Paper 230-2009

Tips and Tricks IV: More SAS/GRAPH® Map Secrets

Darrell Massengill and Jeff Phillips, SAS Institute Inc., Cary, NC

ABSTRACT

"You can't do that with PROC GMAP!" We hear that all the time. Many users don't know the full range of capabilities of the SAS/GRAPH® mapping procedures. This presentation will share the secrets to allow you to exploit the power of SAS/GRAPH maps. Get the maps you really want; you can do that with PROC GMAP! We will demonstrate this using both existing functionality and new SAS® 9.2 functionality.

INTRODUCTION

If a picture is worth a thousand words, then a map can be worth a thousand pictures. Maps today are much more sophisticated than simply colored regions. Maps are found everywhere in data presentation, analysis, and visualization, and can show multiple independent variables. From bubble maps to density plots to non-geographic maps, maps can convey much more information than is apparent at first glance using PROC GMAP. Enhancements in SAS 9.2 to map data preparation routines, in conjunction with preexisting functionality, open a world of new maps for your reporting.

This paper contains real user scenarios and shows how to create rich, informative maps that many users do not think are possible. Many of the tips presented have been available for some time, and a few are new to SAS 9.2. Our examples illustrate how to exploit these capabilities.

We present three types of maps: bubble maps, distance and location maps, and floor plan maps. All of the examples in this paper can be downloaded from http://support.sas.com/rnd/papers.

BUBBLE MAPS

Bubble plots have been around for many years. They are useful to convey additional information interpreted in the size and color of the plotted circle. Maps have used this concept for quite some time, and bubble maps are commonly seen these days. They have been especially popular among the election-tracking Web sites in the United States during the 2008 election cycle. Bubble maps give you a mechanism to show multiple pieces of information on your map at the same time.

BLOCK-AREA MAPS

PROC GMAP has the ability to show multiple pieces of information in BLOCK-AREA maps. The AREA statement was introduced to PROC GMAP in SAS 9.2 so that maps can show two independent variables at once. AREA had previously been an option to the BLOCK statement, but it was limited to using one of the ID variables. Now, any independent variable can be used.

We can use the AREA statement, along with the BLOCK statement, to show two independent variables using the following code:

```
proc gmap data=iaresults map=iamap;
id county;
block margin / nolegend shape=cylinder;
area candidate / legend=legend1;
run;quit;
```

In figure 1, when using the BLOCK/AREA map, we can see the candidate who won each county, and those counties that gave the largest margins of victory. However, each block starts at a different baseline, which makes it difficult to compare to other similar regions and the viewer loses the bigger picture. Also, the tilt of the map with the blocks in front tends to hide some information on the northernmost counties. At a glance, you can see that Obama seemed to get a few more counties than Edwards or Clinton. You can also see that Obama won the eastern portion of the state, while Clinton took the western and Edwards the southern.

While this map conveys all the information about the caucus, it still requires the viewer to infer information to conclude that Obama won. In this and other cases, a bubble map does a better job of conveying the information.

THE SIMPLE BUBBLE

In January 2008, a popular U.S. newspaper Web site contained a map that showed the Iowa Democratic Party Caucus county results using a bubble map. The color of the bubble indicated which candidate won that county, while the size of the bubble showed the margin of victory.

For us to create this in SAS/GRAPH, we had to use the Annotate facility and a little brute force DATA step. The annotate function that is used is PIE with a 360-degree rotation. To display the bubbles correctly, consider the following:

- It is important to note that while the newspaper Web site used transparent colors to show the bubbles on top of the map, we have to use opaque colors in SAS/GRAPH. Therefore, it is crucial that all of the bubbles are rendered before the map is drawn. This is accomplished by setting the annotate variable WHEN to B.
- The bubbles should be sorted by descending size so that large bubbles don't obscure smaller ones.
- The maximum size of a bubble in the map should be relative to the closeness of the most crowded regions. In this example, the state of Iowa is rather evenly laid out, with no crowded, small regions. A map of the United States, however, has a very crowded northeast region and would not support very large bubbles because of the opaque colors.

In figure 2, when we view the bubble map made using the Annotate facility in SAS/GRAPH, we can easily see how Barack Obama won the state. It is much clearer because the color and size are charted together on the same bubble. This makes comparisons with other bubbles easier. In fact, instead of noticing the geography, you notice the amount of green in the map. It is easier and quicker to get the big picture using the bubble map. The bubble map has added a legend and annotated the largest city locations by merging with the MAPS.USCITY data set, provided with SAS.

