Chapter 65
The TRANSREG Procedure

Chapter Table of Contents

OVERVIEW 3369

GETTING STARTED 3370
Main-Effects ANOVA 3370
Detecting Nonlinear Relationships 3373

SYNTAX 3376
PROC TRANSREG Statement 3376
BY Statement 3379
FREQ Statement 3380
ID Statement 3380
MODEL Statement 3380
OUTPUT Statement 3402
WEIGHT Statement 3412

DETAILS 3413
Model Statement Usage 3413
Smoothing Splines 3415
Missing Values 3418
Missing Values, UNTIE, and Hypothesis Tests 3419
Controlling the Number of Iterations 3420
Using the REITERATE Algorithm Option 3421
Avoiding Constant Transformations 3422
Constant Variables 3422
Convergence and Degeneracies 3423
Implicit and Explicit Intercepts 3423
Passive Observations 3424
Point Models 3424
Redundancy Analysis 3425
Optimal Scaling 3428
OPSCORE, MONOTONE, UNTIE, and LINEAR Transformations 3428
SPLINE and MSPLINE Transformations 3430
Specifying the Number of Knots 3431
SPLINE, BSPLINE, and PSPLINE Comparisons 3433
Hypothesis Tests 3433
Output Data Set 3436

OUTTEST= Output Data Set 3444
Computational Resources 3445
Solving Standard Least-Squares Problems 3446
Using the DESIGN Output Option 3470
Choice Experiments: DESIGN, NORESTOREMISSING, NOZEROCONSTANT Usage 3476
ANOVA Codings 3477
Displayed Output 3490
ODS Table Names 3490

EXAMPLES 3491
Example 65.1 Using Splines and Knots 3491
Example 65.2 Nonmetric Conjoint Analysis of Tire Data 3500
Example 65.3 Metric Conjoint Analysis of Tire Data 3504
Example 65.4 Transformation Regression of Exhaust Emissions Data 3516
Example 65.5 Preference Mapping of Cars Data 3523

REFERENCES 3528

Chapter 65
The TRANSREG Procedure

Overview

The TRANSREG (transformation regression) procedure fits linear models, optionally with spline and other nonlinear transformations, and it can be used to code experimental designs prior to their use in other analyses.

The TRANSREG procedure fits many types of linear models, including

- ordinary regression and ANOVA
- metric and nonmetric conjoint analysis (Green and Wind 1975; de Leeuw, Young, and Takane 1976)
- 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 (Meyers 1976; Khuri and Cornell 1987) with variable transformations

The data set can contain variables measured on nominal, ordinal, interval, and ratio scales (Siegel 1956). Any mix of these variable types is allowed for the dependent and independent variables. The TRANSREG procedure can transform

- nominal variables by scoring the categories to minimize squared error (Fisher 1938), or they can be expanded into dummy variables
- 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.
- interval and ratio scale of measurement variables linearly or nonlinearly with spline (de Boor 1978; van Rijckevorsel 1982) or monotone spline (Winsberg and Ramsay 1980) transformations. In addition, smooth, 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 always explicitly optimizes a squared-error loss function.

PROC TRANSREG extends the ordinary general linear model by providing optimal variable transformations that are iteratively derived using the method of alternating least squares (Young 1981). PROC TRANSREG iterates until convergence, alternating

- finding least-squares estimates of the parameters of the model given the current scoring of the data (that is, the current vectors)
- finding least-squares estimates of the scoring parameters given the current set of model parameters

For more background on alternating least-squares optimal scaling methods and transformation regression methods, refer to 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), and Hastie and Tibshirani (1986). (These are just a few of the many relevant sources.)

Getting Started

This section provides several examples that illustrate features of the TRANSREG procedure.

Main-Effects ANOVA

This example shows how to use the TRANSREG procedure to code and fit a main-effects ANOVA model. The input data set contains the dependent variables Y, factors X1 and X2, and 11 observations. The following statements perform a main-effects ANOVA:

```
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;

*---Print TRANSREG output data set---;
proc print label;
   format Intercept -- X2a 5.2;
run;
```

```
                Introductory Main-Effects ANOVA Example

                        The TRANSREG Procedure

     TRANSREG Univariate Algorithm Iteration History for Identity(Y)

 Iteration    Average    Maximum                Criterion
    Number     Change     Change    R-Square       Change    Note
 ---------------------------------------------------------------------------
         1    0.00000    0.00000     0.88144                 Converged

 Algorithm converged.


          The TRANSREG Procedure Hypothesis Tests for Identity(Y)


        Univariate ANOVA Table Based on the Usual Degrees of Freedom

                                 Sum of        Mean
    Source               DF     Squares      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

                              Type II
                               Sum of      Mean
 Variable      DF  Coefficient  Squares    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 65.1. ANOVA Example Output from PROC TRANSREG

The iteration history in Figure 65.1 shows that the final R-Square of 0.88144 is reached on the first iteration.

This is followed by ANOVA, fit statistics, and regression tables. PROC TRANSREG uses an effects (also called deviations from means or 0, 1, -1) coding in this example.

The TRANSREG procedure produces the data set displayed in Figure 65.2.

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		

Figure 65.2. Output Data Set from PROC TRANSREG

The output data set has three kinds of observations, identified by values of _TYPE_.

- When _TYPE_='SCORE', the observation contains information on the dependent and independent variables as follows:
  - 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.
- When _TYPE_='MEAN', the observation contains the marginal means.

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

Detecting Nonlinear Relationships

The TRANSREG procedure can detect nonlinear relationships among variables. For example, suppose 400 observations are generated from the following function

$$t = \frac{x}{4} + \sin(x)$$

and data are created as follows

$$y = t + \epsilon$$

where ϵ is random normal error.

The following statements find a cubic spline transformation of X with four knots. For information on using splines and knots, see Example 65.1.

The following statements produce Figure 65.3 through Figure 65.4:

```
title 'Curve Fitting Example';

*---Create An Artificial Nonlinear Scatter Plot---;
data Curve;
   Pi=3.14159265359;
   Pi4=4*Pi;
   Increment=Pi4/400;
   do X=Increment to Pi4 by Increment;
      T=X/4 + sin(X);
      Y=T + normal(7);
      output;
      end;
run;

*---Request a Spline Transformation of X---;
proc transreg data=Curve dummy;
   model identity(Y)=spline(X / nknots=4);
   output predicted;
   id T;
run;

*---Plot the Results---;
goptions goutmode=replace nodisplay;
%let opts = haxis=axis2 vaxis=axis1 frame cframe=ligr;
* Depending on your goptions, these plot options may work better:
* %let opts = haxis=axis2 vaxis=axis1 frame;

proc gplot;
   title;
   axis1 minor=none label=(angle=90 rotate=0);
   axis2 minor=none;
   plot T*X=2              / &opts name='tregin1';
   plot Y*X=1              / &opts name='tregin2';
```

```
   plot Y*X=1 T*X=2 PY*X=3 / &opts name='tregin3' overlay ;
   symbol1 color=blue   v=star i=none;

   symbol2 color=yellow v=none i=join line=1;
   symbol3 color=red    v=none i=join line=2;
run; quit;

goptions display;
proc greplay nofs tc=sashelp.templt template=l2r2;
   igout gseg;
   treplay 1:tregin1 2:tregin3 3:tregin2;
run; quit;
```

PROC TRANSREG increases the squared multiple correlation from the original value of 0.19945 to 0.47062. The plot of T by X shows the original function, the plot of Y by X shows the error-perturbed data, and the third plot shows the data, the true function as a solid curve, and the regression function as the dashed curve. The regression function closely approximates the true function.

```
                        Curve Fitting Example

                        The TRANSREG Procedure

       TRANSREG MORALS Algorithm Iteration History for Identity(Y)

Iteration    Average    Maximum                Criterion
   Number     Change     Change    R-Square       Change    Note
-------------------------------------------------------------------------
        0    0.74855    1.29047     0.19945
        1    0.00000    0.00000     0.47062      0.27117    Converged

Algorithm converged.
```

Figure 65.3. Curve Fitting Example Output

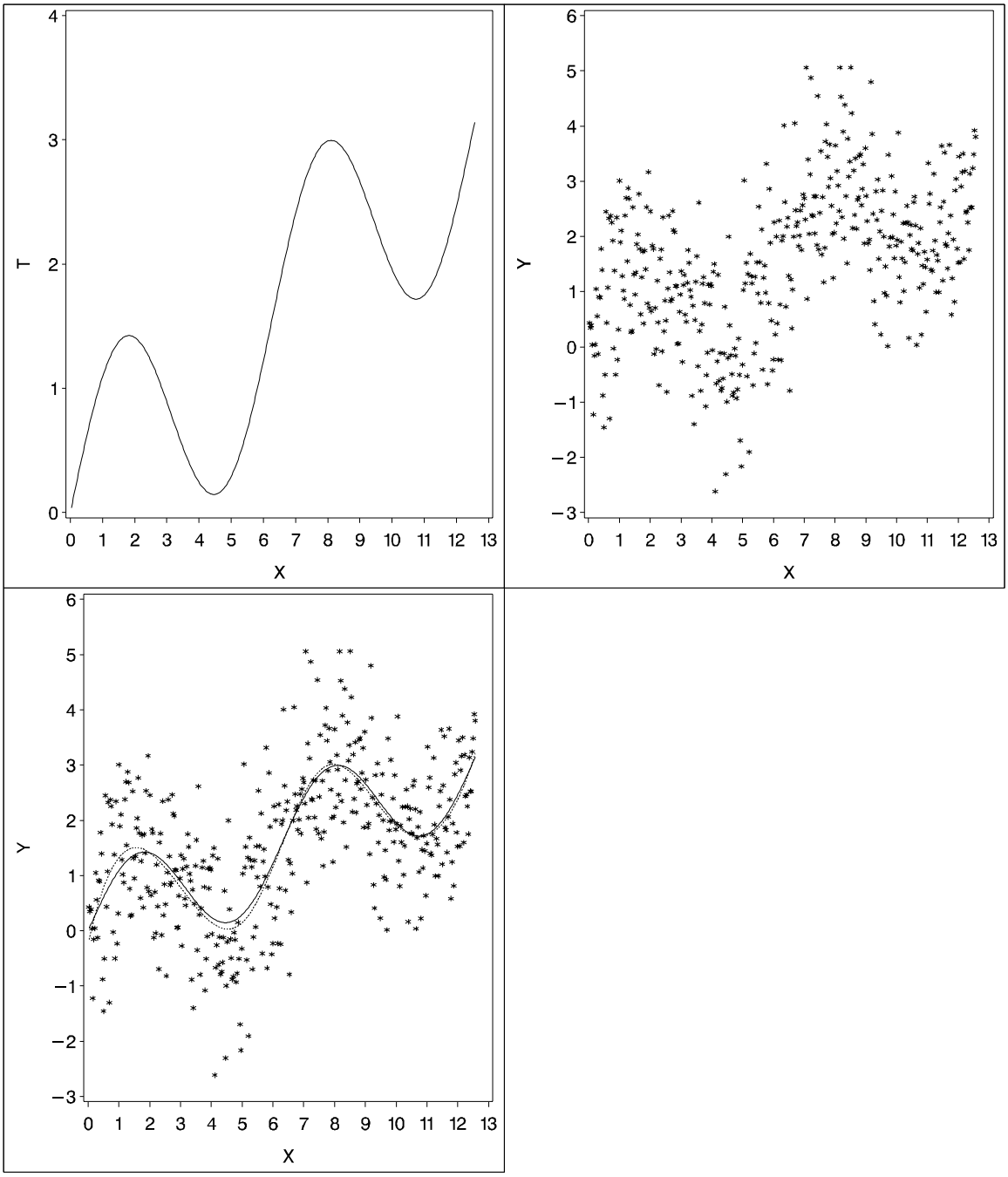


Figure 65.4. Plots for the Curve Fitting Example

Syntax

The following statements are available in PROC TRANSREG.

PROC TRANSREG < **DATA=***SAS-data-set* >
< **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 the TRANSREG procedure, you need 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 the TRANSREG procedure and is not generally true of other SAS/STAT procedures. See Table 65.1 on page 3377.

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* >
< **OUTTEST=***SAS-data-set* >< *a-options* >< *o-options* > **;**

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.

The following table summarizes options available 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.

Table 65.1. Options Available in the TRANSREG Procedure

Task	Option	Statement
Identify input data set		
specifies input SAS data set	DATA=	PROC
Output data set with test statistics		
specifies output test statistics data set	OUTTEST=	PROC
Input data set		
specifies input observation type	TYPE=	MODEL
restarts iterations	REITERATE	MODEL
Specify method and control iterations		
specifies minimum criterion change	CCONVERGE=	MODEL
specifies minimum data change	CONVERGE=	MODEL
specifies canonical dummy-variable initialization	DUMMY	MODEL
specifies maximum number of iterations	MAXITER=	MODEL
specifies iterative algorithm	METHOD=	MODEL
specifies number of canonical variables	NCAN=	MODEL
specifies singularity criterion	SINGULAR=	MODEL
Control missing data handling		
includes monotone special missing values	MONOTONE=	MODEL
excludes observations with missing values	NOMISS	MODEL
unties special missing values	UNTIE=	MODEL
Control intercept and CLASS variables		
CLASS dummy variable name prefix	CPREFIX=	MODEL
CLASS dummy variable label prefix	LPREFIX=	MODEL
no intercept or centering	NOINT	MODEL
order of class variable levels	ORDER=	MODEL
controls output of reference levels	REFERENCE=	MODEL
CLASS dummy variable label separators	SEPARATORS=	MODEL
Control displayed output		
confidence limits alpha	ALPHA=	MODEL
displays parameter estimate confidence limits	CL	MODEL
displays model specification details	DETAIL	MODEL
displays iteration histories	HISTORY	MODEL
suppresses displayed output	NOPRINT	MODEL
suppresses the iteration histories	SHORT	MODEL
displays regression results	SS2	MODEL
displays ANOVA table	TEST	MODEL
displays conjoint part-worth utilities	UTILITIES	MODEL
Control standardization		
fits additive model	ADDITIVE	MODEL
do not zero constant variables	NOZEROCONSTANT	MODEL
specifies transformation standardization	TSTANDARD=	MODEL
Predicted values, residuals, scores		
outputs canonical scores	CANONICAL	OUTPUT
outputs individual confidence limits	CLI	OUTPUT

Table 65.1. (continued)

Task	Option	Statement
outputs mean confidence limits	CLM	OUTPUT
specifies design matrix coding	DESIGN=	OUTPUT
outputs leverage	LEVERAGE	OUTPUT
does not restore missing values	NORESTOREMISSING	OUTPUT
suppresses output of scores	NOSCORES	OUTPUT
outputs predicted values	PREDICTED	OUTPUT
outputs redundancy variables	REDUNDANCY=	OUTPUT
outputs residuals	RESIDUALS	OUTPUT
Output data set replacement		
replaces dependent variables	DREPLACE	OUTPUT
replaces independent variables	IREPLACE	OUTPUT
replaces all variables	REPLACE	OUTPUT
Output data set coefficients		
outputs coefficients	COEFFICIENTS	OUTPUT
outputs ideal point coordinates	COORDINATES	OUTPUT
outputs marginal means	MEANS	OUTPUT
outputs redundancy analysis coefficients	MREDUNDANCY	OUTPUT
Output data set variable name prefixes		
dependent variable approximations	ADPREFIX=	OUTPUT
independent variable approximations	AIPREFIX=	OUTPUT
canonical dependent variables	CDPREFIX=	OUTPUT
conservative individual lower CL	CILPREFIX=	OUTPUT
canonical independent variables	CIPREFIX=	OUTPUT
conservative-individual-upper CL	CIUPREFIX=	OUTPUT
conservative-mean-lower CL	CMLPREFIX=	OUTPUT
conservative-mean-upper CL	CMUPREFIX=	OUTPUT
METHOD=MORALS untransformed dependent	DEPENDENT=	OUTPUT
liberal-individual-lower CL	LILPREFIX=	OUTPUT
liberal-individual-upper CL	LIUPREFIX=	OUTPUT
liberal-mean-lower CL	LMLPREFIX=	OUTPUT
liberal-mean-upper CL	LMUPREFIX=	OUTPUT
residuals	RDPREFIX=	OUTPUT
predicted values	PPREFIX=	OUTPUT
redundancy variables	RPREFIX=	OUTPUT
transformed dependents	TDPREFIX=	OUTPUT
transformed independents	TIPREFIX=	OUTPUT
Output data set macros		
creates macro variables	MACRO	OUTPUT
Output data set details		
dependent and independent approximations	APPROXIMATIONS	OUTPUT
canonical correlation coefficients	CCC	OUTPUT
canonical elliptical point coordinate	CEC	OUTPUT
canonical point coordinates	CPC	OUTPUT
canonical quadratic point coordinates	CQC	OUTPUT

Table 65.1. (continued)

Task	Option	Statement
approximations to transformed dependents	DAPPROXIMATIONS	OUTPUT
approximations to transformed independents	IAPPROXIMATIONS	OUTPUT
elliptical point coordinates	MEC	OUTPUT
point coordinates	MPC	OUTPUT
quadratic point coordinates	MQC	OUTPUT
multiple regression coefficients	MRC	OUTPUT

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.

BY Statement

BY *variables* **;**

You can specify a BY statement with PROC TRANSREG to obtain separate analyses on observations in groups defined by the BY variables. When a BY statement appears, the procedure expects the input data set to be sorted in order of the BY variables.

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

- Sort the data using the SORT procedure with a similar BY statement.
- Specify the BY statement option NOTSORTED or DESCENDING in the BY statement for the 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 using the DATASETS procedure.

For more information on the BY statement, refer to the discussion in *SAS Language Reference: Concepts*. For more information on the DATASETS procedure, refer to the discussion in the *SAS 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.

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 the TRANSREG procedure. The *t-options* are transformation options, and the *a-options* are the 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 dummy 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.

```
   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:

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

See the "Types of Effects" section on page 1518 in Chapter 30, "The GLM Procedure," for a description of the @, *, and | operators and see the "Model Statement Usage" section on page 3413 for information on how to use these operators in PROC TRANSREG. Note that nesting is not allowed in PROC TRANSREG.

The next three sections discuss the transformations available (*transforms*) (see the "Families of Transformations" section on page 3381), the transformation options (*t-options*) (see the "Transformation Options (t-options)" section on page 3387), and the algorithm options (*a-options*) (see the "Algorithm Options (a-options)" section on page 3393).

Families of Transformations

In the MODEL statement, *transform* specifies a transformation in one of four 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.
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 following table summarizes the transformations in each family.

Family	Members of Family
Variable expansions	
B-spline basis	BSPLINE
set of dummy variables	CLASS
elliptical response surface	EPOINT
circular response surface	POINT
piecewise polynomial basis	PSPLINE
quadratic response surface	QPOINT
Nonoptimal transformations	
inverse trigonometric sine	ARSIN
exponential	EXP
logarithm	LOG
logit	LOGIT
raises variables to specified power	POWER
transforms to ranks	RANK
noniterative smoothing spline	SMOOTH
Optimal transformations	
linear	LINEAR
monotonic, ties preserved	MONOTONE
monotonic B-spline	MSPLINE
optimal scoring	OPSCORE
B-spline	SPLINE
monotonic, ties not preserved	UNTIE
Other transformations	
identity, no transformation	IDENTITY
iterative smoothing spline	SSPLINE

You can use any transformation with either dependent or independent variables (except the SMOOTH transformation, which can be used only with independent 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. Optionally, depending on the *transform*, the parentheses can also contain *t-options*, which follow the variables and a slash. For example,

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

finds a LOG transformation of Y and performs a CLASS expansion of X.

```
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 four families of transformations. The *t-options* are discussed in the "Transformation Options (t-options)" section on page 3387.

Variable Expansions

The TRANSREG procedure 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 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 "Point Models" section on page 3424). 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 (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. See the section "SPLINE, BSPLINE, and PSPLINE Comparisons" on page 3433. Variables following BSPLINE must be numeric, and they are typically continuous.

CLASS
CLA

expands the variables to a set of dummy variables. For example, CLASS(X1 X2) is used for 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) creates all main effects, all two-way interactions, and one three-way interaction. See the "Model Statement Usage" section on page 3413 for information on how to use the operators @, *, and | in PROC TRANSREG. To determine class membership, PROC TRANSREG uses the

values of the formatted variables. Variables following CLASS can be either character or numeric; numeric variables should be discrete.

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 parenthesized variable. The regression analysis then uses both sets of variables (original and squared). Variables following EPOINT must be numeric, and they are typically continuous.

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 sums 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 on ideal point regression, refer to Carroll (1972). Variables following POINT must be numeric, and they are typically continuous.

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. Refer to Smith (1979) for a good introduction to piecewise polynomial splines. Also see the section "SPLINE, BSPLINE, and PSPLINE Comparisons" on page 3433. Variables following PSPLINE must be numeric, and they are typically continuous.

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 following QPOINT must be numeric, and they are typically continuous.

Nonoptimal Transformations

Like variable expansions, nonoptimal transformations 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 following ARSIN must be numeric, in the interval $(-1.0 \leq \mathsf{X} \leq 1.0)$, and they are typically continuous.

EXP
exponentiates variables (the variable 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 following EXP must be numeric, and they are typically continuous.

LOG
transforms variables to logarithms (the variable 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 following LOG 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 following LOGIT must be numeric, in the interval $(0.0 < \mathsf{X} < 1.0)$, and they are typically continuous.

POWER
POW
raises variables to a specified power (the variable X is transformed to X^a). You must specify the power parameter a by specifying the PARAMETER= *t-option* following the variables:

```
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 following POWER 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 following RANK must be numeric.

