEPEND_	T_DEPEND_	P_DEPEND_
1	Spline(y1)	SCORE	a	11	13.1600	11.1554
2	Spline(y1)	SCORE	b	10	6.1931	6.8835
3	Spline(y1)	SCORE	a	5	2.4467	4.7140
4	Spline(y1)	SCORE	b	5	2.4467	0.4421
5	Spline(y1)	SCORE	c	4	4.2076	4.2076
6	Spline(y1)	SCORE	b	3	5.5693	6.8835
7	Spline(y1)	SCORE	a	1	4.9766	4.7140
8	Spline(y1)	M COEFFI	y1	.	.	.
9	Spline(y2)	SCORE	a	1	-0.5303	-0.5199
10	Spline(y2)	SCORE	b	4	5.5487	4.5689
11	Spline(y2)	SCORE	a	2	3.8940	4.5575
12	Spline(y2)	SCORE	b	9	9.6358	9.6462
13	Spline(y2)	SCORE	c	3	5.6210	5.6210
14	Spline(y2)	SCORE	b	6	3.5994	4.5689
15	Spline(y2)	SCORE	a	8	5.2314	4.5575
16	Spline(y2)	M COEFFI	y2	.	.	.

Obs	R_DEPEND_	Intercept	n1	n2	TIntercept	Tn1	Tn2	x1	x2
1	2.00464	1	0	0	1.0000	0.06711	-0.09384	a	a
2	-0.69041	1	1	0	1.0000	1.51978	-0.09384	b	a
3	-2.26724	1	0	1	1.0000	0.06711	1.32038	a	b
4	2.00464	1	1	1	1.0000	1.51978	1.32038	b	b
5	0.00000	1	2	2	1.0000	0.23932	1.32038	c	c
6	-1.31422	1	1	0	1.0000	1.51978	-0.09384	b	a
7	0.26261	1	0	1	1.0000	0.06711	1.32038	a	b
8	.	.	.	.	10.9253	-2.94071	-4.55475	y1	y1
9	-0.01043	1	0	0	1.0000	0.03739	-0.09384	a	a
10	0.97988	1	1	0	1.0000	1.51395	-0.09384	b	a
11	-0.66347	1	0	1	1.0000	0.03739	1.32038	a	b
12	-0.01043	1	1	1	1.0000	1.51395	1.32038	b	b
13	0.00000	1	2	2	1.0000	0.34598	1.32038	c	c
14	-0.96945	1	1	0	1.0000	1.51395	-0.09384	b	a
15	0.67390	1	0	1	1.0000	0.03739	1.32038	a	b
16	.	.	.	.	-0.3119	3.44636	3.59024	y2	y2

Figure 120.67 *continued*

METHOD=MORALS Output Data Set Example

The CONTENTS Procedure

Variables in Creation Order				
#	**Variable**	**Type**	**Len**	**Label**
1	_DEPVAR_	Char	42	Dependent Variable Transformation(Name)
2	_TYPE_	Char	8	
3	_NAME_	Char	32	
4	_DEPEND_	Num	8	Dependent Variable
5	T_DEPEND_	Num	8	Dependent Variable Transformation
6	P_DEPEND_	Num	8	Dependent Variable Predicted Values
7	R_DEPEND_	Num	8	Dependent Variable Residuals
8	Intercept	Num	8	Intercept
9	n1	Num	8	
10	n2	Num	8	
11	TIntercept	Num	8	Intercept Transformation
12	Tn1	Num	8	n1 Transformation
13	Tn2	Num	8	n2 Transformation
14	x1	Char	32	
15	x2	Char	32	

If you specify METHOD=MORALS with multiple dependent variables, PROC TRANSREG performs separate univariate analyses and stacks the results in the OUT= data set. For this example, the results of the first analysis are in the partition designated by _DEPVAR_='Spline(y1)' and the results of the second analysis are in the partition designated by _DEPVAR_='Spline(y2)', which are the transformation and dependent variable names. Each partition has _TYPE_='SCORE' observations for the variables and a _TYPE_='M COEFFI' observation for the coefficients. In this example, an ID variable is specified, so the _NAME_ variable contains the formatted values of the first ID variable. Since both dependent variables have to go into the same column, the dependent variable is given a new name, _DEPEND_. The dependent variable transformation is named T_DEPEND_, the predicted values variable is named P_DEPEND_, and the residuals variable is named R_DEPEND_.

The independent variables are character OPSCORE variables. By default, PROC TRANSREG replaces character OPSCORE variables with category numbers and discards the original character variables. To avoid this, the input variables are renamed from x1 and x2 to n1 and n2 and the original x1 and x2 are added to the data set as ID variables. The n1 and n2 variables contain the initial values for the OPSCORE transformations, and the Tn1 and Tn2 variables contain optimal scores. The data set also contains an Intercept and transformed intercept TIntercept variable. The regression coefficients are in the transformation columns, which also contain the variables to which they apply.

Output Data Set Contents

Table 120.7 summarizes the various matrices that can result from PROC TRANSREG processing and that appear in the OUT= data set. The exact contents of an OUT= data set depends on many options.

Table 120.7 PROC TRANSREG OUT= Data Set Contents

TYPE	Contents	Options, Default Prefix
SCORE	dependent variables	DREPLACE not specified
SCORE	independent variables	IREPLACE not specified
SCORE	transformed dependent variables	default, TDPREFIX=T
SCORE	transformed independent variables	default, TIPREFIX=T
SCORE	predicted values	PREDICTED, PPREFIX=P
SCORE	residuals	RESIDUALS, RDPREFIX=R
SCORE	leverage	LEVERAGE, LEVERAGE=Leverage
SCORE	lower individual confidence limits	CLI, LILPREFIX=LIL, CILPREFIX=CIL
SCORE	upper individual confidence limits	CLI, LIUPREFIX=LIU, CIUPREFIX=CIU
SCORE	lower mean confidence limits	CLM, LMLPREFIX=LML, CMLPREFIX=CML
SCORE	upper mean confidence limits	CLM, LMUPREFIX=LMU, CMUPREFIX=CMU
SCORE	dependent canonical variables	CANONICAL, CDPREFIX=Cand
SCORE	independent canonical variables	CANONICAL, CIPREFIX=Cani
SCORE	redundancy variables	REDUNDANCY, RPREFIX=Red
SCORE	ID, CLASS, BSPLINE variables	ID, CLASS, BSPLINE,
SCORE	independent variables approximations	IAPPROXIMATIONS, AIPREFIX=A
M COEFFI	multiple regression coefficients	COEFFICIENTS, MRC
C COEFFI	canonical coefficients	COEFFICIENTS, CCC
MEAN	marginal means	COEFFICIENTS, MEANS
M REDUND	multiple redundancy coefficients	MREDUNDANCY
R REDUND	multiple redundancy coefficients	MREDUNDANCY
M POINT	point coordinates	COORDINATES or MPC, POINT
M EPOINT	elliptical point coordinates	COORDINATES or MEC, EPOINT
M QPOINT	quadratic point coordinates	COORDINATES or MQC, QPOINT
C POINT	canonical point coordinates	COORDINATES or CPC, POINT
C EPOINT	canonical elliptical point coordinates	COORDINATES or CEC, EPOINT
C QPOINT	canonical quadratic point coordinates	COORDINATES or CQC, QPOINT

The independent and dependent variables are created from the original input data. Several potential differences exist between these variables and the actual input data. An intercept variable can be added, new variables can be added for POINT, EPOINT, QPOINT, CLASS, IDENTITY, PSPLINE, and BSPLINE variables, and category numbers are substituted for character OPSCORE variables. These matrices are not always what is input to the first iteration. After the expanded data set is stored for inclusion in the output data set, several things happen to the data before they are input to the first iteration: column means are substituted for missing values; zero-degree SPLINE and MSPLINE variables are transformed so that the iterative algorithms get step-function data as input, which conform to the zero-degree transformation family restrictions; and the nonoptimal transformations are performed.

Details for the UNIVARIATE Method

When you specify METHOD=UNIVARIATE (in the MODEL or PROC TRANSREG statement), PROC TRANSREG can perform several analyses, one for each dependent variable. While each dependent variable can be transformed, their independent variables are not transformed. The OUT= data set optionally contains all of the _TYPE_='SCORE' observations, optionally followed by coefficients or coordinates.

Details for the MORALS Method

When you specify METHOD=MORALS (in the MODEL or PROC TRANSREG statement), successive analyses are performed, one for each dependent variable. Each analysis transforms one dependent variable and the entire set of the independent variables. All information for the first dependent variable (scores then, optionally, coefficients) appears first. Then all information for the second dependent variable (scores then, optionally, coefficients) appears next. This arrangement is repeated for all dependent variables.

Details for the CANALS and REDUNDANCY Methods

For METHOD=CANALS and METHOD=REDUNDANCY (specified in either the MODEL or PROC TRANSREG statement), one analysis is performed that simultaneously transforms all dependent and independent variables. The OUT= data set optionally contains all of the _TYPE_='SCORE' observations, optionally followed by coefficients or coordinates.

Variable Names

As shown in the preceding examples, some variables in the output data set directly correspond to input variables, and some are created. All original optimal and nonoptimal transformation variable names are unchanged.

The names of the POINT, QPOINT, and EPOINT expansion variables are also left unchanged, but new variables are created. When independent POINT variables are present, the sum-of-squares variable _ISSQ_ is added to the output data set. For each EPOINT and QPOINT variable, a new squared variable is created by appending "_2". For example, Dim1 and Dim2 are expanded into Dim1, Dim2, Dim1_2, and Dim2_2. In addition, for each pair of QPOINT variables, a new crossproduct variable is created by combining the two names—for example, Dim1Dim2.

The names of the CLASS variables are constructed from original variable names and levels. Lengths are controlled by the CPREFIX= *a-option*. For example, when x1 and x2 both have values of 'a' and 'b', CLASS(x1 | x2 / ZERO=NONE) creates x1 main-effect variable names x1a x1b, x2 main-effect variable names x2a x2b, and interaction variable names x1ax2a x1ax2b x1bx2a x1bx2b.

PROC TRANSREG then uses these variable names when creating the transformed, predicted, and residual variable names by affixing the relevant prefix and dropping extra characters if necessary.

METHOD=MORALS Variable Names

When you specify METHOD=MORALS and only one dependent variable is present, the output data set is structured exactly as if METHOD=REDUNDANCY (see the section "Details for the CANALS and REDUNDANCY Methods" on page 10011). When more than one dependent variable is present, the dependent variables are output in the variable _DEPEND_, transformed dependent variables are output in the variable T_DEPEND_, predicted values are output in the variable P_DEPEND_, and residuals are output in the variable R_DEPEND_. You can partition the data set into BY groups, one per dependent variable, by referring to the character variable _DEPVAR_, which contains the original dependent variable names and transformations.

Duplicate Variable Names

When the same name is generated from multiple variables in the OUT= data set, new names are created by appending '2', '3', or '4', and so on, until a unique name is created. For 32-character names, the last character is replaced with a numeric suffix until a unique name is created. For example, if there are two output variables that otherwise would be named x, then x and x2 are created instead. If there are two output variables that otherwise would be named ThisIsAThirtyTwoCharacterVarName, then ThisIsAThirtyTwoCharacterVarName and ThisIsAThirtyTwoCharacterVarNam2 are created instead.

OUTTEST= Output Data Set

The OUTTEST= data set contains hypothesis test results. The OUTTEST= data set always contains ANOVA results. When you specify the SS2 *a-option*, regression tables are also output. When you specify the UTILITIES *a-option*, conjoint analysis part-worth utilities are also output. The OUTTEST= data set has the following variables:

DEPVAR	is a 42-character variable that contains the dependent variable transformation and name.
TYPE	is an 8-character variable that contains the table type. The first character is "U" for univariate or "M" for multivariate. The second character is blank. The third character is "A" for ANOVA, "2" for Type II sum of squares, or "U" for UTILITIES. The fourth character is blank. The fifth character is "L" for liberal tests, "C" for conservative tests, or "U" for the usual tests.
Title	is an 80-character variable that contains the table title.
Variable	is a 42-character variable that contains the independent variable transformations and names for regression tables and blanks for ANOVA tables.
Coefficient	contains the multiple regression coefficients for regression tables and underscore special missing values for ANOVA tables.
Statistic	is a 24-character variable that contains the names for statistics in other variables, such as Value.
Value	contains multivariate test statistics and all other information that does not fit in one of the other columns including R square, dependent mean, adjusted R square, and coefficient of variation. Whenever Value is not an underscore special missing value, the Statistic variable describes the contents of the Value variable.
NumDF	contains numerator degrees of freedom for F tests.
DenDF	contains denominator degrees of freedom for F tests.
SSq	contains sums of squares.
MeanSquare	contains mean squares.
F	contains F statistics.
NumericP	contains the p-value for the F statistic, stored in a numeric variable.
P	is a 9-character variable that contains the formatted p-value for the F statistic, including the appropriate $\sim$, <=, >=, or blank symbols.
LowerLimit	contains lower confidence limits on the parameter estimates.

UpperLimit	contains upper confidence limits on the parameter estimates.
StdError	contains standard errors. For SS2 and UTILITIES tables, standard errors are output for each coefficient with one degree of freedom.
Importance	contains the relative importance of each factor for UTILITIES tables.
Label	is a 256-character variable that contains variable labels.

There are several possible tables in the OUTTEST= data set corresponding to combinations of univariate and multivariate tests; ANOVA and regression results; and liberal, conservative, and the usual tests. Each table is composed of only a subset of the variables. Numeric variables contain underscore special missing values when they are not a column in a table. Ordinary missing values (.) appear in variables that are part of a table when a nonmissing value cannot be produced. For example, the F is missing for a test with zero degrees of freedom.

Computational Resources

This section provides information about the computational resources required to use PROC TRANSREG.

Let

$$
\begin{aligned}
n &= \text{number of observations} \\
q &= \text{number of expanded independent variables} \\
r &= \text{number of expanded dependent variables} \\
k &= \text{maximum spline degree} \\
p &= \text{maximum number of knots}
\end{aligned}
$$

More than $56(q + r)$ plus the maximum of the data matrix size, the optimal scaling work space, and the covariance matrix size bytes of array space are required. The data matrix size is $8n(q + r)$ bytes. The optimal scaling work space requires less than $8(6n + (p + k + 2)(p + k + 11))$ bytes. The covariance matrix size is $4(q + r)(q + r + 1)$ bytes.

PROC TRANSREG tries to store the original and transformed data in memory. If there is not enough memory, a utility data set is used, potentially resulting in a large increase in execution time. The amount of memory for the preceding data formulas is an underestimate of the amount of memory needed to handle most problems. These formulas give the absolute minimum amount of memory required. If a utility data set is used, and if memory can be used with perfect efficiency, then roughly the amount of memory stated previously is needed. In reality, most problems require at least two or three times the minimum.

PROC TRANSREG sorts the data once. The sort time is roughly proportional to $(q + r)n^{3/2}$.

One regression analysis per iteration is required to compute model parameters (or two canonical correlation analyses per iteration for METHOD=CANALS). The time required to accumulate the crossproducts matrix is roughly proportional to $n(q + r)^2$. The time required to compute the regression coefficients is roughly proportional to q^3.

Each optimal scaling is a multiple regression problem, although some transformations are handled with faster special-case algorithms. The number of regressors for the optimal scaling problems depends on the original values of the variable and the type of transformation. For each monotone spline transformation, an unknown number of multiple regressions is required to find a set of coefficients that satisfies the constraints. The B-spline basis is generated twice for each SPLINE and MSPLINE transformation for each iteration. The time required to generate the B-spline basis is roughly proportional to nk^2.

Unbalanced ANOVA without CLASS Variables

This section illustrates that an analysis of variance model can be formulated as a simple regression model with optimal scoring. The purpose of the example is to explain one aspect of how PROC TRANSREG works, not to propose an alternative way of performing an analysis of variance.

Finding the overall fit of a large, unbalanced analysis of variance model can be handled as an optimal scoring problem without creating large, sparse design matrices. For example, consider an unbalanced full main-effects and interactions ANOVA model with six factors. Assume that a SAS data set is created with factor-level indicator variables c1 through c6 and dependent variable y. If each factor level consists of nonblank single characters, you can create a cell indicator in a DATA step with the statement as follows:

```
x=compress(c1||c2||c3||c4||c5||c6);
```

The following statements optimally score x (by using the OPSCORE transformation) and do not transform y:

```
proc transreg;
   model identity(y)=opscore(x);
   output;
run;
```

The final R square reported is the R square for the full analysis of variance model. This R square is the same R square that would be reported by both of the following PROC GLM steps:

```
proc glm;
   class x;
   model y=x;
run;

proc glm;
   class c1-c6;
   model y=c1|c2|c3|c4|c5|c6;
run;
```

PROC TRANSREG optimally scores the classes of x, within the space of a single variable with values linearly related to the cell means, so the full ANOVA problem is reduced to a simple regression problem with an optimal independent variable. PROC TRANSREG requires only one iteration to find the optimal scoring of x but, by default, performs a second iteration, which reports no data changes.

Hypothesis Tests for Simple Univariate Models

If the dependent variable has one parameter (IDENTITY, LINEAR with no missing values, and so on) and if there are no monotonicity constraints, PROC TRANSREG fits univariate models, which can also be fit with a DATA step and PROC REG. This is illustrated with the following artificial data set:

```
data htex;
   do i = 0.5 to 10 by 0.5;
      x1 = log(i);
      x2 = sqrt(i) + sin(i);
      x3 = 0.05 * i * i + cos(i);
      y  = x1 - x2 + x3 + 3 * normal(7);
      x1 = x1 + normal(7);
      x2 = x2 + normal(7);
      x3 = x3 + normal(7);
      output;
   end;
run;
```

Both PROC TRANSREG and PROC REG are run to fit the same polynomial regression model as follows:

```
proc transreg data=htex ss2 short;
   title 'Fit a Polynomial Regression Model with PROC TRANSREG';
   model identity(y) = spline(x1);
run;

data htex2;
   set htex;
   x1_1 = x1;
   x1_2 = x1 * x1;
   x1_3 = x1 * x1 * x1;
run;

proc reg;
   title 'Fit a Polynomial Regression Model with PROC REG';
   model y = x1_1 - x1_3;
run; quit;
```

The ANOVA and regression tables from PROC TRANSREG are displayed in Figure 120.68. The ANOVA and regression tables from PROC REG are displayed in Figure 120.69. The SHORT *a-option* is specified with PROC TRANSREG to suppress the iteration history.

Figure 120.68 ANOVA and Regression Output from PROC TRANSREG

Fit a Polynomial Regression Model with PROC TRANSREG

The TRANSREG Procedure

Dependent Variable Identity(y)

Number of Observations Read	20
Number of Observations Used	20

Identity(y)
Algorithm converged.

Figure 120.68 *continued*

The TRANSREG Procedure Hypothesis Tests for Identity(y)

Univariate ANOVA Table Based on the Usual Degrees of Freedom

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	3	5.8365	1.94550	0.14	0.9329
Error	16	218.3073	13.64421		
Corrected Total	19	224.1438			

Root MSE	3.69381	**R-Square**	0.0260
Dependent Mean	0.85490	**Adj R-Sq**	-0.1566
Coeff Var	432.07258		

Univariate Regression Table Based on the Usual Degrees of Freedom

Variable	DF	Coefficient	Type II Sum of Squares	Mean Square	F Value	Pr > F
Intercept	1	1.4612767	18.8971	18.8971	1.38	0.2565
Spline(x1)	3	-0.3924013	5.8365	1.9455	0.14	0.9329

Figure 120.69 ANOVA and Regression Output from PROC REG

Fit a Polynomial Regression Model with PROC REG

The REG Procedure
Model: MODEL1
Dependent Variable: y

Number of Observations Read	20
Number of Observations Used	20

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	3	5.83651	1.94550	0.14	0.9329
Error	16	218.30729	13.64421		
Corrected Total	19	224.14380			

Root MSE	3.69381	**R-Square**	0.0260
Dependent Mean	0.85490	**Adj R-Sq**	-0.1566
Coeff Var	432.07258		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	1.22083	1.47163	0.83	0.4190
x1_1	1	0.79743	1.75129	0.46	0.6550
x1_2	1	-0.49381	1.50449	-0.33	0.7470
x1_3	1	0.04422	0.32956	0.13	0.8949

The PROC TRANSREG regression table differs in several important ways from the parameter estimate table produced by PROC REG. The REG procedure displays standard errors and *t* statistics. PROC TRANSREG displays Type II sums of squares, mean squares, and *F* statistics. The difference is because the numerator degrees of freedom are not always 1, so *t* tests are not uniformly appropriate. When the degrees of freedom for variable x_j is 1, the following relationships hold between the standard errors (s_{β_j}) and the Type II sums of squares (SS_j):

$$s_{\beta_j} = (\hat{\beta}_j^2 / F_j)^{1/2}$$

and

$$\mathrm{SS}_j = \hat{\beta}_j^2 \times \mathrm{MSE} / s_{\beta_j}^2$$

PROC TRANSREG does not provide tests of the individual terms that go into the transformation. (However, it could if BSPLINE or PSPLINE had been specified instead of SPLINE.) The test of **`spline(x1)`** is the same as the test of the overall model. The intercepts are different due to the different numbers of variables and their standardizations.

