Graphical Scatter Plots of Labeled Points

Warren F. Kuhfeld

Abstract

The %PlotIt (PLOT ITeratively) macro creates graphical scatter plots of labeled points. It is designed to make it easy to display raw data, regressions, and the results of many other data analyses. You can draw curves, vectors, and circles, and you can control the colors, sizes, fonts, and general appearance of the plots. The %PlotIt macro is a part of the SAS autocall library.*

Introduction

SAS has provided software for producing scatter plots for many years (for example, the PLOT and GPLOT procedures). For many types of data analyses, it is useful to have each point in the plot labeled with the value of a variable. However, before the creation of the %PlotIt macro, there was not a satisfactory way to do this. PROC GPLOT produces graphical scatter plots. Combined with the Annotate facility, it allows long point labels, but it does not provide any way to optimally position them. The PLOT procedure can optimally position long point labels in the scatter plot, however PROC PLOT cannot create a graphical scatter plot. The PROC PLOT label-placement algorithm was developed by Kuhfeld (1991), and the PROC PLOT options are documented in Base SAS documentation.

The macro, %PlotIt (PLOT ITeratively), creates graphical scatter plots of labeled points. It can fit curves, draw vectors, and draw circles. It has many options, but only a small number are needed for many types of plots. The %PlotIt macro uses DATA steps and multiple procedures, including PLOT and GANNO. The %PlotIt macro is almost 6000 lines long, so it is not displayed here. It is fully documented starting on page 1178 and in the header comments. This article illustrates through examples some of the main features of the %PlotIt macro.

For many years, the %PlotIt macro provided the only convenient way to get graphical scatter plots of labeled points. With SAS Version 9.2, however, there are other and in many cases better options for making these plots. Now, ODS Graphics automatically does much of what the %PlotIt macro was originally designed for, and usually, ODS Graphics does it better and more conveniently. Comparisons between the %PlotIt macro and ODS Graphics are as follows. The %PlotIt macro has a much more

*Copies of this chapter (MR-2010K), the other chapters, sample code, and all of the macros are available on the Web http://support.sas.com/resources/papers/tnote/tnote_marketresearch.html. This chapter originally appeared in the SAS Journal **Observations**, Fourth Quarter, 1994, pages 23–37. This version of the chapter has been updated for SAS Version 9.2. For help, please contact SAS Technical Support. See page 25 for more information.

sophisticated algorithm for placing labels and avoiding label collisions. The %PlotIt macro has a few other capabilities that are not in ODS Graphics (e.g. more sophisticated color ramps or "painting" for some applications). ODS Graphics is superior to the %PlotIt macro in virtually every other way, and ODS Graphics has *many* capabilities that the %PlotIt macro does not have. The algorithm that ODS Graphics has for label placement in 9.2, while clearly nonoptimal, is good enough for many analyses. This paper illustrates the traditional uses of the %PlotIt macro and points out cases where ODS Graphics is a suitable alternative. There is an appendix at the end of this chapter showing many of the same plots that the %PlotIt macro made, but this time made with ODS Graphics.

An Overview of the %PlotIt Macro

The %PlotIt macro performs the following steps.

1. It reads an input data set and preprocesses it. The preprocessed data set contains information such as the axis variables, the point-symbol and point-label variables, and symbol and label types, sizes, fonts, and colors. The nature of the preprocessing depends on the type of data analysis that generated the input data set. For example, if the option DATATYPE=MDPREF was specified with an input data set created by PROC PRINQUAL for a multidimensional preference analysis, then the %PlotIt macro creates by default blue points for _TYPE_ = 'SCORE' observations and red vectors for _TYPE_ = 'CORR' observations.

2. A DATA step, using the DATA Step Graphics Interface, determines how big to make the graphical plot.

3. PROC PLOT determines where to position the point labels. The results are sent to output SAS data sets using ODS. By default, if some of the point label characters are hidden, the %PlotIt macro recreates the printer plot with a larger line and page size, and hence creates more cells and more room for the labels.

4. The PROC PLOT output data sets are read, and information from them and the preprocessed data set are combined to create an Annotate data set. The label position information is read from the PROC PLOT output, and all of the symbol, size, font, and color information is extracted from the preprocessed data set. The Annotate data set contains all of the instructions for drawing the axes, ticks, tick marks, titles, point symbols, point labels, axis labels, and so on.

5. The Annotate data set is displayed with the GANNO procedure. The %PlotIt macro does not use PROC GPLOT.

With the %PlotIt macro, you can:

- display plots and create graphics stream files and gout= entries
- easily display the results of correspondence analysis, multidimensional preference analysis, preference mapping, multidimensional scaling, regression analysis, and density estimation
- use single or multi-character symbols and control their color, font, and size
- use multi-character point labels and control their color, font, and size
- use fixed, variable, and random colors, and use colors to display changes in a third dimension
- automatically determine a good line size, page size, and list of point label placements
- automatically equate the axes for all devices
- control the colors, sizes, fonts, and general appearance of all aspects of the plot

- pre- and post-process the data
- specify many `goptions`

Since `%PlotIt` is a macro, you can modify it, change the defaults, add new options, and so on. The `%PlotIt` macro is heavily commented to make it easier for you to modify it to suit your needs. There is substantial error checking and options to display intermediate results for debugging when you do not get the results you expect. Furthermore, you have complete access to all of the data sets it creates, including the preprocessed version of the input and the Annotate data set. You can modify the results by editing the Annotate and preprocessed data sets.

Changes and Enhancements

The main enhancement is the `%PlotIt` macro has been modified to produce plots that look more like graphs created by ODS Graphics. Primarily, this means that the appearance of the graph is at least in part sensitive to information in ODS styles. You can specify `style=` as an option on any ODS destination statement, for example: `ods html style=statistical;`. By default, the plots are produced using a default style, which depends on the destination. For the HTML destination, the default style is literally named `default`. If you specify `options nogstyle;`, then style information is not used, and the macro should do what it did previously. However, the previous output, particularly the fonts and the colors, are not as nice as the newer style-sensitive appearance.

Examples

This section contains examples of some of the capabilities of the `%PlotIt` macro. Rather than interpreting the plots or discussing the details of the statistical analyses, this section concentrates on showing what the `%PlotIt` macro can do. Most of the examples are based on SAS/STAT example data sets. Data for all of the examples can be found in the SAS/STAT sample program `plotitex.sas`.

Example 1: Principal Components of Mammal's Teeth
Principal component analysis computes a low-dimensional approximation to a set of data. Principal components are frequently displayed graphically. This example is based on the Mammal's Teeth data set. To perform a principal component analysis, specify:

```
proc princomp data=teeth out=scores(keep=prin1 prin2 mammal);
   title "Principal Components of Mammals' Teeth";
   run;

%plotit();
```

The plot is shown in Figure 1. No options were specified in the `%PlotIt` macro, so by default a plot is constructed from the first two numeric variables and the last character variable in the last data set created. The `%PlotIt` macro displayed the following information in the log:

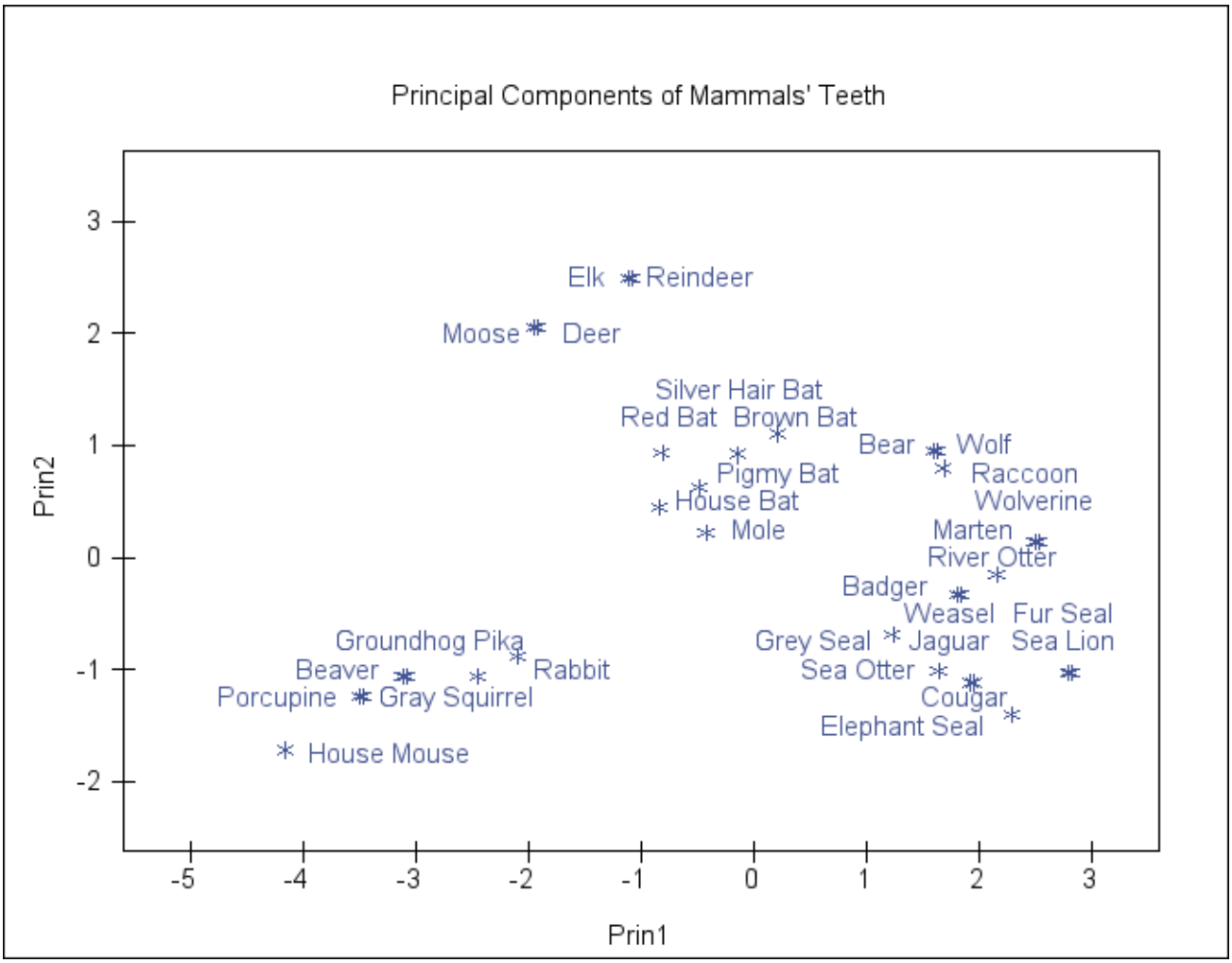

Figure 1. Principal Components

Iterative Scatter Plot of Labeled Points Macro

Iteration	Place	Line Size	Page Size	Penalty
1	2	65	45	34
2	3	80	50	0

The following code will create the printer plot on which the graphical plot is based:

```
options nonumber ls=80 ps=50;
proc plot nolegend formchar='|----|+|---' data=preproc vtoh=2;
   plot Prin2 * Prin1 $ mammal = _symbol_ /
        haxis=by 1 vaxis=by 1 box list=1
        placement=((h=2 -2 : s=right left) (v=1 to 2 by alt * h=0 -1 to -10
        by alt));
   label Prin2 = '#' Prin1 = '#';
   run; quit;

The plot was created with the following goptions:

goptions reset=goptions erase hpos=99 vpos=34 hsize=11.71in vsize=7.98in
         device=WIN;

The OUT=anno Annotate data set has 148 observations.
The PLOTIT macro used 1.7seconds to create OUT=anno.
```

The iteration table shows that the `%PlotIt` macro tried twice to create the plot, with line sizes of 65 and 80. It stopped when all point label characters were plotted with zero penalty.* The `%PlotIt` macro displays PROC PLOT code for the printer plot, on which the graphical plot is based. It also displays the `goptions` statement that was used with PROC GANNO.†

There are several notable features of the plot in Figure 1.

1. Symbols for several pairs of points, such as Elk and Reindeer, are coincident. By default, the `%PlotIt` macro slightly offsets coincident symbols so that it is clear that more than one point maps to the same location.
2. Point labels map into discrete rows, just as they would in a printer plot produced by PROC PLOT. However, unlike printer plots, the `%PlotIt` macro uses proportional fonts.
3. Symbols are not restricted to fixed cells. Their mapping is essentially continuous, more like PROC GPLOT's than PROC PLOT's.
4. A fixed distance represents the same data range along both axes, which means the axes are equated so that distances and angles will have meaning. In contrast, procedures such as PLOT and GPLOT fill the available space by default, so the axes are not equated.

Example 2: Principal Components of Crime Rates

A typical plot has for each point a single-character symbol and a multi-character label; however, this is not required. This example is based on the Crime data set. The point labels are state names, and the symbol for each label is a two-character postal code.

*Penalties accrue when point labels are nonoptimally placed, such as when two label characters map to the same location. PROC PLOT tries to minimize the penalties for all point labels. See PROC PLOT documentation for more information.

†This code could be different depending on your device. By default, the macro uses your default device.