SMOOTH
SMO
is a noniterative smoothing spline transformation. You can specify the smoothing parameter with either the SM= or the PARAMETER= *t-option*. The default smoothing parameter is SM=0. Variables following SMOOTH must be numeric, and they

are typically continuous. The SMOOTH transformation can be used only with independent variables. For more information, see the "Smoothing Splines" section on page 3415.

Optimal Transformations

Optimal transformations are iteratively derived. Missing values for these types of variables can be optimally estimated (see the "Missing Values" section on page 3418).

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 following LINEAR must be numeric.

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 following MONOTONE must be numeric, and they are typically discrete.

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 uses a quadratic spline. Variables following MSPLINE must be numeric, and they are typically continuous.

OPSCORE
OPS

finds an optimal scoring of each variable. The OPSCORE transformation assigns scores to each class (level) of the variable. Fisher's (1938) optimal scoring method is used. Variables following OPSCORE can be either character or numeric; numeric variables should be discrete.

SPLINE
SPL

finds a B-spline transformation (de Boor 1978) of each variable. By default, PROC TRANSREG uses a cubic polynomial transformation. You can specify the DEGREE=, KNOTS=, NKNOTS=, and EVENLY *t-options* with SPLINE. Variables following SPLINE must be numeric, and they are typically continuous.

UNTIE
UNT

finds a monotonic transformation of each variable without the restriction that ties are preserved. The TRANSREG procedure uses the Kruskal (1964) primary least-squares monotonic transformation method. This transformation weakly preserves

order but not category membership (it may untie some previously tied values). Variables following UNTIE must be numeric, and they are typically discrete.

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 "Model Statement Usage" section on page 3413 for information on how to use the operators @, *, and | in PROC TRANSREG.

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. Variables following IDENTITY must be numeric.

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 following SSPLINE must be numeric, and they are typically continuous.

Transformation Options (t-options)

If you use a nonoptimal, 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 dependent and independent variables. For example,

```
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.

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

The following table summarizes the *t-options*.

Table 65.2. t-options Available in the MODEL Statement

Task	Option
Nonoptimal transformation t-options	
uses original mean and variance	ORIGINAL
Parameter t-options	
specifies miscellaneous parameters	PARAMETER=
specifies smoothing parameter	SM=
Spline t-options	
specifies the degree of the spline	DEGREE=
spaces the knots evenly	EVENLY
specifies the interior knots or break points	KNOTS=
creates n knots	NKNOTS=
CLASS Variable t-options	
CLASS dummy variable name prefix	CPREFIX=
requests a deviations-from-means coding	DEVIATIONS
requests a deviations-from-means coding	EFFECTS
CLASS dummy variable label prefix	LPREFIX=
order of class variable levels	ORDER=
CLASS dummy variable label separators	SEPARATORS=
controls reference levels	ZERO=
Other t-options	
operations occur after the expansion	AFTER
renames variables	NAME=
reflects the variable around the mean	REFLECT
specifies transformation standardization	TSTANDARD=

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 EXP, LOG, POWER, SMOOTH, and SSPLINE transformations. 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. (You can specify a SAS/GRAPH-style smoothing parameter with the SM= *t-option*.) The default for the PARAMETER= *t-option* 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.

SM=*n*

specifies a SAS/GRAPH-style smoothing parameter in the range 0 to 100. 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 and MSPLINE optimal transformations and 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
EVE

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

If you specify NKNOTS=k, k knots are created at

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

for $i = 1, \ldots, k$. For example, if you specify

```
spline(X / knots=2 evenly)
```

and 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.

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 $(n-1)$th 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 3431).

NKNOTS=*n*
NKN=*n*

creates n knots, the first at the $100/(n+1)$ percentile, the second at the $200/(n+1)$ 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 twenty-fifth percentile, the median, and the seventy-fifth percentile. By default, NKNOTS=0. The NKNOTS= *t-option* must be ≥ 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 3431).

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 dummy variables. When CPREFIX= is specified as an *a-option* (see the description of the CPREFIX= *a-option* on page 3396) 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-list *t-option*.

DEVIATIONS
DEV
EFFECTS
EFF

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.

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 dummy variables. When LPREFIX= is specified as an *a-option* (see the description of the LPREFIX= *a-option*

on page 3396) 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-list *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 ORDER= is specified as an *a-option* (see the description of the ORDER= *a-option* on page 3398) or as 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.

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 SEPARATORS= is specified as an *a-option* (see the description of the SEPARATORS= *a-option* on page 3399) 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.

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 dummy 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 dummy 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 dummy variable for the category with a formatted value that matches '*formatted-value*', implying a zero coefficient for that category. With ZERO=formatted-value-list, 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 dummy variable coefficients.

The specification CLASS(variable / ZERO=NONE) sets to missing none of the dummy 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 dummy variables, and the coefficients for the dummy 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.

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 external knots.

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

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

NAME=(*variable-list*)
NAM=(*variable-list*)

renames variables as they are used in the MODEL statement. This *t-option* allows a variable to be used 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 allow 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 65.2.)

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 TSTANDARD= is specified as an *a-option* (see the description of the TSTANDARD= *a-option* on page 3400) 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

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

Algorithm Options (a-options)

This section discusses the options that can appear in the PROC TRANSREG or MODEL statements as *a-options*. They are listed after the entire model specification and after a slash.

For 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 65.3. Options Available in the PROC TRANSREG or MODEL Statements

Task	Option
Input data set	
specifies input observation type	TYPE=
restarts iterations	REITERATE
Specify method and control iterations	
specifies minimum criterion change	CCONVERGE=
specifies minimum data change	CONVERGE=
specifies canonical dummy-variable initialization	DUMMY
specifies maximum number of iterations	MAXITER=
specifies iterative algorithm	METHOD=
specifies number of canonical variables	NCAN=
specifies singularity criterion	SINGULAR=
Control missing data handling	
includes monotone special missing values	MONOTONE=
excludes observations with missing values	NOMISS
unties special missing values	UNTIE=
Control intercept and CLASS variables	
CLASS dummy variable name prefix	CPREFIX=
CLASS dummy variable label prefix	LPREFIX=
no intercept or centering	NOINT
order of class variable levels	ORDER=
controls output of reference levels	REFERENCE=
CLASS dummy variable label separators	SEPARATORS=
Control displayed output	
confidence limits alpha	ALPHA=
displays parameter estimate confidence limits	CL
displays model specification details	DETAIL
displays iteration histories	HISTORY
suppresses displayed output	NOPRINT
suppresses the iteration histories	SHORT
displays regression results	SS2
displays ANOVA table	TEST

Table 65.3. (continued)

Task	Option
displays conjoint part-worth utilities	UTILITIES
Control standardization	
fits additive model	ADDITIVE
do not zero constant variables	NOZEROCONSTANT
specifies transformation standardization	TSTANDARD=

The following list provides details on these *a-options*.

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 dummy variables. Dummy 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 dummy variables are named ClassVariable1, ClassVariable2, and ClassVariable3. However, with CPREFIX=5, the dummy variables are named Class1, Class2, and Class3. When CPREFIX=0, dummy 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 CPREFIX= is specified 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, and so on.

DUMMY
DUM

provides a canonical dummy variable initialization. When there are no monotonicity constraints and there is only one canonical variable in each set, PROC TRANSREG (with the DUMMY *a-option*) can usually find the optimal solution in only one iteration. The initialization iteration is number 0, which is slower and uses more memory than other iterations. However, when there are no monotonicity constraints, when there is only one canonical variable in each set, and when there is enough available memory, specifying the DUMMY *a-option* can greatly decrease the amount of time required to find the optimal transformations. Furthermore, by solving for the transformations directly instead of iteratively, PROC TRANSREG avoids certain nonoptimal solutions.

HISTORY
HIS

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

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 dummy variables. Dummy 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 LPREFIX= is specified 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. By default, MAXITER=30. A specification of MAXITER=0 is allowed 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*

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.

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 using 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 "Missing Values" section on page 3418.

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 ≥ 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 allowable value.

NOINT
NOI

omits the intercept from the OUT= data set and suppresses centering of data. The NOINT *a-option* is not allowed with iterative transformations since there is no provision for optimal scaling without an intercept. The NOINT *a-option* is allowed 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 "Missing Values" section on page 3418.

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 "Passive Observations" section on page 3424 for more information on 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* temporarily disables the Output Delivery System (ODS). For more information, see Chapter 15, "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 designs for choice models. When these designs are very large, it may be more efficient to code by subject and choice set. When attributes are constant within choice set, specify the NOZEROCONSTANT option to get the correct results. You can specify this option in the PROC TRANSREG, MODEL, and OUTPUT statements.

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 ORDER= is specified 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.

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. The REFERENCE= option can be specified in the PROC TRANSREG, MODEL, or OUTPUT statement, and it can be independently specified 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 the TRANSREG procedure 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 3421.

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 dummy 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 SEPARATORS= is specified 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. The TRANSREG procedure uses the value of the SINGULAR= *a-option* for checking $1 - R^2$ 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 implied. See the section "Hypothesis Tests" on page 3433. 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 3433.

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 TSTANDARD= is specified 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 LOG(X), not LOG(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 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 value 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 "Missing Values" section on page 3418 and the "Optimal Scaling" section on page 3428.

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= qoption. When you specify the UTILITIES *a-option*, the TEST *a-option* is implied. Refer to SAS Technical Report R-109, *Conjoint Analysis Examples*, for more information on 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 on original and transformed variables. The information on 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 on data set structure, see the "Output Data Set" section on page 3436.

To specify the data set, use the OUT= specification.

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 DATAn convention. If you want to create a permanent SAS data set, you must specify a two-level name (refer to "SAS Files" in *SAS Language Reference: Concepts* and "Introduction to DATA Step Processing" in the *SAS Procedures Guide* for details).

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)

The following table provides a summary of options in the OUTPUT statement. These options include the OUT= option and all of the *o-options*.

Table 65.4. Options Available in the OUTPUT Statement

Task	Option
Identify output data set	
output data set	OUT=
Predicted values, residuals, scores	
outputs canonical scores	CANONICAL
outputs individual confidence limits	CLI
outputs mean confidence limits	CLM
specifies design matrix coding	DESIGN=

Table 65.4. (continued)

Task	Option
outputs leverage	LEVERAGE
does not restore missings	NORESTOREMISSING
suppresses output of scores	NOSCORES
outputs predicted values	PREDICTED
outputs redundancy variables	REDUNDANCY=
outputs residuals	RESIDUALS
Output data set replacement	
replaces dependent variables	DREPLACE
replaces independent variables	IREPLACE
replaces all variables	REPLACE
Output data set coefficients	
outputs coefficients	COEFFICIENTS
outputs ideal point coordinates	COORDINATES
outputs marginal means	MEANS
outputs redundancy analysis coefficients	MREDUNDANCY
Output data set variable name prefixes	
dependent variable approximations	ADPREFIX=
independent variable approximations	AIPREFIX=
canonical dependent variables	CDPREFIX=
conservative individual lower CL	CILPREFIX=
canonical independent variables	CIPREFIX=
conservative-individual-upper CL	CIUPREFIX=
conservative-mean-lower CL	CMLPREFIX=
conservative-mean-upper CL	CMUPREFIX=
METHOD=MORALS untransformed dependent	DEPENDENT=
liberal-individual-lower CL	LILPREFIX=
liberal-individual-upper CL	LIUPREFIX=
liberal-mean-lower CL	LMLPREFIX=
liberal-mean-upper CL	LMUPREFIX=
residuals	RDPREFIX=
predicted values	PPREFIX=
redundancy variables	RPREFIX=
transformed dependents	TDPREFIX=
transformed independents	TIPREFIX=
Output data set macros	
creates macro variables	MACRO
Control CLASS variables	
controls output of reference levels	REFERENCE=
Output data set details	
dependent and independent approximations	APPROXIMATIONS
canonical correlation coefficients	CCC
canonical elliptical point coordinate	CEC
canonical point coordinates	CPC
canonical quadratic point coordinates	CQC

Table 65.4. (continued)

Task	Option
approximations to transformed dependents	DAPPROXIMATIONS
approximations to transformed independents	IAPPROXIMATIONS
elliptical point coordinates	MEC
point coordinates	MPC
quadratic point coordinates	MQC
multiple regression coefficients	MRC

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 on 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. Specifying the ADPREFIX= *o-option* also implies the PREDICTED *o-option*, and 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. Specifying the AIPREFIX= *o-option* also implies the IAPPROXIMATIONS *o-option*.

APPROXIMATIONS
APPROX
APP
is equivalent to specifying both the DAPPROXIMATIONS and the IAPPROXIMATIONS *o-options*. If METHOD=UNIVARIATE, then the APPROXIMATIONS *o-option* implies only the DAPPROXIMATIONS *o-option*.

CANONICAL
CAN
outputs canonical variables to the OUT= data set. When METHOD=CANALS, the CANONICAL *o-option* is implied. 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. Specifying the CDPREFIX= *o-option* also implies the CANONICAL *o-option*.

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. Specifying the CILPREFIX= *o-option* also implies the CLI *o-option*.

CIPREFIX=*name*
CIP=*name*
provides a prefix for naming the canonical independent variables. The default is CIPREFIX=Cani. Specifying the CIPREFIX= *o-option* also implies the CANONICAL *o-option*.

CIUPREFIX=*name*
CIU=*name*
specifies a prefix for naming the conservative-individual-upper confidence limits. The default prefix is CIU. Specifying the CIUPREFIX= *o-option* also implies the CLI *o-option*.

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. Specifying the CMLPREFIX= *o-option* also implies the CLM *o-option*.

CMUPREFIX=*name*
CMU=*name*
specifies a prefix for naming the conservative-mean-upper confidence limits. The default prefix is CMU. Specifying the CMUPREFIX= *o-option* also implies the CLM *o-option*.

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
COO
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 "Point Models" section on page 3424.

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 implied.

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 will 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 section "Using the DESIGN Output Option" on page 3470 and the section "Choice Experiments: DESIGN, NORESTOREMISSING, NOZEROCONSTANT Usage" on page 3476.

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 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. Specifying the AIPREFIX= *o-option* also implies the IAPPROXIMATIONS *o-option*. 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 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. Specifying the LILPREFIX= *o-option* also implies the CLI *o-option*.

LIUPREFIX=*name*
LIU=*name*
specifies a prefix for naming the liberal-individual-upper confidence limits. The default prefix is LIU. Specifying the LIUPREFIX= *o-option* also implies the CLI *o-option*.

LMLPREFIX=*name*
LML=*name*
specifies a prefix for naming the liberal-mean-lower confidence limits. The default prefix is LML. Specifying the LMLPREFIX= *o-option* also implies the CLM *o-option*.

LMUPREFIX=*name*
LMU=*name*
specifies a prefix for naming the liberal-mean-upper confidence limits. The default prefix is LMU. Specifying the LMUPREFIX= *o-option* also implies the CLM *o-option*.

MACRO(*keyword=name...***)**
MAC(*keyword=name...***)**
creates macro variables. Most of the options available within the MACRO *o-option* are rarely needed. By default, the TRANSREG procedure creates a macro variable named _TRGIND with a complete list of independent variables created by the procedure. When the TRANSREG procedure 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 could use the following MODEL statement:

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

The TRANSREG procedure, also by default, creates a macro variable named _TRGINDN, which contains the number of variables in the _TRGIND list. This macro variable could be used in an ARRAY statement as follows:

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

See the section "Using the DESIGN Output Option" on page 3470 and the section "Choice Experiments: DESIGN, NORESTOREMISSING, NOZEROCONSTANT Usage" on page 3476 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, 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, 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 below, 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(ID=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 below, but 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 the TRANSREG procedure is used to code designs for 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. Specifying the PPREFIX= *o-option* also implies the PREDICTED *o-option*.

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. Specifying the PPREFIX= *o-option* also implies the PREDICTED *o-option*, and 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. Specifying the RDPREFIX= *o-option* also implies the RESIDUALS *o-option*.

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 implied by

METHOD=REDUNDANCY. 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. The REFERENCE= option can be specified in the PROC TRANSREG, MODEL, or OUTPUT statement, and it can be independently specified 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. Specifying the RPREFIX= *o-option* also implies the REDUNDANCY *o-option*.

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

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 just 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);
```

Smoothing Splines

You can use PROC TRANSREG to output to a SAS data set the same smoothing splines that the GPLOT procedure creates. The SMOOTH transformation is a noniterative transformation for smoothing splines. The smoothing parameter can be specified with either the SM= or the PARAMETER= *o-option*. The independent variable transformation (Tx in this case) contains the results. The GPLOT request y*x=2 with I=SM50 creates the same curve as Tx*x.

```
title 'Smoothing Splines';

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;
run;

%let opts = haxis=axis2 vaxis=axis1 frame cframe=ligr;
proc gplot;
   axis1 minor=none label=(angle=90 rotate=0);
   axis2 minor=none;
   plot y*x=1 y*x=2 tx*x=3 / &opts overlay;
   symbol1 color=blue   v=star i=none;
   symbol2 color=yellow v=none i=sm50;
   symbol3 color=cyan   v=dot  i=none;
run; quit;
```

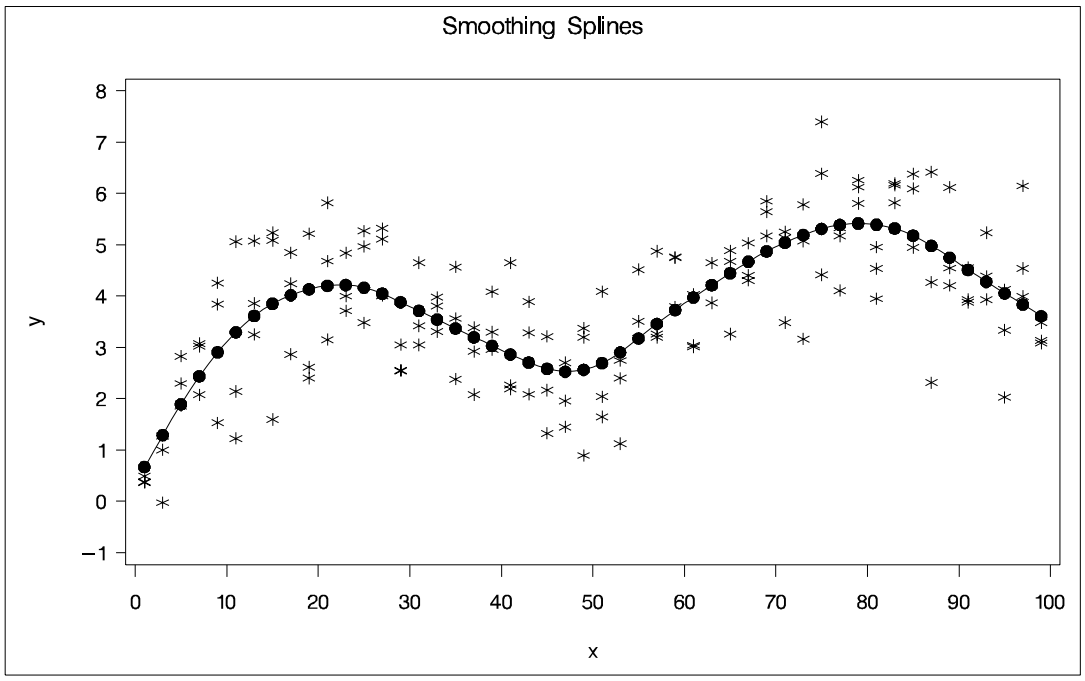


Figure 65.5. Smoothing Spline Example 1

When you cross a SMOOTH variable with a CLASS variable, specify ZERO=NONE with the CLASS expansion and the AFTER *t-option* with the SMOOTH transformation so that separate functions are found within each group.

```
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;
   model identity(y) = class(group / zero=none) |
                       smooth(x / after sm=50);
   output out=curves;
run;

data curves2;
   set curves;
   if group1 = 0 then tgroup1x = .;
```

```
   if group2 = 0 then tgroup2x = .;
run;

%let opts = haxis=axis2 vaxis=axis1 frame cframe=ligr;
proc gplot;
   axis1 minor=none label=(angle=90 rotate=0);
   axis2 minor=none;
   plot y*x=1 tgroup1x*x=2 tgroup2x*x=2 / &opts overlay;
   symbol1 color=blue   v=star i=none;
   symbol2 color=yellow v=none i=join;
run; quit;
```

