Paper 3382-2015

Put Your Data on the Map: Using PROC GEOCODE and PROC GMAP to Create Bubble Maps in SAS®

Caroline Cutting, Warren Rogers Associates

ABSTRACT

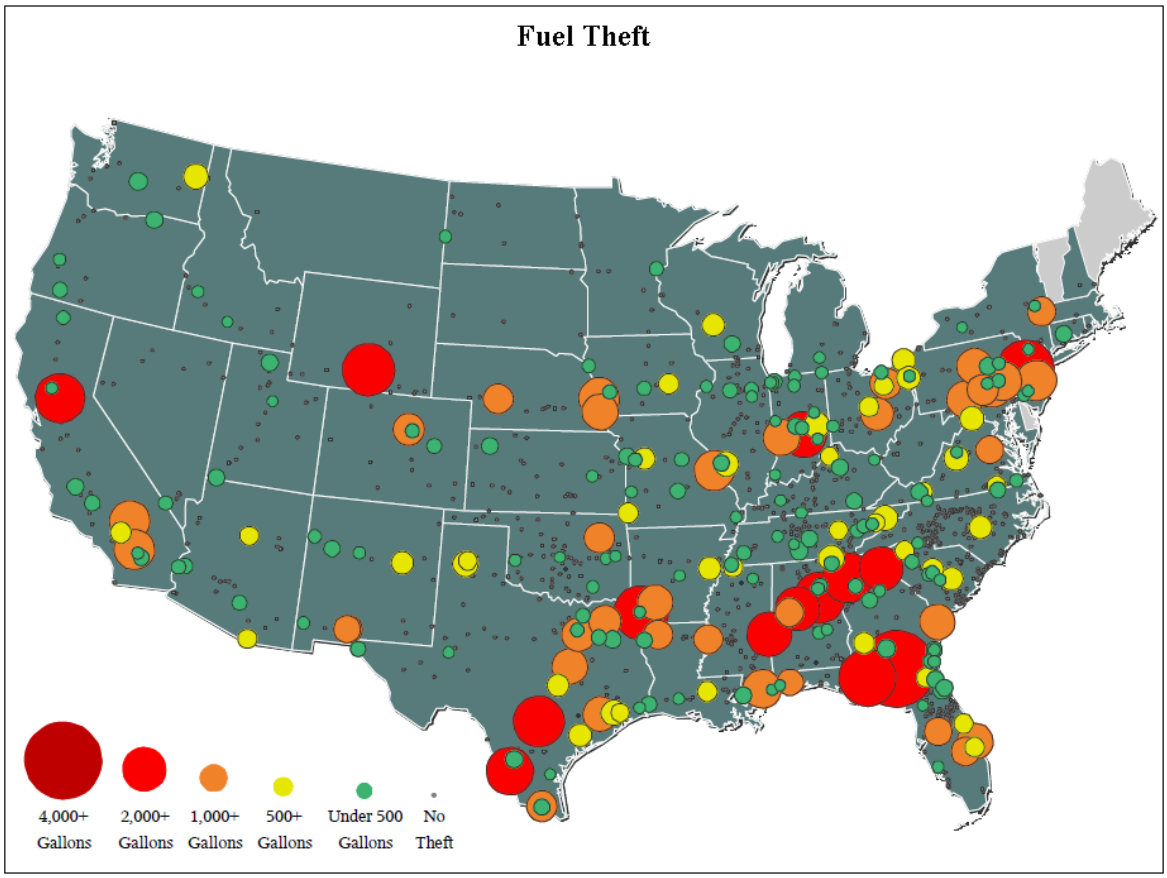
A bubble map is a useful tool for identifying trends and visualizing the geographic proximity and intensity of events. This paper illustrates how to use PROC GEOCODE and PROC GMAP to turn a dataset of addresses and events into a bubble map of the United States with scaled bubbles depicting the location and intensity of events.

INTRODUCTION

A bubble map, also called a cartogram, is a specialized bubble chart useful for displaying three-dimensional data in which the first two dimensions of the data represent the geographic location of an event, and the third dimension is any other quantitative variable of interest. The example illustrated in this paper concerns gasoline theft data. In this context, a geographic bubble map of the theft events is useful for highlighting theft "hotspots" and providing insights into emerging theft trends. When implemented on a large data set, say all gasoline thefts during a time period of interest, an example bubble map might look something like the example in Figure 1.

This paper will describe how to create a similar bubble map using a smaller sample data set.