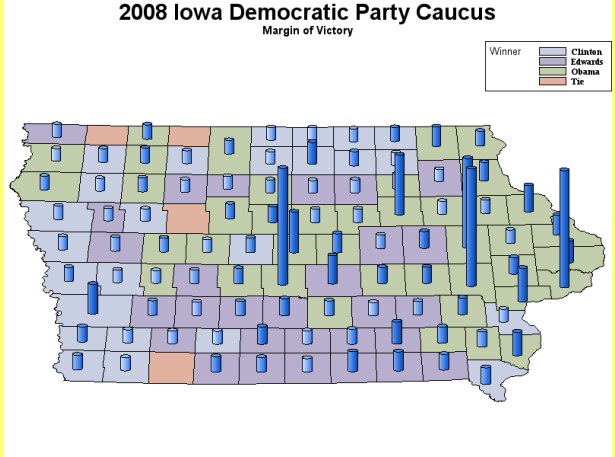

Figure 1: Block Map

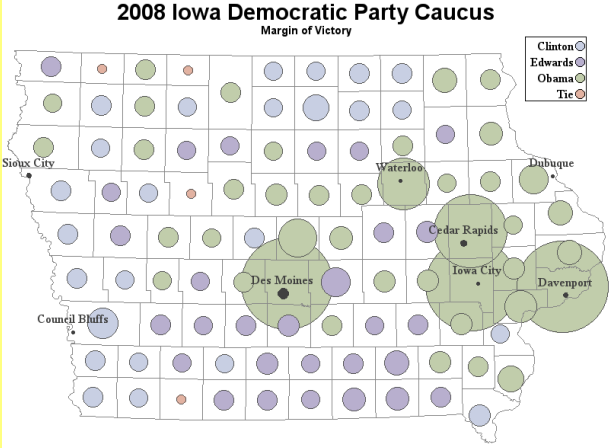

Figure 2: Bubble Map

It is important to note that while we can gain a lot of information quickly from the bubble map, there are still certain regions that might not display well because of crowding and a lack of transparent colors. Future considerations for the development of transparent colors (alpha values) in SAS/GRAPH will allow for this to become a statement in PROC GMAP.

THE COMPLEX BUBBLE: PIE

Because a bubble map is made of a series of PIE functions, we can add even more information to these maps by creating slices in each bubble. Note that a map with pie annotations can get busy very quickly, so these should be used sparingly and only when there are a relatively few number of slices. The eye is not going to be able to reasonably deal with more than about three slices of data. However, this type of map has been requested by a number of SAS users who want to study multiple relationships like market share versus size of market.

If each slice represents a company's market share, and the size of the pie represents the overall size of the market, then you can easily show the company-to-total relationship. Therefore, small slices of big pies can be just as good as or better than a big slice of a smaller pie, depending on the business's perspective on growth and market share.

To illustrate, let's return to the Iowa Democratic Party Caucus map, and instead of charting the margin of victory, let's see how each candidate actually did in each county. This data can be easily substituted as market share, individual sales figures, customer demographics, or any other classified data.

The only changes needed for representing the pies is the annotation variable ROTATION is a static 360 degrees for the bubble map, where each slice represents a percentage of 360 degrees. In addition to all of the issues with displaying bubbles, consideration needs to be given to colors. Because we expect a lot more data in a small area, we need to choose higher-contrasting colors. Figure 3 shows why the pastel colors of the original bubble map make it difficult to discern the correct information. The higher-contrasting colors make it much easier to see all the information at a glance.

At this point, the map shows four pieces of information: geography, candidates, relative number of votes, and the percentages for each candidate. Therefore, the decision to display this level of detail on one map needs to be taken seriously—there is a diminishing return when a map gets too crowded for the viewer to understand the data appropriately. Adjustments should be made to the data so that only the most pertinent information is displayed.

For example, if someone wanted to show the market share of his company against nine competitors, he should have one slice representing his market share, and only one other slice representing all the competition in the market. At this level of detail, it is not going to be discernable when small adjustments to the nine competitors change. It only complicates the map, and the viewer will not see the information that he wants them to see. If more detail is necessary, the information can be provided through HTML drill-down functionality or pop-up windows.

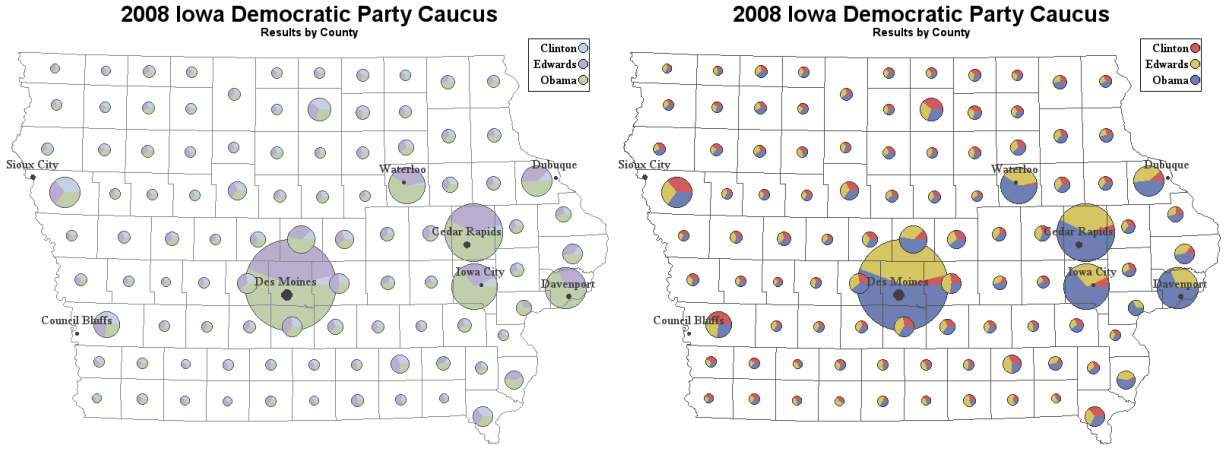

Figure 3: Comparison of Pastel and Higher-Contrasting Colors

This example has shown how bubbles can add an element of data visualization that, while not built into PROC GMAP, is not difficult to produce through a DATA step, color choosing, and sorting. Strong consideration should be given to the layout of the geographic regions being shown so that the map does not become too crowded. Also, based on the crowding of regions, the maximum size of a bubble should be set accordingly. Bubbles and pies do not work in every situation, just like block maps might not be the best choice for presenting your data. These are all considerations that need to be weighed before deciding which route to take, and how much information can be reasonably displayed in one map.

