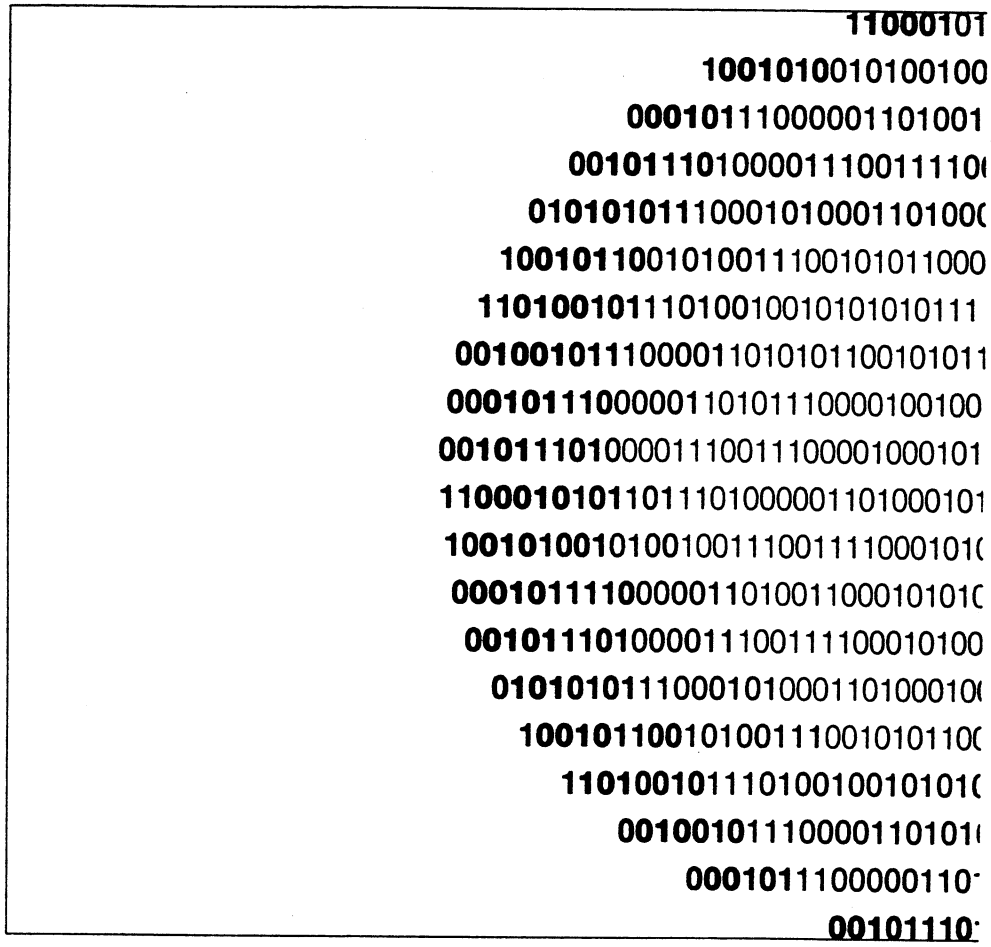
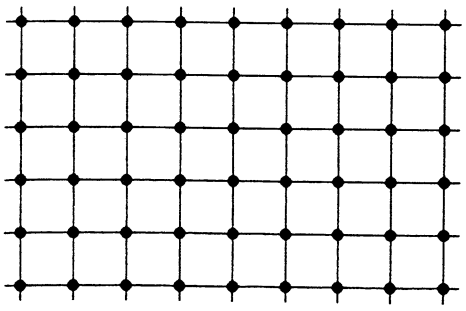
# SAS® Technical Report R-108

Algorithms for the PRINQUAL and TRANSREG Procedures





SAS® SAS Institute Inc.

# SAS® Technical Report R-108 Algorithms for the PRINQUAL and TRANSREG Procedures

SAS Institute Inc.
SAS Circle □ Box 8000
Cary, NC 27512-8000

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., SAS® Technical Report R-108, *Algorithms for the PRINQUAL and TRANSREG Procedures*, Cary, NC: SAS Institute Inc., 1990. 21 pp.

**SAS® Technical Report R-108, Algorithms for the PRINQUAL and TRANSREG Procedures**



ISBN 1-55544-388-5



1st printing, February 1990

# Credits

SAS Technical Report R-108 was created and written by Warren F. Kuhfeld.
Development and support of both the PRINQUAL and TRANSREG procedures is the responsibility of Warren F. Kuhfeld.

# Algorithms for the PRINQUAL and TRANSREG Procedures



## ABSTRACT

The TRANSREG and PRINQUAL procedures obtain linear and nonlinear transformations of variables using the method of alternating least squares. The TRANSREG procedure optimizes fit to linear regression, canonical correlation, and analysis-of-variance models. The PRINQUAL procedure optimizes properties of the transformed variables' covariance or correlation matrix, including the sum of the largest *r* eigenvalues, which is a measure of the fit of the data to an *r* principal components model.

