

SAS Technical Report R-I 09

Conjoint Analysis Examples



SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., **SAS® Technical Report R-109, *Conjoint Analysis Examples***, Cary, NC: SAS Institute Inc., 1993.85 pp.

SAS® Technical Report R-109, Conjoint Analysis Examples

Copyright © 1993 by SAS Institute Inc., Cary, NC, USA.

ISBN 1-55544-581-0

All rights reserved. Printed in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

1st printing, September 1993

The **SAS® System** is an integrated system of software providing complete control over data access, management, analysis, and presentation. Base SAS software is the foundation of the SAS System. Products within the SAS System include **SAS/ACCESS®**, **SAS/AF®**, **SAS/ASSIST®**, **SAS/CALC®**, **SAS/CONNECT®**, **SAS/CPE®**, **SAS/DMI®**, **SAS/EIS®**, **SAS/ENGLISH®**, **SAS/ETS®**, **SAS/FSP®**, **SAS/GRAPH®**, **SAS/IML®**, **SAS/IMS-DL/I®**, **SAS/INSIGHT®**, **SAS/LAB®**, **SAS/OR®**, **SAS/PH-Clinical®**, **SAS/QC®**, **SAS/REPLAY-CICS®**, **SAS/SHARE®**, **SAS/STAT®**, **SAS/TOOLKIT®**, **SAS/TUTOR®**, **SAS/DB2™**, **SAS/GIS™**, **SAS/IMAGE™**, **SAS/NVISION™**, **SAS/SESSION™**, and **SAS/SQL-DS™** software. Other SAS Institute products are **SYSTEM 2000®** Data Management Software, with basic **SYSTEM 2000**, **CREATE**, **Multi-User**, **QueX™**, **Screen Writer**, and **CICS** interface software; **NeoVisuals®** software; **JMP®**, **IMP IN®**, **IMP Serve**, and **IMP Design@** software; **SAS/RTERM®** software; and the **SAS/C®** Compiler and the **SAS/CX® Compiler**; and **Emulus™** software. **MultiVendor Architecture™** and **MVA™** are trademarks of SAS Institute Inc. **SAS Video Productions™** and the **SVP** logo are service marks of SAS Institute Inc. SAS Institute also offers **SAS Consulting Ambassador Select™** and **On-Site Ambassador™** services. **Authorline®**, **Observations®**, **SAS Communications®**, **SAS Training: SAS Views®**, the **SASware Ballot** and **JMPer Cable™** are published by SAS Institute Inc. All trademarks above are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

The Institute is a private company devoted to the support and further development of its software and related services.

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

Table of Contents

Conjoint Analysis Examples	1
Overview	1
Conjoint Measurement	1
Conjoint Analysis	2
Simulating Market Share	3
Design of Experiments	3
Example 1. Chocolate Candy	5
Metric Conjoint Analysis	5
Output 1.1. Metric Conjoint Analysis	6
Nonmetric Conjoint Analysis	9
Output 1.2. Nonmetric Conjoint Analysis	10
Example 2. Tea Tasting (Basic)	12
Metric Conjoint Analysis	13
Output 2.1. Metric Conjoint Analysis	14
Example 3. Tea Tasting (Advanced)	16
Creating a Design Matrix with ADX	16
Adding Labels, Formats, and Holdouts	18
Output 3.1. Design Matrix with Labels and Formats	19
Output 3.2. Design Matrix with Holdouts	21
Print the Stimuli	21
Output 3.3. The First Two Stimuli for the Conjoint Study	22
Data Collection, Entry, and Preprocessing	22
Output 3.4. Data and Design Together	23
Output 3.5. A Subset of the Final Input Data Set	24
Metric Conjoint Analysis	24
Output 3.6. Metric Conjoint Analysis	25
Output 3.7. Predicted Utilities	28
Output 3.8. Holdout Validation Results	29
Output 3.9. Simulation Results	30
Summarizing Results Across Subjects	31
Output 3.10. OUTTEST= Data Set Contents	31
Output 3.11. Individual R-Squares	32
Output 3.12. Conjoint Analysis Summary Statistics	33
Example 4. Spaghetti Sauce	33
Create a Nonorthogonal Design Matrix with PROC OPTEX	34
Output 4.1. Experimental Design Creation (First Ten)	36
Output 4.2. Experimental Design	37
Generate Descriptions of the Sauces	39
Prepare For Data Collection with PROC FSEDIT	40
Collecting Data Using PROC FSEDIT	43

Output 4.3. Conjoint Input Data	45
Metric Conjoint Analysis	46
Output 4.4. Metric Conjoint Analysis	47
Simulating Market Share, Maximum Utility Model	50
Output 4.5. Market Share Simulation	53
Simulating Market Share, Bradley-Terry-Lute and Logit Models	54
Output 4.6. Market Share Simulation	55
Simulator Comparisons	56
Change in Market Share	59
Output 4.7. Effects of New Products	60
Output 4.8. Change in Market Share	62
Brand by Price Interactions	62
Example 5. Choice of Chocolate Candies	64
Multinomial Logit Model	65
Output 5.1. Multinomial Logit Model with Chocolate Data	66
Output 5.2. Multinomial Logit Model with Chocolate Data	68
PROC TRANSREG Specifications	69
PROC TRANSREG Statement	69
Algorithm Options	70
Output Options	72
Transformations and Expansions	72
Transformation Options	74
BY Statement	75
ID Statement	76
WEIGHT Statement	76
Samples of PROC TRANSREG Usage	77
Metric Conjoint Analysis with Rank-Order Data	77
Metric Conjoint Analysis with Rating Scale Data	77
Nonmetric Conjoint Analysis	78
Monotone Splines	78
Constraints on the Utilities	78
A Discontinuous Price Function	79
More Than One Subject	79
References	80
Index	81

Credits

SAS Technical Report R-109 was created and written by Warren F. Kuhfeld. Development and support of the **TRANSREG** procedure is the responsibility of Warren F. Kuhfeld.

Conjoint Analysis Examples

Overview

Conjoint analysis is used to analyze product preference data and simulate consumer choice. This report describes conjoint analysis and provides examples using the SAS System. Topics include metric and nonmetric conjoint analysis, orthogonal and nonorthogonal experimental designs, data collection and manipulation, holdouts, brand by price interactions, maximum utility and multinomial logit simulators, and change in market share. In addition, the multinomial logit model for discrete choice data is briefly discussed.

Conjoint analysis is also used to study the factors that influence consumers' purchasing decisions. Products possess attributes such as price, color, guarantee, environmental impact, predicted reliability, and so on. Consumers typically do not have the option of buying the product that is best in every attribute, particularly when one of those attributes is price. Consumers are forced to make *trade-offs* as they decide which products to purchase. Consider the decision to purchase a car. Increased size generally means increased safety and comfort, which must be traded off with increased cost and pollution. Conjoint analysis is used to study these trade-offs.

Conjoint analysis is a popular marketing research technique. It is used in designing new products, changing or repositioning existing products, evaluating the effects of price on purchase intent, and simulating market share. Refer to Green and Rao (1971) and Green and Wind (1975) for early introductions to conjoint analysis, refer to Louviere (1988) for a more recent introduction, and refer to Green and Srinivasan (1990) for a recent review article.

Conjoint Measurement

Conjoint analysis grew out of the area of conjoint *measurement* in mathematical psychology. Conjoint measurement is used to investigate the joint effect of a set of independent variables on an ordinal-scale-of-measurement dependent variable. The independent variables are typically nominal and sometimes interval-scaled variables. Conjoint measurement simultaneously finds a monotonic scoring of the dependent variable and numerical values for each level of each independent variable. The goal is to monotonically transform the ordinal values to equal the sum of their attribute level values. Hence, conjoint measurement is used to derive an interval variable from ordinal data. The conjoint measurement model is a mathematical model, not a statistical model, since it has no statistical error term.

Conjoint Analysis

Conjoint *analysis* is based on a main effects analysis-of-variance model. Data are collected by asking subjects about their preferences for hypothetical products defined by attribute combinations. Conjoint analysis decomposes the judgment data into components, based on qualitative attributes of the products. A numerical *utility* or *part-worth utility* value is computed for each level of each attribute. Large utilities are assigned to the most preferred levels, and small utilities are assigned to the least preferred levels. The attributes with the largest utility range are considered the most important in predicting preference. Conjoint analysis is a statistical model with an error term and a loss function.

Metric conjoint analysis models the judgments directly. When all of the attributes are nominal, the metric conjoint analysis is a simple main-effects ANOVA with some specialized output. The attributes are the independent variables, the judgments comprise the dependent variable, and the utilities are the parameter estimates from the ANOVA model. The following is a metric conjoint analysis model for three factors.

$$y_{ijk} = \mu + \beta_{1i} + \beta_{2j} + \beta_{3k} + \epsilon_{ijk}$$

where

$$\sum \beta_{1i} = \sum \beta_{2j} = \sum \beta_{3k} = 0$$

This model could be used, for example, to investigate preferences for cars that differ on three attributes: mileage, expected reliability, and price. y_{ijk} is one subject's stated preference for a car with the i th level of mileage, the j th level of expected reliability, and the k th level of price. The grand mean is μ , and the error is ϵ_{ijk} .

Nonmetric conjoint analysis finds a monotonic transformation of the preference judgments. The model, which follows directly from conjoint measurement, iteratively fits the ANOVA model until the transformation stabilizes. The R^2 increases during every iteration until convergence, when the change in R^2 is essentially zero. The following is a metric conjoint analysis model for three factors.

$$\Phi(y_{ijk}) = \mu + \beta_{1i} + \beta_{2j} + \beta_{3k} + \epsilon_{ijk}$$

where $\Phi(y_{ijk})$ designates a monotonic transformation of the variable Y .

The R^2 for a nonmetric conjoint analysis model will always be greater than or equal to the R^2 from a metric analysis of the same data. The smaller R^2 in metric conjoint analysis is not necessarily a disadvantage, since results should be more stable and reproducible with the metric model. Metric conjoint analysis was derived from nonmetric conjoint analysis as a special case. Today, metric conjoint analysis is used

more often than nonmetric conjoint analysis.

In the SAS System, conjoint analysis is performed with the SAS/STAT procedure TRANSREG (transformation regression). Metric conjoint analysis models are fit using ordinary least squares, and nonmetric conjoint analysis models are fit using an alternating least squares algorithm (Young, 1981; Gifi, 1990). Conjoint analysis is explained more fully in the examples. The “PROC TRANSREG Specifications” section of this technical report documents the PROC TRANSREG statements and options that are most relevant to conjoint analysis. The “Samples of PROC TRANSREG Usage” section shows some typical conjoint analysis specifications.

Simulating Market Share

In many conjoint analysis studies, the conjoint analysis is not the primary goal. The conjoint analysis is used to generate utilities, which are then used as input to consumer choice and market share simulators. The end result for a product is its expected market share, which is a prediction of the proportion of times that the product will be purchased. The effects on market share of introducing new products can also be simulated.

One of the most popular ways to simulate market share is with the maximum utility model, which assumes each subject will buy with probability 1.0 the product for which he or she has the highest utility. The probabilities for each product are averaged across subjects to get predicted market share.

Other simulation methods include the Bradley-Terry-Lute (BTL) model and the logit model. In the BTL model, probability of choice is a linear function of utility. In the logit model, probability is a logit function of utility. The logit function is nonlinear and strictly increasing.

Maximum Utility: $p_{ijk} = 1.0$ if $y_{ijk} = \text{MAX}(y_{ijk})$, otherwise $p_{ijk} = 0.0$

BTL: $p_{ijk} = y_{ijk} / \sum y_{ijk}$

Logit: $p_{ijk} = \exp(y_{ijk}) / \sum \exp(y_{ijk})$

Design of Experiments

The design of experiments is a fundamental part of conjoint analysis. During conjoint analysis data collection, subjects are asked to judge their preferences for hypothetical products defined by attribute combinations. Experimental designs are used to select attribute combinations. *The factors* of an experimental design are variables that have two or more fixed values, or *levels*. Experiments are performed to study the effects of the factor levels on *the response*, or dependent variable. In a conjoint study, the factors are the attributes of the **hypothetical** products or services, and the response is preference or choice.

The simplest experimental design to generate is *the full-factorial design*, which consists of all possible combinations of the levels of the factors. With five factors, two with two levels, and three with three levels, denoted $2^2 3^3$, there are 108 possible

combinations. In a **full-factorial** design, all main effects, all two-way interactions, and all higher-order interactions are estimable. The problem with a full-factorial design is that, for most practical problems, it is too difficult for subjects to rate all possible combinations.

For this reason, researchers often use *fractional-factorial designs*, which consist of fewer *runs* (factor level combinations) than full-factorial designs. The problem with having fewer runs is that some effects become confounded. Two effects are said to be *confounded* or *aliased* when their effects cannot be distinguished from each other because the levels they take in the design yield identical partitions of the runs.

A special type of fractional-factorial design is *the orthogonal array*, in which all estimable effects are uncorrelated. Orthogonal arrays for main-effects models are frequently used in marketing research. Orthogonal designs are often practical for main-effects models when the number of factors is small (say six or fewer) and the number of levels of each factor is small (say four or fewer). You should use an orthogonal design whenever possible. However, there are some situations in which orthogonal designs are not practical, such as when

- not all combinations of factor levels are feasible or make sense
- the desired number of runs is not available in an orthogonal design
- a nonstandard model is being used, such as a model with interactions, polynomials, or splines.

When an orthogonal design is not practical, you must make a choice. One choice is to change the factors and levels to fit some known orthogonal design. This choice is undesirable for obvious reasons. When a suitable orthogonal design does not exist, nonorthogonal designs can be used instead. Nonorthogonal designs, where some coefficients may be slightly correlated, can be used in all of the situations listed previously. You do not have to adapt every experiment to fit some known orthogonal array. First you choose the number of runs. You are not restricted by the sizes of orthogonal arrays, which come in specific numbers of runs (such as 16, 18, 27, 32, 36, and so forth) for specific numbers of factors with specific numbers of levels. Then you choose a set of *candidutepoints*, which may be all of the points in a full-factorial design or they may be a subset, excluding unrealistic combinations. Algorithms for generating nonorthogonal designs select a set of *design points*, from the candidate points, that optimize an efficiency criterion.

Measures of the efficiency of an $(N_D \times p)$ design matrix X are based on the *information matrix* $X'X$ and its inverse $(X'X)^{-1}$. The variance-covariance matrix of the vector of parameter estimates β in a least-squares analysis is proportional to $(X'X)^{-1}$. An efficient design will have a “small” variance matrix; variance and efficiency are inversely related. The eigenvalues of $(X'X)^{-1}$ provide measures of the “size” of the variance matrix. *A-efficiency* is a function of the arithmetic mean of the eigenvalues, which is given by $\text{trace}((X'X)^{-1})/p$. *D-efficiency* is a function of the geometric mean of the eigenvalues, which is given by $|X'X|^{-1/p}$. If an orthogonal design exists, then it has optimum efficiency; conversely, the more efficient a design is, the more it tends toward orthogonality. The measures of efficiency can be scaled to range from 0 to 100, as shown in the following.

$$\text{A-efficiency} = 100 \times \frac{1}{N_D \text{ trace}((X'X)^{-1})/p}$$

$$\text{D-efficiency} = 100 \times \frac{1}{N_D |(X'X)^{-1}|^{1/p}}$$

These efficiencies measure the goodness of the design relative to orthogonal designs that may be far from possible, so they are not useful as absolute measures of design efficiency. Instead, they should be used relatively, to compare one design to another for the same situation.

The ADX menu system of SAS/QC software can be used to generate an orthogonal array experimental design. The SAS/QC procedure OPTEX can be used to find nonorthogonal designs. See Example 3 for an illustration of using ADX to generate an orthogonal array, and see Example 4 for an illustration of using PROC OPTEX. Refer to Kuhfeld, Garratt, and Tobias (1993) for more information on nonorthogonal experimental designs.

Example 1. Chocolate Candy

This example illustrates conjoint analysis of rating scale data with a single subject. The subject was asked to rate his preference for eight chocolate candies. The covering was either dark or milk chocolate, the center was either hard or soft, and the candy did or did not contain nuts. Ratings were performed on a 1 to 9 scale where 1 was low preference and 9 was high preference. Conjoint analysis is used to determine the importance of each attribute and the utility for each level of each attribute.

Metric Conjoint Analysis

After data collection, the attributes and the rating data are entered into a SAS data set. Note that the \$& specification on the INPUT statement is used to read character data with embedded blanks.

```

title 'Preference for Chocolate Candies';

data choc;
  input choc $ center $ nuts $& rating;
  datalines;
dark hard nuts      7
dark hard no nuts   6
dark soft nuts      6
dark soft no nuts   4
milk hard nuts      9
milk hard no nuts   8
milk soft nuts      9
milk soft no nuts   7
  ;

```

PROC TRANSREG is then used to perform a metric conjoint analysis. Printed output from the metric conjoint analysis is requested by specifying the UTILITIES option on the PROC statement. The analysis variables, the transformation of each variable,

and transformation specific options are specified on the MODEL statement.

```
proc transreg utilities;
  title2 'Metric Conjoint Analysis';
  model linear(rating) = class(choc center nuts / zero=sum);
run;
```

The MODEL statement provides a syntax for general transformation regression models, so it is markedly different from other SAS/STAT procedure MODEL statements. Variable lists are specified in parentheses after a transformation name, LINEAR(RATING) requests a LINEAR transformation of the dependent variable RATING. A transformation name must be specified for all variable lists, even for the dependent variable in metric conjoint analysis, when no transformation is desired. The linear transformation of RATING will not change the original scoring. An equal sign follows the dependent variable specification, then the attribute variables are specified along with their transformation.

```
class(choc center nuts / zero=sum)
```

designates the attributes as CLASS variables with the restriction that the utilities sum to zero within each attribute. A slash must be specified to separate the variables from the transformation option ZERO=SUM. CLASS creates a **main-effects** design matrix from the specified variables. This example produces only printed output; later examples will show how to store results in output SAS data sets.

CLASS variables never change during the analysis, and LINEAR variables with no missing values also do not change, so iteration stops after just one iteration. The ANOVA table provides a rough indication of the fit of the conjoint model. The ANOVA results are, at best, approximate since the normality and independence assumptions are violated. In this example, $R^2 = 0.95$. See Output 1.1.

Output 1.1. Metric Conjoint Analysis

Preference for Chocolate Candles Metric Conjoint Analysis				
TRANSREG Univariate Algorithm Iteration History for LINEAR(RATING)				
Iteration Number	Average Change	Maximum Change	Squared Multiple R	Criterion Change
1	0.00000	0.00000	0.95000	

Output 1.1. (Continued)

Preference for Chocolate Candies Metric Conjoint Analysis					
The TRANSREG Procedure Hypothesis Tests for LINEAR(RATING)					
Univariate ANOVA Table Based on the Usual Degrees of Freedom					
Source	DF	Sum of Squares	Mean Square	F Value	P
Model	4	19.0000000	6.3333333	25.333	0.0046
Error	7	1.0000000	0.2500000		
Total		20.0000000			
	Root MSE	0.5	R-square	0.95000	
	Dep Mean	7	Adj R-sq	0.91250	
	cv	7.1428571			

Preference for Chocolate Candies Metric Conjoint Analysis				
Utilities Table Based on the Usual Degrees of Freedom				
Label	Utility	Standard Error	Importance (% Utility Range)	Variable
Intercept	7.0000000	0.17678		INTERCEPT
CHOC dark	-1.2500000	0.17678	50.000	CLASS.CHOC DARK
CHOC milk	1.2500000	0.17678		CLASS.CHOC MILK
CENTER hard	0.5000000	0.17678	20.000	CLASS.CENTER HARD
CENTER soft	-0.5000000	0.17678		CLASS.CENTER SOFT
NUTS no nuts	-0.7500000	0.17678	30.000	CLASS.NUTS NO NUTS
NUTS nuts	0.7500000	0.17678		CLASS.NUTS NUTS

The next table displays the part-worth utilities. The pattern of utilities shows the most preferred levels of the attributes. Levels with positive utility are preferred over those with negative utility. Milk chocolate (utility = 1.25) was preferred over dark (-1.25), hard center (0.5) over soft (-0.5), and nuts (0.75) over no nuts (-0.75).

Conjoint analysis provides an approximate decomposition of the original ratings. The utility for a candy is the sum of the intercept and the part-worth utilities. The conjoint analysis model for the preference for chocolate type i , center j , and nut content k is

$$y_{ijk} = \mu + \beta_{1i} + \beta_{2j} + \beta_{3k} + \epsilon_{ijk}$$

for $i = 1, 2$; $j = 1, 2$; $k = 1, 2$; where

$$\beta_{11} + \beta_{12} = \beta_{21} + \beta_{22} = \beta_{31} + \beta_{32} = 0$$

The part-worth utilities for the attribute levels are the parameter estimates $\hat{\beta}_{11}$, $\hat{\beta}_{12}$, $\hat{\beta}_{21}$, $\hat{\beta}_{22}$, $\hat{\beta}_{31}$, and $\hat{\beta}_{32}$ from this main-effects ANOVA model. The estimate of the intercept is $\hat{\mu}$, and the error term is ϵ_{ijk} .

The utility for the ijk combination is

$$\hat{y}_{ijk} = \hat{\mu} + \hat{\beta}_{1i} + \hat{\beta}_{2j} + \hat{\beta}_{3k}$$

For the most preferred milk/hard/nuts combination, the utility and actual preference values are

$$7.0 + 1.25 + 0.5 + 0.75 = 9.5 = \hat{y} \approx y = 9.0$$

For the least preferred dark/soft/no nuts combination, the utility and actual preference values are

$$7.0 + -1.25 + -0.5 + -0.75 = 4.5 = \hat{y} \approx y = 4.0$$

The utilities are regression predicted values; the squared correlation between the utilities for each combination and the actual preference ratings is the R^2 .

The importance value is computed from the utility range for each factor (attribute). Each range is divided by the sum of all ranges and multiplied by 100. The factors with the largest utility ranges are the most important in determining preference. Note that when the attributes have a varying number of levels, attributes with the most levels sometimes have inflated importances (Wittink, Krishnamurthi, and Reibstein; 1989).

The importance values show that type of chocolate, with an importance of 50%, was the most important attribute in determining preference.

$$\frac{100 \times (1.25 - -1.25)}{(1.25 - -1.25) + (0.50 - -0.50) + (0.75 - -0.75)} = 50\%$$

The second most important attribute was whether the candy contained nuts, with an importance of 30%.

$$\frac{100 \times (0.75 - -0.75)}{(1.25 - -1.25) + (0.50 - -0.50) + (0.75 - -0.75)} = 30\%$$

Type of center was least important at 20%.

$$\frac{100 \times (0.50 - -0.50)}{(1.25 - -1.25) + (0.50 - -0.50) + (0.75 - -0.75)} = 20\%$$

Nonmetric Conjoint Analysis

In the next part of this example, PROC TRANSREG is used to perform a nonmetric conjoint analysis of the candy data set. The difference between requesting a nonmetric and metric conjoint analysis is the dependent variable transformation; a MONOTONE transformation of RATING variable is requested instead of a LINEAR transformation. The OUTPUT statement is used to put the transformed rating into the OUT= output data set.

```
proc tranareg utilities;
  title2 'Nonmetric Conjoint Analysis';
  model monotone(rating) = class(choc center nuts / zero=sum);
  output;
  run;
```

Nonmetric conjoint analysis iteratively derives the monotonic transformation of the ratings. The R^2 increases from 0.95 for the metric case to 0.96985 for the nonmetric case. PROC TRANSREG evaluates the fit of the conjoint model, adjusting for the optimal transformation of the dependent variable. In this case, there is one degree of freedom (*df*) for the intercept and one for each of the three attributes, leaving only four error *df*. The variable RATING has five different values, so this nonmetric conjoint analysis is similar to fitting a multivariate ANOVA with four dependent variables, three independent variables, and only eight observations. The adjusted multivariate statistics (Wilks' Lambda, Pillai's Trace, and Hotelling-Lawley Trace) are not significant. This is a common problem in nonmetric conjoint analysis that is due to the lack of error *df*. The importances and utilities are slightly different from the metric analysis, but the overall pattern of results is the same. See Output 1.2.