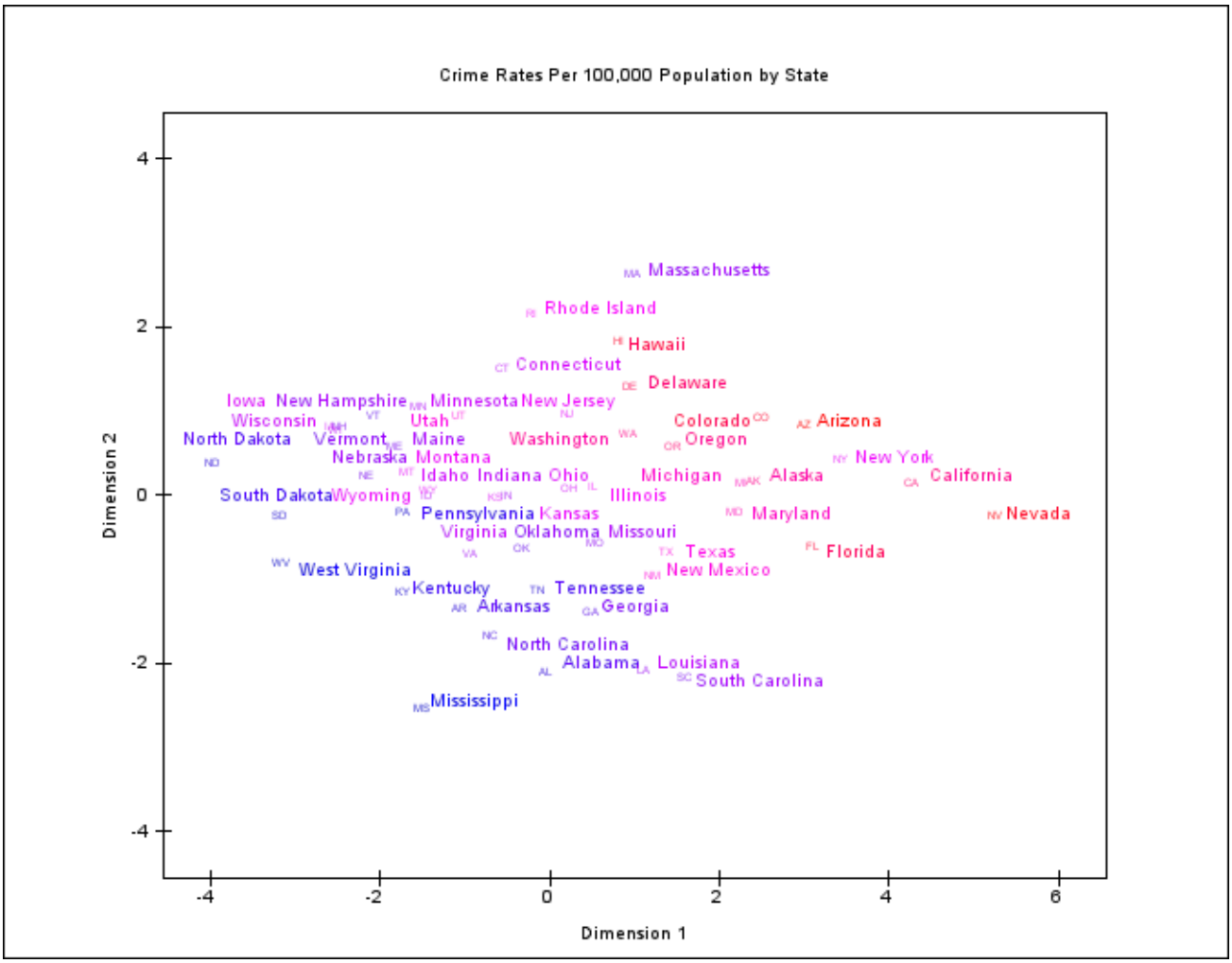

Figure 2. Principal Components with the Third Component "Painted"

```
proc princomp data=crime out=crime2;
   title 'Crime Rates Per 100,000 Population by State';
   run;

%plotit(data=crime2, plotvars=prin2 prin1,
        symvar=postcode, symlen=2, symsize=0.6, paint=larceny,
        labelvar=state, label=typical)
```

This plot request specifies:

- the input data set: `crime2`
- the y-axis and x-axis variables: `prin2` and `prin1`
- the symbol variable: `postcode`
- the number of symbol characters: 2
- the size of the symbol font in the plot: 0.6
- the colors are based on the variable: `larceny`
- the point label variable: `state`
- the typical method of generating variable labels for the plot axes

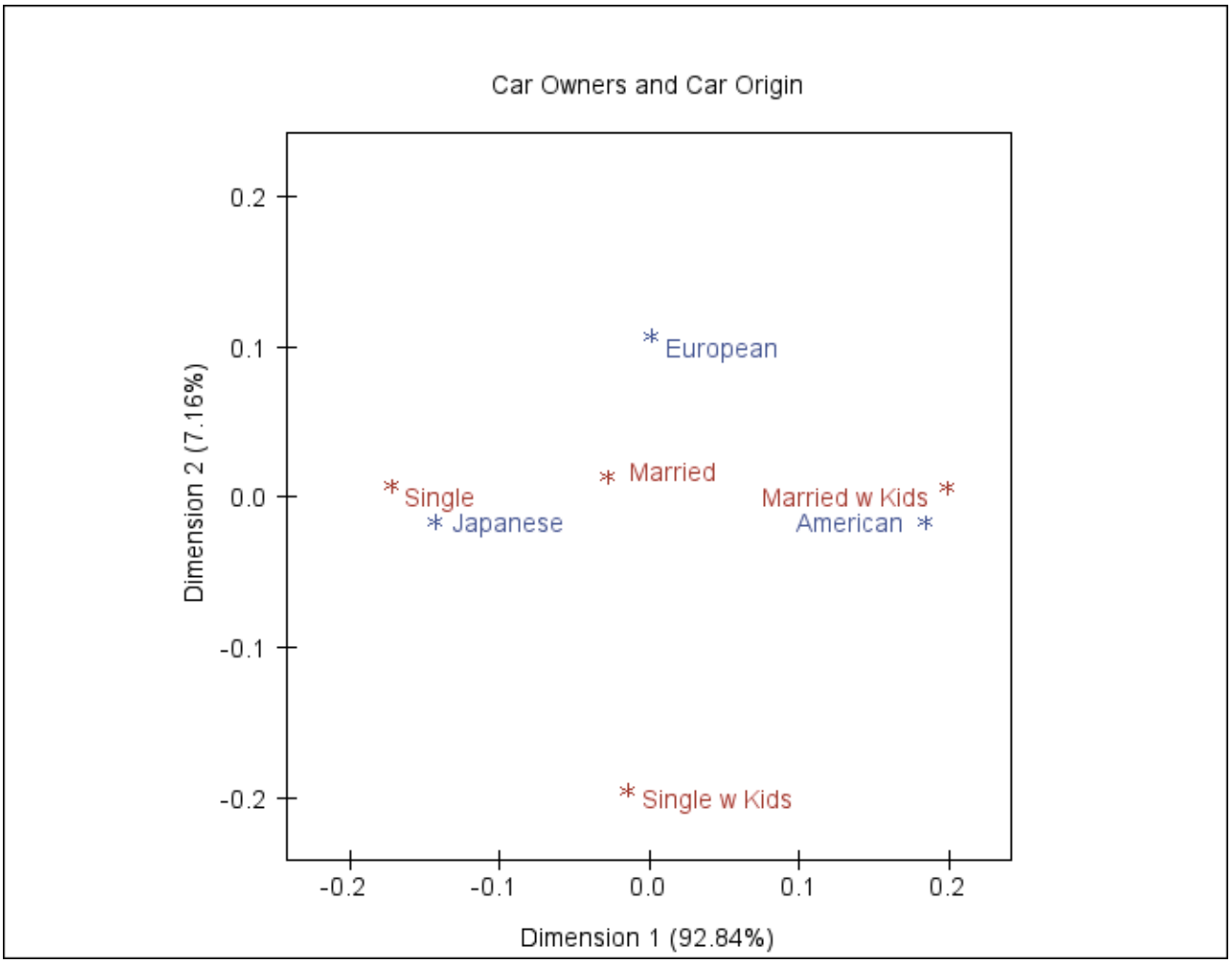

Figure 3. Simple Correspondence Analysis

A symbol size of 0.6 instead of the normal 1.0 is specified to make the symbol small, because two characters are mapping to a location where there is usually just one. The option `paint=larceny` creates interpolated label and symbol colors, by default between blue, magenta, and red, so that states that have a low larceny rate are blue and high-rate states are red. `Label=typical` for variables `prin2` and `prin1` generates the following label statement:

```
label prin2 = 'Dimension 2' prin1 = 'Dimension 1';
```

This plot request is much more complicated than most. Often, you need to specify only the type of analysis that generated the data set. The plot is shown in Figure 2.

Examples 3 & 4: Correspondence Analysis of the Car Ownership Survey
Correspondence analysis graphically displays crosstabulations. These examples use the Car Survey data. To perform a correspondence analysis, specify:

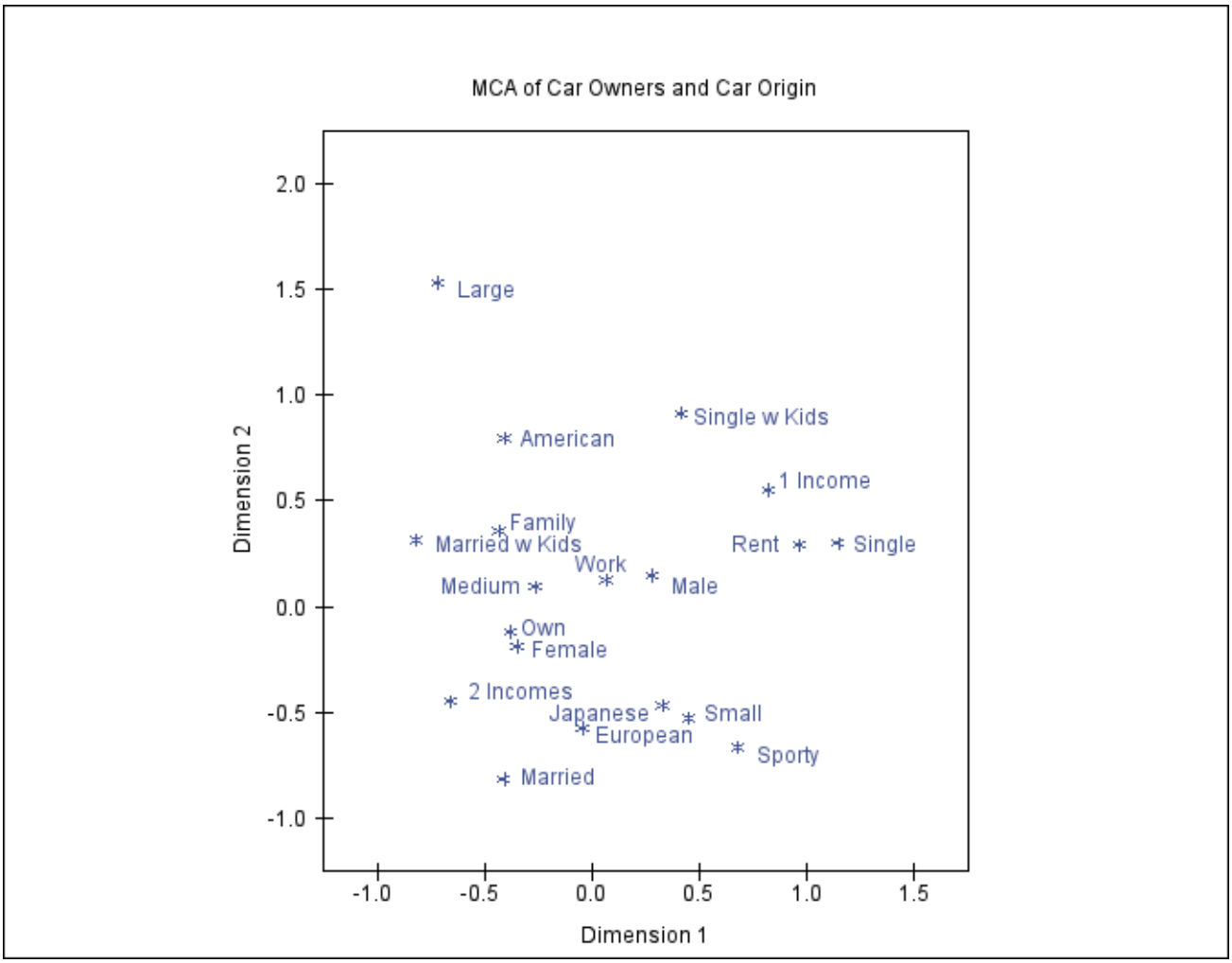

Figure 4. Multiple Correspondence Analysis

```
proc corresp data=cars outc=coors;
   title 'Car Owners and Car Origin';
   tables marital, origin;
   run;

%plotit(data=coors,datatype=corresp)
```

The plot is shown in Figure 3. With `datatype=corresp`, the `%PlotIt` macro automatically incorporates the proportion of inertia* into the axis labels and plots the row points in red and the column points in blue.

For a multiple correspondence analysis, specify:

```
proc corresp mca observed data=cars outc=coors;
   title 'MCA of Car Owners and Car Origin';
   tables origin size type income home marital sex;
   run;
```

*Inertia in correspondence analysis is analogous to variance in principal component analysis.

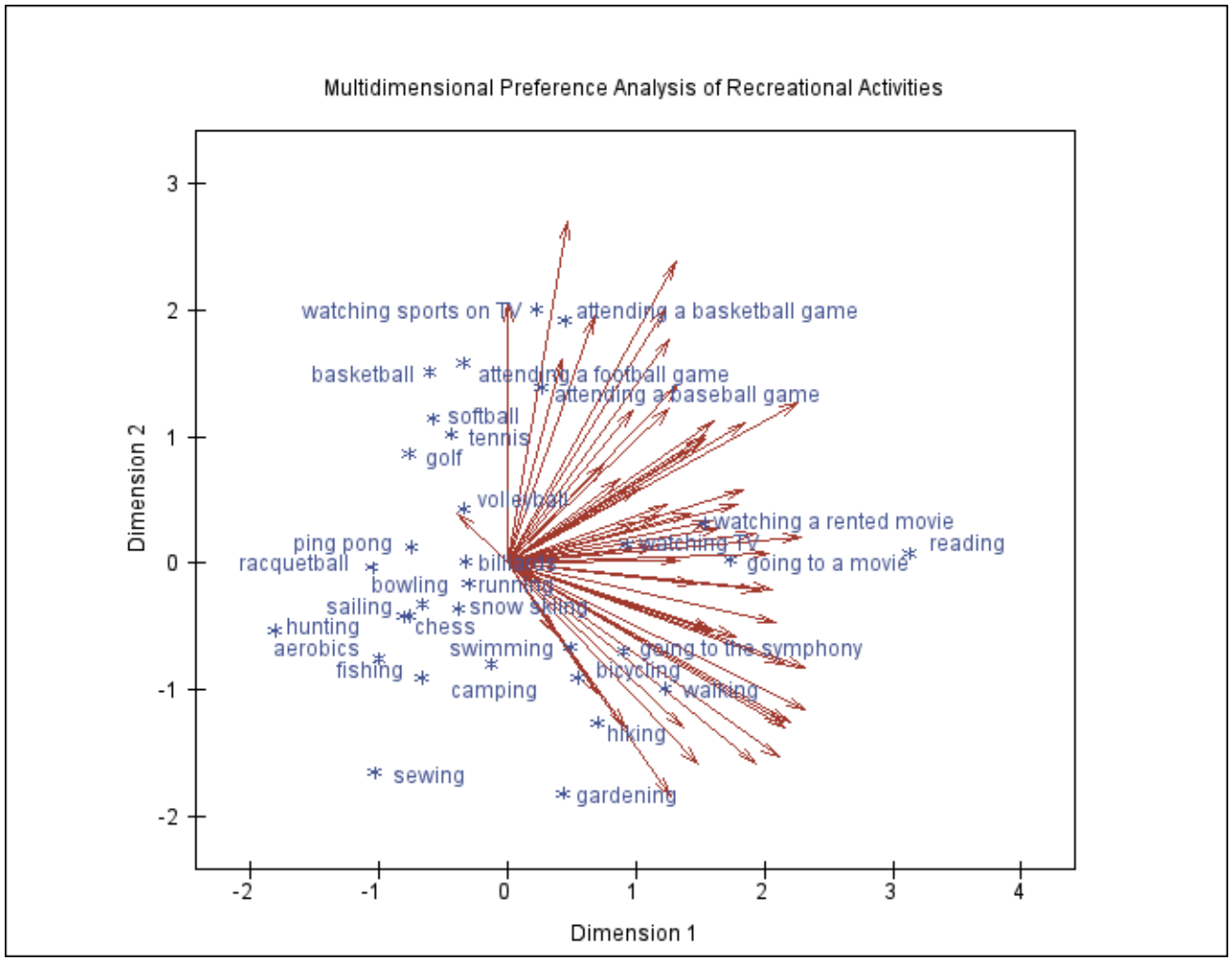

Figure 5. Multidimensional Preference Analysis

```
%plotit(data=coors,datatype=mca)
```

The plot is shown in Figure 4.