### Intended Audience

This report is intended for advanced users who want to know the details of the internal algorithms used in the PRINQUAL and TRANSREG procedures. The

PRINQUAL and TRANSREG procedures are described in Volume 2 of the *SAS/STAT User's Guide, Version 6, Fourth Edition*. This report provides the six optimal scaling algorithms and the seven METHOD=*name* algorithms used in these procedures. While some background information is provided here, it is assumed that the reader is familiar with the PRINQUAL and TRANSREG chapters (34 and 40, respectively) of the *SAS/STAT User's Guide, Version 6, Fourth Edition*.

# INTRODUCTION

The TRANSREG (transformation regression) procedure is a data transformation procedure that performs many types of analyses.

TRANSREG fits many types of linear models including

- simple and generalized conjoint analysis and other ANOVA models with optional variable transformations (de Leeuw, Young, and Takane 1976; Green and Wind 1975)
- metric and nonmetric vector and ideal point preference regression (Carroll 1972)
- simple, multiple, and multivariate regression with optional variable transformations (Young, de Leeuw, and Takane 1976; Winsberg and Ramsay 1980; Breiman and Friedman 1985)
- redundancy analysis (Stewart and Love 1968) with optional variable transformations (Israels 1984)
- canonical correlation analysis with optional variable transformations (van der Burg and de Leeuw 1983)
- response surface regression (Myers 1976; Khuri and Cornell 1987) with a variety of response surface models and optional variable transformations.

The PRINQUAL procedure (principal components of qualitative data) is a data transformation procedure that is based on the work of Kruskal and Shepard (1974); Young, Takane, and de Leeuw (1978); and Winsberg and Ramsay (1983). PROC PRINQUAL can be used to

- generalize ordinary principal component analysis to a method capable of analyzing data that are not quantitative.
- perform metric and nonmetric multidimensional preference (MDPREF) analyses (Carroll 1972).
- preprocess data, transforming variables prior to their use in other data analyses.
- estimating missing values in multivariate data prior to subsequent data analyses. When used with survey data, PROC PRINQUAL can estimate optimal scores for nominal classes of otherwise ordered variables (such as *unfamiliar with the product* in an ordered preference rating).
- summarize mixed quantitative and qualitative data, and detect nonlinear relationships.
- reduce the number of variables for subsequent use in regression analyses, cluster analyses, and other analyses.

The data can contain variables with nominal, ordinal, interval, and ratio scales of measurement (Siegel 1956). Any mix of these variable types is allowed for the dependent and independent variables.

- Nominal variables can be transformed by scoring the categories to minimize squared error (Fisher 1938) with the OPSCORE transformation.
- Ordinal variables can be transformed monotonically by scoring the ordered categories so that order is weakly preserved (adjacent categories

can be merged) and squared error is minimized with the MONOTONE transformation. Ties can be untied optimally (UNTIE transformation) or left tied (Kruskal 1964).

- Interval and ratio scale of measurement variables can be linearly transformed (LINEAR) or nonlinearly transformed with spline (de Boor 1978; van Rijckevorsel 1982) transformations by specifying SPLINE, or monotone spline (Winsberg and Ramsay 1980) transformations (MSPLINE).
- For all transformations, missing data can be estimated without constraint, with category constraints (that is, missing values within the same group get the same value), and with order constraints (that is, missing value estimates in adjacent groups can be tied to weakly preserve a specified ordering) (Gifi 1981; Young 1981; Kuhfeld and de Leeuw, in preparation).

## Background

The TRANSREG and PRINQUAL procedures extend the ordinary general linear model by providing optimal variable transformations that are iteratively derived using the method of alternating least squares (Young 1981). The ordinary regression and principal component models assume that the variables are all measured on an equal interval scale and, therefore, can be geometrically represented as vectors in an $n$ (the number of observations) dimensional space. In analysis of variance, the independent variables are nominal variables, so they are not correctly represented as single vectors. Nominal independent variables are expanded to design matrices, each column of which can be treated as a vector. Nominal dependent variables can also be handled within the framework of the ordinary general linear model (discriminant analysis).

An ordinary general linear model analysis can be cursorily described as taking a set of interval and nominal variables, expanding the nominal variables to a set of variables that can be treated as vectors, then fitting a regression (or canonical correlation) model to the expanded set of variables. The alternating least-squares algorithm adds one additional capability to the general linear model; it allows variables whose full representation is a matrix consisting of more than one vector to be represented by a single vector, which is an optimal linear combination of the columns of the matrix. For any type of linear model, an alternating least-squares program can solve for an optimal vector representation of any number of variables simultaneously.

Because the alternating least-squares algorithm can replace a matrix with a vector, it can be used for fitting a linear model for many types of variables, including

- interval variables
- ordinal variables with or without category constraints (Kruskal 1964)
- nominal variables within the space of a single vector (Fisher 1938)
- interval variables that should be nonlinearly transformed (van Rijckevorsel 1982)
- interval variables that should be nonlinearly but monotonically transformed (Winsberg and Ramsay 1980)
- any type of variable with some additional ordered or unordered categories (Kuhfeld and de Leeuw, in preparation)
- any type of variable with any mixture of missing value types that should be scored with or without category constraints (Gifi 1981; Young 1981)
- variables that consist of more than one measurement partition (Kuhfeld and de Leeuw, in preparation).

