CHAPTER 7

Annotating on Maps

Purpose: This chapter demonstrates how to annotate markers and paths on maps.

Annotating Simple Point Markers

Many times, all you want to do is place simple markers at specific locations on a map. You're not trying to represent any data with the size/color/shape of the markers – you just want to mark some locations. Let's start with this simple case, and later we'll work our way up to more complex variations.

For example, let's place markers on a map for a few of the cities in Texas. The following code will get the latitude and longitude (lat/long) for the desired cities from one of the datasets that come included with SAS/Graph. Of course, you don't have to get the lat/long from this file – you could get the coordinates any way you want and store them in a similar dataset (look them up in a book, get them from a GPS, do a Google search, etc).

```
data anno_cities; set mapsgfk.uscity
(where=(statecode='TX' and
 city in ('Houston' 'Dallas' 'Lubbock' 'San Antonio')));
flag=1;
run;
```

The resulting dataset looks like this:

anno_cities

CITY	LAT	LONG	flag
Dallas	32.7833	-96.800	1
Houston	29.7631	-95.363	1
Lubbock	33.5778	-101.855	1
San Antonio	29.4239	-98.493	1

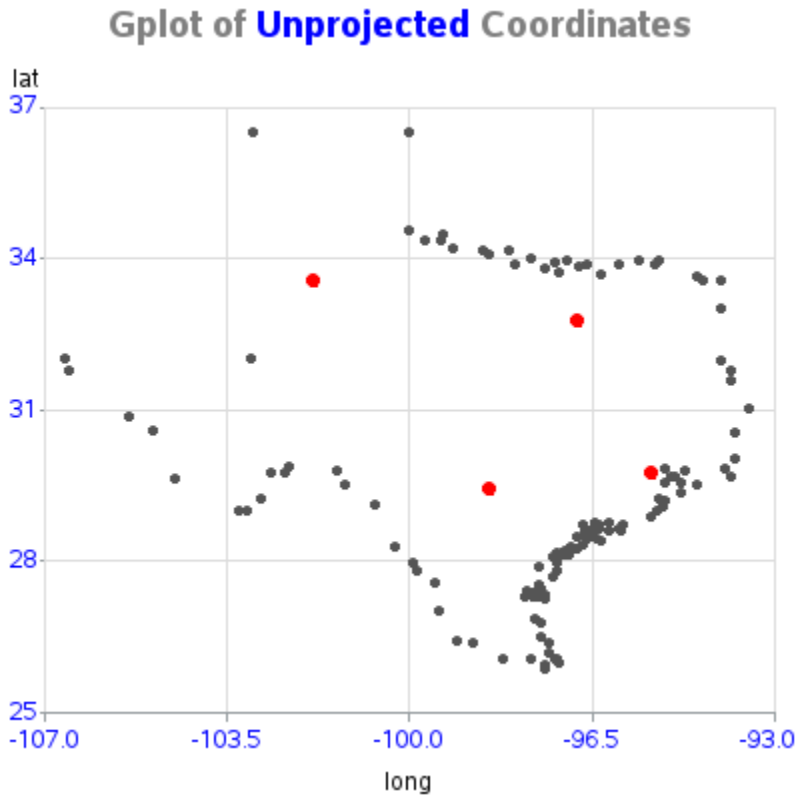
Next, you'll need the map of Texas. When plotting lat/long values on a map, I recommend using maps from the mapsgfk libname (in this case mapsgfk.us_states) rather than the traditional/old SAS maps such as maps.states. The reason being that your city coordinates are likely to be in eastlong degrees, and the mapsgfk maps are already in these coordinates (whereas the old/traditional SAS map coordinates are in westlong radians, and you would either have to convert your map or your city coordinates so that they match).