Alternatively, you can specify the `source` option to create a variable `_var_`, which you can use as a type variable to get different colors for each set of categories corresponding to each of the input variables, for example as follows:

```
proc corresp mca observed data=cars outc=coors;
   title 'MCA of Car Owners and Car Origin';
   tables origin size type income home marital sex;
   run;

%plotit(data=coors, plotvars=dim2 dim1, typevar=_var_)
```

The results of this step are not shown.

Examples 5 & 6: Multidimensional Preference Analysis of Recreational Activities
Multidimensional preference analysis is a variant on principal component analysis that simultaneously

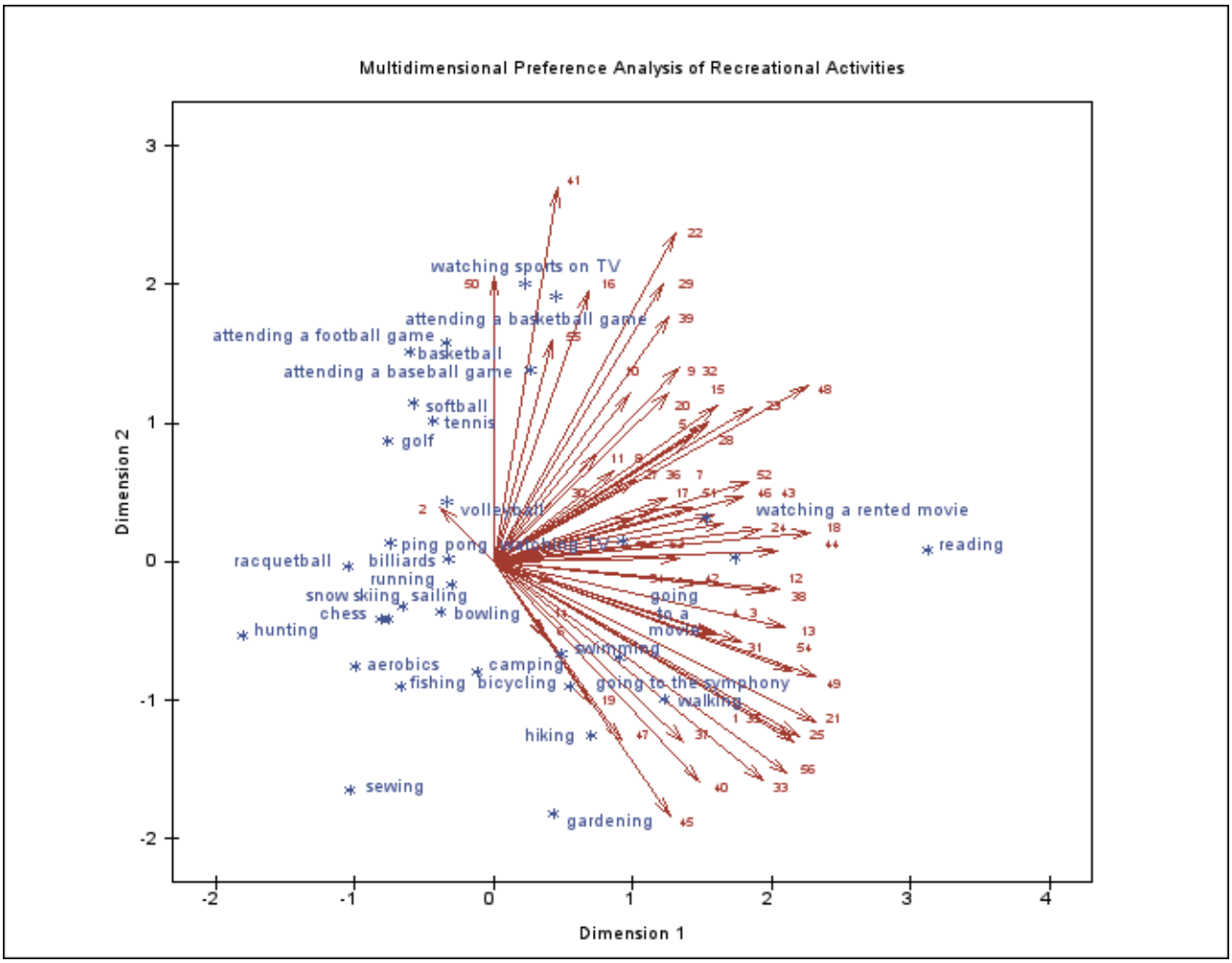

Figure 6. Multidimensional Preference Analysis

displays people and their preferences for objects. Each person is a variable in the input data set, and each object is a row. Each person is represented in the plot as a vector that points in approximately the direction of his or her most preferred objects. These examples use the Preferences for Recreational Activities data set. For multidimensional preference analysis, specify:

```
proc prinqual data=recreate out=rec mdpref rep;
   title1 'Multidimensional Preference Analysis of Recreational Activities';
   transform identity(sub1-sub56);
   id activity;
   run;

%plotit(data=rec,datatype=mdpref 3)
```

The plot is shown in Figure 5. With `datatype=mdpref`, the `%PlotIt` macro automatically displays the people as vectors and the activities as points (based on the `_TYPE_` variable). The `3` after the MDPREF is a scaling factor for the vectors. The lengths of all the vectors are increased by a factor of three to create a better graphical display. You can also label the vectors by specifying:

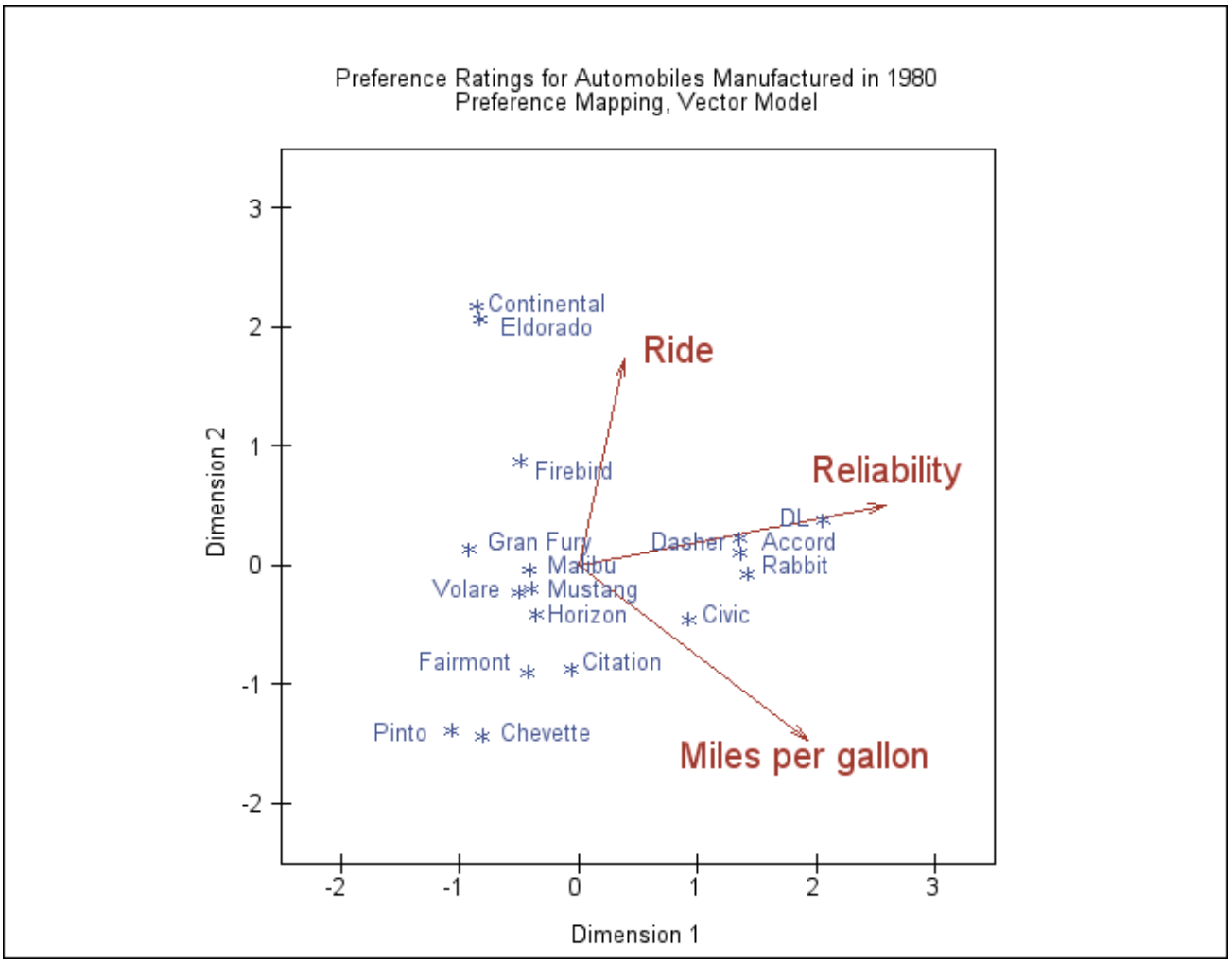

Figure 7. Preference Mapping, Vector Model

```
%plotit(data=rec,datatype=mdpref2 3)
```

MDPREF2 specifies a MDPREF analysis with labeled vectors (the `2` means labels *too*). This plot is not shown because in this input data set, each subject is identified by a variable name of the form `sub1`, `sub2`, ..., and the graphical display looks cluttered with all those `sub`'s. The default point label variable is the ID statement variable `activity`, because it is the last character variable in the data set. PROC PRINQUAL fills in this variable for the `_TYPE_ = 'CORR'` observations (the people that plot as vectors) with the variable names: `sub1-sub56`. You can preprocess the input data set directly in the `%PlotIt` macro to remove the `sub`'s as follows:

```
%plotit(data=rec,datatype=mdpref2 3,
        adjust1=%str(if _type_ = 'CORR' then
                       activity = substr(activity,4);))
```

The plot is shown in Figure 6. The `adjust1` option adds DATA step statements to the end of the preprocessing step. By default, the `%PlotIt` macro tries to position the vector labels outward, not between the vector head and the origin.

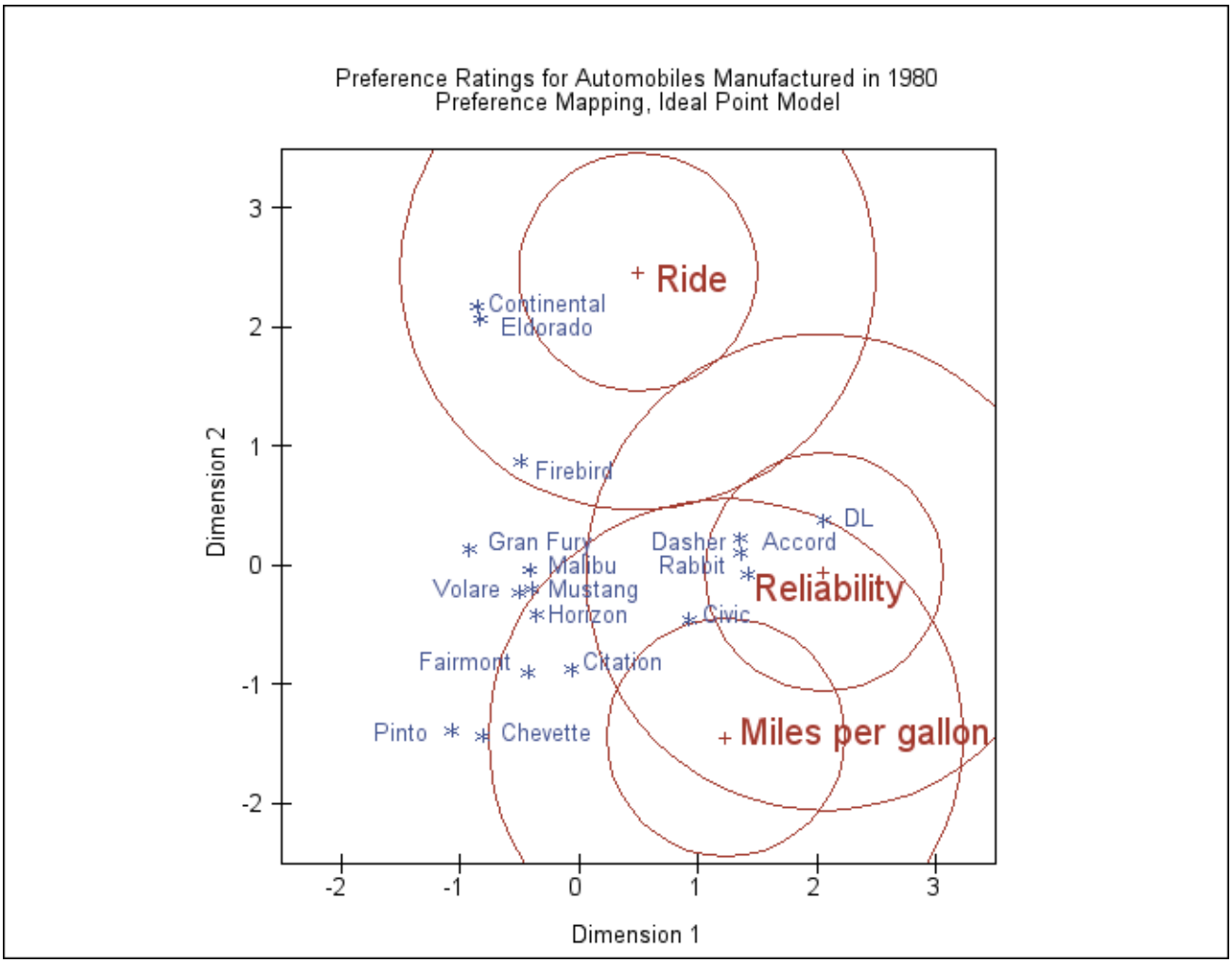

Figure 8. Preference Mapping, Ideal Point Model

Examples 7 & 8: Preference Mapping of Cars

Preference mapping simultaneously displays objects and attributes of those objects. These examples use the Car Preference data set to illustrate preference mapping. The following code fits a preference mapping vector model:

```
* Compute Coordinates for a 2-Dimensional Scatter plot of Cars;
proc prinqual data=carpref out=presults(drop=judge1-judge25) n=2
              replace mdpref;
   title 'Preference Ratings for Automobiles Manufactured in 1980';
   id model mpg reliable ride;
   transform identity(judge1-judge25);
   run;
```