In the next example, both `x1` and `x2` are transformed in the first PROC TRANSREG step, and PROC TRANSREG is used instead of a DATA step to create the polynomials for PROC REG. Both PROC TRANSREG and PROC REG fit the same polynomial regression model. The following statements run PROC TRANSREG and PROC REG and produce Figure 120.70 and Figure 120.71:

```
title 'Two-Variable Polynomial Regression';

proc transreg data=htex ss2 solve;
   model identity(y) = spline(x1 x2);
run;

proc transreg noprint data=htex maxiter=0;
   /* Use PROC TRANSREG to prepare input to PROC REG */
   model identity(y) = pspline(x1 x2);
   output out=htex2;
run;

proc reg data=htex2;
   model y = x1_1-x1_3 x2_1-x2_3;
   test x1_1, x1_2, x1_3;
   test x2_1, x2_2, x2_3;
run; quit;
```

Figure 120.70 Two-Variable Polynomial Regression Output from PROC TRANSREG

Two-Variable Polynomial Regression

The TRANSREG Procedure

Dependent Variable Identity(y)

Number of Observations Read	20
Number of Observations Used	20

Figure 120.70 *continued*

TRANSREG MORALS Algorithm Iteration History for Identity(y)

Iteration Number	Average Change	Maximum Change	R-Square	Criterion Change	Note
0	0.69502	4.73421	0.08252		
1	0.00000	0.00000	0.17287	0.09035	Converged

Algorithm converged.

Hypothesis Test Iterations Excluding Spline(x1)

TRANSREG MORALS Algorithm Iteration History for Identity(y)

Iteration Number	Average Change	Maximum Change	R-Square	Criterion Change	Note
0	0.03575	0.32390	0.15097		
1	0.00000	0.00000	0.15249	0.00152	Converged

Algorithm converged.

Hypothesis Test Iterations Excluding Spline(x2)

TRANSREG MORALS Algorithm Iteration History for Identity(y)

Iteration Number	Average Change	Maximum Change	R-Square	Criterion Change	Note
0	0.45381	1.43736	0.00717		
1	0.00000	0.00000	0.02604	0.01886	Converged

Algorithm converged.

The TRANSREG Procedure Hypothesis Tests for Identity(y)

Univariate ANOVA Table Based on the Usual Degrees of Freedom

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	6	38.7478	6.45796	0.45	0.8306
Error	13	185.3960	14.26123		
Corrected Total	19	224.1438			

Root MSE	3.77640	**R-Square**	0.1729
Dependent Mean	0.85490	**Adj R-Sq**	-0.2089
Coeff Var	441.73431		

Univariate Regression Table Based on the Usual Degrees of Freedom

Variable	DF	Coefficient	Type II Sum of Squares	Mean Square	F Value	Pr > F
Intercept	1	3.5437125	35.2282	35.2282	2.47	0.1400
Spline(x1)	3	0.3644562	4.5682	1.5227	0.11	0.9546
Spline(x2)	3	-1.3551738	32.9112	10.9704	0.77	0.5315

There are three iteration histories: one for the overall model and two for the two independent variables. The first PROC TRANSREG iteration history shows the R square of 0.17287 for the fit of the overall model. The second is for the following model:

```
model identity(y) = spline(x2);
```

This model excludes `spline(x1)`. The third iteration history is for the following model:

```
model identity(y) = spline(x1);
```

This model excludes `spline(x2)`. The difference between the first and second R square times the total sum of squares is the model sum of squares for `spline(x1)`:

$$(0.17287 - 0.15249) \times 224.143800 = 4.568165$$

The difference between the first and third R square times the total sum of squares is the model sum of squares for `spline(x2)`:

$$(0.17287 - 0.02604) \times 224.143800 = 32.911247$$

Figure 120.71 displays the PROC REG results. The TEST statement in PROC REG tests the null hypothesis that the vector of parameters for x1_1 x1_2 x1_3 is zero. This is the same test as the `spline(x1)` test used by PROC TRANSREG. Similarly, the PROC REG test that the vector of parameters for x2_1 x2_2 x2_3 is zero is the same as the PROC TRANSREG SPLINE(x2) test. So for models with no monotonicity constraints and no dependent variable transformations, PROC TRANSREG provides little more than a different packaging of standard least squares methodology.

Figure 120.71 Two-Variable Polynomial Regression Output from PROC REG

Two-Variable Polynomial Regression

The REG Procedure
Model: MODEL1
Dependent Variable: y

Number of Observations Read	20
Number of Observations Used	20

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	6	38.74775	6.45796	0.45	0.8306
Error	13	185.39605	14.26123		
Corrected Total	19	224.14380			

Root MSE	3.77640	**R-Square**	0.1729
Dependent Mean	0.85490	**Adj R-Sq**	-0.2089
Coeff Var	441.73431		

Figure 120.71 *continued*

Parameter Estimates

Variable	Label	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	Intercept	1	10.77824	7.55244	1.43	0.1771
x1_1	x1 1	1	0.40112	1.81024	0.22	0.8281
x1_2	x1 2	1	0.25652	1.66023	0.15	0.8796
x1_3	x1 3	1	-0.11639	0.36775	-0.32	0.7567
x2_1	x2 1	1	-14.07054	12.50521	-1.13	0.2809
x2_2	x2 2	1	5.95610	5.97952	1.00	0.3374
x2_3	x2 3	1	-0.80608	0.87291	-0.92	0.3726

Two-Variable Polynomial Regression

The REG Procedure
Model: MODEL1

Test 1 Results for Dependent Variable y

Source	DF	Mean Square	F Value	Pr > F
Numerator	3	1.52272	0.11	0.9546
Denominator	13	14.26123		

Two-Variable Polynomial Regression

The REG Procedure
Model: MODEL1

Test 2 Results for Dependent Variable y

Source	DF	Mean Square	F Value	Pr > F
Numerator	3	10.97042	0.77	0.5315
Denominator	13	14.26123		

Hypothesis Tests with Monotonicity Constraints

Now consider a model with monotonicity constraints. This model has no counterpart in PROC REG. The following statements fit a monotone-spline model and produce Figure 120.72:

```
title 'Monotone Splines';

proc transreg data=htex ss2 short;
   model identity(y) = mspline(x1-x3 / nknots=3);
run;
```

The SHORT *a-option* is specified to suppress the iteration histories. Two ANOVA tables are displayed—one by using liberal degrees of freedom and one by using conservative degrees of freedom. All sums of squares and the R squares are the same for both tables. What differs are the degrees of freedom and statistics that use degrees of freedom. The liberal test has 8 model degrees of freedom and 11 error degrees of freedom, whereas the conservative test has 15 model degrees of freedom and only 4 error degrees of freedom. The “true” *p*-value is between 0.8462 and 0.9997, so clearly you would fail to reject the null hypothesis. Unfortunately, results are not always this clear. (See Figure 120.72.)

Figure 120.72 Monotone Spline Transformations

Monotone Splines

The TRANSREG Procedure

Dependent Variable Identity(y)

Number of Observations Read	20
Number of Observations Used	20

Identity(y)
Algorithm converged.

The TRANSREG Procedure Hypothesis Tests for Identity(y)

Univariate ANOVA Table Based on Liberal Degrees of Freedom

Source	DF	Sum of Squares	Mean Square	F Value	Liberal p
Model	8	58.0534	7.25667	0.48	>= 0.8462
Error	11	166.0904	15.09913		
Corrected Total	19	224.1438			

Root MSE	3.88576	**R-Square**	0.2590
Dependent Mean	0.85490	**Adj R-Sq**	-0.2799
Coeff Var	454.52581		

Univariate ANOVA Table Based on Conservative Degrees of Freedom

Source	DF	Sum of Squares	Mean Square	F Value	Conservative p
Model	15	58.0534	3.87022	0.09	<= 0.9997
Error	4	166.0904	41.52261		
Corrected Total	19	224.1438			

Root MSE	6.44380	**R-Square**	0.2590
Dependent Mean	0.85490	**Adj R-Sq**	-2.5197
Coeff Var	753.74578		

Univariate Regression Table Based on Liberal Degrees of Freedom

Variable	DF	Coefficient	Type II Sum of Squares	Mean Square	F Value	Liberal p
Intercept	1	4.8687676	54.7372	54.7372	3.63	>= 0.0834
Mspline(x1)	2	-0.6886834	12.1943	6.0972	0.40	>= 0.6773
Mspline(x2)	3	-1.8237319	46.3155	15.4385	1.02	>= 0.4199
Mspline(x3)	3	0.8646155	24.6840	8.2280	0.54	>= 0.6616

Figure 120.72 *continued*

Univariate Regression Table Based on Conservative Degrees of Freedom

Variable	DF	Coefficient	Type II Sum of Squares	Mean Square	F Value	Conservative p
Intercept	1	4.8687676	54.7372	54.7372	1.32	<= 0.3149
Mspline(x1)	5	-0.6886834	12.1943	2.4389	0.06	<= 0.9959
Mspline(x2)	5	-1.8237319	46.3155	9.2631	0.22	<= 0.9344
Mspline(x3)	5	0.8646155	24.6840	4.9368	0.12	<= 0.9809

Hypothesis Tests with Dependent Variable Transformations

PROC TRANSREG can also provide approximate tests of hypotheses when the dependent variable is transformed, but the output is more complicated. When a dependent variable has more than one degree of freedom, the problem becomes multivariate. Hypothesis tests are performed in the context of a multivariate linear model with the number of dependent variables equal to the number of scoring parameters for the dependent variable transformation. The transformation regression model with a dependent variable transformation differs from the usual multivariate linear model in two important ways. First, the usual assumption of multivariate normality is always violated. This fact is simply ignored. This is one reason why all hypothesis tests in the presence of a dependent variable transformation should be considered approximate at best. Multivariate normality is assumed even though it is known that the assumption is violated.

The second difference concerns the usual multivariate test statistics: Pillai's trace, Wilks' lambda, Hotelling-Lawley trace, and Roy's greatest root. The first three statistics are defined in terms of all the squared canonical correlations. Here, there is only one linear combination (the transformation), and hence only one squared canonical correlation of interest, which is equal to the R square. It might seem that Roy's greatest root, which uses only the largest squared canonical correlation, is the only statistic of interest. Unfortunately, Roy's greatest root is very liberal and provides only a lower bound on the *p*-value. Approximate upper bounds are provided by adjusting the other three statistics for the one linear combination case. Wilks' lambda, Pillai's trace, and Hotelling-Lawley trace are a conservative adjustment of the usual statistics.

These statistics are normally defined in terms of the squared canonical correlations, which are the eigenvalues of the matrix $\mathbf{H}(\mathbf{H}+\mathbf{E})^{-1}$, where $\mathbf{H}$ is the hypothesis sum-of-squares matrix and $\mathbf{E}$ is the error sum-of-squares matrix. Here the R square is used for the first eigenvalue, and all other eigenvalues are set to 0 since only one linear combination is used. Degrees of freedom are computed assuming that all linear combinations contribute to the lambda and trace statistics, so the *F* tests for those statistics are conservative. The *p*-values for the liberal and conservative statistics provide approximate lower and upper bounds on *p*. In practice, the adjusted Pillai's trace is very conservative—perhaps too conservative to be useful. Wilks' lambda is less conservative, and the Hotelling-Lawley trace seems to be the least conservative. The conservative statistics and the liberal Roy's greatest root provide a bound on the true *p*-value. Unfortunately, they sometimes report a bound of 0.0001 and 1.0000.

The following example has a dependent variable transformation and produces Figure 120.73:

```
title 'Transform Dependent and Independent Variables';

proc transreg data=htex ss2 solve short;
   model spline(y) = spline(x1-x3);
run;
```

The univariate results match Roy's greatest root results. Clearly, the proper action is to fail to reject the null hypothesis. However, as stated previously, results are not always this clear.

Figure 120.73 Transform Dependent and Independent Variables

Transform Dependent and Independent Variables

The TRANSREG Procedure

Dependent Variable Spline(y)

Number of Observations Read	20
Number of Observations Used	20

Spline(y)
Algorithm converged.

The TRANSREG Procedure Hypothesis Tests for Spline(y)

Univariate ANOVA Table Based on the Usual Degrees of Freedom

Source	DF	Sum of Squares	Mean Square	F Value	Liberal p
Model	9	110.8822	12.32025	1.09	>= 0.4452
Error	10	113.2616	11.32616		
Corrected Total	19	224.1438			

The above statistics are not adjusted for the fact that the dependent variable was transformed and so are generally liberal.

Root MSE	3.36544	**R-Square**	0.4947
Dependent Mean	0.85490	**Adj R-Sq**	0.0399
Coeff Var	393.66234		

Adjusted Multivariate ANOVA Table Based on the Usual Degrees of Freedom

Dependent Variable Scoring Parameters=3 S=3 M=2.5 N=3

Statistic	Value	F Value	Num DF	Den DF	p
Wilks' Lambda	0.505308	0.23	27	24.006	<= 0.9998
Pillai's Trace	0.494692	0.22	27	30	<= 0.9999
Hotelling-Lawley Trace	0.978992	0.26	27	11.589	<= 0.9980
Roy's Greatest Root	0.978992	1.09	9	10	>= 0.4452

The Wilks' Lambda, Pillai's Trace, and Hotelling-Lawley Trace statistics are a conservative adjustment of the normal statistics. Roy's Greatest Root is liberal. These statistics are normally defined in terms of the squared canonical correlations which are the eigenvalues of the matrix H*inv(H+E). Here the R-Square is used for the first eigenvalue and all other eigenvalues are set to zero since only one linear combination is used. Degrees of freedom are computed assuming all linear combinations contribute to the Lambda and Trace statistics, so the F tests for those statistics are conservative. The p values for the liberal and conservative statistics provide approximate lower and upper bounds on p. A liberal test statistic with conservative degrees of freedom and a conservative test statistic with liberal degrees of freedom yield at best an approximate p value, which is indicated by a "~" before the p value.

Figure 120.73 *continued*

Univariate Regression Table Based on the Usual Degrees of Freedom						
Variable	DF	Coefficient	Type II Sum of Squares	Mean Square	F Value	Liberal p
Intercept	1	6.9089087	117.452	117.452	10.37	>= 0.0092
Spline(x1)	3	-1.0832321	32.493	10.831	0.96	>= 0.4504
Spline(x2)	3	-2.1539191	45.251	15.084	1.33	>= 0.3184
Spline(x3)	3	0.4779207	10.139	3.380	0.30	>= 0.8259

The above statistics are not adjusted for the fact that the dependent variable was transformed and so are generally liberal.

Adjusted Multivariate Regression Table Based on the Usual Degrees of Freedom							
Variable	Coefficient	Statistic	Value	F Value	Num DF	Den DF	p
Intercept	6.9089087	Wilks' Lambda	0.49092	2.77	3	8	0.1112
		Pillai's Trace	0.50908	2.77	3	8	0.1112
		Hotelling-Lawley Trace	1.036993	2.77	3	8	0.1112
		Roy's Greatest Root	1.036993	2.77	3	8	0.1112
Spline(x1)	-1.0832321	Wilks' Lambda	0.777072	0.24	9	19.621	<= 0.9840
		Pillai's Trace	0.222928	0.27	9	30	<= 0.9787
		Hotelling-Lawley Trace	0.286883	0.24	9	9.8113	<= 0.9784
		Roy's Greatest Root	0.286883	0.96	3	10	>= 0.4504
Spline(x2)	-2.1539191	Wilks' Lambda	0.714529	0.32	9	19.621	<= 0.9572
		Pillai's Trace	0.285471	0.35	9	30	<= 0.9494
		Hotelling-Lawley Trace	0.399524	0.33	9	9.8113	<= 0.9424
		Roy's Greatest Root	0.399524	1.33	3	10	>= 0.3184
Spline(x3)	0.4779207	Wilks' Lambda	0.917838	0.08	9	19.621	<= 0.9998
		Pillai's Trace	0.082162	0.09	9	30	<= 0.9996
		Hotelling-Lawley Trace	0.089517	0.07	9	9.8113	<= 0.9997
		Roy's Greatest Root	0.089517	0.30	3	10	>= 0.8259

These statistics are adjusted in the same way as the multivariate statistics above.

Hypothesis Tests with One-Way ANOVA

One-way ANOVA models are fit with either an explicit or implicit intercept. In implicit intercept models, the ANOVA table of PROC TRANSREG is the correct table for a model with an intercept, and the regression table is the correct table for a model that does not have a separate explicit intercept. The PROC TRANSREG implicit intercept ANOVA table matches the PROC REG table when the NOINT *a-option* is not specified, and the PROC TRANSREG implicit intercept regression table matches the PROC REG table when the NOINT *a-option* is specified. The following statements illustrate this relationship and produce Figure 120.74:

```
data oneway;
   input y x $;
   datalines;
0 a
1 a
2 a
7 b
8 b
9 b
3 c
4 c
5 c
;

title 'Implicit Intercept Model';

proc transreg ss2 data=oneway short;
   model identity(y) = class(x / zero=none);
   output out=oneway2;
run;

proc reg data=oneway2;
   model y = xa xb xc;          /* Implicit Intercept ANOVA      */
   model y = xa xb xc / noint; /* Implicit Intercept Regression */
run; quit;
```

Figure 120.74 Implicit Intercept Model

Implicit Intercept Model

The TRANSREG Procedure

Dependent Variable Identity(y)

Class Level Information		
Class	**Levels**	**Values**
x	3	a b c

Number of Observations Read	9
Number of Observations Used	9
Implicit Intercept Model	

The TRANSREG Procedure Hypothesis Tests for Identity(y)

Univariate ANOVA Table Based on the Usual Degrees of Freedom

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	2	74.00000	37.00000	37.00	0.0004
Error	6	6.00000	1.00000		
Corrected Total	8	80.00000			

Figure 120.74 *continued*

Root MSE	1.00000	**R-Square**	0.9250
Dependent Mean	4.33333	**Adj R-Sq**	0.9000
Coeff Var	23.07692		

Univariate Regression Table Based on the Usual Degrees of Freedom

Variable	DF	Coefficient	Type II Sum of Squares	Mean Square	F Value	Pr > F	Label
Class.xa	1	1.00000000	3.000	3.000	3.00	0.1340	x a
Class.xb	1	8.00000000	192.000	192.000	192.00	<.0001	x b
Class.xc	1	4.00000000	48.000	48.000	48.00	0.0004	x c

Implicit Intercept Model

The REG Procedure
Model: MODEL1
Dependent Variable: y

Number of Observations Read	9
Number of Observations Used	9

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	2	74.00000	37.00000	37.00	0.0004
Error	6	6.00000	1.00000		
Corrected Total	8	80.00000			

Root MSE	1.00000	**R-Square**	0.9250
Dependent Mean	4.33333	**Adj R-Sq**	0.9000
Coeff Var	23.07692		

Note: Model is not full rank. Least-squares solutions for the parameters are not unique. Some statistics will be misleading. A reported DF of 0 or B means that the estimate is biased.

Note: The following parameters have been set to 0, since the variables are a linear combination of other variables as shown.

xc =	Intercept - xa - xb

Parameter Estimates

Variable	Label	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	Intercept	B	4.00000	0.57735	6.93	0.0004
xa	x a	B	-3.00000	0.81650	-3.67	0.0104
xb	x b	B	4.00000	0.81650	4.90	0.0027
xc	x c	0	0	.	.	.

Figure 120.74 *continued*

Implicit Intercept Model

The REG Procedure
Model: MODEL2
Dependent Variable: y

Number of Observations Read	9
Number of Observations Used	9

Note: No intercept in model. R-Square is redefined.

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	3	243.00000	81.00000	81.00	<.0001
Error	6	6.00000	1.00000		
Uncorrected Total	9	249.00000			

Root MSE	1.00000	**R-Square**	0.9759
Dependent Mean	4.33333	**Adj R-Sq**	0.9639
Coeff Var	23.07692		

Parameter Estimates

Variable	Label	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
xa	x a	1	1.00000	0.57735	1.73	0.1340
xb	x b	1	8.00000	0.57735	13.86	<.0001
xc	x c	1	4.00000	0.57735	6.93	0.0004

Using the DESIGN Output Option

This example uses PROC TRANSREG and the DESIGN *o-option* to prepare an input data set with classification variables for the LOGISTIC procedure. The DESIGN *o-option* specifies that the goal is design matrix creation, not analysis. When you specify DESIGN, dependent variables are not required. The DEVIATIONS (or EFFECTS) *t-option* requests a deviations-from-means $(1, 0, -1)$ coding of the classification variables, which is the same coding the CATMOD procedure uses. PROC TRANSREG automatically creates a macro variable &_TrgInd that contains the list of independent variables created. This macro is used in the PROC LOGISTIC MODEL statement. (See Figure 120.75.) For comparison, the same analysis is also performed with PROC CATMOD. The following statements create Figure 120.75:

```
title 'Using PROC TRANSREG to Create a Design Matrix';

data a;
   do y = 1, 2;
      do a = 1 to 4;
         do b = 1 to 3;
            w = ceil(uniform(1) * 10 + 10);
            output;
         end;
```

```
      end;
   end;
run;

proc transreg data=a design;
   model class(a b / deviations);
   id y w;
   output out=coded;
run;

proc print;
   title2 'PROC TRANSREG Output Data Set';
run;

title2 'PROC LOGISTIC with Classification Variables';

proc logistic;
   freq w;
   model y = &_trgind;
run;

title2 'PROC CATMOD Should Produce the Same Results';

proc catmod data=a;
   model y = a b;
   weight w;
run;
```

Figure 120.75 The PROC TRANSREG Design Matrix

Using PROC TRANSREG to Create a Design Matrix
PROC LOGISTIC with Classification Variables

The LOGISTIC Procedure

Model Information	
Data Set	WORK.CODED
Response Variable	y
Number of Response Levels	2
Frequency Variable	w
Model	binary logit
Optimization Technique	Fisher's scoring

Number of Observations Read	24
Number of Observations Used	24
Sum of Frequencies Read	375
Sum of Frequencies Used	375

Figure 120.75 *continued*

Response Profile		
Ordered Value	**y**	**Total Frequency**
1	1	188
2	2	187

Probability modeled is y=1.