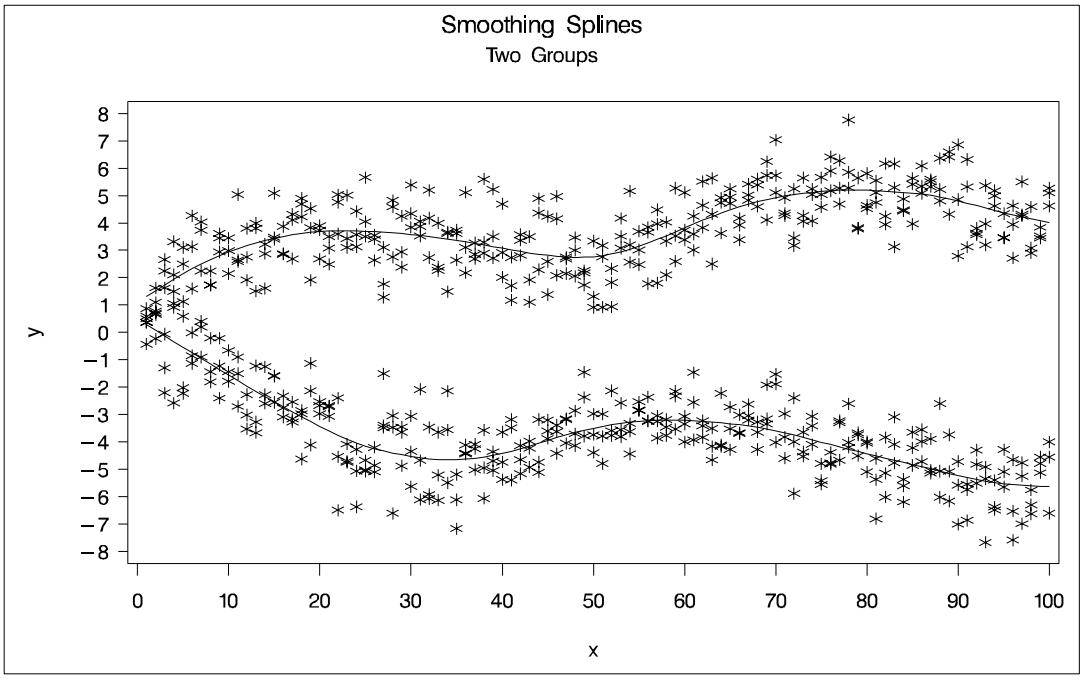


Figure 65.6. Smoothing Spline Example 2

The SMOOTH transformation is valid only with independent variables; typically, it is used in models with a single independent and a single dependent variable. When there are multiple independent variables designated as SMOOTH, the TRANSREG procedure tries to smooth the ith independent variable using the ith 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, for the model

```
model identity(y1-y3) = smooth(x1-x5);
```

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

The SMOOTH transformation is a noniterative transformation; smoothing occurs once per variable 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.

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, 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, using variable means as initial estimates.

You can specify the LINEAR, OPSCORE, MONOTONE, UNTIE, SPLINE, MSPLINE, SSPLINE, LOG, LOGIT, POWER, ARSIN, 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 (refer to "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 using ._ 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 allow for 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

The TRANSREG procedure has the ability to estimate missing data and monotonically transform variables while untying tied values. Estimates of ordinary missing values (.) may all 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.

```
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 dummy;
   model linear(y) = linear(x1 x2);
   output;
run;

proc print;
run;
```

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

Figure 65.7. Missing Values Example

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 option if you wish to define convergence only in terms of the other option. The criterion change can become negative when the data have converged so that 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 other 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 DUMMY *a-option*, divergence may be perfectly normal.

When there are no monotonicity constraints and there is only one canonical variable in each set, PROC TRANSREG (with the DUMMY *a-option*) can usually find the optimal solution in only one iteration. (There are no monotonicity constraints when the MONOTONE, MSPLINE, or UNTIE transformations and the UNTIE= and MONOTONE= *a-options* are not specified. 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 DUMMY *a-option* even if nothing changes in iteration 0. The MONOTONE, MSPLINE, and UNTIE variables are not transformed by the dummy variable initialization. Note that divergence with the DUMMY *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 DUMMY *a-option* can greatly decrease the amount of time required to find the optimal transformations. Furthermore, by solving for the transformations directly instead of iteratively, PROC TRANSREG avoids certain nonoptimal solutions.

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 code:

```
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.

```
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 may 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, a 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. As long as the dependent variable is not constant, PROC TRANSREG produces an iteration history table for all models, not just models in which the variables can change. When constant variables are expected and should not be zeroed, specify the NOZEROCONSTANT option.

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, both in terms of 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.

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

Passive Observations

Observations may 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, 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^2, 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-options*, $\mathbf{b}'$ and $\mathbf{R}$ are created from multiple regression coefficients. When coordinates are requested with the CPC, CEC, or CQC *o-options*, $\mathbf{b}'$ 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 standardizes 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.

```
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;
```

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

Figure 65.8. Redundancy Analysis Example

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.

These following 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 65.9:

```
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;
```

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

Figure 65.9. Redundancy Analysis Example

The following statements produce the coefficients for predicting the redundancy variables from the independent variables, shown in Figure 65.10:

```
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;
```

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

Figure 65.10. Redundancy Analysis Example

The following statements produce the coefficients for predicting the independent variables from the redundancy variables, shown in Figure 65.11:

```
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;
```

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

Figure 65.11. Redundancy Analysis Example

Optimal Scaling

An alternating least-squares optimal scaling algorithm can be divided into two major stages. The first 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, 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 using (possibly constrained) least squares. Many optimal scaling problems are solved without actually constructing design and projection matrices. The following 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 System collating sequence for missing and nonmissing values is used. Sorting simply allows constraints to be specified in terms of relations 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 datum forms a separate category for all other transformations (1 1 1 2 2 3 form six categories). Once 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, 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 method is applied to them also, independently of the other missing and nonmissing partitions. Once 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, 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 using least-squares regression coefficients. 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 in Table 65.5, and the natural piecewise polynomial spline basis is the right matrix. 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.

Table 65.5. Spline Bases

B-Spline Basis Piecewise Polynomial

0.171	0.557	0.250	0.022	0	0
0.037	0.447	0.443	0.073	0	0
0.001	0.251	0.576	0.172	0	0
0	0.093	0.572	0.334	0.001	0
0	0.020	0.437	0.517	0.027	0
0	0.001	0.253	0.623	0.123	0
0	0	0.108	0.557	0.332	0.003
0	0	0.032	0.341	0.548	0.079
0	0	0.004	0.109	0.523	0.364

Piecewise Polynomial Splines

1	1	1	1	0.000	0.000
1	2	4	8	0.000	0.000
1	3	9	27	0.000	0.000
1	4	16	64	0.125	0.000
1	5	25	125	3.375	0.000
1	6	36	216	15.625	0.000
1	7	49	343	42.875	0.125
1	8	64	512	91.125	3.375
1	9	81	729	166.375	15.625

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 nonzero elements in a row 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 ten, 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 five or ten 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 curve 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). 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.

```
proc transreg;
   model identity(Y)=spline(X / knots=1.5 2.4 3.0 3.0 4.0);
   output;
run;
```

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 ($\mathsf{X} > 3.0$). The two curves are not required to be continuous at 3.0.

```
proc transreg;
   model identity(y)=spline(x / knots=3 3 3 degree=2);
   output;
run;
```

The following statements categorize Y into 10 intervals and find a step-function transformation. 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.

```
proc transreg;
   model identity(Y)=spline(X / degree=0 nknots=9);
   output;
run;
```

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 comprise the 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

The TRANSREG procedure 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.

For two reasons 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 since 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 by-products 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), 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 transforma-

tions. 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 may 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 using 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 just implies 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 over-parameterization. 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. Use as few knots as possible to avoid this problem.

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 allowed 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 allowed to change to test the effect of variable x_j. This can be quite time consuming for a large model if the DUMMY *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 and is organized for direct input into the %PLOTIT macro or graphical or analysis procedures. For information on the %PLOTIT macro, see Appendix B, "Using the %PLOTIT Macro."

Output Data Set Examples

The next section provides a complete list of the contents of the OUT= data set. However, before presenting complete details, this section provides three brief examples, illustrating some typical output data sets.

The first example shows the output data set from a two-way ANOVA model. The following statements produce Figure 65.12:

```
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;
```

ANOVA Output Data Set Example

Obs	_TYPE_	_NAME_	Y	PY	RY	Intercept	X1a	X2a	X1a X2a	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 Ordered by Position-----

#	Variable	Type	Len	Pos	Label
1	_TYPE_	Char	8	56	
2	_NAME_	Char	32	64	
3	Y	Num	8	0	
4	PY	Num	8	8	Y Predicted Values
5	RY	Num	8	16	Y Residuals
6	Intercept	Num	8	24	Intercept
7	X1a	Num	8	32	X1 a
8	X2a	Num	8	40	X2 a
9	X1aX2a	Num	8	48	X1 a * X2 a
10	X1	Char	8	96	
11	X2	Char	8	104	

Figure 65.12. ANOVA Example Output Data Set Contents

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 con-

tains 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.

```
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;
```

These statements produce Figure 65.13.

Output Data Set for Curve Fitting Example

The CONTENTS Procedure

-----Variables Ordered by Position-----

#	Variable	Type	Len	Pos	Label
1	_TYPE_	Char	8	56	
2	_NAME_	Char	32	64	
3	Y	Num	8	0	
4	TY	Num	8	8	Y Transformation
5	PY	Num	8	16	Y Predicted Values
6	Intercept	Num	8	24	Intercept
7	X	Num	8	32	
8	TIntercept	Num	8	40	Intercept Transformation
9	TX	Num	8	48	X Transformation

Figure 65.13. Predicted Values Example Output Data Set Contents

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 next 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 dummy;
   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 65.14.

METHOD=MORALS Output Data Set Example

Obs	_DEPVAR_	_TYPE_	_NAME_	_DEPEND_	T_DEPEND_	P_DEPEND_	R_DEPEND_
1	Spline(Y1)	SCORE	a	11	13.1600	11.1554	2.00464
2	Spline(Y1)	SCORE	b	10	6.1931	6.8835	-0.69041
3	Spline(Y1)	SCORE	a	5	2.4467	4.7140	-2.26724
4	Spline(Y1)	SCORE	b	5	2.4467	0.4421	2.00464
5	Spline(Y1)	SCORE	c	4	4.2076	4.2076	-0.00000
6	Spline(Y1)	SCORE	b	3	5.5693	6.8835	-1.31422
7	Spline(Y1)	SCORE	a	1	4.9766	4.7140	0.26261
8	Spline(Y1)	M COEFFI	Y1	.	.	.	.
9	Spline(Y2)	SCORE	a	1	-0.5303	-0.5199	-0.01043
10	Spline(Y2)	SCORE	b	4	5.5487	4.5689	0.97988
11	Spline(Y2)	SCORE	a	2	3.8940	4.5575	-0.66347
12	Spline(Y2)	SCORE	b	9	9.6358	9.6462	-0.01043
13	Spline(Y2)	SCORE	c	3	5.6210	5.6210	0.00000
14	Spline(Y2)	SCORE	b	6	3.5994	4.5689	-0.96945
15	Spline(Y2)	SCORE	a	8	5.2314	4.5575	0.67390
16	Spline(Y2)	M COEFFI	Y2	.	.	.	.

Obs	Intercept	N1	N2	TIntercept	TN1	TN2	X1	X2
1	1	0	0	1.0000	0.06711	-0.09384	a	a
2	1	1	0	1.0000	1.51978	-0.09384	b	a
3	1	0	1	1.0000	0.06711	1.32038	a	b
4	1	1	1	1.0000	1.51978	1.32038	b	b
5	1	2	2	1.0000	0.23932	1.32038	c	c
6	1	1	0	1.0000	1.51978	-0.09384	b	a
7	1	0	1	1.0000	0.06711	1.32038	a	b
8	.	.	.	10.9253	-2.94071	-4.55475	Y1	Y1
9	1	0	0	1.0000	0.03739	-0.09384	a	a
10	1	1	0	1.0000	1.51395	-0.09384	b	a
11	1	0	1	1.0000	0.03739	1.32038	a	b
12	1	1	1	1.0000	1.51395	1.32038	b	b
13	1	2	2	1.0000	0.34598	1.32038	c	c
14	1	1	0	1.0000	1.51395	-0.09384	b	a
15	1	0	1	1.0000	0.03739	1.32038	a	b
16	.	.	.	-0.3119	3.44636	3.59024	Y2	Y2

Figure 65.14. METHOD=MORALS Rolled Output Data Set

METHOD=MORALS Output Data Set Example

The CONTENTS Procedure

-----Variables Ordered by Position-----

#	Variable	Type	Len	Pos	Label
1	_DEPVAR_	Char	42	80	Dependent Variable Transformation(Name)
2	_TYPE_	Char	8	122	
3	_NAME_	Char	32	130	
4	_DEPEND_	Num	8	0	Dependent Variable
5	T_DEPEND_	Num	8	8	Dependent Variable Transformation
6	P_DEPEND_	Num	8	16	Dependent Variable Predicted Values
7	R_DEPEND_	Num	8	24	Dependent Variable Residuals
8	Intercept	Num	8	32	Intercept
9	N1	Num	8	40	
10	N2	Num	8	48	
11	TIntercept	Num	8	56	Intercept Transformation
12	TN1	Num	8	64	N1 Transformation
13	TN2	Num	8	72	N2 Transformation
14	X1	Char	8	162	
15	X2	Char	8	170	

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 first 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

This section presents 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 65.6. 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, IAPREFIX=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) appear first. Then all information for the second dependent variable (scores then, optionally, coefficients) appear 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 possibly dropping extra characters.

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 3443).

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, Adj R-Sq, and Coeff Var. Whenever Value is not an underscore special missing value, Statistic describes the contents of Value.

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 ∼, <=, >=, 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 on 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 for accumulating 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.

Solving Standard Least-Squares Problems

This section illustrates how to solve some ordinary least-squares problems and generalizations of those problems by formulating them as transformation regression problems. One problem involves finding linear and nonlinear regression functions in a scatter plot. The next problem involves simultaneously fitting two lines or curves through a scatter plot. The last problem involves finding the overall fit of a multi-way main-effects and interactions analysis-of-variance model.

Nonlinear Regression Functions

This example uses PROC TRANSREG in simple regression to find the optimal regression line, a nonlinear but monotone regression function, and a nonlinear nonmonotone regression function. A regression line can be found by specifying

```
proc transreg;
   model identity(y) = identity(x);
   output predicted;
run;
```

A monotone regression function (in this case, a monotonically decreasing regression function, since the correlation coefficient is negative) can be found by requesting an MSPLINE transformation of the independent variable, as follows.

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

The monotonicity restriction can be relaxed by requesting a SPLINE transformation of the independent variable, as shown below.

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

In this example, it is not useful to plot the transformation TX, since TX is just an intermediate result used in finding a regression function through the original X and Y scatter plot.

The following statements provide a specific example of using the TRANSREG procedure for fitting nonlinear regression functions. These statements produce Figure 65.15 through Figure 65.18.

```
title 'Linear and Nonlinear Regression Functions';
*---Generate an Artificial Nonlinear Scatter Plot---;
*---SAS/IML Software is Required for this Example---;
proc iml;
   N   = 500;
   X   = (1:N)';
   X   = X/(N/200);
   Y   = -((X/50)-1.5)##2 + sin(X/8) + sqrt(X)/5 + 2*log(X) + cos(X);
   X   = X - X[:,];
   X   = -X / sqrt(X[##,]/(n-1));
   Y   = Y - Y[:,];
   Y   = Y / sqrt(Y[##,]/(n-1));
   all = Y || X;
   create outset from all;
   append       from all;
   quit;

data A;
   set outset(rename=(col1=Y col2=X));
   if Y<-2 then Y=-2 + ranuni(7654321)/2;
   X1=X; X2=X; X3=X; X4=X;
run;

*---Predicted Values for the Linear Regression Line---;
proc transreg data=A;
   title2 'A Linear Regression Line';
   model identity(Y)=identity(X);
   output out=A pprefix=L;
   id X1-X4;
run;
```

```
*---Predicted Values for the Monotone Regression Function---;
proc transreg data=A;
   title2 'A Monotone Regression Function';
   model identity(Y)=mspline(X / nknots=9);
   output out=A pprefix=M;
   id X1-X4 LY;
run;

*---Predicted Values for the Nonmonotone Regression Function---;
proc transreg data=A;
   title2 'A Nonmonotone Regression Function';
   model identity(Y)=spline(X / nknots=9);
   output out=A predicted;
   id X1-X4 LY MY;
run;

*---Plot the Results---;
goptions goutmode=replace nodisplay;
%let opts = haxis=axis2 vaxis=axis1 frame cframe=ligr;
* Depending on your goptions, these plot options may work better:
* %let opts = haxis=axis2 vaxis=axis1 frame;

proc gplot data=A;
   title;
   axis1 minor=none label=(angle=90 rotate=0)
         order=(-2 to 2 by 2);
   axis2 minor=none order=(-2 to 2 by 2);
   plot Y*X1=1         /         &opts name='tregnl1';
   plot Y*X2=1 LY*X2=2 / overlay &opts name='tregnl2';
   plot Y*X3=1 MY*X3=2 / overlay &opts name='tregnl3';
   plot Y*X4=1 PY*X4=2 / overlay &opts name='tregnl4';
   symbol1 color=blue   v=star i=none;
   symbol2 color=yellow v=none i=join;
   label X1 = 'Nonlinear Scatter Plot'
         X2 = 'Linear Regression, r**2 = 0.14580'
         X3 = 'Monotone Function, r**2 = 0.60576'
         X4 = 'Nonlinear Function, r**2 = 0.89634';
run; quit;

goptions display;
proc greplay nofs tc=sashelp.templt template=l2r2;
   igout gseg;
   treplay 1:tregnl1 2:tregnl3 3:tregnl2 4:tregnl4;
run; quit;
```

```
                Linear and Nonlinear Regression Functions
                         A Linear Regression Line

                           The TRANSREG Procedure

      TRANSREG Univariate Algorithm Iteration History for Identity(Y)

Iteration    Average    Maximum                Criterion
   Number     Change     Change    R-Square       Change    Note
-------------------------------------------------------------------------
        1    0.00000    0.00000     0.14580                 Converged

Algorithm converged.
```

Figure 65.15. A Linear Regression Line

```
                Linear and Nonlinear Regression Functions
                      A Monotone Regression Function

                           The TRANSREG Procedure

       TRANSREG MORALS Algorithm Iteration History for Identity(Y)

Iteration    Average    Maximum                Criterion
   Number     Change     Change    R-Square       Change    Note
-------------------------------------------------------------------------
        1    0.62131    1.34209     0.14580
        2    0.00000    0.00000     0.60576      0.45995    Converged

Algorithm converged.
```