Output 1.2. Nonmetric Conjoint Analysis

Preference for Chocolate Candies Nonmetric Conjoint Analysis				
TRANSREG Univariate Algorithm Iteration History for MONOTONE(RATING)				
Iteration Number	Average Change	Maximum Change	Squared Multiple R	Criterion Change
1	0.08995	0.23179	0.95000	.
2	0.01263	0.03113	0.96939	0.01939
3	0.00345	0.00955	0.96981	0.00042
4	0.00123	0.00423	0.96984	0.00003
5	0.00050	0.00182	0.96985	0.00000
6	0.00021	0.00078	0.96985	0.00000
7	0.00009	0.00033	0.96985	0.00000
8	0.00004	0.00014	0.96985	0.00000
9	0.00002	0.00006	0.96985	0.00000
10	0.00001	0.00003	0.96985	0.00000

Preference for Chocolate Candies Nonmetric Conjoint Analysis					
The TRANSREG Procedure Hypothesis Tests for MONOTONE(RATING)					
Univariate ANOVA Table Based on the Usual Degrees of Freedom					
Source	DF	sum of Squares	Mean Square	F Value	Liberal p
Model	3	19.3969262	6.4656421	42.885	>= 0.0017
Error	4	0.6030738	0.1507684		
Total	7	20.0000000			

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

Root MSE	0.3882891	R-square	0.96985
Dep Mean	7	Adj B-sq	0.94723
cv	5.5469876		

Preference for Chocolate Candies Nonmetric Conjoint Analysis					
Adjusted Multivariate ANOVA Table Based on the Usual Degrees of Freedom					
Dependent Variable Scoring Parameters=4 S=3 M=0 N=-0.5					
Statistic	Value	F Value	Num DF	Den DF	p
Wilks' Lambda	0.03015369	0.675	12	2.93725	<= 0.7310
Pillai's Trace	0.96984631	0.358	12	9	<= 0.9497
Hotelling-Lawley Trace	32.1634374	.	12	0	
Roy's Greatest Root	32.1634374	24.123	4	3	>= 0.0129

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 $H \cdot \text{inv}(H+E)$. 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.

Output 1.2. (Continued)

Preference for Chocolate Candies Nonmetric Conjoint Analysis Utilities Table Based on the Usual Degrees of Freedom				
Label	Utility	Standard Error	Importance (% Utility Range)	Variable
Intercept	7.0000000	0.13728		INTERCEPT
CHOC dark	-1.3142511	0.13728	53.209	CLASS.CHOC DARK
CHOC milk	1.3142511	0.13728		CLASS.CHOC MILK
CENTER hard	0.4564317	0.13728	10.479	CLASS.CENTER HA
CENTER soft	-0.4564317	0.13728		CLASS.CENTER SO
NUTS no nuts	-0.6993060	0.13728	28.312	CLASS.NUTS NO_N
NUTS nuts	0.6993068	0.13728		CLASS.NUTS NUTS
The standard errors are not adjusted for the fact that the dependent variable was transformed and so are generally liberal (too small).				

When the dependent variable is monotonically transformed in PROC TRANSREG, the procedure computes the *df* for the multivariate tests in two ways. One *df* is associated with each parameter estimate. If there are m categories in a MONOTONE! variable, a *conservative* count of the number of scoring parameters is $m - 1$. However, there will typically be fewer than $m - 1$ unique parameter estimates since some of those $m - 1$ parameter estimates may be tied to impose *monotonicity*. Imposing ties 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. Basing the results on $m - 1$ is conservative since it does not compensate for the fact that the transformation is restricted to be monotone. Using the smaller count (the number of scoring parameter estimates that are different, minus one for the intercept) 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 *df* to provide lower and upper bounds on the “true” p-values. When both *df* counts are the same, as in this example, only one set of tests is reported, based on the *usual df*.

The GPLOT procedure is used to plot the transformation of the ratings. In this case, the transformation is nearly linear. In practice, the R^2 may increase much more than it did in this example, and the transformation may be markedly nonlinear.

```
proc sort;
  by rating;
run;

goptiona reset=goptions device=pslepsz gsfmde=replace
  gaccea=gaafile hsize=4.5in vsize=4.5in
  ftext=swiss colors=(black);
filename gsafile "choc1.ps";

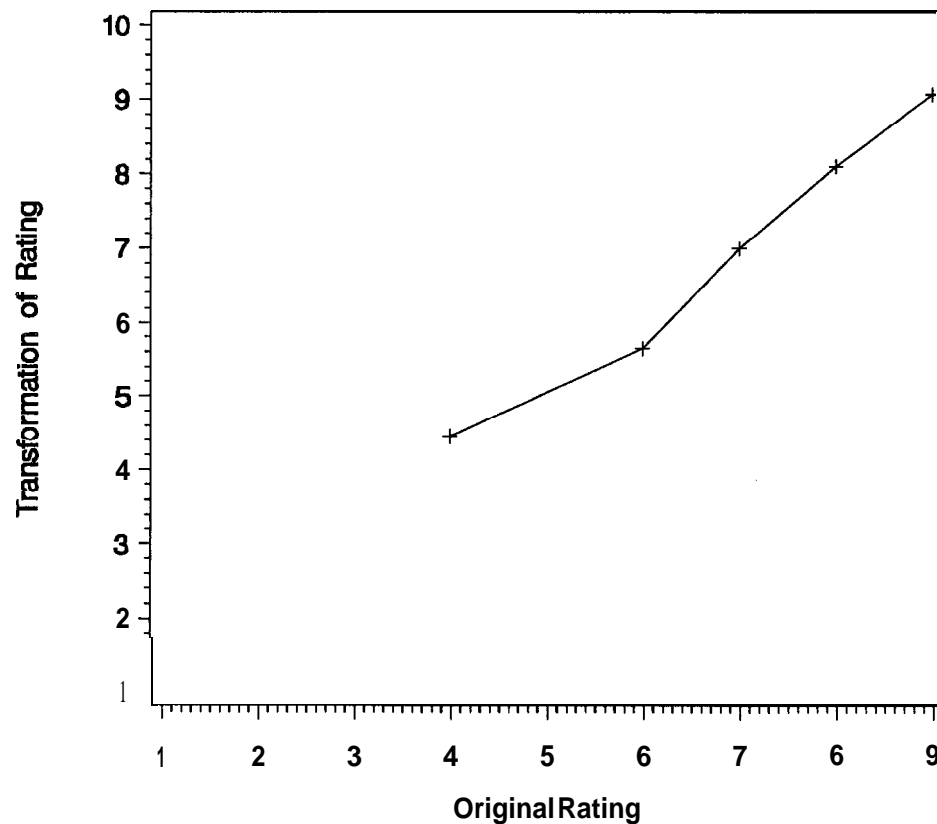
proc gplot;
  title h=1.5 'Preference for Chocolate Candies';
  title2 h=1 'Nonmetric Conjoint Analysis';
  plot rating * rating = 1 / frame haxis=axis2 vaxis=axis1;
  symbol1 v=plus i=join;
```

```

axis1 order=(1 to 10)
      label=(angle=90 'Transformation of Rating');
axis2 order=(1 to 9) label=('Original Rating');
format trating 4.;
run;

```

Preference for Chocolate Candies Nonmetric Conjoint Analysis



Example 2. Tea Tasting (Basic)

This example (inspired by Carroll 1972) uses PROC TRANSREG to perform a conjoint analysis on a set of tea-tasting data. The tea has four attributes: three with three levels and one with two levels. The attributes are temperature (hot, warm, and iced), sweetness (no sugar, 1 teaspoon, 2 teaspoons), strength (strong, moderate, weak), and lemon (with lemon, no lemon). The experimental design is a $2^1 3^3$ orthogonal array in 18 runs. (See Example 3 for an illustration of how to create an orthogonal array.) The subject was asked to assign a 1 to the most preferred combination and an 18 to the least preferred combination.

Metric Conjoint Analysis

This example shows conjoint analysis in one of its simplest forms. First, the FORMAT procedure is used to create descriptive labels, such as Hot and No Sugar, for numeric factor levels, coded with values such as 0 and 1. Then the experimental design and data are entered together in a SAS data set. Formats and labels are assigned to the design variables. The first observation in the data set was the most preferred (RANKING = 1). It is the LEMON /STRENGTH/SWEET/ TEMP combination 1 / 1 / 0 / 3, which means: lemon, yes / strength, strong / sweetness, no sugar / temperature, iced.

```

title 'Conjoint Analysis of Tea-Tasting Data';

proc format;
  value temf
    1 = 'Hot      '
    2 = 'Warm    '
    3 = 'Iced     ' ;
  value swef
    0 = 'No Sugar '
    1 = '1 Teaspoon '
    2 = '2 Teaspoons' ;
  value strf
    1 = 'Strong  '
    2 = 'Moderate '
    3 = 'Weak    ' ;
  value lemf
    1 = 'Yes     '
    2 = 'No      ' ;
run;

data combos;
  input lemon strength sweet temp ranking;
  format lemon lemf. strength strf. sweet swef. temp temf.;
  label lemon      = 'Lemon:'
        strength   = 'Strength:'
        sweet      = 'Sweetness:'
        temp       = 'Temperature: ';
  datalines;
1 1 0 3 1
1 1 1 1 9
1 1 2 2 17
1 2 0 1 3
1 2 1 2 11
1 2 2 3 13
1 3 0 2 5
1 3 1 3 7
13 2 115
2 1 0 1 4
2 1 1 2 12
2 1 2 3 14
2 2 0 2 6
2 2 1 3 8
22 2 116
2 3 0 3 2
2 3 1 1 10
2 3 2 2 18
;

```

The **UTILITIES** option on the PROC TRANSREG statement requests the conjoint analysis results. The **SHORT** option suppresses the iteration history tables, since there is only one iteration. The MODEL statement is like the earlier metric conjoint analysis MODEL statement: **LINEAR** is used for the dependent variable, and the attributes are designated as **CLASS** variables with the restriction that the utilities sum to zero within each attribute. The difference is that the **REFLECT** option is applied to the dependent variable **RANKING**. A small rank means high preference, so the data must be reflected so that high preference corresponds to a large utility. With ranks ranging from 1 to 18, **REFLECT** transforms 1 to 18, 2 to 17, ..., r to $(19 - r)$, ..., and 18 to 1.

The **OUTPUT** statement creates the **OUT=** data set, which contains the original variables, transformed variables, and dummy variables. The utilities for each combination are written to this data set by the **DAPPROXIMATIONS** option (for dependent variable approximations, which are the predicted values). The **IREPLACE** option specifies that the transformed independent variables replace the original independent variables, since both are the same.

Finally, the **OUT=** data set is sorted and the combinations are printed along with their rank, transformed (reflected) rank, and rank approximation (predicted utility).

```
proc transreg utilities short;
  title2 'Use PROC TRANSREG to Perform the Conjoint Analysis';
  model linear(ranking / reflect) =
    class(lemon temp sweet strength / zero=sum);
  output ireplace dapproximations;
run;

proc sort;
  by ranking;
run;

proc print;
  title2 'Some of the OUT= Data Set';
  var ranking tranking aranking lemon temp sweet strength;
run;
```

Output 2.1. Metric Conjoint Analysis

Conjoint Analysis of Tea-Tasting Data					
Use PROC TRANSREG to Perform the Conjoint Analysis					
The TRANSREG Procedure Hypothesis Tests for LINEAR(RANKING)					
Univariate ANOVA Table Based on the Usual Degrees of Freedom					
Source	DF	sum of Squares	Mean Square	F Value	p
Model	7	484.500000	69.214286	9999.999	0.0001
Error	10	0.000000	0.000000		
Total	17	484.500000			
	Root MSE	0	R-square	1.00000	
	Deg Mean	9.5	AdjR-Sq	1.00000	
	cv	0			

Output 2.1. (Continued)

Conjoint Analysis of Tea-Tasting Data Use PROC TRANSREG to Perform the Conjoint Analysis Utilities Table Based on the Usual Degrees of Freedom								
Label		Utility	Standard Error	Importance (% Utility Range)	Variable			
Intercept		9.5000000	0.0000		INTERCEPT			
Lemon:	NO	-0.5000000	0.0000	5.882	CLASS.LEMONNO			
Lemon:	Yes	0.5000000	0.0000		CLASS.LEMONYES			
Temperature:	Hot	-0.0000000	0.0000	23.529	CLASS.TEMPHOT			
Temperature:	Iced	2.0000000	0.0000		CLASS.TEMPICED			
Temperature:	Warm	-2.0000000	0.0000		CLASS.TEMPWARM			
Sweetness:	1 Teaspoon	0.0000000	0.0000	70.588	CLASS.SWEET1_T			
Sweetness:	2 Teaspoons	-6.0000000	0.0000		CLASS.SWEET2_T			
Sweetness:	No Sugar	6.0000000	0.0000		CLASS.SWEETNO			
Strength:	Moderate	0.0000000	0.0000	0.000	CLASS.STRENGMO			
Strength:	Strong	-0.0000000	0.0000		CLASS.STRENGST			
Strength:	Weak	0.0000000	0.0000		CLASS.STRENGWE			

Conjoint Analysis of Tea-Tasting Data Some of the OUT= Data Set								
OBS	RANKING	TRANKING	ARANKING	LEMON	TEMP	SWBBT		STRENGTH
1	1	18	18	Yes	Iced	No	Sugar	Strong
2	2	17	17	No	Iced	No	Sugar	Weak
3	3	16	16	Yes	Hot	No	Sugar	Moderate
4	4	15	15	No	Hot	No	Sugar	Strong
5	5	14	14	Yes	Warm	No	Sugar	Weak
6	6	13	13	No	Warm	No	Sugar	Moderate
7	7	12	12	Yea	Iced	1	Teaspoon	Weak
8	8	11	11	No	Iced	1	Teaspoon	Moderate
9	9	10	10	Yea	Hot	1	Teaspoon	Strong
10	10	9	9	No	Hot	1	Teaspoon	Weak
11	11	8	8	Yes	Warm	1	Teaspoon	Moderate
12	12	7	7	No	Warm	1	Teaspoon	Strong
13	13	6	6	Yes	Iced	2	Teaspoons	Moderate
14	14	5	5	No	Iced	2	Teaspoons	Strong
15	15	4	4	Ye0	Hot	2	Teaspoons	Weak
16	16	3	3	No	Hot	2	Teaspoons	Moderate
17	17	2	2	Yea	Warm	2	Teaspoons	Strong
18	18	1	1	No	Warm	2	Teaspoons	Weak

See Output 2.1 for the conjoint analysis results. The R^2 in the ANOVA table is 1.0, so this subject's data perfectly fits the main effects ANOVA model. The F-value of 9999.999 is printed when the mean square error is essentially zero. The transformed ranking (TRANKJNG) is exactly reproduced by summing the intercept and appropriate utility sums. In practice, results are almost never this clean.

The output dataset is sorted by rank, so that the most preferred combinations are printed first and the least preferred combinations are printed last. The sorted listing shows why the conjoint analysis came out as it did. Sweetness was the most important attribute (70.588%), and the observations are sorted by the sweetness utilities. All of the no sugar observations are printed first (utility 6.0), followed by all of the 1 teaspoon observations (utility 0.0), followed by all of the 2 teaspoons observations (utility -6.0). Temperature was the second most important attribute (23.529%) and

within sugar groups, iced tea (utility 2.0) is always preferred to hot tea (utility 0.0), which is always preferred over warm tea (utility -2.0). Lemon has importance 5.882%, and lemon (utility 0.5) is preferred over no lemon (utility -0.5). The order of the observations after sorting by rank is the same as if they had been sorted by the utilities on the most important through least important variables — that is, sorted by utility of sweetness, followed by temperature, lemon, and strength. The order is completely determined by the first three variables, so strength cannot help in determining preference and has a zero importance. With perfect fit, the transformed ranking is exactly equal to the total utility for each combination, which is the sum of the part-worth utilities and the intercept.

Example 3. Tea Tasting (Advanced)

This example is an advanced version of the previous example. It illustrates conjoint analysis with more than one subject. It has six parts.

1. The ADX menu system, a component of SAS/QC software, is used to generate an orthogonal array experimental design.
2. Labels and formats are added to the design, and holdout observations are generated.
3. The descriptions of the tea are printed for data collection.
4. The data are collected, entered, and preprocessed.
5. The metric conjoint analysis is performed.
6. Results are summarized across subjects.

Creating a Design Matrix with ADX

The first step in a conjoint analysis is to decide on the attributes and their levels and to create the design matrix. The ADX menu system can be used to generate an orthogonal array experimental design. If you are not using ADX to generate a design, enter your design matrix into a SAS data set, as in the previous examples.

Invoke the ADX menu system from the display manager by typing “DESIGN” or “ADX” on any command line. Alternatively, invoke ADX from SAS/ASSIST by selecting PLANNING TOOLS from the main menu, then DESIGN OF EXP. If this is the first time you have invoked ADX, a series of initialization and introductory help screens appear. Move from screen to screen by placing the cursor on specific locations in the window and pressing the ENTER key. Option selection and tabbing varies across operating systems and terminals. For some terminals it may be necessary to press RETURN instead of pressing ENTER or clicking a mouse. For example, the instruction ‘select OK’ means place the cursor on OK and press ENTER, or place the cursor on OK and press RETURN, or place the cursor or mouse pointer on OK and click the mouse.

To create the design matrix, perform the following steps:

1. Invoke ADX.
2. If this is the first invocation of ADX, answer the initialization questions until you get to the first help screen. (Select OK. Select HIGH if you have a graphics terminal; otherwise, select LOW.) It is not necessary to read all of the help screens to create your first design. To exit the helps, select Exit Help. You will be placed in an ADX: Prompt Window and instructed to Select what you want to do:.
3. ADX automatically randomizes the design. In most experiments automatic randomization is desirable, and so it is the default. However, if you are working through this example and wish to ensure that your results will match those presented here, you must turn automatic randomization off. To turn off automatic randomization:
 - a. Select NoPrompt.
 - b. Select File.
 - c. Select Set global parameters.
 - d. Select No (after Automatic randomization:).
 - e. Select OK.
 - f. Select Help.
 - g. Select Prompt.

You should be back in the ADX: Prompt Window, and you should again see Select what you want to do:.

4. To construct the orthogonal array for this experiment, select Add a new design. If this is the first time you have invoked the ADX menu system, this will be the only selection available.
5. The main design definition screen appears next with a prompting window requesting a name for the design. This name will become the name of the data set holding the constructed design and will appear in ADX's primary list of designs. Type the design name, "TEA". Select OK.
6. The design type prompt appears next. Select Orthogonal Array.
7. The next screen requests a description of the design. The descriptive label will appear in ADX's primary list. Type "Tea-Tasting Experiment". Select OK.
8. The next screen defines the factors of the experiment. Enter the number of levels. Type "1" for one 2-level factor, tab to the next field, type "3" for three 3-level factors, and press ENTER.

9. A table of default factor names, number of levels, and values will appear. This table can be edited. The default table for this example is:

Factor	Low	Middle	High
X1	2 -1		1
x2	3 -1	0	1
x3	3 -1	0	1
x4	3 -1	0	1

10. Change the default values to represent the factors and levels for the current experiment. Tab to each field and type over the default values. For this example, edit this screen as follows:

Factor	Low	Middle	High
LEMON	2 No		Yes
TEMP	3 Iced	Warm	Hot
SWEET	3 0	1	2
STRENGTH	3 Weak	Moderate	Strong

11. Select OK.
12. ADX requires a dependent (response) variable name even though it is not needed now. Type “1” for number of responses. Select OK.
13. Finally, a list of possible designs is presented. In this example, the list contains only one design, the 18 run (18 observation) orthogonal array. When this design is selected, ADX constructs it in the background and returns to the primary prompt screen. Select 18 Main effects only.
14. Select Exit ADX.

The design is now in a permanent SAS data set, SASUSER.TEA. It can be printed or manipulated like any other SAS data set.