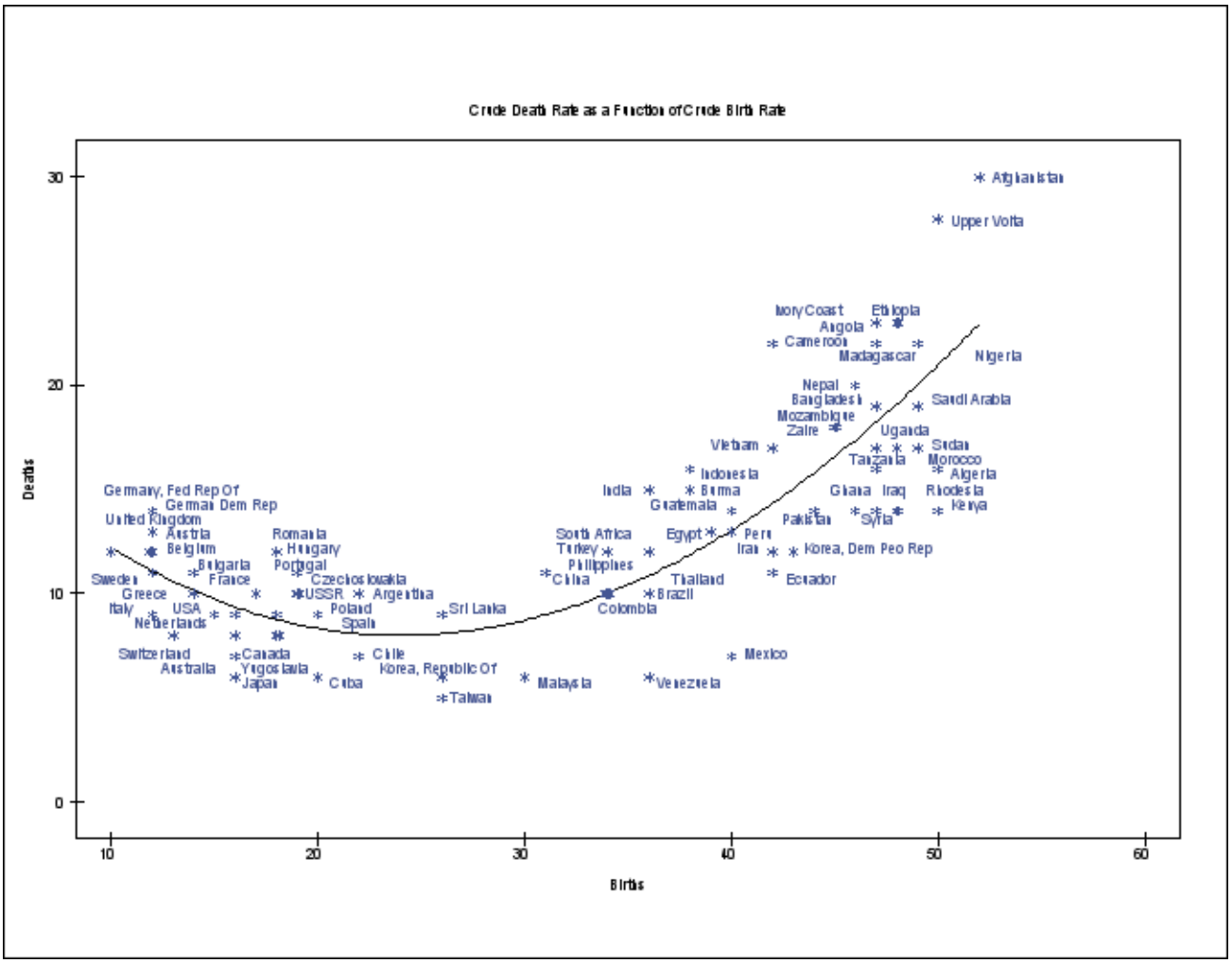

Figure 9. Spline Regression Model

```
* Compute Endpoints for Vectors;
proc transreg data=presults;
   title2 'Preference Mapping, Vector Model';
   model identity(mpg reliable ride)=identity(prin1 prin2);
   output tstandard=center coefficients replace out=vector;
   id model;
   run;


%plotit(data=vector,datatype=vector 2.5)
```

The plot is shown in Figure 7. Each attribute is represented as a vector that points in approximately the direction of the objects with larger values of the attribute. The `datatype=vector 2.5` option requests the vector model, with the vectors stretched by a factor of 2.5. Alternatively, you can represent attributes as points by specifying:

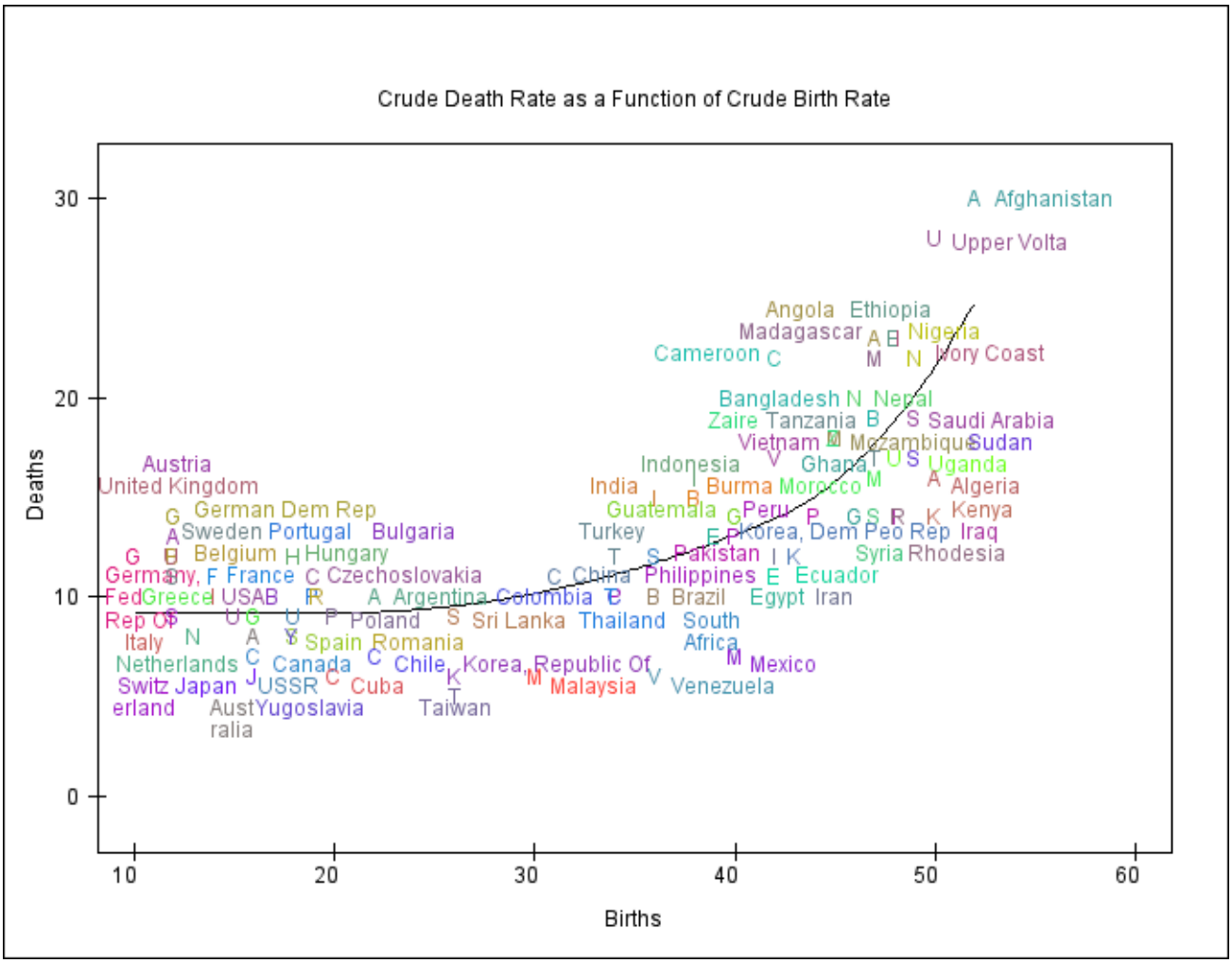

Figure 10. Spline Regression Model

```
* Compute Ideal Point Coordinates;
proc transreg data=presults;
   title2 'Preference Mapping, Ideal Point Model';
   model identity(mpg reliable ride)=point(prin1 prin2);
   output tstandard=center coordinates replace out=ideal;
   id model;
   run;

%plotit(data=ideal,datatype=ideal,antiidea=1)
```

The plot is shown in Figure 8. The option `datatype=ideal` requests a preference mapping, with each attribute represented as an ideal point. Circles are drawn to show distances between the cars and the ideal points, which are locations of hypothetical cars that have the ideal amount of the attribute. The `antiidea=1` option specifies how anti-ideal points are recognized.* By default, the labels for the attributes are larger than the other point labels and hence sometimes extend slightly beyond the plot. This happened with "Miles per gallon" in Figure 8. You can move the label up one character unit and

*Anti-ideal points have their signs wrong, so the macro must reverse them before plotting. When small ratings are good, specify `antiidea=-1`, and when small ratings are not good, specify `antiidea=1`.

to the left 12 character units by adding the following option:

```
adjust4=%str(if text =: 'Miles' then do; y = y + 1; x = x - 12; end;)
```

The `adjust4` option adds DATA step statements to the end of the final Annotate DATA step. The `%PlotIt` macro has no sense of aesthetics; sometimes a little human intervention is needed for the final production plots.

Examples 9 & 10: Curve Fitting of Birth and Death Rates
It is often useful to display a set of points along with a regression line or nonlinear function. The `%PlotIt` macro can fit and display lines and curves (and optionally display the regression and ANOVA table). These examples use the Vital Statistics data set. The following requests a cubic-polynomial regression function:

```
title 'Crude Death Rate as a Function of Crude Birth Rate';

%plotit(data=vital,vtoh=1.75,datatype=curve2)
```

The plot is shown in Figure 9. The option `vtoh=1.75` specifies the PROC PLOT aspect ratio (vertical to horizontal). The default is 2.0. Smaller values create plots with more cells for label characters, which is helpful when the point cloud is relatively dense. The option `datatype=curve2` instructs the `%PlotIt` macro to fit a curve and have the point labels avoid the curve (the `2` means label avoidance *too*).

You can control the type of curve. The `%PlotIt` macro uses PROC TRANSREG to fit the curve, and you can specify PROC TRANSREG options. For example, to request a monotone spline regression function with two knots, specify:

```
%plotit(data=vital,datatype=curve,bright=128,maxiter=4,
        symvar=country,regfun=mspline,nknots=2)
```

The plot is shown in Figure 10. There are several differences between Figures 9 and 10, in addition to the difference in the regression function. The option `datatype=curve` was specified, not `datatype=curve2`, so there is more overlap between the point labels and the curve. For each point in the plot, the plotting symbol is the first letter of the country and the point label is the country. Each label/symbol pair is a different random color with brightness (average RGB or red-green-blue value) of 128. These options make it much easier to find the symbol that corresponds to each label. Also, the default `vtoh=2` was used to decrease the number of cells and make the labels larger. With these data, the penalty sum at iteration four is eight. Specifying `maxiter=4` prevents the algorithm from reaching iteration 5, which prevents the line size from increasing from 125 to 150. This also makes the labels larger. The price is that some label characters collide (for example, "Germany" and "S") and the plot looks more cluttered because there are fewer cells with white space.

Availability

If your site has installed the latest autocall libraries supplied by SAS and uses the standard configuration of SAS supplied software, you need only to ensure that the SAS system option `mautosource` is in effect to begin using autocall macros. That is, the macros do *not* have to be included (for example with a `%include` statement). They can be called directly. For more information about autocall libraries, see

SAS Macro Language: Reference and your host documentation. Also see pages 803–805, in the macros chapter for more information.

If you do not have the latest autocall macros installed, you can get them from the Web `http://support.sas.com/resources/papers/tnote/tnote_marketresearch.html`.

Base SAS and SAS/GRAPH software are required to run the `%PlotIt` macro. The `datatype=curve` and `datatype=curve2` options use PROC TRANSREG, which is in SAS/STAT. All of the other `datatype=` values assume an input data set in a form created by a SAS/STAT procedure.

Conclusions

The `%PlotIt` macro provides a convenient way to display the results from many types of data analyses. Usually, only a small number of options are needed; the macro does the rest. The `%PlotIt` macro does not replace procedures like GPLOT and PLOT. Instead, it makes it easy to generate many types of plots that are extremely difficult to produce with standard procedures.

Appendix: ODS Graphics

With SAS Version 9.2, ODS Graphics automatically does much of what the `%PlotIt` macro was originally designed to do, and usually, ODS Graphics does it better and more conveniently. This appendix shows how to generate some of the same plots as were shown in the main body of this paper, and a few additional plots as well. All plots in this appendix are generated with ODS Graphics. Most of these plots come out of a procedure by default. Some must be requested with a `plots=` option (e.g. `plots=score` in PROC PRINCOMP). Others are directly produced by the SG (Statistical Graphics) plotting procedures SGRENDER and SGPLOT. You can use PROC SGRENDER and PROC SGPLOT on both raw data sets or output data sets produced by other procedures. This chapter also shows the ODS statements and options that are used for capturing plots in files so that they may later be included into a document (such as this one). Usually, of course, books show you the code to make a plot, not the full code needed to display this plot in this place in this document. This technique can be directly applied to other documents composed using LaTeX, and less directly in other documents.