Figure 65.16. A Monotone Regression Function

```
                Linear and Nonlinear Regression Functions
                    A Nonmonotone Regression Function

                           The TRANSREG Procedure

       TRANSREG MORALS Algorithm Iteration History for Identity(Y)

Iteration    Average    Maximum                Criterion
   Number     Change     Change    R-Square       Change    Note
-------------------------------------------------------------------------
        1    0.83948    2.78984     0.14580
        2    0.00000    0.00000     0.89634      0.75054    Converged

Algorithm converged.
```

Figure 65.17. A Nonmonotone Regression Function

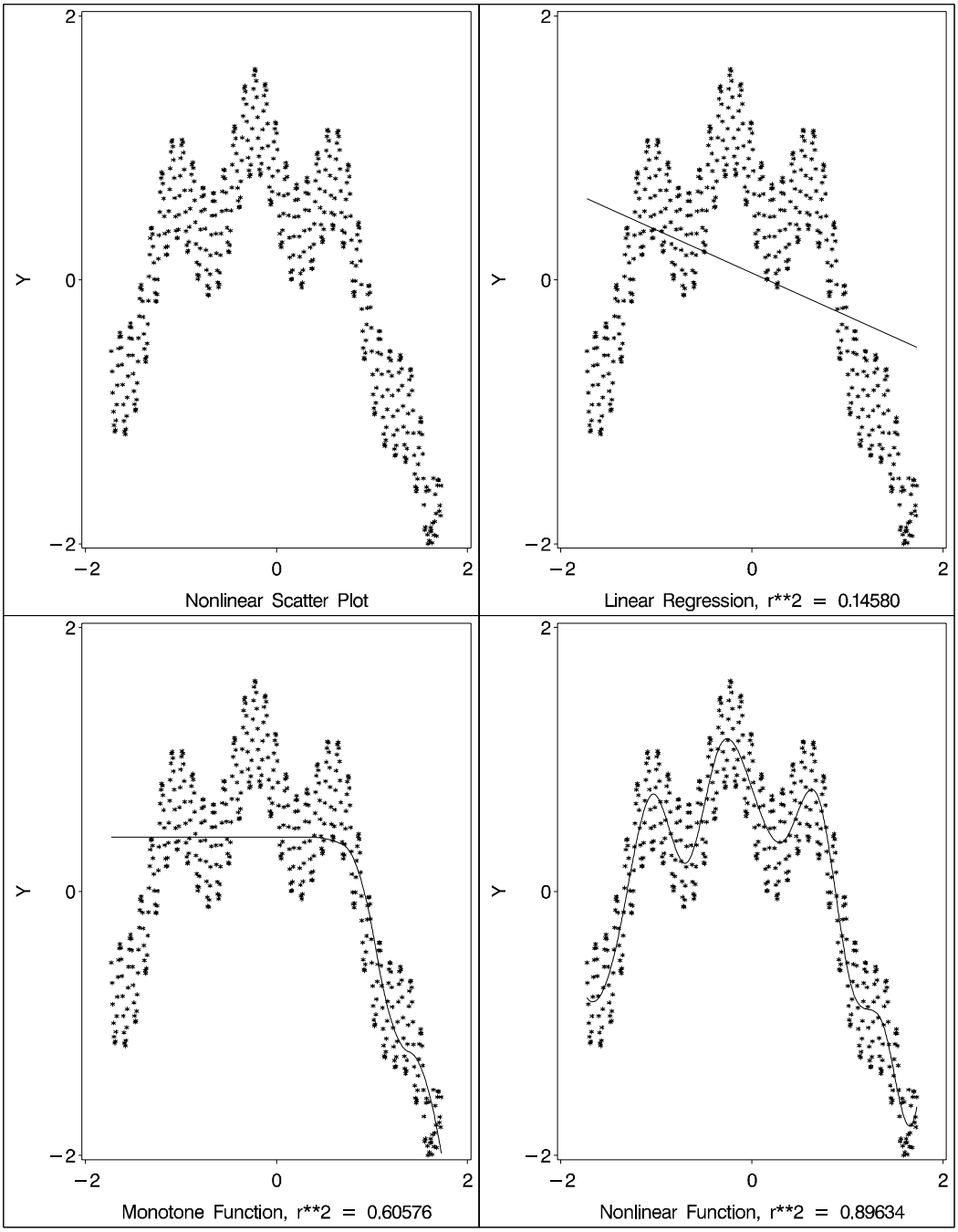


Figure 65.18. Linear, Monotone, and Nonmonotone Regression Functions

The squared correlation is only 0.15 for the linear regression, showing that 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) to 0.90 (nonmonotone). Relaxing the linearity constraint allows the regression function to bend and more closely follow the right portion of the scatter plot. Relaxing the monotonicity constraint allows 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 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 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^2 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^2 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^2 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. The data set has a continuous independent variable X, a continuous dependent variable Y, and a group membership variable G that has the value 1 for one group and 2 for the other group. The following code shows how PROC TRANSREG can be used to fit two lines, curves, and monotone curves simultaneously through a scatter plot. You can use this code with an appropriate number-list for the KNOTS= *t-option*.

```
proc transreg data=A dummy;
   title 'Parallel Lines, Separate Intercepts';
   model identity(Y)=class(G) identity(X);
   output predicted;
run;

proc transreg data=A;
   title 'Parallel Monotone Curves, Separate Intercepts';
   model identity(Y)=class(G) mspline(X / knots=-1.5 to 2.5 by 0.5);
   output predicted;
run;

proc transreg data=A dummy;
   title 'Parallel Curves, Separate Intercepts';
   model identity(Y)=class(G) spline(X / knots=-1.5 to 2.5 by 0.5);
   output predicted;
run;
```

```
proc transreg data=A;
   title 'Separate Slopes, Same Intercept';
   model identity(Y)=class(G / zero=none) * identity(X);
   output predicted;
run;

proc transreg data=A;
   title 'Separate Monotone Curves, Same Intercept';
   model identity(Y) = class(G / zero=none) *
                       mspline(X / knots=-1.5 to 2.5 by 0.5);
   output predicted;
run;

proc transreg data=A dummy;
   title 'Separate Curves, Same Intercept';
   model identity(Y) = class(G / zero=none) *
                       spline(X / knots=-1.5 to 2.5 by 0.5);
   output predicted;
run;

proc transreg data=A;
   title 'Separate Slopes, Separate Intercepts';
   model identity(Y) = class(G / zero=none) | identity(X);
   output predicted;
run;

proc transreg data=A;
   title 'Separate Monotone Curves, Separate Intercepts';
   model identity(Y) = class(G / zero=none) |
                       mspline(X / knots=-1.5 to 2.5 by 0.5);
   output predicted;
run;

proc transreg data=A dummy;
   title 'Separate Curves, Separate Intercepts';
   model identity(Y) = class(G / zero=none) |
                       spline(X / knots=-1.5 to 2.5 by 0.5);
   output predicted;
run;
```

Since the variables X1 and X2 both have a large partition of zeros, the KNOTS= *t-option* is specified instead of the NKNOTS= *t-option*. The following example generates an artificial data set with two curves. In the interest of space, only the preceding separate curves, separate intercepts example is run.

```
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;

proc transreg data=A dummy;
   model identity(Y) = class(G / zero=none) |
                       spline(X / knots=-1.5 to 2.5 by 0.5);
   output predicted;
run;

proc gplot;
   axis1 minor=none;
   axis2 minor=none label=(angle=90 rotate=0);
   symbol1 color=blue   v=star i=none;
   symbol2 color=yellow v=dot  i=none;
   plot Y*X=1 PY*X=2 /overlay frame cframe=ligr haxis=axis1
                      vaxis=axis2 href=0 vref=0;
run; quit;
```

The previous statements produce Figure 65.19 through Figure 65.20.

```
                  Separate Curves, Separate Intercepts

                        The TRANSREG Procedure

       TRANSREG MORALS Algorithm Iteration History for Identity(Y)

Iteration    Average    Maximum                Criterion
   Number     Change     Change    R-Square       Change    Note
-------------------------------------------------------------------------
        0    0.42724    4.48710     0.71020
        1    0.00000    0.00000     0.86604      0.15584    Converged

Algorithm converged.
```

Figure 65.19. Fitting Models: Separate Curves, Separate Intercepts

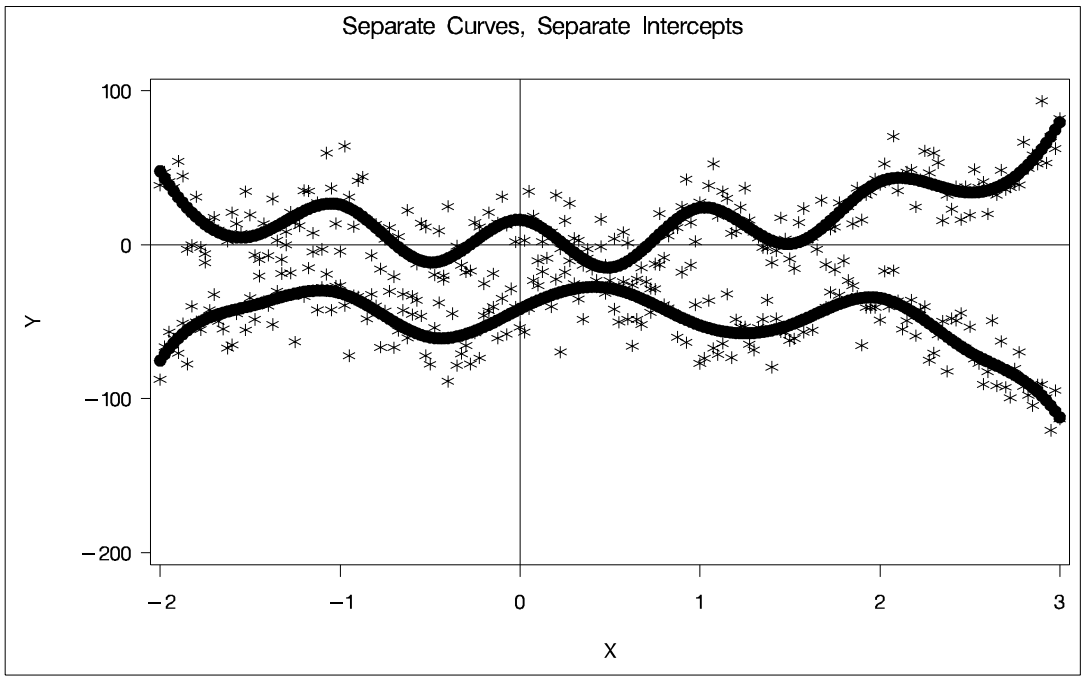


Figure 65.20. Plot for the Separate Curves, Separate Intercepts Example

Unbalanced ANOVA without Dummy Variables

This example 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

```
x=compress(c1||c2||c3||c4||c5||c6);
```

The following statements optimally score X (using the OPSCORE transformation) and do not transform Y. The final R^2 reported is the R^2 for the full analysis of variance model.

```
proc transreg;
   model identity(y)=opscore(x);
   output;
run;
```

The R^2 displayed by the preceding statements is the same as the R^2 that would be reported by both of the following PROC GLM runs.

```
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 an 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. The ANOVA and regression tables from PROC TRANSREG are displayed in Figure 65.21. The ANOVA and regression tables from PROC REG are displayed in Figure 65.22. The SHORT *a-option* is specified to suppress the iteration history.

```
proc transreg data=htex ss2 short;
   title 'Fit a Polynomial Regression Model with PROC TRANSREG';
   model identity(y) = spline(x1);
run;
```

```
            Fit a Polynomial Regression Model with PROC TRANSREG

                          The TRANSREG Procedure

 Identity(y)
 Algorithm converged.


          The TRANSREG Procedure Hypothesis Tests for Identity(y)


        Univariate ANOVA Table Based on the Usual Degrees of Freedom

                                  Sum of        Mean
    Source                DF     Squares      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

                                  Type II
                                   Sum of       Mean
 Variable        DF    Coefficient    Squares     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 65.21. ANOVA and Regression Output from PROC TRANSREG

```
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;
```

```
              Fit a Polynomial Regression Model with PROC REG

                              The REG Procedure
                                Model: MODEL1
                            Dependent Variable: y

                              Analysis of Variance

                                        Sum of           Mean
Source                    DF          Squares         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

                         Parameter       Standard
     Variable     DF       Estimate          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
```

Figure 65.22. ANOVA and Regression Output from PROC REG

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*s. PROC TRANSREG displays Type II sums of squares, mean squares, and *F*s. 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

$$SS_j = \hat{\beta}_j^2 \times MSE / s_{\beta_j}^2$$

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 polyno-

mial regression model. The output from PROC TRANSREG is in Figure 65.23. The output from PROC REG is in Figure 65.24.

```
proc transreg data=htex ss2 dummy;
   title 'Two-Variable Polynomial Regression';
   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;
   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;
```

```
                  Two-Variable Polynomial Regression

                        The TRANSREG Procedure

       TRANSREG MORALS Algorithm Iteration History for Identity(y)

Iteration    Average    Maximum                Criterion
   Number     Change     Change    R-Square       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    Average    Maximum                Criterion
   Number     Change     Change    R-Square       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    Average    Maximum                Criterion
   Number     Change     Change    R-Square       Change    Note
-------------------------------------------------------------------------
        0    0.45381    1.43736     0.00717
        1    0.00000    0.00000     0.02604      0.01886    Converged

Algorithm converged.
```

Figure 65.23. Two-Variable Polynomial Regression Output from PROC TRANSREG

Two-Variable Polynomial Regression

The TRANSREG Procedure

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

Two-Variable Polynomial Regression

The REG Procedure
Model: MODEL1
Dependent Variable: y

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		

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

Figure 65.24. Two-Variable Polynomial Regression Output from PROC REG

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		

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^2 of 0.17287 for the fit of the overall model. The second is for

```
model identity(y) = spline(x2);
```

which excludes SPLINE(X1). The third is for

```
model identity(y) = spline(x1);
```

which excludes SPLINE(X2). The difference between the first and second R^2 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^2 times the total sum of squares is the model sum of squares for SPLINE(X2)

$$(0.17287 - 0.02604) \times 224.143800 = 32.911247$$

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.

Hypothesis Tests with Monotonicity Constraints

Now consider a model with monotonicity constraints. This model has no counterpart in PROC REG.

```
proc transreg data=htex ss2 short;
   title 'Monotone Splines';
   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 using liberal degrees of freedom and one using conservative degrees of freedom. All sums of squares and the R^2s are the same for both tables. What differs are the degrees of freedom and statistics that are computed using 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 65.25.

Monotone Splines

The TRANSREG Procedure

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		

Figure 65.25. Monotone Spline Transformations

Monotone Splines

The TRANSREG Procedure

The TRANSREG Procedure Hypothesis Tests for Identity(y)

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

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 that 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^2. It may 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. The Wilks' Lambda, Pillai's Trace, and Hotelling-Lawley Trace statistics 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^2 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.

Here is an example with a dependent variable transformation.

```
proc transreg data=htex ss2 dummy short;
   title 'Transform Dependent and Independent Variables';
   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. See Figure 65.26.

Transform Dependent and Independent Variables

The TRANSREG Procedure

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 65.26. Transform Dependent and Independent Variables

Transform Dependent and Independent Variables

The TRANSREG Procedure

The TRANSREG Procedure Hypothesis Tests for Spline(y)

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 code illustrates this relationship. See Figure 65.27 through Figure 65.28 for the results.

```
data oneway;
   input y x $;
   datalines;
0 a
1 a
2 a
7 b
8 b
9 b
3 c
4 c
5 c
;

proc transreg ss2 data=oneway short;
   title 'Implicit Intercept Model';
   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;
```

Implicit Intercept Model

The TRANSREG Procedure

Identity(y)
Algorithm converged.

The TRANSREG Procedure Hypothesis Tests for Identity(y)

Univariate ANOVA Table Based on the Usual Degrees of Freedom

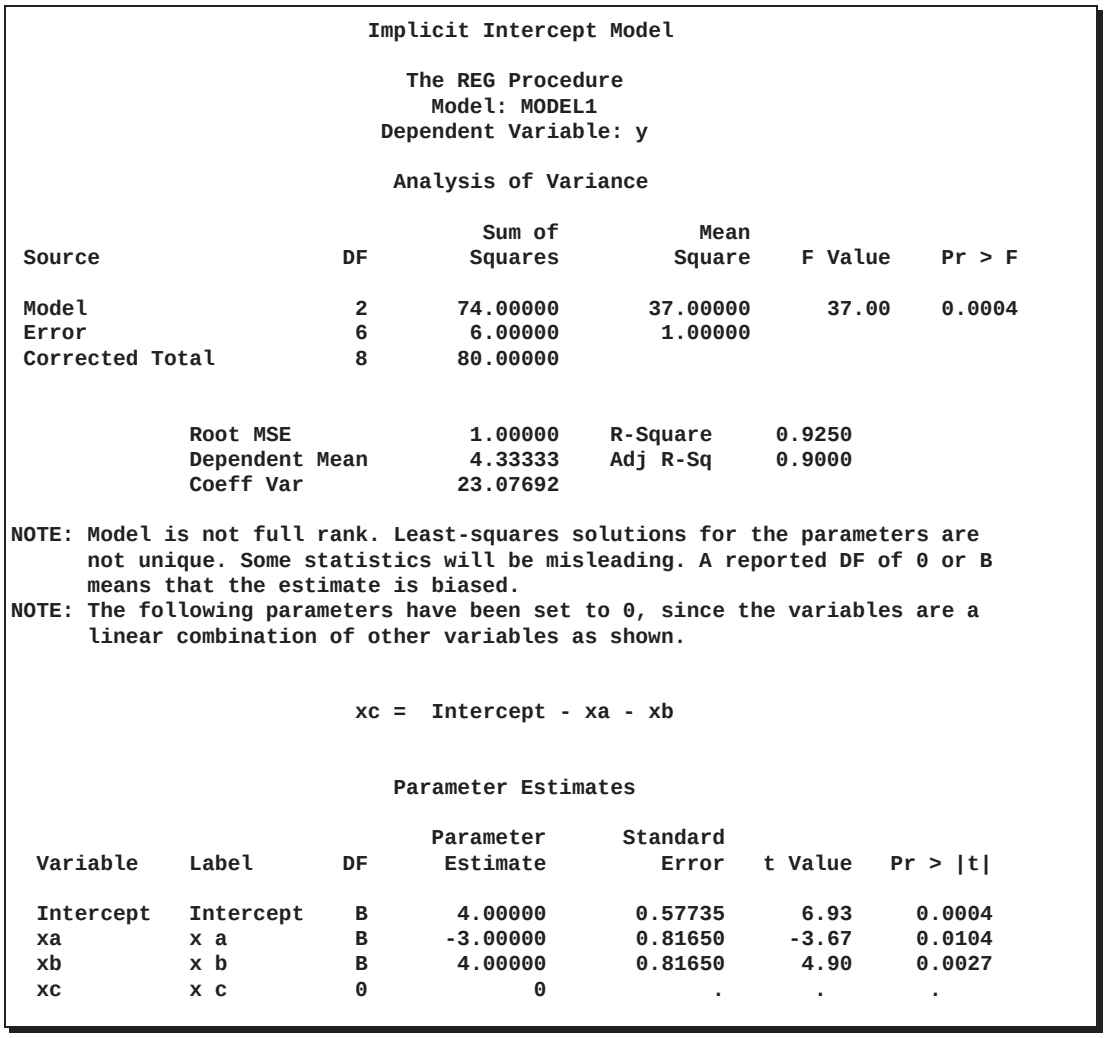
Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	2	74.00000	37.00000	37.00	0.0004
Error	6	6.00000	1.00000		
Corrected Total	8	80.00000			

Root MSE	1.00000	R-Square	0.9250
Dependent Mean	4.33333	Adj R-Sq	0.9000
Coeff Var	23.07692		

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

Figure 65.27. Implicit Intercept Model (TRANSREG Procedure)