Figure 1. A bubble map of fuel theft locations, created using PROC GMAP.



PART 1: CREATING THE BACKGROUND MAP

THE BASIC MAP

The first step in creating the bubble map is to generate a "background" map, this is the map of the United States onto which the theft "bubbles" will later be plotted. The MAPS library in SAS® provides the data sets (in map data set format) needed to create maps for a variety of geographic regions. You can use the data set MAPS.STATES to create a map of the contiguous 48 U.S. states. A couple of important things to know about this data set before beginning:

- The variables X and Y in the data set represent longitude and latitude coordinates, respectively, for the points in the map. The coordinates are given in radians, not degrees, and they represent un-projected spherical coordinates. We will need to convert these into Cartesian coordinates in a later step, so that they can be plotted in two-dimensions with minimal distortions. Additionally, the longitude coordinates given by X increase from east to west.
- Each latitude and longitude coordinate in the data set has a corresponding DENSITY value between zero and four. You can control the level of detail in the map you create by filtering out points with higher density values.
- The variable STATE gives the two-digit numeric code identifying the state associated with each set of latitude and longitude coordinates. These codes are the Federal Information Processing Standard (FIPS) state codes, and can be converted to more familiar two letter state abbreviations using the SAS function FIPSTATE().
- The variable SEGMENT groups coordinates that should be plotted together as contiguous polygons when the map is being drawn.

If you wish to create a map showing only the 48 contiguous U.S. states, with the same level of detail as in Figure 1, then filter out all geographical coordinates associated with Alaska, Hawaii, and Puerto Rico, as well as any points with density values greater than 2 when reading in the MAPS.STATES data set:

```
data MyMap;
set maps.states ;
if fipstate(state) not in ('AK' 'HI' 'PR');
if density <= 2;
```

To take a look at the plot of the data set you've just created, you can use the GMAP procedure, but you'll need to make one adjustment first. Add in a dummy response variable (for example, DUMMY = 1) so PROC GMAP has something to plot in the choropleth map it will construct. Setting all values of DUMMY to the same value means all states will be filled with the same color in the resulting map. Then use PROC GMAP to plot the map:

```
data MyMap;
set MyMap;
dummy = 1;

proc gmap data = MyMap map = MyMap;
id state;
choro dummy / statistic = first nolegend;
```

Unfortunately, the resulting map, shown in Figure 2, leaves something to be desired. It is flipped from east-to-west and some geographical areas appear distorted, a result of plotting the longitude/latitude coordinates in a two-dimensional plane.

Figure 2. A plot of data from maps.states, using unprojected coordinates.

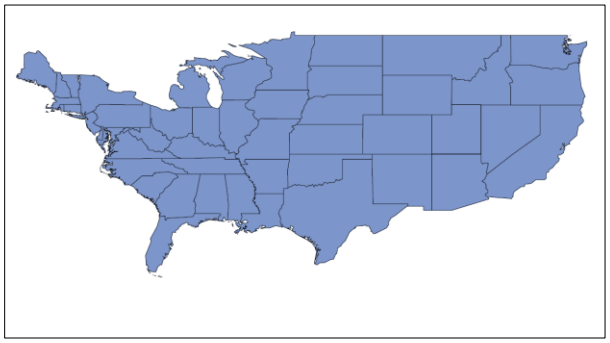
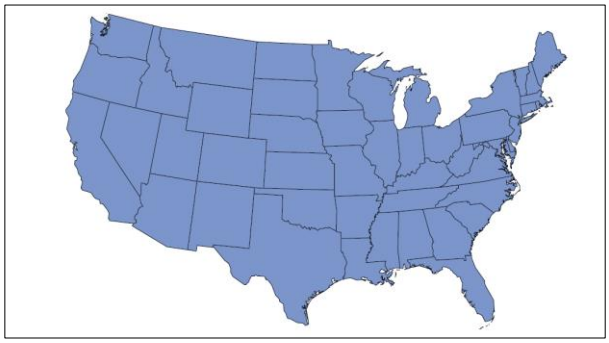


Figure 3. A plot of data from maps.states, after projecting the coordinates with PROC GPROJECT.