TRANSREG and PRINQUAL can handle any mixture of these cases. They iterate until convergence, alternating between these two steps:

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

For more background on these and related topics, see Kruskal and Shepard (1974); Young, de Leeuw and Takane (1976); de Leeuw, Young and Takane (1976); Tenenhaus and Vachette (1977); Young, Takane and de Leeuw (1978); Winsberg and Ramsay, (1980, 1981, 1983); Young (1981); Gifi (1981); Schiffman, Reynolds, and Young (1981); Coolen, van Rijckevorsel, and de Leeuw (1982); van Rijckevorsel (1982); van der Burg and de Leeuw (1983); Israels (1984); Breiman and Friedman (1985); Hastie and Tibshirani (1986); de Leeuw (1986, 1988); Kuhfeld and de Leeuw (In preparation); and many more.

### Optimal Scaling

An alternating least-squares optimal scaling algorithm can be divided into two major sections. The first section 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 shown in the algorithms in later sections. 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 specified 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.

Optimal scaling can be defined as a possibly constrained, least-squares regression problem. When an optimal transformation family other than LINEAR is specified for a variable, 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, is closest to the target (in terms of minimum squared error), and does 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 the method of possibly constrained, least squares.

Optimal scaling problems are typically solved without actually constructing design and projection matrices. The following sections describe the algorithms used by the TRANSREG and PRINQUAL procedures for optimal scaling. The first section below 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 **x** and the target vector **y**. For convenience, both vectors are first sorted on the values of the initial scaling vector. For UNTIE transformations, 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 partitions **x** and **y** into missing and nonmissing parts $(\mathbf{x}_m' \ \mathbf{x}_n')'$, and $(\mathbf{y}_m' \ \mathbf{y}_n')'$.

Next, category membership is determined. 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= 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, for 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 too, 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 $\mathbf{y}_n$=(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 (except if 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 piece-wise polynomials; however, they are not the most natural method of describing splines. See Smith (1979) for an excellent introduction to 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 below, and the natural piece-wise polynomial spline basis is the right matrix below. 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 **x** element $x$) by the formula: $(x-k)^3*(x>k)$. The logical expression $(x>k)$ evaluates to 1.0 if $x$ is greater than $k$ and 0.0 otherwise. If knot $k$ had been repeated, there would be a $(x-k)^2*(x>k)$ 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.

B-Spline Basis

$$\begin{bmatrix} .171 & .557 & .250 & .022 & 0 & 0 \\ .037 & .447 & .443 & .073 & 0 & 0 \\ .001 & .251 & .576 & .172 & 0 & 0 \\ 0 & .093 & .572 & .334 & .001 & 0 \\ 0 & .020 & .437 & .517 & .027 & 0 \\ 0 & .001 & .253 & .623 & .123 & 0 \\ 0 & 0 & .108 & .557 & .332 & .003 \\ 0 & 0 & .032 & .341 & .548 & .079 \\ 0 & 0 & .004 & .109 & .523 & .364 \end{bmatrix}$$

Piecewise Polynomial Splines

$$\begin{bmatrix} 1&1&1&1&0&0 \\ 1&2&4&8&0&0 \\ 1&3&9&27&0&0 \\ 1&4&16&64&0.125&0 \\ 1&5&25&125&3.375&0 \\ 1&6&36&216&15.625&0 \\ 1&7&49&343&42.875&0.125 \\ 1&8&64&512&91.125&3.375 \\ 1&9&81&729&166.375&15.625 \end{bmatrix}$$

The numbers in the B-spline basis do not have a simple interpretation like the numbers in the natural piece-wise polynomial basis. The B-spline basis has a diagonally banded structure. The band shifts to the right one column 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 procedures take advantage of the sparseness of the B-spline basis when they accumulate 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 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.

This algorithm shows how optimal scaling is handled for SPLINE and MSPLINE transformations.

1. Let **X** be the full ($n \times m$) B−spline basis and **y** be the target. The partitioned crossproduct matrix

$$\mathbf{S} = \begin{bmatrix} \mathbf{X'X} & \mathbf{X'y} \\ \mathbf{y'X} & \mathbf{y'y} \end{bmatrix}$$