Implicit Intercept Model

The REG Procedure
Model: MODEL1
Dependent Variable: y

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 65.28. Implicit Intercept Model (REG Procedure)

Implicit Intercept Model

The REG Procedure
Model: MODEL2
Dependent Variable: y

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. See Figure 65.29. 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 65.30. For comparison, the same analysis is also performed with PROC CATMOD. See Figure 65.31.

```
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;
run;

proc print;
   title2 'PROC TRANSREG Output Data Set';
run;

proc logistic;
   title2 'PROC LOGISTIC with Classification Variables';
   freq w;
   model y = &_trgind;
run;

proc catmod data=a;
   title2 'PROC CATMOD Should Produce the Same Results';
   model y = a b;
   weight w;
run;
```

Using PROC TRANSREG to Create a Design Matrix
PROC TRANSREG Output Data Set

Obs	_TYPE_	_NAME_	Intercept	a1	a2	a3	b1	b2	a	b	y	w
1	SCORE	1	1	1	0	0	1	0	1	1	1	12
2	SCORE	1	1	1	0	0	0	1	1	2	1	20
3	SCORE	1	1	1	0	0	-1	-1	1	3	1	14
4	SCORE	1	1	0	1	0	1	0	2	1	1	13
5	SCORE	1	1	0	1	0	0	1	2	2	1	20
6	SCORE	1	1	0	1	0	-1	-1	2	3	1	20
7	SCORE	1	1	0	0	1	1	0	3	1	1	16
8	SCORE	1	1	0	0	1	0	1	3	2	1	16
9	SCORE	1	1	0	0	1	-1	-1	3	3	1	11
10	SCORE	1	1	-1	-1	-1	1	0	4	1	1	11
11	SCORE	1	1	-1	-1	-1	0	1	4	2	1	19
12	SCORE	1	1	-1	-1	-1	-1	-1	4	3	1	16
13	SCORE	2	1	1	0	0	1	0	1	1	2	19
14	SCORE	2	1	1	0	0	0	1	1	2	2	11
15	SCORE	2	1	1	0	0	-1	-1	1	3	2	20
16	SCORE	2	1	0	1	0	1	0	2	1	2	13
17	SCORE	2	1	0	1	0	0	1	2	2	2	13
18	SCORE	2	1	0	1	0	-1	-1	2	3	2	17
19	SCORE	2	1	0	0	1	1	0	3	1	2	20
20	SCORE	2	1	0	0	1	0	1	3	2	2	13
21	SCORE	2	1	0	0	1	-1	-1	3	3	2	17
22	SCORE	2	1	-1	-1	-1	1	0	4	1	2	15
23	SCORE	2	1	-1	-1	-1	0	1	4	2	2	16
24	SCORE	2	1	-1	-1	-1	-1	-1	4	3	2	13

Figure 65.29. 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.DATA8
          Response Variable             y
          Number of Response Levels     2
          Number of Observations        24
          Frequency Variable            w
          Sum of Frequencies            375
          Link Function                 Logit
          Optimization Technique        Fisher's scoring


                        Response Profile

               Ordered                        Total
                 Value            y     Frequency

                     1            1           188
                     2            2           187


                    Model Convergence Status

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


                    Model Fit Statistics

                                          Intercept
                                 Intercept      and
             Criterion             Only     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
```

Figure 65.30. PROC LOGISTIC Output

Using PROC TRANSREG to Create a Design Matrix
PROC LOGISTIC with Classification Variables

The LOGISTIC Procedure

Analysis of Maximum Likelihood Estimates

Parameter	DF	Estimate	Standard Error	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

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

Using PROC TRANSREG to Create a Design Matrix
PROC CATMOD Should Produce the Same Results

The CATMOD Procedure

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

Figure 65.31. PROC CATMOD Output

Using PROC TRANSREG to Create a Design Matrix
PROC CATMOD Should Produce the Same Results

The CATMOD Procedure

Maximum Likelihood Analysis

Iteration	Sub Iteration	-2 Log Likelihood	Convergence Criterion
0	0	519.86039	1.0000
1	0	512.3792	0.0144
2	0	512.37786	2.608E-6
3	0	512.37786	9.929E-13

Maximum Likelihood Analysis

	Parameter Estimates					
Iteration	1	2	3	4	5	6
0	0	0	0	0	0	0
1	-0.001162	-0.0790	0.1965	-0.1327	-0.2365	0.3393
2	-0.000404	-0.0802	0.2001	-0.1350	-0.2392	0.3433
3	-0.000403	-0.0802	0.2001	-0.1350	-0.2392	0.3434

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

Analysis of Maximum Likelihood Estimates

Effect	Parameter	Estimate	Standard Error	Chi-Square	Pr > ChiSq
Intercept	1	-0.00040	0.1044	0.00	0.9969
a	2	-0.0802	0.1791	0.20	0.6542
	3	0.2001	0.1800	1.24	0.2662
	4	-0.1350	0.1819	0.55	0.4578
b	5	-0.2392	0.1500	2.54	0.1107
	6	0.3434	0.1474	5.42	0.0199

Choice Experiments: DESIGN, NORESTOREMISSING, NOZEROCONSTANT Usage

A 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 for use in the PHREG procedure to fit the choice model. In the interest of space, only the fifth choice set is displayed in Figure 65.32.

```
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;
```

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

Figure 65.32. The Fifth Choice Set

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 *a-option*.

The data set was coded by choice set (BY set;). This is a small problem, but with very large problems, it may 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 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.

ANOVA Codings

This set of examples illustrates several different ways to code the same two-way ANOVA model. Figure 65.33 displays the input data set.

```
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;
```

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

Figure 65.33. Input Data Set

The following statements fit a cell-means model. See Figure 65.34 and Figure 65.35.

```
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;
```

Two-way ANOVA Models
Cell-Means Model

The TRANSREG Procedure

Identity(y)
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	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

Figure 65.34. Cell-Means Model

The parameter estimates are

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

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

Figure 65.35. Cell-Means Model, Design Matrix

The following statements fit a reference cell model. The default reference level is the last cell (3,2). See Figure 65.36 and Figure 65.37.

```
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;
```

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

The TRANSREG Procedure

Identity(y)
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	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

Figure 65.36. Reference Cell Model, (3,2) Reference Cell

The parameter estimates are

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

The structural zeros are

$$\alpha_3 \equiv \beta_2 \equiv \gamma_{12} \equiv \gamma_{22} \equiv \gamma_{31} \equiv \gamma_{32} \equiv 0$$

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

Figure 65.37. Reference Cell Model, (3,2) Reference Cell, Design Matrix

The following statements fit a reference cell model, but this time the reference level is the first cell (1,1). See Figure 65.38 through Figure 65.39.

```
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;
```

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

The TRANSREG Procedure

Identity(y)
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	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

Figure 65.38. Reference Cell Model, (1,1) Reference Cell

The parameter estimates are

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

The structural zeros are

$$\alpha_1 \equiv \beta_1 \equiv \gamma_{11} \equiv \gamma_{12} \equiv \gamma_{21} \equiv \gamma_{31} \equiv 0$$

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

Figure 65.39. Reference Cell Model, (1,1) Reference Cell, Design Matrix

The following statements fit a deviations-from-means model. The default reference level is the last cell (3,2). This coding is also called effects coding. See Figure 65.40 and Figure 65.41.

```
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;
```

Two-way ANOVA Models
Deviations From Means, (3,2) Reference Cell

The TRANSREG Procedure

Identity(y)
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	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

Figure 65.40. Deviations-From-Means Model, (3,2) Reference Cell

The parameter estimates are

$$
\begin{aligned}
\hat{\mu} &= \overline{y} = 13.33333 \\
\hat{\alpha}_1 &= (\overline{y}_{11} + \overline{y}_{12})/2 - \overline{y} = (15 + 14)/2 - 13.33333 = 1.16667 \\
\hat{\alpha}_2 &= (\overline{y}_{21} + \overline{y}_{22})/2 - \overline{y} = (5 + 16)/2 - 13.33333 = -2.83333 \\
\hat{\beta}_1 &= (\overline{y}_{11} + \overline{y}_{21} + \overline{y}_{31})/3 - \overline{y} = (15 + 5 + 11)/3 - 13.33333 = -3 \\
\hat{\gamma}_{11} &= \overline{y}_{11} - (\overline{y} + \hat{\alpha}_1 + \hat{\beta}_1) = 15 - (13.33333 + 1.16667 + -3) = 3.5 \\
\hat{\gamma}_{21} &= \overline{y}_{21} - (\overline{y} + \hat{\alpha}_2 + \hat{\beta}_1) = 5 - (13.33333 + -2.83333 + -3) = -2.5
\end{aligned}
$$

The structural zeros are

$$\alpha_3 \equiv \beta_2 \equiv \gamma_{12} \equiv \gamma_{22} \equiv \gamma_{31} \equiv \gamma_{32} \equiv 0$$

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

Figure 65.41. Deviations-From-Means Model, (3,2) Reference Cell, Design Matrix

The following statements fit a deviations-from-means model, but this time the reference level is the first cell (1,1). This coding is also called effects coding. See Figure 65.42 through Figure 65.43.

```
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;
```

Two-way ANOVA Models
Deviations From Means, (1,1) Reference Cell

The TRANSREG Procedure

Identity(y)
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	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

Figure 65.42. Deviations-From-Means Model, (1,1) Reference Cell

The parameter estimates are

$$
\begin{aligned}
\hat{\mu} &= \overline{y} = 13.33333 \\
\hat{\alpha}_2 &= (\overline{y}_{21} + \overline{y}_{22})/2 - \overline{y} = (5+16)/2 - 13.33333 = -2.8333 \\
\hat{\alpha}_3 &= (\overline{y}_{31} + \overline{y}_{32})/2 - \overline{y} = (11+19)/2 - 13.33333 = 1.66667 \\
\hat{\beta}_2 &= (\overline{y}_{12} + \overline{y}_{22} + \overline{y}_{32})/3 - \overline{y} = (14+16+19)/3 - 13.33333 = 3 \\
\hat{\gamma}_{22} &= \overline{y}_{22} - (\overline{y} + \hat{\alpha}_2 + \hat{\beta}_2) = 16 - (13.33333 + -2.8333 + 3) = 2.5 \\
\hat{\gamma}_{32} &= \overline{y}_{32} - (\overline{y} + \hat{\alpha}_3 + \hat{\beta}_2) = 19 - (13.33333 + 1.66667 + 3) = 1
\end{aligned}
$$

The structural zeros are

$$\alpha_1 \equiv \beta_1 \equiv \gamma_{11} \equiv \gamma_{12} \equiv \gamma_{21} \equiv \gamma_{31} \equiv 0$$

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

Figure 65.43. Deviations-From-Means Model, (1,1) Reference Cell, Design Matrix

The following statements fit a less-than-full-rank model. The parameter estimates are constrained to sum to zero within each effect. See Figure 65.44 and Figure 65.45.

```
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;
```

```
                          Two-way ANOVA Models
                        Less Than Full Rank Model

                          The TRANSREG Procedure

  Identity(y)
  Algorithm converged.


          The TRANSREG Procedure Hypothesis Tests for Identity(y)


        Univariate ANOVA Table Based on the Usual Degrees of Freedom

                                  Sum of        Mean
    Source                DF     Squares      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

                              Type II
                               Sum of      Mean
  Variable      DF  Coefficient  Squares   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.
```

Figure 65.44. Less-Than-Full-Rank Model

The parameter estimates are

$$
\begin{aligned}
\hat{\mu} &= \overline{y} = 13.33333 \\
\hat{\alpha}_1 &= (\overline{y}_{11} + \overline{y}_{12})/2 - \overline{y} = (15 + 14)/2 - 13.33333 = 1.16667 \\
\hat{\alpha}_2 &= (\overline{y}_{21} + \overline{y}_{22})/2 - \overline{y} = (5 + 16)/2 - 13.33333 = -2.8333 \\
\hat{\alpha}_3 &= (\overline{y}_{31} + \overline{y}_{32})/2 - \overline{y} = (11 + 19)/2 - 13.33333 = 1.66667 \\
\hat{\beta}_1 &= (\overline{y}_{11} + \overline{y}_{21} + \overline{y}_{31})/3 - \overline{y} = (15 + 5 + 11)/3 - 13.33333 = -3 \\
\hat{\beta}_2 &= (\overline{y}_{12} + \overline{y}_{22} + \overline{y}_{32})/3 - \overline{y} = (14 + 16 + 19)/3 - 13.33333 = 3 \\
\hat{\gamma}_{11} &= \overline{y}_{11} - (\overline{y} + \hat{\alpha}_1 + \hat{\beta}_1) = 15 - (13.33333 + 1.16667 + -3) = 3.5
\end{aligned}
$$

$$\begin{aligned}
\hat{\gamma}_{12} &= \overline{y}_{12} - (\overline{y} + \hat{\alpha}_1 + \hat{\beta}_2) = 14 - (13.33333 + 1.16667 + 3) = -3.5 \\
\hat{\gamma}_{21} &= \overline{y}_{21} - (\overline{y} + \hat{\alpha}_2 + \hat{\beta}_1) = 5 - (13.33333 + -2.83333 + -3) = -2.5 \\
\hat{\gamma}_{22} &= \overline{y}_{22} - (\overline{y} + \hat{\alpha}_2 + \hat{\beta}_2) = 16 - (13.33333 + -2.8333 + 3) = 2.5 \\
\hat{\gamma}_{31} &= \overline{y}_{31} - (\overline{y} + \hat{\alpha}_3 + \hat{\beta}_1) = 11 - (13.33333 + 1.66667 + -3) = -1 \\
\hat{\gamma}_{32} &= \overline{y}_{32} - (\overline{y} + \hat{\alpha}_3 + \hat{\beta}_2) = 19 - (13.33333 + 1.66667 + 3) = 1
\end{aligned}$$

The constraints are

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

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

Two-way ANOVA Models
Less Than Full Rank Model

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

Obs	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	0	1	0	0	0	0	0	1	1
2	0	1	0	0	0	0	0	1	1
3	1	0	1	0	0	0	0	1	2
4	1	0	1	0	0	0	0	1	2
5	0	0	0	1	0	0	0	2	1
6	0	0	0	1	0	0	0	2	1
7	1	0	0	0	1	0	0	2	2
8	1	0	0	0	1	0	0	2	2
9	0	0	0	0	0	1	0	3	1
10	0	0	0	0	0	1	0	3	1
11	1	0	0	0	0	0	1	3	2
12	1	0	0	0	0	0	1	3	2

Figure 65.45. Less-Than-Full-Rank Model, Design Matrix

Displayed Output

The display options control the amount of displayed output. The displayed output can contain

- an iteration history and convergence status table, by default
- 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 are MONOTONE, UNTIE, or MSPLINE transformations 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 the following table.

For more information on ODS, see Chapter 15, "Using the Output Delivery System."

Table 65.7. ODS Tables Produced in PROC TRANSREG

ODS Table Name	Description	Statement	Option
ANOVA	ANOVA	MODEL/PROC	TEST/SS2
LiberalANOVA	ANOVA, *1	MODEL/PROC	TEST/SS2
ConservANOVA	ANOVA, *1	MODEL/PROC	TEST/SS2
FitStatistics	Fit statistics like R-square	MODEL/PROC	TEST/SS2
LiberalFitStatistics	Fit statistics, *1	MODEL/PROC	TEST/SS2
ConservFitStatistics	Fit statistics, *1	MODEL/PROC	TEST/SS2
MVANOVA	Multivariate ANOVA, *2	MODEL/PROC	TEST/SS2
LiberalMVANOVA	Multivariate ANOVA, *1, *2	MODEL/PROC	TEST/SS2
ConservMVANOVA	Multivariate ANOVA, *1, *2	MODEL/PROC	TEST/SS2
Coef	Regression results	MODEL/PROC	SS2
LiberalCoef	Regression results, *1	MODEL/PROC	SS2
ConservCoef	Regression results, *1	MODEL/PROC	SS2
MVCoef	Multivariate regression results, *2	MODEL/PROC	SS2
LiberalMVCoef	Multivariate regression results, *1, *2	MODEL/PROC	SS2

Table 65.7. (continued)

ODS Table Name	Description	Statement	Option
ConservMVCoef	Multivariate regression results, *1, *2	MODEL/PROC	SS2
Utilities	Conjoint Analysis Utilities	MODEL/PROC	UTILITY
LiberalUtilities	Conjoint Analysis Utilities, *1	MODEL/PROC	UTILITY
ConservUtilities	Conjoint Analysis Utilities, *1	MODEL/PROC	UTILITY
Equation	Linear Dependency Equation		less-than-full-rank model
Details	Model Details	MODEL/PROC	DETAIL
Univariate	Univariate Iteration History	MODEL/PROC	METHOD=UNIVARIATE
MORALS	MORALS Iteration History	MODEL/PROC	METHOD=MORALS
CANALS	CANALS Iteration History	MODEL/PROC	METHOD=CANALS
Redundancy	Redundancy Iteration History	MODEL/PROC	METHOD=REDUNDANCY
TestIterations	Hypothesis Test Iterations Iteration History	MODEL/PROC	SS2
ConvergenceStatus	Convergence Status		default

*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.

Examples

Example 65.1. Using Splines and Knots

This example illustrates some properties of splines. *Splines* are curves, which 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 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. Refer to Smith (1979) for an excellent introduction to splines.

In this example, an artificial data set is created with a variable Y that is a discontinuous function of X. See the first plot in Output 65.1.7. 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.

The first PROC TRANSREG analysis fits a linear regression model. The predicted values of Y given X are output and plotted to form the linear regression line. The R^2 for the linear regression is 0.10061, and it can be seen from the second plot in Output 65.1.7 that the linear regression model is not appropriate for these data. The

following statements create the data set and perform the first PROC TRANSREG analysis. These statements produce Output 65.1.1.

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

* Create in Y a discontinuous function of X.
*
* Store copies of X in V1-V7 for use in PROC GPLOT.
* These variables are only necessary so that each
* plot can have its own x-axis label while putting
* four plots on a page.;

data A;
   array V[7] V1-V7;
   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);
      do J=1 to 7;
         V[J]=X;
         end;
      output;
      end;
run;

* Each of the PROC TRANSREG steps fits a
* different spline model to the data set created
* previously.  The TRANSREG steps build up a data set with
* various regression functions.  All of the functions
* are then plotted with the final PROC GPLOT step.
*
* The OUTPUT statements add new predicted values
* variables to the data set, while the ID statements
* save all of the previously created variables that
* are needed for the plots.;

proc transreg data=A;
   model identity(Y) = identity(X);
   title2 'A Linear Regression Function';
   output out=A pprefix=Linear;
   id V1-V7;
run;
```

Output 65.1.1. Fitting a Linear Regression Model with PROC TRANSREG

```
                  An Illustration of Splines and Knots
                      A Linear Regression Function

                         The TRANSREG Procedure

     TRANSREG Univariate Algorithm Iteration History for Identity(Y)

Iteration    Average    Maximum                Criterion
   Number     Change     Change    R-Square       Change    Note
-------------------------------------------------------------------------
        1    0.00000    0.00000     0.10061                 Converged

Algorithm converged.
```

The second PROC TRANSREG analysis 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 R^2 increases from 0.10061, which is the linear fit value from before, to 0.40720. It can be seen from the third plot in Output 65.1.7 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, a degree three spline with no knots (not shown) increases R^2 by only a small amount. The following statements perform the quadratic analysis and produce Output 65.1.2.

```
proc transreg data=A;
   model identity(Y)=spline(X / degree=2);
   title2 'A Quadratic Polynomial Regression Function';
   output out=A pprefix=Quad;
   id V1-V7 LinearY;
run;
```

Output 65.1.2. Fitting a Quadratic Polynomial

```
                  An Illustration of Splines and Knots
                A Quadratic Polynomial Regression Function

                         The TRANSREG Procedure

       TRANSREG MORALS Algorithm Iteration History for Identity(Y)

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

Algorithm converged.
```

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 new R^2 is 0.61730, and it can be seen from the fourth plot (in Output 65.1.7) that the spline is less smooth than the quadratic polynomial and it follows the data more closely than the quadratic polynomial. The following statements perform this analysis and produce Output 65.1.3:

```
proc transreg data=A;
   model identity(Y) = spline(X / knots=5 10 15);
   title2 'A Cubic Spline Regression Function';
   title3 'The Third Derivative is Discontinuous at X=5, 10, 15';
   output out=A pprefix=Cub1;
   id V1-V7 LinearY QuadY;
run;
```

Output 65.1.3. Fitting a Piecewise Cubic Polynomial

```
                An Illustration of Splines and Knots
                 A Cubic Spline Regression Function
            The Third Derivative is Discontinuous at X=5, 10, 15

                          The TRANSREG Procedure

        TRANSREG MORALS Algorithm Iteration History for Identity(Y)

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

Algorithm converged.
```

The same model could be fit with a DATA step and PROC REG, as follows. (The output from the following code is not displayed.)

```
data B;            /* A is the data set used for 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;
```

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 required to be 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
- and another cubic curve for the portion of X greater than 15

The spline is continuous since there is not a separate constant in the formula for the spline for each knot. Now the R^2 is 0.95542, and the spline closely follows the data, except at the knots. The following statements perform this analysis and produce Output 65.1.4:

```
proc transreg data=A;
   model identity(y) = spline(x / knots=5 5 5 10 10 10 15 15 15);
   title3 'First - Third Derivatives Discontinuous at X=5, 10, 15';
   output out=A pprefix=Cub3;
   id V1-V7 LinearY QuadY Cub1Y;
run;
```

Output 65.1.4. Piecewise Polynomial with Discontinuous Derivatives

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

                          The TRANSREG Procedure

        TRANSREG MORALS Algorithm Iteration History for Identity(Y)

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

Algorithm converged.
```

The same model could be fit with a DATA step and PROC REG, as follows. (The output from the following code is not displayed.)