Here's the code to subset Texas out of mapsgfk.us_states (using the lower density values will create a map that shows slightly less detail around the borders, and the code will also run a little faster):

```
data my_map; set mapsgfk.us_states
 (where=(statecode='TX' and density<=3));
run;
```

Our end goal is to plot markers at the city coordinates on the map. But first, it might be useful to see everything (the map border coordinates and the city coordinates) in a scatter plot, just so you can get a better feel for the data, and make sure that your city coordinates are lining up with your map coordinates. Here's the code to do that (a little 'fancier' than necessary, but I wanted the graph to look pretty for the book!)

```
data combined; set my_map anno_cities; run;
symbol1 value=dot height=1.8 interpol=none color=gray55;
symbol2 value=dot height=3 interpol=none color=red;
axis1 major=(number=5) minor=none value=(c=blue)
  offset=(0,0);
proc gplot data=combined;
label lat='lat' long='long';
plot lat*long=flag / nolegend
 vaxis=axis1 haxis=axis1 noframe
 autohref autovref chref=graydd cvref=graydd;
run;
```



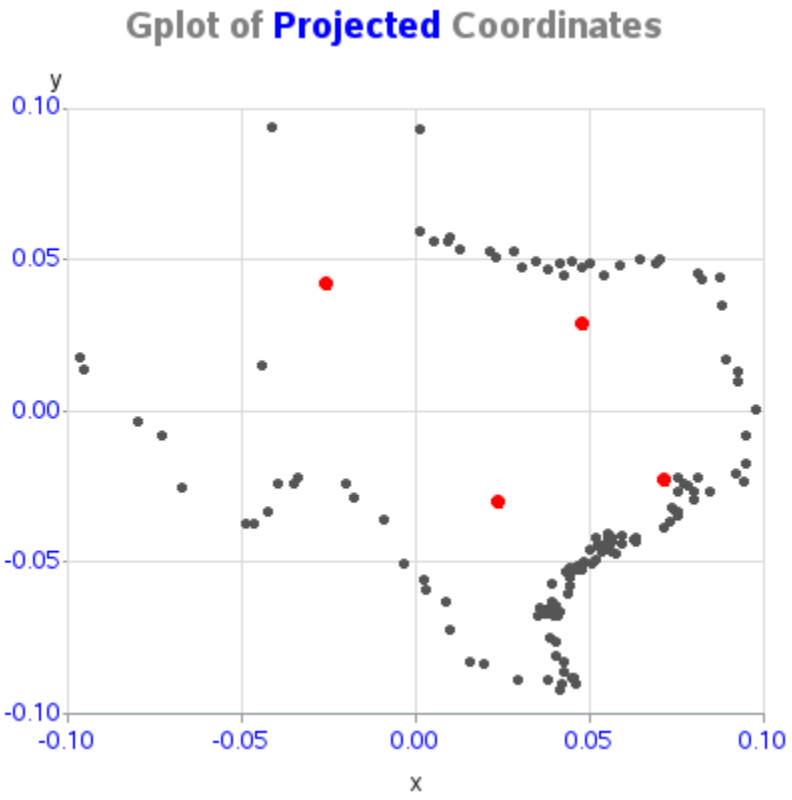
The next step is to project the map and city lat/long coordinates, to get the projected y & x coordinates you will plot with Proc GMap. In the case of Texas, projecting the coordinates doesn't make much visual difference in the shape of the map, but the farther north you go, the more important projecting the coordinates becomes. Whether the map visually needs projecting, it is the 'proper' thing to do, and I recommend you always go through this step.

There are a couple of different ways you could project the coordinates, but the one I use most frequently is to combine the map and the annotate data, project them together, and then separate them into two datasets again. Here's the code to do that (notice that I'm using the flag variable to separate the two datasets – this is the main purpose of having the flag variable). The GProject options signify that the coordinates are in eastlong degrees (as opposed to westlong radians), and stored in the lat and long variables (as opposed to y and x).

```
data combined; set my_map anno_cities; run;
proc gproject data=combined out=combined
 eastlong degrees latlong dupok;
id statecode;
run;
```

```
data anno_cities my_map; set combined;
if flag=1 then output anno_cities;
else output my_map;
run;
```

The resulting datasets contain projected values, stored in the X and Y variables. Here's a scatter plot so you can easily see what the data looks like:



The map dataset is ready to plot with Proc GMap, but the annotate datset (for the cities) still needs a little work. Currently, it has the following variables that we'll be using:

anno_cities

CITY	X	Y
Dallas	0.048	0.029
Houston	0.071	-0.023
Lubbock	-0.026	0.042
San Antonio	0.024	-0.030