   is created without storing the full **X** matrix. The DEGREE = $p$ plus one nonzero elements of each row of **X** are generated one row at a time, **S** is accumulated taking advantage of the sparseness of **X**, and the row of **X** is discarded.
2. If a SPLINE transformation is specified, the first $m$ columns of **S** are swept (Goodnight 1978), so the first $m$ rows of the last column of **S** contain ordinary least-squares regression coefficients, **b**. Go to DONE (step 17).
3. An MSPLINE transformation is requested, so compute least-squares monotone coefficients (steps 4 through 16).
4. Create a tie flags $m$-vector **t** initialized to indicate that each column of **X** starts a new tie block (no ties). The $i$th element of **t** is set to $i$.
5. Create a backup copy of **S**: **C**=**S**.
6. START ITERATING:
7. Restore the original **S**: **S**=**C**.
8. Impose on **S** the ties indicated by the tie flags: if the $i$ th and ($i$+1)th B-spline columns are to be tied to impose monotonicity constraints (if $t_i = t_{i+1}$), add the ($i$ +1)th column of **S** to the $i$ th column and zero the ($i$ +1)th column; then add the ($i$ +1)th row of **S** to the $i$ th row, and zero the ($i$ +1)th row. Check backwards, so a three-way tie is imposed by adding the third row and column to the second, then the new second to the first, resulting in two rows and columns that are zero. Multiple ties can be imposed on **S** this way.
9. Sweep the first $m$ columns of **S** on the nonzero pivots, so the first $m$ rows of the last column of **S** contain ordinary least-squares regression coefficients, (**b**) given the current tie structure.
10. If a coefficient is zero because it is not the first in a tie block (its row and column in **S** is zero), set the coefficient to the nonzero coefficient for the first row in the tie block. When **S** is swept to find ordinary least-squares coefficients, column $\mathbf{x}_i$ can result from the original $\mathbf{x}_i$ plus $\mathbf{x}_{i+1}$.

Creating $b_{i+1} = b_i$ for all ties defines the constrained coefficient vector in terms of the original problem of finding a monotone linear combination of $m$ B-spline columns.

11. Check the coefficients for monotonicity. $b_1$ should be less than $b_2$ and so on. Find the biggest monotonicity violation (if one exists), and set the tie flags so that the offending columns will be combined in the next iteration. If the biggest violation is between $b_i$ and $b_{i+1}$, set $t_{i+1}$ to $t_i$.
12. If a monotonicity violation was found, go to START ITERATING (step 6).
13. Since no monotonicity violations in the coefficients were found, check to see if any of the ties previously imposed can be unimposed (steps 14 through 16). However, if only zero or one tie was imposed, untying is impossible, so go to DONE (step 17).
14. Create the sums of crossproducts between the residuals and the columns of $\mathbf{X}$: $\mathbf{d} = \mathbf{X}'(\mathbf{y} - \mathbf{X}\mathbf{b}) = \mathbf{X}'\mathbf{y} - \mathbf{X}'\mathbf{X}\mathbf{b}$, using the pieces stored in **C** and **S**. While **d** is a zero vector in ordinary least squares, it is not by definition all zero when constraints exist.
15. Create $m$ sums of the elements in **d**: sum the first $i$ elements for $i = 1, 2, \ldots, m$. If the $i$th sum is negative, the $i$th and $(i-1)$th columns can be untied, given the other ties. (The first sum, $d_1$, is always zero.) Find the smallest sum that is less than the negative of the singularity criterion. If one exists, set the tie flags so that the indicated columns will not be tied on the next iteration: for $j = i+1, i+2, \ldots, m$, if $t_j = t_i$ set $t_j = i+1$.
16. If two columns were flagged to be untied, go to START ITERATING (step 6).
17. DONE: (Done computing coefficients.)
18. The **b** vector now contains the (unconstrained for SPLINE or constrained for MSPLINE) least-squares estimates of the coefficients for combining the B-spline columns.
19. The nonzero elements of **X** are generated again and the linear combination **Xb** is computed, again taking advantage of the sparseness in **X**.

# ALGORITHMS

The four TRANSREG and three PRINQUAL alternating least-squares optimal scaling algorithms are discussed in this section.

## Terminology and Notation

optimal scaling
: involves taking the variable approximations and imposing the appropriate measurement level constraints, resulting in a transformed variable that fits the model at least as well as the previous scaling.

the necessary initializations
: computes power, log, logit, inverse sine, rank, and exponent transformations (TRANSREG and PRINQUAL); optionally adds the intercept variable (TRANSREG); expands classification variables into design matrices (TRANSREG); creates the additional variables for external unfolding: a sums-of-squares variable for point models, squared variables for the elliptical and quadratic point models, and crossproduct variables for quadratic point

models (TRANSREG); the dummy variable or MAC initializations (PRINQUAL, both described below).

missing value initialization
: replaces missing values by the variable mean.

internal mean
: is zero if NOINT is not specified; otherwise, the mean of the original variable or, for nonoptimal transformations, the mean of the transformed variable, by default.

the final standardization
: is described in the TSTANDARD= option (PRINQUAL and TRANSREG) and ADDITIVE option (TRANSREG).

the necessary expansions
: optionally add the intercept variable, substitute ranks for RANK variables, expand CLASS variables into design matrices, create the POINT sums-of-squares variable, create EPOINT and QPOINT squared variables, and create QPOINT crossproduct variables.

**X**
: is the current scaling of the independent variables in TRANSREG or, in PRINQUAL, all variables.

**Y**
: is the current scaling of the dependent variable(s).

**B**
: is the matrix of ordinary regression coefficients $\mathbf{X}^{-}\mathbf{Y}$ (where $\mathbf{X}^{-}$ is a generalized inverse of **X**) in the univariate, MORALS and redundancy algorithms, and canonical coefficients for the **X** variables in the CANALS algorithm.