```
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;
```

When the knots are repeated four times in the next step, the spline function is discontinuous at the knots and follows the data even more closely, with an R^2 of 0.99254. In this step, each separate curve is approximated by a cubic polynomial (with no knots within the separate polynomials). The following statements perform this analysis and produce Output 65.1.5:

```
proc transreg data=A;
   model identity(Y) = spline(X / knots=5 5 5 5 10 10 10 10 15 15 15 15);
   title3 'Discontinuous Function and Derivatives';
   output out=A pprefix=Cub4;
   id V1-V7 LinearY QuadY Cub1Y Cub3Y;
run;
```

Output 65.1.5. Discontinuous Function and Derivatives

```
                An Illustration of Splines and Knots
                  A Cubic Spline Regression Function
                Discontinuous Function and Derivatives

                        The TRANSREG Procedure

       TRANSREG MORALS Algorithm Iteration History for Identity(Y)

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

Algorithm converged.
```

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 last two steps use the NKNOTS= *t-option* to specify the number of knots but not their location. NKNOTS=4 places knots at the quintiles while NKNOTS=9 places knots at the deciles. The spline and its first two derivatives are continuous. The R^2 values are 0.74450 and 0.95256. Even though the knots are placed in the wrong places, the spline can closely follow the data with NKNOTS=9. The following statements produce Output 65.1.6.

```
proc transreg data=A;
   model identity(Y) = spline(X / nknots=4);
   title3 'Four Knots';
   output out=A pprefix=Cub4k;
   id V1-V7 LinearY QuadY Cub1Y Cub3Y Cub4Y;
run;

proc transreg data=A;
   model identity(Y) = spline(X / nknots=9);
   title3 'Nine Knots';
   output out=A pprefix=Cub9k;
   id V1-V7 LinearY QuadY Cub1Y Cub3Y Cub4Y Cub4kY;
run;
```

Output 65.1.6. Specifying Number of Knots instead of Knot Location

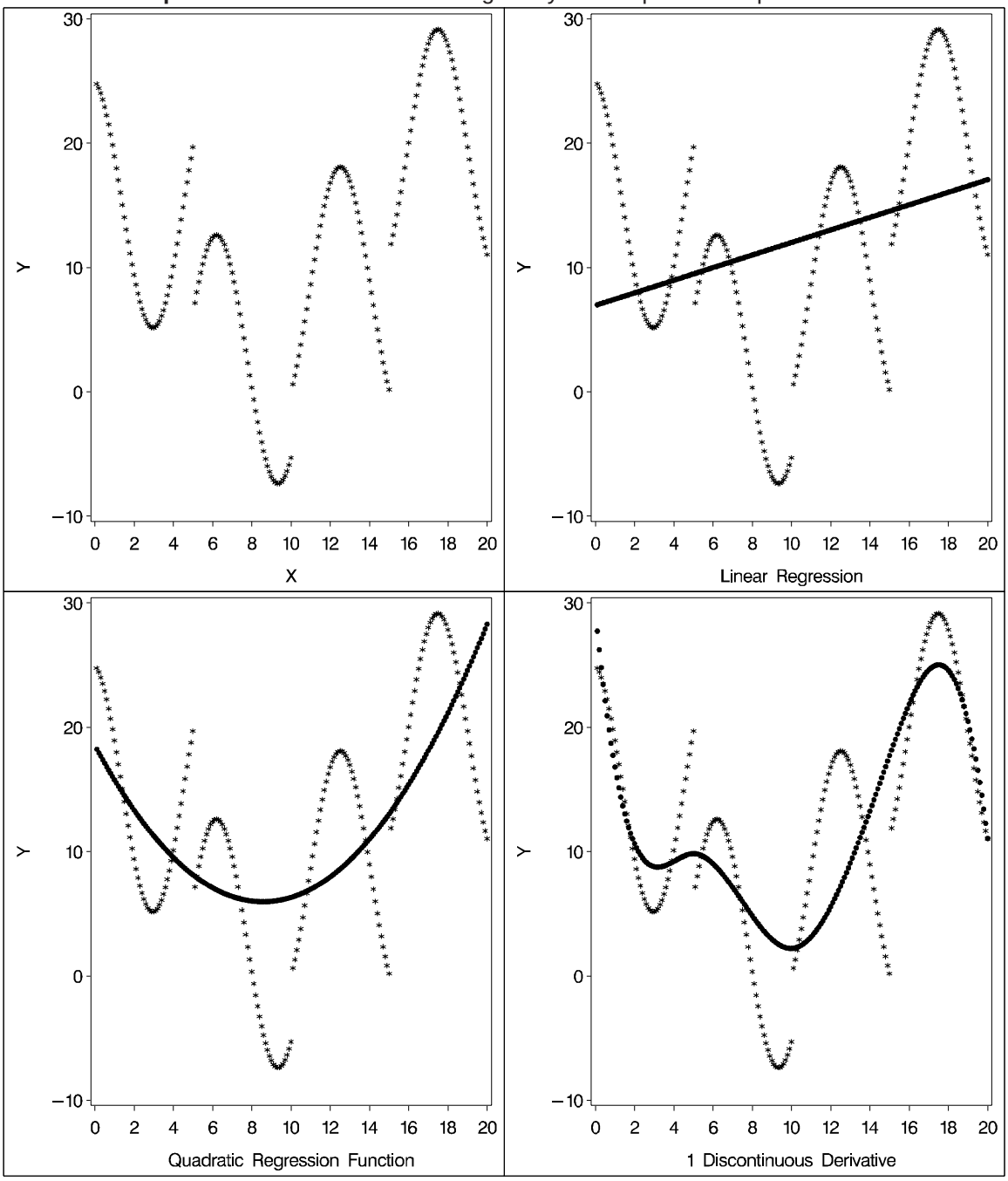
```
                  An Illustration of Splines and Knots
                   A Cubic Spline Regression Function
                               Four Knots

                         The TRANSREG Procedure

       TRANSREG MORALS Algorithm Iteration History for Identity(Y)

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

Algorithm converged.
```

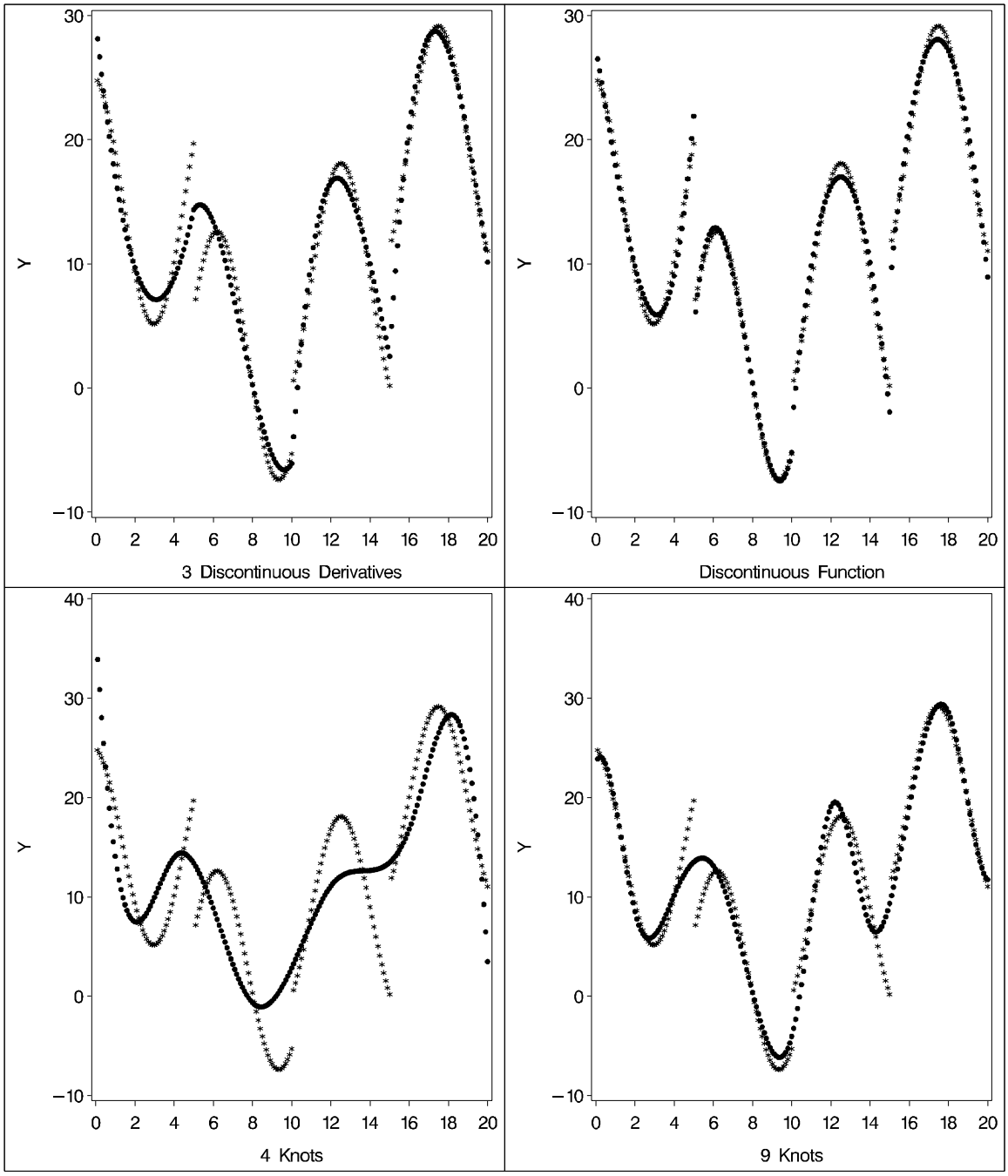
```
                  An Illustration of Splines and Knots
                   A Cubic Spline Regression Function
                               Nine Knots

                         The TRANSREG Procedure

       TRANSREG MORALS Algorithm Iteration History for Identity(Y)

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

Algorithm converged.
```

The following statements produce plots that show the data and fit at each step of the analysis. These statements produce Output 65.1.7.

```
goptions goutmode=replace nodisplay;
%let opts = haxis=axis2 vaxis=axis1 frame cframe=ligr;
* Depending on your goptions, these plot options may work better:
* %let opts = haxis=axis2 vaxis=axis1 frame;

proc gplot data=A;
   title;
   axis1 minor=none label=(angle=90 rotate=0);
   axis2 minor=none;
   plot Y*X=1              /       &opts name='tregdis1';
   plot Y*V1=1 linearY*X=2 /overlay &opts name='tregdis2';
   plot Y*V2=1 quadY  *X=2 /overlay &opts name='tregdis3';
   plot Y*V3=1 cub1Y  *X=2 /overlay &opts name='tregdis4';
   plot Y*V4=1 cub3Y  *X=2 /overlay &opts name='tregdis5';
   plot Y*V5=1 cub4Y  *X=2 /overlay &opts name='tregdis6';
   plot Y*V6=1 cub4kY *X=2 /overlay &opts name='tregdis7';
   plot Y*V7=1 cub9kY *X=2 /overlay &opts name='tregdis8';
   symbol1 color=blue   v=star i=none;
   symbol2 color=yellow v=dot  i=none;
   label V1      = 'Linear Regression'
         V2      = 'Quadratic Regression Function'
         V3      = '1 Discontinuous Derivative'
         V4      = '3 Discontinuous Derivatives'
         V5      = 'Discontinuous Function'
         V6      = '4 Knots'
         V7      = '9 Knots'
         Y       = 'Y' LinearY = 'Y' QuadY  = 'Y' Cub1Y  = 'Y'
         Cub3Y   = 'Y' Cub4Y   = 'Y' Cub4kY = 'Y' Cub9kY = 'Y';
run; quit;

goptions display;
proc greplay nofs tc=sashelp.templt template=l2r2;
   igout gseg;
   treplay 1:tregdis1 2:tregdis3 3:tregdis2 4:tregdis4;
   treplay 1:tregdis5 2:tregdis7 3:tregdis6 4:tregdis8;
run; quit;
```

Output 65.1.7. Plots Summarizing Analysis for Spline Example





Example 65.2. 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 prod-

uct 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)*, prices ($69.99, $74.99, $79.99), expected tread life (50,000, 60,000, 70,000), 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

$$20.64 = 9.50 + 1.90 + 5.87 + 2.41 + 0.96$$

The following statements produce Output 65.2.1:

```
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'
```

*In real conjoint experiments, real brand names are used.

```
                3 = '70,000';
   value HazardF
                1 = 'Yes'
                2 = 'No ';
run;

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 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 65.2.1. 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		

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).

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 65.3. 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 65.2, 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)*, prices ($69.99, $74.99, $79.99), expected tread life (50,000, 60,000, 70,000), 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*. While it is easy to get a design for this situation from ADX software or a table, you can also use the more general approach of using the OPTEX procedure to find an efficient design. First, the PLAN procedure is used to construct a full-factorial design consisting of all possible combinations of the factors. Then, PROC OPTEX is used to find an efficient design for a main-effects model.

*In real conjoint experiments, real brand names would be used

The FACTORS statement in PROC PLAN specifies each of the factors and the number of levels. The full-factorial design is output to the data set Candidates, and no displayed output is produced from PROC PLAN. The OPTEX procedure searches the Candidates data set for an efficient experimental design. The option CODING=ORTHCAN specifies an orthogonal coding of the internal design matrix. All factors are designated as CLASS variables, and a main-effects model (no interactions) is specified. The GENERATE statement requests a design with N=18 products using the Modified Federov algorithm. For most conjoint studies, this is the best algorithm to use. The best experimental design is output to a SAS data set called sasuser.TireDesign. For this study, PROC OPTEX has no trouble finding a perfect, 100% efficient experimental design because a standard, balanced, and orthogonal design exists for this problem. (It is frequently the case in practice that 100% efficiency is unobtainable.) Specifying random number seeds on the design procedures, while not strictly necessary, helps ensure that the design is reproducible. However, in examples like this in which PROC OPTEX finds many designs, all tied with the same efficiency, different but equivalent designs are sometimes output. When this happens, you get different results from those shown. The experimental design is displayed, and the SUMMARY procedure is used to examine one-way and two-way frequencies for all of the factors. All frequencies within each crosstabulation are constant, which is consistent with the 100% efficiency reported by PROC OPTEX. Finally, the tires are sorted into a random order and stored into a permanant SAS data set. In the interest of space, only the final design is shown. (The output from PROC OPTEX and PROC SUMMARY is not displayed.)

```
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 ';
run;

proc plan seed=070787;
   factors Brand=3 Price=3 Life=3 Hazard=2 / noprint;
   output out=Candidates;
run;
```

```
proc optex data=Candidates coding=orthcan seed=080489;
   class Brand Price Life Hazard;
   model Brand Price Life Hazard;
   generate n=18 method=m_federov;
   output out=TireDesign;
   format Brand BrandF9. Price PriceF9. Life LifeF6. Hazard HazardF3.;
run;

proc sort;
   by Brand Price Life Hazard;
run;

proc print;
run;

proc summary print;
   class Brand -- Hazard;
   ways 1 2;
run;

data TireDesign2; /* Randomize the order of the tires */
   set TireDesign;
   r = uniform(7);
run;

proc sort out=sasuser.TireDesign(drop=r);
   by r;
run;
```

Output 65.3.1. Tire Study, Experimental Design

Tire Study, Experimental Design

Obs	Brand	Price	Life	Hazard
1	Goodstone	$69.99	50,000	No
2	Goodstone	$69.99	50,000	No
3	Goodstone	$74.99	60,000	Yes
4	Goodstone	$74.99	60,000	Yes
5	Goodstone	$79.99	70,000	Yes
6	Goodstone	$79.99	70,000	No
7	Pirogi	$69.99	60,000	Yes
8	Pirogi	$69.99	60,000	No
9	Pirogi	$74.99	70,000	No
10	Pirogi	$74.99	70,000	No
11	Pirogi	$79.99	50,000	Yes
12	Pirogi	$79.99	50,000	Yes
13	Machismo	$69.99	70,000	Yes
14	Machismo	$69.99	70,000	Yes
15	Machismo	$74.99	50,000	Yes
16	Machismo	$74.99	50,000	No
17	Machismo	$79.99	60,000	No
18	Machismo	$79.99	60,000	No

Next, the questionnaires are printed, and subjects are given the questionnaires and are asked to rate the tires.