The first statement that you run is as follows:

```
ods listing style=statistical gpath='png';
```

This opens the LISTING destination, which is open by default, specifies that output will be generated with the Statistical style, and specifies that the graphical output will go to a directory called 'png', which must be created as a subdirectory under the current working directory. The style controls the color scheme and the look and feel of the output. There are many styles, and the Statistical style is the one that is used with SAS/STAT documentation. The following statements produce Figure 11:

```
ods graphics / reset=index imagename="obsodspc1";
proc princomp data=teeth plots=score n=2;
   title "Principal Components of Mammals' Teeth";
   ods select scoreplot;
   id mammal;
   run;
```

The ODS Graphics plot in Figure 11 corresponds to the `%PlotIt` macro plot in Figure 1. The ODS Graphics statement specifies that plots are to be generated in the `png` directory, with a name constructed from the `imagename=` value, possibly followed by a numeric suffix if the step produces more than one plot, and the suffix '.png'. The generated graph name, including the path is `png/obsodspc1.png`. The file is included into the LaTeX document with the command `\includegraphics{png/obsodspc1.png}`. Before the first use of `includegraphics`, there is a single line to enable the `includegraphics` command: `\usepackage{graphicx}`. You can bring in the graph slightly more indirectly by defining a command `getgraph` in LaTeX as follows:

```
\newcommand{\getgraph}[1]{%
\begin{center}%
\includegraphics{/u/saswfk/text/book2006/png/#1.png}%
\end{center}}
```

Figure 11 was included into the document as follows:

```
\begin{figure}[t]
\getgraph{obsodspc1}
```

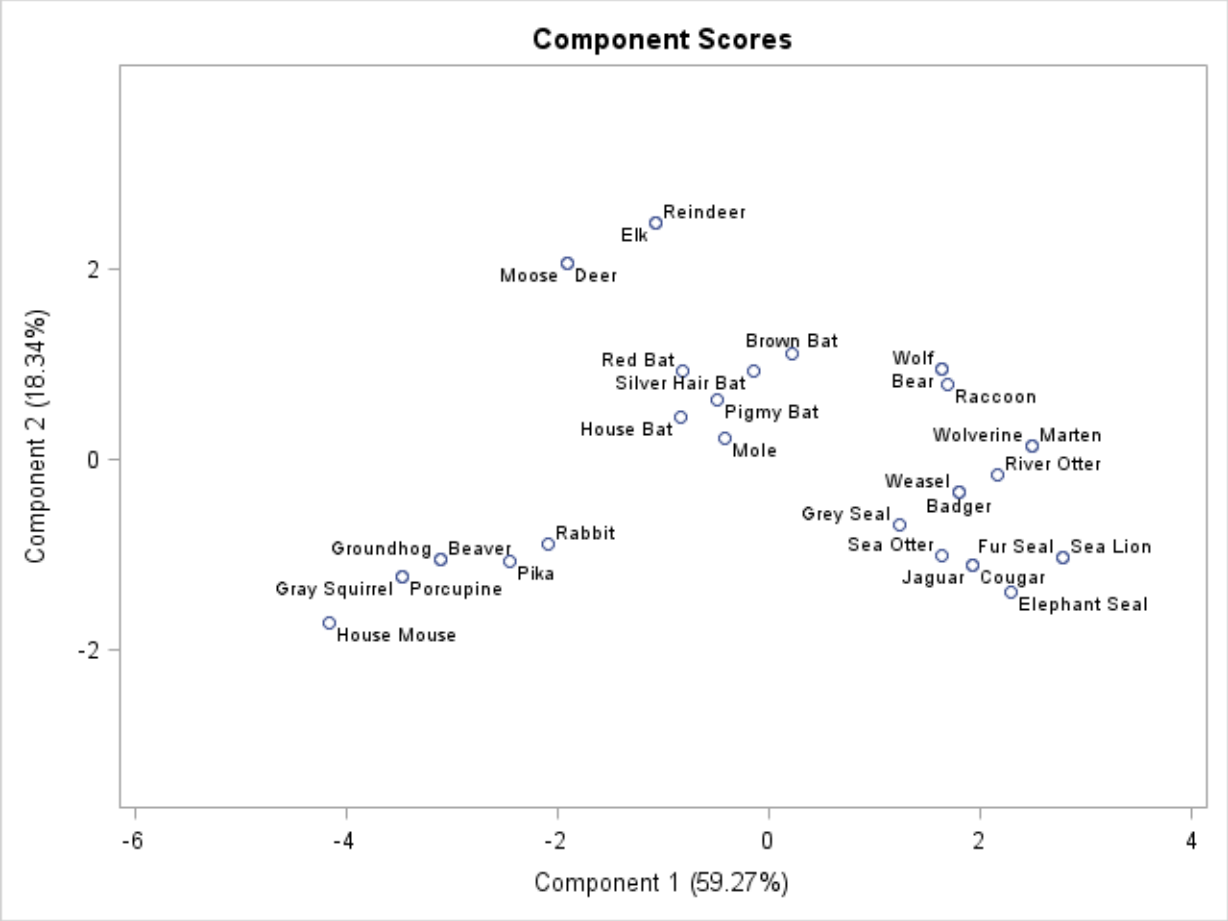

Figure 11. Score Plot from PROC PRINCOMP

```
{\it Figure 11. \hspace{0.001in} Score Plot from PROC PRINCOMP}
\end{figure}
```

Let's return to the SAS statements again, which are as follows:

```
ods graphics / reset=index imagename="obsodspc1";
proc princomp data=teeth plots=score n=2;
   title "Principal Components of Mammals' Teeth";
   id mammal;
   run;
```

PROC PRINCOMP is run to perform the principal component analysis. The nondefault scores plot is requested by the `plots=score` option. The ODS SELECT statement specifies the plot that we want to see. This statement is not necessary, but it does exclude all other output and ensures that an integer need not be appended to the graph name. If the ODS SELECT statement had not been specified, and if the score plot were the second plot, then the plot names would be `obsodspc1.png` and `obsodspc11.png`, then the plot we need is `obsodspc11.png`. If the PROC PRINCOMP step is run again, without the `ods graphics / reset=index` statement, subsequent plots would be named `obsodspc1.png12`, `obsodspc13.png`, and so on. With the `ods graphics / reset=index` statement, the names are the same each time since the index is reset to zero or blank.

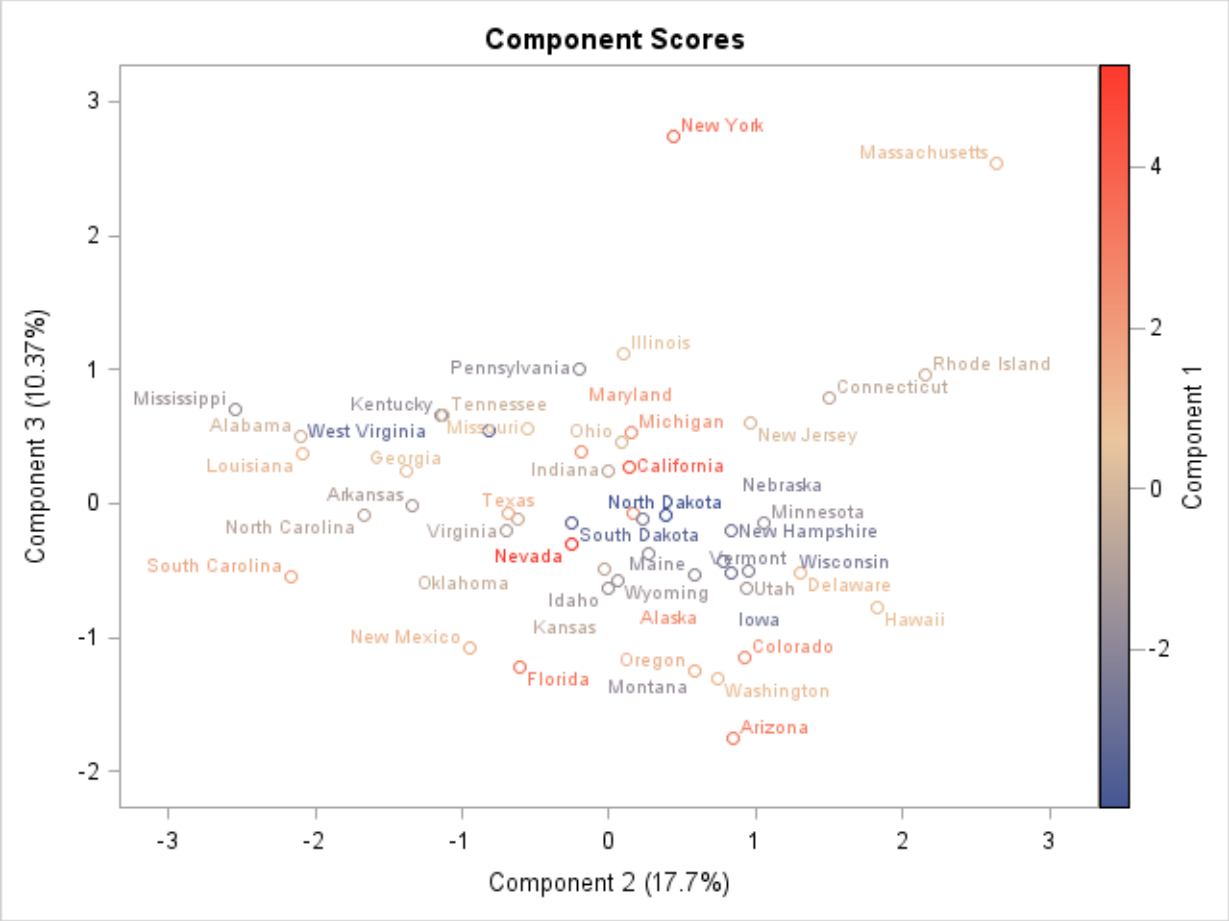

Figure 12. Painted Score Plot from PROC PRINCOMP

The following statements create Figure 12, which is in some ways similar to Figure 2:

```
ods graphics / reset=index imagename="obsodspc2";
proc princomp data=crime out=crime2 plots=score n=3;
   title 'Crime Rates Per 100,000 Population by State';
   ods select paintedscoreplot;
   id state;
   run;
```

Figure 12 consists of component 3 by component 2, colored by (or "painted" by) the values of component 1. In contrast, Figure 2 consists of component 2 by component 1, painted by the values of component 3. We can use ODS Graphics to make a plot like the one shown in Figure 2 with PROC TEMPLATE, the GTL (graphical template language), and PROC SGRENDER. This is a fairly straight-forward graphical template. It consists of a `define statgraph` and `end` block, which provides the name of the template. The template name, in this case `plot`, must be specified with PROC SGRENDER. There is a nested `begingraph`/`endgraph` block containing an `entrytitle` (plot title). Then there is a `layout overlay`/`endlayout` block. This is the outer shell most frequently used in making plots with SGRENDER, although other `layout` statements are possible. Inside the layout is a `scatterplot` statement that names the y axis variable, the x axis variable, the label variable, and the third variable whose values are shown with a gradient of colors. The `continuouslegend` statement produces the

color "thermometer" legend for the statement named `'scores'`. The PROC SGRENDER step names the input data set and the template. The optional LABEL statement provides more descriptive labels for the axis variables. The following statements create Figure 13, which is similar to Figure 2:

```
proc template;
   define statgraph plot;
      begingraph;
         entrytitle 'Crime Rates Per 100,000 Population by State';
         layout overlay;
            scatterplot y=prin2 x=prin1 / datalabel=state
                        markercolorgradient=prin3 name='scores';
            continuouslegend 'scores' / title='Component 3';
            endlayout;
         endgraph;
      end;
   run;

ods graphics / reset=index imagename="obsodspc3";
proc sgrender data=crime2 template=plot;
   label prin1 = 'Component 1' prin2 = 'Component 2';
   run;
```

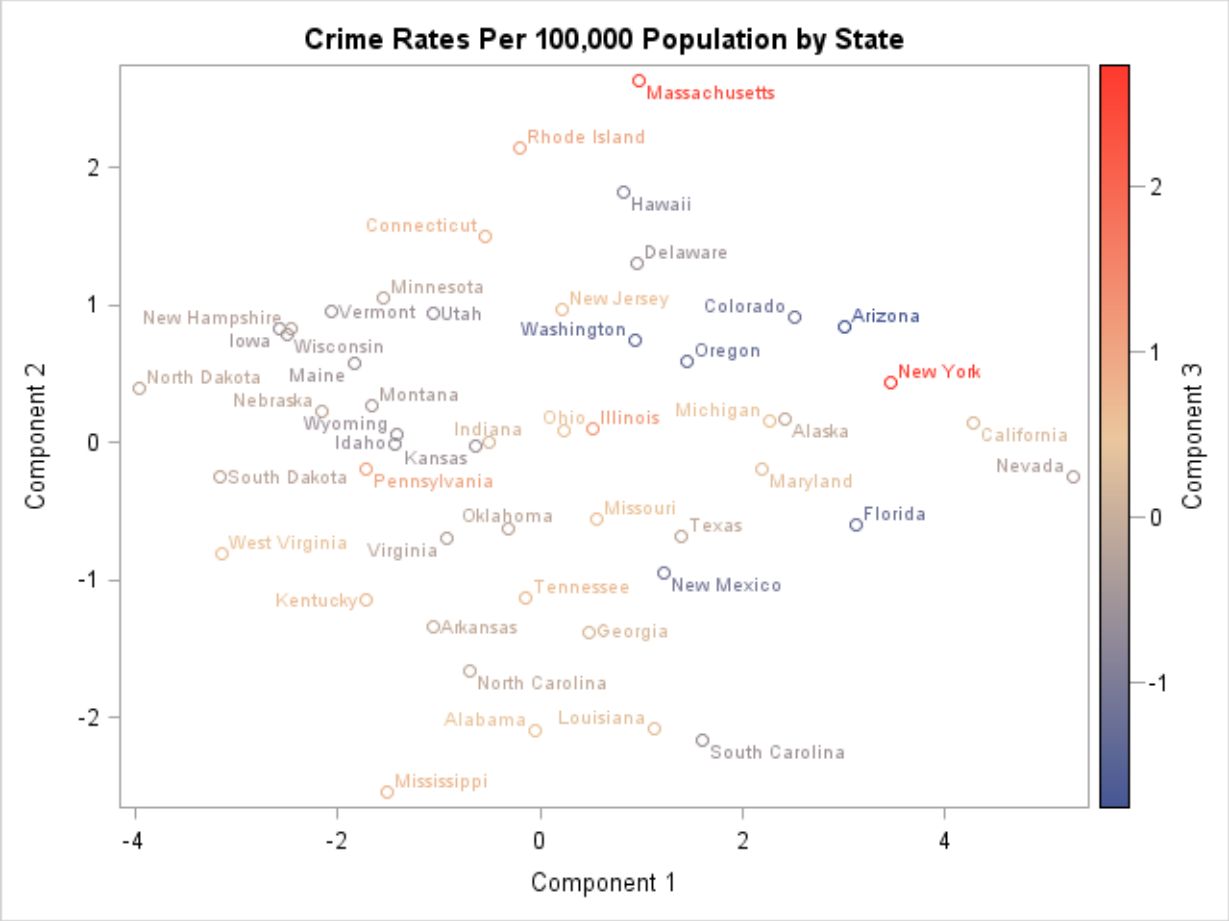

Figure 13. Painted Score Plot from PROCs TEMPLATE and SGRENDER

The following statements create Figure 14, which is similar to Figure 3:

```
ods graphics / reset=index imagename="obsodsca";
proc corresp data=cars outc=coors;
   title 'Car Owners and Car Origin';
   ods select configplot;
   tables marital, origin;
   run;
```

When ODS Graphics is enabled, PROC CORRESP automatically produces the correspondence analysis configuration plot.