**A**
: is the matrix of canonical coefficients for the **Y** variables in the CANALS algorithm.

appropriate variance
: is the variance of the variable if COVARIANCE is specified; the default is one.

standardized design matrix
: is a matrix constructed by creating a cell indicator design matrix, discarding the last column, centering the remaining columns, and standardizing the matrix so that the rolled out design matrix has the appropriate variance.

*n*
: the number of components specified by N=*n* (PRINQUAL).

## The TRANSREG Univariate Method Algorithm

The univariate algorithm is based on the MORALS method of Young, de Leeuw, and Takane (1976). The univariate algorithm is a simple generalization of ordinary multiple regression to allow dependent variable transformations. The independent variables are not transformed while each dependent variable is separately transformed to be as much as possible, given the transformation family constraints, like a linear combination of the independent variables. While more than one dependent variable can be transformed with a single analysis, the algorithm is univariate since each dependent variable is transformed independently of each other dependent variable. The following algorithm is used:

```
input the data
perform any necessary initializations
store a copy of the data for use in optimal scaling
perform missing value initialization
set variable variances and means
put the initialized independent variables in X
do for each dependent variable that can be transformed:
  put the current dependent variable in Y
  repeat for a maximum number of iterations or until convergence:
    compute regression coefficients B
    compute regression predicted values XB
    set Y equal to the optimally scaled XB
    standardize Y to internal mean and variance one
    evaluate change and output iteration convergence information
    end iteration loop
  end dependent variable loop
compute and output all required output data set information.
```

## The TRANSREG MORALS Method Algorithm

The MORALS algorithm is based on the MORALS method of Young, de Leeuw, and Takane (1976). The MORALS algorithm is a generalization of the univariate algorithm that allows all variables, both dependent and independent, to be transformed. Each dependent variable is transformed to be like a linear combination of the independent variables as much as possible, given the transformation family constraints. Then each independent variable is transformed to maximize the squared multiple correlation as much as possible, given the transformation family constraints. While more than one dependent variable can be transformed with a single analysis, the algorithm is univariate since each dependent variable is transformed independently of each other dependent variable, and the set of independent variables is independently transformed for each dependent variable. The following algorithm is used:

input the data
perform any necessary initializations
store a copy of the data for use in optimal scaling
perform missing value initialization
set variable variances and means
store a copy of the initialized independent variables
do for each dependent variable that can be transformed:
  put initialized independent variables in **X**
  put current dependent variable in **Y**
  repeat for a maximum number of iterations or until convergence:
    compute regression coefficients **B**
    compute regression predicted values **XB**
    replace the **Y** variable with optimally scaled **XB**
    standardize **Y** to internal mean and variance one
    compute residuals: **R** = **Y** − **XB**
    do for each independent variable that can be transformed:
      **b** = coefficient for current independent variable
      **x** = current independent variable
      update residuals: **R** = **R** + **bx**
      compute the target for **x**: **R**/**b**
      replace **x** with the optimally scaled target
      standardize **x** to internal mean and variance one
      update residuals: **R** = **R** − **bx**
      store **x** back in **X**
      end independent variable loop
    evaluate change and output iteration convergence information
    end iteration loop
  compute and output all required output data set information
  end dependent variable loop.

## The TRANSREG Redundancy Method Algorithm

The redundancy analysis algorithm is a logical extension of Young, de Leeuw, and Takane's (1976) work. The redundancy analysis algorithm jointly transforms a set of independent and dependent variables to maximize the average of the $p$ -squared multiple correlations. This is a multivariate method since the entire set of dependent and independent variables are transformed together. The following algorithm is used:

input the data
perform any necessary initializations
store a copy of the data for use in optimal scaling
perform missing value initialization
set variable variances and means
put independent variables in **X**
put dependent variable in **Y**
repeat for a maximum number of iterations or until convergence:
  compute regression coefficients **B**
  compute regression predicted values **XB**
  replace the **Y** variables with optimally scaled **XB**
  standardize **Y** to internal means and variances one
  compute residuals: $\mathbf{R} = \mathbf{Y} - \mathbf{XB}$
  do for each independent variable that can be transformed:
    **x** = current independent variable
    $\mathbf{b}'$ = coefficient row vector for current **x**
    update residuals: $\mathbf{R} = \mathbf{R} + \mathbf{x}\mathbf{b}'$
    compute the target for **x**: $\mathbf{R}\mathbf{b}(\mathbf{b}'\ \mathbf{b})^{-1}$
    replace **x** with the optimally scaled target
    standardize **x** to internal mean and variance one
    update residuals: $\mathbf{R} = \mathbf{R} - \mathbf{x}\mathbf{b}'$
    store **x** back in **X**
    end independent variable loop
  evaluate change and output iteration convergence information
  end iteration loop
compute and output all required output data set information.