The reverse orientation in Figure 2 occurred because the longitude coordinates in the MYMAP data set are increasing from east to west. You can correct this by using the GPROJECT procedure with the WESTLONG option. The GPROJECT procedure will also correct the distortion of geographical areas by applying an Albers projection to the X and Y coordinates. This is the default projection applied unless a different option is specified, for clarity it is shown explicitly in the code below:

```
proc gproject data = mymap out = mymap2 westlong project = albers;
id;

proc gmap data = mymap2 map = mymap2;
id state;
choro dummy / statistic = first nolegend;
```

Running these statements generates the map shown in Figure 3.

GETTING FANCY: COLORS AND SHADING

The background map is looking good, but why not make it a little better? For starters, you can change the fill color used for the states. This is done by including a PATTERN statement prior to the GMAP procedure:

```
pattern1 color = gray;
proc gmap data = mymap2 map = mymap2;
id state;
choro dummy / statistic = first nolegend;
```

There are over 16 million colors available in SAS, if you're short on ideas, check out the SAS Knowledge Base paper "TS-688 Defining Colors using Hex Values" (web address given in references) for inspiration. Next you might consider changing the color of the state outlines. You can do this by adding a COUTLINE argument when calling the GMAP procedure:

```
proc gmap data = mymap2 map = mymap2;
id state;
choro dummy / statistic = first nolegend coutline = graycc;
```

Finally, you can add a second, shifted, outline around the outside perimeter of the map to create a slight 3D effect. This is a technique described by Mike Zdeb and Robert Allison in their paper "Stretching the Bounds of SAS/Graph® Software" from SUGI-30. It takes a little more code to do this. You'll need to create an annotation data set with instructions for outlining the boundary lines of the map in a different color (gray, in this example).

First, take the data set you've been using for the background map, MYMAP2, and shift the X and Y coordinates of the points just slightly:

```
data mapshift;
set mymap2;
x = x + 0.001;
y = y - 0.001;
```

Next add information about how PROC GMAP should use these shifted coordinates. For each set of coordinates, specify XSYS = '2' and YSYS = '2' indicating that the X and Y variables in this annotation data set should be treated as part of the same coordinate system (same units) as the data values in the MYMAP2 data set. Also specify WHEN = 'b', this will cause these shifted perimeter values to be drawn first, *before* the actual map. Many of these shifted points will fall "underneath" actual map data points, and so won't be seen after the final map is drawn, creating a shadow effect.
Finally assign FUNCTION = 'poly' to the first segment of each state, to specify the beginning point of a polygon. For all other segments in the state assign FUNCTION = 'polycont', instructing the annotation to continue drawing the polygon that was begun in the first segment for that state.

```
data perim;
set mapshift;
length function $10;
by state segment notsorted;
xsys='2'; ysys='2'; when='b';
style='solid'; color='gray33';
if first.segment then function='poly';
else function='polycont';
```

You can now use this data set as an annotation data set in the CHORO statement of PROC GMAP:

```
pattern1 color = gray;
proc gmap data = mymap2 map = mymap2;
id state;
choro dummy / statistic = first nolegend coutline = grayCC anno = perim;
```

This produces the map shown in Figure 4.

Figure 4. Background map after formatting.

PART 2: ANNOTATE THE MAP WITH BUBBLES

Now that the background map is ready, you can add bubbles showing where the thefts occurred. This is done by creating another annotation data set giving the theft locations, to be plotted over the background map.

Most data sets typically do not arrive on your desk with latitude and longitude coordinates included. More likely, you might begin with something like the example THEFTS data set below, listing the zip code in which each theft occurred, and the amount of that theft, in gallons.

```
   data thefts;
   input zip amount;
   datalines;
      89109 278
      80110 861
      67212 450
      34745 85
      75202 1471
;
```

The GEOCODE procedure can be used to obtain the longitude and latitude coordinates associated with each of these zip codes:

```
   proc geocode data = thefts out = thefts2;
```

The output data set, THEFTS2, will contain variables X and Y, giving the respective longitude and latitude coordinates associated with the geographical center of each zip code. Note however, that the longitude coordinates in this data set increase from west to east, you will need to switch them to match the increasing east to west convention used in the map data set MYMAP2. This is done by multiplying each X value by -1. Also note that the X and Y coordinates in THEFTS2 are given in degrees. You will need to convert these to radians for compatibility with the MYMAP2 data set. Recalling that 360 degrees = 2π radians, you can make these conversions using the SAS code:

```
   data thefts3;
   set thefts2;
   x=constant('PI')/180 * (-1)*x;
   y=constant('PI')/180 * y;
```