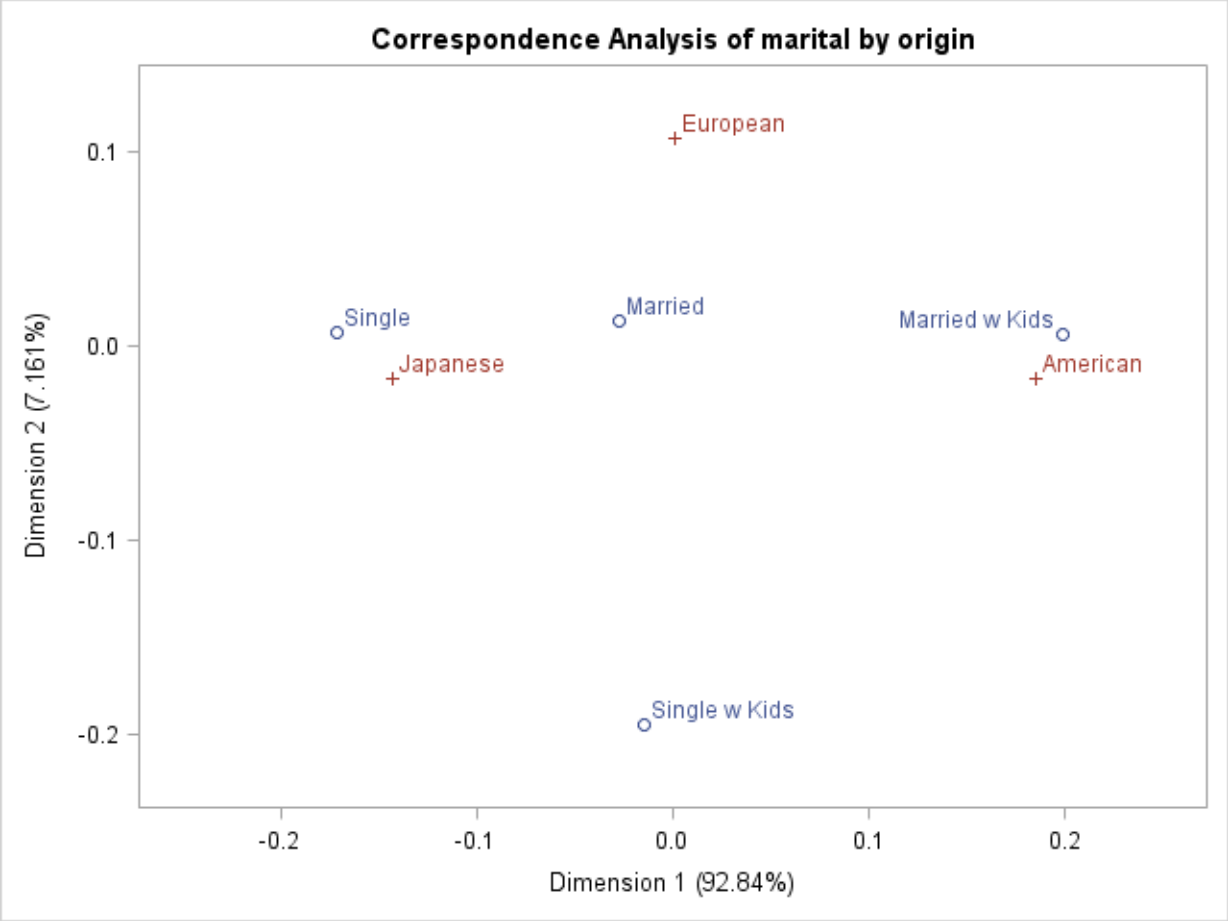

Figure 14. Simple Correspondence Analysis from PROC CORRESP

The following statements create Figure 15, which is similar to Figure 4:

```
ods graphics / reset=index imagename="obsodsmca";
proc corresp mca observed data=cars outc=coors;
   title 'MCA of Car Owners and Car Origin';
   ods select configplot;
   tables origin size type income home marital sex;
   run;
```

When ODS Graphics is enabled, PROC CORRESP automatically produces the multiple correspondence analysis configuration plot.

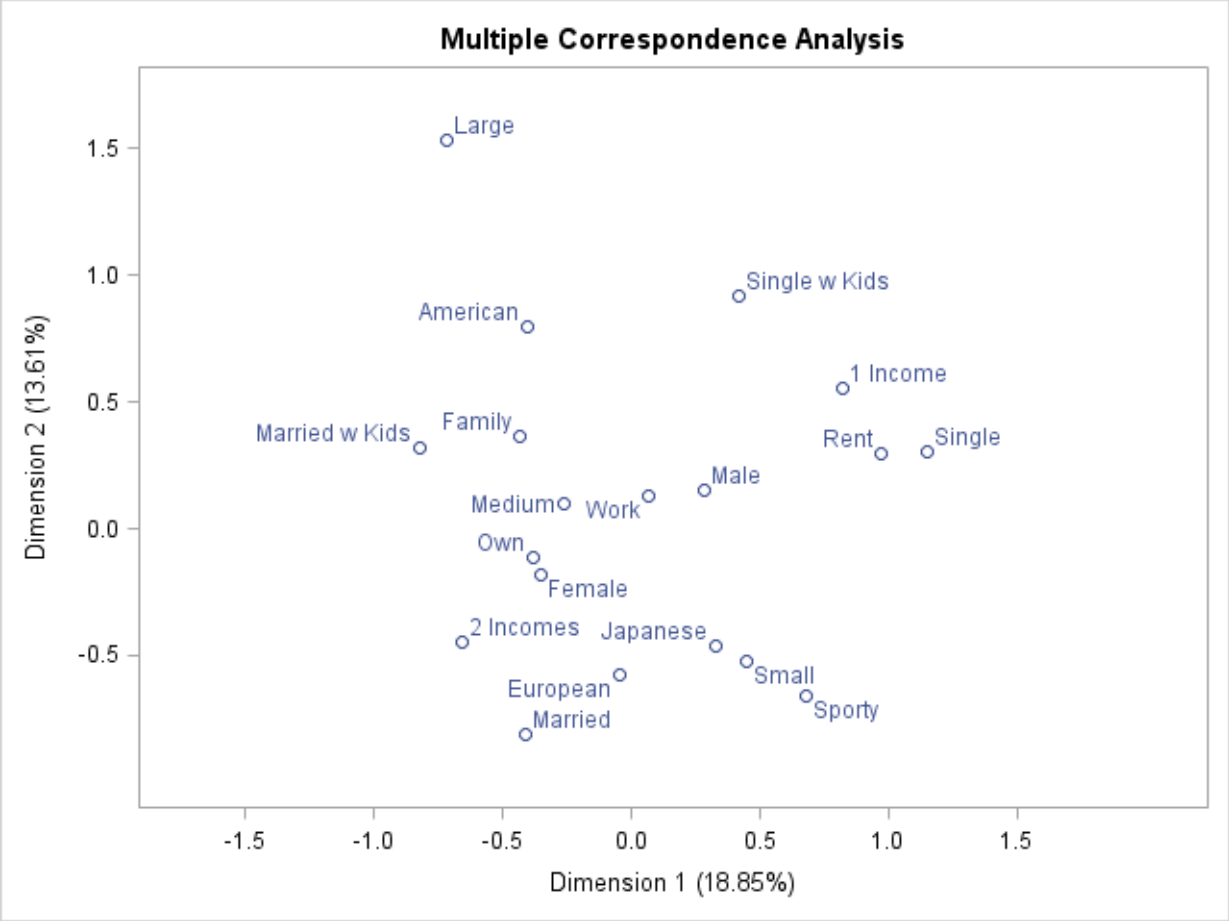

Figure 15. Multiple Correspondence Analysis from PROC CORRESP

The following statements create Figure 16, which is similar to Figure 6:

```
%macro ren(p,n); rename=(%do i = 1 %to &n; &p&i = "&i"n %end;) %mend;

options validvarname=any;
data rec2(%ren(sub,56)); set recreate; run;

ods graphics / reset=index imagename="obsodsmdp";
proc prinqual data=rec2 mdpref;
   title1 'Multidimensional Preference Analysis of Recreational Activities';
   ods select mdprefplot;
   transform identity('1'n-'56'n);
   id activity;
   run;
```

When ODS Graphics is enabled, PROC PRINQUAL and the `mdpref` option automatically produce the MDPREF plot. The variable names are displayed in the plot as labels for the vectors. In an MDPREF analysis, each column in the data set actually corresponds to a subject not a variable. This example uses a macro to rename the input variables from `sub1`, `sub2`, and so on to `1`, `2`, and so on to make a better graphical display—one that is not cluttered up by the prefix `sub` appearing 56 times.

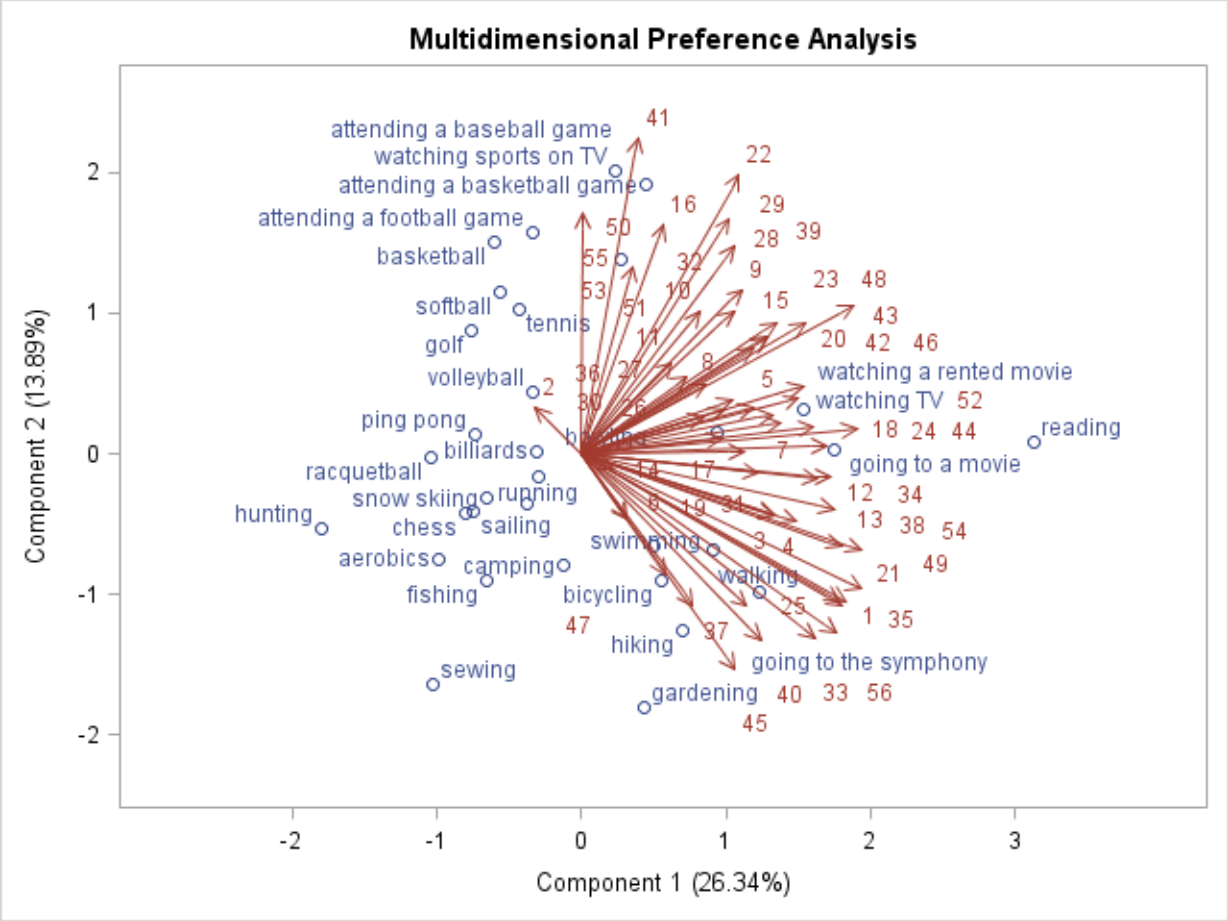

Figure 16. MDPREF Analysis from PROC PRINQUAL

The following statements create Figure 17:

```
data carpref2(%ren(judge,25)); set carpref; run;

ods graphics / reset=index imagename="obsodsmdp2";
proc prinqual data=carpref2 n=2 out=presults(drop='1'n-'25'n) replace mdpref;
   title 'Preference Ratings for Automobiles Manufactured in 1980';
   ods select mdprefplot;
   id model mpg reliable ride;
   transform identity('1'n-'25'n);
   run;

options validvarname=v7;
```

Like the previous example, the variables are renamed, and PROC PRINQUAL and the `mdpref` option automatically produce the MDPREF plot when ODS Graphics is enabled.