GEOGRAPHIC DISTANCE AND LOCATION MAPS

The demand for location analysis and discovery is increasing in reports. These types of maps allow exploration if distance and location are important decision-making factors. For example, are your most profitable customers close to your store? Do your most important crisis-management employees live near your office? When you mail sales fliers to customers, is distance a factor in who buys afterward?

For our example, let's create a user scenario. As an analyst at Santa Anna State University (better known as SAS U), located in Temple, Texas, your assignment is to analyze last year's freshman class and find out why so many students did not return for their sophomore year. SAS U is a state university, so only in-state students will be analyzed.

Texas is a large state and clearly homesickness might be a reason a student would not return. By plotting your students' home locations on a map, you might discover something. First, you need the location of each in-state student. In your database, you have only a student ID and an address, so there are no other biases.

Several new tools have been added in SAS 9.2 that aid in the location and distance analysis:

- PROC GEOCODE—calculates a longitude and latitude for a given address. This is useful for plotting points on a map where only an address or postal code is available.
- GEODIST Function—calculates a distance from point to point using geographic longitude and latitude coordinates. The algorithm uses the Great Circle Distance for calculations, making the measurements more accurate because longitudes become closer to each other the farther the latitude is from the equator. The ZIPCITYDIST function is a companion that calculates distances between U.S. Zip codes.
- %REDUCE_PIXEL macro— available from SAS Maps Online, this macro reduces map data optimally for a desired resolution.
- PROC GINSIDE—applies regional (polygonal) values to a point. If the point is inside a geographic region, then it inherits the attributes of that region.

These new features are important in analyzing location information. What role does distance and location play in a given scenario? Certainly, the distance a customer has to travel has implications in the retail and education industries. Property and asset location is important in banking, the supply chain, and government entities. These are only a few examples of how the demand for location analysis has increased.

THE GEOCODE PROCEDURE

Using the following syntax, we get a data set with X and Y variables added:

```
proc geocode data=studentdb out=studloc zip nocity attributevar(city);
run;
```

The NOCITY option tells PROC GEOCODE that there is not a city variable in the input data set. By using the ZIP option, it uses SASHELP.ZIPCODE to locate the longitude and latitude. The ATTRIBUTEVAR(CITY) option copies the city variable from the SASHELP.ZIPCODE data set to the output data set. The resulting data set has the original values, plus the X (longitude), Y (latitude), and CITY columns added.

It is important to note that the location variables from the SASHELP.ZIPCODE data set are in degrees, but the unprojected maps available in the MAPS library are in radians. There is a very simple DATA step that converts these degrees to radians. Also, the MAPS library data sets have not been corrected for the Western Hemisphere, so we will need to negate the X value during the conversion.

```
data studloc;
retain anno_flag 1;
set studloc;
x = -x * atan(1) / 45;
y =  y * atan(1) / 45;
run;
```

The ANNO_FLAG value comes in handy in the next step, where we put the map data together with the annotation data. This is done so that PROC GPROJECT uses the combined data to calculate defaults for the projection. After projecting, the data sets can be separated again by checking the ANNO_FLAG value.

THE %REDUCE_PIXEL MACRO

Many SAS/GRAPH map data sets are quite detailed and contain a large number of points. Most of these points are not visible at certain resolutions. In other words, there can be more than one map point per pixel at smaller resolutions, making them superfluous and unnecessary. As SAS/GRAPH map data sets continue to evolve, these data sets will grow larger and more detailed in the future. In addition, the desire for higher-resolution maps will increase. However, the availability of the data will make it easier to supply these high-density maps.

Therefore, if our output resolution is 800×600, the %REDUCE_PIXEL macro (available for free at http://support.sas.com/rnd/datavisualization/mapsonline/html/tools.html) can reduce the number of points in the Texas county map from 22,685 to 3,042, which is a reduction of over 86%. This greatly increases the speed of processing the map data without sacrificing output quality.

```
data txmap;
length x y 8;
set maps.counties;
where state=48;
run;

%include "reduce_pixel.sas";
%reduce_pixel( txmap, txmap, county, 800, 600 );

data combined;
set txmap studloc;
run;

proc gproject data=combined out=combined dupok;
id county;
run;

data txmap anno_dots;
set combined;
if anno_flag > 0 then output anno_dots;
else output txmap;
drop anno_flag;
run;
```