The following statements produce Output 65.3.2. This output is abbreviated; the statements produce stimuli for all combinations.

```
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;
```

Output 65.3.2. Conjoint Analysis, Stimuli Descriptions

```
                                                    Subject _________


1) 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


2) For your next tire purchase, how likely are you to buy this product?

   Goodstone brand tires at $69.99,
   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


3) For your next tire purchase, how likely are you to buy this product?

   Pirogi brand tires at $74.99,
   with a 70,000 tread life guarantee,
   and without road hazard insurance.

   Definitely Would                 Definitely Would
   Not Purchase                             Purchase

   1     2     3     4     5     6     7     8     9


4) For your next tire purchase, how likely are you to buy this product?

   Goodstone brand tires at $79.99,
   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
```

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 con-

structed 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 65.3.3:

```
title 'Tire Study, Data Entry, Preprocessing';

data Results;
   input (c1-c18) (1.);
   datalines;
366479338236695228
583448157149666228
127799316264575448
335869145193567449
366379238246685229
;

*---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;

*---Print Input Data Set---;
proc print;
   title2 'Data Set for Conjoint Analysis';
run;

*---Fit Each Subject Individually---;
proc transreg data=Both utilities short outtest=Utils separators='  ';
   ods select FitStatistics Utilities;
   title2 'Individual Conjoint Analyses';
   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 65.3.3. Conjoint Analysis

Tire Study, Data Entry, Preprocessing
Data Set for Conjoint Analysis

Obs	Brand	Price	Life	Hazard	Subj1	Subj2	Subj3	Subj4	Subj5
1	Machismo	$74.99	50,000	No	3	5	1	3	3
2	Goodstone	$69.99	50,000	No	6	8	2	3	6
3	Pirogi	$74.99	70,000	No	6	3	7	5	6
4	Goodstone	$79.99	70,000	Yes	4	4	7	8	3
5	Pirogi	$74.99	70,000	No	7	4	9	6	7
6	Machismo	$69.99	70,000	Yes	9	8	9	9	9
7	Pirogi	$79.99	50,000	Yes	3	1	3	1	2
8	Machismo	$74.99	50,000	Yes	3	5	1	4	3
9	Pirogi	$69.99	60,000	No	8	7	6	5	8
10	Pirogi	$79.99	50,000	Yes	2	1	2	1	2
11	Goodstone	$79.99	70,000	No	3	4	6	9	4
12	Goodstone	$69.99	50,000	No	6	9	4	3	6
13	Goodstone	$74.99	60,000	Yes	6	6	5	5	6
14	Pirogi	$69.99	60,000	Yes	9	6	7	6	8
15	Goodstone	$74.99	60,000	Yes	5	6	5	7	5
16	Machismo	$79.99	60,000	No	2	2	4	4	2
17	Machismo	$79.99	60,000	No	2	2	4	4	2
18	Machismo	$69.99	70,000	Yes	8	8	8	9	9

Tire Study, Data Entry, Preprocessing
Individual Conjoint Analyses

The TRANSREG Procedure

The TRANSREG Procedure Hypothesis Tests for Identity(Subj1)

Root MSE	0.49441	R-Square	0.9760
Dependent Mean	5.11111	Adj R-Sq	0.9592
Coeff Var	9.67330		

Utilities Table Based on the Usual Degrees of Freedom

Label		Utility	Standard Error	Importance (% Utility Range)	Variable
Intercept		5.1111	0.11653		Intercept
Brand	Goodstone	-0.1111	0.16480	14.286	Class.BrandGoodstone
Brand	Pirogi	0.7222	0.16480		Class.BrandPirogi
Brand	Machismo	-0.6111	0.16480		Class.BrandMachismo
Price	$69.99	2.5556	0.16480	53.571	Class.Price_69_99
Price	$74.99	-0.1111	0.16480		Class.Price_74_99
Price	$79.99	-2.4444	0.16480		Class.Price_79_99
Life	50,000	-1.2778	0.16480	25.000	Class.Life50_000
Life	60,000	0.2222	0.16480		Class.Life60_000
Life	70,000	1.0556	0.16480		Class.Life70_000
Hazard	Yes	0.3333	0.11653	7.143	Class.HazardYes
Hazard	No	-0.3333	0.11653		Class.HazardNo

Tire Study, Data Entry, Preprocessing
Individual Conjoint Analyses

The TRANSREG Procedure

The TRANSREG Procedure Hypothesis Tests for Identity(Subj2)

Root MSE	0.47140	R-Square	0.9792
Dependent Mean	4.94444	Adj R-Sq	0.9647
Coeff Var	9.53402		

Utilities Table Based on the Usual Degrees of Freedom

Label	Utility	Standard Error	Importance (% Utility Range)	Variable
Intercept	4.9444	0.11111		Intercept
Brand Goodstone	1.2222	0.15713	30.201	Class.BrandGoodstone
Brand Pirogi	-1.2778	0.15713		Class.BrandPirogi
Brand Machismo	0.0556	0.15713		Class.BrandMachismo
Price $69.99	2.7222	0.15713	64.430	Class.Price_69_99
Price $74.99	-0.1111	0.15713		Class.Price_74_99
Price $79.99	-2.6111	0.15713		Class.Price_79_99
Life 50,000	-0.1111	0.15713	4.027	Class.Life50_000
Life 60,000	-0.1111	0.15713		Class.Life60_000
Life 70,000	0.2222	0.15713		Class.Life70_000
Hazard Yes	0.0556	0.11111	1.342	Class.HazardYes
Hazard No	-0.0556	0.11111		Class.HazardNo

Tire Study, Data Entry, Preprocessing
Individual Conjoint Analyses

The TRANSREG Procedure

The TRANSREG Procedure Hypothesis Tests for Identity(Subj3)

Root MSE	0.80277	R-Square	0.9425
Dependent Mean	5.00000	Adj R-Sq	0.9022
Coeff Var	16.05546		

Utilities Table Based on the Usual Degrees of Freedom

Label	Utility	Standard Error	Importance (% Utility Range)	Variable
Intercept	5.0000	0.18922		Intercept
Brand Goodstone	-0.1667	0.26759	13.291	Class.BrandGoodstone
Brand Pirogi	0.6667	0.26759		Class.BrandPirogi
Brand Machismo	-0.5000	0.26759		Class.BrandMachismo
Price $69.99	1.0000	0.26759	18.987	Class.Price_69_99
Price $74.99	-0.3333	0.26759		Class.Price_74_99
Price $79.99	-0.6667	0.26759		Class.Price_79_99
Life 50,000	-2.8333	0.26759	62.658	Class.Life50_000
Life 60,000	0.1667	0.26759		Class.Life60_000
Life 70,000	2.6667	0.26759		Class.Life70_000
Hazard Yes	0.2222	0.18922	5.063	Class.HazardYes
Hazard No	-0.2222	0.18922		Class.HazardNo

Tire Study, Data Entry, Preprocessing
Individual Conjoint Analyses

The TRANSREG Procedure

The TRANSREG Procedure Hypothesis Tests for Identity(Subj4)

Root MSE	0.96032	R-Square	0.9160
Dependent Mean	5.11111	Adj R-Sq	0.8572
Coeff Var	18.78895		

Utilities Table Based on the Usual Degrees of Freedom

Label	Utility	Standard Error	Importance (% Utility Range)	Variable
Intercept	5.1111	0.22635		Intercept
Brand Goodstone	0.7222	0.32011	19.880	Class.BrandGoodstone
Brand Pirogi	-1.1111	0.32011		Class.BrandPirogi
Brand Machismo	0.3889	0.32011		Class.BrandMachismo
Price $69.99	0.7222	0.32011	14.458	Class.Price_69_99
Price $74.99	-0.1111	0.32011		Class.Price_74_99
Price $79.99	-0.6111	0.32011		Class.Price_79_99
Life 50,000	-2.6111	0.32011	56.024	Class.Life50_000
Life 60,000	0.0556	0.32011		Class.Life60_000
Life 70,000	2.5556	0.32011		Class.Life70_000
Hazard Yes	0.4444	0.22635	9.639	Class.HazardYes
Hazard No	-0.4444	0.22635		Class.HazardNo

Tire Study, Data Entry, Preprocessing
Individual Conjoint Analyses

The TRANSREG Procedure

The TRANSREG Procedure Hypothesis Tests for Identity(Subj5)

Root MSE	0.52705	R-Square	0.9740
Dependent Mean	5.05556	Adj R-Sq	0.9558
Coeff Var	10.42509		

Utilities Table Based on the Usual Degrees of Freedom

Label	Utility	Standard Error	Importance (% Utility Range)	Variable
Intercept	5.0556	0.12423		Intercept
Brand Goodstone	-0.0556	0.17568	9.259	Class.BrandGoodstone
Brand Pirogi	0.4444	0.17568		Class.BrandPirogi
Brand Machismo	-0.3889	0.17568		Class.BrandMachismo
Price $69.99	2.6111	0.17568	57.407	Class.Price_69_99
Price $74.99	-0.0556	0.17568		Class.Price_74_99
Price $79.99	-2.5556	0.17568		Class.Price_79_99
Life 50,000	-1.3889	0.17568	29.630	Class.Life50_000
Life 60,000	0.1111	0.17568		Class.Life60_000
Life 70,000	1.2778	0.17568		Class.Life70_000
Hazard Yes	0.1667	0.12423	3.704	Class.HazardYes
Hazard No	-0.1667	0.12423		Class.HazardNo

These following statements summarize the results. Three tables are displayed: 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.

```
*---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 Utilites---;
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 65.3.4. Summary of Conjoint Analysis Results

Tire Study, Data Entry, Preprocessing
Importance Values

Obs	Brand	Price	Life	Hazard
1	14.2857	53.5714	25.0000	7.14286
2	30.2013	64.4295	4.0268	1.34228
3	13.2911	18.9873	62.6582	5.06329
4	19.8795	14.4578	56.0241	9.63855
5	9.2593	57.4074	29.6296	3.70370

Tire Study, Data Entry, Preprocessing
Average Importance

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
Brand	5	17.3833946	8.1066973	9.2592593	30.2013423
Price	5	41.7707079	23.2500570	14.4578313	64.4295302
Life	5	35.4677599	23.9482430	4.0268456	62.6582278
Hazard	5	5.3781376	3.1802662	1.3422819	9.6385542

Tire Study, Data Entry, Preprocessing Utilities

Obs	Intercept	Brand Goodstone	Brand Pirogi	Brand Machismo	Price $69.99	Price $74.99
1	5.11111	-0.11111	0.72222	-0.61111	2.55556	-0.11111
2	4.94444	1.22222	-1.27778	0.05556	2.72222	-0.11111
3	5.00000	-0.16667	0.66667	-0.50000	1.00000	-0.33333
4	5.11111	0.72222	-1.11111	0.38889	0.72222	-0.11111
5	5.05556	-0.05556	0.44444	-0.38889	2.61111	-0.05556

Obs	Price $79.99	Life 50,000	Life 60,000	Life 70,000	Hazard Yes	Hazard No
1	-2.44444	-1.27778	0.22222	1.05556	0.33333	-0.33333
2	-2.61111	-0.11111	-0.11111	0.22222	0.05556	-0.05556
3	-0.66667	-2.83333	0.16667	2.66667	0.22222	-0.22222
4	-0.61111	-2.61111	0.05556	2.55556	0.44444	-0.44444
5	-2.55556	-1.38889	0.11111	1.27778	0.16667	-0.16667

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 closely by expected tread life. Brand and road hazard insurance are less important. Both Goodstone and Pirogi are the most preferred brands by some of the respondents. All respondents preferred a lower price over a higher price, a longer tread life, and road hazard insurance.

Example 65.4. Transformation Regression of Exhaust Emissions Data

In this example, the MORALS algorithm is applied to data 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).

The equivalence ratio and nitrogen oxide variables are continuous and numeric, so spline transformations of these variables are requested. Each spline is degree three with nine knots (one at each decile) in order to allow 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. The character variable Fuel is nominal, so it is designated as a classification variable. No monotonicity constraints are placed on any of the transformations. Observations with missing values are excluded with the NOMISS *a-option*.

The squared multiple correlation for the initial model is less than 0.25. PROC TRANSREG increases the R^2 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^2 is due to this transformation. The transformation of nitrogen oxide (TNOx) is something like a log transformation.

These results suggest the parametric model

$$\log(\text{NOX}) = b_0 + b_1 \times \text{EqRatio} + b_2 \times \text{EqRatio}^2 + b_3 \times \text{CpRatio} + \sum_j b_j \text{class}_j(\text{Fuel}) + \text{error} .$$

You can perform this analysis with PROC TRANSREG using the following MODEL statement:

```
model log(NOx)= psp(EqRatio / deg=2) identity(CpRatio)
                class(Fuel / zero=first);
```

The LOG transformation computes the natural log. The PSPLINE expansion expands EqRatio into a linear term, EqRatio, and a squared term, EqRatio^2. A linear transformation of CpRatio and a dummy variable expansion of Fuel is requested with the first level as the reference level. These should provide a good parametric operationalization of the optimal transformations. The final model has an R^2 of 0.91 (smaller than before since the model uses fewer degrees of freedom, but still quite good).

The following statements produce Output 65.4.1 through Output 65.4.3:

```
title 'Gasoline Example';

data Gas;
   input Fuel :$8. CpRatio EqRatio NOx @@;
   label Fuel    = 'Fuel'
         CpRatio = 'Compression Ratio (CR)'
         EqRatio = 'Equivalence Ratio (PHI)'
         NOx     = 'Nitrogen Oxide (NOx)';
   datalines;
Ethanol  12.0 0.907 3.741 Ethanol  12.0 0.761 2.295
Ethanol  12.0 1.108 1.498 Ethanol  12.0 1.016 2.881
Ethanol  12.0 1.189 0.760 Ethanol   9.0 1.001 3.120
Ethanol   9.0 1.231 0.638 Ethanol   9.0 1.123 1.170
Ethanol  12.0 1.042 2.358 Ethanol  12.0 1.215 0.606
Ethanol  12.0 0.930 3.669 Ethanol  12.0 1.152 1.000
Ethanol  15.0 1.138 0.981 Ethanol  18.0 0.601 1.192
Ethanol   7.5 0.696 0.926 Ethanol  12.0 0.686 1.590
Ethanol  12.0 1.072 1.806 Ethanol  15.0 1.074 1.962
Ethanol  15.0 0.934 4.028 Ethanol   9.0 0.808 3.148
Ethanol   9.0 1.071 1.836 Ethanol   7.5 1.009 2.845
Ethanol   7.5 1.142 1.013 Ethanol  18.0 1.229 0.414
Ethanol  18.0 1.175 0.812 Ethanol  15.0 0.568 0.374
Ethanol  15.0 0.977 3.623 Ethanol   7.5 0.767 1.869
Ethanol   7.5 1.006 2.836 Ethanol   9.0 0.893 3.567
Ethanol  15.0 1.152 0.866 Ethanol  15.0 0.693 1.369
```

```
Ethanol  15.0 1.232 0.542 Ethanol  15.0 1.036 2.739
Ethanol  15.0 1.125 1.200 Ethanol   9.0 1.081 1.719
Ethanol   9.0 0.868 3.423 Ethanol   7.5 0.762 1.634
Ethanol   7.5 1.144 1.021 Ethanol   7.5 1.045 2.157
Ethanol  18.0 0.797 3.361 Ethanol  18.0 1.115 1.390
Ethanol  18.0 1.070 1.947 Ethanol  18.0 1.219 0.962
Ethanol   9.0 0.637 0.571 Ethanol   9.0 0.733 2.219
Ethanol   9.0 0.715 1.419 Ethanol   9.0 0.872 3.519
Ethanol   7.5 0.765 1.732 Ethanol   7.5 0.878 3.206
Ethanol   7.5 0.811 2.471 Ethanol  15.0 0.676 1.777
Ethanol  18.0 1.045 2.571 Ethanol  18.0 0.968 3.952
Ethanol  15.0 0.846 3.931 Ethanol  15.0 0.684 1.587
Ethanol   7.5 0.729 1.397 Ethanol   7.5 0.911 3.536
Ethanol   7.5 0.808 2.202 Ethanol   7.5 1.168 0.756
Indolene  7.5 0.831 4.818 Indolene  7.5 1.045 2.849
Indolene  7.5 1.021 3.275 Indolene  7.5 0.970 4.691
Indolene  7.5 0.825 4.255 Indolene  7.5 0.891 5.064
Indolene  7.5 0.710 2.118 Indolene  7.5 0.801 4.602
Indolene  7.5 1.074 2.286 Indolene  7.5 1.148 0.970
Indolene  7.5 1.000 3.965 Indolene  7.5 0.928 5.344
Indolene  7.5 0.767 3.834 Ethanol   7.5 0.749 1.620
Ethanol   7.5 0.892 3.656 Ethanol   7.5 1.002 2.964
82rongas  7.5 0.873 6.021 82rongas  7.5 0.987 4.467
82rongas  7.5 1.030 3.046 82rongas  7.5 1.101 1.596
82rongas  7.5 1.173 0.835 82rongas  7.5 0.931 5.498
82rongas  7.5 0.822 5.470 82rongas  7.5 0.749 4.084
82rongas  7.5 0.625 0.716 94%Eth    7.5 0.818 2.382
94%Eth    7.5 1.128 1.004 94%Eth    7.5 1.191 0.623
94%Eth    7.5 1.132 1.030 94%Eth    7.5 0.993 2.593
94%Eth    7.5 0.866 2.699 94%Eth    7.5 0.910 3.177
94%Eth   12.0 1.139 1.151 94%Eth   12.0 1.267 0.474
94%Eth   12.0 1.017 2.814 94%Eth   12.0 0.954 3.308
94%Eth   12.0 0.861 3.031 94%Eth   12.0 1.034 2.537
94%Eth   12.0 0.781 2.403 94%Eth   12.0 1.058 2.412
94%Eth   12.0 0.884 2.452 94%Eth   12.0 0.766 1.857
94%Eth    7.5 1.193 0.657 94%Eth    7.5 0.885 2.969
94%Eth    7.5 0.915 2.670 Ethanol  18.0 0.812 3.760
Ethanol  18.0 1.230 0.672 Ethanol  18.0 0.804 3.677
Ethanol  18.0 0.712  .    Ethanol  12.0 0.813 3.517
Ethanol  12.0 1.002 3.290 Ethanol   9.0 0.696 1.139
Ethanol   9.0 1.199 0.727 Ethanol   9.0 1.030 2.581
Ethanol  15.0 0.602 0.923 Ethanol  15.0 0.694 1.527
Ethanol  15.0 0.816 3.388 Ethanol  15.0 0.896  .
Ethanol  15.0 1.037 2.085 Ethanol  15.0 1.181 0.966
Ethanol   7.5 0.899 3.488 Ethanol   7.5 1.227 0.754
Indolene  7.5 0.701 1.990 Indolene  7.5 0.807 5.199
Indolene  7.5 0.902 5.283 Indolene  7.5 0.997 3.752
Indolene  7.5 1.224 0.537 Indolene  7.5 1.089 1.640
Ethanol   9.0 1.180 0.797 Ethanol   7.5 0.795 2.064
Ethanol  18.0 0.990 3.732 Ethanol  18.0 1.201 0.586
Methanol  7.5 0.975 2.941 Methanol  7.5 1.089 1.467
Methanol  7.5 1.150 0.934 Methanol  7.5 1.212 0.722
Methanol  7.5 0.859 2.397 Methanol  7.5 0.751 1.461
Methanol  7.5 0.720 1.235 Methanol  7.5 1.090 1.347
```

```
Methanol  7.5 0.616 0.344 Gasohol   7.5 0.712 2.209
Gasohol   7.5 0.771 4.497 Gasohol   7.5 0.959 4.958
Gasohol   7.5 1.042 2.723 Gasohol   7.5 1.125 1.244
Gasohol   7.5 1.097 1.562 Gasohol   7.5 0.984 4.468
Gasohol   7.5 0.928 5.307 Gasohol   7.5 0.889 5.425
Gasohol   7.5 0.827 5.330 Gasohol   7.5 0.674 1.448
Gasohol   7.5 1.031 3.164 Methanol  7.5 0.871 3.113
Methanol  7.5 1.026 2.551 Methanol  7.5 0.598 0.204
Indolene  7.5 0.973 5.055 Indolene  7.5 0.980 4.937
Indolene  7.5 0.665 1.561 Ethanol   7.5 0.629 0.561
Ethanol   9.0 0.608 0.563 Ethanol  12.0 0.584 0.678
Ethanol  15.0 0.562 0.370 Ethanol  18.0 0.535 0.530
94%Eth    7.5 0.674 0.900 Gasohol   7.5 0.645 1.207
Ethanol  18.0 0.655 1.900 94%Eth    7.5 1.022 2.787
94%Eth    7.5 0.790 2.645 94%Eth    7.5 0.720 1.475
94%Eth    7.5 1.075 2.147
;

*---Fit the Nonparametric Model---;
proc transreg data=Gas dummy test nomiss;
   model spline(NOx / nknots=9)=spline(EqRatio / nknots=9)
                       opscore(CpRatio) class(Fuel / zero=first);
   title2 'Iteratively Estimate NOx, CPRATIO and EQRATIO';
   output out=Results;
run;

*---Plot the Results---;
goptions goutmode=replace nodisplay;
%let opts = haxis=axis2 vaxis=axis1 frame cframe=ligr;
* Depending on your goptions, these plot options may work better:
* %let opts = haxis=axis2 vaxis=axis1 frame;

proc gplot data=Results;
   title;
   axis1 minor=none label=(angle=90 rotate=0);
   axis2 minor=none;
   symbol1 color=blue v=dot i=none;
   plot TCpRatio*CpRatio / &opts name='tregex1';
   plot TEqRatio*EqRatio / &opts name='tregex2';
   plot TNOx*NOx         / &opts name='tregex3';
run; quit;

goptions display;
proc greplay nofs tc=sashelp.templt template=l2r2;
   igout gseg;
   treplay 1:tregex1 2:tregex3 3:tregex2;
run; quit;
```

```
*-Fit the Parametric Model Suggested by the Nonparametric Analysis-;
proc transreg data=Gas dummy ss2 short nomiss;
   title 'Gasoline Example';
   title2 'Now fit log(NOx) = b0 + b1*EqRatio + b2*EqRatio**2 +';
   title3 'b3*CpRatio + Sum b(j)*Fuel(j) + Error';
   model log(NOx)= pspline(EqRatio / deg=2) identity(CpRatio)
                   class(Fuel / zero=first);
   output out=Results2;
run;
```