Adding Labels, Formats, and Holdouts

After the design is generated, it is helpful to add variable labels and formats to the design matrix. Labels and formats can more fully describe the variables and their levels than the original variable names and values. The variable labels all end in colons to facilitate generating summary statistics across subjects. See Output 3.1 for the design matrix.

```
proc format;
  value swef 0 = 'No Sugar'
            1 = '1 Teaspoon'
            2 = '2 Teaspoons';
run;

data sasuser.tea2;
  set sasuser.tea;
  label lemon = 'Lemon:'
         temp  = 'Temperature:'
         sweet = 'Sweetness:'
```



```

        strength = 'Strength: ';
format sweet swef;
drop y;
run;

proc print label;
  title2 'Tea-Tasting Design Matrix';
run;

proc contents position;
run;

```

Output 3.1. Design Matrix with Labels and Formats

Conjoint Analysis of Tea-Tasting Data				
Tea-Tasting Design Matrix				
OBS	Lemon:	Strength:	sweetness:	Temperature:
1	NO	Weak	No Sugar	Iced
2	NO	Moderate	1 Teaspoon	Iced
3	NO	Strong	2 Teaspoons	Iced
4	NO	Moderate	No Sugar	Warm
5	No	Strong	1 Teaspoon	Warm
6	No	Weak	2 Teaspoons	Warm
7	No	Strong	No Sugar	Hot
8	No	Weak	1 Teaspoon	Hot
9	No	Moderate	2 Teaspoons	Hot
10	Yes	Strong	No Sugar	Iced
11	Yes	Weak	1 Teaspoon	Iced
12	Yes	Moderate	2 Teaspoons	Iced
13	Yes	Weak	No Sugar	Warm
14	Yes	Moderate	1 Teaspoon	Warm
15	Yes	Strong	2 Teaspoons	Warm
16	Yes	Moderate	No Sugar	Hot
17	Yea	Strong	1 Teaspoon	Hot
18	Yea	Weak	2 Teaspoons	Hot

Conjoint Analysis of Tea-Tasting Data						
Tea-Tasting Design Matrix						
CONTENTS PROCEDURE						
Data Set Name:	SASUSER.TEA2	Observations:	18			
Member Type:	DATA	Variables:	4			
Engine :	SASEB	Indexes:	0			
Created:	DDMMYY:00:00:00	Observation Length:	27			
Last Modified:	DDMMYY:00:00:00	Deleted Observations:	0			
Protection:		Compressed:	NO			
Data Set Type:		Sorted:	NO			
Label:						
-----Variables Ordered by Position-----						
#	Variable	Type	Len	Pos	Format	Label
1	LEMON	Char	3	0		Lemon:
2	STRENGTH	Char	8	3		Strength:
3	SWEET	Num	8	11	SWEF.	Sweetness:
4	TEMP	Char	8	19		Temperature:

The next steps add *holdout* observations. **Holdouts** are ranked by the subjects but are analyzed with zero weight to exclude them from contributing to the utility computations. The correlation between the ranks for **holdouts** and their predicted utilities provide an indication of the validity of the results of the study. The next steps select four random **holdouts** from those combinations not in the design. See Output 3.2.

```

* Generate all combinations in sorted order;
data sasuser.allcombo;
  a0 lemon = 'No ', 'Yes';
  a0 temp = 'Hot ', 'Iced', 'Warm';
  a0 sweet = 0, 1, 2;
  do strength = 'Moderate', 'Strong', 'Weak';
    output;
  end;
end;
end;
end;
run;

proc sort data=sasuser.tea2 out=sorted;
  by lemon temp sweet strength;
run;

* Generate holdouts from observations not in the design;
data holdout;
  merge sasuser.allcombo sorted(in=s);
  by lemon temp sweet strength;
  if not s;
  rand = uniform(7);
run;

proc sort data=holdout out=holdout;
  by rand;
run;

data sasuser.tea3;
  set sasuser.tea2(in=w)
      holdout(obs=4); /* Choose 4 holdouts at random. */
  weight = w;
  rand = uniform(7);
run;

proc sort data=sasuser.tea3 out=sasuser.tea3(drop=rand);
  by rand;
run;

proc print label;
  title2 'Design with Holdouts, in a Random Order';
run;

```

Output 3.2. Design Matrix with Holdouts

Conjoint Analysis of Tea-Tasting Design with Holdouts, in a Random Order					
OBS	Lemon:	Strength:	Sweetness:	Temperature:	WEIGHT
1	Yes	Moderate	No Sugar	Hot	1
2	No	Weak	No Sugar	Iced	1
3	No	Moderate	2 Teaspoons	Hot	1
4	No	Strong	1 Teaspoon	Warm	1
5	Yes	Strong	No Sugar	Iced	1
6	Yes	Moderate	1 Teaspoon	Warm	1
7	Yes	Strong	2 Teaspoons	Iced	0
8	Yes	Moderate	2 Teaspoons	Iced	1
9	Yes	Strong	2 Teaspoons	Warm	1
10	No	Weak	1 Teaspoon	not	1
11	Yes	Weak	1 Teaspoon	Iced	1
12	No	Weak	2 Teaspoons	Warm	1
13	No	Moderate	1 Teaspoon	Iced	1
14	No	Strong	2 Teaspoons	Iced	1
15	No	Strong	No Sugar	Hot	1
16	No	Moderate	No Sugar	Warm	1
17	No	Weak	1 Teaspoon	Warm	0
18	Yes	Weak	2 Teaspoons	Hot	1
19	Yes	Strong	2 Teaspoons	Hot	0
20	Yes	Weak	2 Teaspoons	Warm	0
21	Yes	Strong	1 Teaspoon	Hot	1
22	Yes	Weak	No Sugar	Warm	1

Print the Stimuli

Once the design matrix is generated, the *stimuli* (descriptions of the combinations) must be generated for data collection. The next DATA step prints the combinations, which are cut up into cards to be given to the subjects to rank. Only the first two stimuli are printed in the interest of space. See Output 3.3.

```

title;
data _null_;
  set sasuser.tea3;
  file print;
  if mod(_n_,4) eq 1 then put gage-;
  put // 'Combination Number ' _n_
      // 'Temperature = ' temp
      // 'Sugar       = ' sweet
      // 'Strength    = ' strength
      // 'Lemon       = ' lemon;
run;

```

Output 3.3. The First Two Stimuli for the Conjoint Study

```

Combination Number 1
Temperature = Hot
Sugar       = No Sugar
Strength    = Moderate
Lemon       = Yes

Combination Number 2
Temperature = Iced
Sugar       = No Sugar
Strength    = Weak
Lemon       = No

```

Data Collection, Entry, and Preprocessing

The next step in the conjoint analysis study is data collection and entry. Subjects were individually asked to take the 22 cards and rank them from the most preferred combination to the least preferred combination. (In the interest of space, data from only two subjects are analyzed here.) The combination numbers are entered as data from the most preferred to the least preferred. The data follow the DATALINES statement in the next DATA step. For the first subject, 5 was most preferred, 2 was second most preferred, and so on until 12 was the least preferred combination. The DATA step validates the data entry and converts the input to ranks.

```

%let m = 22; /* number of combinations */

* Read the input data and convert to ranks;
data ranks(drop=i k cl-c&m ml);
  input cl-c&m;
  array c[&m] cl-c&m;
  array r[&m] r1-r&m;
  ml = -1;
  do i = 1 to &m;
    k = c[i];
    if 1 <= k <= &m then do;
      if r[k] ne . then
        put 'ERROR: For subject ' _n_ +ml ', combination ' k
            'is given more than once.';
      r[k] = i; /* Convert to ranks. */
    end;
    else put 'ERROR: For subject ' _n_ tml ', combination ' k
            'is invalid.';
  end;
  do i = 1 to &m;
    if r[i] = . then
      put 'ERROR: For subject ' _n_ tml ', combination ' i
          'is not given.';
  end;
  datalines;
5 2 1 15 22 16 11 13 21 10 6 4 17 7 8 14 19 18 3 9 20 12
19 7 14 3 8 13 21 4 9 6 10 11 18 12 17 20 15 1 16 5 2 22
;

```

The next step transposes the data set from one row per subject to one row per product. The PREFIX=SUB J option on PROC TRANSPOSE names the rank variables SUBJ 1 and SUBJ2. Then the input data set is merged with the design matrix. See Output 3.4.

```
proc transpose data=ranks out=ranks prefix=subj;
run;

data both;
merge sasuser.tea3 ranks;
drop _name_;
run;

proc print label;
title2 'Data and Design Together';
run;
```

Output 3.4. Data and Design Together

Conjoint Analysis of Tea-Tasting Data Data and Design Together							
OBS	Lemon:	Strength:	Sweetness:	Temperature:	WEIGHT	SUBJ1	SUBJ2
1	Yes	Moderate	No Sugar	Hot	1	3	18
2	NO	Weak	No Sugar	Iced	1	2	21
3	NO	Moderate	2 Teaspoons	Hot	1	19	4
4	NO	Strong	1 Teaspoon	Warm	1	12	8
5	Yes	Strong	No Sugar	Iced	1	1	20
6	Yea	Moderate	1 Teaspoon	Warm	1	11	10
7	Yes	Strong	2 Teaspoons	Iced	0	14	2
8	Yes	Moderate	2 Teaspoons	Iced	1	15	5
9	Yes	Strong	2 Teaspoon8	Warm	1	20	9
10	NO	Weak	1 Teaspoon	Hot	1	10	11
11	Yes	Weak	1 Teaspoon	Iced	1	7	12
12	NO	Weak	2 Teaspoons	Warm	1	22	14
13	NO	Moderate	1 Teaegoon	Iced	1	8	6
14	NO	Strong	2 Teaspoons	Iced	1	16	3
15	NO	Strong	No Sugar	Hot	1	4	17
16	NO	Moderate	No Sugar	Warm	1	6	19
17	NO	Weak	1 Teaspoon	Warm	0	13	15
18	Yes	Weak	2 Teaspoona	not	1	18	13
19	Yes	Strong	2 Teaspoons	not	0	17	1
20	Yes	Weak	2 Teaspoons	Warm	0	21	16
21	Yes	Strong	1 Teaspoon	not	1	9	7
22	Yea	Weak	No Sugar	Warm	1	5	22

One more data set manipulation is sometimes necessary — the addition of simulation observations. Simulation observations are not rated by the subjects and do not contribute to the analysis. They are scored as passive observations. Simulations are what-if combinations. They are combinations that are entered to get a prediction of what their utility would have been if they had been rated. In this example, all combinations are added as simulations. Simulation observations are given a weight of -1.0 to exclude them from the analysis and to distinguish them from the holdouts. Notice that the dependent variable has missing values for the simulations and non-missing values for the holdouts and active observations. See Output 3.5 for a subset of the final data set.

```

proc format;
  value wf 1 = 'Active'
           0 = 'Holdout'
          -1 = 'Simulation';
run;

data all;
  set both sasuser. allcombo(in=w);
  if w then weight = -1;
  format weight wf.;
run;

proc print data=all(obs=30) label;
  title2 'The First 30 Observations of the Final Data Set';
run;

```

Output 3.5. A Subset of the Final Input Data Set

Conjoint Analysis of Tea-Tasting Data The First 30 Observations of the Final Data Set							
OBS	Lemon:	Strength:	Sweetness:	Temperature:	WEIGHT	SUBJ1	SUBJ2
1	Yes	Moderate	No Sugar	Hot	Active	3	18
2	No	Weak	No Sugar	Iced	Active	2	21
3	No	Moderate	2 Teaspoons	Hot	Active	19	4
4	No	Strong	1 Teaspoon	Warm	Active	12	8
5	Yea	Strong	No Sugar	Iced	Active	1	20
6	Yes	Moderate	1 Teaspoon	Warm	Active	11	10
7	Yes	Strong	2 Teaspoons	Iced	Holdout	14	2
8	Yea	Moderate	2 Teaspoons	Iced	Active	15	5
9	Yes	Strong	2 Teaspoons	Warm	Active	20	9
10	No	Weak	1 Teaspoon	Hot	Active	10	11
11	Yea	Weak	1 Teaspoon	Iced	Active	7	12
12	No	Weak	2 Teaspoons	Warm	Active	22	14
13	No	Moderate	1 Teaspoon	Iced	Active	8	6
14	No	Strong	2 Teaspoons	Iced	Active	16	3
15	No	Strong	No Sugar	Hot	Active	4	17
16	No	Moderate	No Sugar	Warm	Active	6	19
17	No	Weak	1 Teaspoon	Warm	Holdout	13	15
18	Yes	Weak	2 Teaspoons	Hot	Active	18	13
19	Yes	Strong	2 Teaspoons	Hot	Holdout	17	1
20	Yes	Weak	2 Teaspoons	Warm	Holdout	21	16
21	Yes	Strong	1 Teaspoon	Hot	Active	9	7
22	Yes	Weak	No Sugar	Warm	Active	5	22
23	No	Moderate	No Sugar	Hot	Simulation	.	.
24	No	Strong	No Sugar	1-lot	Simulation	.	.
25	No	Weak	No Sugar	Hot	Simulation	.	.
26	No	Moderate	1 Teaspoon	Hot	Simulation	.	.
27	No	Strong	1 Teaspoon	not	Simulation	.	.
28	No	Weak	1 Teaspoon	not	Simulation	.	.
29	No	Moderate	2 Teaspoons	not	Simulation	.	.
30	No	Strong	2 Teaspoons	Hot	Simulation	.	.

Metric Conjoint Analysis

In this part of this example, the conjoint analysis is performed with PROC TRANSREG. The PROC, MODEL, and OUTPUT statements are typical for a metric conjoint analysis of rank-order data with more than one subject. The PROC statement specifies **METHOD=MORALS**, which fits the conjoint analysis model separately for each subject and creates an **OUT=** data set beginning with all observations that contain results for the first subject, followed by all subject two observations, and so on. The PROC statement also requests an **OUTTEST=** data set, which contains the ANOVA and utilities tables from the printed output. The dependent variable list

SUBJ: specifies all variables in the DATA= data set that begin with the prefix SUBJ (in this case SUBJ1 -SUBJ2). The WEIGHT variable designates the active (WEIGHT = 1), holdout (WEIGHT = 0), and simulation (WEIGHT = -1) observations. Only the active observations are used to compute the part-worth utilities. However, predicted utilities are computed for all observations, including active, holdouts, and simulations, using those part-worths.

```
proc transreg data=all utilities short
  method=morals outtest=utils;
  title2'Use PROC TRANSREG to Perform Conjoint Analysis';
  model linear(subj: / reflect) =
    class(lemon temp sweet strength / zero=sum);
  weight weight;
  output dapproximations ireplace
    out=results(keep=_depend_ t-depend a-depend weight
      _depvar_ lemon temp sweet strength);
run;
```

Output 3.6. Metric Conjoint Analysis

Conjoint Analysis of Tea-Tasting Data						
Use PROC TRANSREG to Perform Conjoint Analysis						
The TRANSREG Procedure Hypothesis Tests for LINEAR(SUBJ1)						
Univariate ANOVA Table Based on the Usual Degrees of Freedom						
Source	DF	Sum of Squares	Mean Square	F Value	P	
Model	7	734.888889	104.984127	674.898	0.0001	
Error	10	1.555556	0.155556			
Total	17	136.444444				
Root MSE		0.3944053	R-square	0.99789		
Dep Mean		10.444444	Adj R-sq	0.99641		
cv		3.7762211				

Output 3.6. (Continued)

Conjoint Analysis of Tea-Tasting Data						
Use PROC TRANSREG to Perform Conjoint Analysis						
Utilities Table Based on the Usual Degrees of Freedom						
Label		Utility	Standard Error	Importance (% Utility Range)	Variable	
Intercept		10.4444444	0.0930		INTERCEPT	
Lemon:	No	-0.5555556	0.0930	5.348	CLASS.LEMONNO	
Lemon:	Yes	0.5555556	0.0930		CLASS.LEMONYES	
Temperature:	Hot	-0.0555556	0.1315	21.658	CLASS.TEMPHOT	
Temperature:	Iced	2.2171118	0.1315		CLASS.TEMPICED	
Temperature:	Warm	-2.2222222	0.1315		CLASS.TEMPWARM	
Sweetness:	1 Teaspoon	0.9444444	0.1315	71.390	CLASS.SWEET1_T	
Sweetness:	2 Teaspoons	-7.8888889	0.1315		CLASS.SWEET2_T	
Sweetness:	No Sugar	6.9444444	0.1315		CLASS.SWEETNO	
Strength:	Moderate	0.1111111	0.1315	1.604	CLASS.STRENGMO	
Strength:	Strong	0.1111111	0.1315		CLASS.STRENGST	
Strength:	Weak	-0.2222222	0.1315		CLASS.STRENGWE	

Conjoint Analysis of Tea-Tasting Data						
Use PROC TRANSREG to Perform Conjoint Analysis						
The TRANSREG Procedure Hypothesis Tests for LINEAR(SUBJ2)						
Univariate ANOVA Table Based on the Usual Degrees of Freedom						
Source	DF	sum of Squares	Mean Square	F Value	P	
Model	7	611.722222	88.246032	32.955	0.0001	
Error	10	26.777778	2.671110			
Total	17	644.500000				
	Root MSE	1.6363917	R-square	0.95845		
	Dep Mean	12.166661	Adj R-sq	0.92937		
	cv	13.449795				

Output 3.6. (Continued)

Conjoint Analysis of Tea-Tasting Data Use PROC TRANSRRG to Perform Conjoint Analysis Utilities Table Based on the Usual Degrees of Freedom				
Label	Utility	Standard Error	Importance (% Utility Range)	Variable
Intercept	12.1666667	0.3857		INTERCEPT
Lemon: No	0.7222222	0.3857	7.008	CLASS.LEMONNO
Lemon: Yes	-0.7222222	0.3857		CLASS.LEMONYES
Temperature: Hot	0.5000000	0.5455	12.129	CLASS.TEMPHOT
Temperature: Iced	1.0000000	0.5455		CLASS.TEMPICED
Temperature: Warm	-1.5000000	0.5455		CLASS.TEMPWARM
Sweetness: 1 Teaspoon	3.1666667	0.5455	55.795	CLASS.SWEET1_T
Sweetness: 2 Teaspoons	4.1666667	0.5455		CLASS.SWEET2_T
Sweetness: No Sugar	-7.3333333	0.5455		CLASS.SWEETNO
Strength: Moderate	1.6333333	0.5455	25.067	CLASS.STRENGMO
Strength: Strong	1.5000000	0.5455		CLASS.STRNGST
Strength: Weak	-3.3333333	0.5455		CLASS.STRENGWE

The results are displayed in Output 3.6. The fit for both subjects in this example is quite good. The R^2 s are 0.99789 and 0.95845. The predicted utilities are output by the DAPPROXIMATIONS option and are displayed in Output 3.7. The simulation observations are excluded from the listing. These utilities range from -0.722 (observation 98) to 20.3333 (observation 5).

```
proc print data=results(drop=_depend_ t_depend) label;
  title2 'Predicted Utility';
  where weight > -1;
  by notsorted _depvar_;
  label a-depend = 'Predicted Utility';
run;
```

Output 3.7. Predicted Utilities

Conjoint Analysis of Tea-Tasting Data Predicted Utility						
----- Dependent Variable Transformation(Name)=LINEAR(SUBJ1) -----						
OBS	WEIGHT	Predicted Utility	Lemon:	Temperature:	Sweetness:	Strength:
1	Active	18.0000	Yes	Hot	No Sugar	Moderate
2	Active	18.8889	No	Iced	No Sugar	Weak
3	Active	2.0556	No	Hot	2 Teaspoons	Moderate
4	Active	8.7222	No	Warm	1 Teaspoon	Strong
5	Active	20.3333	Yes	Iced	No Sugar	Strong
6	Active	9.8333	Yes	Warm	1 Teaspoon	Moderate
7	Holdout	5.5000	Yes	Iced	2 Teaspoons	Strong
8	Active	5.5000	Yes	Iced	2 Teaspoons	Moderate
9	Active	1.0000	Yes	Warm	2 Teaspoons	Strong
10	Active	10.5556	No	Hot	1 Teaspoon	Weak
11	Active	14.0000	Yes	Iced	1 Teaspoon	Weak
12	Active	-0.4444	No	Warm	2 Teaspoons	Weak
13	Active	13.2222	No	Iced	1 Teaspoon	Moderate
14	Active	4.3889	No	Iced	2 Teaspoons	Strong
15	Active	16.8889	No	Hot	No Sugar	Strong
16	Active	14.7222	No	Warm	No Sugar	Moderate
17	Holdout	8.3889	No	Warm	1 Teaspoon	Weak
18	Active	2.8333	Yes	Hot	2 Teaspoons	Weak
19	Holdout	3.1667	Yes	Hot	2 Teaspoons	Strong
20	Holdout	0.6667	Yes	Warm	2 Teaspoons	Weak
21	Active	12.0000	Yes	Hot	1 Teaspoon	Strong
22	Active	15.5000	Yes	Warm	No Sugar	Weak
----- Dependent Variable Transformation(Name)=LINEAR(SUBJ2) -----						
OBS	WEIGHT	Predicted Utility	Lemon:	Temperature:	Sweetness:	Strength:
77	Active	6.4444	Yes	Hot	No Sugar	Moderate
78	Active	3.2222	No	Iced	No Sugar	Weak
79	Active	19.3889	No	Hot	2 Teaspoons	Moderate

Conjoint Analysis of Tea-Tasting Data Predicted Utility						
----- Dependent Variable Transformation(Name)=LINEAR(SUBJ2) ----- (continued)						
OBS	WEIGHT	Predicted Utility	Lemon:	Temperature:	Sweetness:	Strength:
80	Active	16.0556	No	Warm	1 Teaspoon	Strong
81	Active	6.6111	Yes	Iced	No Sugar	Strong
82	Active	14.9444	Yes	Warm	1 Teaspoon	Moderate
83	Holdout	18.1111	Yes	Iced	2 Teaspoons	Strong
84	Active	10.4444	Yes	Iced	2 Teaspoons	Moderate
85	Active	15.6111	Yes	Warm	2 Teaspoons	Strong
86	Active	13.2222	No	Hot	1 Teaspoon	Weak
87	Active	12.2778	Yes	Iced	1 Teaspoon	Weak
88	Active	12.2222	No	Warm	2 Teaspoons	Weak
89	Active	18.8889	No	Iced	1 Teaspoon	Moderate
90	Active	19.5556	No	Iced	2 Teaspoons	strong
91	Active	7.5556	No	Hot	No Sugar	Strong
92	Active	5.8889	No	Warm	No Sugar	Moderate
93	Holdout	11.2222	No	Warm	1 Teaspoon	Weak
94	Active	12.7770	Yes	Hot	2 Teaspoons	Weak
95	Holdout	17.6111	Yes	Hot	2 Teaspoons	Strong
96	Holdout	10.7778	Yes	Warm	2 Teaspoons	Weak
97	Active	16.6111	Yes	Hot	1 Teaspoon	Strong
98	Active	-0.7222	Yes	Warm	No Sugar	Weak

The next step displays correlations between the predicted utility for holdout observations and their actual ratings. These correlations provide a measure of the validity of the results, since the holdout observations have zero weight and do not contribute to any of the calculations. The Pearson correlations (0.96 and 0.99) are the ordinary correlation coefficients, and the Kendall Tau's (1.00 and 0.67) are rank based measures of correlation. These correlations should always be large. Subjects whose correlations are small may be unreliable. See Output 3.8.

```
proc corr nosimple noprob kendall pearson
  data=results(where=(weight=0));
  title2 'Holdout Validation Results';
  var a-depend;
  with t-depend;
  bynotsorted _depvar_;
run ;
```

Output 3.8. Holdout Validation Results

Conjoint Analysis of Tea-Tasting Data Holdout Validation Results	
-----	Dependent Variable Transformation(Name)=LINEAR(SUBJ1) -----
Correlation Analysis	
1 'WITH' Variables: T-DEPEND	
1 'VAR' Variables: A-DEPEND	
Pearson Correlation Coefficients / N = 4	
	A_DEPEND
T-DEPEND	0.95969
Dependent Variable Transformation	
Kendall Tau b Correlation Coefficients / N = 4	
	A_DEPEND
T-DEPEND	1.00000
Dependent Variable Transformation	

Conjoint Analysis of Tea-Tasting Data Holdout Validation Results	
-----	Dependent Variable Transformation(Name)=LINEAR(SUBJ2) -----
Correlation Analysis	
1 'WITH' Variables: T-DEPEND	
1 'VAR' Variables: A-DEPEND	
Pearson Correlation Coefficients / N = 4	
	A_DEPEND
T-DEPEND	0.99481
Dependent Variable Transformation	
Kendall Tau b Correlation Coefficients / N = 4	
	A_DEPEND
T-DEPEND	0.66667
Dependent Variable Transformation	

The next steps display simulation observations. The most preferred combinations are printed for each subject. The first subject prefers iced tea with no sugar. The second subject likes lots of sugar. See Output 3.9.

```
proc sort data=results(where=(weight=-1)) out=sims;
  by _depvar_ descending a-depend;
run;

data sims; /* Pull out first 10 for each subject. */
  set sims;
  by _depvar_;
  retain n 0;
  if first._depvar_ then n = 0;
  n = n + 1;
  if n le 10;
  drop weight -depend-t-depend n;
run;

proc print data=sims label;
  by _depvar_ ;
  title2 'Simulations Sorted by Decreasing Predicted Utility';
  title3 'Just the Ten Most Preferred Combinations are Printed';
  label a-depend = 'Predicted Utility';
run;
```