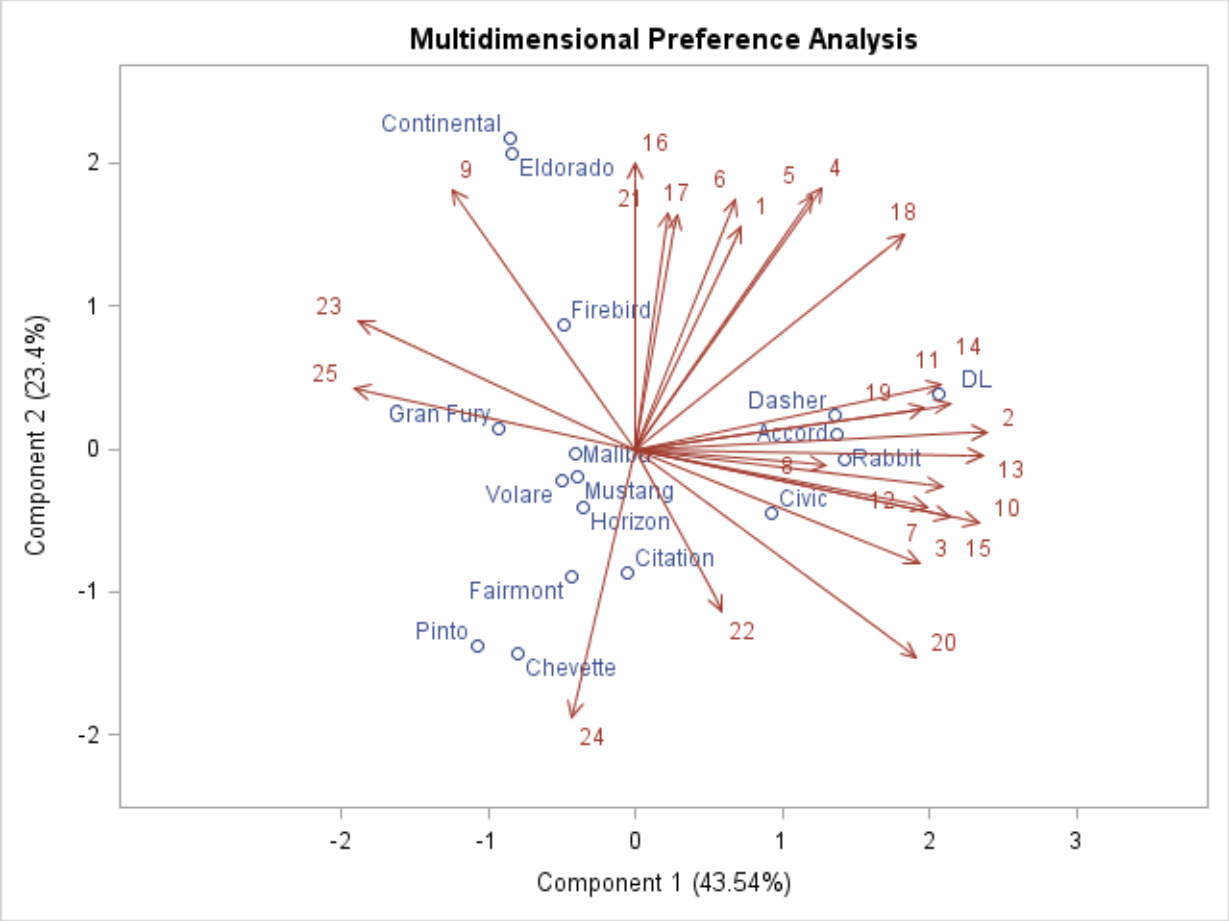

Figure 17. MDPREF Analysis from PROC PRINQUAL

The following statements create Figure 18, which is similar to Figure 7:

```
ods graphics / reset=index imagename="obsodsvec";
proc transreg data=presults tstandard=center coordinates;
   title2 'Preference Mapping, Vector Model';
   ods select prefmapvecplot;
   model identity(mpg reliable ride)=identity(prin1 prin2);
   id model;
   run;
```

When ODS Graphics is enabled, PROC TRANSREG and the `coordinates` option automatically produce a PREFMAP plot. Since the independent variable are designated as `identity` variables, the PREFMAP plot is a vector plot.

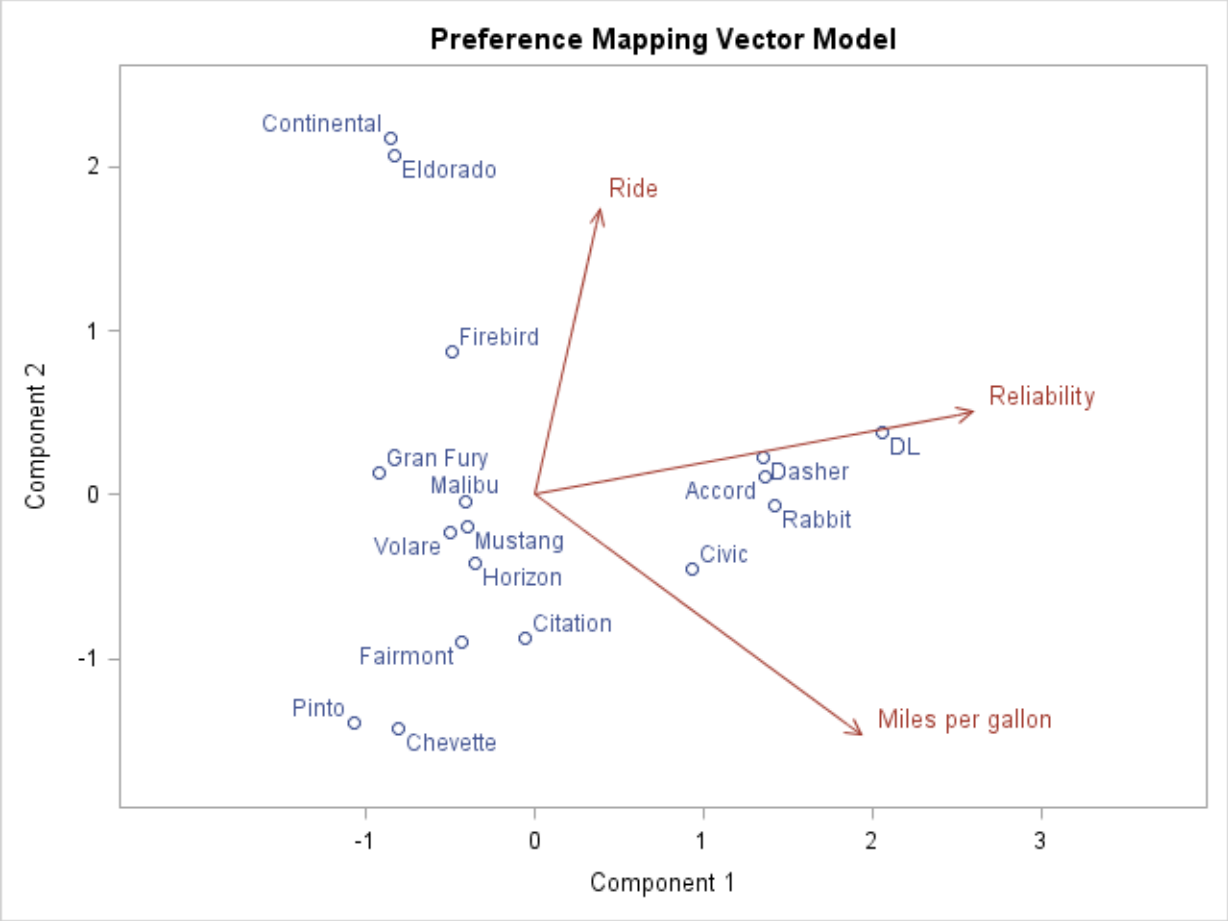

Figure 18. PREFMAP Vector Model from PROC TRANSREG

The following statements create Figure 19, which is similar to Figure 8:

```
ods graphics / reset=index imagename="obsodsidp";
proc transreg data=presults tstandard=center coordinates ;
   title2 'Preference Mapping, Ideal Point Model';
   ods select prefmapidealplot;
   model identity(mpg reliable ride)=point(prin1 prin2);
   id model;
   run;
```

When ODS Graphics is enabled, PROC TRANSREG and the `coordinates` option automatically produce a PREFMAP plot. Since the independent variable are designated as `point` variables, the PREFMAP plot is an ideal-point plot. ODS Graphics does not automatically draw the circles like the `%PlotIt` macro does.

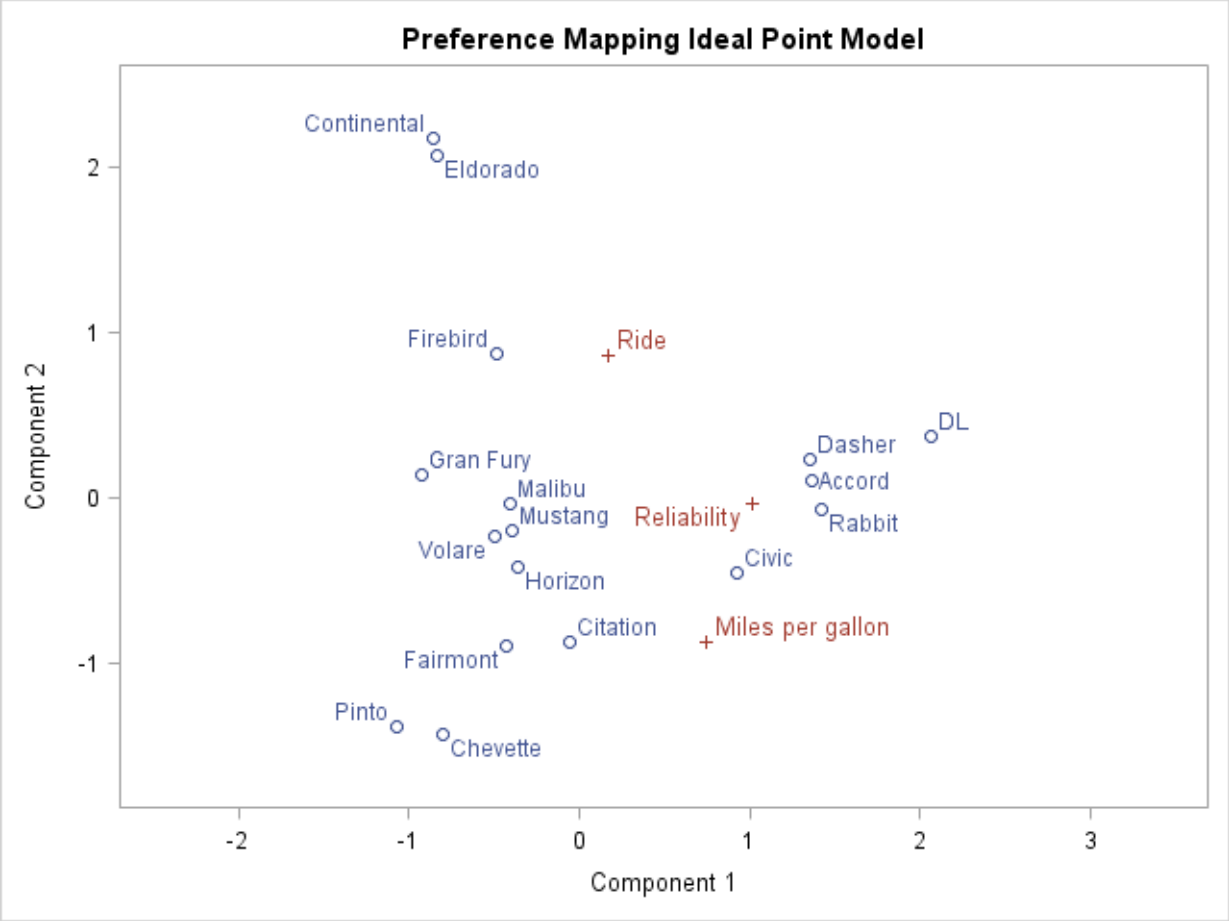

Figure 19. PREFMAP Ideal Point Model from PROC TRANSREG

The following statements create Figure 20, which is similar to Figures 9 and 10:

```
ods graphics / reset=index imagename="obsodsreg";
proc sgplot data=vital;
   title 'Crude Death Rate as a Function of Crude Birth Rate';
   pbspline y=death x=birth / datalabel=country nolegfit;
   run;
```

You can use PROC SGPLOT to directly create many types of scatter plots without having to make a template and use PROC SGRENDER.* This example uses the `pbspline` statement to plot a scatter plot of points and fit a nonlinear regression function using penalized B-splines. This statement automatically finds a smooth function based on an automatically-chosen smoothing parameter.

With PROC SGPLOT, the `title` statement provides that title that actually appears in the `png` file with the plot. In PROC SGRENDER, the `entrytitle` statement provides that title that actually appears in the `png` file with the plot. In regular, non-SG procedures, the title appears in the output but outside the plot, not as part of the plot. The plot titles are determined by the plot template.

*PROC TEMPLATE and PROC SGRENDER were used in a previous example because PROC SGPLOT does not currently support the continuous legend. Occasionally, you will need other capabilities not in PROC SGPLOT, and then you will have to use PROC TEMPLATE and PROC SGRENDER. For example, when you want to equate the axes in a plot, you will have to use PROC TEMPLATE and PROC SGRENDER.

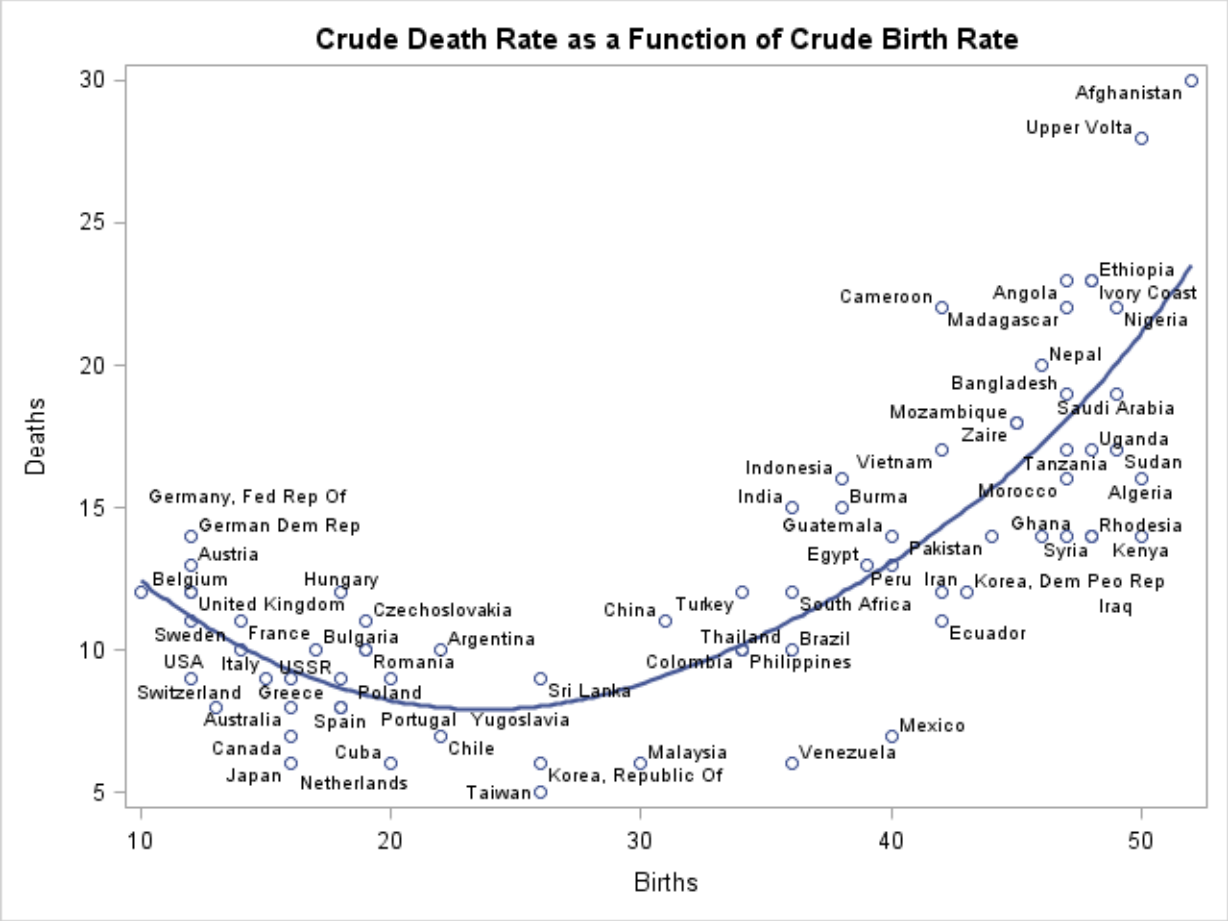

Figure 20. Spline Regression Fit from PROC SGPLOT

The following statements create Figure 21:

```
ods graphics / reset=index imagename="obsodstreg";
proc transreg data=vital;
   title 'Crude Death Rate as a Function of Crude Birth Rate';
   ods select fitplot;
   id country;
   model identity(death) = mspline(birth / nknots=9);
   run;
```