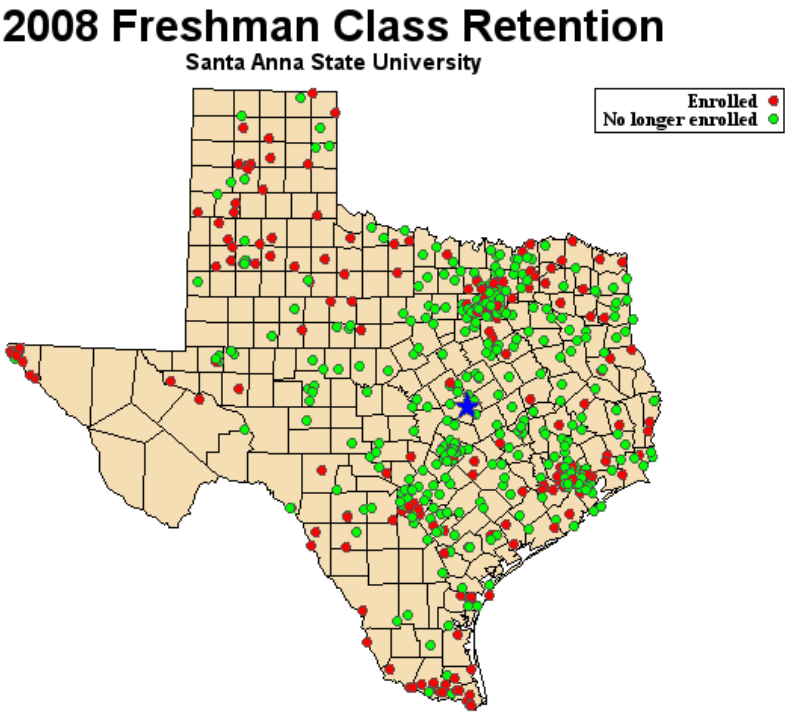

Figure 4: Distribution of Freshmen and Their Matriculation Status

Now the ANNO_DOTS data set can be extended to use the Annotate Facility functions to plot the points. The PIE function generally looks better than the POINT function. The map in figure 4 shows the dots (pies) colored red for non-matriculated and green for students still at SAS U. To increase the aesthetics of the dots, we drew an outline around each pie. This is done with successive PIE calls, alternating PSOLID and PEMPTY styles at the same location. The PROC GMAP statement to produce figure 4 is the following:

```
proc gmap data=txmap map=txmap anno=anno_dots;
id county;
choro state / nolegend;
run;
quit;
```

It does become somewhat evident based on the distribution and color of the points that the closer a student lives to the campus, the more likely they are still matriculated. But how much of a role does distance play in a student's decision not to return? A deeper analysis is necessary.

Now that we have the student locations from PROC GEOCODE, we can use the GEODIST function to calculate the distance each student is from the school. Once we have the distances, we can use PROC SUMMARY to calculate the average distance.

THE GEODIST FUNCTION

The GEODIST function is a DATA step function that can be used to add another variable to the student location data set. Note that the calculation of the distance needs to happen before the data is projected. The function uses the unprojected geographic coordinates in its calculations. Once the data has been projected, the original longitudes and latitudes are lost.

The calculation should occur in the program before the data is merged with the map data. In our example, the first observation in the data set is the school location.

```
data studloc;
retain sasu_x sasu_y;
set studloc;
if _n_ = 1 then do;
   sasu_x = x;
   sasu_y = y;
end;
else dist = geodist( y, x, sasu_y, sasu_x, 'DM' );
drop sasu_x sasu_y;
run;
```

This code excerpt should happen just after the PROC GEOCODE statements. At this point, the X and Y values are still in degrees. The last parameter passed to the GEODIST function is a string, where the first character is the input units (D=degrees, R=radians), and the second character is the output units (M=miles, K=kilometers). If we wait until after the X and Y values are converted to radians, then RM would be used instead of DM.

At this point, the DIST variable has the distance, in miles, of each student to the SAS U campus. Now, we can calculate the average of these distances and save it into a macro variable to be used later.

```
proc summary data=studloc;
var dist;
output out=temp mean=mndist;
run;

data _null_;
set temp;
call symput( "mndist", left(put(round(mndist), f3.0)));
run;
```

To see what the average radius looks like, it needs to be plotted on the map. An empty PIE function in annotate would show it nicely, but we have yet to project the map, and the radius of the pie does not get relatively projected. In other words, the center of the pie gets projected, but the radius is still in degrees or radians. These are not the same coordinate systems, so even if it showed on the map, it would be wrong.

The only way to preserve the radius is to create a polygon using points along the arc. These created points get projected into the new map. This requires a bit of trigonometry, but should be relatively simple to calculate. To show this polygon on the Texas map, annotate variables are added to the data set.

```
data circle;
length text $25 color function $8 style $20;
retain xsys ysys '2' anno_flag 2 when 'A' text ' ';
set studloc(obs=1 keep=x y);

/* degrees to radians constant */
d2r=atan(1)/45;

/* miles per radian constant */
r2m=3958.739565;
```

```
/* save the center point */
xcen = x;
ycen = y;

/* Create 72 points at 5 degree intervals around the circle */
do degree = 0 to 355 by 5;
   if degree = 0 then do;
      function='poly';
      style='empty';
      line=1;
   end;
   else do;
      function='polycont';
      color='black';
   end;

   y=arsin( cos(degree*d2r) * sin(&mndist/r2m) * cos(ycen) +
            cos(&mndist/r2m) * sin(ycen) );
   x=xcen + arsin( sin(degree*d2r) * sin(&mndist/r2m) / cos(y) );
   output;
end;
drop degree r2m d2r xcen ycen;
run;
```

At this point, the resulting data set has X and Y pairs for points in a circle, and a constant ANNO_FLAG value of 2. Like the previous step showing the annotated points, this data set needs to be merged with the Texas map, projected, and then split up again using the ANNO_FLAG value. Only non-missing values need to be checked, because we only need a single annotate data set for PROC GMAP. (Remember, our original points used an ANNO_FLAG value of 1.) Figure 5 shows the average radius on the map with the student locations.