Output 3.9. Simulation Results

Conjoint Analysis of Tea-Tasting Data Simulations Sorted by Decreasing Predicted Utility Just the Ten Most Preferred Combinations are Printed					
----- Dependent Variable Transformation(Name)=LINEAR(SUBJ1) -----					
OBS	Predicted Utility	Lemon:	Temperature:	Sweetness:	Strength:
1	20.3333	Yes	Iced	No Sugar	Moderate
2	20.3333	Yes	Iced	No Sugar	Strong
3	20.0000	Yes	Iced	No Sugar	Weak
4	19.2222	No	Iced	No Sugar	Moderate
5	19.2222	No	Iced	No Sugar	Strong
6	18.8889	No	Iced	No Sugar	Weak
7	18.0000	Yes	Hot	No Sugar	Moderate
8	18.0000	Yes	not	No Sugar	Strong
9	17.6667	Yes	Hot	No Sugar	Weak
10	16.8889	No	Hot	No Sugar	Moderate
----- Dependent Variable Transformation(Name)=LINEAR(SUBJ2) -----					
OBS	Predicted Utility	Lemon:	Temperature:	Sweetness:	Strength:
11	19.8889	No	Iced	2 Teaspoons	Moderate
12	19.5556	No	Iced	2 Teaspoons	Strong
13	19.3889	No	Hot	2 Teaspoons	Moderate
14	19.0556	No	Hot	2 Teaspoons	Strong
15	18.8889	No	Iced	1 Teaspoon	Moderate
16	18.5556	No	Iced	1 Teaspoon	Strong
17	10.4444	Yes	Iced	2 Teaspoons	Moderate
18	18.3889	No	Hot	1 Teaspoon	Moderate
19	18.1111	Yes	Iced	2 Teaspoons	Strong
20	18.0556	No	Hot	1 Teaspoon	Strong

Summarizing Results Across Subjects

Conjoint analyses are performed on an individual basis, but usually the goal is to summarize the results across subjects. The **OUTTEST=** data set contains all of the information in the printed output and can be manipulated to create additional reports including a list of the individual R^2 s and the average of the importance values across subjects. See Output 3.10 for a listing of the variables in the **OUTTEST=** data set.

```
proc content8 data=utils position;
run ;
```

Output 3.10. OUTTEST= Data Set Contents

Conjoint Analysis of Tea-Tasting Data					
CONTENTS PROCEDURE					
Data Set Name: WORK.UTILS		Observations:		38	
Member Type: DATA		Variables:		17	
Engine: SASEB		Indexes:		0	
Created: DDMMYY:00:00:00		Observation Length:		277	
Last Modified: DDMMYY:00:00:00		Deleted Observations:		0	
Protection:		Compressed:		NO	
Data Set Type:		Sorted:		NO	
Label:					
-----Variables Ordered by Position-----					
#	Variable	Type	Len	Pos	Label
1	<u>DEPVAR</u>	Char	18	0	Dependent Variable Transformation(Name)
2	<u>TYPE</u>	Char	8	18	
3	TITLE	Char	80	26	Title
4	VARIABLE	Char	18	106	Variable
5	COEFFICI	Num	8	124	Coefficient
6	STATIST1	Char	24	132	Statistic
7	VALUE	Num	8	156	Value
8	NUMDF	Num	8	164	Num DF
9	DENDF	Num	8	172	Den DF
10	SSQ	Num	8	180	Sum of Squares
11	MEANSQUA	Num	8	188	Mean Square
12	F	Num	8	196	F Value
13	NUMERICP	Num	8	204	Numeric (Approximate) p Value
14	P	Char	9	212	Formatted p Value
15	STDERROR	Num	8	221	Standard Error
16	IYPORTAN	Num	8	229	Importance (% Utility Range)
17	LABEL	Char	80	237	Label

The individual R^2 s are displayed by printing the **VALUE** variable for observations whose **STATISTI** value is R-square. See Output 3.11.

```
proc print data=utils;
  title2 'R-Squares';
  id _depvar_;
  var value;
  where statisti = 'R-square';
run;
```

Output 3.11 . Individual R-Squares

Conjoint Analysis of Tea-Tasting Data R-Squares	
<u>DEPVAR</u>	VALUE
LINEAR(SUBJ1)	0.99789
LINEAR(SUBJ2)	0.95845

The next steps extract the importance values and create a table. The DATA step extracts the importance values and creates row and column labels. The PROC TRANSPOSE step creates a subjects by attributes matrix from a vector (of the number of subjects times the number of attributes values). PROC PRINT displays the importance values, and PROC MEANS displays the average importances. See output 3.12.

```

data im;
  set utils;
  if n(importan); /* Exclude all missing, including specials. */
  _depvar_ = scan(_depvar_,2); /* Discard transformation. */
  name      = scan(label,1,' : '); /* Use first word as var name. */
  label     = scan(label,1,' : '); /* Use up to colon for label. */
  keep importan _depvar_ label name;
run;

proc transpose data=im out=im(drop=_name_ _label_);
  id name;
  id label;
  bynotsorted _depvar_;
  var importan;
  label _depvar_ = 'Subject';
run;

proc print label;
  title2 'Importances';
run;

proc means mean;
  title2 'Average Importances';
run;

```

Output 3.12. Conjoint Analysis Summary Statistics

Conjoint Analysis of Tea-Tasting Data Importance8					
OBS	Subject	Lemon	Temperature	Sweetness	Strength
1	SUBJ1	5.34759	21.6578	71.3904	1.6043
2	SUBJ2	7.00809	12.1294	55.7951	25.0674

Conjoint Analysis of Tea-Tasting Data Average Importances		
Variable	Label	Mean
LEMON	Lemon	6.1778399
TEMPERAT	Temperature	16.8935670
SWEETNES	Sweetness	63.5927613
STRENGTH	Strength	13.3358318

Example 4. Spaghetti Sauce

This example uses conjoint analysis in a study of spaghetti sauce preferences. The goal is to investigate the main effects for all of the attributes and the interaction of brand and price, and to simulate market share. Rating scale data are gathered interactively from a group of subjects. The example has nine parts.

1. A nonorthogonal experimental design is generated with PROC OPTEX.
2. Descriptions of the spaghetti sauces are generated.
3. A screen layout is defined for data collection with PROC FSEDIT.
4. PROC FSEDIT is used to collect the data.
5. The metric conjoint analysis is performed with PROC TRANSREG.
6. Market share is simulated with the maximum utility model.
7. Market share is simulated with the Bradley-Terry-Lute and logit models.
8. Change in market share is investigated.
9. Brand by price interactions are plotted.

Create a Nonorthogonal Design Matrix with PROC OPTEX

In this example, subjects were asked to rate their interest in purchasing hypothetical spaghetti sauces. Table 1 shows the attributes, the attribute levels, and the number of *df* associated with each effect. The brand names X, Y, and Z are artificial. Usually, real brand names would be used—your client's brand and competitors' brands. The absence of a feature (for example, no mushrooms) is not mentioned in the product description, hence the "No Mention" in Table 1.

In this design there are 21 model *df* plus 1 for the intercept. A design with more than 22 runs must be generated if there are to be error *df*. A good rule of thumb is to limit the design size to 30 runs. In this example, this rule was violated. In order to have two observations in each of the 18 brand by price cells, a design with 36 runs was generated. When subjects are required to make many judgments, there is the risk that the quality of the data will be poor. Caution should be used when generating designs with this many runs.

Table 1 Experimental Design

Effects	Levels	<i>df</i>
Intercept		1
Brand	X, Y, Z,	2
Meat Content	Vegetarian, Hamburger, Italian Sausage	2
Mushroom Content	Mushrooms, No Mention	1
Natural Ingredients	All Natural Ingredients, No Mention	1
Price	\$1.50, \$1.75, \$1.99, \$2.00, \$2.25, \$2.49	5
Brand x Price		10
Error		14
Total Runs		36

Orthogonal arrays do not exist for designs such as this one, with interactions and 2, 3, and 6 level factors, so a nonorthogonal design will be used. First, PROC PLAN generates the full-factorial design. The FACTORS statement names the attributes and the number of levels of each attribute. The OUTPUT statement is used to name the SAS data set, character values (CVALS) for some attributes, and numeric values (NVALS) for price. The resulting data set has $3 \times 3 \times 2 \times 2 \times 6 = 216$ observations.


```

title 'Conjoint Analysis of Spaghetti Sauce Data';

* Generate a full-factorial design;
proc plan ordered;
  factors brand=3 meat=3 mushroom=2 natural=2 price=6 / noprint;
  output out=design1
    brand      cvals=('X' 'Y' 'Z')
    meat       cvals=('Vegetarian' 'Hamburger' 'Italian Sausage')
    mushroom   cvals=('Mushrooms' 'No Mention')
    natural    cvals=('All Natural Ingredients' 'No Mention')
    price      nvals=(1.50 1.75 1.99 2.00 2.25 2.49);
run;
quit;

```

Then a DATA step eliminates unrealistic combinations. Specifically, combinations at \$1.50 with meat and Italian Sausage with All Natural Ingredients are eliminated. This data set, with 162 observations, will be the candidate set for creating a nonorthogonal design.

```

title 'Conjoint Analysis of Spaghetti Sauce Data';

* Exclude unrealistic combinations;
data design2;
  set design1;
  * Note, =: is the 'begins with' operator;
  if (meat =: 'Ham' or meat =: 'Ita') and price < 1.51
    then delete;
  if meat =: 'Ita' and natural =: 'All' then delete;
run;

```

PROC OPTEX generates the design. The data set of candidate points and seed for the random number generator are specified on the PROC statement. A seed is explicitly set in this example so that you can reproduce the results. A CLASS statement names the factors, and a MODEL statement names the effects to be estimated, including all main effects and the brand by price interaction. The GENERATE statement requests a design with N=36 runs. Two hundred designs are generated using the modified Federov algorithm. This algorithm is slower than the default (METHOD=EXCHANGE) method but is better at finding an efficient design. The most efficient design is output with the OUTPUT statement and printed. In the interest of space, only the efficiency criteria from the first ten designs are shown here.

```

title 'Conjoint Analysis of Spaghetti Sauce Data';

proc optex data=design2 seed=123;
  class price brand meat mushroom natural;
  model price brand brand*price meat mushroom natural;
  generate n=36 iter=200 method=m_federov;
  output out=sasuser.design;
run;

```

Output 4.1. Experimental Design Creation (First Ten)

Conjoint Analysis of Spaghetti Sauce Data				
Design Number	D-efficiency	A-efficiency	Q-efficiency	Average Prediction Standard Error
1	24.9031	15.0497	86.3655	0.8172
2	24.9031	14.9610	66.3655	0.6172
3	24.9031	14.9505	66.3655	0.8172
4	24.9031	14.9208	86.3655	0.8172
5	24.9031	14.8884	86.3655	0.8172
6	24.9031	14.8877	86.3655	0.8172
7	24.9031	14.8668	86.3655	0.8172
8	24.9031	14.8504	86.3655	0.8172
9	24.9031	14.8444	86.3655	0.8172
10	24.9031	14.8418	86.3655	0.8172

The design matrix is shown in Output 4.2; the design is not perfectly balanced. The frequencies for the MEAT levels are not all 12, and the frequencies for the NATURAL levels are not all 18.

```
proc print data=sasuser.design;
run:

proc freq data=sasuser.design;
  title2 'Report on Balance';
  tables price -- natural brand*price;
run;
```

Output 4.2. Experimental Design

Conjoint Analysis of Spaghetti Sauce Data						
OBS	PRICE	BRAND	MEAT	MUSHROOM	NATURAL	
1	2.49	Z	Vegetarian	Mushrooms	No Mention	
2	2.49	Z	Italian Sausage	No Mention	No Mention	
3	2.49	Y	Vegetarian	No Mention	No Mention	
4	2.49	Y	Italian Sausage	Mushrooms	No Mention	
5	2.49	X	Vegetarian	Mushrooms	No Mention	
6	2.49	X	Italian Sausage	No Mention	No Mention	
7	2.25	Z	Italian Sausage	Mushrooms	No Mention	
8	2.25	Z	Hamburger	No Mention	All Natural	Ingredients
9	2.25	Y	Vegetarian	No Mention	All Natural	Ingredients
10	2.25	Y	Hamburger	Mushrooms	No Mention	
11	2.25	X	Vegetarian	No Mention	All Natural	Ingredients
12	2.25	X	Hamburger	Mushrooms	No Mention	
13	1.99	Z	Vegetarian	Mushrooms	All Natural	Ingredients
14	1.99	Z	Italian Sausage	No Mention	No Mention	
15	1.99	Y	Vegetarian	Mushrooms	All Natural	Ingredients
16	1.99	Y	Hamburger	No Mention	No Mention	
17	1.99	X	Vegetarian	No Mention	No Mention	
18	1.99	X	Hamburger	Mushrooms	All Natural	Ingredients
19	1.75	Z	Italian Sausage	No Mention	No Mention	
20	1.75	Z	Hamburger	Mushrooms	All Natural	Ingredients
21	1.75	Y	Vegetarian	No Mention	No Mention	
22	1.75	Y	Hamburger	Mushrooms	All Natural	Ingredients
23	1.75	X	Vegetarian	No Mention	No Mention	
24	1.75	X	Hamburger	Mushrooms	All Natural	Ingredients
25	1.50	Z	Vegetarian	No Mention	All Natural	Ingredients
26	1.50	Z	Vegetarian	Mushrooms	No Mention	
27	1.50	Y	Vegetarian	No Mention	All Natural	Ingredients
28	1.50	Y	Vegetarian	Mushrooms	No Mention	
29	1.50	X	Vegetarian	No Mention	All Natural	Ingredients
30	1.50	X	Vegetarian	Mushrooms	No Mention	
31	2.00	Z	Italian Sausage	Mushrooms	No Mention	
32	2.00	Z	Hamburger	No Mention	All Natural	Ingredients
33	2.00	Y	Italian Sausage	Mushrooms	No Mention	
34	2.00	Y	Hamburger	No Mention	No Mention	
35	2.00	X	Vegetarian	Mushrooms	All Natural	Ingredients
36	2.00	X	Hamburger	No Mention	No Mention	

Output 4.2. (Continued)

Conjoint Analysis of Spaghetti Sauce Data
Report on Balance

PRICE	Frequency	Percent	Cumulative Frequency	Cumulative Percent
1.5	6	16.7	6	16.7
1.75	6	16.7	12	33.3
1.99	6	16.7	18	50.0
2	6	16.7	24	66.7
2.25	6	16.7	30	83.3
2.49	6	16.7	36	100.0

BRAND	Frequency	Percent	Cumulative Frequency	Cumulative Percent
X	12	33.3	12	33.3
Y	12	33.3	24	66.7
Z	12	33.3	36	100.0

MEAT	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Hamburger	11	30.6	11	30.6
Italian Sausage	8	22.2	19	52.8
Vegetarian	17	47.2	36	100.0

Conjoint Analysis of Spaghetti Sauce Data
Report on Balance

MUSHROOM	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Mushrooms	18	50.0	18	50.0
No Mention	18	50.0	36	100.0

NATURAL	Frequency	Percent	Cumulative Frequency	Cumulative Percent
All Natural Ingr	14	38.9	14	38.9
No Mention	22	61.1	36	100.0

Output 4.2. (Continued)

Conjoint Analysis of Spaghetti Sauce Data Report on Balance								
TABLE OF BRAND BY PRICE								
BRAND	PRICE							
Frequency I								
Percent I								
Row Pet I								
col Pet	1	2	3	4	5	6	7	Total
X	1.51	1.751	1.991	2	2.251	2.491		
	5.56	5.56	5.56	5.56	5.56	5.56		12
	16.67	16.67	16.67	16.67	16.67	16.67		33.33
	33.33	33.33	33.33	33.33	33.33	33.33		
Y	2	2	2	2	2	2		12
	5.56	5.56	5.56	5.56	5.56	5.56		33.33
	16.67	16.67	16.67	16.67	16.67	16.67		
	33.33	33.33	33.33	33.33	33.33	33.33		
Z	2	2	2	2	2	2		12
	5.56	5.56	5.56	5.56	5.56	5.56		33.33
	16.67	16.67	16.67	16.67	16.67	16.67		
	33.33	33.33	33.33	33.33	33.33	33.33		
Total	6	6	6	6	6	6		36
	16.67	16.67	16.67	16.67	16.67	16.67		100.00

Generate Descriptions of the Sauces

Next, preparations are made for data collection. First, an observation number is added to the design. The combinations will be presented in a random order, and the observation number will be used later to restore the original order. The next DATA step creates three-line, verbal descriptions of the combinations in the design. Here is an example:

Try Brand Z Vegetarian spaghetti sauce, now with Mushrooms and All Natural Ingredients. A 26 ounce jar serves four adults for only \$1.99.

Remember that “No Mention” is not mentioned.

```

data sasuser.design;
  set sasuser.design;
  nob5 = _n_;
run;

data sasuser.lines;
  set sasuser.design;
  length line $ 200 line1-line3 $ 60;
  response = .;

  * Create a product description:

  line = 'Try Brand ' || trim(branch) || ' ' || trim(meat) || ' ' ||
        ' spaghetti sauce';

  n = (natural =: 'All');
  m = (mushroom =: 'Mus');

  if n or m then do;
    line = trim(line) || ', now with';

    if m then do;
      line = trim(line) || ' ' || mushroom;
      if n then line = trim(line) || ' and';
    end;
    if n then line = trim(line) || ' ' || natural;
  end;

  line = trim(line) ||
        '. A 26 ounce jar serves four adults for only $' ||
        put(price,4.2) || '.';

  * Break up description;

  do i = 60 to 1 by -1 while(substr(line,i,1) ne ' '); end;
  line1 = substr(line,1,i);
  i = i;

  do i = 1 + 60 to 1 by -1 while(substr(line,i,1) ne ' '); end;
  line2 = substr(line,i+1,i - 1);
  i = i;

  do i = 1 + 60 to 1 by -1 while(substr(line,i,1) ne ' '); end;
  line3 = substr(line,i+1,i - 1);

  keep line1-line3 response nob5;

run;

```

Prepare For Data Collection with PROC FSEDIT

Data can be collected interactively in the SAS System using the FSEDIT procedure in SAS/FSP software. This section illustrates the initial setup, and the data collection is described in the next section. If you do not plan on using the FSEDIT procedure, you can create an input data set like the one in Output 4.3, and skip ahead to the “Metric Conjoint Analysis” section.

First, use PROC FSEDIT to create a customized screen layout for data collection. Option selection varies across operating systems and terminals. For some terminals it may be necessary to press RETURN instead of pressing ENTER or clicking a mouse. For example, the instruction ‘select OK’ means place the cursor on OK and press ENTER, or place the cursor on OK and press RETURN, or place the cursor

or mouse pointer on OK and click the mouse.

1. Run PROC FSEDIT. The SCREEN= option specifies a name for your customized screen.

```
proc fseedit data=sasuser.lines screen=sasuser.profile.lines.screen;
  var line1-line3 response;
run;
```

2. To customize, do the following:
 - a. Select Locals.
 - b. Select Modify screen. . . .
 - c. Select OK.
 - d. Select 2 Screen Modification and Field Identification.
 - e. Select Edit.
 - f. Select Options.
 - g. Select Numbers.
3. You are now in display manager mode and can insert and delete lines and columns to better position the variables. Use standard display manager editor commands. You can provide a better description than RESPONSE: (such as Enter Purchase Interest:) and blank out the strings LINE1:,LINE2:, and LINE3:. Furthermore, you can enter instructions. The following is a sample of a customized screen:

Enter Purchase Interest: _____

```

Purchase Interest
9 Definitely Will Buy
8
7 Probably Will Buy
6
5 Might or Might Not Buy
4
3 Probably Will Not Buy
  ---
2
1 Definitely Will Not Buy
  ---

```

4. When you have finished customizing the appearance, exit the display manager mode:
 - a. Select File.
 - b. Select End.
5. You will be asked if you created any computational or repeated fields. Type "N" at the underscore. If it will not accept your N, try deleting any characters such as blanks that are at the prompt, then type the "N" again.
6. You will be asked to identify the fields:
 - a. Put your cursor on the first underscore of the LINE1 field and press ENTER.
 - b. Put your cursor on the first underscore of the LINE2 field and press ENTER.
 - c. Do the same for LINE3 and RESPONSE.

You should get a note that all fields are identified.

7. Now modify the parameters of the fields:
 - a. Select View.
 - b. Select End.
 - c. Select 5 Modification of General Parameters.
8. You can overwrite parameters, such as the window size.


```
Window rows:      45
Window cols:      80
```
9. Exit this screen.
 - a. Select View.
 - b. Select End.
10. Select 4 Assign **Special** Attributes to Fields.
11. Now assign field attributes. The first attribute will be INITIAL. Press Page Down and Page Up to cycle through the attributes.
 - a. For MAXIMUM, type “9” over the first response underscore.
 - b. For MINIMUM, type “1” over the first response underscore.
 - c. For REQUIRED, type “R” over the first response underscore.
 - d. For CAPS, type “_” over the C’s.
 - e. Set the colors if you wish.
 - f. Skip PAD and go on to PROTECT.
 - g. For PROTECT, type “P” over the first underscores in the line fields.
 - h. Go back to PAD.
 - i. For PAD, type “ ” over the first underscore of the three line fields.
 - j. Select View.
 - k. Select End.
 - l. Select Goback.
12. You have now finished customizing your FSEDIT application. The next time you run the following, you should get your customized application:

```
proc fsedit data=lines screen=sasuser.profile.lines.screen;
  var line1-line3 response;
run;
```

Collecting Data Using PROC FSEDIT

This section shows how data can be collected interactively using the FSEDIT procedure. Each subject sits at a computer and directly enters his or her data. The combinations are presented in a different random order to each subject. To make copies of your design sorted into different random orders, use the following macro. In the interest of space, data from only eight subjects will be analyzed.

```

%macro make(n);
%do i = 1 %to &n;

    data sasuser.s&i;
        set sasuser.lines;
        r = uniform(0);
        run;

    proc sort;
        by r;
        run;
    %end;

%mend;

%make(8)

```

To collect data for subject 1, run:

```

proc fsedit data=sasuser.s1 screen=sasuser.profile.lines.screen;
    var line1-line3 response;
    run;

```

For subject 2, change s1 to s2, and so on:

```

proc fsedit data=sasuser.s2 screen=sasuser.profile.lines.screen;
    var line1-line3 response;
    run;

```

The subject presses Page Up and Page Down to cycle through the observations, and selects File and End to exit.