In order to annotate markers at those x/y coordinates, we'll need to add a few variables to the dataset so that annotate will know what to do. The xsys and ysys tell annotate that the X and Y variables are in the same coordinate system as the map. The hsys tells it how to handle the size variable. When='a' tells it to draw the annotation after it draws the map (so it will be in front of the map, rather than behind it). And the rest of the variables tell it to draw a solid red pie at the specified x/y location.

```
data anno_cities; set anno_cities;
length function $8;
xsys='2'; ysys='2'; hsys='3'; when='a';
function='pie'; style='psolid'; rotate=360; size=1;
 color='red';
run;
```

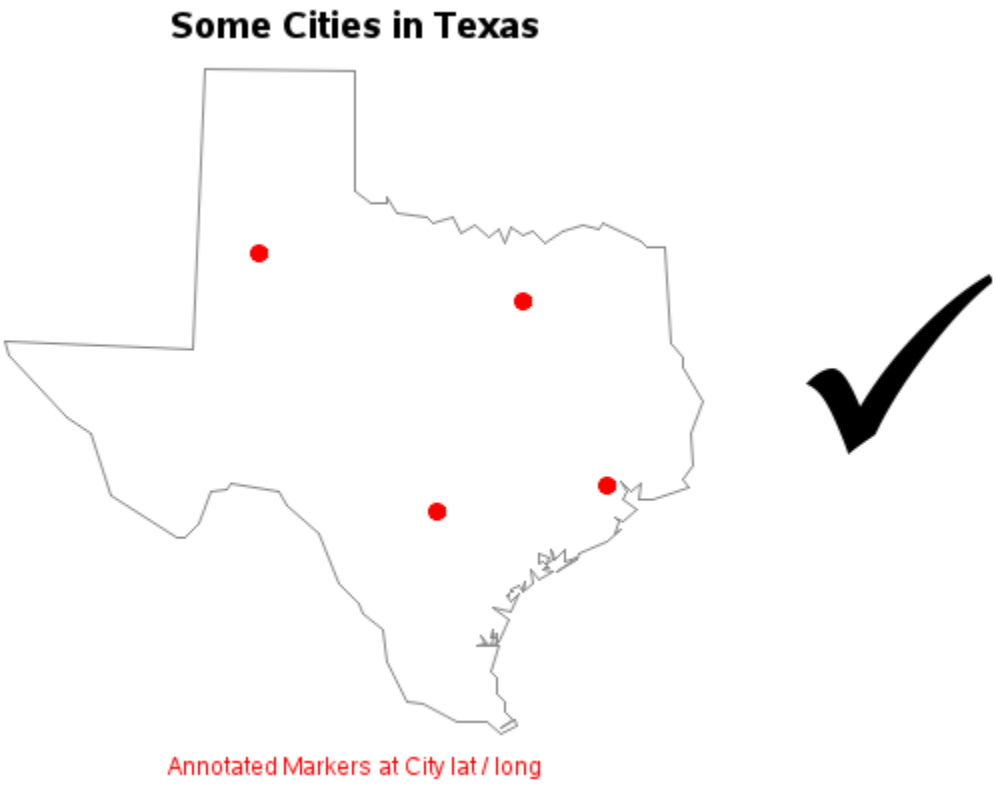
Here's are the essential variables of the completed annotate dataset:

anno_cities

CITY	X	Y	xsys	ysys	hsys	when	function	style	rotate	size	color
Dallas	0.048	0.029	2	2	3	a	pie	psolid	360	1	red
Houston	0.071	-0.023	2	2	3	a	pie	psolid	360	1	red
Lubbock	-0.026	0.042	2	2	3	a	pie	psolid	360	1	red
San Antonio	0.024	-0.030	2	2	3	a	pie	psolid	360	1	red

Now we can draw the map (with Proc GMap) and tell it to use anno_cities as the annotate dataset, with the following code:

```
title1 ls=1.5 "Some Cities in Texas";
pattern1 v=s c=white;
proc gmap map=my_map data=my_map anno=anno_cities;
id statecode;
choro segment / levels=1 nolegend coutline=gray88;
run;
```