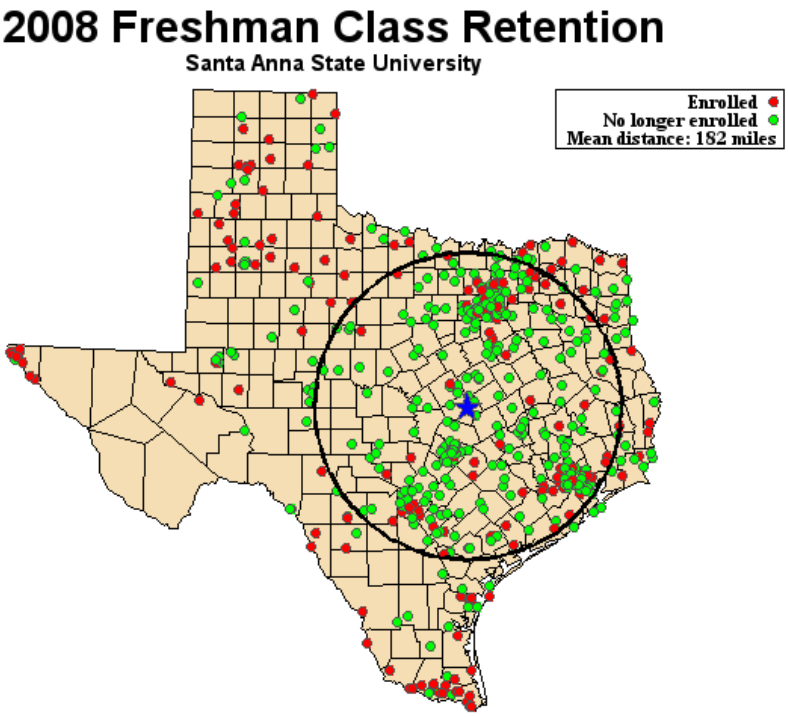

Figure 5: Annotation of All Students and the Average Distance

It certainly seems that most of the matriculated sophomores who attended SAS U last year are inside the average distance circle, but we are still not sure what role distance played in deciding if a student returned. To find out, we also plot the average distance of those who stayed versus those who left the university.

This is done much the same way as the overall average distance was calculated. A WHERE clause needs to be used with PROC SUMMARY to subset the data on which the average is calculated. In the student data, there is a Boolean column that indicates whether a student is still matriculated. If that value is used in a WHERE clause, then two new average distances can be calculated.

```
proc summary data=studloc;
where matric=1;
var dist;
output out=temp mean=mndist;
run;

data _null_;
set temp;
call symput( "staydis", left(put(round(mndist), f3.0)));
run;
```

Because we now need to duplicate the circle polygon code, we create a simple macro to pass in a suffix for the data set, the color for the circle in the annotation, and the calculated distance. We update our annotated legend to include the new circle representations. Also, for presentation purposes, the student points should have their colors dimmed so that the circles stand out a little more. For example, the red color has the red RGB component reduced from FF to CC (from 255 to 204, in decimal), and likewise with green.

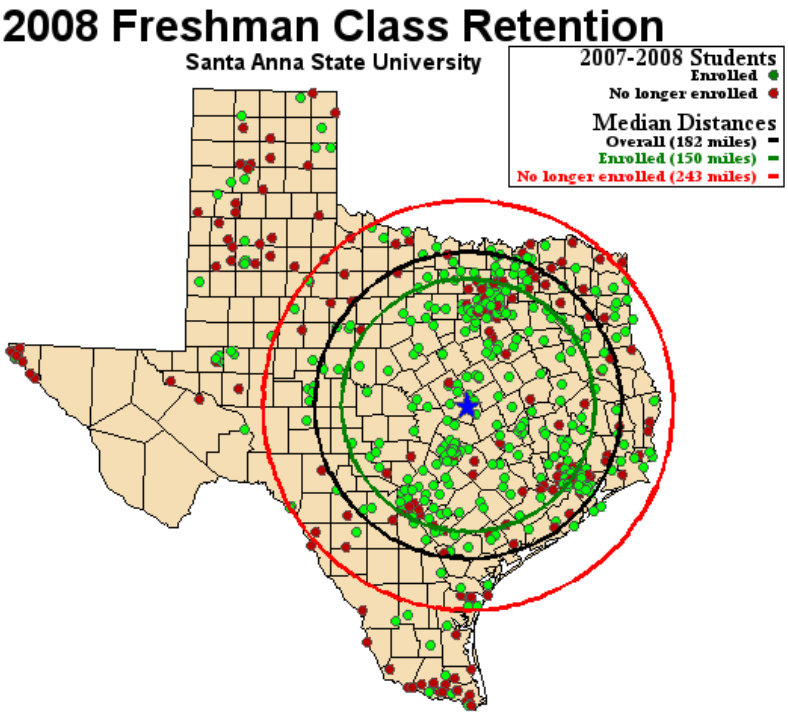

Figure 6: Distribution of Freshmen and the Three Average Distances

It becomes clear in figure 6 that distance definitely plays a role in a freshman student's decision to remain at SAS U. The overall average distance is 182 miles, but the average distance of the students still matriculated is 150 miles—32 miles closer than the overall average. Likewise, the average of those who left is 243 miles, 61 miles greater than the overall average, and almost 100 miles greater than those who stayed.

However, it is evident that there are students who stayed at SAS U who reside more than the average distance away, and there are those who left the school after their first year who are inside the circle. Because we have determined

that distance is a contributing factor in student retention, we now turn the focus of our analysis on the students whose distance apparently did not play a factor.

THE GINSIDE PROCEDURE

There is nothing that the school administration can do to solve the distance problem, but if it can better understand what makes SAS U attractive to students at farther distances, and find out what would make a student leave the university who resides relatively close, then the school administration can better market the university to prospective and current students.

At this point, we could simply subset the data based on matriculation status and the distance we calculated in the previous steps. However, we do not want to split counties that bisect the circle. Therefore, we need to do the following:

1. Calculate the centroids of each county in the map data.
2. Calculate the distance from each county centroid to the school.
3. Mark the county as inside or outside the average radius.
4. Mark the student point as inside or outside the calculated counties using PROC GINSIDE.