After the data are collected, use the next step to assemble the individual data sets back into one data set. New variables are added for the brand by price interactions. PRICE_X equals PRICE when BRAND is X, PRICE_Y equals PRICE when BRAND is Y, and PRICE_Z equals PRICE when BRAND is Z; the variables are zero otherwise. See Output 4.3.

```

%macro combine(n);
options nonotes;
%do i = 1 %to &n;
    proc sort data=sasuser.s&i out=s&i(keep=nobs response);
        by nob;
        run;
    %end;
options notes;

data sasuser.all;
    merge sasuser.design
        %do i = 1 %to &n; s&i(rename=(response=sub&i)) %end;;

    * Create variables for brand by price interactions;
    pricex = (brand = 'X') * price;
    pricey = (brand = 'Y') * price;
    pricez = (brand = 'Z') * price;
    by nob;
    run;

%mend;

```

```

%combine(8)

* Here are the data for this example;
data _null_;
  title2 'Raw Input Data';
  set sasuser.all;
  file print;
  put price 4.2 +1 brand $2. meat $16. mushroom $16. natural $24.
    (sub1-sub8) (1.);
run;

* Brand by price interaction variables;
proc print data=sasuser.all;
  title2 "Brand by Price Interaction Variables";
  var brand price;;
run;

```

Output 4.3. Conjoint Input Data

Conjoint Analysis of Spaghetti Sauce Data					
Raw Input Data					
2.49	Z	Vegetarian	Mushrooms	No Mention	84511364
2.49	Z	Italian Sausage	No Mention	No Mention	13131213
2.49	Y	Vegetarian	No Mention	No Mention	64541164
2.49	Y	Italian Sausage	Mushrooms	No Mention	13111114
2.49	X	Vegetarian	Mushrooms	No Mention	95511564
2.49	X	Italian Sausage	No Mention	No Mention	14131112
2.25	Z	Italian Sausage	Mushrooms	No Mention	13111114
2.25	Z	Hamburger	No Mention	All Natural Ingredients	16541214
2.25	Y	Vegetarian	No Mention	All Natural Ingredients	76151275
2.25	Y	Hamburger	Mushrooms	No Mention	15711513
2.25	X	Vegetarian	No Mention	All Natural Ingredients	15541575
2.25	X	Hamburger	Mushrooms	No Mention	15511614
1.99	Z	Vegetarian	Mushrooms	All Natural Ingredients	96513574
1.99	Z	Italian Sausage	No Mention	No Mention	14163114
1.99	Y	Vegetarian	Mushrooms	All Natural Ingredients	91713675
1.99	Y	Hamburger	No Mention	No Mention	15763513
1.99	X	Vegetarian	No Mention	No Mention	65573576
1.99	X	Hamburger	Mushrooms	All Natural Ingredients	10713617
1.75	Z	Italian Sausage	No Mention	No Mention	13194116
1.75	Z	Hamburger	Mushrooms	All Natural Ingredients	17713616
1.75	Y	Vegetarian	No Mention	No Mention	65993783
1.15	Y	Hamburger	Mushrooms	All Natural Ingredients	16515416
1.15	X	Vegetarian	No Mention	No Mention	55173404
1.75	X	Hamburger	Mushrooms	All Natural Ingredients	17715418
1.50	Z	Vegetarian	No Mention	All Natural Ingredients	97196497
1.50	Z	Vegetarian	Mushrooms	No Mention	97916796
1.50	Y	Vegetarian	No Mention	All Natural Ingredients	86796797
1.50	Y	Vegetarian	Mushrooms	No Mention	95716797
1.50	X	Vegetarian	No Mention	All Natural Ingredients	76796497
1.50	X	Vegetarian	Mushrooms	No Mention	95116191
2.00	Z	Italian Sausage	Mushrooms	No Mention	13113714
2.00	Z	Hamburger	No Mention	All Natural Ingredients	17762515
2.00	Y	Italian Sausage	Mushrooms	No Mention	13113116
2.00	Y	Hamburger	No Mention	No Mention	15554515
2.00	X	Vegetarian	Mushrooms	All Natural Ingredients	97713675
2.00	X	Hamburger	No Mention	No Mention	15764315

Output 4.3. (Continued)

Conjoint Analysis of Spaghetti Sauce Data Brand by Price Interaction Variables					
OBS	BRAND	PRICE	PRICEX	PRICEY	PRICEZ
1	Z	2.49	0.00	0.00	2.49
2	Z	2.49	0.00	0.00	2.49
3	Y	2.49	0.00	2.49	0.00
4	Y	2.49	0.00	2.49	0.00
5	X	2.49	2.49	0.00	0.00
6	X	2.49	2.49	0.00	0.00
7	Z	2.25	0.00	0.00	2.25
8	Z	2.25	0.00	0.00	2.25
9	Y	2.25	0.00	2.25	0.00
10	Y	2.25	0.00	2.25	0.00
11	X	2.25	2.25	0.00	0.00
12	X	2.25	2.25	0.00	0.00
13	Z	1.99	0.00	0.00	1.99
14	Z	1.99	0.00	0.00	1.99
15	Y	1.99	0.00	1.99	0.00
16	Y	1.99	0.00	1.99	0.00
17	X	1.99	1.99	0.00	0.00
18	X	1.99	1.99	0.00	0.00
19	Z	1.75	0.00	0.00	1.75
20	Z	1.75	0.00	0.00	1.15
21	Y	1.75	0.00	1.75	0.00
22	Y	1.75	0.00	1.75	0.00
23	X	1.75	1.15	0.00	0.00
24	X	1.75	1.75	0.00	0.00
25	Z	1.50	0.00	0.00	1.50
26	Z	1.50	0.00	0.00	1.50
27	Y	1.50	0.00	1.50	0.00
28	Y	1.50	0.00	1.50	0.00
29	X	1.50	1.50	0.00	0.00
30	X	1.50	1.50	0.00	0.00
31	Z	2.00	0.00	0.00	2.00
32	Z	2.00	0.00	0.00	2.00
33	Y	2.00	0.00	2.00	0.00
34	Y	2.00	0.00	2.00	0.00
35	X	2.00	2.00	0.00	0.00

Conjoint Analysis of Spaghetti Sauce Data Brand by Price Interaction Variables					
OBS	BRAND	PRICE	PRICEX	PRICEY	PRICEZ
36	X	2	2	0	0

Metric Conjoint Analysis

Now that the data have been collected and stored in a SAS data set, the metric conjoint analysis can be performed. In this example, as in many real-life examples, the conjoint analysis is not the primary goal. The conjoint analysis is used to generate the input for the market share simulator, which is described in the next section.

The TRANSREG procedure TEST option prints an ANOVA table for each subject (see Output 4.4). In the interest of space, UTILITIES is not specified. METHOD=MORALS is the default when there are independent variable transformations but not for all conjoint analyses. It was specified here to emphasize that a METHOD=MORALS output data set is needed for the simulation steps. The DUMMY option requests the canonical initialization. The brand by price interactions are quadratic functions with discontinuities at \$1.995. PRICEX, PRICEY, and

PRICEZ each use 5 *df* due to the linear term, the quadratic term, the change in intercept, the linear change, and the quadratic change.

```
* Fit a conjoint analysis for each subject individually;
proc transreg data=sasuser.all test short method=morals;
  title2 'Individual Conjoint Analyses';
  model linear(sub1-sub8) =
    spline(pricex pricey pricez /
      degree=2 knots=1.995 1.995 1.995)
    class(brand meat mushroom natural / zero=sum) / dummy;
  output out=utils dapproximations ireplace;
  id price;
run;
```

Output 4.4. Metric Conjoint Analysis

Conjoint Analysis of Spaghetti Sauce Data Individual Conjoint Analyses					
The TRANSREG Procedure Hypothesis Tests for LINEAR(SUB1)					
Univariate ANOVA Table Based on the Usual Degrees of Freedom					
Source	DF	sum of Squares	Mean Square	F Value	p
Model	21	421.545980	20.073618	101.838	0.0001
Error	14	2.759576	0.197113		
Total	35	424.305556			
	Root MSE	0.4439736	R-square	0.99350	
	Dep Mean	4.1388889	Adj R-sq	0.98374	
	cv	10.726878			

Conjoint Analysis of Spaghetti Sauce Data Individual Conjoint Analyses					
The TRANSREG Procedure Hypothesis Tests for LINEAR(SUB2)					
Univariate ANOVA Table Based on the Usual Degrees of Freedom					
Source	DF	sum of Squares	Mean Square	F Value	p
Model	21	73.3460145	3.4926674	14.365	0.0001
Error	14	3.4039855	0.2431410		
Total	35	76.7500000			
	Root MSE	0.4930941	R-square	0.95565	
	Dep Mean	5.25	Adj R-sq	0.88912	
	cv	9.3922692			

Output 4.4. (Continued)

Conjoint Analysis of Spaghetti Sauce Data Individual Conjoint Analyses					
The TRANSREG Procedure Hypothesis Tests for LINEAR(SUB3)					
Univariate ANOVA Table Based on the Usual Degrees of Freedom					
Source	DF	sum of Squares	Mean Square	F Value	P
Model	21	207.388889	9.875661	9.321	0.0001
Error	14	14.833333	1.059524		
Total	35	222.222222			
	Root MSE	1.0293317	R-square	0.93325	
	Dep Mean	5.2222222	Adj R-sq	0.63312	
	cv	19.710609			

Conjoint Analysis of Spaghetti Sauce Data Individual Conjoint Analyses					
The TRANSREG Procedure Hypothesis Tests for LINEAR(SUB4)					
Univariate ANOVA Table Based on the Usual Degrees of Freedom					
Source	DF	sum of Squares	Mean Square	F Value	P
Model	21	293.497067	13.976051	5.042	0.0016
Error	14	38.808499	2.772035		
Total	35	332.305556			
	Root MSE	1.6649429	R-square	0.88321	
	Dep Mean	3.6388689	Adj R-sq	0.70804	
	cv	45.754156			

Conjoint Analysis of Spaghetti Sauce Data Individual Conjoint Analyses					
The TRANSREG Procedure Hypothesis Tests for LINEAR(SUB5)					
Univariate ANOVA Table Based on the Usual Degrees of Freedom					
Source	DF	sum of Squares	Mean Square	F Value	P
Model	21	109.344138	5.206964	15.657	0.0001
Error	14	4.655662	0.332562		
Total	35	114.000000			
	Root MSE	0.5766815	R-square	0.95916	
	Dep Mean		Adj R-sq	0.89790	
	cv	19.2227173			

Output 4.4. (Continued)

Conjoint Analysis of Spaghetti Sauce Data Individual Conjoint Analyses					
The TRANSREG Procedure Hypothesis Tests for LINEAR(SUB6)					
Univariate ANOVA Table Based on the Usual Degrees of Freedom					
Source	DF	Sum of Squares	Mean Square	F Value	p
Model	21	121.727851	5.796564	2.491	0.0417
Error	14	32.577704	2.326979		
Total	35	154.305556			
	Root MSE	1.5254438	R-square	0.78888	
	Dep Mean	4.3611111	Adj R-sq	0.47219	
	cv	34.97833			

Conjoint Analysis of Spaghetti Sauce Data Individual Conjoint Analyses					
The TRANSREG Procedure Hypothesis Tests for LINEAR(SUB7)					
Univariate ANOVA Table Based on the Usual Degrees of Freedom					
Source	DF	sum of Squares	Mean Square	F Value	p
Model	21	416.634683	19.839747	166.234	0.0001
Error	14	1.670872	0.119348		
Total	35	418.305556			
	Root MSE	0.3454678	R-square	0.99601	
	Dep Mean	4.1388889	Adj R-sq	0.99001	
	cv	8.3468734			

Conjoint Analysis of Spaghetti Sauce Data Individual Conjoint Analyses					
The TRANSREG Procedure Hypothesis Tests for LINEAR(SUB8)					
Univariate ANOVA Table Based on the Usual Degrees of Freedom					
Source	DF	sum of Squares	Mean Square	F Value	p
Model	21	63.9343728	3.0444939	3.862	0.0062
Error	14	11.0378494	0.7884178		
Total	35	74.9722222			
	Root MSE	0.8879289	R-square	0.85277	
	Dep Mean	4.9722222	Adj R-sq	0.63194	
	cv	17.857789			

Simulating Market Share, Maximum Utility Model

This section shows how to use the utilities from a conjoint analysis to simulate choice and predict market share. The end result for a hypothetical product is its expected market share, which is a prediction of the proportion of times that the product will be purchased. A SAS macro is used to simulate market share. It takes a METHOD=MORALS output data set from PROC TRANSREG and creates a data set with expected market share for each combination. First, market share is computed with the maximum utility model, which assumes each subject would always buy the product with the highest utility. If two or more products have the same maximum utility, this model assumes that the subject would randomly choose between them with equal probability. The macro finds the most preferred combination(s) for each subject, which are those combinations with the largest total utility, and assigns the probability that each combination will be purchased. When there is no tie for the maximum utility within a subject, that subject will have one probability of 1.0 and the rest will be zero. When two utilities are tied for the maximum, that subject will have two probabilities of 0.5 and the rest will be zero. The probabilities are then averaged across subjects for each combination to get market share. Subjects can be differentially weighted.

```

                                /*-----*/
                                /* Simulate Market Share */
                                /*-----*/
%macro sim(data=_last_, /* SAS data set with utilities. */
            idvars=, /* Additional variables to display with */
            weights=, /* market share results. */
                  /* By default, each subject contributes */
                  /* equally to the market share */
                  /* computations. To differentially */
                  /* weight the subjects, specify a vector */
                  /* of weights, one per subject. */
                  /* Separate the weights by blanks. */
            out=shares, /* Output data set name. */
            method=max /* max - maximum utility model. */
                      /* btl - Bradley-Terry-Lute model. */
                      /* logit - logit model. */
                      /* WARNING: The Bradley-Terry-Lute model */
                      /* and the logit model results are not */
                      /* invariant under linear */
                      /* transformations of the utilities. */
                      /*-----*/
);

options nonotes;

%if &method = btl or &method = logit %then
    %put WARNING: The Bradley-Terry-Lute model and the logit model
    results are not invariant under linear transformations of the
    utilities.;
%else %if &method ne max %then %do;
    %put WARNING: Invalid method &method.. Assuming method=max.;
    %let method = max;
%end;

* Eliminate coefficient observations, if any;
data temp1;
    set &data(where=(_type_ = 'SCORE' or _type_ = ' '));

```



```

run;

* Determine number of runs and subjects.;
proc sql;
  create table temp2 as select nruns,
    count(nruns9 as nsubs, count(distinct nruns) as chk
    from (select count( _depvar_ ) as nruns
    from temp1 where _type_ in ('SCORE',' '9 group by _depvar_);
quit;

data _null_;
  set temp2;
  call symput('nruns',compress(put(nruns,5.0)));
  call symput('nsubs',compress(put(nsubs,5.0)));
  if chk > 1 then do;
    put 'ERROR: Corrupt input data set.';
    call symput('okay','no');
  end;
  else call symput('okay','yes');
run;

%if &okay ne yes %then %do;
  proc print;
    title2 'Number of runs should be constant across subjects';
  run;
  %goto endit;
%end;

%else %put NOTE: &nruns runs and &nsubs subjects.;

%let w = %scan(&weights,%eval(&nsubs + 1),%str( ));
%if %length(&w) > 0 %then %do;
  %put ERROR: Too many weights.;
  %goto endit;
%end;

* Form nruns by nsubs data set of utilities:
data temp2;
  keep _u1 - _u&nsubs &idvars;
  array u[&nsubs] _u1 - _u&nsubs;

  do j = 1 to Grnruns;

    * Read ID variables;
    set temp1(keep=&idvars) point = j;

    * Read utilities;
    k = j;
    do i = 1 to &nsubs;
      set temp1(keep=a_depend) point = k;
      u[i] = a_depend;
      %if &method = logit %then u[i] = exp(u[i]);;
      k = k + &nruns;
    end;

    output;
  end;

stop;
run;

* Set up for maximum utility model;
%if &method = max %then %do;

  * Compute maximum utility for each subject;
  proc means data=temp2 noprint;

```

52 □ □ □ Conjoint Analysis Examples

```

var _u1-_u&nsubs;
output out=templ max=_sum1 - _sum&nsubs;
run:

* Flag maximum utility;
data temp2(keep=_u1 - _u&nsubs &idvars);
  if _n_ = 1 then set temp1(drop=_type_ _freq_);
  array u[&nsubs] _u1 - _u&nsubs;
  array m[&nsubs] _sum1 - _sum&nsubs;
  set temp2;
  do i = 1 to &nsubs;
    u[i] = ((u[i] - m[i]) > -1e-8); /* < 1e-8 is considered 0 */
  end;
run:

%end;

* Compute sum for each subject;
proc means data=temp2 noprint;
  var _u1-_u&nsubs;
  output out=templ sum=_sum1 - _sum&nsubs;
run;

* Compute expected market share;
data &out(keep=share &idvars);
  if _n_ = 1 then set temp1(drop=_type_ _freq_);
  array u[&nsubs] _u1 - _u&nsubs;
  array m[&nsubs] _sum1 - _sum&nsubs;
  set temp2;

  * Compute final probabilities;

  do i = 1 to &nsubs;
    u[i] = u[i] / m[i];
  end;

  * Compute expected market share;

  %if %length(&weights) = 0 %then %do;
    share = mean(of _u1 - _u&nsubs);
  %end;

  %else %do;
    share = 0;
    wsum = 0;
    %do i = 1 %to &nsubs;
      %let w = %scan(&weights,&i,%str( ));
      %if %length(&w) = 0 %then %let w = .;
      if &w < 0 then do;
        if _n_ > 1 then stop;
        put "ERROR: Invalid weight &w..";
        call symput('okay','no');
      end;
      share = share + &w * _u&i;
      wsum = wsum + &w;
    %end;
    share = share / wsum;
  %end;
run:

options notes;

%if &okay ne yes %then %goto endit;

proc sort;
  by descending share &idvars;

```

```

run:

proc print:
  title2 'Expected Market Share';
  title3 %if          &method = max%then "Maximum Utility Model";
         %else %if &method = btl %then "Bradley-Terry-Lute Model";
         %else
         "Logit Model";
run:

%endit:

%mend;

%sim(data=utils,out=maxutils,method=max,
      idvars=price brand meatmushroom natural);

```

The largest market share (25%) is for Brand Z, vegetarian sauce with mushrooms, costing \$1.50. The next largest share (18.75%) is Brand X sauce, with hamburger, mushrooms, and all natural ingredients, costing \$1.99. Only eight combinations have an expected market share greater than zero. See Output 4.5.

Output 4.5. Market Share Simulation

Conjoint Analysis of Spaghetti Sauce Data							
Expected Market Share							
Maximum Utility Model							
OBS	BRAND	MEAT	MUSHROOM	NATURAL		PRICE	SHARE
1	z	Vegetarian	Mushrooms	No Mention		1.50	0.25000
2	x	Hamburger	Mushrooms	All Natural	Ingredients	1.99	0.16750
3	y	Vegetarian	Mushrooms	No Mention		1.50	0.12500
4	y	Vegetarian	No Mention	No Mention		1.75	0.12500
5	x	Vegetarian	No Mention	All Natural	Ingredients	1.50	0.08333
6	y	Vegetarian	No Mention	All Natural	Ingredients	1.50	0.08333
7	z	Vegetarian	No Mention	All Natural	Ingredients	1.50	0.08333
8	y	Hamburger	Mushrooms	All Natural	Ingredients	1.75	0.06250
9	x	Vegetarian	Mushrooms	No Mention		1.50	0.00000
10	x	Hamburger	Mushrooms	All Natural	Ingredients	1.75	0.00000
11	x	Vegetarian	No Mention	No Mention		1.75	0.00000
12	z	Hamburger	Mushrooms	All Natural	Ingredients	1.75	0.00000
13	z	Italian Sausage	No Mention	No Mention		1.75	0.00000
14	x	Vegetarian	No Mention	No Mention		1.99	0.00000
15	y	Hamburger	No Mention	No Mention		1.99	0.00000
16	y	Vegetarian	Mushrooms	All Natural	Ingredients	1.99	0.00000
17	z	Italian Sausage	No Mention	No Mention		1.99	0.00000
18	z	Vegetarian	Mushrooms	All Natural	Ingredients	1.99	0.00000
19	x	Hamburger	No Mention	No Mention		2.00	0.00000
20	x	Vegetarian	Mushrooms	All Natural	Ingredients	2.00	0.00000
21	y	Hamburger	No Mention	No Mention		2.00	0.00000
22	y	Italian Sausage	Mushrooms	No Mention		2.00	0.00000
23	z	Hamburger	No Mention	All Natural	Ingredients	2.00	0.00000
24	z	Italian Sausage	Mushrooms	No Mention		2.00	0.00000
25	x	Hamburger	Mushrooms	No Mention		2.25	0.00000
26	x	Vegetarian	No Mention	All Natural	Ingredients	2.25	0.00000
27	y	Hamburger	Mushrooms	No Mention		2.25	0.00000
28	y	Vegetarian	No Mention	All Natural	Ingredients	2.25	0.00000
29	z	Hamburger	No Mention	All Natural	Ingredients	2.25	0.00000
30	z	Italian Sausage	Mushrooms	No Mention		2.25	0.00000
31	x	Italian Sausage	No Mention	No Mention		2.49	0.00000
32	x	Vegetarian	Mushrooms	No Mention		2.49	0.00000
33	y	Italian Sausage	Mushrooms	No Mention		2.49	0.00000
34	y	Vegetarian	No Mention	No Mention		2.49	0.00000

Output 4.5. (Continued)

Conjoint Analysis of Spaghetti Sauce Data						
Expected Market Share						
Maximum Utility Model						
OBS	BRAND	MEAT	MUSHROOM	NATURAL	PRICE	SHARE
35	Z	Italian Sausage	No Mention	No Mention	2.49	0
36	Z	Vegetarian	Mushrooms	No Mention	2.49	0

Simulating Market Share, Bradley-Perry-Lute and Logit Models

The maximum utility model is just one of many ways to simulate market share. Two alternatives, which are also available in the **%SIM** macro, are the Bradley-Terry-Lute (BTL) model and the **logit** model. Unlike the maximum utility model, the BTL and the **logit** models do not assign all of the probability of choice to the most preferred alternative. Probability is a continuous function of utility. In the maximum utility model, probability of choice is a binary step function of utility. In the BTL model, probability of choice is a linear function of utility. In the **logit** model, probability of choice is an increasing curvilinear function of utility. The **BTL** model computes the probabilities by dividing each utility by the sum of the utilities within each subject. The **logit** model divides the exponentiated utilities by the sum of exponentiated utilities, again within subject. See the next section for a comparison of these three models.

```
%sim(data=utils,out=bt1,method=bt1,
      idvars=price brand meat mushroom natural);

%sim(data=utils,out=logit,method=logit,
      idvars=price brand meat mushroom natural);
```

The three methods produce different results. All three methods find that these subjects prefer the lower-priced vegetarian sauces. Since the BTL and **logit** models do not assign zero probability of purchase to most of the combinations, these methods do not produce a large number of zero market share proportions, as the maximum utility model did. See Output 4.6.