Before you can add these theft locations to the map, you need to make sure they are projected with the same projection that was applied to the original MYMAP data set in Part 1. To do this combine both data sets and then apply the projection. When combining, add an indicator variable distinguishing the observations that came from the THEFTS3 data set, so that they can easily be separated back out again:

```
   data both;
   set mymap thefts3 (in = A);
   theftdata = A;

   proc gproject data = both out = bothProj westlong project = albers;
   id;

   data theftsProj mapProj;
   set bothProj;
   if theftdata = 1 then output theftsproj;
   else output mapProj;
```

You now have two data sets: MAPPROJ which contains the same background map you created in Part 1, and THEFTPROJ which contains theft information in location coordinates that are compatible with the

background map. Now you need to add some annotation directions to the THEFTPROJ data set, indicating what should be plotted on the map at each of those theft locations.

To start, specify XSYS = '2' and YSYS = '2' for each observation, just as was done in the annotation data set in Part 1. Also specify HSYS = '1' indicating that any size coordinates associated with the annotations for these observations should be interpreted as an overall percent of the area spanned by the data. You would like the bubbles created at each of these locations to appear on top of the background map, so specify WHEN = 'a' indicating that these annotations should be applied *after* the background map is drawn. To draw circles at each location specify FUNCTION = 'pie', ROTATE = '360', and STYLE = 'psolid'. To start with, you can make uniform red circles at each location, with a radius equal to 1% of the total area spanned by the data (SIZE = 1, COLOR = 'red'):

```
data theftsProj1;
set theftsProj;
xsys='2';
ysys='2';
hsys='1';
when = 'A';
function = 'pie';
rotate = 360;
size = 1;
style='psolid';
color = 'red';
```

To take a look at the bubbles you just made, add THEFTSPROJ1 as an annotation data set in the PROC GMAP statement:

```
pattern1 color =gray;
proc gmap data = mapProj map = mapProj anno = theftsproj1;
id state;
choro dummy / statistic = first nolegend coutline = grayCC anno = perim;
```

This creates the map shown in Figure 5.

Figure 5. Map with annotated dots indicating theft locations.

This bubble map will be more useful if the size and color of each bubbles relates to the volume of the associated theft. Modify the annotation data set by adding IF statements specifying which colors to use for thefts of varying magnitudes. Also make the SIZE variable a calculated value based on the theft AMOUNT. When doing this, it is good practice to make SIZE proportional to the *square root* of AMOUNT, so that the area of the plotted bubble is proportional to the actual theft amount:

```
data theftsProj2;
set theftsProj;
format color $10.;
xsys='2'; ysys='2'; hsys='1'; when = 'A';
function = 'pie';
rotate = 360;
style='psolid';
size = 0.115*sqrt(amount);
if amount le 500 then color = 'cxe6e600';
else if amount le 1000 then color = 'cxf08229';
else if amount le 2000 then color = 'cxfa0000';
else color = 'cxbe0000';
```

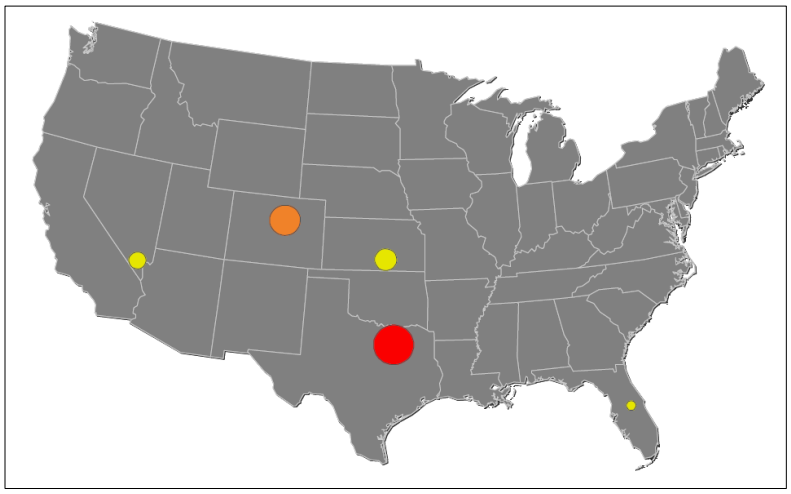
Finally, you can make the map a little more polished by adding solid outlines around each of the bubbles. This is done by replicating each observation in the annotation data set, then modifying those replicate observation so that the second circle drawn on top of the original circle uses a darker color (COLOR = 'gray50') and is drawn only in outline, with no fill (STYLE = 'pempty'):

```
data theftsProj3;
set theftsProj;
format color $10.;
xsys='2'; ysys='2'; hsys='1'; when = 'A';
function = 'pie';
rotate = 360;
style='psolid';
size = 0.115*sqrt(amount);
if amount le 500 then color = 'cxe6e600';
else if amount le 1000 then color = 'cxf08229';
else if amount le 2000 then color = 'cxfa0000';
else color = 'cxbe0000'; output;
style = 'pempty'; color = 'gray50'; output;

pattern1 color =gray;
proc gmap data = mapProj map = mapProj anno = theftsproj3;
id state;
choro dummy / statistic = first nolegend coutline = grayCC anno = perim;
```