One simple enhancement that is often useful, is to annotate a text label in addition to the pie marker. You can do this using the … you guessed it! … the label annotate function! The following code annotates the city with each marker. The position='2' causes the label to be above the marker, and I reset the style/rotate/size/color variables so the default text characteristics will be used.

```
data anno_cities; set anno_cities;
length function $8 text $100;
xsys='2'; ysys='2'; hsys='3'; when='a';
function='pie'; style='psolid'; rotate=360; size=1;
color='red'; output;
function='label'; position='2';
style=''; rotate=.; size=.; color='';
text=trim(left(city)); output;
run;
```

Here's the resulting dataset:

anno_cities

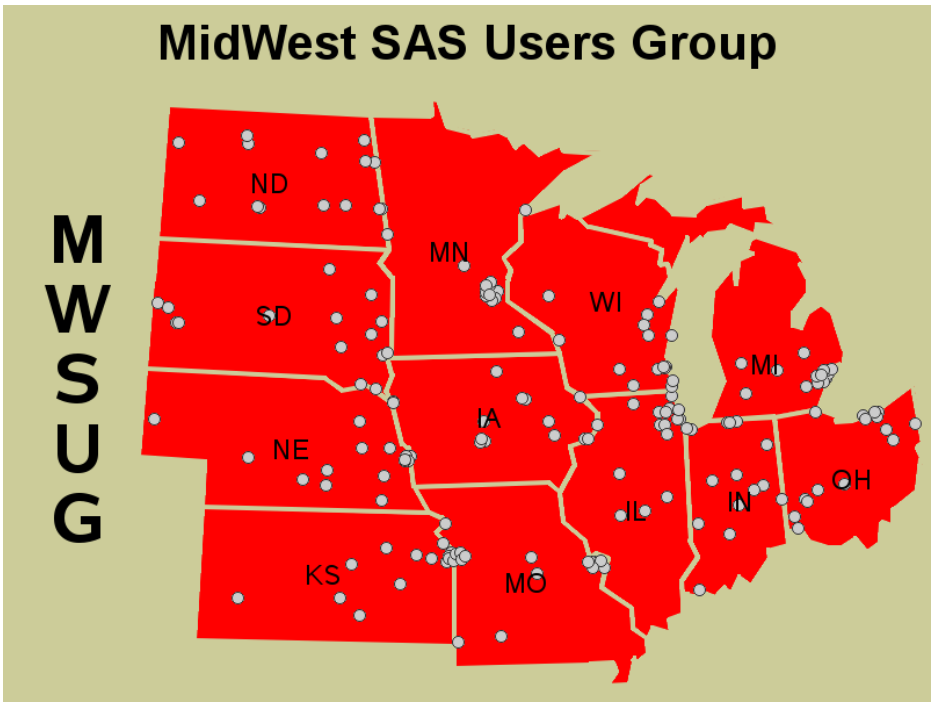
CITY	X	Y	xsys	ysys	hsys	when	function	style	rotate	size	color	text
Dallas	0.048	0.029	2	2	3	a	pie	psolid	360	1	red	
Dallas	0.048	0.029	2	2	3	a	label		.	.		Dallas
Houston	0.071	-0.023	2	2	3	a	pie	psolid	360	1	red	
Houston	0.071	-0.023	2	2	3	a	label		.	.		Houston
Lubbock	-0.026	0.042	2	2	3	a	pie	psolid	360	1	red	
Lubbock	-0.026	0.042	2	2	3	a	label		.	.		Lubbock
San Antonio	0.024	-0.030	2	2	3	a	pie	psolid	360	1	red	
San Antonio	0.024	-0.030	2	2	3	a	label		.	.		San Antonio