Output 4.6. Market Share Simulation

Conjoint Analysis of Spaghetti Sauce Data									
Expected Market Share									
Bradley-Terry-Lute Model									
OBS	BRAND	MEAT	MUSHROOM	NATURAL		PRICE	SHARE		
1	Z	Vegetarian	No Mention	All Natural	Ingredients	1.50	0.047866		
2	Y	Vegetarian	No Mention	All Natural	Ingredients	1.50	0.046915		
3	x	Vegetarian	No Mention	All Natural	Ingredients	1.50	0.045302		
4	z	Vegetarian	Mushrooms	No Mention		1.50	0.043496		
5	Y	Vegetarian	Mushrooms	No Mention		1.50	0.042546		
6	X	Vegetarian	Mushrooms	No Mention		1.50	0.040932		
7	Y	Vegetarian	No Mention	No Mention		1.75	0.038579		
8	x	Vegetarian	No Mention	No Mention		1.99	0.036780		
9	x	Vegetarian	No Mention	No Mention		1.75	0.036728		
10	x	Vegetarian	Mushrooms	All Natural	Ingredients	2.00	0.035059		
11	Y	Vegetarian	Mushrooms	All Natural	Ingredients	1.99	0.034578		
12	z	Vegetarian	Mushrooms	All Natural	Ingredients	1.99	0.032326		
13	Y	Vegetarian	No Mention	All Natural	Ingredients	2.25	0.031995		
14	x	Vegetarian	No Mention	All Natural	Ingredients	2.25	0.031799		
15	Y	Vegetarian	No Mention	No Mention		2.49	0.029454		
16	Z	Hamburger	No Mention	All Natural	Ingredients	2.00	0.027995		
17	Y	Hamburger	Mushrooms	All Natural	Ingredients	1.75	0.026594		
18	x	Vegetarian	Mushrooms	No Mention		2.49	0.026432		
19	x	Hamburger	No Mention	No Mention		2.00	0.026217		
20	Y	Hamburger	No Mention	No Mention		1.99	0.025736		
21	z	Hamburger	Mushrooms	All Natural	Ingredients	1.75	0.025716		
22	z	Vegetarian	Mushrooms	No Mention		2.49	0.025302		
23	X	Hamburger	Mushrooms	All Natural	Ingredients	1.99	0.024795		
24	X	Hamburger	Mushrooms	All Natural	Ingredients	1.75	0.024743		
25	Y	Hamburger	No Mention	No Mention		2.00	0.024005		
26	Z	Italian Sausage	No Mention	No Mention		1.75	0.021169		
27	Z	Hamburger	No Mention	All Natural	Ingredients	2.25	0.019423		
28	z	Italian Sausage	Mushrooms	No Mention		2.00	0.017507		
29	z	Italian Sausage	No Mention	No Mention		1.99	0.017365		
30	Y	Hamburger	Mushrooms	No Mention		2.25	0.017212		
31	x	Hamburger	Mushrooms	No Mention		2.25	0.017016		
32	Y	Italian Sausage	Mushrooms	No Mention		2.00	0.014916		
33	x	Italian Sausage	No Mention	No Mention		2.49	0.012870		
34	z	Italian Sausage	No Mention	No Mention		2.49	0.011741		

Conjoint Analysis of Spaghetti Sauce Data									
Expected Market Share									
Bradley-Terry-Lute Model									
OBS	BRAND	MEAT	MUSHROOM	NATURAL		PRICE	SHARE		
35	Y	Italian Sausage	Mushrooms	No Mention		2.49	.0099516		
36	Z	Italian Sausage	Mushrooms	No Mention		2.25	.0089351		

Output 4.6. (Continued)

Conjoint Analysis of Spaghetti Sauce Data Expected Market Share Logit Model						
OBS	BRAND	MEAT	MUSHROOM	NATURAL	PRICE	SHARE
1	Z	Vegetarian	No Mention	All Natural	Ingredients	1.50 0.10262
2	Y	Vegetarian	Mushrooms	No Mention		1.50 0.09367
3	z	Vegetarian	Mushrooms	No Mention		1.50 0.08572
4	Y	Vegetarian	No Mention	All Natural	Ingredients	1.50 0.08330
5	x	Vegetarian	No Mention	All Natural	Ingredients	1.50 0.07678
6	X	Vegetarian	Mushrooms	No Mention		1.50 0.06024
7	Y	Vegetarian	No Mention	No Mention		1.75 0.04150
8	X	Hamburger	Mushrooms	All Natural	Ingredients	1.99 0.04018
9	Y	Vegetarian	Mushrooms	All Natural	Ingredients	1.99 0.03418
10	x	Vegetarian	Mushrooms	All Natural	Ingredients	2.00 0.03361
11	Y	Hamburger	Mushrooms	All Natural	Ingredients	1.75 0.03156
12	X	Hamburger	Mushrooms	All Natural	Ingredients	1.75 0.02993
13	Z	Vegetarian	Mushrooms	All Natural	Ingredients	1.99 0.02645
14	Z	Hamburger	No Mention	All Natural	Ingredients	2.00 0.02605
15	x	Vegetarian	No Mention	No Mention		1.75 0.02413
16	X	Vegetarian	No Mention	No Mention		1.99 0.02327
17	z	Hamburger	Mushrooms	All Natural	Ingredients	1.75 0.02201
18	Z	Italian Sausage	No Mention	No Mention		1.75 0.02185
19	X	Vegetarian	Mushrooms	No Mention		2.49 0.01799
20	Y	Vegetarian	No Mention	All Natural	Ingredients	2.25 0.01583
21	Y	Hamburger	No Mention	No Mention		1.99 0.01385
22	X	Hamburger	No Mention	No Mention		2.00 0.01347
23	Z	Vegetarian	Mushrooms	No Mention		2.49 0.01168
24	X	Vegetarian	No Mention	All Natural	Ingredients	2.25 0.01151
25	Y	Hamburger	Mushrooms	No Mention		2.25 0.00887
26	X	Hamburger	Mushrooms	No Mention		2.25 0.00868
27	Y	Vegetarian	No Mention	No Mention		2.49 0.00859
28	Y	Hamburger	No Mention	No Mention		2.00 0.00750
29	z	Hamburger	No Mention	All Natural	Ingredients	2.25 0.00595
30	Y	Italian Sausage	Mushrooms	No Mention		2.00 0.00512
31	z	Italian Sausage	Mushrooms	No Mention		2.00 0.00478
32	Z	Italian Sausage	No Mention	No Mention		1.99 0.00450
33	Y	Italian Sausage	Mushrooms	No Mention		2.49 0.00133
34	x	Italian Sausage	No Mention	No Mention		2.49 0.00130

Conjoint Analysis of Spaghetti Sauce Data Expected Market Share Logit Model						
OBS	BRAND	MEAT	MUSHROOM	NATURAL	PRICE	SHARE
35	Z	Italian Sausage	No Mention	No Mention	2.49	.0011974
36	Z	Italian Sausage	Mushrooms	No Mention	2.25	.0007509

Simulator Comparisons

The maximum utility, BTL, and logit models are based on different assumptions and produce different results. The maximum utility model has the advantage of being scale-free. Any strictly monotonic transformation of each subject's utilities will produce the same market share. However, this model is unstable because it assigns a zero probability of choice to all alternatives that do not have the maximum utility, including those that have utilities near the maximum. The disadvantage of the BTL and logit models is that results are not invariant under linear transformations of the utilities. These methods are considered inappropriate by some researchers for this reason. With negative utilities, the BTL method produces negative probabilities, which are invalid. The BTL results change when a constant is added to the utilities

but do not change when a constant is multiplied by the utilities. Conversely, the logit results change when a constant is multiplied by the utilities but do not change when a constant is added to the utilities. The BLT method is not often used in practice, the logit model is sometimes used, and the maximum utility model is most often used. Refer to Finkbeiner (1988) for a discussion of conjoint analysis choice simulators. Do not confuse a logit model choice simulator and the multinomial logit model; they are quite different.

The following steps illustrate the different assumptions made by the three choice simulators. A plot is generated to show expected market share for a subject with utilities ranging from one to nine. Try other minima and maxima to see the effects of linear transformations of the utilities.

```

goptions reset=goptions device=pslepf gsfmde=replace
          gaccess=gsasfile hsize=4.5in vsize=4.5in
          ftext=swiss colors=(black);
filename gsasfile "sim1.ps";

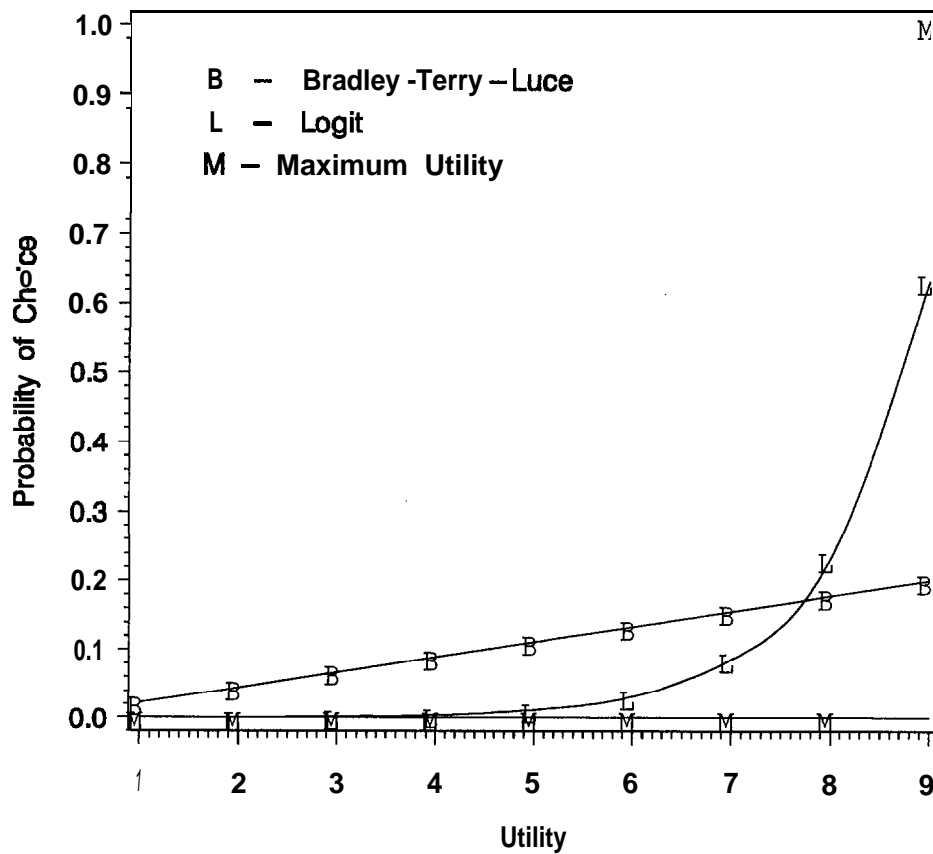
%let min = 1;
%let max = 9;
%let by = 1;
%let list = &min to &max by &by;
data a;
  sumb = 0;
  suml = 0;
  do u = &list;
    logit = exp(u);
    btl = u;
    sumb = sumb + btl;
    suml = suml + logit;
  end;

  do u = &list;
    logit = exp(u);
    btl = u;
    max = abs(u - (&max)) < (0.5 * (&by));
    btl = btl / sumb;
    logit = logit / suml;
  output;
  end;
run;

proc gplot;
  title h=1.5 'Simulator Comparisons';
  plot max * u = 1 btl * u = 2 logit * u = 3 /
        vaxis=axis2 haxis=axis1 overlay frame;
  symbol1 v=M i=step;
  symbol2 v=B i=join;
  symbol3 v=L i=spline;
  axis1 order=(&list) label=('Utility');
  axis2 order=(0 to 1 by 0.1)
        label=(angle=90 "Probability of Choice");
  note move=(2.5cm,9.2cm)
        font=swissu 'B - ' font=swiss 'Bradley-Terry-Lute';
  note move=(2.5cm,8.7cm)
        font=swissu 'L - ' font=swiss 'Logit';
  note move=(2.5cm,8.2cm)
        font=swissu 'M - ' font=swiss 'Maximum Utility';
run; quit;

```

Simulator Comparisons



The maximum utility line is flat at zero until it reaches the maximum utility, where it jumps to 1.0. The y-axis of the BTL curve linearly increase from 0.02 to 0.20 as utility ranges from 1 to 9. The logit curve increases exponentially, with small utilities mapping to near-zero probabilities and the largest utility mapping to a proportion of **0.63**.

Change in Market Share

The following steps simulate what would happen to the market if new products were introduced. Simulation observations are added to the data set and given zero weight. The conjoint analyses are rerun to compute the utilities for the active observations and the simulations. The maximum utility model is used. See Output 4.7.

```
data simulat;
  input brand $1. +1 meat $10. +1 mushroom $10. +1
         natural $23. price;
  datalines;
X Vegetarian Mushrooms All Natural Ingredients 1.50
Y Vegetarian Mushrooms All Natural Ingredients 1.50
Z Vegetarian Mushrooms All Natural Ingredients 1.50
X Vegetarian No Mention No Mention              1.50
Y Vegetarian No Mention No Mention              1.50
Z Vegetarian No Mention No Mention              1.50

data simulat2;
  set sasuser.all(in=act) simulat;

  * Give active obs weight 1, simulations weight 0;
  weight = act;

  * Create variables for brand by price interactions;
  pricex = (brand = 'X') * price;
  pricey = (brand = 'Y') * price;
  pricez = (brand = 'Z') * price;
  run;

  * Fit each subject individually;
proc transreg data=simulat2 short method=morals;
  title2 'Individual Conjoint Analyses';
  model linear(sub1-sub8) =
    spline(pricex pricey prices /
           degree-2 knots=1.995 1.995 1.995)
    class(brand meat mushroom natural / zero=sum) / dummy;
  output out=utils2 dapproximations ireplace;
  weight weight;
  id price;
run;

%sim(data=utils2,out=shares2,method=max,
     idvars=price brand meatmushroom natural weight);
```

Output 4.7. Effects of New Products

Conjoint Analysis of Spaghetti Sauce Data Expected Market Share Maximum Utility Model								
	W E I G H T	B R A N D	M E A T	M U S H R O O M	N A T U R A L		P R I C E	S H A R E
1	0	Z	Vegetarian	Mushrooms	All Natural	Ingredients	1.50	0.29167
2	1	Y	Vegetarian	Mushrooms	No Mention		1.50	0.12500
3	1	Y	Vegetarian	No Mention	No Mention		1.75	0.12500
4	0	z	Vegetarian	No Mention	No Mention		1.50	0.10417
5	0	x	Vegetarian	Mushrooms	All Natural	Ingredients	1.50	0.10417
6	0	Y	Vegetarian	Mushrooms	All Natural	Ingredients	1.50	0.10417
7	1	z	Vegetarian	Mushrooms	No Mention		1.50	0.06250
8	0	x	Vegetarian	No Mention	No Mention		1.50	0.04167
9	0	Y	Vegetarian	No Mention	No Mention		1.50	0.04167
10	1	x	Vegetarian	Mushrooms	No Mention		1.50	0.00000
11	1	x	Vegetarian	No Mention	All Natural	Ingredients	1.50	0.00000
12	1	Y	Vegetarian	No Mention	All Natural	Ingredients	1.50	0.00000
13	1	z	Vegetarian	No Mention	All Natural	Ingredients	1.50	0.00000
14	1	x	Hamburger	Mushrooms	All Natural	Ingredients	1.75	0.00006
15	1	x	Vegetarian	No Mention	No Mention		1.75	0.00000
16	1	Y	Hamburger	Mushrooms	All Natural	Ingredients	1.75	0.00000
17	1	z	Hamburger	Mushrooms	All Natural	Ingredients	1.75	0.00000
18	1	z	Italian Sausage	No Mention	No Mention		1.75	0.00000
19	1	x	Hamburger	Mushrooms	All Natural	Ingredients	1.99	0.00000
20	1	X	Vegetarian	No Mention	No Mention		1.99	0.00000
21	1	Y	Hamburger	No Mention	No Mention		1.99	0.00000
22	1	Y	Vegetarian	Mushrooms	All Natural	Ingredients	1.99	0.00000
23	1	z	Italian Sausage	No Mention	No Mention		1.99	0.00000
24	1	z	Vegetarian	Mushrooms	All Natural	Ingredients	1.99	0.00000
25	1	X	Hamburger	No Mention	No Mention		2.00	0.00000
26	1	X	Vegetarian	Mushrooms	All Natural	Ingredients	2.00	0.00000
27	1	Y	Hamburger	No Mention	No Mention		2.00	0.00000

Conjoint Analysis of Spaghetti Sauce Data Expected Market Share Maximum Utility Model								
OBS	WEIGHT	BRAND	MEAT	MUSHROOM	NATURAL		PRICE	SHARE
28	1	Y	Italian Sausage	Mushrooms	No Mention		2.00	0
29	1	Z	Hamburger	No Mention	All Natural	Ingredients	2.00	0
30	1	Z	Italian Sausage	Mushrooms	No Mention		2.00	0
31	1	X	Hamburger	Mushrooms	No Mention		2.25	0
32	1	X	Vegetarian	No Mention	All Natural	Ingredients	2.25	0
33	1	Y	Hamburger	Mushrooms	No Mention		2.25	0
34	1	Y	Vegetarian	No Mention	All Natural	Ingredients	2.25	0
35	1	Z	Hamburger	No Mention	All Natural	Ingredients	2.25	0
36	1	Z	Italian Sausage	Mushrooms	No Mention		2.25	0
37	1	X	Italian Sausage	No Mention	No Mention		2.49	0
38	1	X	Vegetarian	Mushrooms	No Mention		2.49	0
39	1	Y	Italian Sausage	Mushrooms	No Mention		2.49	0
40	1	Y	Vegetarian	No Mention	No Mention		2.49	0
41	1	Z	Italian Sausage	No Mention	No Mention		2.49	0
42	1	Z	Vegetarian	Mushrooms	No Mention		2.49	0

These steps merge the data set containing the old market shares with the data set containing the new market shares to show the effect of adding the new products. See Output 4.8.

Brand Z vegetarian sauce with mushrooms and all natural ingredients at \$1.50 would pick up a 29% market share according to this analysis. This gain would largely come

at the expense of Brand Z vegetarian sauce, with mushrooms at \$1.50, which went from a 25% share down to a 6.25% share when the new products were introduced. These differ only in that the former has all natural ingredients. So advertising all natural ingredients in Brand Z vegetarian sauce, with mushrooms at \$1.50, should increase market share.

```

options ls=120 ps=60;
proc sort data=maxutils;
  by price brand meat mushroom natural;
run;

proc sort data=shares2;
  by price brand meat mushroom natural;
run;

data both;
  merge maxutils(rename=(share=oldshare)) shares2;
  by price brand meat mushroom natural;
  if oldshare = . then change = 0;
  else change = oldshare;
  change = share - change;
run;

proc sort;
  by descending share price brand meat mushroom natural;
run;

proc print;
  title2 'Expected Market Share and Change';
  var natural mushroom brand meat price
      weight oldshare share change;
run;

```

Output 4.8. Change in Market Share

Conjoint Analysis of Spaghetti Sauce Data Expected Market Share and Change									
OBS	NATURAL	MUSHROOM	BRAND	MEAT	PRICE	WEIGHT	OLDSHARE	SHARE	CHANGE
1	All Natural Ingredients	Mushrooms	Z	Vegetarian	1.50	0	.	0.29167	0.29167
2	No Mention	Mushrooms	Y	Vegetarian	1.50	1	0.12500	0.12500	0.00000
3	No Mention	No Mention	Y	Vegetarian	1.75	1	0.12500	0.12500	0.00000
4	No Mention	No Mention	Z	Vegetarian	1.50	0		0.10417	0.10417
5	All Natural Ingredients	Mushrooms	X	Vegetarian	1.50	0		0.10417	0.10417
6	All Natural Ingredients	Mushrooms	Y	Vegetarian	1.50	0	.	0.10417	0.10417
7	No Mention	Mushrooms	Z	Vegetarian	1.50	1	0.25000	0.06250	-0.18750
8	No Mention	No Mention	X	Vegetarian	1.50	0		0.04167	0.04167
9	No Mention	No Mention	Y	Vegetarian	1.50	0		0.04167	0.04167
10	No Mention	Mushrooms	X	Vegetarian	1.50	1	0.00000	0.00000	0.00000
11	All Natural Ingredients	No Mention	X	Vegetarian	1.50	1	0.08333	0.00000	-0.08333
12	All Natural Ingredients	No Mention	Y	Vegetarian	1.50	1	0.08333	0.00000	-0.08333
13	All Natural Ingredients	No Mention	Z	Vegetarian	1.50	1	0.08333	0.00000	-0.08333
14	All Natural Ingredients	Mushrooms	X	Hamburger	1.75	1	0.00000	0.00000	0.00000
15	No Mention	No Mention	X	Vegetarian	1.75	1	0.00000	0.00000	0.00000
16	All Natural Ingredients	Mushrooms	Y	Hamburger	1.75	1	0.06250	0.00000	-0.06250
17	All Natural Ingredients	Mushrooms	Z	Hamburger	1.75	1	0.00000	0.00000	0.00000
18	No Mention	No Mention	Z	Italian Sausage	1.75	1	0.00000	0.00000	0.00000
19	All Natural Ingredients	Mushrooms	X	Hamburger	1.99	1	0.18750	0.00000	-0.18750
20	No Mention	No Mention	X	Vegetarian	1.99	1	0.00000	0.00000	0.00000
21	No Mention	No Mention	Y	Hamburger	1.99	1	0.00000	0.00000	0.00000
22	All Natural Ingredients	Mushrooms	Y	Vegetarian	1.99	1	0.00000	0.00000	0.00000
23	No Mention	No Mention	Z	Italian Sausage	1.99	1	0.00000	0.00000	0.00000
24	All Natural Ingredients	Mushrooms	Z	Vegetarian	1.99	1	0.00000	0.00000	0.00000
25	No Mention	No Mention	X	Hamburger	2.00	1	0.00000	0.00000	0.00000
26	All Natural Ingredients	Mushrooms	X	Vegetarian	2.00	1	0.00000	0.00000	0.00000
27	No Mention	No Mention	Y	Hamburger	2.00	1	0.00000	0.00000	0.00000
28	No Mention	Mushrooms	Y	Italian Sausage	2.00	1	0.00000	0.00000	0.00000
29	All Natural Ingredients	No Mention	Z	Hamburger	2.00	1	0.00000	0.00000	0.00000
30	No Mention	Mushrooms	Z	Italian Sausage	2.00	1	0.00000	0.00000	0.00000
31	No Mention	Mushrooms	X	Hamburger	2.25	1	0.00000	0.00000	0.00000
32	All Natural Ingredients	No Mention	X	Vegetarian	2.25	1	0.00000	0.00000	0.00000
33	No Mention	Mushrooms	Y	Hamburger	2.25	1	0.00000	0.00000	0.00000
34	All Natural Ingredients	No Mention	Y	Vegetarian	2.25	1	0.00000	0.00000	0.00000
35	All Natural Ingredients	No Mention	Z	Hamburger	2.25	1	0.00000	0.00000	0.00000
36	No Mention	Mushrooms	Z	Italian Sausage	2.25	1	0.00000	0.00000	0.00000
37	No Mention	No Mention	X	Italian Sausage	2.49	1	0.00000	0.00000	0.00000
38	No Mention	Mushrooms	X	Vegetarian	2.49	1	0.00000	0.00000	0.00000
39	No Mention	Mushrooms	Y	Italian Sausage	2.49	1	0.00000	0.00000	0.00000
40	No Mention	No Mention	Y	Vegetarian	2.49	1	0.00000	0.00000	0.00000
41	No Mention	No Mention	Z	Italian Sausage	2.49	1	0.00000	0.00000	0.00000
42	No Mention	Mushrooms	Z	Vegetarian	2.49	1	0.00000	0.00000	0.00000