The first step is straightforward because there are tools available in the annotate macro library for this task. We have seen how to calculate the distances. So, the only step that is left in the program is to mark the counties and the points using output from PROC GINSIDE.

```
/* Step 1:  Calculate the centroids */
%annomac;
%centroid(txmap, txcent, county);

/* Step 2:  Calculate the distances */
data txcent;
retain sasu_x sasu_y;
set studloc(obs=1 in=want keep=x y) txcent;
if want then do;
   sasu_x = x;
   sasu_y = y;
end;
else do;
   dist = geodist( y, x, sasu_y, sasu_x, 'RM' );
   if dist > &mndist then inside=0;
   else inside=1;
   output;
end;
drop sasu_x sasu_y dist;
run;

/* Step 3:  Mark counties as inside or outside */
proc sort data=txcent nodupkey;
by county;
run;

data txmap;
set txmap txcent(keep=county inside);
by county;
run;

/* Step 4: Mark student locations as inside or outside */
proc ginside data=studloc map=txmap;
id inside county;
run;
```

At this point, the student location data set has an additional variable—INSIDE—that indicates whether the student's county is inside the radius or not. The name of that variable came from our new map data set created in step 2. Recall that our interest lies with those no longer matriculated inside the radius, and those still matriculated outside the

radius. We can now subset the data for a report, but first, let's see the points on a map (Figure 7). We have shaded the inside counties slightly darker in the output to further emphasize our goal.

```
data studloc;
set studloc;
where (inside=1 and matric=0) or (inside=0 and matric=1);
run;
```

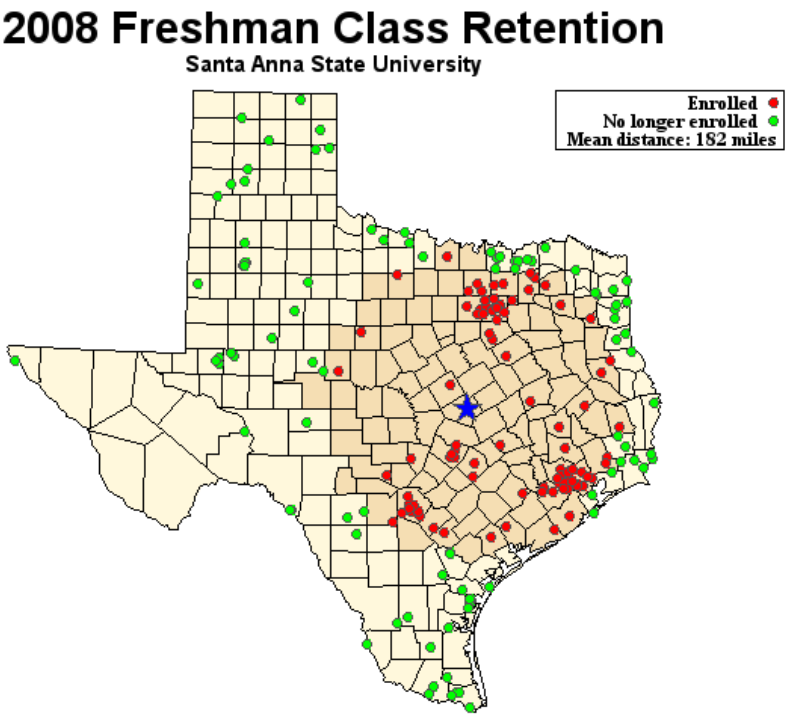

Figure 7: Students of Interest Indicated by Distance and Matriculation

By subsetting the data from the original 619 students, we have been able to identify 94 students who are inside the average distance, yet left the university, and 98 students who are outside the average distance who were retained. Of the 94, we can further subset based on GPA to identify only those who left for non-educational (grade) reasons. Likewise, we might be interested in those who stayed who have a GPA that indicates the initiative to complete their degree. Further interviews might be conducted with a sampling of these students to best determine where and how to market and advertise the university to prospective new students.

Location and distance discovery will continue to present challenges, but with these new procedures and functions from SAS, you can easily write a report or create an interactive tool to produce the results that are needed to tackle these challenges.

FLOOR PLAN MAPS

A floor plan map is an example of a non-geographic map. While non-geographic maps look very different from typical maps, they behave in exactly the same way. The map data is the only real difference. Instead of showing the information for country, state, county, or province, the information is displayed for rooms, equipment tables, products, and other items in a room.

Floor plan maps are extremely useful for tracking, reporting, and analyzing activity. This type of map could be used to show where the most popular products are located in a store. It could be used to view which tables in a restaurant are the most profitable. Our example uses the floor plan to determine which slot machines in a casino are the most profitable.

CASINO FLOOR PLAN

Modern casinos are electronic marvels. Game results are tracked and collected electronically. Slot machines contain computers that track every play, every payout, and all other relevant information. The owners of the Casino 401K ("where your retirement investment is as safe as Wall Street") want to visually see the income from each slot machine, the payout of each machine, and the number of times a machine is played. With this information, they can tell if a machine is malfunctioning or, perhaps, if there is a spill or obstacle hindering the use of the machine.

A casino floor map is ideal for visually examining the performance of the slot machines. Our example displays only the income generated by each slot machine, but any of the collected data could be displayed.

To meet the casino owners' needs, our example includes the following steps:

1. Create casino map data.
2. Process a blueprint image.
3. Process the response data to view.
4. Draw the map.

The key element in making a floor plan map is having the map data showing the floor plan.

CREATE MAP DATA

There are numerous ways that you can get the map data for the floor plan. The floor plan can be as simple as an outline of the building, with rectangles placed in the building to show the slot machines. Or, it can be as complex as you want. For this example, we wanted to make the floor plan look more like the building by having a blueprint image displayed under the drawing.