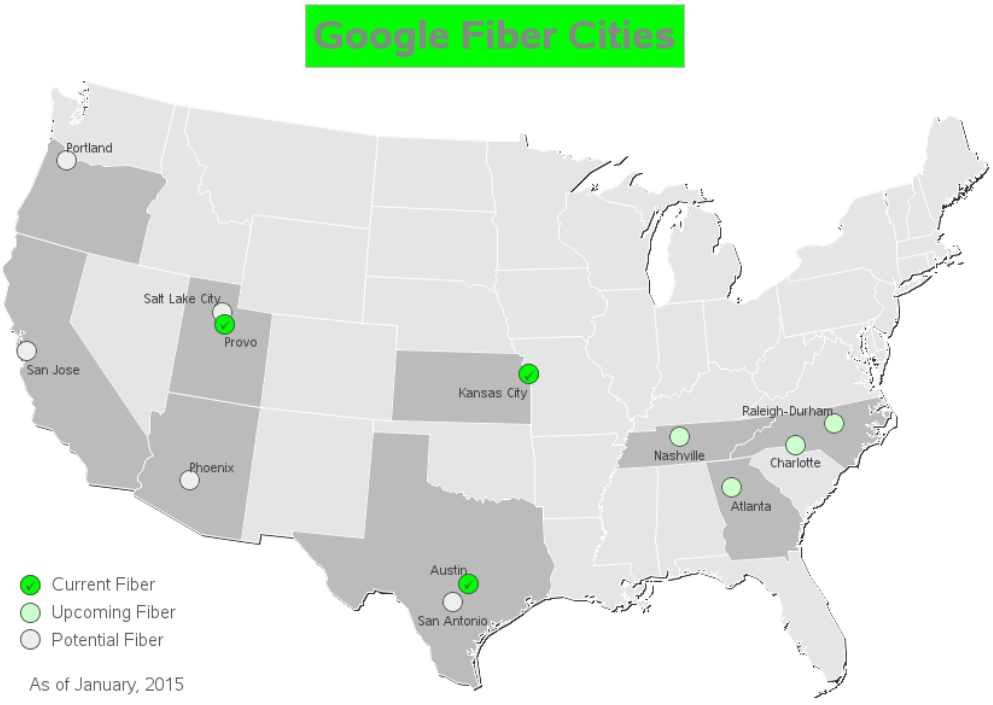
And we can create the map using the same code as before:

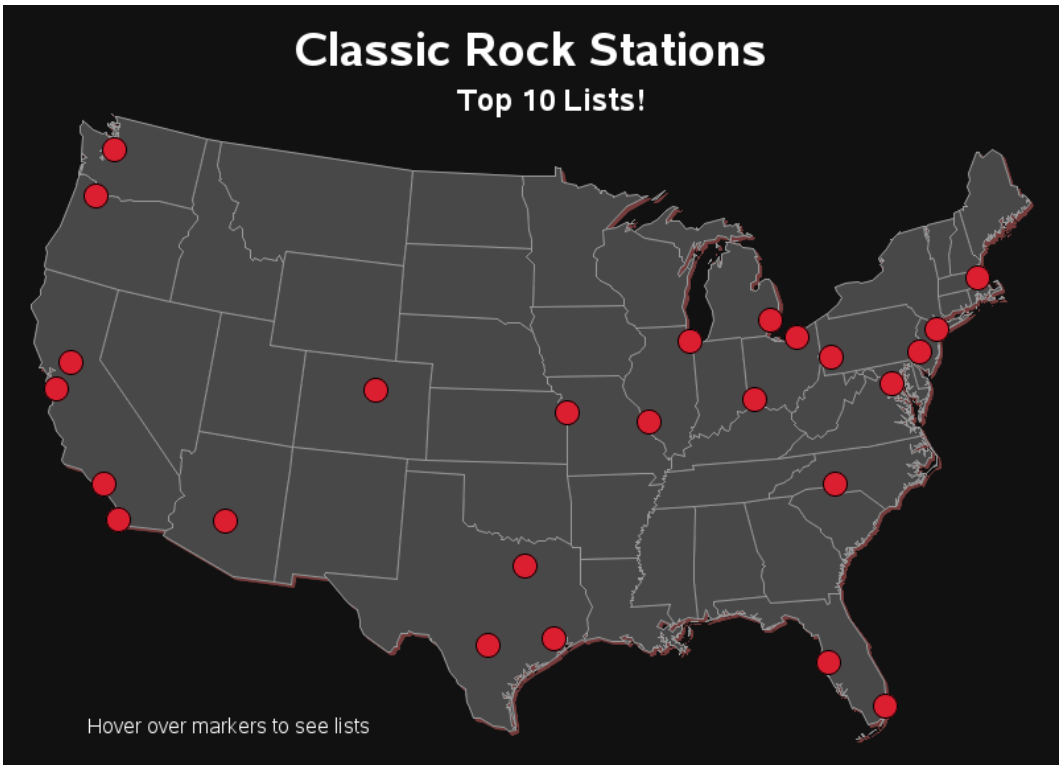
```
title1 ls=1.5 "Some Cities in Texas";
pattern1 v=s c=white;
proc gmap map=my_map data=my_map anno=anno_cities;
id statecode;
choro segment / levels=1 nolegend coutline=gray88;
run;
```