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

Model Fit Statistics		
Criterion	**Intercept Only**	**Intercept and Covariates**
AIC	521.858	524.378
SC	525.785	547.939
-2 Log L	519.858	512.378

Testing Global Null Hypothesis: BETA=0			
Test	**Chi-Square**	**DF**	**Pr > ChiSq**
Likelihood Ratio	7.4799	5	0.1873
Score	7.4312	5	0.1905
Wald	7.3356	5	0.1969

Analysis of Maximum Likelihood Estimates					
Parameter	**DF**	**Estimate**	**Standard Error**	**Wald Chi-Square**	**Pr > ChiSq**
Intercept	1	-0.00040	0.1044	0.0000	0.9969
a1	1	-0.0802	0.1791	0.2007	0.6542
a2	1	0.2001	0.1800	1.2363	0.2662
a3	1	-0.1350	0.1819	0.5514	0.4578
b1	1	-0.2392	0.1500	2.5436	0.1107
b2	1	0.3433	0.1474	5.4223	0.0199

Odds Ratio Estimates			
Effect	**Point Estimate**	**95% Wald Confidence Limits**	
a1	0.923	0.650	1.311
a2	1.222	0.858	1.738
a3	0.874	0.612	1.248
b1	0.787	0.587	1.056
b2	1.410	1.056	1.882

Association of Predicted Probabilities and Observed Responses			
Percent Concordant	54.0	**Somers' D**	0.163
Percent Discordant	37.8	**Gamma**	0.177
Percent Tied	8.2	**Tau-a**	0.082
Pairs	35156	**c**	0.581

Figure 120.75 *continued*

Using PROC TRANSREG to Create a Design Matrix
PROC CATMOD Should Produce the Same Results

The CATMOD Procedure

Data Summary			
Response	y	**Response Levels**	2
Weight Variable	w	**Populations**	12
Data Set	A	**Total Frequency**	375
Frequency Missing	0	**Observations**	24

Population Profiles

Sample	a	b	Sample Size
1	1	1	31
2	1	2	31
3	1	3	34
4	2	1	26
5	2	2	33
6	2	3	37
7	3	1	36
8	3	2	29
9	3	3	28
10	4	1	26
11	4	2	35
12	4	3	29

Response Profiles

Response	y
1	1
2	2

Maximum Likelihood Analysis
Maximum likelihood computations converged.

Maximum Likelihood Analysis of Variance

Source	DF	Chi-Square	Pr > ChiSq
Intercept	1	0.00	0.9969
a	3	1.50	0.6823
b	2	5.64	0.0597
Likelihood Ratio	6	2.81	0.8329

Figure 120.75 *continued*

Analysis of Maximum Likelihood Estimates

Parameter		Estimate	Standard Error	Chi-Square	Pr > ChiSq
Intercept		-0.00040	0.1044	0.00	0.9969
a	1	-0.0802	0.1791	0.20	0.6542
	2	0.2001	0.1800	1.24	0.2662
	3	-0.1350	0.1819	0.55	0.4578
b	1	-0.2392	0.1500	2.54	0.1107
	2	0.3434	0.1474	5.42	0.0199

Discrete Choice Experiments: DESIGN, NORESTORE, NOZERO

A discrete choice experiment is constructed consisting of four product brands, each available at three different prices, $1.49, $1.99, $2.49. In addition, each choice set contains a constant "other" alternative available at $1.49. In the fifth choice set, price is constant. PROC TRANSREG is used to code the design, and the PHREG procedure fits the multinomial logit choice model (not shown). See Kuhfeld (2010) for more information about discrete choice modeling and the multinomial logit model; look for the latest "Discrete Choice" report. The following statements produce Figure 120.76:

```
title 'Choice Model Coding';

data design;
   array p[4];
   input p1-p4 @@;
   set = _n_;
   do brand = 1 to 4;
      price = p[brand];
      output;
   end;
   brand = .; price = 1.49; output; /* constant alternative */
   keep set brand price;
   datalines;
1.49 1.99 1.49 1.99 1.99 1.99 2.49 1.49 1.99 1.49 1.99 1.49
1.99 1.49 2.49 1.99 1.49 1.49 1.49 1.49 2.49 1.49 1.99 2.49
1.49 1.49 2.49 2.49 2.49 2.49 1.49 1.49 1.49 2.49 2.49 1.99
2.49 2.49 2.49 1.49 1.99 2.49 1.49 2.49 2.49 1.99 2.49 2.49
2.49 1.49 1.49 1.99 1.49 1.99 1.99 1.49 2.49 1.99 1.99 1.99
1.99 1.99 1.49 2.49 1.99 2.49 1.99 1.99 1.49 2.49 1.99 2.49
;

proc transreg data=design design norestoremissing nozeroconstant;
   model class(brand / zero=none) identity(price);
   output out=coded;
   by set;
run;

proc print data=coded(firstobs=21 obs=25);
   var set brand &_trgind;
run;
```

In the interest of space, only the fifth choice set is displayed in Figure 120.76.

Figure 120.76 The Fifth Choice Set

Choice Model Coding

Obs	set	brand	brand1	brand2	brand3	brand4	price
21	5	1	1	0	0	0	1.49
22	5	2	0	1	0	0	1.49
23	5	3	0	0	1	0	1.49
24	5	4	0	0	0	1	1.49
25	5	.	0	0	0	0	1.49

For the constant alternative (Brand = .), the brand coding is a row of zeros due to the NORESTOREMISSING *o-option*, and Price is a constant \$1.49 (instead of 0) due to the NOZEROCONSTANT.

The data set was coded by choice set (BY set;). This is a small problem. With very large problems, it might be necessary to restrict the number of observations that are coded at one time so that the procedure uses less time and memory. Coding by choice set is one option. When coding is performed after the data are merged in, coding by subject and choice set combinations is another option. Alternatively, you can specify DESIGN=*n*, where *n* is the number of observations to code at one time. For example, you can specify DESIGN=100 or DESIGN=1000 to process the data set in blocks of 100 or 1000 observations. Specify the NOZEROCONSTANT *a-option* to ensure that constant variables within blocks are not zeroed. When you specify DESIGN=*n*, or perform coding after the data are merged in, specify the dependent variable and any other variables needed for analysis as ID variables.

Centering

You can use transformation options to center and standardize the variables in several ways. For example, the following MODEL statement creates three independent variables, x, x^2, and x^3:

```
model identity(y) = pspline(x);
```

The variables are not centered.

When the CENTER *t-option* is specified, as in the following statement, the independent variable is centered before squaring and cubing:

```
model identity(y) = pspline(x / center);
```

The three independent variables are $x - \bar{x}$, $(x - \bar{x})^2$, and $(x - \bar{x})^3$.

Since operations such as squaring occur after the centering, the resulting variables are not always centered. The CENTER *t-option* is particularly useful with polynomials since centering before squaring and cubing can help reduce collinearity and numerical problems. For example, if one of your variables is year, with values all greater than 1900, squaring and cubing without centering first will create variables that are all essentially perfectly correlated.

When the TSTANDARD=CENTER *t-option* is specified, as in the following model, the three independent variables are squared and cubed and then centered:

```
model identity(y) = pspline(x / tstandard=center);
```

The three independent variables are $x - \bar{x}$, $x^2 - \overline{x^2}$, and $x^3 - \overline{x^3}$.

You can specify both the CENTER and TSTANDARD=CENTER *t-options* to center the variables, then square and cube them, and then center the results, as in the following statement:

```
model identity(y) = pspline(x / center tstandard=center);
```

The three independent variables are $x - \bar{x}$, $(x - \bar{x})^2 - \overline{(x - \bar{x})^2}$, and $(x - \bar{x})^3 - \overline{(x - \bar{x})^3}$.

Displayed Output

The display options control the amount of displayed output. The displayed output can contain the following:

- an iteration history and convergence status table (by default when there are iterations)
- an ANOVA table when the TEST, SS2, or UTILITIES *a-option* is specified
- a regression table when the SS2 *a-option* is specified
- conjoint analysis part-worth utilities when the UTILITIES *a-option* is specified
- model details when the DETAIL *a-option* is specified
- a multivariate ANOVA table when the dependent variable is transformed and the TEST or SS2 *a-option* is specified
- a multivariate regression table when the dependent variable is transformed and it is specified
- liberal and conservative ANOVA, multivariate ANOVA, regression, and multivariate regression tables when there is a MONOTONE, UNTIE, or MSPLINE transformation and the TEST or SS2 *a-option* is specified

ODS Table Names

PROC TRANSREG assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in Table 120.8. For more information about ODS, see Chapter 20, "Using the Output Delivery System."

Table 120.8 ODS Tables Produced by PROC TRANSREG

ODS Table Name	Description	Statement & Option
ANOVA	ANOVA	MODEL/PROC, TEST/SS2
BoxCox	Box-Cox transformation results	MODEL, BOXCOX
CANALS	CANALS iteration history	MODEL/PROC, METHOD=CANALS

Table 120.8 *continued*

ODS Table Name	Description	Statement & Option
ClassLevels	ANOVA	MODEL/PROC, TEST/SS2
Coef	Regression results	MODEL/PROC, SS2
ConservANOVA	ANOVA, *1	MODEL/PROC, TEST/SS2
ConservCoef	Regression results, *1	MODEL/PROC, SS2
ConservFitStatistics	Fit statistics, *1	MODEL/PROC, TEST/SS2
ConservMVANOVA	Multivariate ANOVA, *1, *2	MODEL/PROC, TEST/SS2
ConservMVCoef	Multivariate regression results, *1, *2	MODEL/PROC, SS2
ConservUtilities	Conjoint analysis utilities, *1	MODEL/PROC, UTILITIES
ConvergenceStatus	Convergence status	default
Details	Model Details	MODEL/PROC, DETAIL
Equation	Linear dependency equation	less-than-full-rank model
FitStatistics	Fit statistics like R square	MODEL/PROC, TEST/SS2
Footnotes	Iteration history footnotes	default
Formula	Fit plot formula (nonprinting)	PROC, PLOTS=FIT(FORMULA)
LiberalANOVA	ANOVA, *1	MODEL/PROC, TEST/SS2
LiberalCoef	Regression results, *1	MODEL/PROC, SS2
LiberalFitStatistics	Fit statistics, *1	MODEL/PROC, TEST/SS2
LiberalMVANOVA	Multivariate ANOVA, *1, *2	MODEL/PROC, TEST/SS2
LiberalMVCoef	Multivariate regression results, *1, *2	MODEL/PROC, SS2
LiberalUtilities	Conjoint analysis utilities, *1	MODEL/PROC, UTILITIES
MORALS	MORALS iteration history	MODEL/PROC, METHOD=MORALS
MVANOVA	Multivariate ANOVA, *2	MODEL/PROC, TEST/SS2
MVCoef	Multivariate regression results, *2	MODEL/PROC, SS2
NObs	ANOVA	MODEL/PROC, TEST/SS2
PBSplineCriteria	Penalized B-spline criteria (nonprinting)	MODEL, PBSPLINE
RSquare	R square	MODEL/PROC, RSQUARE
Redundancy	Redundancy iteration history	MODEL/PROC, METHOD=REDUNDANCY
SplineCoef	Spline coefficients (nonprinting)	MODEL, SPLINE/MSPLINE
TestIterations	Hypothesis test iterations iteration history	MODEL/PROC, SS2
Univariate	Univariate iteration history	MODEL/PROC, METHOD=UNIVARIATE
Utilities	Conjoint analysis utilities	MODEL/PROC, UTILITIES

*1. Liberal and conservative test tables are produced when a MONOTONE, UNTIE, or MSPLINE transformation is requested.

*2. Multivariate tables are produced when the dependent variable is iteratively transformed.

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 615 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 614 in Chapter 21, "Statistical Graphics Using ODS."

Some graphs are produced by default; other graphs are produced by using statements and options. You can reference every graph produced through ODS Graphics with a name. The names of the graphs that PROC TRANSREG generates are listed in Table 120.9, along with the required statements and options.

Table 120.9 Graphs Produced by PROC TRANSREG

ODS Graph Name	Plot Description	Statement & Option
BoxCoxFPlot	Box-Cox $F = t^2$	MODEL & PROC, BOXCOX *transform* & PLOTS(UNPACK)
BoxCoxLogLikePlot	Box-Cox Log Likelihood	MODEL & PROC, BOXCOX *transform* & PLOTS(UNPACK)
BoxCoxPlot	Box-Cox t or $F = t^2$ & Log Likelihood	MODEL, BOXCOX *transform*
BoxCoxtPlot	Box-Cox t	MODEL & PROC, BOXCOX *transform* & PLOTS(UNPACK)=BOXCOX(T)
FitPlot	Simple Regression and Separate Group Regressions	MODEL, a dependent variable that is not transformed, one non-CLASS independent variable, and at most one CLASS variable
ObservedByPredicted	Dependent Variable by Predicted Values	MODEL, PLOTS=OBSERVEDBYPREDICTED
PBSPlineCritPlot	Penalized B-Spline Criterion Plot	MODEL, PBSPLINE *transform*
PrefMapVecPlot	Preference Mapping Vector Plot	MODEL & PROC, IDENTITY *transform* & COORDINATES
PrefMapIdealPlot	Preference Mapping Ideal Point Plot	MODEL & PROC, POINT expansion & COORDINATES
ResidualPlot	Residuals	PROC, PLOTS=RESIDUALS
RMSEPlot	Box-Cox Root Mean Square Error	MODEL & PROC, BOXCOX *transform* & PLOTS=BOXCOX(RMSE)
ScatterPlot	Scatter Plot of Observed Data	MODEL, one non-CLASS independent variable, and at most one CLASS variable, PLOTS=SCATTER
TransformationPlot	Variable Transformations	PROC, PLOTS=TRANSFORMATION

The PLOTS(INTERPOLATE) Option

This section illustrates one use of the PLOTS(INTERPOLATE) option for use with ODS Graphics. The data set has two groups of observations, c = 1 and c = 2. Each group is sparse, having only five observations, so the plots of the transformations and fit functions are not smooth. A second DATA step adds additional observations to the data set, over the range of x, with y missing. These observations do not contribute to the analysis, but they are used in computations of transformed and predicted values. The resulting plots are much smoother in the latter case than in the former. The other results of the analysis are the same. The following statements produce Figure 120.77 and Figure 120.78:

```
title 'Smoother Interpolation with PLOTS(INTERPOLATE)';

data a;
   input c y x;
   output;
   datalines;
1 1 1
1 2 2
1 4 3
1 6 4
1 7 5
2 3 1
2 4 2
2 5 3
2 4 4
2 5 5
;

ods graphics on;

proc transreg data=a plots=(tran fit) ss2;
   model ide(y) = pbs(x) * class(c / zero=none);
run;

data b;
   set a end=eof;
   output;
   if eof then do;
      y = .;
      do x = 1 to 5 by 0.05;
         c = 1; output;
         c = 2; output;
      end;
   end;
run;

proc transreg data=b plots(interpolate)=(tran fit) ss2;
   model ide(y) = pbs(x) * class(c / zero=none);
run;
```

The results with no interpolation are shown in Figure 120.77. The transformation and fit functions are not at all smooth. The results with interpolation are shown in Figure 120.78. The transformation and fit functions are smooth in Figure 120.78, because there are intermediate points to plot.

Figure 120.77 No Interpolation

Smoother Interpolation with PLOTS(INTERPOLATE)

The TRANSREG Procedure

Univariate ANOVA Table, Penalized B-Spline Transformation

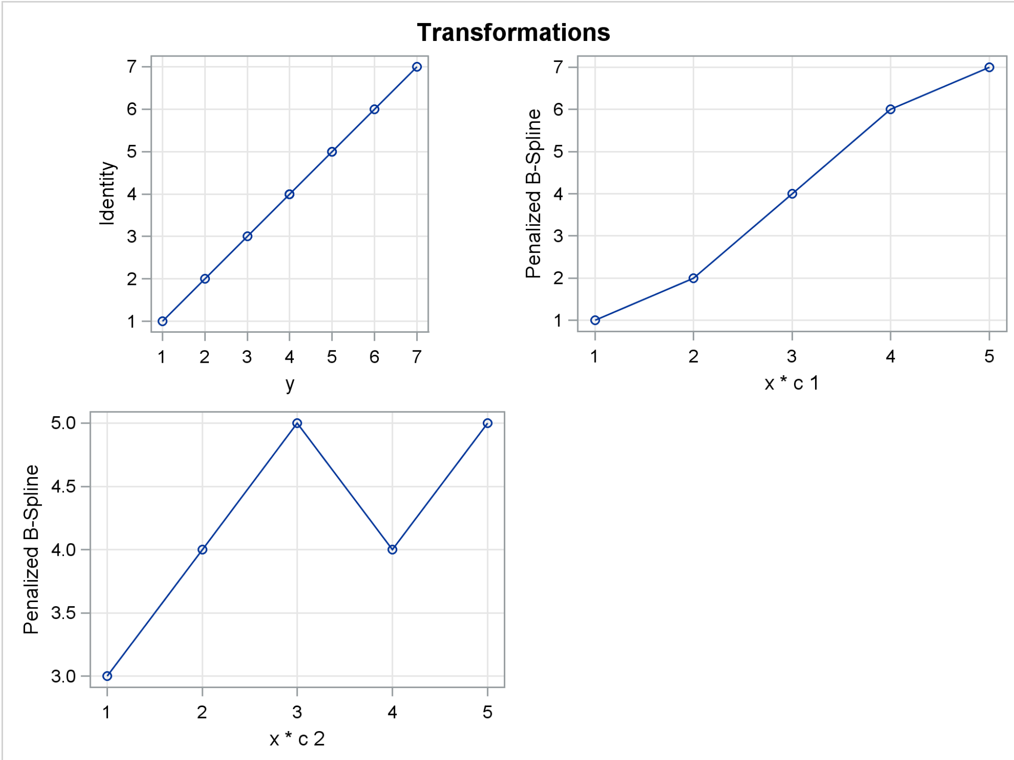
Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	9	28.90000	3.211111	Infty	<.0001
Error	12E-10	0.00000	0.000000		
Corrected Total	9	28.90000			

Root MSE	0	**R-Square**	1.0000
Dependent Mean	4.10000	**Adj R-Sq**	1.0000
Coeff Var	0		

Penalized B-Spline Transformation

Variable	DF	Coefficient	Lambda	AICC	Label
Pbspline(xc1)	5.0000	1.000	2.642E-7	-66.4281	x * c 1
Pbspline(xc2)	5.0000	1.000	2.516E-7	-60.6430	x * c 2

Figure 120.77 *continued*



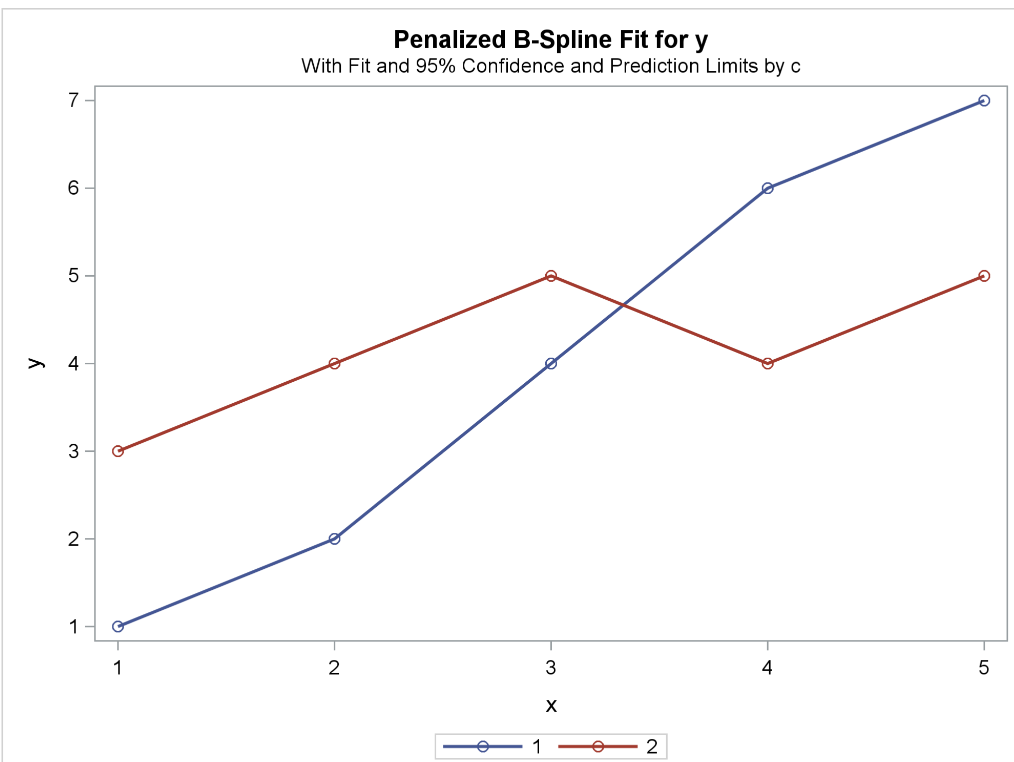


Figure 120.78 Interpolation with PLOTS(INTERPOLATE)

Smoother Interpolation with PLOTS(INTERPOLATE)