In this example, the floor plan is created from three pieces of information:

- an image file containing the blueprint drawing of the floor plan
- map coordinates for the exterior walls of the floor plan
- map coordinates for each slot machine on the casino floor

We start with a drawing of the floor plan that was scanned as an image file (casinofloor.gif). The image file was traced with external software to generate coordinates for the external walls. This process only needs to be done once. The resulting coordinates can be used to create a map data set of the external walls.

Our example uses the following code to create a map data set with the coordinates:

```
data work.casino;
   length id $20;
   input x y segment id $;
   cards;
  0.00000000 320.03425125  1 Room
  0.00000000   0.00000000  1 Room
250.17123654   0.00000000  1 Room
250.17123654 109.93150845  1 Room
190.06849593 109.93150845  1 Room
190.06849593 169.52055042  1 Room
281.50685343 169.52055042  1 Room
281.50685343 320.03425125  1 Room
  0.00000000 320.03425125  1 Room
;
run;
```

The slot machine locations are needed next. These can be created as map data set polygons or as annotated polygons. For this example, it is easier to work with map data set polygons. The positions of the slot machines were captured with the same software that captured the external wall coordinates. Drawing these small rectangles by hand was difficult, so we created a program to simplify the drawing.

MAKERECTANGLE.SAS contains a macro program that draws a bank of rectangles from the X and Y coordinates of the upper-left corner of the bank. It draws a specified number of rows and columns of rectangles, and draws them either horizontally or vertically. The slot machines sit inside the building polygon, so a hole must be created for each polygon. This macro creates the rectangle and the hole for it. The size of the rectangles was customized for this

map, so changes might be required for other maps. This file not only contains the macro, but also contains the calls to the macro to create the slot machines in this example. It creates SASUSER.SLOTS.

Our program simply includes this program to create the slot machine data set:

```
%include '.\makerectangle.sas';
```

Once we create these two data sets, we combine them into a single map and the map data set creation is complete:

```
data sasuser.casinofloor (keep=x y segment id htmlvar);
   set work.casino sasuser.slots;
run;
```

PROCESS BLUEPRINT IMAGE

At this point, you could use the map data set you created to represent the casino floor. However, we wanted to make our floor plan look more like a blueprint. Therefore, we displayed the created image behind the map data set. Figure 8 shows what our floor plan would look like with only the map data set. Figure 9 shows the addition of the image.

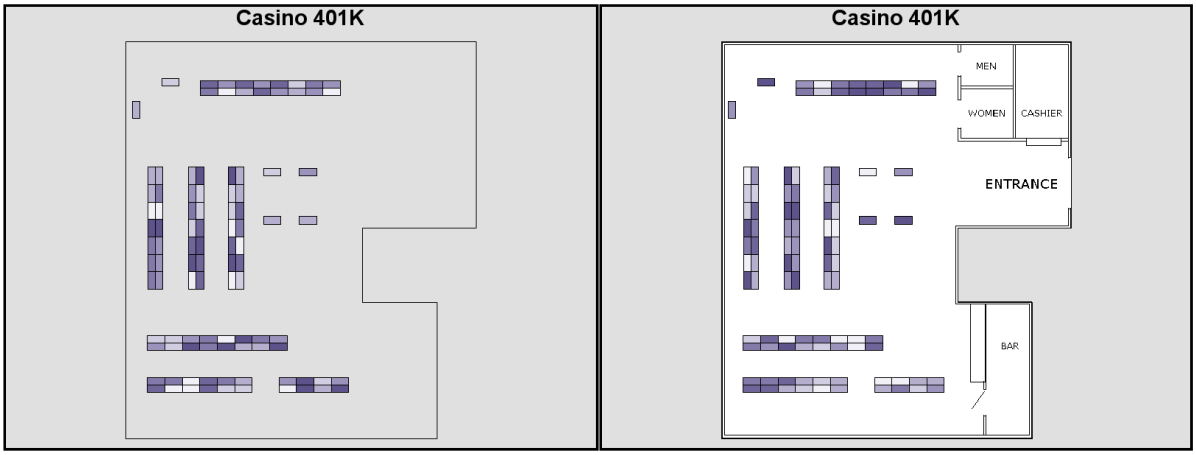

Figure 8: Floor Plan with No Background Image **Figure 9: Floor Plan with Background Image**

First, we scanned the blueprint and created an image. Now, we process this image so that it can be used with the map. The trick to using an image like this is to first crop the image to the outside walls. That is, the parts of the image that are outside of the building are cut away. This gives us the ability to exactly place the image on the map.

To place the image and map in the same location, we must know the minimum and maximum coordinates of the building map data set. In our example, we know that the coordinates are all positive numbers. The following code is used to determine the MIN and MAX values, which are then assigned to macro variables for later use:

```
data temp1;
   set work.casino;
   retain maxx 0 maxy 0;
   retain minx 9999999999 miny 999999999;
   if x <= minx and y <= miny then do;
      minx=x;
      miny=y;
   end;
   else if x >= maxx and y >= maxy then do;
      maxx=x;
      maxy=y;
   end;
run;

data temp1;  /* assign the max/min values to macro variables */
   set temp1;
   call symput("minx",minx);
   call symput("miny",miny);
```

```
   call symput("maxx",maxx);
   call symput("maxy",maxy);
run;
```

To display the image in the correct location, use the MIN and MAX values in an annotate data set to position the image at the same location as the exterior walls:

```
data imageplan;
   length  function style color $ 8 html $ 1000 text $ 50;
   retain xsys '2' ysys '2' hsys '3' when 'b';

   /*start with the minimum x,y values of the floor plan map data */
   function='move';x=&minx; y=&miny; output;
   /*and draw the image to the max x,y values of the floor plan map data */
   function='image'; x=&maxx; y=&maxy;
          imgpath='.\casinofloor.gif'; /* the floor plan image */
          style='fit';output;
run;
```

The MOVE function positions the starting point of the map at the minimum coordinates. The IMAGE function causes the image to draw from the starting location to the maximum coordinates. Because the image was cropped to the exterior walls, the image and map match. The variable WHEN is set to B to draw the image before the map.