Brand by Price Interactions

Brand and price may not have independent effects on preference. Brand by price interaction plots contain one curve for each brand and show how utility for each brand changes as a function of price. These steps plot the brand by price interaction for each subject. In the interest of space, only one plot is shown.

```
goptions reset=goptions device=pslepsz gsfmde=replace
      gaccess=gsasfile hsize=4.5in vsize=4.5in
      ftext=swiss colors=(black);
filename gsasfile "bpint.ps";
```

```
* Add extra zero weight observations to fill in curve;
data addprice;
  set sasuser.all;
  weight = 1;
  output ;
  weight = 0;
  if _n_ = 1 then do;
    array s[8] sub1-sub8;
    do j = 1 to 8; s[j] = .; end; drop j;
```

```

do brand = 'X', 'Y', 'Z'; do price = 1.50 to 2.49 by 0.01;
  pricex = (brand = 'X') * price;
  pricey = (brand = 'Y') * price;
  pricez = (brand = 'Z') * price;
  output;
end; end;
end;
run;

* Control for nominal factors;
proc transreg data=addprice short method=univariate;
  title2 'Individual Conjoint Analyses';
  model identity(sub1-sub8) =
    class(brand meat mushroom natural / zero=sum);
  output out=resids dapproximations ireplace residuals;
  id price;;
  weight weight;
run;

* Find brand by price interaction curves;
proc transreg data=resids short method=morals;
  title2 'Individual Conjoint Analyses';
  model linear(rsub1-rsub8) =
    spline(pricex pricey pricez / degree=2 knots=1.995 1.995 1.995)
    /dummy;
  output out=sasuser.inters dapproximations ireplace;
  id price brand;
  weight weight;
run;

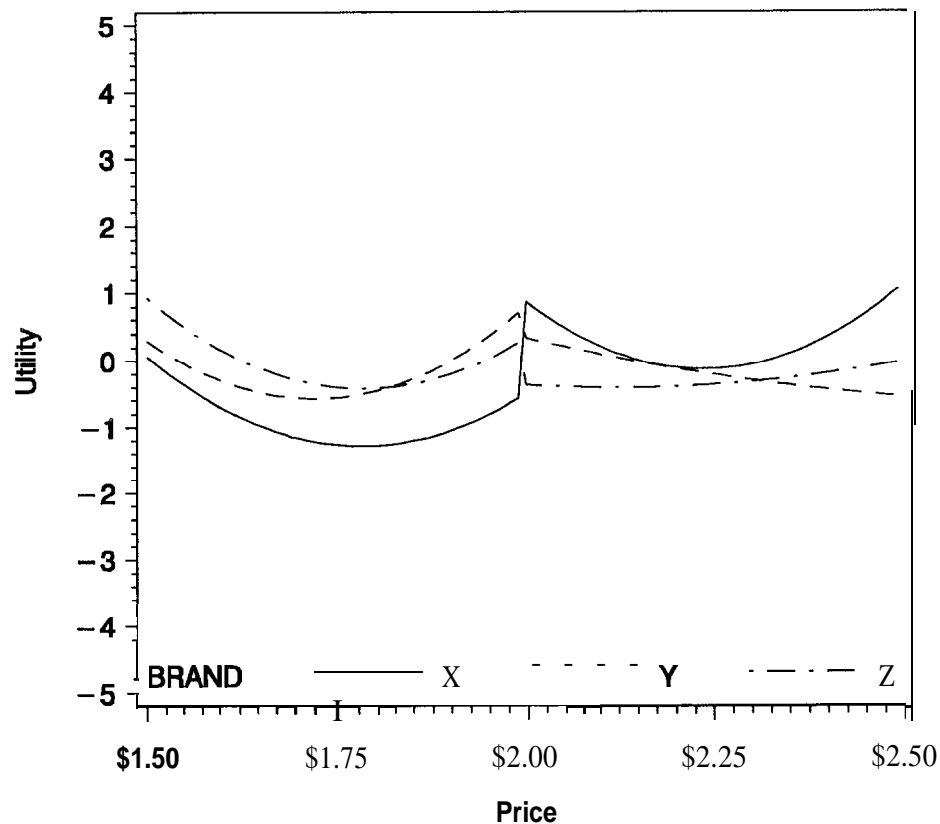
proc sort;
  by _depvar_ price;
run;

* Plot the results;
proc gplot data=sasuser.inters;          /* Normally, use BY to plot all. */
  * by _depvar_;                          /* Here, in the interest of      */
  where _depvar_ = 'LINEAR(RSUB1)'; /* space, use WHERE to subset. */
  title h=1.5 'Conjoint Analysis of Spaghetti Sauce Data';
  title2 h=1 'Brand by Price Interactions';
  plot a-depend * price = brand /
    vaxis=axis1 haxis=axis2 frame legend=legend1;
  axis1 order=(-5 to 5) label=(angle=90 'Utility');
  axis2 order=(1.50 to 2.50 by 0.25) label=('Price');
  symbol1 i=join line=1;
  symbol2 i=join line=20;
  symbol3 i=join line=41;
  legend1 position=inside mode=protect;
  format price dollar5.2;
run; quit;

```

Conjoint Analysis of Spaghetti Sauce Data

Brand by Price Interactions



Brand X has a lower utility over the lower range of price than the other brands and a higher utility over the higher ranges. Only slight discontinuities in the price functions were found at \$2.00. The discontinuities appear to be too small to be meaningful in this example. In an example with real brands, you would expect to see more pronounced effects.

Example 5. Choice of Chocolate Candies

This example illustrates using a multinomial **logit** model to directly investigate consumer choice behavior. The multinomial **logit** model (Manski and McFadden, 1981) is a choice model. It is an alternative to conjoint analysis that is becoming increasingly popular in marketing research (Louviere, 1991). Choice models allow you to study choice directly, whereas conjoint models generate utilities that have to be input to a simulator to model choice.

Multinomial Logit Model

In this example, ten subjects were presented with eight different types of chocolate candies. They contained dark or milk chocolate, soft or hard center, and nuts or no nuts. All combinations were generated. Ten subjects were presented with all eight alternatives and asked to choose one. Choice is indicated by CHOOSE= 1. The SAS data set CHOCS contains the input data set.

```
data chocs;
  input subj choose dark soft nuts @@;
  * Create dummy time variable, t, so that the value of
    t for chosen items is smaller-than that for nonchosen;
  t = 2 - choose;
  datalines;
10 0 0 0 1 10 0 0 1 10 0 10 10 0 11
1 1 1 0 0 1 0 1 0 1 1 0 1 1 0 111
2 0 0 0 0 2 0 0 0 1 2 0 0 1 0 2 0 0 1 1
2 0 1 0 0 2 1 1 0 1 2 0 1 1 0 2 0 1 1 1
3 0 0 0 0 3 0 0 0 1 3 0 0 1 0 3 0 0 1 1
3 0 1 0 0 3 0 1 0 1 3 1 1 1 0 3 0 1 1 1
4 0 0 0 0 4 0 0 0 1 4 0 0 1 0 4 0 0 1 1
4 1 10 0 4 0 1 0 1 4 0 1 1 0 4 0 1 1 1
5 0 0 0 0 5 1 0 0 1 5 0 0 1 0 5 0 0 1 1
5 0 1 0 0 5 0 1 0 1 5 0 1 1 0 5 0 1 1 1
6 0 0 0 0 6 0 0 0 1 6 0 0 1 0 6 0 0 1 1
6 0 1 0 0 6 1 1 0 1 6 0 1 1 0 6 0 1 1 1
7 0 0 0 0 7 1 0 0 1 7 0 0 1 0 7 0 0 1 1
7 0 1 0 0 7 0 1 0 1 7 0 1 1 0 7 0 1 1 1
8 0 0 0 0 8 0 0 0 1 8 0 0 1 0 8 0 0 1 1
8 0 1 0 0 8 1 1 0 1 8 0 1 10 8 0 1 1 1
9 0 0 0 0 9 0 0 0 1 9 0 0 1 0 9 0 0 1 1
9 0 1 0 0 9 1 1 0 1 9 0 1 1 0 9 0 1 1 1
10 0 0 0 0 10 0 0 0 1 10 0 0 10 100 011
10 0 10 0 10 110 1 10 0 110 10 0 111
;
```

Choice is investigated by using the SAS/STAT procedure PHREG to fit a multinomial logit model. The multinomial logit model assumes that the probability that an individual will choose one of the m alternatives c_i from choice set C is

$$p(c_i | C) = \frac{\exp(U(c_i))}{\sum_{j=1}^m \exp(U(c_j))} = \frac{\exp(\mathbf{x}_i \boldsymbol{\beta})}{\sum_{j=1}^m \exp(\mathbf{x}_j \boldsymbol{\beta})}$$

where $U(c_i) = \mathbf{x}_i \boldsymbol{\beta}$ is the utility for alternative c_i , \mathbf{x}_i is a vector of alternative attributes, and $\boldsymbol{\beta}$ is a vector of unknown parameters. The PHREG procedure is used to compute the parameter estimates.

```

* Multinomial logit model (conditional logistic).
Set up each subject as a separate stratum.
Make the choice an event and represent the others as
censored values;

proc phreg data=chocs outest=betas;
  strata subj;
  model t * choose(0) = dark soft nuts;
run;

proc print data=betas;
run;

```

The parameter estimate with the smallest p-value is for soft. Since the parameter estimate is negative, hard is the most preferred level. Dark is preferred over milk, and nuts are preferred over no nuts. See Output 5.1.

Output 5.1. Multinomial Logit Model with Chocolate Data

Choice of Chocolate Candies					
The PHREG Procedure					
Data Set: WORK.CHOCs					
Dependent Variable: T					
Censoring Variable: CHOOSE					
Censoring Value(s): 0					
Ties Handling: BRESLOW					
Summary of the Number of Event and Censored Values					
Stratum	SUBJ	Total	Event	Censored	Percent Censored
1	1	8	1	7	87.50
2	2	8	1	7	87.50
3	3	8	1	7	87.50
4	4	8	1	7	87.50
5	5	8	1	7	87.50
6	6	8	1	7	87.50
7	7	8	1	7	87.50
8	8	8	1	7	87.50
9	9	8	1	7	87.50
10	10	8	1	7	87.50

Total		80	10	70	87.50

Output 5.1. (Continued)

Choice of Chocolate Candies						
The PHREG Procedure						
Testing global Null Hypothesis: BETA=0						
Criterion		Without Covariates	With Covariates	Model	Chi-Square	
-2 LOG L		41.589	28.727	12.862	with 3 DF	(p=0.0049)
Score		.	.	11.600	with 3 DF	(p=0.0089)
Wald				8.928	with 3 DF	(p=0.0303)
Analysis of Maximum Likelihood Estimates						
Variable	DF	Parameter Estimate	Standard Error	Wald Chi-Square	Pr > Chi-Square	Risk Ratio
DARK	1	1.386294	0.79057	3.07490	0.0795	4.000
SOFT	1	-2.197225	1.05409	4.34502	0.0371	0.111
NUTS	1	0.847298	0.69007	1.50762	0.2195	2.333

Choice of Chocolate Candies							
OBS	-TIES-	-TYPE-	_NAME_	DARK	SOFT	NUTS	_LNLIKE_
1	BRBSLOW	PARMS	ESTIMATE	1.38629	-2.19722	0.84730	-14.3635

The parameter estimates are output and used to construct the estimated probability of choice for each alternative.

```

* Matrix with all combinations of the attributes;
data combos;
  input dark soft nuts;
  datalines;
1 1 1
1 1 0
1 0 1
1 0 0
0 1 1
0 1 0
0 0 1
0 0 0
;

data p;
  retain sum 0;
  set combos end=eof;
  if _n_ = 1 then
    set betas(rename=(dark=b1 soft=b2 nuts=b3));
  keep dark soft nuts p;
  array x[3] dark soft nuts;
  array b[3] b1-b3;

  * For each combination, create x * b;
  p = 0;
  do j = 1 to 3;
    p = p + x[j]* b[j];
  end;

  * Exponentiate x * b and sum them up;
  p = exp(p);
  sum = sum + p;

```

```

      * Output sum exp(x * b) ;
      if eof then call symput('sum',put(sum,best12.));
      run ;

proc format ;
  value df 1 = 'dark' 0 = 'milk';
  value sf 1 = 'soft' 0 = 'hard';
  value nf 1 = 'nuts' 0 = 'no nuts';
run;

* Divide each exp(x * b) by sum exp(x * b);
data p;
  set p;
  p = p / (&sum);
  format dark df. soft sf. nuts nf.;
run;

proc sort;
  by descending p;
run;

proc print;
run ;

```

The three most preferred alternatives are dark/hard/nuts, dark/hard/no nuts, and milk/hard/nuts. See Output 5.2.

Output 5.2. Multinomial Logit Model with Chocolate Data

Choice of Chocolate Candies				
OBS	DARK	SOFT	NUTS	P
1	dark	hard	nuts	0.504
2	dark	hard	no nuts	0.216
3	milk	hard	nuts	0.126
4	dark	soft	nuts	0.056
5	milk	hard	no nuts	0.054
6	dark	soft	no nuts	0.024
7	milk	soft	nuts	0.014
8	milk	soft	no nuts	0.006

This example fits a main-effects model and uses a full-factorial design. Fractional-factorial designs, nonorthogonal designs, and models with interactions and splines can also be used with the multinomial logit model.

PROC TRANSREG Specifications

PROC TRANSREG (transformation regression) is used to perform conjoint analysis and many other types of analyses, including simple regression, multiple regression, redundancy analysis, canonical correlation, main-effects analysis of variance, and external unfolding, all with nonlinear transformations of the variables. This section documents the statements and options available in PROC TRANSREG that are used in conjoint analyses. Refer to “The TRANSREG Procedure” in the *SAS/STAT User’s Guide, Version 6, Fourth Edition* and in *SAS Technical Report P-229, SAS/STAT Software: Changes and Enhancements, Release 6.07* for more information on PROC TRANSREG.

The following statements are used in the TRANSREG procedure for conjoint analysis:

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

Specify the PROC and MODEL statements to use PROC TRANSREG. The OUTPUT statement is required to produce an OUT= output data set, which contains the transformations, dummy variables, and predicted utility for each product. The OUTTEST= data set, which contains the ANOVA, regression, and utility tables, is requested on the PROC statement. All options can be abbreviated to their first three letters.

PROC TRANSREG Statement

```
PROC TRANSREG <DATA= SAS-data-set >
               <OUTTEST= SAS-data-set >
               <a-options > co-options >;
```

The DATA= and OUTTEST= options can appear only on the PROC TRANSREG statement. The algorithm options (*a-options*) appear on the PROC or MODEL statement. The output options (*o-options*) can appear on the PROC or OUTPUT statement.

DATA= SASdata-set

the input SAS data. If the DATA= option is not specified, PROC TRANSREG uses the most recently created SAS data set.

OUTTEST= SASdata-set

specifies an output data set to contain the ANOVA table, R^2 , and the conjoint analysis part-worth utilities, and the attribute importances.

Algorithm Options

```
PROC TRANSREG <DATA= SAS-data-set >
               <OUTTEST= SAS-data-set >
               <a-options > <o-options >;
MODEL transform(dependents </ t-options >) =
      transform(independents </ t-options >)
      <transform(independents </ t-options >) . . . > </ a-options >;
```

Algorithm options can appear on the PROC or MODEL statement as *u-options*.

CONVERGE= *n*

specifies the minimum average absolute change in standardized variable scores that is required to continue iterating. By default, CONVERGE=0.00001.

CPREFIX= *n*

specifies the number of first characters of a CLASS variable's name to use in constructing names for binary variables in the output data set. The default is CPREFIX=6. When a format is specified for a CLASS variable, then the default for that variable is $7 - \min(6, \max(1, f))$, where f is the format length.

For example if CLASS(X) is specified and X has values 1, 2, 3, then by default the expanded names are X1, X2, and X3. Both the levels and the variable name are short, so CPREFIX= does not have to be specified. If CLASS(XVAR) is specified and XVAR has values CLASS1, CLASS2, and CLASS3, then by default the expanded names are XVARCLAS, XVARCLAS, and XVARCLAS. Since two variables in the data set cannot have the same name, this causes an error. When CPREFIX=1 is specified, the expanded variable names are XCLASS1, XCLASS2, and XCLASS3, which are all valid and different. When CPREFIX=0 is specified, the expanded variable names are CLASS 1, CLASS2, and CLASS3, which are also all valid and different.

DUMMY

requests a canonical initialization. When SPLINE transformations are requested, specify DUMMY to solve for the optimal transformations without iteration. Iteration is only necessary when there are **monotonicity** constraints.

MAXITER= *n*

specifies the maximum number of iterations. By default, MAXITER=30.

METHOD= MORALS
METHOD= UNIVARIATE

specifies the iterative algorithm. Both METHOD=MORALS and METHOD=UNIVARIATE fit univariate multiple regression models with the possibility of nonlinear transformations of the variables. They differ in the way they structure the output data set when there is more than one dependent variable. When it can be used, METHOD=UNIVARIATE is more efficient than METHOD=MORALS.

METHOD=UNIVARIATE is used when no transformations of the independent variables are requested, for example when the independent variables are all designated CLASS, IDENTITY, or PSPLINE. In this case the final set of independent variables will be the same for all subjects. If transformations such as MONOTONE, LINEAR, SPLINE or MSPLINE are specified for the independent variables, the transformed independent variables may be different for each dependent variable and so must be output separately for each dependent variable. In conjoint analysis, there will typically be one dependent variable for each subject. This is illustrated in the examples.

METHOD=UNIVARIATE with more than one dependent variable creates a data set with the same number of score observations as the original but with more variables. The untransformed dependent variable names are unchanged. The default transformed dependent variable names consist of the prefix T and the original variable names. The default dependent variable approximation names consist of the prefix A and the original variable names. The full set of independent variables appears once.

When more than one dependent variable is specified, METHOD=MORALS creates a *rolled-out* data set with the dependent variable in -DEPEND-, its transformation in T-DEPEND, and its approximation in A-DEPEND. The full set of independents is repeated for each (original) dependent variable.

The procedure chooses a default method based on what is specified on the model statement. When transformations of the independent variables are requested, the default method is MORALS, Otherwise the default method is UNIVARIATE.

SHORT

suppresses the iteration histories. There are no iterations when there are no monotonicity constraints, so specifying SHORT eliminates unnecessary output.

TEST

prints an overall ANOVA table for each subject. The ANOVA results are, at best, approximate since the normality and independence assumptions are violated.

UTILITIES

prints the part-worth utilities table and an ANOVA table.

Output Options

```
PROC TRANSREG <DATA= SAS-data-set >
               <OUTTEST= SAS-data-set>
               <a-options> co-options>;
OUTPUT <OUT= SAS-data-set> <o-options>;
```

The OUT= option can only appear on the OUTPUT statement. The other options can appear on the PROC or OUTPUT statement as *o-options*.

DAPPROXIMATIONS

includes the approximations to the transformed dependent variables in the output data set, which are the predicted utilities for each product. By default, the approximation variable name is the original dependent variable name prefixed with an A.

IREPLACE

replaces the original independent variables with the transformed independent variables in the output data set. The names of the transformed variables in the output data set correspond to the names of the original independent variables in the input data set.

OUT= SAS-data-set

names the output data set. When an OUTPUT statement is specified without the OUT= option, PROC TRANSREG creates a data set and uses the **DATA_n** convention. To create a permanent SAS data set, specify a two-level name. The data set will contain the original input variables, the dummy variables, the transformation of the dependent variable, and the predicted utilities for each product.

RESIDUALS

outputs to the OUT= data set the differences between the observed and predicted utilities. By default, the residual variable name is the original dependent variable name prefixed with an R.

Transformations and Expansions

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

The following are specified on the MODEL statement as *transforms*. PSPLINE and CLASS are expansions that create more than one output variable for each input variable. The rest are transformations that create one output variable for each input variable.

CLASS

designates variables for analysis as nominal-scale-of-measurement variables. For conjoint analysis the ZERO=SUM *t-option* is typically specified: CLASS(*variables*/ZERO=SUM). CLASS expands each variable to a set of dummy variables. Usually the number output variables for each CLASS variable is the number of different values in the input variables. CLASS should not be

specified with the dependent variables.

IDENTITY

variables are not changed by the iterations. **IDENTITY(variables)** designates interval-scale-of-measurement variables when no transformation is permitted. When small data values mean high preference, you will need to use the REFLECT transformation option. Do not use IDENTITY for dependent variables when there are simulation observations, because IDENTITY does not score missing values.

LINEAR

linearly transforms variables. **LINEAR(variables)** designates interval-scale-of-measurement variables. LINEAR, unlike IDENTITY, allows observations with zero weight to be scored as passive observations. This is useful when some of the observations are simulation observations and have missing values in the dependent variable. **Missing values are optimally scored in LINEAR variables.** When LINEAR is used with dependent variables, a metric conjoint analysis is performed. When small data values mean high preference, you will need to use the REFLECT transformation option. LINEAR can also be specified for independent variables.

LOGIT

yields logit transformations of variables with values in the interval $(0.0 < x < 1.0)$. **LOGIT(variables)** specifies the transformation $\log(x / (1 - x))$ for each data value x . Unlike other transformations, **LOGIT** does not have a three-letter abbreviation since LOG means logarithm. **LOGIT** is typically used only for dependent variables.

MONOTONE

monotonically transforms variables; ties are preserved. When **MONOTONE(variables)** is used with dependent variables, a nonmetric conjoint analysis is performed. When small data values mean high preference, you will need to use the REFLECT transformation option. MONOTONE can also be used with independent variables to impose monotonicity on the utilities. When it is known that monotonicity should exist in an attribute variable, using MONOTONE instead of CLASS for that attribute may improve prediction. An option exists in PROC TRANSREG for optimally untying tied values, but this option should not be used because it almost always produces a degenerate result.

MSPLINE

monotonically and smoothly transforms variables. By default, **MSPLINE(variables)** fits a monotonic quadratic spline with no knots. Knots are specified as *t-options*, for example **MSPLINE(variables / NKNOTS=3)** or **MSPLINE(variables / KNOTS=5 TO 15 BY 5)**. MSPLINE, like MONOTONE, finds a monotonic transformation. Unlike MONOTONE, **MSPLINE** places a bound on the *df* (number of knots + degree) used by the transformation. With MSPLINE it is possible to allow for nonlinearity in the responses and still have error *df*. This is not always possible with MONOTONE. When small data values mean high preference, you will need to use the REFLECT transformation option. MSPLINE can also be used with attribute variables to impose monotonicity on the utilities.

PSPLINE

expands each variable to apiece-wise polynomial spline basis. By default, **PSPLINE(variables)** uses a cubic spline with no knots. Knots are specified as *t-options*. Specify **PSPLINE(variable / DEGREE=2)** for an attribute variable to add a quadratic term to the model. For each PSPLINE variable, $d + k$ output variables are created, where d is the degree of the polynomial and k is the number of knots. PSPLINE should not be specified with the dependent variables.