The TRANSREG Procedure

Univariate ANOVA Table, Penalized B-Spline Transformation

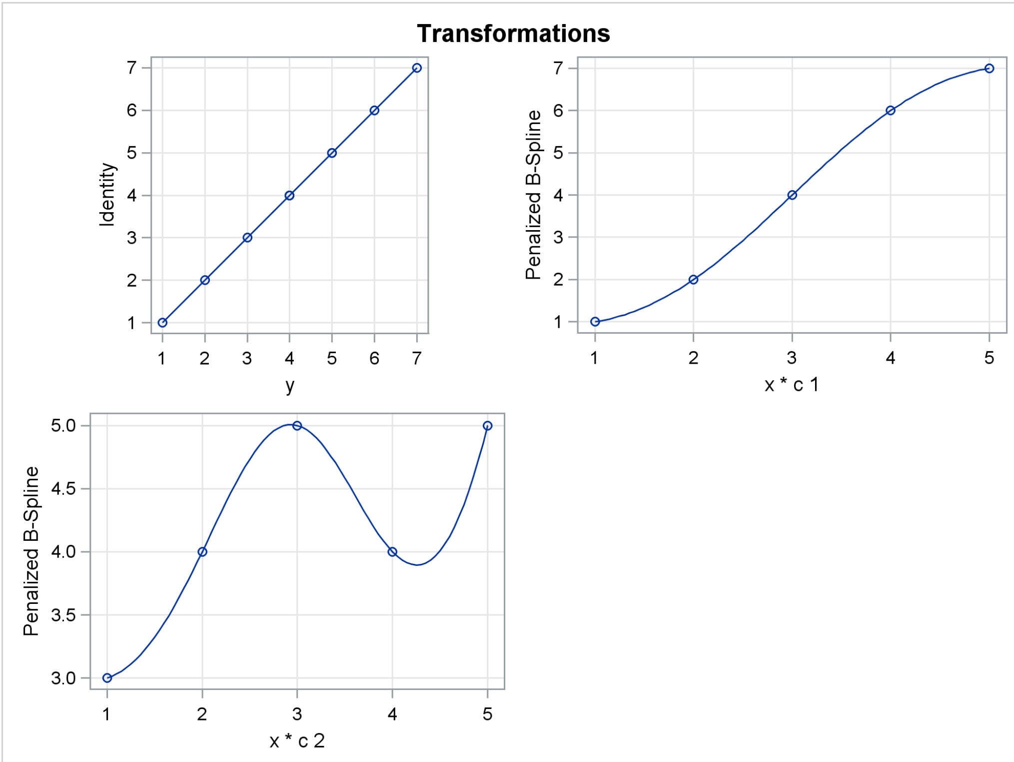
Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	9	28.90000	3.211111	Infty	<.0001
Error	12E-10	0.00000	0.000000		
Corrected Total	9	28.90000			

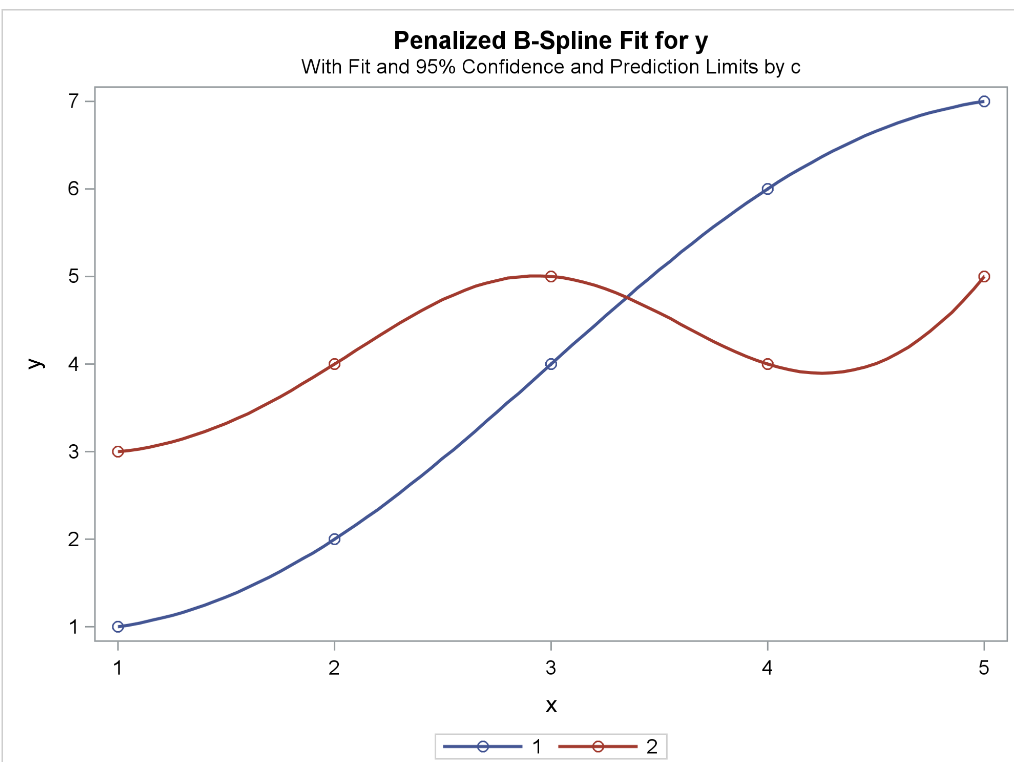
Root MSE	0	**R-Square**	1.0000
Dependent Mean	4.10000	**Adj R-Sq**	1.0000
Coeff Var	0		

Penalized B-Spline Transformation

Variable	DF	Coefficient	Lambda	AICC	Label
Pbspline(xc1)	5.0000	1.000	2.642E-7	-66.4281	x * c 1
Pbspline(xc2)	5.0000	1.000	2.516E-7	-60.6430	x * c 2

Figure 120.78 *continued*





Examples: TRANSREG Procedure

Example 120.1: Transformation Regression of Exhaust Emissions Data

In this example, the data are from an experiment in which nitrogen oxide emissions from a single cylinder engine are measured for various combinations of fuel, compression ratio, and equivalence ratio. The data are provided by Brinkman (1981). This gas data set is available from the Sashelp library.

The equivalence ratio and nitrogen oxide variables are continuous and numeric, so spline transformations of these variables are requested. The spline transformation of the dependent variable is restricted to be monotonic. Each spline is degree three with nine knots (one at each decile) in order to give PROC TRANSREG a great deal of freedom in finding transformations. The compression ratio variable has only five discrete values, so an optimal scoring is requested with monotonicity constraints. The character variable Fuel is nominal, so it is optimally scored without any monotonicity constraints. Observations with missing values are excluded with the NOMISS *a-option*.

```
ods graphics on;

title 'Gasoline Example';
title2 'Iteratively Estimate NOx, CpRatio, EqRatio, and Fuel';

* Fit the Nonparametric Model;
proc transreg data=sashelp.Gas solve test nomiss plots=all;
   ods exclude where=(_path_ ? 'MV');
   model mspline(NOx / nknots=9) = spline(EqRatio / nknots=9)
                                   monotone(CpRatio) opscore(Fuel);
run;
```

Output 120.1.1 Transformation Regression Example: The Nonparametric Model

Gasoline Example
Iteratively Estimate NOx, CpRatio, EqRatio, and Fuel

The TRANSREG Procedure

Dependent Variable Mspline(NOx)
Nitrogen Oxide

Number of Observations Read	171
Number of Observations Used	169

Output 120.1.1 *continued*

TRANSREG MORALS Algorithm Iteration History for Mspline(NOx)

Iteration Number	Average Change	Maximum Change	R-Square	Criterion Change	Note
0	0.41900	3.80550	0.05241		
1	0.11984	0.83327	0.91028	0.85787	
2	0.03727	0.17688	0.93981	0.02953	
3	0.02795	0.10880	0.94969	0.00987	
4	0.02088	0.07279	0.95382	0.00413	
5	0.01530	0.05031	0.95582	0.00201	
6	0.01130	0.03922	0.95688	0.00106	
7	0.00852	0.03197	0.95748	0.00060	
8	0.00657	0.02531	0.95783	0.00035	
9	0.00510	0.01975	0.95805	0.00022	
10	0.00398	0.01534	0.95818	0.00013	
11	0.00314	0.01200	0.95827	0.00009	
12	0.00250	0.00953	0.95832	0.00005	
13	0.00199	0.00752	0.95836	0.00003	
14	0.00159	0.00594	0.95838	0.00002	
15	0.00127	0.00470	0.95839	0.00001	
16	0.00102	0.00373	0.95840	0.00001	
17	0.00081	0.00297	0.95841	0.00001	
18	0.00065	0.00237	0.95841	0.00000	
19	0.00052	0.00189	0.95841	0.00000	
20	0.00042	0.00151	0.95842	0.00000	
21	0.00033	0.00120	0.95842	0.00000	
22	0.00027	0.00096	0.95842	0.00000	
23	0.00021	0.00077	0.95842	0.00000	
24	0.00017	0.00061	0.95842	0.00000	
25	0.00014	0.00049	0.95842	0.00000	
26	0.00011	0.00039	0.95842	0.00000	
27	0.00009	0.00031	0.95842	0.00000	
28	0.00007	0.00025	0.95842	0.00000	
29	0.00006	0.00020	0.95842	0.00000	
30	0.00005	0.00016	0.95842	0.00000	Not Converged

WARNING: Failed to converge, however criterion change is less than 0.0001.

Output 120.1.1 *continued*

The TRANSREG Procedure Hypothesis Tests for Mspline(NOx)
Nitrogen Oxide

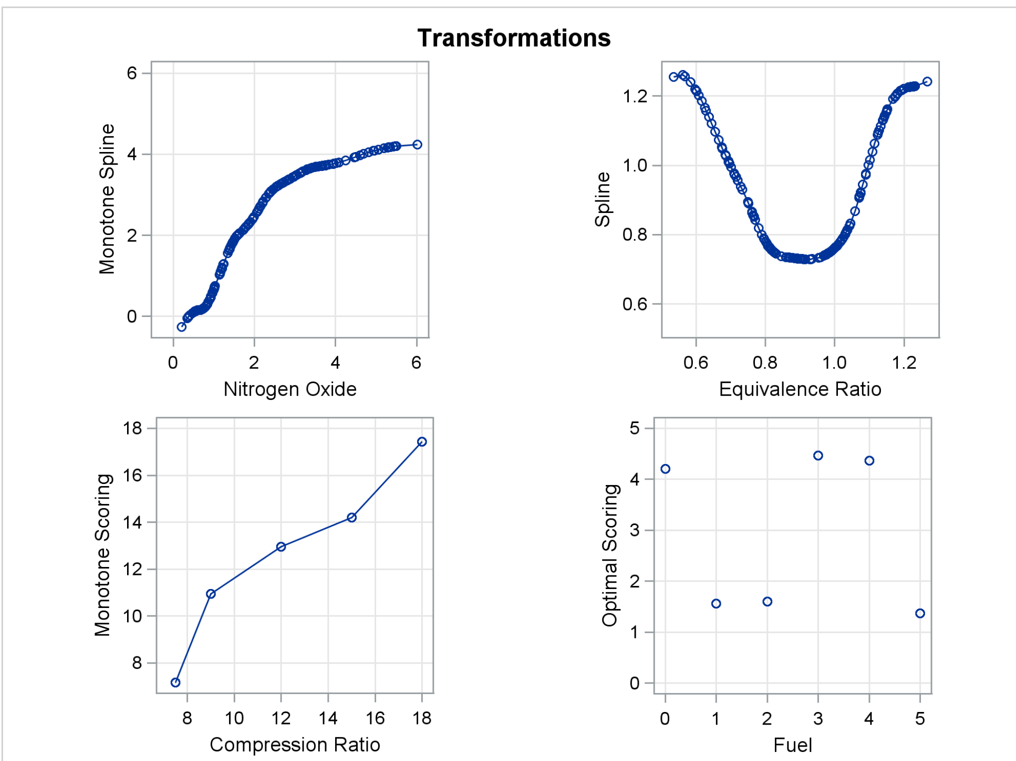
Univariate ANOVA Table Based on the Usual Degrees of Freedom

Source	DF	Sum of Squares	Mean Square	F Value	Liberal p
Model	21	326.0176	15.52465	161.35	>= <.0001
Error	147	14.1443	0.09622		
Corrected Total	168	340.1619			

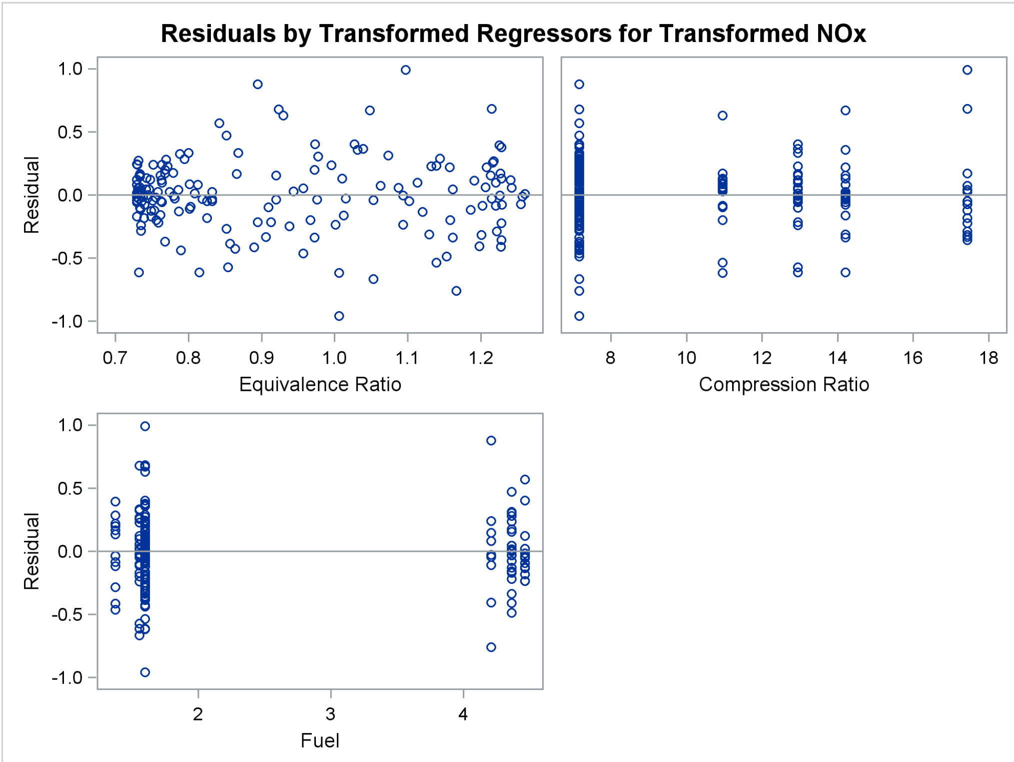
The above statistics are not adjusted for the fact that the dependent variable was transformed and so are generally liberal.

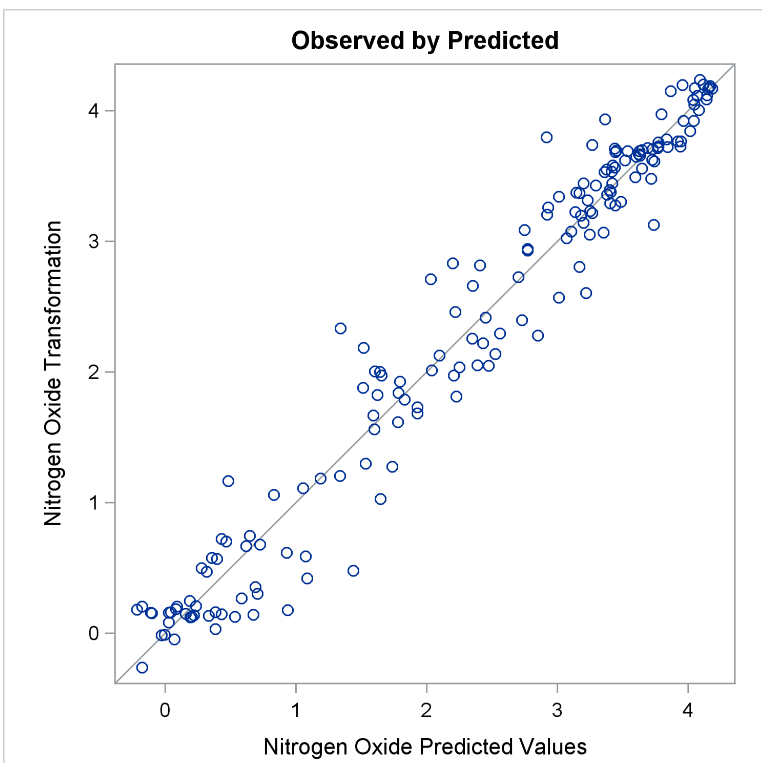
Root MSE	0.31019	**R-Square**	0.9584
Dependent Mean	2.34593	**Adj R-Sq**	0.9525
Coeff Var	13.22262		

Output 120.1.1 *continued*



Output 120.1.1 *continued*





The squared multiple correlation for the initial model is approximately 0.05. PROC TRANSREG increases the R square to over 0.95 by transforming the variables. The transformation plots show how each variable is transformed. The transformation of compression ratio (TCpRatio) is nearly linear. The transformation of equivalence ratio (TEqRatio) is nearly parabolic. It can be seen from this plot that the optimal transformation of equivalence ratio is nearly uncorrelated with the original scoring. This suggests that the large increase in R square is due to this transformation. The transformation of nitrogen oxide (TNOx) is similar to a log transformation. The final plot shows the transformed dependent variable plotted as a function of the predicted values. This plot is reasonably linear, showing that the nonlinearities in the data are being accounted for fairly well by the TRANSREG model.

These results suggest the parametric model

$$
\begin{aligned}
\log(\mathrm{NOX}) &= b_0 + b_1 \times \mathsf{EqRatio} + b_2 \times \mathsf{EqRatio}^2 + b_3 \times \mathsf{CpRatio} \\
&\quad + \sum_j b_j \,\mathrm{class}_j(\mathsf{Fuel}) + \mathrm{error}
\end{aligned}
$$

You can perform this analysis with PROC TRANSREG. The following statements produce Output 120.1.2:

```
title2 'Now fit log(NOx) = b0 + b1*EqRatio + b2*EqRatio**2 +';
title3 'b3*CpRatio + Sum b(j)*Fuel(j) + Error';

*-Fit the Parametric Model Suggested by the Nonparametric Analysis-;
proc transreg data=sashelp.Gas solve ss2 short nomiss plots=all;
   model log(NOx) = pspline(EqRatio / deg=2) identity(CpRatio)
                    opscore(Fuel);
run;
```

Output 120.1.2 Transformation Regression Example: The Parametric Model

Gasoline Example
Now fit log(NOx) = b0 + b1*EqRatio + b2*EqRatio2 +**
b3*CpRatio + Sum b(j)*Fuel(j) + Error

The TRANSREG Procedure

Dependent Variable Log(NOx)
Nitrogen Oxide

Number of Observations Read	171
Number of Observations Used	169

Log(NOx)
Algorithm converged.