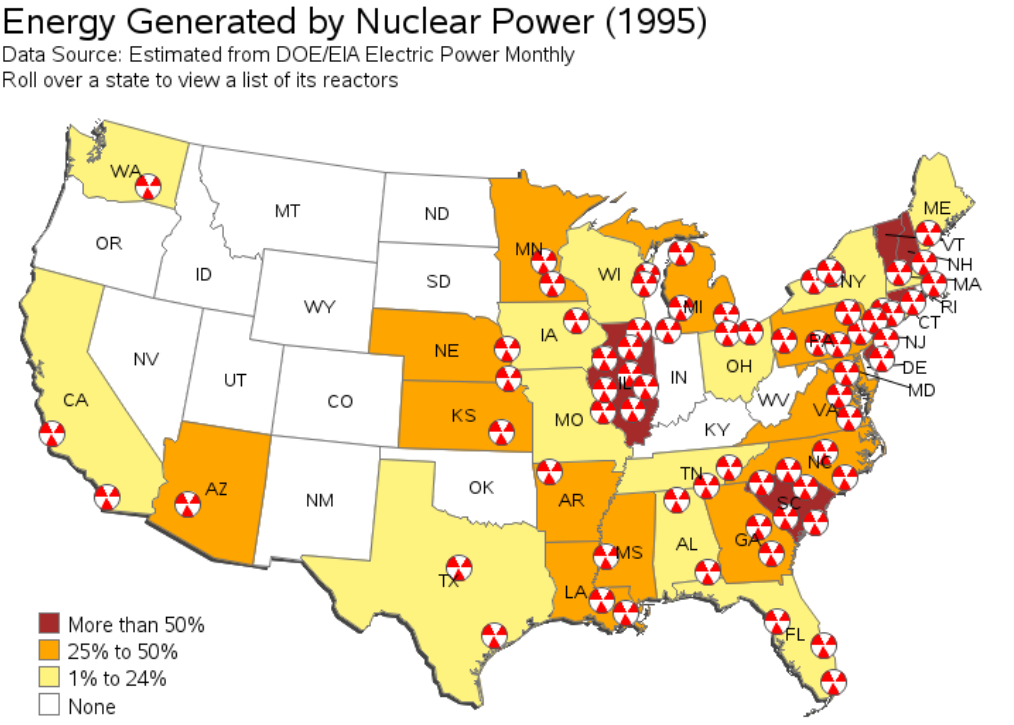

Some Cities in Texas
Lubbock
Dallas
San Antonio
Houston
Annotated Markers and Text

The following are a few examples of SAS maps with annotated markers at specific coordinates. Some are simple, and some are a bit fancier …









Imitation/Enhancement of map from PBS.org website.
Just a proof-of-concept -- this information is not guaranteed complete & accurate!

Annotating Bubble Markers

What if you want to represent some response data (such as population or sales) at the specified latitude/longitude locations? Instead of using a simple point marker, you might consider using bubbles of varying size, and maybe varying color. The technique is essentially the same as point markers (see example above), but with the size and/or color variables in the annotate dataset controlled by values in the data.

Of course, I don't want you to get bored, therefore instead of using the exact same example, let's use something new where I can also teach you a few other tricks along the way. Instead of using mapsgfk.uscity (which already has the latitude/longitude), let's just start with a list of cities and their population:

```
data anno_pies;
format population_2013 comma12.0;
input population 2013 city $ 9-50;
statecode='TX';
anno_flag=1;
datalines;
1258000 Dallas
2196000 Houston
1409000 San Antonio
 100223 Tyler
;
run;
```

anno_pies

city	statecode	population_2013	anno_flag
Dallas	TX	1,258,000	1
Houston	TX	2,196,000	1
San Antonio	TX	1,409,000	1
Tyler	TX	100,223	1