## The TRANSREG CANALS Method Algorithm

The CANALS algorithm is based on (but not identical to) the CANALS method of van der Burg and de Leeuw (1983). The canonical analysis algorithm jointly transforms a set of independent and dependent variables to maximize the average of the first $r$-squared canonical correlations. This is a multivariate method since the entire set of dependent and independent variables are transformed together. The following algorithm is used:

input the data
perform any necessary initializations
store a copy of the data for use in optimal scaling
perform missing value initialization
set variable variances and means
put independent variables in **X**
put dependent variable in **Y**
repeat for a maximum number of iterations or until convergence:
  compute ordinary canonical coefficients **B** and **A**
  keep only the first $r$ columns of **B** and **A**
  multiply each column of **A** by its canonical correlation
  compute residuals: $\mathbf{R} = \mathbf{XB} - \mathbf{YA}$
  do for each dependent variable that can be transformed:
    **y** = current dependent variable
    $\mathbf{a}'$ = coefficient row vector for current **y**
    update residuals: $\mathbf{R} = \mathbf{R} + \mathbf{ya}'$
    compute the target for **y**: $\mathbf{Ra}(\mathbf{a}'\mathbf{a})^{-1}$
    replace **y** with the optimally scaled target
    standardize **y** to internal mean and variance one
    update residuals: $\mathbf{R} = \mathbf{R} - \mathbf{ya}'$
    store **y** back in **Y**
    end dependent variable loop
  compute ordinary canonical coefficients **B** and **A**
  keep only the first $r$ columns of **B** and **A**
  multiply each column of **B** by its canonical correlation
  compute residuals: $\mathbf{R} = \mathbf{YA} - \mathbf{XB}$
  do for each independent variable that can be transformed:
    **x** = current independent variable
    $\mathbf{b}'$ = coefficient row vector for current **x**
    update residuals: $\mathbf{R} = \mathbf{R} + \mathbf{xb}'$
    compute the target for **x**: $\mathbf{R}(\mathbf{b}')^{-1}$
    replace **x** with the optimally scaled target
    standardize **x** to internal mean and variance one
    update residuals: $\mathbf{R} = \mathbf{R} - \mathbf{xb}'$
    store **x** back in **X**
    end independent variable loop
  evaluate change and output iteration convergence information
  end iteration loop
compute and output all required output data set information

## The PRINQUAL MAC Method Algorithm

The MAC method (de Leeuw 1985) uses an iterated constrained multiple regression algorithm in an attempt to maximize the average of the elements of the correlation matrix. This method transforms each variable to be (in a least-squares sense) as similar to the average of the remaining variables as possible (de Leeuw 1985).

On each iteration for each variable, the MAC algorithm alternates computing an equally weighted average of the other variables with optimal scaling. The MAC method is similar to the MGV method in that each variable is scaled to be as similar to a linear combination of the other variables as possible, given the constraints on the transformation. However, optimal weights are not computed. The MAC method can be used when all variables are positively correlated, or when no monotonicity constraints are placed on any transformations. Do not use this method with negatively correlated variables when some optimal transformations are constrained to be increasing because the signs of the correlations are not taken into account. The MAC method is useful as an initialization method for the MTV and MGV methods. This method uses the following algorithm:

```
input the data matrix X
perform the nonoptimal transformations
store a copy of X for use in optimal scaling
perform missing value initialization
scale the variables of X to mean zero and appropriate variance
repeat for a maximum number of iterations or until convergence:
  do for all variables:
    select the i th variable as a criterion
    approximate the criterion using the mean of the remaining variables
    optimally scale the approximation and store in x
    standardize x to mean zero and appropriate variance
    replace the i th column of X with x
    end variable loop
  evaluate change and output iteration convergence information
  end iteration loop
perform the final standardization
output the results.
```

### MAC Method Notes

The MAC algorithm can be used alone by specifying METHOD=MAC, or used as an initialization algorithm for METHOD=MTV and METHOD=MGV analyses by specifying the iteration option INITITER=*n*. If any variables are negatively correlated, do not use the MAC algorithm with monotonic transformations (MONOTONE, UNTIE, and MSPLINE) because the signs of the correlations among the variables are not used when computing variable approximations. If an approximation is negatively correlated with the original variable, monotone constraints would make the optimally scaled variable a constant, which is not allowed. When used with other transformations, the MAC algorithm can reverse the scoring of the variables. So, for example, if variable X is designated LOG(X) with METHOD=MAC and TSTANDARD=ORIGINAL, the final transformation (for example, TX) may not be LOG(X). If TX is not LOG(X), it will have the same mean as LOG(X), the same variance as LOG(X), and it will be perfectly negatively correlated with LOG(X). PROC PRINQUAL prints a note for every variable that is reversed in this manner.

The METHOD=MAC algorithm can be used to deliberately reverse the scorings of some rating variables before a factor analysis. The correlations among

bipolar ratings such as 'like - dislike', 'hot - cold', and 'fragile - monumental' are typically both positive and negative. If some items are reversed to say 'dislike - like', 'cold - hot', and 'monumental - fragile', some of the negative signs could be eliminated, and the factor pattern matrix would be cleaner. PROC PRINQUAL can be used with METHOD=MAC and LINEAR transformations to reverse some items, maximizing the average of the intercorrelations.