Output 120.1.2 *continued*

The TRANSREG Procedure Hypothesis Tests for Log(NOx)
Nitrogen Oxide

Univariate ANOVA Table Based on the Usual Degrees of Freedom

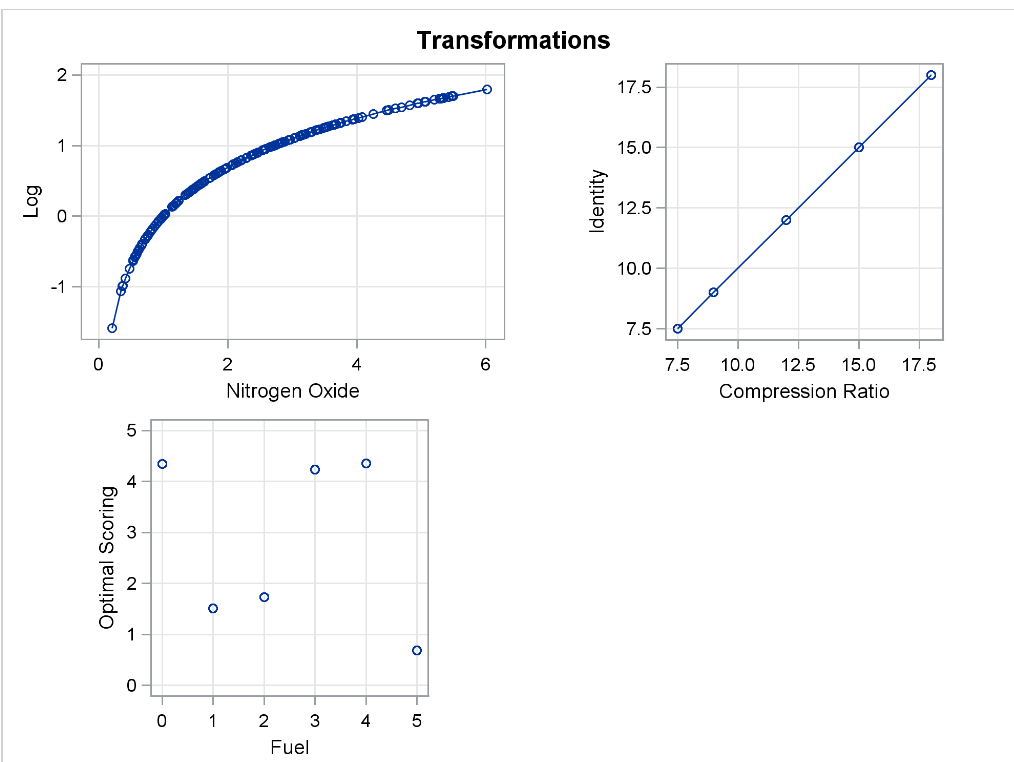
Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	8	79.33838	9.917298	213.09	<.0001
Error	160	7.44659	0.046541		
Corrected Total	168	86.78498			

Root MSE	0.21573	**R-Square**	0.9142
Dependent Mean	0.63130	**Adj R-Sq**	0.9099
Coeff Var	34.17294		

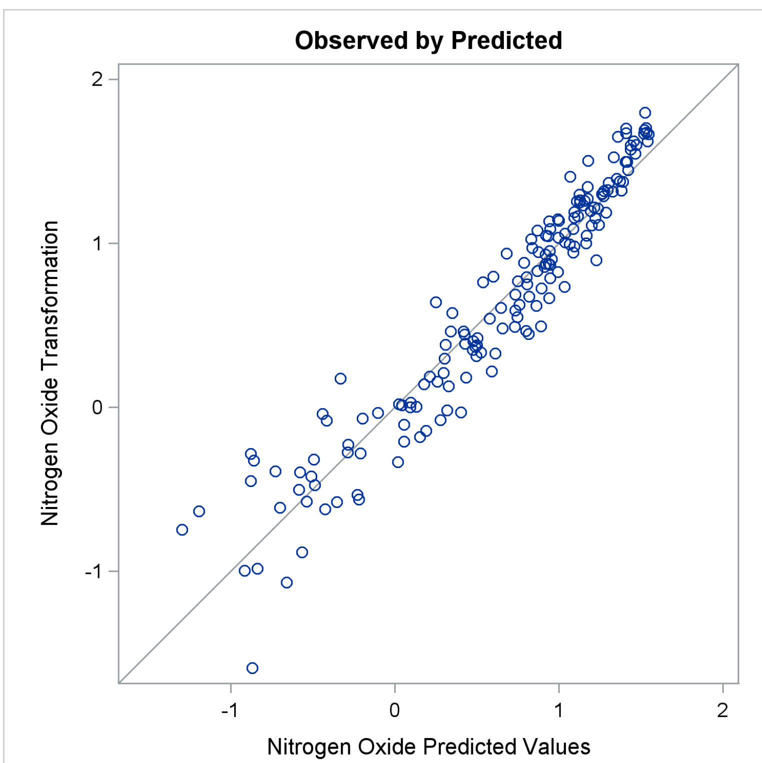
Univariate Regression Table Based on the Usual Degrees of Freedom

Variable	DF	Coefficient	Type II Sum of Squares	Mean Square	F Value	Pr > F	Label
Intercept	1	-15.274649	57.1338	57.1338	1227.60	<.0001	Intercept
Pspline.EqRatio_1	1	35.102914	62.7478	62.7478	1348.22	<.0001	Equivalence Ratio 1
Pspline.EqRatio_2	1	-19.386468	64.6430	64.6430	1388.94	<.0001	Equivalence Ratio 2
Identity(CpRatio)	1	0.032058	1.4445	1.4445	31.04	<.0001	Compression Ratio
Opscore(Fuel)	5	0.158388	5.5619	1.1124	23.90	<.0001	Fuel

Output 120.1.2 *continued*



Output 120.1.2 *continued*



The LOG transformation computes the natural log. The PSPLINE expansion expands EqRatio into a linear term, EqRatio, and a squared term, EqRatio^2. An identity transformation of CpRatio and an optimal scoring of Fuel is requested. These should provide a good parametric operationalization of the optimal transformations. The final model has an R square of 0.91 (smaller than before since the model has fewer parameters, but still quite good).

Example 120.2: Box-Cox Transformations

This example shows Box-Cox transformations with a yarn failure data set. For more information about Box-Cox transformations, including using a Box-Cox transformation in a model with no independent variable, to normalize the distribution of the data, see the section "Box-Cox Transformations" on page 9923. In this example, a simple 3^3 design was used to study the effects of different factors on the failure of a yarn manufacturing process. The design factors are as follows:

- the length of test specimens of yarn, with levels of 250, 300, and 350 mm
- the amplitude of the loading cycle, with levels of 8, 9, and 10 mmd
- the load with levels of 40, 45, and 50 grams

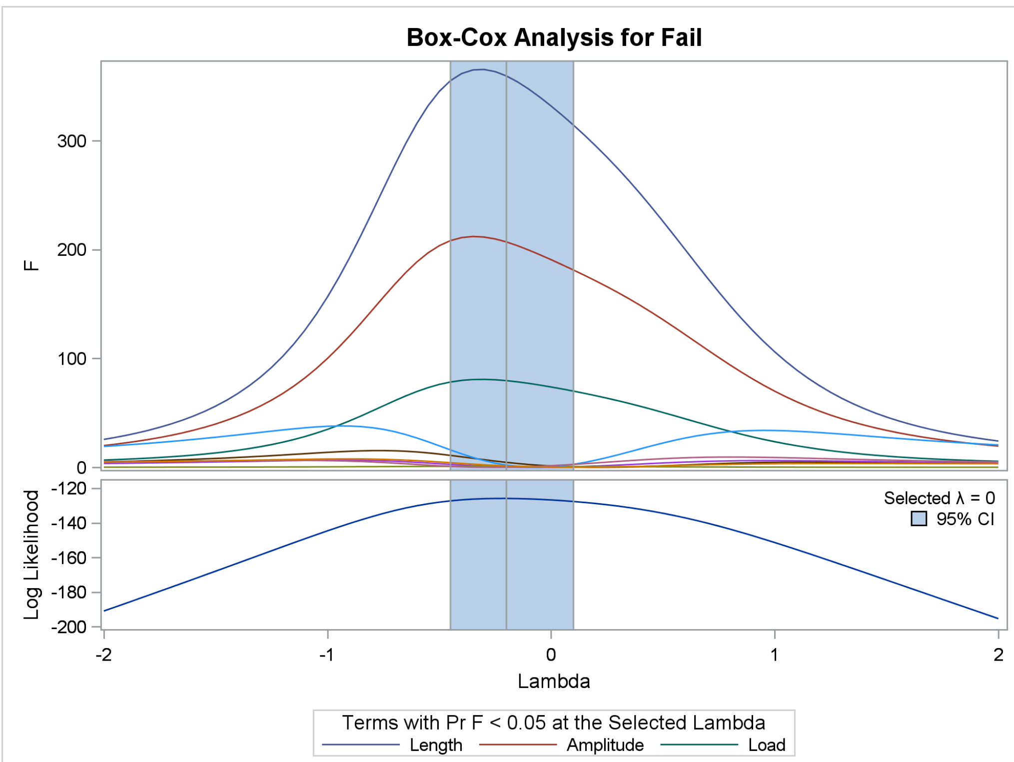
The measured response was time (in cycles) until failure. However, you could just as well have measured the inverse of time until failure (in other words, the failure rate). Hence, the correct metric with which to analyze the response is not apparent. You can use PROC TRANSREG to find an optimum power transformation for the analysis. The following statements create the input SAS data set:

```
title 'Yarn Strength';

proc format;
   value a -1 =   8 0 =   9 1 =  10;
   value l -1 = 250 0 = 300 1 = 350;
   value o -1 =  40 0 =  45 1 =  50;
run;

data yarn;
   input Fail Amplitude Length Load @@;
   format amplitude a. length l. load o.;
   label fail = 'Time in Cycles until Failure';
   datalines;
 674 -1 -1 -1    370 -1 -1  0    292 -1 -1  1    338  0 -1 -1
 266  0 -1  0    210  0 -1  1    170  1 -1 -1    118  1 -1  0
  90  1 -1  1   1414 -1  0 -1   1198 -1  0  0    634 -1  0  1
1022  0  0 -1    620  0  0  0    438  0  0  1    442  1  0 -1
 332  1  0  0    220  1  0  1   3636 -1  1 -1   3184 -1  1  0
2000 -1  1  1   1568  0  1 -1   1070  0  1  0    566  0  1  1
1140  1  1 -1    884  1  1  0    360  1  1  1
;
```

PROC TRANSREG is run to find the Box-Cox transformation. The lambda list is –2 TO 2 BY 0.05, which produces 81 lambdas, and a convenient lambda is requested. This many power parameters makes a nice graphical display with plenty of detail around the confidence interval. In the interest of space, only part of this table is displayed. The independent variables are designated with the QPOINT expansion. QPOINT, for quadratic point model, gets its name from PROC TRANSREG's ideal point modeling capabilities, which process variables for a response surface analysis. What QPOINT does is create a set of independent variables consisting of the following: the m original variables (Length Amplitude Load), the m original variables squared (Length_2 Amplitude_2 Load_2), and the $m \times (m-1)/2 = 3$ pairs of products between the m variables (LengthAmplitude LengthLoad AmplitudeLoad). The following statements produce Output 120.2.1:

```
ods graphics on;

proc transreg details data=yarn ss2
              plots=(transformation(dependent) obp);
   model BoxCox(fail / convenient lambda=-2 to 2 by 0.05) =
         qpoint(length amplitude load);
run;
```

Output 120.2.1 Box-Cox Yarn Data



Output 120.2.1 *continued*

Dependent Variable BoxCox(Fail)
Time in Cycles until Failure

Number of Observations Read	27
Number of Observations Used	27

Output 120.2.1 *continued*

Model Statement Specification Details

Type	DF	Variable	Description	Value
Dep	1	BoxCox(Fail)	Lambda Used	0
			Lambda	-0.2
			Log Likelihood	-125.9
			Conv. Lambda	0
			Conv. Lambda LL	-126.7
			CI Limit	-127.8
			Alpha	0.05
			Options	Convenient Lambda Used
			Label	Time in Cycles until Failure
Ind	1	Qpoint.Length	DF	1
Ind	1	Qpoint.Amplitude	DF	1
Ind	1	Qpoint.Load	DF	1
Ind	1	Qpoint.Length_2	DF	1
Ind	1	Qpoint.Amplitude_2	DF	1
Ind	1	Qpoint.Load_2	DF	1
Ind	1	Qpoint.LengthAmplitude	DF	1
Ind	1	Qpoint.LengthLoad	DF	1
Ind	1	Qpoint.AmplitudeLoad	DF	1

The TRANSREG Procedure Hypothesis Tests for BoxCox(Fail)
Time in Cycles until Failure

Univariate ANOVA Table Based on the Usual Degrees of Freedom

Source	DF	Sum of Squares	Mean Square	F Value	Liberal p
Model	9	22.56498	2.507220	66.73	>= <.0001
Error	17	0.63871	0.037571		
Corrected Total	26	23.20369			

The above statistics are not adjusted for the fact that the dependent variable was transformed and so are generally liberal.

Root MSE	0.19383	**R-Square**	0.9725
Dependent Mean	6.33466	**Adj R-Sq**	0.9579
Coeff Var	3.05987	**Lambda**	0.0000

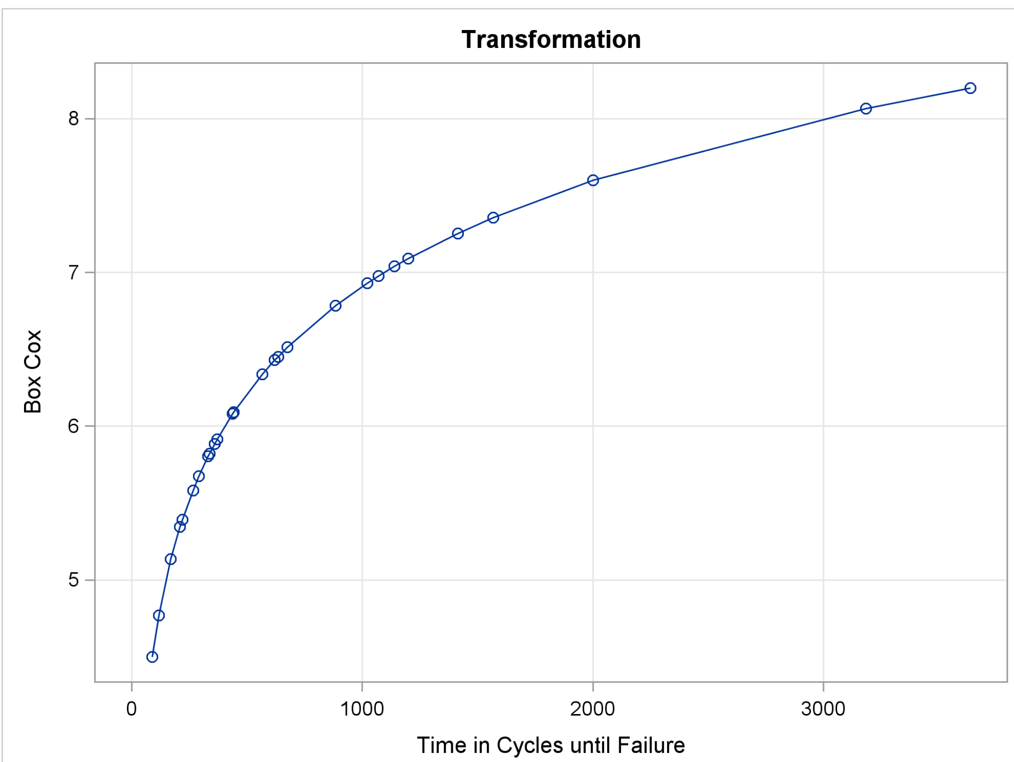
Output 120.2.1 *continued*

Univariate Regression Table Based on the Usual Degrees of Freedom

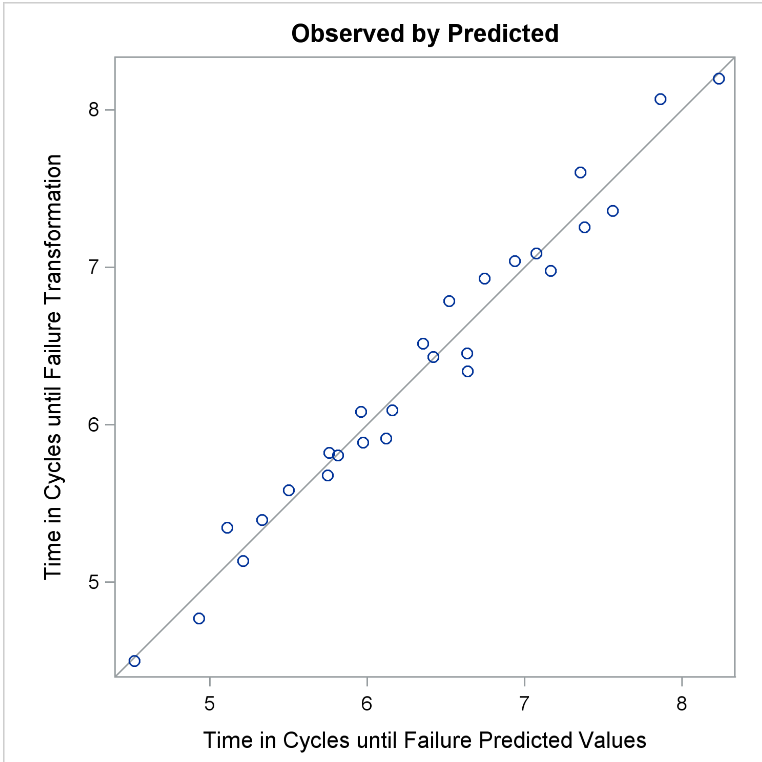
Variable	DF	Coefficient	Type II Sum of Squares	Mean Square	F Value	Liberal p	Label
Intercept	1	6.4206207	159.008	159.008	4232.19	>= <.0001	Intercept
Qpoint.Length	1	0.8323842	12.472	12.472	331.94	>= <.0001	Length
Qpoint.Amplitude	1	-0.6309916	7.167	7.167	190.75	>= <.0001	Amplitude
Qpoint.Load	1	-0.3924940	2.773	2.773	73.80	>= <.0001	Load
Qpoint.Length_2	1	-0.0856974	0.044	0.044	1.17	>= 0.2939	Length_2
Qpoint.Amplitude_2	1	0.0242183	0.004	0.004	0.09	>= 0.7633	Amplitude_2
Qpoint.Load_2	1	-0.0674555	0.027	0.027	0.73	>= 0.4058	Load_2
Qpoint.LengthAmplitude	1	-0.0382414	0.018	0.018	0.47	>= 0.5035	LengthAmplitude
Qpoint.LengthLoad	1	-0.0684146	0.056	0.056	1.49	>= 0.2381	LengthLoad
Qpoint.AmplitudeLoad	1	-0.0208340	0.005	0.005	0.14	>= 0.7142	AmplitudeLoad

The above statistics are not adjusted for the fact that the dependent variable was transformed and so are generally liberal.

Output 120.2.1 *continued*



Output 120.2.1 *continued*



The optimal power parameter is –0.20, but since 0.0 is in the confidence interval, and since the CONVENIENT *t-option* was specified, the procedure chooses a log transformation. The $F = t^2$ plot shows in the vicinity of the optimal Box-Cox transformation, the parameters for the three original variables (Length Amplitude Load), particularly Length, are significant and the others become essentially zero.

Example 120.3: Penalized B-Spline

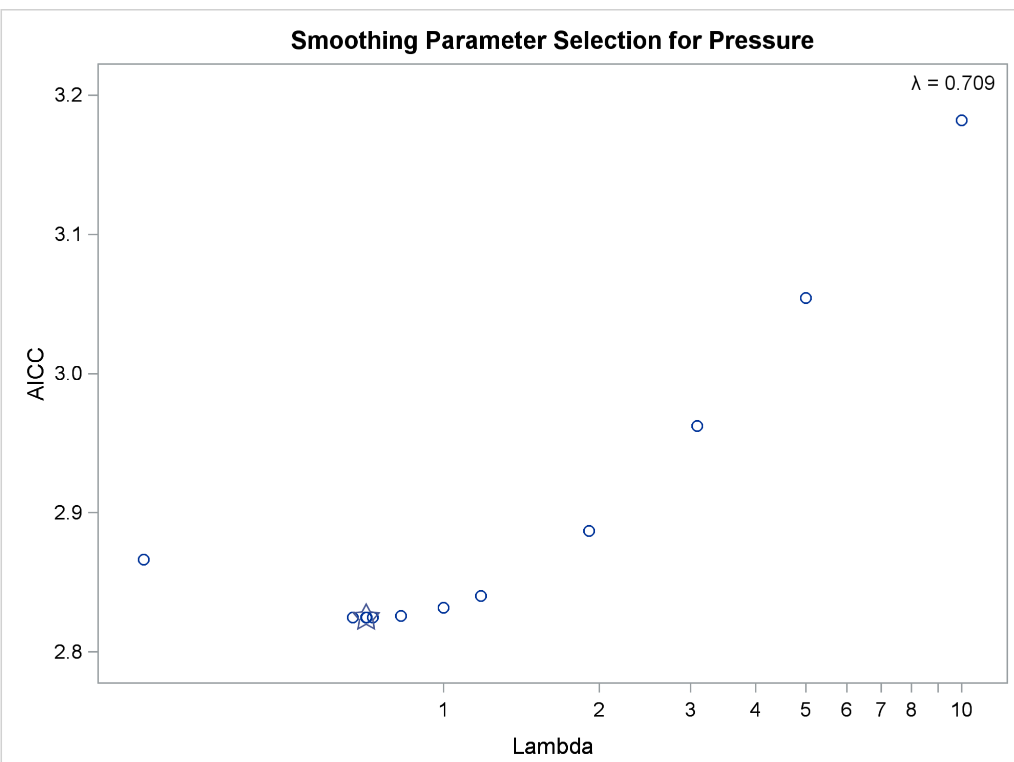
The ENSO data set contains measurements of monthly averaged atmospheric pressure differences between Easter Island and Darwin, Australia, for a period of 168 months (National Institute of Standards and Technology 1998). The ENSO data set is available from the Sashelp library.

You can fit a curve through these data by using a penalized B-spline (Eilers and Marx 1996) function and the following statements:

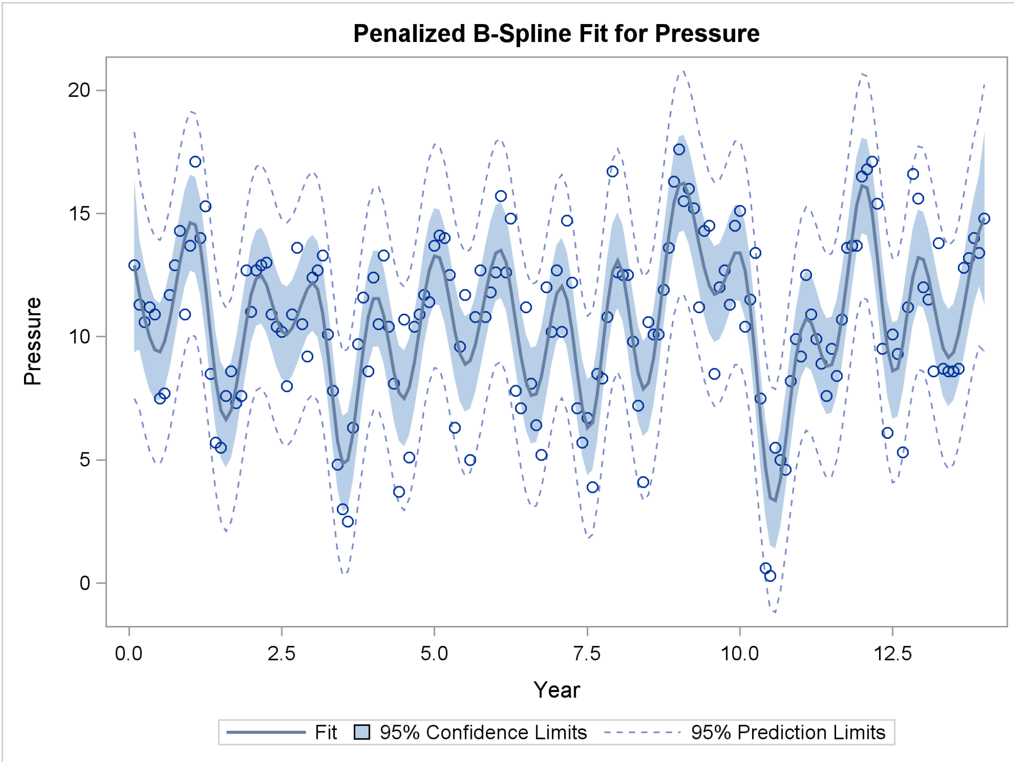
```
title 'Atmospheric Pressure Changes Between'
      ' Easter Island & Darwin, Australia';
ods graphics on;

proc transreg data=sashelp.enso;
   model identity(pressure) = pbspline(year);
run;
```

The dependent variable Pressure is specified along with an IDENTITY transformation, so Pressure is analyzed as is, with no transformations. The independent variable Year is specified with a PBSPLINE transformation, so a penalized B-spline model is fit. By default, a DEGREE=3 B-spline basis is used along with 100 evenly spaced knots and three evenly spaced exterior knots on each side of the data. The penalized spline function is typically much smoother than you would get by using a SPLINE transformation or a BSPLINE expansion since changes in the coefficients of the basis are penalized to make a smoother fit. The output is shown next in Output 120.3.1.