This gives the map in Figure 6.

Figure 6. Map with bubbles scaled in proportion to theft volume.



PART 3: ANNOTATING THE LEGEND

No good statistical graphic is complete without a legend. To add a legend to this bubble plot, create an annotation data set with the instructions for drawing the legend. Specify XSYS = '3' and YSYS = '3' indicating that the X and Y coordinates for these observations correspond to percentages of the area spanned by the data (so an observation with coordinates x = 50, y = 50 would be plotted in the center of the graph). Assign HSYS = '1' just as when the bubbles were plotted, so that the radii of the legend bubbles will be plotted using the same scale as the bubbles on the map. Then add one observation to the data set for each bubble you'd like to display in the legend. It will likely take some experimenting to get the X and Y coordinates exactly where you'd like.

```
data legend;
format color function $10. text $50.;
xsys='3'; ysys='3'; hsys='1'; when='a';
function='pie';  rotate=360; ; style='psolid';
y =14; x=7; color='cxbe0000'; size = 0.115*sqrt(2000); style='psolid';
output;
y = 13; x=16; color= 'cxfa0000'; size = 0.115*sqrt(1000);style='psolid';
output;
y = 12; x=24; color='cxf08229'; size = 0.115*sqrt(500); style='psolid';
output;
y = 11; x=31; color='cxe6e600'; size = 0.115*sqrt(400); style='psolid';
output;
```

Next add a second set of observations to the LEGEND data set, describing the text you'd like to display beneath each bubble. For each of these observations assign FUNCTION = 'label'. To center the text specify POSITION = '5'. Set the variables SIZE, COLOR, and STYLE as desired. Again it may take some experimenting to determine which values of X and Y appropriately position the text beneath each bubble.

```
data legend;
format color function $10. text $50.;
xsys='3'; ysys='3'; hsys='1'; when='a';
function='pie';  rotate=360; ; style='psolid';
y =14; x=7; color='cxbe0000'; size = 0.115*sqrt(2000); style='psolid';
output;
y = 13; x=16; color= 'cxfa0000'; size = 0.115*sqrt(1000);style='psolid';
output;
y = 12; x=24; color='cxf08229'; size = 0.115*sqrt(500); style='psolid';
output;
y = 11; x=31; color='cxe6e600'; size = 0.115*sqrt(400); style='psolid';
output;
function='label'; position='5'; size=2; color = "black"; style='times
amt/bold';
y=8;
x=7; text="2,000+"; output;
x=16; text="1,000+"; output;
x=24; text="500+"; output;
x=31; text="Under 500"; output;
y = 5.5;
x=7; text="Gallons"; output;
x=16; text="Gallons"; output;
x=24; text="Gallons"; output;
x=31; text="Gallons"; output;
```