RANK

performs a rank transformation, with ranks averaged within ties. Rating-scale data can be transformed to ranks by specifying **RANK(variables)**. When small data values mean high preference, you will need to use the REFLECT transformation option. RANK is typically used only for dependent variables.

For example if a rating-scale variable has sorted values 1, 1, 1, 2, 3, 3, 4, 5, 5, 5, then the rank transformation is 2, 2, 2, 4, 5.5, 5.5, 7, 9, 9, 9. A conjoint analysis of the original rating-scale variable will not usually be the same as a conjoint analysis of a rank transformation of the ratings. With ordinal-scale-of-measurement data, it is often good to analyze rank transformations instead of the original data. An alternative is to specify MONOTONE, which performs a nonmetric conjoint analysis. For real data, MONOTONE will always produce better fit than RANK, but RANK may lead to better prediction.

SPLINE

smoothly transforms variables. By default, **SPLINE(variables)** fits a cubic spline with no knots. Knots are specified as *t-options*. Like PSPLINE, SPLINE models nonlinearities in the attributes.

Transformation Options

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

The following are specified on the MODEL statement as *t-options*'s.

DEGREE=*n*

specifies the degree of the spline. The defaults are DEGREE=3 for SPLINE and PSPLINE, and DEGREE=2 for MSPLINE. For example, to request a quadratic spline, specify **SPLINE(variables / DEGREE=2)**.

EVENLY

is used with the **NKNOTS=** option to evenly space the knots for splines. For example, if **SPLINE(X / NKNOTS=2 EVENLY)** is specified and X has a minimum of 4 and a maximum of 10, then the two interior knots are 6 and 8. Without **EVENLY**, **NKNOTS=** places knots at percentiles, so the knots are not evenly spaced.

KNOTS= *numberlist*

specifies the interior knots or break points for splines. By default, there are no knots. For example, to request knots at 1, 2, 3, 4, 5, specify `SPLINE(variable / KNOTS=1 TO 5)`.

NKNOTS= *k*

creates *k* knots for splines: the first at the $100 / (k + 1)$ percentile, the second at the $200 / (k + 1)$ percentile, and so on. Knots are always placed at data values; there is no interpolation. For example, with `SPLINE(variable / NKNOTS=3)`, knots are placed at the twenty-fifth percentile, the median, and the seventy-fifth percentile. By default, `NKNOTS=0`.

REFLECT

reflects the transformation around its mean, $Y = -(Y - \bar{Y}) + \bar{Y}$, after the iterations are completed and before the final standardization and results calculations. This option is particularly useful with the dependent variable. When the dependent variable consists of ranks with the most preferred combination assigned 1.0, `LINEAR(variable / REFLECT)` will reflect the transformation so that positive utilities mean high preference.

ZERO= SUM

constrains the utilities to sum to zero within each attribute. `CLASS(variables / ZERO=SUM)` creates a less than full rank model, but the coefficients are uniquely determined due to the sum-to-zero constraint.

BY Statement

BY *variables*;

A BY statement can be used 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 the input data set is not sorted in ascending order, use one of the following alternatives:

Use the SORT procedure with a similar BY statement to sort the data.

Use the BY statement options `NOTSORTED` or `DESCENDING` in the BY statement for the TRANSREG procedure. As a cautionary note, the `NOTSORTED` option does not mean that the data are unsorted. It means that the data are arranged in groups (according to values of the BY variables), and these groups are not necessarily in alphabetical or increasing numeric order.

Use the **DATASETS** procedure (in base SAS software) to create an index on the BY variables.

For more information on the BY statement, refer to the discussion in *SAS Language: Reference, Version 6, First Edition*. For more information on the **DATASETS** procedure, refer to the discussion in *SAS Procedures Guide, Release 6.06 Edition*.

ID Statement

ID variables;

The ID statement includes additional character or numeric variables from the input data set in the OUT= data set.

WEIGHT Statement

WEIGHT variable;

A WEIGHT statement can be used in conjoint analysis to distinguish ordinary *active* observations, holdouts, and simulation observations. When a WEIGHT statement is used, a weighted residual sum of squares is minimized. The observation is used in the analysis only if the value of the WEIGHT statement variable is greater than zero. For observations with positive weight, the WEIGHT statement has no effect on *df* or number of observations, but the weights affect most other calculations.

Assign each active observation a weight of 1. Assign each holdout observation a weight that excludes it from the analysis, such as 0. Assign each simulation observation a different weight that excludes it from the analysis, such as -1.0 . **Holdouts** are rated by the subjects and so have nonmissing values in the dependent variables. Simulation observations are not rated and so have missing values in the dependent variable. It is useful to create a format for the WEIGHT variable that distinguishes the three types of observations in the input and output data sets.

```
proc format;
  value wf 1  = 'Active'
           0  = 'Holdout'
          -1  = 'Simulation';
run;
```

PROC TRANSREG does not distinguish between weights of zero and -1.0 . **Both** weights are nonpositive and exclude the observations from the analysis. The holdout and simulation observations are given different nonpositive values and a format to make them easy to distinguish in subsequent analyses and listings. The utilities for each attribute are computed using only those observations with positive weight. The predicted utility is computed for all products, even those with nonpositive weights.

Samples of PROC TRANSREG Usage

Conjoint analysis can be performed in many ways with PROC TRANSREG. This section provides samples specifications for some typical and more esoteric conjoint analyses. The dependent variable, **RATING** or **RANKING**, typically contains ratings or a ranking of the products. The independent variables, **X1-X5**, are the attributes. For metric conjoint analysis, the dependent variable is designated **LINEAR**. For non-metric conjoint analysis, **MONOTONE** is used. Attributes are usually designated as **CLASS** variables with the restriction that the utilities within each attribute sum to zero.

The **UTILITIES** option requests an overall ANOVA table, a table of part-worth utilities, their standard errors, and the importance of each attribute. The **DAPPROXIMATIONS** (dependent variable approximations) option outputs to a data set the predicted utility for each product. The **IREPLACE** option suppresses the separate output of transformed independent variables since the independent variable transformations are the same as the raw independent variables. The **WEIGHT** variable is used to distinguish active observations from holdouts and simulation observations. The **REFLECT** transformation option reflects the transformation of the ranking so that large transformed values, positive utility, and positive evaluation will all correspond.

Today, metric conjoint analysis is used more often than nonmetric conjoint analysis, and rating-scale data are collected more often than rankings.

Metric Conjoint Analysis with Rank-Order Data

This is a metric conjoint analysis with rank-order data.

```
proc transreg data=a utilities;
  model linear(ranking / reflect) = class(x1-x5 / zero=sum);
  output dapproximations ireplace;
  weight w;
run;
```

Metric Conjoint Analysis with Rating Scale Data

Here is a typical metric conjoint analysis specification with rating-scale data.

```
proc transreg data=a utilities;
  model linear(rating) = class(x1-x5 / zero=sum);
  output dapproximations ireplace;
  weight w;
run ;
```

Nonmetric Conjoint Analysis

Next is a nonmetric conjoint analysis specification, which has many parameters for the transformation.

```
proc transreg data=a utilities;
  model monotone(rating) = class(x1-x5 / zero=sum);
  output dapproximations ireplace;
  weight w;
run;
```

Monotone Splines

This is a conjoint analysis that is more restrictive than a nonmetric analysis but less restrictive than a metric conjoint analysis. By default, the monotone spline transformation has two parameters (degree two with no knots).

```
proc transreg data=a utilities;
  model mspline(rating) = class(x1-x5 / zero=sum);
  output dapproximations ireplace;
  weight w;
run;
```

If less smoothness is desired, specify knots. For example:

```
proc transreg data=a utilities;
  model mspline(rating / nknots=3) = class(x1-x5 / zero=sum);
  output dapproximations ireplace;
  weight w;
run;
```

Each knot uses an extra *df*.

Constraints on the Utilities

Here is a metric conjoint analysis specification with linearity constraints imposed on X4 and monotonicity constraints imposed on X5.

```
proc transreg data=a utilities;
  model linear(rating) = class(x1-x3 / zero=sum)
    identity(x4) monotone(x5);
  output dapproximations ireplace;
  weight w;
run;
```

Here is another variation:

```
proc transreg data=a utilities;
  model linear(rating) = monotone(x1-x5) mspline(price);
  output dapproximations ireplace;
  weight w;
  run;
```

A Discontinuous Price Function

The utility of price may not be a continuous function of price. It has been frequently found that utility is discontinuous at round numbers such as \$1.00, \$2.00, \$100, \$1000, and so on. If PRICE has many values in the data set, say over the range \$1.05 to \$3.95, then a monotone function of price with discontinuities at \$2.00 and \$3.00 can be requested as follows.

```
proc transreg data=a utilities;
  model linear(ranking / reflect) = class(x1-x5 / zero=sum)
    mspline(price / knots=2 2 2 3 3 3);
  output dapproximations ireplace;
  weight w;
  run;
```

The monotone spline is degree two. The *order* of the spline is one greater than the degree; in this case the order is three. When the same knot value is specified *order* times, the transformation is discontinuous at the knot. Refer to Kuhfeld and Garratt (1992) for possible applications of splines to conjoint analysis.

More Than One Subject

Typically data are collected for many individuals and then analyzed individually. Each subject's ratings are entered as a separate dependent variable. Here is an example.

```
proc transreg data=a utilities;
  model linear(rate1-rate100) = class(x1-x5 / zero=sum);
  output dapproximations ireplace;
  weight w;
  run;
```

References

- Carroll, J.D. (1972), "Individual Differences and Multidimensional Scaling," in Shepard, R.N., Romney, A.K., and Nerlove, S.B. (Editors), *Multidimensional Scaling: Theory and Applications in the Behavioral Sciences* (Volume 1), New York: Seminar Press.
- Finkbeiner, C.T. (1988), "Comparison of Conjoint Choice Simulators," Sawtooth Software Conference Proceedings.
- Gifi, A. (1990), *Nonlinear Multivariate Analysis*, Chichester UK: Wiley.
- Green, P.E., and Rao, V.R. (1971), "Conjoint Measurement for Quantifying Judgmental Data," *Journal of Marketing Research*, 8, 355–363.
- Green, P.E., and Srinivasan, V. (1990), "Conjoint Analysis in Marketing: New Developments with Implications for Research and Practice," *Journal of Marketing*, 54, 3–19.
- Green, P.E., and Wind, Y. (1975), "New Way to Measure Consumers' Judgments," *Harvard Business Review*, July-August, 107–117.
- Kuhfeld, W.F., and Garratt, M. (1992), "Linear Models and Conjoint Analysis with Nonlinear Spline Transformations," Paper presented to the American Marketing Association Advanced Research Techniques Forum, Lake Tahoe, Nevada.
- Kuhfeld, W.F., Garratt, M., and Tobias, R.D. (1993), "Nonorthogonal Experimental Design Theory with Marketing Research Applications," Paper presented to the American Marketing Association Advanced Research Techniques Forum, Monterey, California.
- Louviere, J.J. (1988), *Analyzing Decision Making, Metric Conjoint Analysis*, Sage University Papers, Beverly Hills: Sage.
- Louviere, J.J. (1991), "Consumer Choice Models and the Design and Analysis of Choice Experiments," Tutorial presented to the American Marketing Association Advanced Research Techniques Forum, Beaver Creek, Colorado.
- Manski, C.F., and McFadden, D. (1981), *Structural Analysis of Discrete Data with Econometric Applications*. Cambridge: MIT Press.
- Tobias, R.D. (1992), "Saturated Second-Order Two-Level Designs: An Empirical Approach," unpublished manuscript.
- Wittink, D.R., Krishnamurthi L., and Reibstein D.J. (1989), "The Effect of Differences in the Number of Attribute Levels in Conjoint Results," *Marketing Letters*, 1:2, 113–123.
- Young, F.W. (1981), "Quantitative Analysis of Qualitative Data," *Psychometrika*, 46, 357–388.

Index

A

- A-efficiency, design 4
- abbreviations of option names 69
- active observations 76
- ADX, automated design of experiments 16
- algorithm options 70, 71
- aliased, design 4
- attributes 1
- average across subjects, importance 31, 33
- A-DEPEND variable name 71

B

- Bradley 54
- Bradley-Terry-Lucemodel
 - compared to other simulators 56
 - defined 3
 - market share 54
- brand by price interaction, plot 62
- BY statement 75

C

- candidate points, design 4
- canonical initialization 70
- cards, data collection 22
- cards, printing in a DATA step 21
- Carroll, J.D. 12
- change, market share 59, 62
- chocolate candy, example 5, 64
- choice simulators
 - Bradley-Terry-Lucemodel 3
 - compared 54, 56
 - defined 3
 - example 53, 54
 - logit model 3
 - macro 50
 - maximum utility model 3, 50
- choice study 64
- CLASS
 - example 5, 9, 25, 47, 59, 62
 - expansion 72
 - sample specification 77-79
 - transformation 72
 - variable names 70

- combinations
 - printing in a DATA step 21
 - unrealistic 34
- confounded, design 4
- conjoint analysis
 - defined 2
 - model 7
 - typical options 77
- conjoint measurement 1
- conservative, degrees of freedom 11
- constrained
 - part-worth utilities 78
 - utilities 78
- consumer choice study 64
- CONVERGE=, option 70
- convergence criterion 70
- CORR procedure 29
- CPREFIX=, option 70

D

- D-efficiency, design 4, 36
- DAPPROXIMATIONS
 - example 14, 25, 47, 59, 62
 - option 72
 - predicted utilities 72
 - sample specification 77-79
- data collection
 - cards 22
 - FSEDIT procedure 43
 - interactive 39, 40, 43
 - rank-order data 22
- data entry; rank-order data 22
- data validation, rank-order data 22
- DATA=
 - example 25, 47, 59, 62
 - option 70
 - sample specification 77-79
- degree, spline 74
- DEGREE=
 - example 47, 59, 62
 - option 74
- degrees of freedom
 - conservative 11
 - liberal 11
 - nonmetric conjoint analysis 9

- usual 11
- _DEPEND_** variable name 71
- design**
 - A-efficiency 4
 - aliased** 4
 - candidate points 4
 - confounded 4
 - D-efficiency 4, 36
 - design points 4
 - factors 3
 - fractional-factorial 4
 - full-factorial 3, 34
 - information matrix 4
 - levels 3
 - nonorthogonal 33
 - orthogonal array 4
 - response 3
 - runs 4
- design creation
 - example 16
 - factor names and levels 17
 - holdouts** 19
 - label 17
 - labels and formats 18
 - orthogonal array 17
 - randomization 17
 - response variable 18
- design points, design** 4
- discontinuous function
 - price 46
 - sample specification 79
- discrete choice study 64
- DUMMY**
 - example 47, 59, 62
 - option 70

E

- efficiency
 - A-efficiency 4
 - D-efficiency 4
- EVENLY**, option 74
- evenly spaced, knots 74
- example
 - ADX menu system 16
 - average R-square 31, 33
 - Bradley-Terry-Lute model 54
 - brand by price interaction 62
 - chocolate candy 5, 64
 - choice study 64
 - FSEDIT procedure 39, 40
 - logit model 54
 - market share 50, 54, 59
 - maximum utility model 50
 - metric conjoint analysis 5, 13, 24, 46
 - multinomial logit model 64, 66, 68
 - new products 59, 60
 - nonmetric conjoint analysis 9

- nonorthogonal design 34, 36
- orthogonal array 16
- simulation 50, 54, 59
- spaghetti sauce 33
- stimulus creation 21, 22
- tea tasting (advanced) 16
- tea tasting (basic) 12
- expansion
 - CLASS** 72
 - polynomial spline 74

F

- factor names and levels, design creation 17
- factors, design 3
- Federov 35
- Finkbeiner 56
- FORMAT** procedure 13, 18, 23, 67
- fractional-factorial, design 4
- FSEDIT** procedure
 - data collection 43
 - screen customization 40, 41
- full-factorial, design 3, 34

G

- Garratt, M. 5, 79
- Gifi, A. 3
- GPLOT** procedure 11, 57, 62
- Green, P.E. 1

H

- holdouts**
 - design creation 19
 - example 21
 - validation 29
- Hotelling-Lawley Trace 9
- hypothesis tests 71

I

- ID** statement
 - example 47, 59, 62
 - syntax 76
- IDENTITY**
 - example 62
 - sample specification 78
 - transformation 73
- importance
 - average across subjects 31, 33
 - defined 8
 - inflated 8
 - OUTTEST=** 70
- independence 15
- individual, R-square 31, 32
- inflated, importance 8
- information matrix, design 4

interaction 62
 interval, variables 73
IREPLACE
 example 14, 25, 47, 59, 62
 option 72
 replacing independent variables 72
 sample specification 77-79
 iteration
 history suppressed 71
 maximum number of 70
 metric conjoint analysis 6
 nonmetric conjoint analysis 9
 iterative algorithm 71

K

Kendall Tau 29
 knots
 evenly spaced 74
 number of 75
 sample specification 79
 specifications 75
KNOTS=
 example 47, 59, 62
 option 75
 Krishnamurthi, L. 8
 Kuhfeld, W.F. 5, 79

L

label, design creation 17
 labels and formats, design creation 18
 levels, design 3
 liberal, degrees of freedom 11
LINEAR
 example 5, 14, 25, 47, 59, 62
 sample specification 77-79
 transformation 73
 linear attribute, sample specification 78
logit, transformation 73
logit model
 compared to other simulators 56
 defined 3
 market share 54
 Louviere, J.J. 1, 64
Luce 5 4

M

Manski, C.F. 64
 market share
 Bradley-Terry-Lucemodel 3, 54
 change 59, 62
 example 50
 logit model 3, 54
 maximum utility model 3, 50
 simulating 50
 maximum number of iterations 70

maximum utility model
 compared to other simulators 56
 defined 3
 example 50
MAXITER=, option 70
 McFadden, D. 64
METHOD=
 MORALS example 25, 47, 59, 62
 option 71
 UNIVARIATE example 62
 metric conjoint analysis
 defined 2
 example 5, 6, 13, 14, 24, 46
 iteration 6
 sample specification 77
 versus nonmetric 2, 77
 missing values 73
MODEL statement
 example 5, 9, 14, 25, 47, 59, 62
 options 70
 sample specification 77-79
 transformation 72
 transformation options 74
MONOTONE
 example 9
 sample specification 78, 79
 transformation 73
 monotone attribute, sample specification 78
 monotone spline
 sample specification 78, 79
 transformation 73
MORALS algorithm 71
 more than one subject, sample specification 79
MSPLINE
 sample specification 78, 79
 transformation 73
 multinomial **logit** model, example 64, 66, 68
 multivariate statistics 9

N

new products, example 59, 60
NKNOTS=
 option 75
 sample specification 78
 naminal, variables 72
 nonmetric conjoint analysis
 defined 2
 degrees of freedom 9
 example 9, 10
 iteration 9
 sample specification 78
 versus metric 2, 77
 nonorthogonal, design 33
 normality 15
 number of, runs 34

O

OPTEX procedure 35
 order, spline 79
 ordinal, variables 73
 orthogonal array
 design 4
 design creation 17
 example 12
 OUT=
 example 25, 47, 62
 option 72
 predicted utilities 72
 transformation 72
 output options 72
 OUTPUT statement
 example 25, 47, 59, 62
 options 72
 sample specification 77-79
 OUTTEST=
 example 25
 importance 70
 option 70
 part-worth utilities 70
 R-square 70
 used 31
 utilities 70

P

part-worth utilities
 constrained 78
 defined 2
 outputting predicted 72
 OUTTEST= 70
 printing 71
 summing to zero 75
 Pearsonr 29
 perfect fit 15
 PHREG procedure 65
 Pillai's Trace 9
 PLAN procedure 35
 plot
 brand by price interaction 62
 transformation 9
 polynomial spline, expansion 74
 predicted utilities
 DAPPROX 72
 OUT= 72
 variables 27, 28
 price
 discontinuous function 46
 sample specification 79
 PROC statement 69
 PSPLINE, transformation 74

R

R-square
 individual 31, 32
 OUTTEST= 70
 randomization, design creation 17
 rank, transformation 74
 rank-order data
 data collection 22
 dataentry 22
 data validation 22
 REFLECT 77
 sample specification 77
 versus rating-scale data 77
 Rao, V.R. 1
 rating-scale data
 sample specification 77
 versus rank-order data 77
 REFLECT
 example 14, 25
 option 75
 rank-order data 77
 sample specification 77, 79
 reflection 14, 75
 Reibstein, D.J. 8
 replacing independent variables, IREPLACE
 72
 residual variables, RESIDUALS 72
 RESIDUALS
 example 62
 option 72
 residual variables 72
 response, design 3
 response variable, design creation 18
 rolled out data set 71
 runs
 design 4
 number of 34

S

sample specification
 discontinuous function 79
 linear attribute 78
 metric conjoint analysis 77
 monotone attribute 78
 monotone spline 78, 79
 more than one subject 79
 nonmetric conjoint analysis 78
 price 79
 rank-order data 77
 rating-scale data 77
 screen customization, FSEDIT procedure
 40, 41
 SHORT
 example 14, 25, 47, 59, 62
 option 71
 SIM macro 50

simulating, market share 50
 simulation, example 50
 simulation observations 23, 30, 76
 simulators
 Bradley-Terry-Luce model 3
 compared 54, 56
 example 53, 54
 logit model 3
 macro 50
 maximum utility model 3, 50
 spaghetti sauce, example 33
 specifications 69-76
 spline
 degree 74
 example 47, 59, 62
 order 79
 transformation 74
Srinivasan, V. 1
 stimulus creation, DATA step 21
 summing to zero
 part-worth utilities 75
 utilities 75

T

tea tasting
 advanced example 16
 basic example 12
 Terry 54
 TEST
 example 47
 option 71
 tests of hypothesis 71
Tobias, R.D. 5
 trade-offs 1
 transformation
 class 72
 linear 73
 logit 73
 monotone 73
 monotone spline 73
 none 73
 options 74, 75
 OUT= 72
 plot 9
 rank 74
 spline 74
TRANSREG procedure
 advanced sample 78, 79
 brand by price interaction 62
 discontinuous price sample 79
 monotone spline sample 78
 nonmetric example 9
 nonmetric sample 78
 price spline function 47
 rank data sample 77
 rating data sample 77
 simple example 5, 14

 simulation 59
 specifications 69
 typical example 25
 typical sample 79
 typical options, conjoint analysis 77
 T-DEPEND variable name 71

U

UNIVARIATE algorithm 71
 unrealistic, combinations 34
 usual, degrees of freedom 11
 utilities
 constrained 78
 defined 2, 7
 example 5, 14, 25
 option 71
 outputting predicted 72
 OUTTEST= 70
 predicted 27, 28
 printing 71
 sample specification 77-79
 summing to zero 75

V

validation, holdouts 29
 variables
 interval 73
 nominal 72
 ordinal 73
 predicted utilities 27
 replacing in OUT= 72
 residuals in OUT= 72

W

WEIGHT statement
 example 25, 59, 62
 holdouts 76
 sample specification 77-79
 weighted loss function 76
 Wilks' Lambda 9
 Wind, Y. 1
 Wittink, D.R. 8

Y

Young, F.W. 3

Z

ZERO=
 option 75
 ZERO=SUM example 5, 9, 14, 25, 47,
 59, 62
 ZERO=SUM sample specification 77-
 79