Output 120.3.1 Change in Atmospheric Pressure, AICC



Output 120.3.1 *continued*

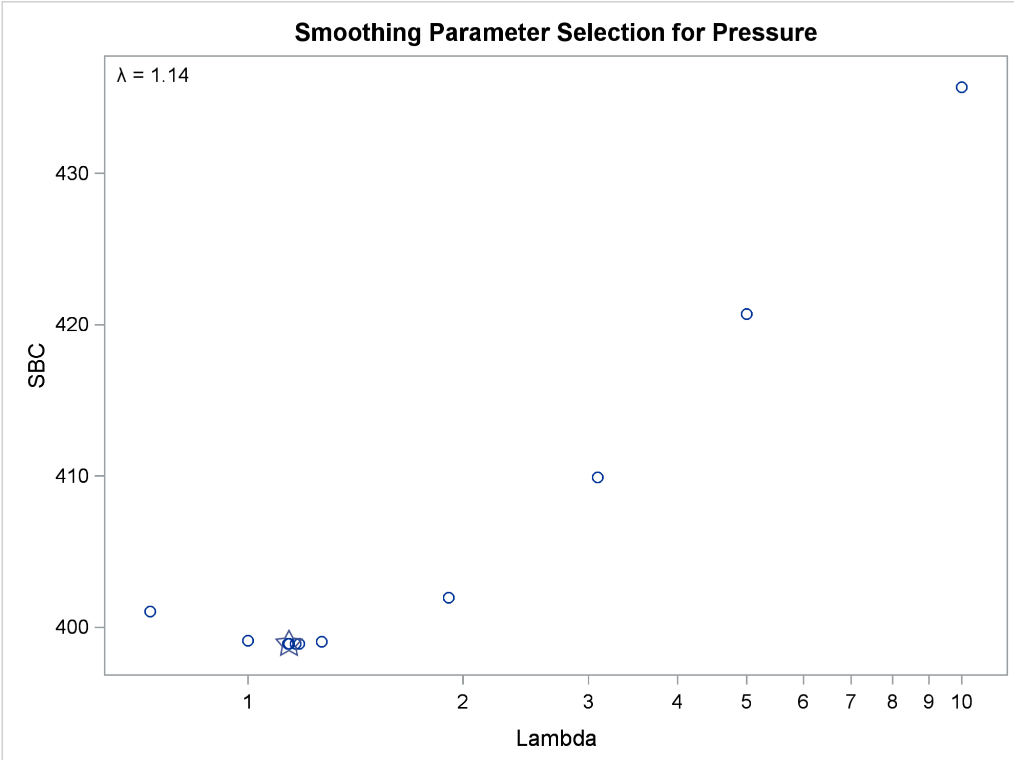


The results show a yearly cycle of pressure change. The procedure chose a smoothing parameter of $\lambda = 0.709$. With data such as these, with many peaks and valleys, it might be useful to perform another analysis, this time asking for a smoother plot. The Schwarz Bayesian criterion (SBC) is sometimes a better choice than the default criterion when you want a smoother plot. The following PROC TRANSREG step requests a penalized B-spline analysis minimizing the SBC criterion:

```
proc transreg data=sashelp.enso;
   model identity(pressure) = pbspline(year / sbc);
run;
```

The plot of SBC as a function of λ is shown in Output 120.3.2.

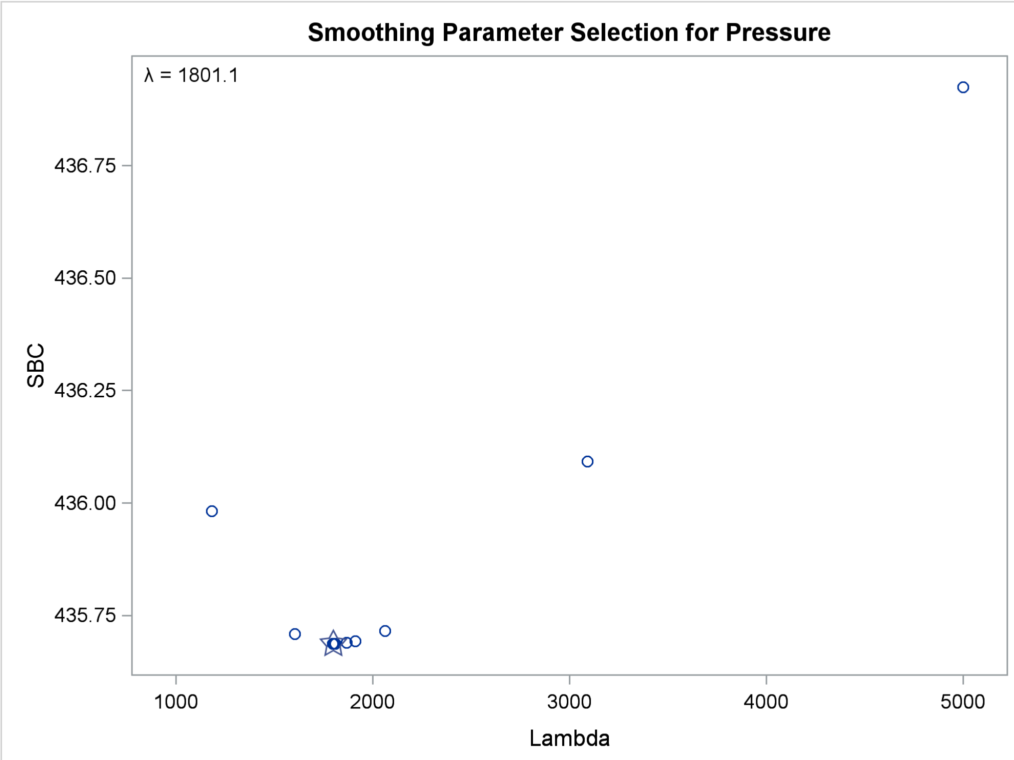
Output 120.3.2 Change in Atmospheric Pressure, SBC



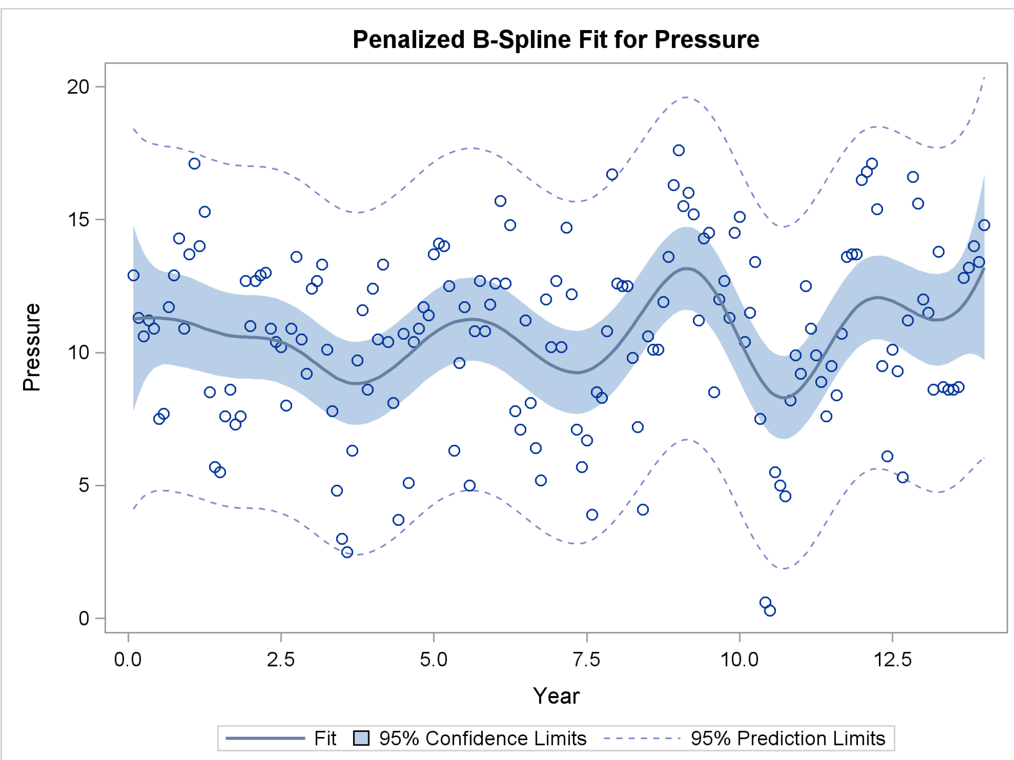
The fit plot (not shown) is essentially the same as the one shown in Output 120.3.1 due to the similar choice of smoothing parameters: $\lambda = 0.709$ versus $\lambda = 1.14$. You can analyze these data again, this time forcing PROC TRANSREG to consider only larger smoothing parameters. The specification LAMBDA=2 10000 RANGE eliminates from consideration the two lambdas that you previously saw and considers only $2 \le \lambda \le 10,000$. The following statements produce Output 120.3.3:

```
proc transreg data=sashelp.enso;
   model identity(pressure) = pbspline(year / sbc lambda=2 10000 range);
run;
```

Output 120.3.3 Change in Atmospheric Pressure, SBC, Lambda > 1



Output 120.3.3 *continued*



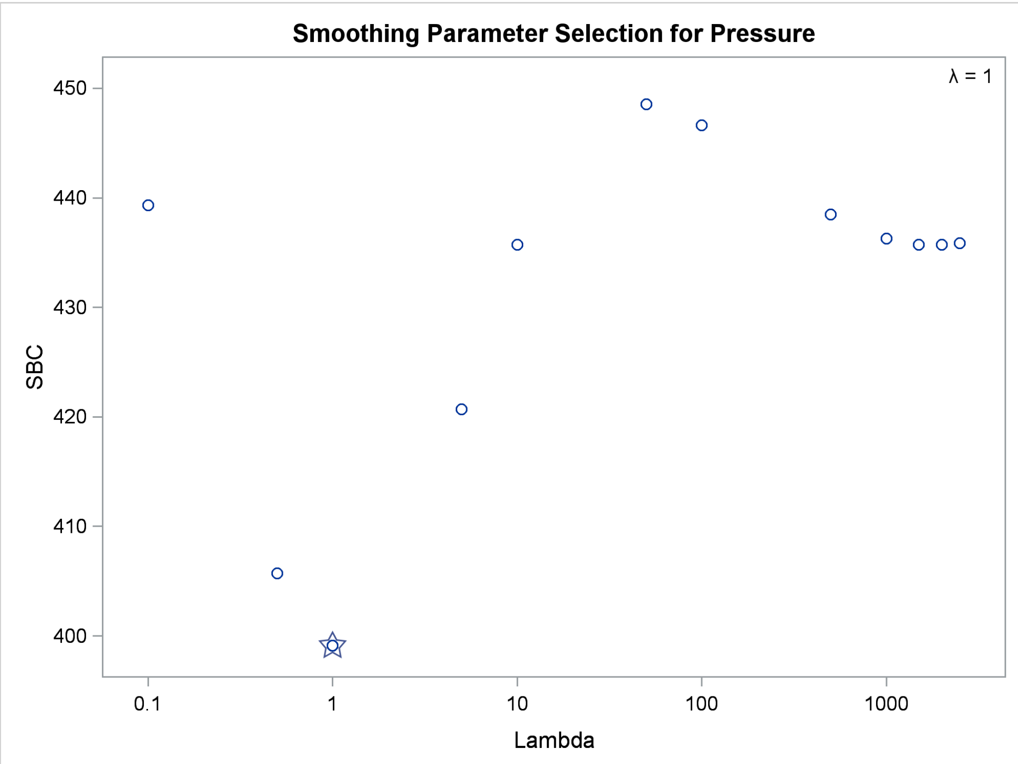
The results clearly show that there is a local minimum in the SBC(λ) function at $\lambda = 1801.1$. Using this lambda results in a much smoother regression function with a longer cycle than you saw previously. This second cycle can be identified as the periodic warming of the Pacific Ocean known as "El Niño." The SBC(λ) function has at least two minima since there are at least two trends in the data. In the first analysis, PROC TRANSREG found what is probably the globally optimal solution, and in the second set of analyses, with a little nudging away from the global optimum, it found a very interesting locally optimal solution.

You can specify a list of lambdas to see SBC as a function of lambda over the range that includes both minima as follows:

```
proc transreg data=sashelp.enso;
   model identity(pressure) = pbspline(year / sbc lambda=.1 .5 1 5
                                    10 50 100 500 to 2500 by 500);
run;
```

The plot of SBC as a function of λ is shown in Output 120.3.4.

Output 120.3.4 Change in Atmospheric Pressure, SBC, Over the Range of Both Minima



Example 120.4: Nonmetric Conjoint Analysis of Tire Data

This example uses PROC TRANSREG to perform a nonmetric conjoint analysis of tire preference data. Conjoint analysis decomposes rank-ordered evaluation judgments of products or services into components based on qualitative product attributes. For each level of each attribute of interest, a numerical "part-worth utility" value is computed. The sum of the part-worth utilities for each product is an estimate of the utility for that product. The goal is to compute part-worth utilities such that the product utilities are as similar as possible to the original rank ordering. (This example is a greatly simplified introductory example.)

The stimuli for the experiment are 18 hypothetical tires. The stimuli represent different brands (Goodstone, Pirogi, Machismo),[2] prices ($69.99, $74.99, $79.99), expected tread life (50,000, 60,000, 70,000 miles), and road hazard insurance plans (Yes, No). There are $3 \times 3 \times 3 \times 2 = 54$ possible combinations. From these, 18 combinations are selected that form an efficient experimental design for a main-effects model. The combinations are then ranked from 1 (most preferred) to 18 (least preferred). In this simple example, there is one set of rankings. A real conjoint study would have many more.

First, the FORMAT procedure is used to specify the meanings of the factor levels, which are entered as numbers in the DATA step along with the ranks. PROC TRANSREG is used to perform the conjoint analysis. A maximum of 50 iterations is requested. The specification **monotone(Rank / reflect)** in the MODEL statement requests that the dependent variable Rank should be monotonically transformed and reflected so that positive utilities mean high preference. The variables Brand, Price, Life, and Hazard are designated as CLASS variables, and the part-worth utilities are constrained by ZERO=SUM to sum to zero within each factor. The UTILITIES *a-option* displays the conjoint analysis results.

The importance column of the utilities table shows that price is the most important attribute in determining preference (57%), followed by expected tread life (18%), brand (15%), and road hazard insurance (10%). Looking at the utilities table for the maximum part-worth utility within each attribute, you see from the results that the most preferred combination is Pirogi brand tires, at $69.99, with a 70,000-mile expected tread life and road hazard insurance. This product is not actually in the data set. The sum of the part-worth utilities for this combination is as follows:

$$20.64 = 9.50 + 1.90 + 5.87 + 2.41 + 0.96$$

The following statements produce Output 120.4.1.

```
title 'Nonmetric Conjoint Analysis of Ranks';

proc format;
   value BrandF
              1 = 'Goodstone'
              2 = 'Pirogi   '
              3 = 'Machismo ';
   value PriceF
              1 = '$69.99'
              2 = '$74.99'
              3 = '$79.99';
   value LifeF
              1 = '50,000'
              2 = '60,000'
              3 = '70,000';
   value HazardF
              1 = 'Yes'
              2 = 'No ';
run;
```

[2] In real conjoint experiments, real brand names would be used.

```
data Tires;
   input Brand Price Life Hazard Rank;
   format Brand BrandF9. Price PriceF9. Life LifeF6. Hazard HazardF3.;
   datalines;
1 1 2 1  3
1 1 3 2  2
1 2 1 2 14
1 2 2 2 10
1 3 1 1 17
1 3 3 1 12
2 1 1 2  7
2 1 3 2  1
2 2 1 1  8
2 2 3 1  5
2 3 2 1 13
2 3 2 2 16
3 1 1 1  6
3 1 2 1  4
3 2 2 2 15
3 2 3 1  9
3 3 1 2 18
3 3 3 2 11
;

proc transreg maxiter=50 utilities short;
   ods select TestsNote ConvergenceStatus FitStatistics Utilities;
   model monotone(Rank / reflect) =
         class(Brand Price Life Hazard / zero=sum);
   output ireplace predicted;
run;

proc print label;
   var Rank TRank PRank Brand Price Life Hazard;
   label PRank = 'Predicted Ranks';
run;
```

Output 120.4.1 Simple Conjoint Analysis

Nonmetric Conjoint Analysis of Ranks

The TRANSREG Procedure

Monotone(Rank)
Algorithm converged.

The TRANSREG Procedure Hypothesis Tests for Monotone(Rank)

Root MSE	0.49759	**R-Square**	0.9949
Dependent Mean	9.50000	**Adj R-Sq**	0.9913
Coeff Var	5.23783		

Output 120.4.1 *continued*

Utilities Table Based on the Usual Degrees of Freedom

Label	Utility	Standard Error	Importance (% Utility Range)	Variable
Intercept	9.5000	0.11728		Intercept
Brand Goodstone	-1.1718	0.16586	15.463	Class.BrandGoodstone
Brand Pirogi	1.8980	0.16586		Class.BrandPirogi
Brand Machismo	-0.7262	0.16586		Class.BrandMachismo
Price $69.99	5.8732	0.16586	56.517	Class.Price_69_99
Price $74.99	-0.5261	0.16586		Class.Price_74_99
Price $79.99	-5.3471	0.16586		Class.Price_79_99
Life 50,000	-1.2350	0.16586	18.361	Class.Life50_000
Life 60,000	-1.1751	0.16586		Class.Life60_000
Life 70,000	2.4101	0.16586		Class.Life70_000
Hazard Yes	0.9588	0.11728	9.659	Class.HazardYes
Hazard No	-0.9588	0.11728		Class.HazardNo

The standard errors are not adjusted for the fact that the dependent variable was transformed and so are generally liberal (too small).

Output 120.4.1 *continued*

Nonmetric Conjoint Analysis of Ranks

Obs	Rank	Rank Transformation	Predicted Ranks	Brand	Price	Life	Hazard
1	3	14.4462	13.9851	Goodstone	$69.99	60,000	Yes
2	2	15.6844	15.6527	Goodstone	$69.99	70,000	No
3	14	5.7229	5.6083	Goodstone	$74.99	50,000	No
4	10	5.7229	5.6682	Goodstone	$74.99	60,000	No
5	17	2.6699	2.7049	Goodstone	$79.99	50,000	Yes
6	12	5.7229	6.3500	Goodstone	$79.99	70,000	Yes
7	7	14.4462	15.0774	Pirogi	$69.99	50,000	No
8	1	18.7699	18.7225	Pirogi	$69.99	70,000	No
9	8	11.1143	10.5957	Pirogi	$74.99	50,000	Yes
10	5	14.4462	14.2408	Pirogi	$74.99	70,000	Yes
11	13	5.7229	5.8346	Pirogi	$79.99	60,000	Yes
12	16	3.8884	3.9170	Pirogi	$79.99	60,000	No
13	6	14.4462	14.3708	Machismo	$69.99	50,000	Yes
14	4	14.4462	14.4307	Machismo	$69.99	60,000	Yes
15	15	5.7229	6.1139	Machismo	$74.99	60,000	No
16	9	11.1143	11.6166	Machismo	$74.99	70,000	Yes
17	18	1.1905	1.2330	Machismo	$79.99	50,000	No
18	11	5.7229	4.8780	Machismo	$79.99	70,000	No

Example 120.5: Metric Conjoint Analysis of Tire Data

This example, which is more detailed than the previous one, uses PROC TRANSREG to perform a metric conjoint analysis of tire preference data. Conjoint analysis can be used to decompose preference ratings of products or services into components based on qualitative product attributes. For each level of each attribute of interest, a numerical "part-worth utility" value is computed. The sum of the part-worth utilities for each product is an estimate of the utility for that product. The goal is to compute part-worth utilities such that the product utilities are as similar as possible to the original ratings. Metric conjoint analysis, as shown in this example, fits an ordinary linear model directly to data assumed to be measured on an interval scale. Nonmetric conjoint analysis, as shown in Example 120.4, finds an optimal monotonic transformation of original data before fitting an ordinary linear model to the transformed data.

This example has three parts. In the first part, an experimental design is created. In the second part, a DATA step creates descriptions of the stimuli for the experiment. The third part of the example performs the conjoint analyses.

The stimuli for the experiment are 18 hypothetical tires. The stimuli represent different brands (Goodstone, Pirogi, Machismo),[3] prices ($69.99, $74.99, $79.99), expected tread life (50,000, 60,000, 70,000 miles), and road hazard insurance plans (Yes, No).