When ODS Graphics is enabled, PROC TRANSREG automatically produces a fit plot when it is appropriate. When a simple regression model is being fit, optionally with an independent variable transformation and up to one classification variable, a fit plot is automatically produced. In this example, a nonlinear but monotone (always goes up or stays flat, or always goes down or stays flat, but never goes both up and down) fit function is used. A fit plot with separate functions for each level of the classification variable could be produced by using a model like the following with PROC TRANSREG and ODS Graphics enabled:

```
model identity(death) = class(continent / zero=none) |
                        spline(birth / nknots=9);
```

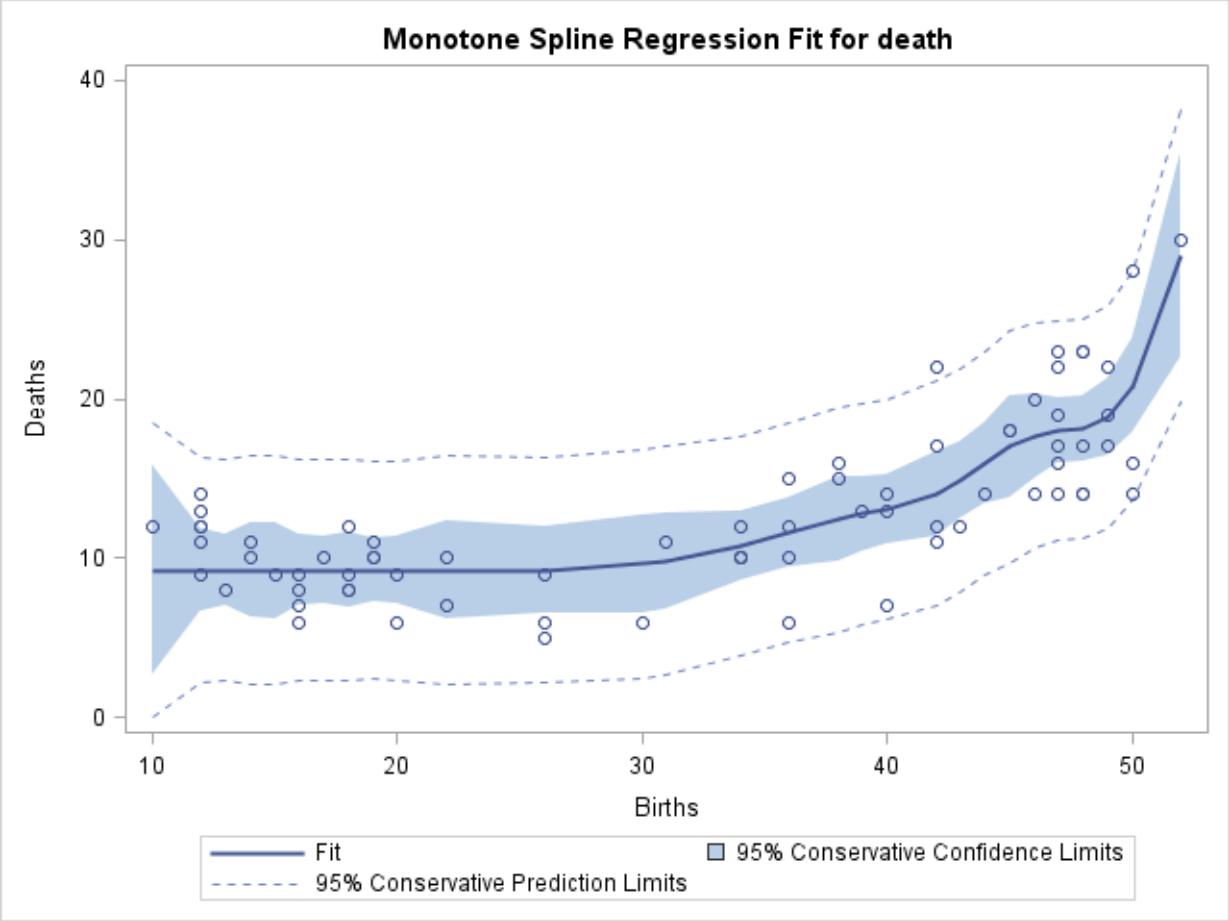

Figure 21. Monotone Spline Regression Fit from PROC TRANSREG

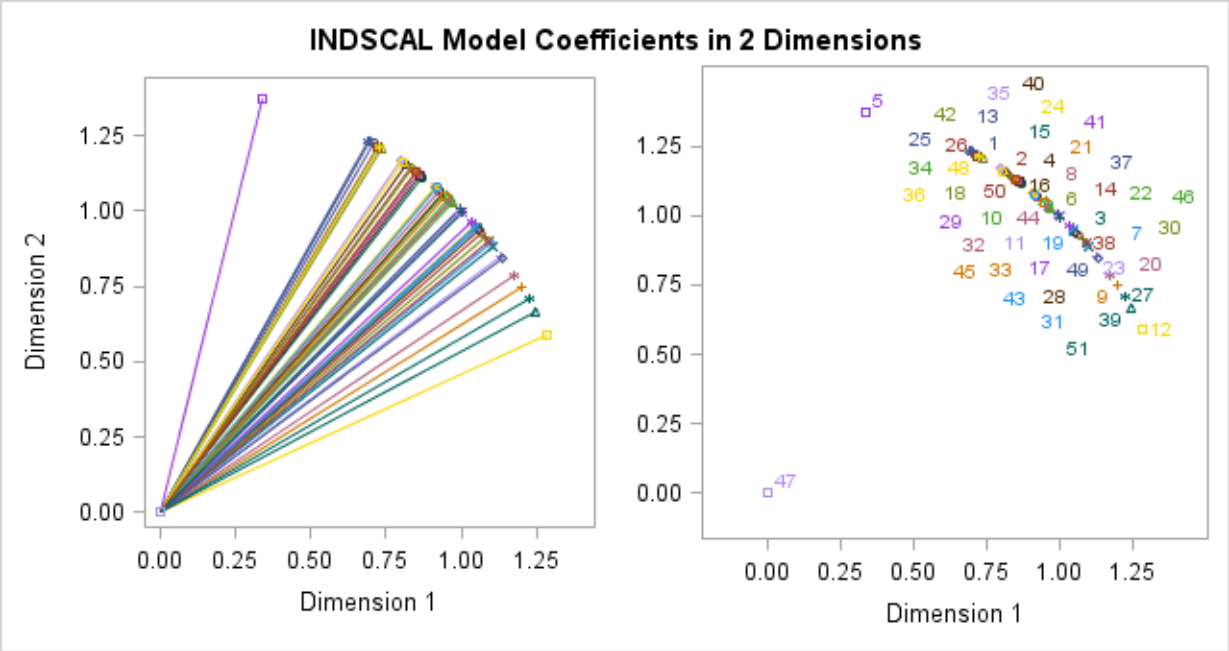

Figure 23. Coefficient Plot from PROC MDS

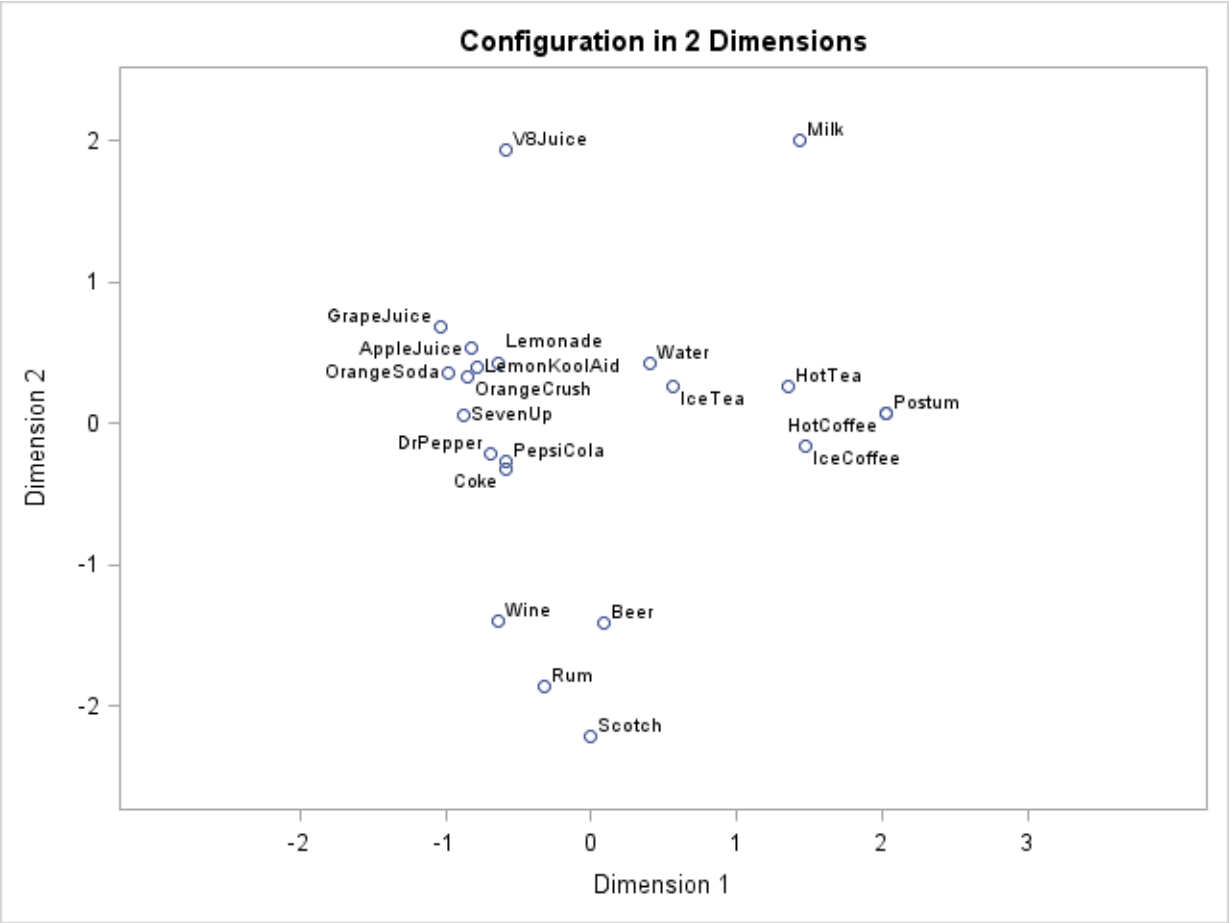

Figure 22. Configuration Plot from PROC MDS

The following statements create Figures 22, 23, and 24:

```
ods graphics / reset=index imagename="obsodsmds";
proc mds data=dissim level=interval model=indscal dimens=2 header;
   ods select configplot coefficientsplot fitplot;
   title 'Multidimensional Scaling of Beverages';
   run;

ods graphics off;
```

When ODS Graphics is enabled, PROC MDS automatically produces plots of the results of the MDS analysis. Figure 22 shows the configuration of points. Figure 23 shows the coefficients vectors for the INDSCAL model. The plot on the left of the panel shows the appropriate geometry with the coefficients displayed as vectors. The plot on the right, provides in some sense, a legend for the plot on the left. Just the vector end points are shown and they are labeled with subject ID information, which in this example is just the subject number. Figure 24 shows the fit of the data to the two-dimensional solution.

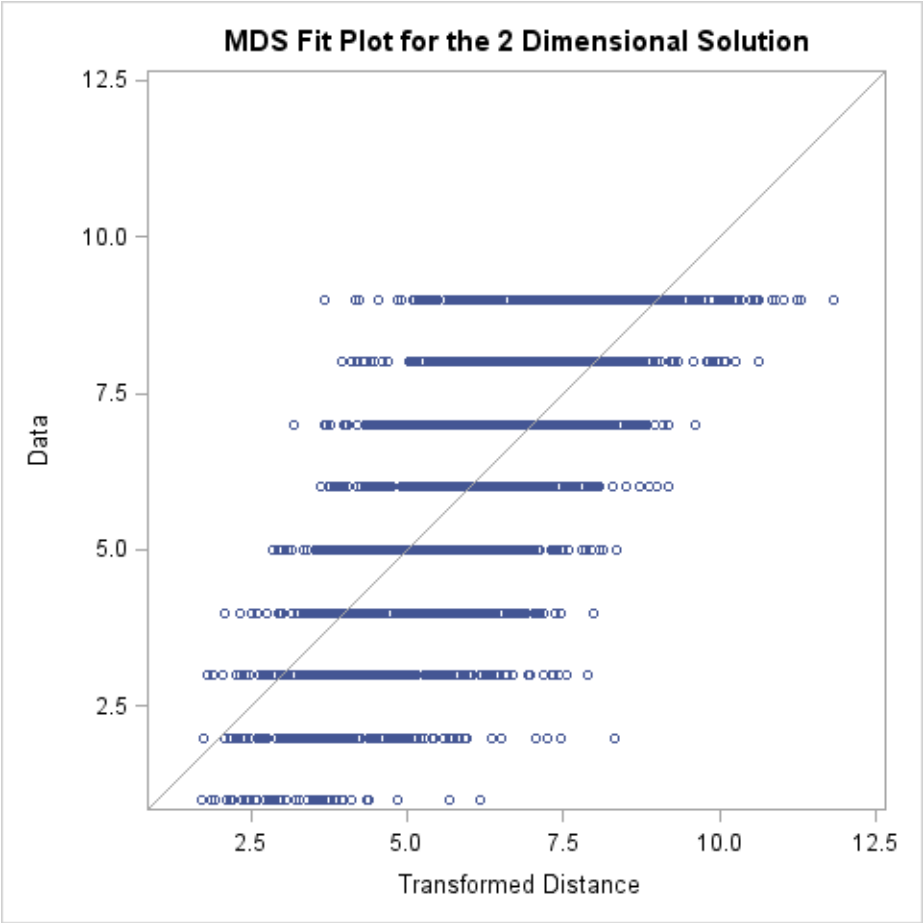

Figure 24. Fit Plot from PROC MDS

Since this step produces three plots, the file names are: `png/obsodsmds.png`, `png/obsodsmds1.png`, and `png/obsodsmds2.png`.