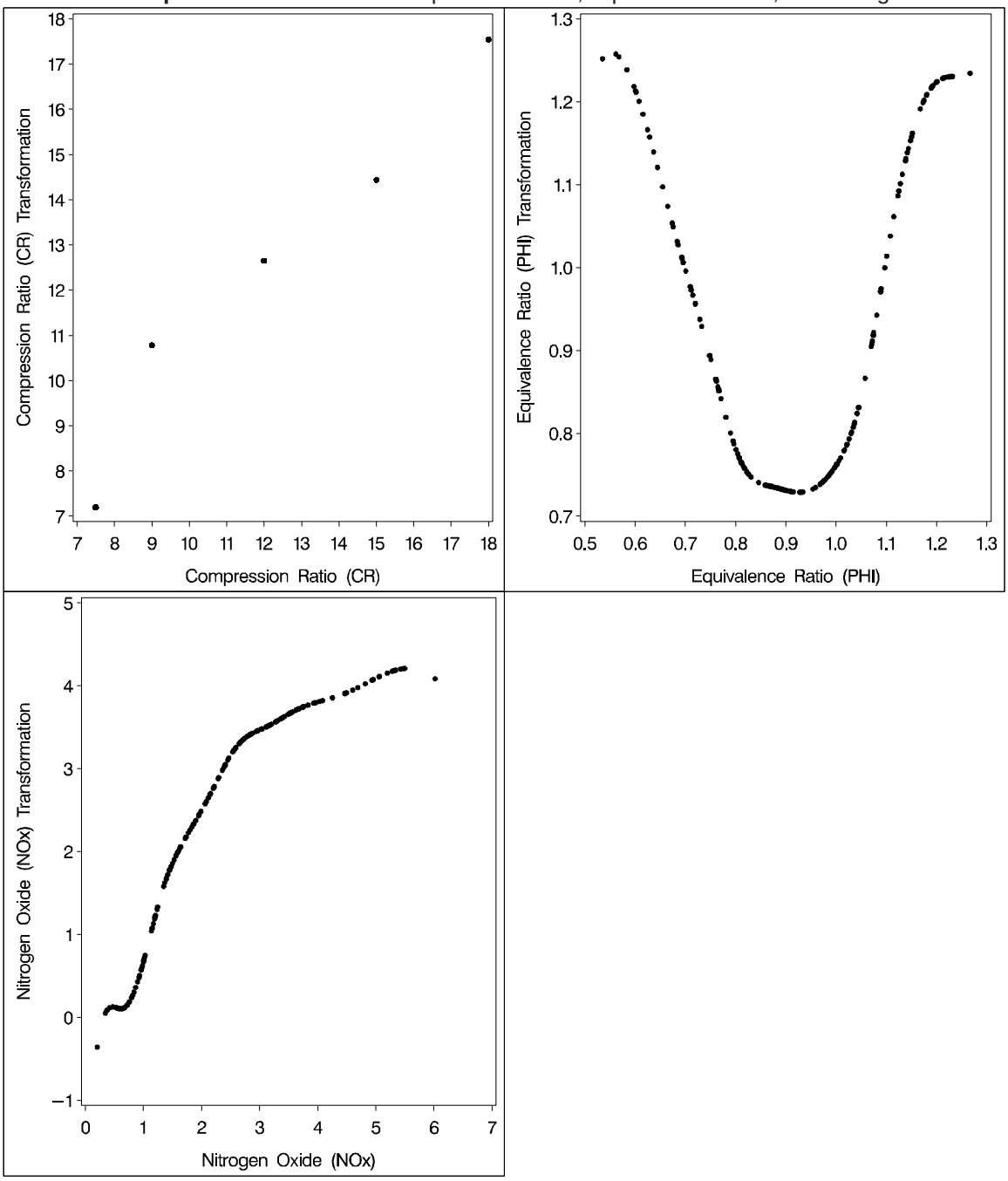
Output 65.4.1. Transformation Regression Example: The Nonparametric Model

```
                          Gasoline Example
                Iteratively Estimate NOx, CPRATIO and EQRATIO

                          The TRANSREG Procedure

         TRANSREG MORALS Algorithm Iteration History for Spline(NOx)

   Iteration    Average    Maximum                Criterion
      Number     Change     Change    R-Square       Change    Note
   ----------------------------------------------------------------------
           0    0.48074    3.86778     0.24597
           1    0.00000    0.00000     0.95865      0.71267    Converged

   Algorithm converged.


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


          Univariate ANOVA Table Based on the Usual Degrees of Freedom

                                  Sum of        Mean
    Source                 DF     Squares      Square    F Value    Liberal p

    Model                  21    326.0946    15.52831     162.27    >= <.0001
    Error                 147     14.0674     0.09570
    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           0.30935    R-Square    0.9586
                Dependent Mean     2.34593    Adj R-Sq    0.9527
                Coeff Var         13.18661


    Adjusted Multivariate ANOVA Table Based on the Usual Degrees of Freedom

          Dependent Variable Scoring Parameters=12   S=12   M=4   N=67

 Statistic                     Value    F Value    Num DF    Den DF            p

 Wilks' Lambda              0.041355       2.05       252      1455    <= <.0001
 Pillai's Trace             0.958645       0.61       252      1764    <= 1.0000
 Hotelling-Lawley Trace     23.18089      12.35       252    945.01    <= <.0001
 Roy's Greatest Root        23.18089     162.27        21       147    >= <.0001

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.
```

Output 65.4.2. Transformation Regression Example: The Parametric Model

```
                          Gasoline Example
            Now fit log(NOx) = b0 + b1*EqRatio + b2*EqRatio**2 +
                   b3*CpRatio + Sum b(j)*Fuel(j) + Error

                         The TRANSREG Procedure

  Log(NOx)
  Algorithm converged.


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


        Univariate ANOVA Table Based on the Usual Degrees of Freedom

                                  Sum of        Mean
    Source                DF     Squares      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

                                  Type II
                                   Sum of    Mean
Variable            DF Coefficient Squares  Square F Value Pr > F Label

Intercept            1  -14.586532 49.9469 49.9469 1073.18 <.0001 Intercept
Pspline.EqRatio_1    1   35.102914 62.7478 62.7478 1348.22 <.0001 Equivalence
                                                                  Ratio (PHI) 1
Pspline.EqRatio_2    1  -19.386468 64.6430 64.6430 1388.94 <.0001 Equivalence
                                                                  Ratio (PHI) 2
Identity(CpRatio)    1    0.032058  1.4445  1.4445   31.04 <.0001 Compression
                                                                  Ratio (CR)
Class.Fuel94_Eth     1   -0.449583  1.3158  1.3158   28.27 <.0001 Fuel 94%Eth
Class.FuelEthanol    1   -0.414242  1.2560  1.2560   26.99 <.0001 Fuel Ethanol
Class.FuelGasohol    1   -0.016719  0.0015  0.0015    0.03 0.8584 Fuel Gasohol
Class.FuelIndolene   1    0.001572  0.0000  0.0000    0.00 0.9853 Fuel Indolene
Class.FuelMethanol   1   -0.580133  1.7219  1.7219   37.00 <.0001 Fuel Methanol
```

Output 65.4.3. Plots of Compression Ratio, Equivalence Ratio, and Nitrogen Oxide



Example 65.5. Preference Mapping of Cars Data

This example uses PROC TRANSREG to perform a preference mapping (PREFMAP) analysis (Carroll 1972) of car 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 zero (very weak preference) to nine (very strong preference) scale. 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 one (bad) to five (good) scale. 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.

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 cars differ on that attribute. Refer to Carroll (1972) for more information. The Prin1 and Prin2 columns of the TResult1 OUT= data set contain the car 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 IDENTIY, 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 cars do not exist in the data set. The ideal points for MPG and Reliability are far removed from the plot of the cars. 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. Refer to 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).

The final DATA step combines the two output data sets and creates a data set suitable for the %PLOTIT macro. (For information on the %PLOTIT macro, see Appendix B, "Using the %PLOTIT Macro.") The plot contains one point per car and one point for each of the three ratings. The %PLOTIT macro options specify the input data set, how to handle anti-ideal points (described later), and where to draw horizontal and vertical reference lines. The DATATYPE= option specifies that the input data set contains results of a PREFMAP vector model and a PREFMAP ideal point model. This instructs the macro to draw vectors to _Type_='M COEFFI' observations and circles around _Type_='M POINT' observations.

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 Cars 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 cars, on the average, get better gas mileage. The ideal point for Ride is in the top, just right of the center of the plot. Cars near the Ride ideal point tend to have a better ride than cars far away. It can

be seen from the iteration history tables that none of these ratings perfectly fits the model, so all of the interpretations are approximate.

The Ride point is a "negative-negative" ideal point. The point models assume that small ratings mean the object (car) 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 is negative, so the interpretation is reversed again, back to the original interpretation. Anti-ideal points are taken care of in the %PLOTIT macro. Specify ANTIIDEA=1 when large values are positive or ideal and ANTIIDEA=-1 when small values are positive or ideal.

The following statements produce Output 65.5.1 through Output 65.5.2:

```
title 'Preference Ratings for Automobiles Manufactured in 1980';
data CarPreferences;
   input Make $ 1-10 Model $ 12-22 @25 (Judge1-Judge25) (1.)
         MPG Reliability Ride;
   datalines;
Cadillac   Eldorado     800799049124050897109380 9 3 2 4
Chevrolet  Chevette     005120042345104300351569 8 5 3 2
Chevrolet  Citation     405330581416164354474779 5 4 1 5
Chevrolet  Malibu       602740072312134554566865 8 3 3 4
Ford       Fairmont     202400671502144353064865 5 3 3 4
Ford       Mustang      500719770502110185065755 5 3 2 2
Ford       Pinto        002100030303020150051407 8 4 1 1
Honda      Accord       595689760969995299897507 8 5 5 3
Honda      Civic        483670950748885256776507 5 5 5 3
Lincoln    Continental  700899059223040996209190 9 2 4 5
Plymouth   Gran Fury    700600043410110733345870 8 2 1 5
Plymouth   Horizon      300500563546130244467565 5 4 3 3
Plymouth   Volare       400500361402160275447655 5 2 1 3
Pontiac    Firebird     010789561320120695826590 7 1 1 5
Volkswagen Dasher       485869650887779537789500 0 5 3 4
Volkswagen Rabbit       485850970969579548788500 0 5 4 3
Volvo      DL           998999890999998798991900 0 4 5 5
;

*---Compute Coordinates for a 2-Dimensional Scatter Plot of Cars---;
proc prinqual data=CarPreferences out=PResults(drop=Judge1-Judge25)
              n=2 replace standard scores;
   id Model MPG Reliability Ride;
   transform identity(Judge1-Judge25);
   title2 'Multidimensional Preference (MDPREF) Analysis';
run;

*---Compute Endpoints for MPG and Reliability Vectors---;
proc transreg data=PResults;
   Model identity(MPG Reliability)=identity(Prin1 Prin2);
   output tstandard=center coordinates replace out=TResult1;
   id Model;
   title2 'Preference Mapping (PREFMAP) Analysis';
run;
```

```
*---Compute Ride Ideal Point Coordinates---;
proc transreg data=PResults;
   Model identity(Ride)=point(Prin1 Prin2);
   output tstandard=center coordinates replace noscores out=TResult2;
   id Model;
run;

proc print; run;

*---Combine Data Sets and Plot the Results---;
data plot;
   title3 'Plot of Automobiles and Ratings';
   set Tresult1 Tresult2;
run;

%plotit(data=plot, datatype=vector ideal, antiidea=1, href=0, vref=0);
```

Output 65.5.1. Preference Ratings Example Output

```
          Preference Ratings for Automobiles Manufactured in 1980
               Multidimensional Preference (MDPREF) Analysis

                          The PRINQUAL Procedure

                   PRINQUAL MTV Algorithm Iteration History

Iteration    Average    Maximum     Proportion    Criterion
   Number     Change     Change    of Variance       Change    Note
-------------------------------------------------------------------------
        1    0.00000    0.00000        0.66946                 Converged

 Algorithm converged.
WARNING: The number of observations is less than or equal to the number of
                              variables.
               WARNING: Multiple optimal solutions may exist.
```

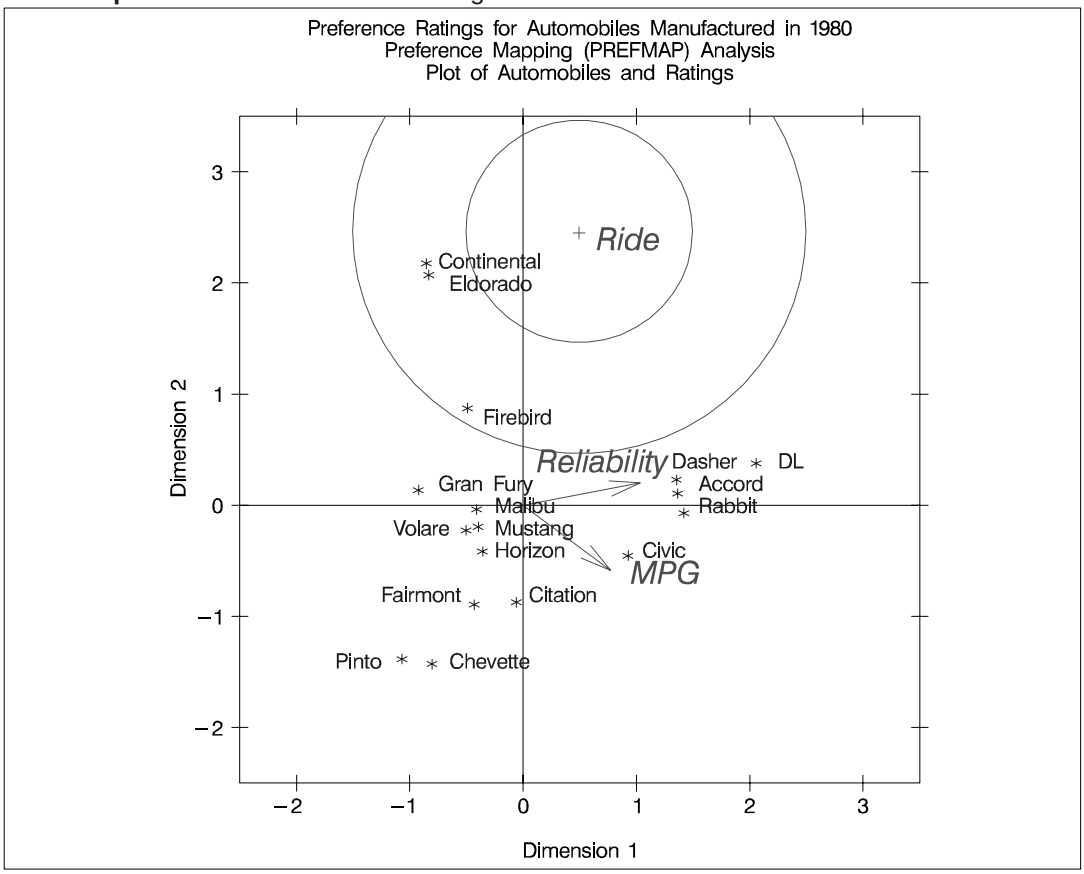
```
          Preference Ratings for Automobiles Manufactured in 1980
                   Preference Mapping (PREFMAP) Analysis

                          The TRANSREG Procedure

     TRANSREG Univariate Algorithm Iteration History for Identity(MPG)

 Iteration    Average    Maximum                Criterion
    Number     Change     Change    R-Square       Change    Note
 ------------------------------------------------------------------------
         1    0.00000    0.00000     0.57197                 Converged

 Algorithm converged.
```

```
          Preference Ratings for Automobiles Manufactured in 1980
                   Preference Mapping (PREFMAP) Analysis

                              The TRANSREG Procedure

  TRANSREG Univariate Algorithm Iteration History for Identity(Reliability)

  Iteration    Average    Maximum                Criterion
     Number     Change     Change    R-Square       Change    Note
  ------------------------------------------------------------------------
          1    0.00000    0.00000     0.50859                 Converged

  Algorithm converged.
```

```
          Preference Ratings for Automobiles Manufactured in 1980
                   Preference Mapping (PREFMAP) Analysis

                              The TRANSREG Procedure

      TRANSREG Univariate Algorithm Iteration History for Identity(Ride)

  Iteration    Average    Maximum                Criterion
     Number     Change     Change    R-Square       Change    Note
  ------------------------------------------------------------------------
          1    0.00000    0.00000     0.37797                 Converged

  Algorithm converged.
```

```
          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
```

Output 65.5.2. Preference Ratings for Automobiles Manufactured in 1980



References

de Boor, C. (1978), *A Practical Guide to Splines*, New York: Springer Verlag.

Breiman, L., and Friedman, J.H. (1985), "Estimating Optimal Transformations for Multiple Regression and Correlation," (with discussion), *Journal of the American Statistical Association*, 77, 580–619.

Brinkman, N.D. (1981), "Ethanol Fuel—A Single-Cylinder Engine Study of Efficiency and Exhaust Emissions," *Society of Automotive Engineers Transactions*, 90, 1410–1424.

van der Burg, E., and de Leeuw, J. (1983), "Non-linear Canonical Correlation," *British Journal of Mathematical and Statistical Psychology*, 36, 54–80.

Carroll, J.D. (1972), "Individual Differences and Multidimensional Scaling," in R.N. Shepard, A.K. Romney, and S.B. Nerlove (eds.), *Multidimensional Scaling: Theory and Applications in the Behavioral Sciences (Volume 1)*, New York: Seminar Press.

Fisher, R. (1938), *Statistical Methods for Research Workers (10th Edition)*, Edinburgh: Oliver and Boyd Press.

Gabriel, K.R. (1981), "Biplot Display of Multivariate Matrices for Inspection of Data and Diagnosis," *Interpreting Multivariate Data*, ed. Barnett, V., London: John Wiley & Sons.

Gifi, A. (1990), *Nonlinear Multivariate Analysis*, New York: John Wiley & Sons, Inc.

Goodnight, J.H. (1978), *SAS Technical Report R-106, The Sweep Operator: Its Importance in Statistical Computing*, Cary NC: SAS Institute Inc.

Green, P.E., and Wind, Y. (1975), "New Way to Measure Consumers' Judgements," *Harvard Business Review*, July–August, 107–117.

Hastie, T., and Tibshirani, R. (1986), "Generalized Additive Models," *Statistical Science*, 3, 297–318.

Israels, A.Z. (1984), "Redundancy Analysis for Qualitative Variables," *Psychometrika*, 49, 331–346.

Khuri, A.I., and Cornell, J.A. (1987), *Response Surfaces*, New York: Marcel Dekker.

Kruskal, J.B. (1964), "Multidimensional Scaling By Optimizing Goodness of Fit to a Nonmetric Hypothesis," *Psychometrika*, 29, 1–27.

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.

de Leeuw, J. (1986), "Regression with Optimal Scaling of the Dependent Variable," Department of Data Theory, The Netherlands: The University of Leiden.

Meyers, R.H. (1976), *Response Surface Methodology*, Blacksburg, VA: Virginia Polytechnic Institute and State University.

van Rijckeveorsel, J. (1982), "Canonical Analysis with B-Splines," in H. Caussinus, P. Ettinger, and R. Tomassone (ed.), *COMPUSTAT 1982*, Part I, Wein, Physica Verlag.

SAS Institute Inc. (1993), SAS Technical Report R-109, *Conjoint Analysis Examples*, Cary, NC: SAS Institute Inc.

Schiffman, S.S., Reynolds, M.L., and Young, F.W. (1981), *Introduction to Multidimensional Scaling*, New York: Academic Press.

Siegel, S. (1956), *Nonparametric Statistics*, New York: McGraw-Hill.

Smith, P.L. (1979), "Splines as a Useful and Convenient Statistical Tool," *The American Statistician*, 33, 57–62.

Stewart, D., and Love, W. (1968), "A General Canonical Correlation Index," *Psychological Bulletin*, 70, 160–163.

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.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/STAT® User's Guide, Version 8*, Cary, NC: SAS Institute Inc., 1999.

SAS/STAT® User's Guide, Version 8

ISBN 1–58025–494–2

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

1st printing, October 1999