For a conjoint study such as this, you need to create an experimental design with 3 three-level factors, 1 two-level factor, and 18 combinations or *runs*. The easiest way to get this design is with the %MktEx autocall macro. The %MktEx macro requires you to specify the number of levels of each of the four factors, followed by N=18, the number of runs. Specifying a random number seed, while not strictly necessary, helps ensure that the design is reproducible. The %MktLab macro assigns the actual factor names instead of the default names `x1`, `x2`, and so on, and it assigns formats to the factor levels. The %MktEval macro helps you evaluate the design. It shows how correlated or independent the factors are, how often each factor level appears in the design, how often each pair occurs for every factor pair, and how often each product profile or run occurs in the design. See Kuhfeld (2010) for more information about experimental design and conjoint analysis; look for the latest "Conjoint Analysis" report. The following statements create, evaluate, and display the design:

```
title 'Tire Study, Experimental Design';

proc format;
   value BrandF
              1 = 'Goodstone'
              2 = 'Pirogi   '
              3 = 'Machismo ';
   value PriceF
              1 = '$69.99'
              2 = '$74.99'
              3 = '$79.99';
   value LifeF
              1 = '50,000'
              2 = '60,000'
              3 = '70,000';
   value HazardF
              1 = 'Yes'
              2 = 'No ';
```

[3] In real conjoint experiments, real brand names would be used.

```
run;

%mktex(3 3 3 2, n=18, seed=448)

%mktlab(vars=Brand Price Life Hazard, out=sasuser.TireDesign,
        statements=format Brand BrandF9. Price PriceF9.
                   Life LifeF6. Hazard HazardF3.)

%mkteval;

proc print data=sasuser.TireDesign;
run;
```

The %MktEx macro (Kuhfeld 2010) output displayed in Output 120.5.1 shows you that the design is 100% efficient, which means it is orthogonal and balanced. The %MktEval macro output displayed in Output 120.5.2 shows you that all of the factors are uncorrelated or orthogonal, the design is balanced (each level occurs once), and every pair of factor levels occurs equally often (again showing that the design is orthogonal). The *n*-way frequencies show that each product profile occurs once (there are no duplicates). The design is shown in Output 120.5.3. The design is automatically randomized (the profiles were sorted into a random order and the original levels are randomly reassigned). Orthogonality, balance, randomization, and other design concepts are discussed in detail in Kuhfeld (2010), in the "Experimental Design, Efficiency, Coding, and Choice Designs" report.

Output 120.5.1 Tire Study, Design Efficiency

Tire Study, Experimental Design

Algorithm Search History

Design	Row,Col	Current D-Efficiency	Best D-Efficiency	Notes
1	Start	100.0000	100.0000	Tab
1	End	100.0000		

Tire Study, Experimental Design

The OPTEX Procedure

Class Level Information

Class	Levels	Values
x1	3	1 2 3
x2	3	1 2 3
x3	3	1 2 3
x4	2	1 2

Output 120.5.1 *continued*

Tire Study, Experimental Design

Design Number	D-Efficiency	A-Efficiency	G-Efficiency	Average Prediction Standard Error
1	100.0000	100.0000	100.0000	0.6667

Output 120.5.2 Tire Study, Design Evaluation

Tire Study, Experimental Design
Canonical Correlations Between the Factors
There are 0 Canonical Correlations Greater Than 0.316

	Brand	Price	Life	Hazard
Brand	1	0	0	0
Price	0	1	0	0
Life	0	0	1	0
Hazard	0	0	0	1

Tire Study, Experimental Design
Summary of Frequencies
There are 0 Canonical Correlations Greater Than 0.316

	Frequencies
Brand	6 6 6
Price	6 6 6
Life	6 6 6
Hazard	9 9
Brand Price	2 2 2 2 2 2 2 2 2
Brand Life	2 2 2 2 2 2 2 2 2
Brand Hazard	3 3 3 3 3 3
Price Life	2 2 2 2 2 2 2 2 2
Price Hazard	3 3 3 3 3 3
Life Hazard	3 3 3 3 3 3
N-Way	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Output 120.5.3 Tire Study, Design

Tire Study, Experimental Design

Obs	Brand	Price	Life	Hazard
1	Pirogi	$79.99	50,000	No
2	Machismo	$79.99	60,000	No
3	Machismo	$74.99	70,000	Yes
4	Machismo	$74.99	50,000	No
5	Goodstone	$74.99	60,000	Yes
6	Pirogi	$69.99	60,000	Yes
7	Goodstone	$69.99	50,000	Yes
8	Machismo	$69.99	50,000	Yes
9	Pirogi	$74.99	60,000	Yes
10	Pirogi	$74.99	50,000	No
11	Goodstone	$79.99	60,000	No
12	Goodstone	$69.99	70,000	No
13	Pirogi	$79.99	70,000	Yes
14	Goodstone	$74.99	70,000	No
15	Machismo	$69.99	60,000	No
16	Machismo	$79.99	70,000	Yes
17	Pirogi	$69.99	70,000	No
18	Goodstone	$79.99	50,000	Yes

The %MktEx macro requires SAS/STAT, SAS/QC, and SAS/IML software. Alternatively, you can make a design for this experiment using the %MktDes macro, which requires only SAS/STAT and SAS/QC software. The %MktDes macro contains a small subset of the functionality of the %MktEx macro. It can be used as follows:

```
%mktdes(factors=Brand=3 Price=3 Life=3 Hazard=2, n=18)
```

The results of this step are not shown or used.

Next, the questionnaires are printed and given to the subjects, who are asked to rate the tires.

The following statements produce Output 120.5.4:

```
data _null_;
   title;
   set sasuser.TireDesign;
   file print;
   if mod(_n_,4) eq 1 then do;
      put _page_;
      put +55 'Subject ________';
   end;
   length hazardstring $ 7.;
   if put(hazard, hazardf3.) = 'Yes'
      then hazardstring = 'with';
      else hazardstring = 'without';

   s = 3 + (_n_ >= 10);
   put // _n_ +(-1) ') For your next tire purchase, '
          'how likely are you to buy this product?'
       // +s Brand 'brand tires at ' Price +(-1) ','
       /  +s 'with a ' Life 'tread life guarantee, '
       /  +s 'and ' hazardstring 'road hazard insurance.'
       // +s 'Definitely Would                 Definitely Would'
       /  +s 'Not Purchase                             Purchase'
       // +s '1     2     3     4     5     6     7     8     9 ';
run;
```

This output in Output 120.5.4 is abbreviated in the interest of conserving space; the statements actually produce stimuli for all combinations.

Output 120.5.4 Conjoint Analysis, Stimuli Descriptions

```
                                                          Subject ________


1) For your next tire purchase, how likely are you to buy this product?

   Pirogi brand tires at $79.99,
   with a 50,000 tread life guarantee,
   and without road hazard insurance.

   Definitely Would                 Definitely Would
   Not Purchase                             Purchase

   1      2      3      4      5      6      7      8      9


2) For your next tire purchase, how likely are you to buy this product?

   Machismo brand tires at $79.99,
   with a 60,000 tread life guarantee,
   and without road hazard insurance.

   Definitely Would                 Definitely Would
   Not Purchase                             Purchase

   1      2      3      4      5      6      7      8      9


3) For your next tire purchase, how likely are you to buy this product?

   Machismo brand tires at $74.99,
   with a 70,000 tread life guarantee,
   and with road hazard insurance.

   Definitely Would                 Definitely Would
   Not Purchase                             Purchase

   1      2      3      4      5      6      7      8      9


4) For your next tire purchase, how likely are you to buy this product?

   Machismo brand tires at $74.99,
   with a 50,000 tread life guarantee,
   and without road hazard insurance.

   Definitely Would                 Definitely Would
   Not Purchase                             Purchase

   1      2      3      4      5      6      7      8      9
```

The third part of the example performs the conjoint analyses. The DATA step reads the data. Only the ratings are entered, one row per subject. Real conjoint studies have many more subjects than five. The TRANSPOSE procedure transposes this (5×18) data set into an (18×5) data set that can be merged with the factor level data set sasuser.TireDesign. The next DATA step does the merge. The PRINT procedure displays the input data set.

PROC TRANSREG fits the five individual conjoint models, one for each subject. The UTILITIES *a-option* displays the conjoint analysis results. The SHORT *a-option* suppresses the iteration histories, OUTTEST=UTILS creates an output data set with all of the conjoint results, and the SEPARATORS= option requests that the labels constructed for each category contain two blanks between the variable name and the level value. The ODS SELECT statement is used to limit the displayed output. The MODEL statement specifies IDENTITY for the ratings, which specifies a metric conjoint analysis—the ratings are not transformed. The variables Brand, Price, Life, and Hazard are designated as CLASS variables, and the part-worth utilities are constrained to sum to zero within each factor.

The following statements produce Output 120.5.5:

```
title 'Tire Study, Data Entry, Preprocessing';

data Results;
   input (c1-c18) (1.);
   datalines;
233279766526376493
124467885349168274
262189456534275794
184396375364187754
133379775526267493
;

* Create an Object by Subject Data Matrix;
proc transpose data=Results out=Results(drop=_name_) prefix=Subj;
run;

* Merge the Factor Levels with the Data Matrix;
data Both;
   merge sasuser.TireDesign Results;
run;

proc print;
   title2 'Data Set for Conjoint Analysis';
run;

title 'Tire Study, Individual Conjoint Analyses';

* Fit Each Subject Individually;
proc transreg data=Both utilities short outtest=utils separators='  ';
   ods select TestsNote FitStatistics Utilities;
   model identity(Subj1-Subj5) =
         class(Brand Price Life Hazard / zero=sum);
run;
```

The output contains two tables per subject, one with overall fit statistics and one with the conjoint analysis results.

Output 120.5.5 Conjoint Analysis

Tire Study, Data Entry, Preprocessing
Data Set for Conjoint Analysis

Obs	Brand	Price	Life	Hazard	Subj1	Subj2	Subj3	Subj4	Subj5
1	Pirogi	$79.99	50,000	No	2	1	2	1	1
2	Machismo	$79.99	60,000	No	3	2	6	8	3
3	Machismo	$74.99	70,000	Yes	3	4	2	4	3
4	Machismo	$74.99	50,000	No	2	4	1	3	3
5	Goodstone	$74.99	60,000	Yes	7	6	8	9	7
6	Pirogi	$69.99	60,000	Yes	9	7	9	6	9
7	Goodstone	$69.99	50,000	Yes	7	8	4	3	7
8	Machismo	$69.99	50,000	Yes	6	8	5	7	7
9	Pirogi	$74.99	60,000	Yes	6	5	6	5	5
10	Pirogi	$74.99	50,000	No	5	3	5	3	5
11	Goodstone	$79.99	60,000	No	2	4	3	6	2
12	Goodstone	$69.99	70,000	No	6	9	4	4	6
13	Pirogi	$79.99	70,000	Yes	3	1	2	1	2
14	Goodstone	$74.99	70,000	No	7	6	7	8	6
15	Machismo	$69.99	60,000	No	6	8	5	7	7
16	Machismo	$79.99	70,000	Yes	4	2	7	7	4
17	Pirogi	$69.99	70,000	No	9	7	9	5	9
18	Goodstone	$79.99	50,000	Yes	3	4	4	4	3

Output 120.5.5 *continued*

Tire Study, Individual Conjoint Analyses

The TRANSREG Procedure

The TRANSREG Procedure Hypothesis Tests for Identity(Subj1)

Root MSE	1.34164	**R-Square**	0.8043
Dependent Mean	5.00000	**Adj R-Sq**	0.6674
Coeff Var	26.83282		

Output 120.5.5 *continued*

Utilities Table Based on the Usual Degrees of Freedom				
Label	**Utility**	**Standard Error**	**Importance (% Utility Range)**	**Variable**
Intercept	5.0000	0.31623		Intercept
Brand Goodstone	0.3333	0.44721	20.833	Class.BrandGoodstone
Brand Pirogi	0.6667	0.44721		Class.BrandPirogi
Brand Machismo	-1.0000	0.44721		Class.BrandMachismo
Price $69.99	2.1667	0.44721	54.167	Class.Price_69_99
Price $74.99	0.0000	0.44721		Class.Price_74_99
Price $79.99	-2.1667	0.44721		Class.Price_79_99
Life 50,000	-0.8333	0.44721	16.667	Class.Life50_000
Life 60,000	0.5000	0.44721		Class.Life60_000
Life 70,000	0.3333	0.44721		Class.Life70_000
Hazard Yes	0.3333	0.31623	8.333	Class.HazardYes
Hazard No	-0.3333	0.31623		Class.HazardNo

Tire Study, Individual Conjoint Analyses

The TRANSREG Procedure

The TRANSREG Procedure Hypothesis Tests for Identity(Subj2)

Root MSE	0.56765	**R-Square**	0.9710
Dependent Mean	4.94444	**Adj R-Sq**	0.9506
Coeff Var	11.48049		

Utilities Table Based on the Usual Degrees of Freedom				
Label	**Utility**	**Standard Error**	**Importance (% Utility Range)**	**Variable**
Intercept	4.9444	0.13380		Intercept
Brand Goodstone	1.2222	0.18922	25.658	Class.BrandGoodstone
Brand Pirogi	-0.9444	0.18922		Class.BrandPirogi
Brand Machismo	-0.2778	0.18922		Class.BrandMachismo
Price $69.99	2.8889	0.18922	65.132	Class.Price_69_99
Price $74.99	-0.2778	0.18922		Class.Price_74_99
Price $79.99	-2.6111	0.18922		Class.Price_79_99
Life 50,000	-0.2778	0.18922	7.895	Class.Life50_000
Life 60,000	0.3889	0.18922		Class.Life60_000
Life 70,000	-0.1111	0.18922		Class.Life70_000
Hazard Yes	0.0556	0.13380	1.316	Class.HazardYes
Hazard No	-0.0556	0.13380		Class.HazardNo

Output 120.5.5 *continued*

Tire Study, Individual Conjoint Analyses

The TRANSREG Procedure

The TRANSREG Procedure Hypothesis Tests for Identity(Subj3)

Root MSE	2.48104	**R-Square**	0.3902
Dependent Mean	4.94444	**Adj R-Sq**	-0.0367
Coeff Var	50.17832		

Utilities Table Based on the Usual Degrees of Freedom

Label	Utility	Standard Error	Importance (% Utility Range)	Variable
Intercept	4.9444	0.58479		Intercept
Brand Goodstone	0.0556	0.82701	18.261	Class.BrandGoodstone
Brand Pirogi	0.5556	0.82701		Class.BrandPirogi
Brand Machismo	-0.6111	0.82701		Class.BrandMachismo
Price $69.99	1.0556	0.82701	31.304	Class.Price_69_99
Price $74.99	-0.1111	0.82701		Class.Price_74_99
Price $79.99	-0.9444	0.82701		Class.Price_79_99
Life 50,000	-1.4444	0.82701	41.739	Class.Life50_000
Life 60,000	1.2222	0.82701		Class.Life60_000
Life 70,000	0.2222	0.82701		Class.Life70_000
Hazard Yes	0.2778	0.58479	8.696	Class.HazardYes
Hazard No	-0.2778	0.58479		Class.HazardNo

Tire Study, Individual Conjoint Analyses

The TRANSREG Procedure

The TRANSREG Procedure Hypothesis Tests for Identity(Subj4)

Root MSE	1.90321	**R-Square**	0.6185
Dependent Mean	5.05556	**Adj R-Sq**	0.3514
Coeff Var	37.64598		

Output 120.5.5 *continued*

Utilities Table Based on the Usual Degrees of Freedom				
Label	**Utility**	**Standard Error**	**Importance (% Utility Range)**	**Variable**
Intercept	5.0556	0.44859		Intercept
Brand Goodstone	0.6111	0.63440	36.885	Class.BrandGoodstone
Brand Pirogi	-1.5556	0.63440		Class.BrandPirogi
Brand Machismo	0.9444	0.63440		Class.BrandMachismo
Price $69.99	0.2778	0.63440	12.295	Class.Price_69_99
Price $74.99	0.2778	0.63440		Class.Price_74_99
Price $79.99	-0.5556	0.63440		Class.Price_79_99
Life 50,000	-1.5556	0.63440	49.180	Class.Life50_000
Life 60,000	1.7778	0.63440		Class.Life60_000
Life 70,000	-0.2222	0.63440		Class.Life70_000
Hazard Yes	0.0556	0.44859	1.639	Class.HazardYes
Hazard No	-0.0556	0.44859		Class.HazardNo

Tire Study, Individual Conjoint Analyses

The TRANSREG Procedure

The TRANSREG Procedure Hypothesis Tests for Identity(Subj5)

Root MSE	1.36219	**R-Square**	0.8162
Dependent Mean	4.94444	**Adj R-Sq**	0.6875
Coeff Var	27.54987		

Utilities Table Based on the Usual Degrees of Freedom				
Label	**Utility**	**Standard Error**	**Importance (% Utility Range)**	**Variable**
Intercept	4.9444	0.32107		Intercept
Brand Goodstone	0.2222	0.45406	9.023	Class.BrandGoodstone
Brand Pirogi	0.2222	0.45406		Class.BrandPirogi
Brand Machismo	-0.4444	0.45406		Class.BrandMachismo
Price $69.99	2.5556	0.45406	67.669	Class.Price_69_99
Price $74.99	-0.1111	0.45406		Class.Price_74_99
Price $79.99	-2.4444	0.45406		Class.Price_79_99
Life 50,000	-0.6111	0.45406	15.789	Class.Life50_000
Life 60,000	0.5556	0.45406		Class.Life60_000
Life 70,000	0.0556	0.45406		Class.Life70_000
Hazard Yes	0.2778	0.32107	7.519	Class.HazardYes
Hazard No	-0.2778	0.32107		Class.HazardNo

The next steps summarize the results. Three tables are displayed, showing the following: all of the importance values, the average importance, and the part-worth utilities. The first DATA step selects the importance information from the UTILS data set. The final assignment statement stores just the variable name from the label, relying on the fact that the separator is two blanks. PROC TRANSPOSE creates the data set of importances, one row per subject, and PROC PRINT displays the results. The MEANS procedure displays the average importance of each attribute across the subjects. The next DATA step selects the part-worth utilities information from the UTILS data set. PROC TRANSPOSE creates the data set of utilities, one row per subject, and PROC PRINT displays the results. The following statements produce Output 120.5.6:

```
title 'Tire Study Results';

* Gather the Importance Values;
data Importance;
   set utils(keep=_depvar_ Importance Label);
   if n(Importance);
   label = substr(label, 1, index(label, '  '));
run;

proc transpose out=Importance2(drop=_:);
   by _depvar_;
   id Label;
run;

proc print;
   title2 'Importance Values';
run;

proc means;
   title2 'Average Importance';
run;

* Gather the Part-Worth Utilities;
data Utilities;
   set utils(keep=_depvar_ Coefficient Label);
   if n(Coefficient);
run;

proc transpose out=Utilities2(drop=_:);
   by _depvar_;
   id Label;
   idlabel Label;
run;

proc print label;
   title2 'Utilities';
run;
```

Output 120.5.6 Summary of Conjoint Analysis Results

Tire Study Results
Importance Values

Obs	Brand	Price	Life	Hazard
1	20.8333	54.1667	16.6667	8.33333
2	25.6579	65.1316	7.8947	1.31579
3	18.2609	31.3043	41.7391	8.69565
4	36.8852	12.2951	49.1803	1.63934
5	9.0226	67.6692	15.7895	7.51880

Output 120.5.6 *continued*

Tire Study Results
Average Importance

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
Brand	5	22.1319800	10.2301014	9.0225564	36.8852459
Price	5	46.1133697	23.7391251	12.2950820	67.6691729
Life	5	26.2540671	18.0547195	7.8947368	49.1803279
Hazard	5	5.5005832	3.6989117	1.3157895	8.6956522

Output 120.5.6 *continued*

Tire Study Results
Utilities

Obs	Intercept	Brand Goodstone	Brand Pirogi	Brand Machismo	Price $69.99	Price $74.99	Price $79.99	Life 50,000	Life 60,000	Life 70,000	Hazard Yes	Hazard No
1	5.00000	0.33333	0.66667	-1.00000	2.16667	0.00000	-2.16667	-0.83333	0.50000	0.33333	0.33333	-0.33333
2	4.94444	1.22222	-0.94444	-0.27778	2.88889	-0.27778	-2.61111	-0.27778	0.38889	-0.11111	0.05556	-0.05556
3	4.94444	0.05556	0.55556	-0.61111	1.05556	-0.11111	-0.94444	-1.44444	1.22222	0.22222	0.27778	-0.27778
4	5.05556	0.61111	-1.55556	0.94444	0.27778	0.27778	-0.55556	-1.55556	1.77778	-0.22222	0.05556	-0.05556
5	4.94444	0.22222	0.22222	-0.44444	2.55556	-0.11111	-2.44444	-0.61111	0.55556	0.05556	0.27778	-0.27778

Based on the importance values, price is the most important attribute for some of the respondents, but expected tread life is most important for others. On the average, price is most important, followed by expected tread life and brand. Road hazard insurance is less important. Each of the brands is preferred by some of the respondents. All respondents preferred a lower price over a higher price, a longer tread life, and road hazard insurance.

Example 120.6: Preference Mapping of Automobile Data

This example uses PROC TRANSREG to perform a preference mapping (PREFMAP) analysis (Carroll 1972) of automobile preference data after a PROC PRINQUAL principal component analysis. The PREFMAP analysis is a response surface regression that locates ideal points for each dependent variable in a space defined by the independent variables.