PROCESS RESPONSE DATA

This example doesn't have any real data showing the slot machine activity, so it is generated with a random number generator each time it is run. This data contains only the slot machine ID and the income of that machine. This data could be used without modification on the map; however, this example is going to add tooltips to the data so that hovering over a slot machine will show the machine number (ID), the type of machine (for example, $5 Slot Machine), and the amount of income from the machine. The slot machine map data knows about the type of machine, but the response data does not. To get this information, the slot machine map data must be merged into the response data. The variable containing the tooltip information is named HTMLVAR.

```
proc sort data=sasuser.slots; by id; run;
data usage2;
  merge usage sasuser.slots; by id;
  if (id ne 'Room') then /*skip the Room polygon*/
  htmlvar= 'title=' || quote(trim(left(id)) || '  (' || trim(left(type))
                || ')  $' || left(trim(left(Income))));
  run;
```

Figure 10 shows the tooltips created by this code:

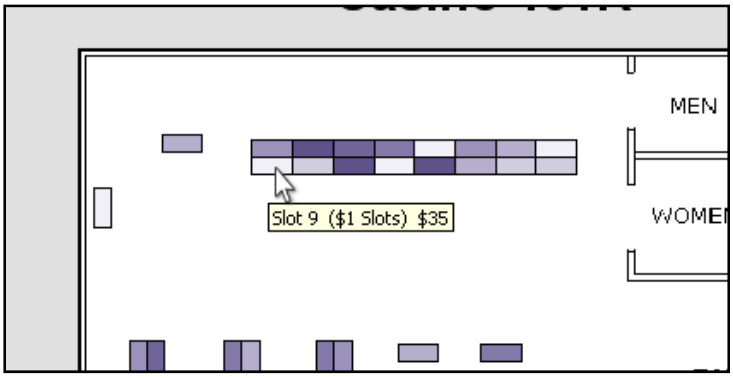

Figure 10: Tooltip Showing Slot Machine ID and Income

DRAW THE MAP

The final step is to display the map with PROC GMAP. We are using ODS to create our output. The map uses our modified response data with tooltips, the combined map data, and the annotate data set containing the image. The

ALL option causes all polygons to be displayed, even if they do not have a response value. The HTML= option specified the name of the HTML tooltip variable (HTMLVAR) that was created in the response data.

```
GOPTIONS reset=all;
GOPTIONS DEVICE=&dev;
ODS LISTING CLOSE;
ODS HTML path=odsout body="&name._&dev..html";
  Title 'Casino 401K';

  /* Display the combined map with the annotate data set containing the image */
  proc gmap data=usage2 map=sasuser.casinofloor anno=imageplan ALL;
      choro Income / nolegend html=htmlvar; id id; run; quit;
ODS HTML CLOSE;
ODS LISTING;
```

You now have a working floor plan for Casino 401K. Figure 9 shows the results. Darker colors indicate higher income. Hovering over the data gives you tooltips so that machines with low income can be examined in more detail. The owners of the casino might want to examine why Slot 9 (shown in Figure 10) is generating only $35 in income.

CONCLUSION

Maps are very important visual tools. SAS/GRAPH mapping has powerful capabilities that many users might not be aware of. These capabilities can create maps that satisfy most mapping needs. Tools like the Annotate Facility have been around for quite some time. In addition, we have outlined some of the newer tools available in SAS 9.2. These few examples should illustrate that with the tools provided and a little imagination, you can create whatever you want with PROC GMAP.

REFERENCES

SAS/GRAPH 9.2 Documentation: http://support.sas.com/documentation/onlinedoc/graph/index.html

SAS Maps Online Web Site: http://support.sas.com/mapsonline

RESOURCES

“Tips and Tricks IV: More SAS/Graph Map Secrets.” SAS Global Forum 2009 paper and example source code download. SAS Institute Inc. http://support.sas.com/rnd/papers.

“PROC GEOCODE: Creating Map Locations from Your Data.” SAS Global Forum 2009 paper and example source code download. SAS Institute Inc. http://support.sas.com/rnd/papers/.

“SAS Mapping: Technologies, Techniques, Tips and Tricks.” SUGI 28 SAS Presents handout and example source code download. SAS Institute Inc. http://support.sas.com/rnd/papers/.

“Tips and Tricks II: Getting the most from your SAS/GRAPH maps.” SUGI 29 SAS Presents handout and example source code download. SAS Institute Inc. http://support.sas.com/rnd/papers/.

“Tips and Tricks III: More Unique SAS/GRAPH Maps.” SUGI 30 SAS Presents handout and example source code download. SAS Institute Inc. http://support.sas.com/rnd/papers/.

SAS Global Forum Proceedings. http://support.sas.com/events/sasglobalforum/previous/index.html

SAS Customer Support: http://support.sas.com/.

SAS Online Documentation: http://support.sas.com/documentation/index.html

CONTACT INFORMATION

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

Darrell Massengill
SAS Institute
SAS Campus Drive
Cary, NC 27513
Darrell.Massengill@sas.com

Jeff Phillips
SAS Institute
SAS Campus Drive
Cary, NC 27513
Jeff.Phillips@sas.com