## The PRINQUAL MTV Method Algorithm

The MTV method (Young, Takane, and de Leeuw 1978) is based on the principal component model and attempts to maximize the sum of the first $r$ eigenvalues of the covariance matrix. This method transforms variables to be (in a least-squares sense) as similar to linear combinations of $r$ principal component score variables as possible, where $r$ can be much smaller than the number of variables. This maximizes the total variance of the first $r$ components (the trace of the covariance matrix of the first $r$ principal components) (Kuhfeld, Sarle, and Young 1985).

On each iteration, the MTV algorithm alternates classical principal component analysis (Hotelling 1933) with optimal scaling (Young 1981). When all variables are ordinal preference ratings, this corresponds to Carroll's (1972) MDPREF analysis. The iterations can be initialized using the method suggested by Tenenhaus and Vachette (1977), who independently proposed the same iterative algorithm for nominal and interval scale-of-measurement variables. This method uses the following iterated principal component algorithm:

input the data matrix **X**
perform the nonoptimal transformations
store a copy of **X** for use in optimal scaling
perform missing value initialization
scale the variables of **X** to mean zero and appropriate variance
perform any necessary initializations
repeat for a maximum number of iterations or until convergence:
  compute **R**, the covariance matrix of **X**
  compute **W**, the first $r$ eigenvectors of **R**
  approximate **X** with **XWW'**
  replace **X** with the optimally scaled variables of **XWW'**
  scale the variables of **X** to mean zero and appropriate variance
  evaluate change and output iteration convergence information
  end iteration loop
perform the final standardization
output the results.

## The MGV Method Algorithm

The MGV method (Sarle 1984) uses an iterated multiple regression algorithm in an attempt to minimize the determinant of the covariance matrix of the transformed variables. This method transforms each variable to be (in a least-squares sense) as similar to linear combinations of the remaining variables as possible. This locally minimizes the generalized variance of the transformed variables, the determinant of the covariance matrix, the volume of the parallelepiped defined by the transformed variables, and sphericity (the extent to which a quadratic form in the optimized covariance matrix defines a sphere) (Kuhfeld, Sarle, and Young 1985).

On each iteration for each variable, the MGV algorithm alternates multiple regression with optimal scaling. The multiple regression involves predicting the selected variable from all other variables. The iterations can be initialized using a modification of the Tenenhaus and Vachette (1977) method that is appropriate with a regression algorithm. This method can be viewed as a way of investigating the nature of the linear and nonlinear dependencies in, and the rank of, a data matrix containing variables that can be nonlinearly transformed. This method tries to create a less-than-full rank data matrix. The matrix contains the transformation of each variable that is most similar to what the other transformed variables predict. This method uses the following iterated regression algorithm:

```
input the data matrix X
perform the nonoptimal transformations
store a copy of X for use in optimal scaling
perform missing value initialization
set the variables of X to mean zero and appropriate variance
perform any necessary initializations
repeat for a maximum number of iterations or until convergence:
  do for all variables:
    select the i th variable as a criterion
    select a full rank set of predictors from all other variables
    approximate the criterion using regression
    optimally scale the approximation and store in x
    standardize x to mean zero and appropriate variance
    replace the ith variable of X with x
    end variable loop
  evaluate change and output iteration convergence information
  end iteration loop
perform the final standardization
output the results.
```

### MGV Sweep Algorithm Details

The MGV method sweeps (Goodnight 1978) the corrected cross product matrix to select the full rank set of predictor variables and compute the regression coefficients. This algorithm does not require that the cross product matrix be recomputed and reswept for each variable for each iteration. The cross products, inverse, and regression coefficients are updated for each variable.

The MGV algorithm selects a full rank set of predictor variables by checking the $R^2$, for predicting a candidate predictor variable from those variables already in the set of predictors. If $(1-R^2)$ is less than the value of SINGULAR= (1E–8 by default), the candidate variable is not included in the set of predictors for this model. When the next model is fit, the new criterion variable is removed from the set of predictors (by sweeping) if it was a predictor. Then the $R^2$ for each of the variables that is not in the set of predictors is checked, as before, to see if

it should be included (swept) into the set of predictors. Once a variable is in the set of predictors it is not removed until it becomes a criterion, or until as many models as is specified by REFRESH= have been fit.

Periodically the computations need to be refreshed to eliminate accumulated rounding error. Refreshing removes all variables from the predictor set so that a new inverse, new regression coefficients, and so on, are computed from the cross product matrix. The REFRESH= option controls how many models are fit before this happens. When the variables are highly collinear, the effects of rounding error are more severe, so REFRESH= should be smaller. The default, REFRESH=5, is conservative. In many cases a larger value does not result in appreciable rounding error. Increasing REFRESH= might result in only a small reduction in iteration time. The sweeps are very fast compared to the time required for accumulating the cross product matrix and performing the other calculations. So, for example, doubling the value of REFRESH= will not halve the iteration time.