The data are ratings obtained from 25 judges of their preference for each of 17 automobiles. The ratings were made on a scale of zero (very weak preference) to nine (very strong preference). These judgments were made in 1980 about that year's products. There are two character variables that indicate the manufacturer and model of the automobile. The data set also contains three ratings: miles per gallon (MPG), projected reliability (Reliability), and quality of the ride (Ride). These ratings are on a scale of one (bad) to five (good). PROC PRINQUAL creates an OUT= data set containing standardized principal component scores (Prin1 and Prin2), along with the ID variables Model, MPG, Reliability, and Ride.

While this data set contains all of the information needed for the subsequent preference mapping, you can make slightly more informative plots by adding new variable labels to the principal component score variables. The default labels are 'Component 1', 'Component 2', and so on. These are by necessity rather generic since they are created before any data are read, and they must be appropriate across BY groups when a BY variable is specified. In contrast, the MDPREF plot in PROC PRINQUAL has axis labels of the form 'Component 1 (43.54%)' and 'Component 2 (23.4%)' that show the proportion of variance accounted for by each component. You can create an output data set from the MDPREF plot by using the ODS OUTPUT statement and then use only the label information from it to reset the labels in the output data set from PROC PRINQUAL. In the DATA PLOT step, the SET statement for the MD data set is specified before the SET statement for the PRESULTS data set. The `if 0` ensures that no data are actually read from it, but nevertheless the properties of the Prin1 and Prin2 variables including the variable labels are set based on the properties of those variables in the MD data set.

The first PROC TRANSREG step fits univariate regression models for MPG and Reliability. All variables are designated IDENTITY. A vector drawn in the plot of Prin1 and Prin2 from the origin to the point defined by an attribute's regression coefficients approximately shows how the autos differ on that attribute. See Carroll (1972) for more information. The Prin1 and Prin2 columns of the TResult1 OUT= data set contain the automobile coordinates (_Type_='SCORE' observations) and endpoints of the MPG and Reliability vectors (_Type_='M COEFFI' observations).

The second PROC TRANSREG step fits a univariate regression model with Ride designated IDENTITY, and Prin1 and Prin2 designated POINT. The POINT expansion creates an additional independent variable _ISSQ_, which contains the sum of Prin1 squared and Prin2 squared. The OUT= data set TResult2 contains no _Type_='SCORE' observations, only ideal point (_Type_='M POINT') coordinates for Ride. The coordinates of both the vectors and the ideal points are output by specifying COORDINATES in the OUTPUT statement in PROC TRANSREG.

A vector model is used for MPG and Reliability because perfectly efficient and reliable automobiles do not exist in the data set. The ideal points for MPG and Reliability are far removed from the plot of the automobiles. It is more likely that an ideal point for quality of the ride is in the plot, so an ideal point model is used for the ride variable. See Carroll (1972) and Schiffman, Reynolds, and Young (1981) for discussions of the vector model and point models (including the EPOINT and QPOINT versions of the point model that are not used in this example). For the vector model, the default coordinates stretch factor of 2.5 was used. This extends the vectors by a factor of 2.5 from their standard lengths, making a better graphical display. Sometimes the default vectors are short and near the origin, and they look better when they are extended.

The following statements produce Output 120.6.1 through Output 120.6.5:

```
title 'Preference Ratings for Automobiles Manufactured in 1980';

options validvarname=any;

data CarPreferences;
   input Make $ 1-10 Model $ 12-22 @25 ('1'n-'25'n) (1.)
         MPG Reliability Ride;
   datalines;
Cadillac   Eldorado     8007990491240508971093809 3 2 4
Chevrolet  Chevette     0051200423451043003515698 5 3 2
Chevrolet  Citation     4053305814161643544747795 4 1 5
Chevrolet  Malibu       6027400723121345545668658 3 3 4
Ford       Fairmont     2024006715021443530648655 3 3 4
Ford       Mustang      5007197705021101850657555 3 2 2
Ford       Pinto        0021000303030201500514078 4 1 1
Honda      Accord       5956897609699952998975078 5 5 3
Honda      Civic        4836709507488852567765075 5 5 3
```

```
Lincoln    Continental  7008990592230409962091909 2 4 5
Plymouth   Gran Fury    7006000434101107333458708 2 1 5
Plymouth   Horizon      3005005635461302444675655 4 3 3
Plymouth   Volare       4005003614021602754476555 2 1 3
Pontiac    Firebird     0107895613201206958265907 1 1 5
Volkswagen Dasher       4858696508877795377895000 5 3 4
Volkswagen Rabbit       4858509709695795487885000 5 4 3
Volvo      DL           9989998909999987989919000 4 5 5
;

ods graphics on;

* Compute Coordinates for a 2-Dimensional Scatter Plot of Automobiles;
proc prinqual data=CarPreferences out=PResults(drop='1'n-'25'n)
              n=2 replace standard scores mdpref=2;
   id Model MPG Reliability Ride;
   transform identity('1'n-'25'n);
   title2 'Multidimensional Preference (MDPREF) Analysis';
   ods output mdprefplot=md;
run;

options validvarname=v7;

title2 'Preference Mapping (PREFMAP) Analysis';

* Add the Labels from the Plot to the Results Data Set;
data plot;
   if 0 then set md(keep=prin:);
   set presults;
run;

* Compute Endpoints for MPG and Reliability Vectors;
proc transreg data=plot rsquare;
   Model identity(MPG Reliability)=identity(Prin1 Prin2);
   output tstandard=center coordinates replace out=TResult1;
   id Model;
run;

* Compute Ride Ideal Point Coordinates;
proc transreg data=plot rsquare;
   Model identity(Ride)=point(Prin1 Prin2);
   output tstandard=center coordinates replace noscores out=TResult2;
   id Model;
run;

proc print;
run;
```

Output 120.6.1 Preference Ratings Example Output

Preference Ratings for Automobiles Manufactured in 1980
Multidimensional Preference (MDPREF) Analysis

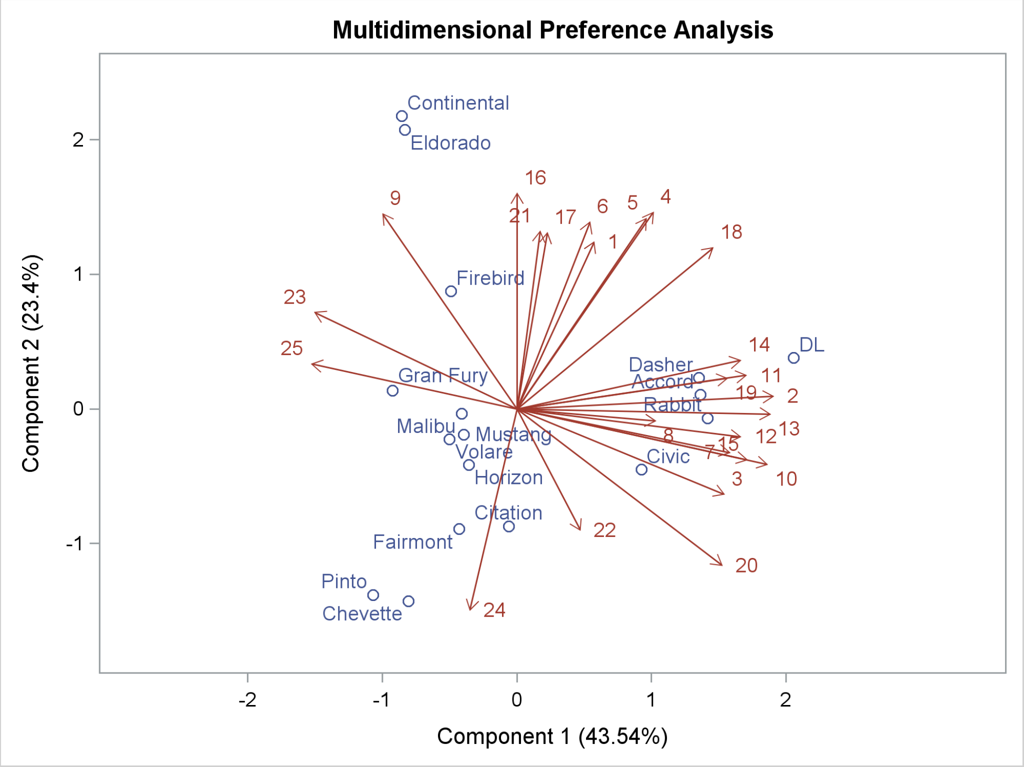
The PRINQUAL Procedure

PRINQUAL MTV Algorithm Iteration History

Iteration Number	Average Change	Maximum Change	Proportion of Variance	Criterion Change	Note
1	0.00000	0.00000	0.66946		Converged

Algorithm converged.

Output 120.6.2 MDPREF Plot



Output 120.6.3 shows that an unreliable-to-reliable direction extends from the left and slightly below the origin to the right and slightly above the origin. The Japanese and European automobiles are rated, on the average, as more reliable. A low MPG to good MPG direction extends from the top left of the plot to the bottom right. The smaller automobiles, on the average, get better gas mileage.

Output 120.6.3 Preference Mapping Vector Plot

Preference Ratings for Automobiles Manufactured in 1980
Preference Mapping (PREFMAP) Analysis

The TRANSREG Procedure

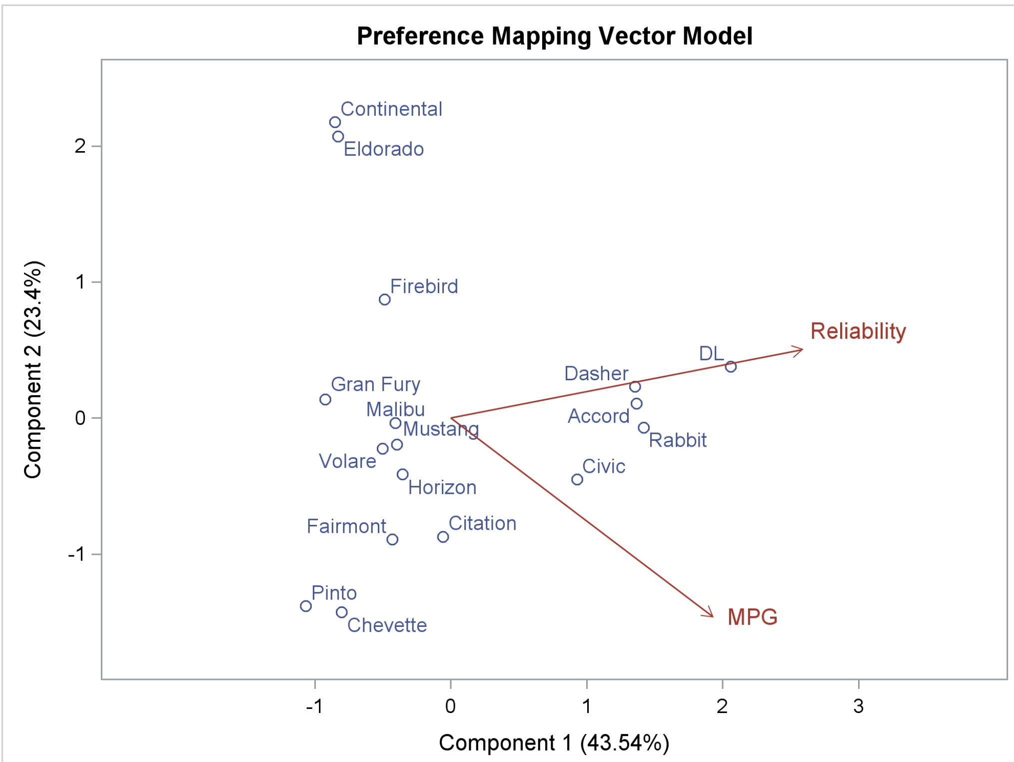
The TRANSREG Procedure Hypothesis Tests for Identity(MPG)

R-Square	0.5720

The TRANSREG Procedure Hypothesis Tests for Identity(Reliability)

R-Square	0.5086

Output 120.6.3 *continued*



The ideal point for Ride in Output 120.6.4 is in the top, just right of the center of the plot. Automobiles near the Ride ideal point tend to have a better ride than automobiles far away. It can be seen from the R squares that none of these ratings perfectly fits the model, so all of the interpretations are approximate.

Output 120.6.4 Preference Mapping Ideal Point Plot

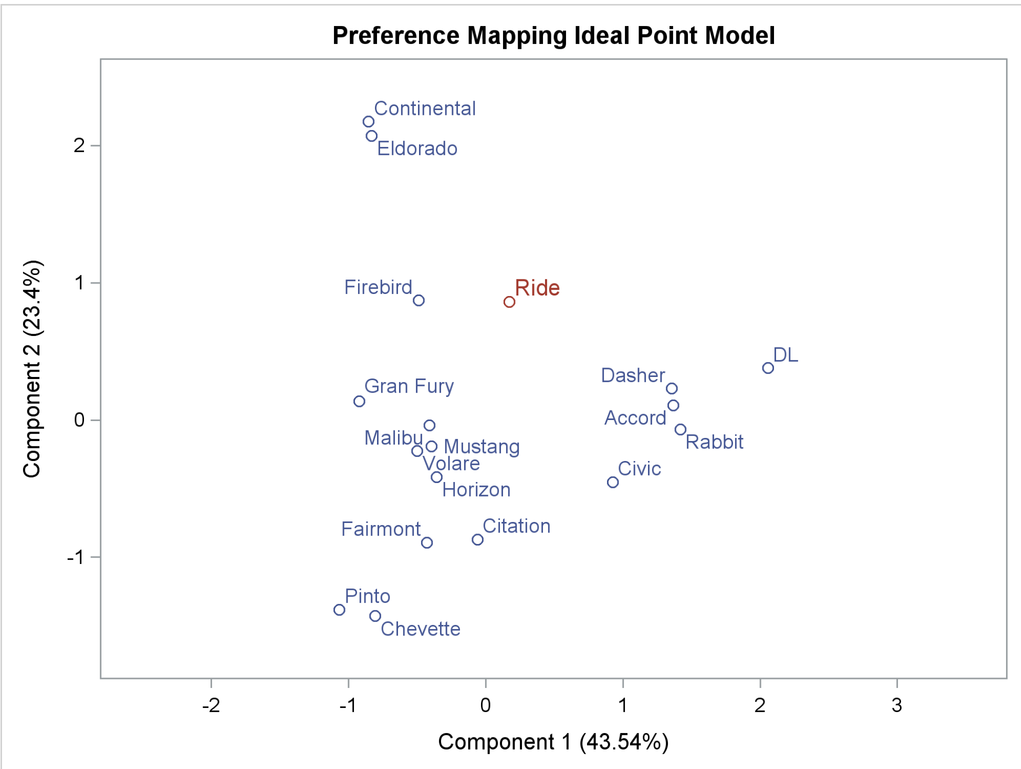
Preference Ratings for Automobiles Manufactured in 1980
Preference Mapping (PREFMAP) Analysis

The TRANSREG Procedure

The TRANSREG Procedure Hypothesis Tests for Identity(Ride)

R-Square	0.3780

Output 120.6.4 *continued*



The Ride point is a "negative-negative" ideal point. The point models assume that small ratings mean the object (automobile) is similar to the rating name and large ratings imply dissimilarity to the rating name. Because the opposite scoring is used, the interpretation of the Ride point must be reversed to a negative ideal point (bad ride). However, the coefficient for the _ISSQ_ variable in Output 120.6.5 is negative, so the interpretation is reversed again, back to the original interpretation.

Output 120.6.5 Preference Mapping Ideal Point Coefficients

Preference Ratings for Automobiles Manufactured in 1980
Preference Mapping (PREFMAP) Analysis

Obs	_TYPE_	_NAME_	Ride	Intercept	Prin1	Prin2	_ISSQ_	Model
1	M POINT	Ride	.	.	0.49461	2.46539	-0.17448	Ride

References

Akaike, H. (1973). "Information Theory and an Extension of the Maximum Likelihood Principle." In *Proceedings of the Second International Symposium on Information Theory*, edited by B. N. Petrov and F. Csáki, 267–281. Budapest: Akademiai Kiado.

Box, G. E. P., and Cox, D. R. (1964). "An Analysis of Transformations." *Journal of the Royal Statistical Society, Series B* 26:211–234.

Breiman, L., and Friedman, J. H. (1985). "Estimating Optimal Transformations for Multiple Regression and Correlation." *Journal of the American Statistical Association* 77:580–619. With discussion.

Brent, R. P. (1973). *Algorithms for Minimization without Derivatives*. Englewood Cliffs, NJ: Prentice-Hall. Chapter 5.

Brinkman, N. D. (1981). "Ethanol Fuel: A Single-Cylinder Engine Study of Efficiency and Exhaust Emissions." *Society of Automotive Engineers Transactions* 90:1410–1424.

Carroll, J. D. (1972). "Individual Differences and Multidimensional Scaling." In *Multidimensional Scaling: Theory and Applications in the Behavioral Sciences*, vol. 1, edited by R. N. Shepard, A. K. Romney, and S. B. Nerlove, 105–155. New York: Seminar Press.

Craven, P., and Wahba, G. (1979). "Smoothing Noisy Data with Spline Functions." *Numerical Mathematics* 31:377–403.

De Boor, C. (1978). *A Practical Guide to Splines*. New York: Springer-Verlag.

De Leeuw, J. (1986). *Regression with Optimal Scaling of the Dependent Variable*. Leiden, Netherlands: Department of Data Theory, University of Leiden.

De Leeuw, J., Young, F. W., and Takane, Y. (1976). "Additive Structure in Qualitative Data: An Alternating Least Squares Approach with Optimal Scaling Features." *Psychometrika* 41:471–503.

Draper, N. R., and Smith, H. (1981). *Applied Regression Analysis*. 2nd ed. New York: John Wiley & Sons.

Eilers, P. H. C., and Marx, B. D. (1996). "Flexible Smoothing with *B*-Splines and Penalties." *Statistical Science* 11:89–121. With discussion.

Fisher, R. A. (1938). *Statistical Methods for Research Workers*. 10th ed. Edinburgh: Oliver & Boyd.

Gabriel, K. R. (1981). "Biplot Display of Multivariate Matrices for Inspection of Data and Diagnosis." In *Interpreting Multivariate Data*, edited by V. Barnett, 571–572. Chichester, UK: John Wiley & Sons.

Gifi, A. (1990). *Nonlinear Multivariate Analysis*. New York: John Wiley & Sons.

Green, P. E., and Wind, Y. (1975). "New Way to Measure Consumers' Judgments." *Harvard Business Review* 53:107–117.

Hastie, T. J., and Tibshirani, R. J. (1986). "Generalized Additive Models." *Statistical Science* 3:297–318.

Hurvich, C. M., Simonoff, J. S., and Tsai, C.-L. (1998). "Smoothing Parameter Selection in Nonparametric Regression Using an Improved Akaike Information Criterion." *Journal of the Royal Statistical Society, Series B* 60:271–293.

Israels, A. Z. (1984). "Redundancy Analysis for Qualitative Variables." *Psychometrika* 49:331–346.

Judge, G. G., Griffiths, W. E., Hill, R. C., and Lee, T.-C. (1980). *The Theory and Practice of Econometrics*. New York: John Wiley & Sons.

Khuri, A. I., and Cornell, J. A. (1987). *Response Surfaces*. New York: Marcel Dekker.

Kruskal, J. B. (1964). "Nonmetric Multidimensional Scaling by Optimizing Goodness of Fit to a Nonmetric Hypothesis." *Psychometrika* 29:1–27.

Kuhfeld, W. F. (2010). *Marketing Research Methods in SAS*. Technical report, SAS Institute Inc., Cary, NC. `http://support.sas.com/resources/papers/tnote/tnote_marketresearch.html`.

Myers, R. H. (1976). *Response Surface Methodology*. Blacksburg: Virginia Polytechnic Institute and State University.

National Institute of Standards and Technology (1998). "Statistical Reference Data Sets." Accessed June 6, 2011. `http://www.itl.nist.gov/div898/strd/general/dataarchive.html`.

Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. (1989). *Numerical Recipes in PASCAL*. Cambridge: Cambridge University Press.

Reinsch, C. H. (1967). "Smoothing by Spline Functions." *Numerische Mathematik* 10:177–183.

SAS Institute Inc. (1993). *Algorithms for the PRINQUAL and TRANSREG Procedures*. Technical Report R-108, SAS Institute Inc., Cary, NC. `http://support.sas.com/publishing/pubcat/techreports/59040.pdf`.

Schiffman, S. S., Reynolds, M. L., and Young, F. W. (1981). *Introduction to Multidimensional Scaling*. New York: Academic Press.

Schwarz, G. (1978). "Estimating the Dimension of a Model." *Annals of Statistics* 6:461–464.

Siegel, S. (1956). *Nonparametric Statistics*. New York: McGraw-Hill.

Smith, P. L. (1979). "Splines as a Useful and Convenient Statistical Tool." *American Statistician* 33:57–62.

Stewart, D. K., and Love, W. A. (1968). "A General Canonical Correlation Index." *Psychological Bulletin* 70:160–163.

Van der Burg, E., and de Leeuw, J. (1983). "Non-linear Canonical Correlation." *British Journal of Mathematical and Statistical Psychology* 36:54–80.

Van Rijckevorsel, J. L. (1982). "Canonical Analysis with B-Splines." In *COMPUSTAT 1982, Part I*, edited by H. Caussinus, P. Ettinger, and R. Tomassone, 393–398. Vienna: Physica-Verlag.

Winsberg, S., and Ramsay, J. O. (1980). "Monotonic Transformations to Additivity Using Splines." *Biometrika* 67:669–674.

Young, F. W. (1981). “Quantitative Analysis of Qualitative Data.” *Psychometrika* 46:357–388.

Young, F. W., de Leeuw, J., and Takane, Y. (1976). “Regression with Qualitative and Quantitative Variables: An Alternating Least Squares Approach with Optimal Scaling Features.” *Psychometrika* 41:505–529.

Subject Index

Syntax Index