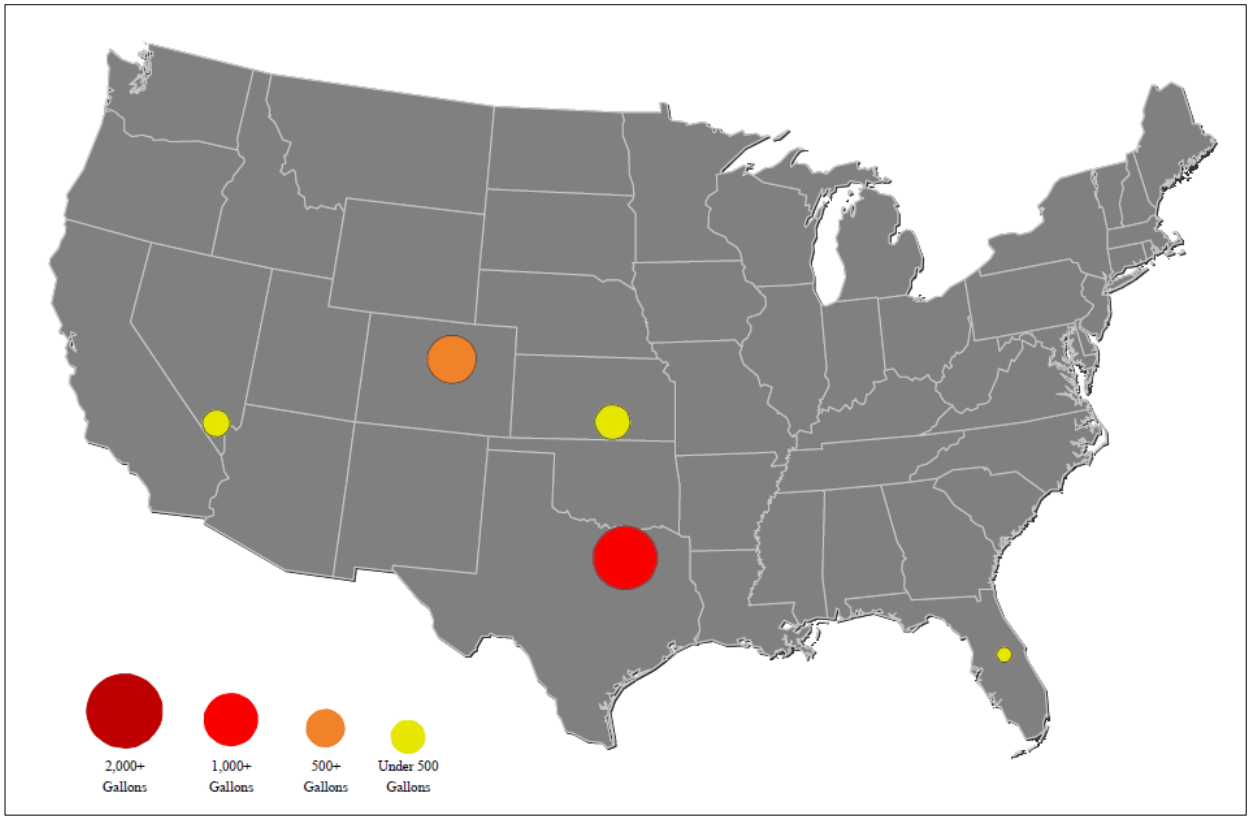
To display the legend on the map, combine it with the existing annotation data set THEFTSPROJ3 created in Part 2, and use the new combined data set as the annotation data set in the GMAP procedure:

```
data thefts_and_legend;
set legend theftsproj3;

pattern1 color =gray;
proc gmap data = mapProj map = mapProj anno = thefts_and_legend;
id state;
choro dummy / statistic = first nolegend coutline = graycc anno = perim;
```

This generates the map shown in Figure 7.

Figure 7. Bubble map with annotated legend.



CONCLUSION

The steps outlined above can be used to create bubble maps for data from a variety of contexts. Begin by using PROC GMAP to create a background map for the geographical region of interest. Next, use PROC GEOCODE and PROC GPROJECT to convert the locations of the bubble events into coordinates compatible with the background map. Then add instructions for drawing the bubbles as annotations. Finally, add in the instructions needed to draw the legend, and plot that as an annotation data set when drawing the map.

REFERENCES AND RECOMMENDED READING

- Allison, R. 2012. *SAS/GRAPH®: BEYOND THE BASICS*. Cary, NC: SAS Institute Inc.
- SAS Knowledge Base / Papers. "TS-688 Defining Colors using Hex Values." Available at: http://support.sas.com/techsup/technote/ts688/ts688.html
- SAS Institute Inc 2012. *SAS/GRAPH® 9.3: Reference, Third Edition*. Cary, NC: SAS Institute Inc.
- SAS Institute Inc 2015. *The SAS Training Post Blog*. Cary, NC: SAS Institute Inc. Available at: http://blogs.sas.com/content/sastraining/
- Zdeb, M. and R. Allison. "Stretching the Bounds of SAS/Graph® Software." *Proceedings of the 30th Annual SAS Users Group International Conference*, Philadelphia, PA. April 2005. Available at: http://www2.sas.com/proceedings/sugi30/137-30.pdf

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Caroline Cutting
Warren Rogers Associates
401-846-4747 x180
cutting@warrenrogers.com
www.warrenrogers.com