One method of detecting when too many sweeps have occurred without refreshing is to compute $R^2$ as the squared length of the predicted values vector, divided by the total sum of squares. If this ratio is different from the $R^2$ computed from the error sum of squares (which is a by-product of the sweep algorithm) by more than the square root of SINGULAR=$n$, a warning is printed, the value of REFRESH= is halved, and the model is refit after refreshing. This error checking can be turned off by specifying NOCHECK. Specifying NOCHECK speeds up the algorithm slightly.

## PRINQUAL Initialization Algorithms

When the DUMMY option is specified, PROC PRINQUAL uses the following initialization algorithms. The first step always replaces each OPSCORE variable in **X** by a standardized design matrix. This process expands **X**, because each single column OPSCORE variable is replaced by a design matrix. By the end of the initialization, each OPSCORE variable in **X** is replaced by a single column which is the optimal scoring, so the order of **X** matches its original order.

The results of the initializations are not reported separately, but are incorporated into the results of the first iteration. The MTV initialization affects all variables, while the MGV initialization only affects OPSCORE variables. The MTV initialization was proposed by Tenenhaus and Vachette (1977), and the MGV initialization is a modification of the Tenenhaus and Vachette algorithm.

### MTV OPSCORE Variable Initialization Algorithm

replace each OPSCORE variable in **X** by a standardized design matrix
compute **R**, the covariance matrix of the expanded **X**
compute **W**, the first $r$ eigenvectors of **R**
compute **F** = **XW**
approximate each OPSCORE variable of **X** with the first
  canonical variable predicting the design matrix from **F**
approximate the other variables of **X** with **XWW'**
replace **X** with the optimally scaled approximations
set the variables of **X** to mean zero and appropriate variance.

The canonical correlation approximation process is as follows: let **Y** be the standardized design matrix. The canonical correlation model is $\mathbf{Ya} \cong \mathbf{Fb}$. The first column of **Fb** is the approximation to the OPSCORE variable.

**MGV OPSCORE Variable Initialization Algorithm**

replace each OPSCORE variable in **X** by a standardized design matrix
do for each OPSCORE variable in **X** (variable *i* ):
  approximate the variable with the first canonical variable
    predicting the design matrix from the rest of **X**
  optimally scale the approximation and store in **x**
  standardize **x** to mean zero and appropriate variance
  replace the *i* th variable of **X** with **x**
  end OPSCORE variable loop.

The canonical correlation approximation process is as follows: let **Y** be the standardized design matrix and **Z** be the rest of **X** . The canonical correlation model is **Ya** $\cong$ **Zb** . The first column of **Zb** is the approximation to the OPSCORE variable.

## 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.

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.

Coolen, H., van Rijckevorsel, J., & de Leeuw, J. (1982), "An Algorithm for Nonlinear Principal Components with B-splines by Means of Alternating Least Squares," in H. Caussinus, P. Ettinger, and R. Tomassone (Eds.), *COMPUSTAT 1982*, Part 2, Vienna: Physica Verlag.

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

Gifi, A. (1981), *Nonlinear Multivariate Analysis*, Department of Data Theory, The Netherlands: The University of Leiden.

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.

Hotelling, H. (1933), "Analysis of a Complex of Statistical Variables into Principal Components," *Journal of Educational Psychology*, 24, 498–520.

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.

Kruskal, J.B., and Shepard, R.N. (1974), "A Nonmetric Variety of Linear Factor Analysis," *Psychometrika*, 38, 123–157.

Kuhfeld, W.F., and de Leeuw, J., "Optimal Scaling of Partitioned Variables," (in preparation).

Kuhfeld, W.F., Sarle, W.S., and Young, F.W. (1985), "Methods of Generating Model Estimates in the PRINQUAL Macro," *SAS Users Group International Conference Proceedings: SUGI 10*, Cary, NC: SAS Institute Inc., 962–971.

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. (1985), (Personal Communication).

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

de Leeuw, J., Young, F.W., & Takane, Y. (1976), "Additive Structure in Qualitative Data: an Alternating Least Squares Method with Optimal Scaling Features. *Psychometrika*, 41, 471–503.

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 1 Vienna: Physica Verlag.

Sarle, W.S. (1984), (Personal Communication).

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.

Tenenhaus, M. and Vachette, J.L. (1977), "PRINQUAL: Un Programme d'Analyse en Composantes Principales d'un Ensemble de Variables Nominales ou Numeriques," *Les Cahiers de Recherche #68*, CESA, Jouy-en-Josas, France.

Winsberg, S., and Ramsay, J.O. (1980), "Monotonic Transformations to Additivity Using Splines," *Biometrika*, 67, 669–674.

Winsberg, S., and Ramsay, J.O. (1981), "Analysis of Pairwise Preference Data Using Integrated B-splines," *Psychometrika*, 46, 171–186.

Winsberg, S. and Ramsay, J.O. (1983), "Monotone Spline Transformations for Dimension Reduction," *Psychometrika*, 48, 575–595.

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.

Young, F.W., Takane, Y., and deLeeuw, J. (1978), "The Principal Components of Mixed Measurement Level Multivariate Data: An Alternating Least Squares Method with Optimal Scaling Features," *Psychometrika*, 43, 279–281.

# Index