We can now use the cool (relatively new) Proc Geocode to estimate the latitude and longitude centroid of each city. Note that I used the name 'statecode' instead of 'state' in my data, so that it will match my map data. And similarly, I rename 'x' and 'y' to 'long' and 'lat' to match the variables used in the map dataset.

```
proc geocode data=anno_pies
 out=anno_pies (rename=(x=long y=lat))
 addressstatevar=statecode method=city;
run;
```

anno_pies (geocoded)

city	statecode	population_2013	anno_flag	LONG	LAT
Dallas	TX	1,258,000	1	-96.800	32.783
Houston	TX	2,196,000	1	-95.363	29.763
San Antonio	TX	1,409,000	1	-98.493	29.424
Tyler	TX	100,223	1	-95.301	32.351

Now that we have the lat/long centroids, we can combine the response data with the map, gproject them, and separate them again (similar to the previous example):

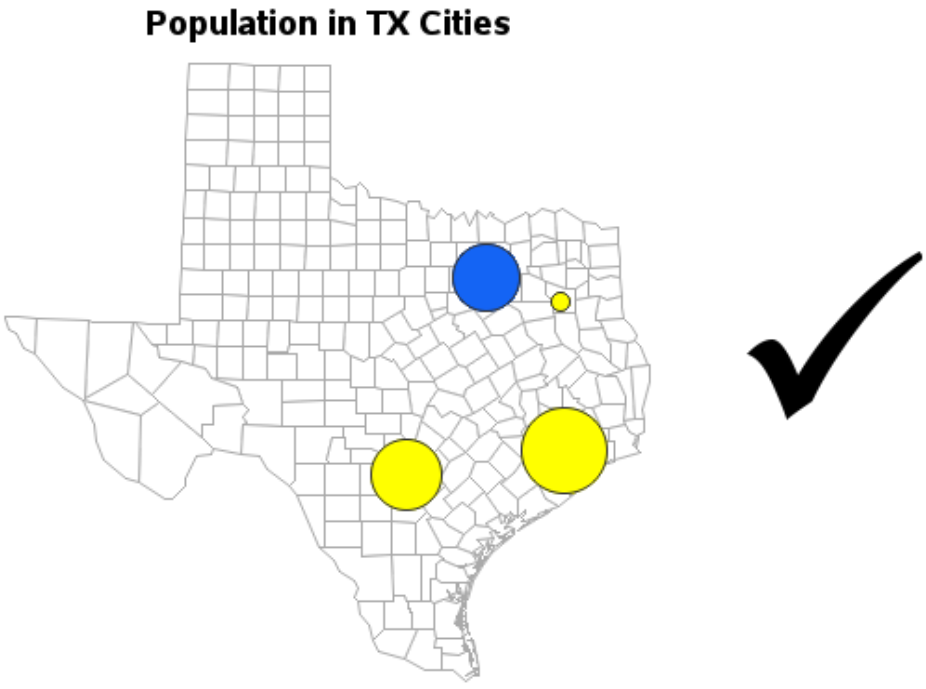
```
data my_map; set mapsgfk.us_counties
 (where=(statecode='TX' and density<=1));
run;

data combined; set my_map anno_pies; run;
proc gproject data=combined out=combined
 eastlong degrees latlong dupok;
id statecode county;
run;

data my_map anno_pies; set combined;
if anno_flag=1 then output anno_pies;
else output my_map;
run;
```

anno_pies (gprojected)

city	population_2013	anno_flag	LONG	LAT	X	Y
Dallas	1,258,000	1	-96.800	32.783	0.048	0.029
Houston	2,196,000	1	-95.363	29.763	0.072	-0.023
San Antonio	1,409,000	1	-98.493	29.424	0.024	-0.030
Tyler	100,223	1	-95.301	32.351	0.071	0.022

We now need to add a few variables to the dataset, so that it can be interpreted as annotate commands. This is the same as was explained in the previous example, except this time the size is based on the population, and the color is based on the city name. The size is the radius of the pie, and I want calculate a radius (size) that makes the area of the bubbles proportional to their population. I use the following equation (where the .007 is just a scaling factor): size=sqrt(population_2013/3.14)*.007. Note that there are two observations for each bubble – one to draw the solid filled bubble, and one to draw a gray outline around it

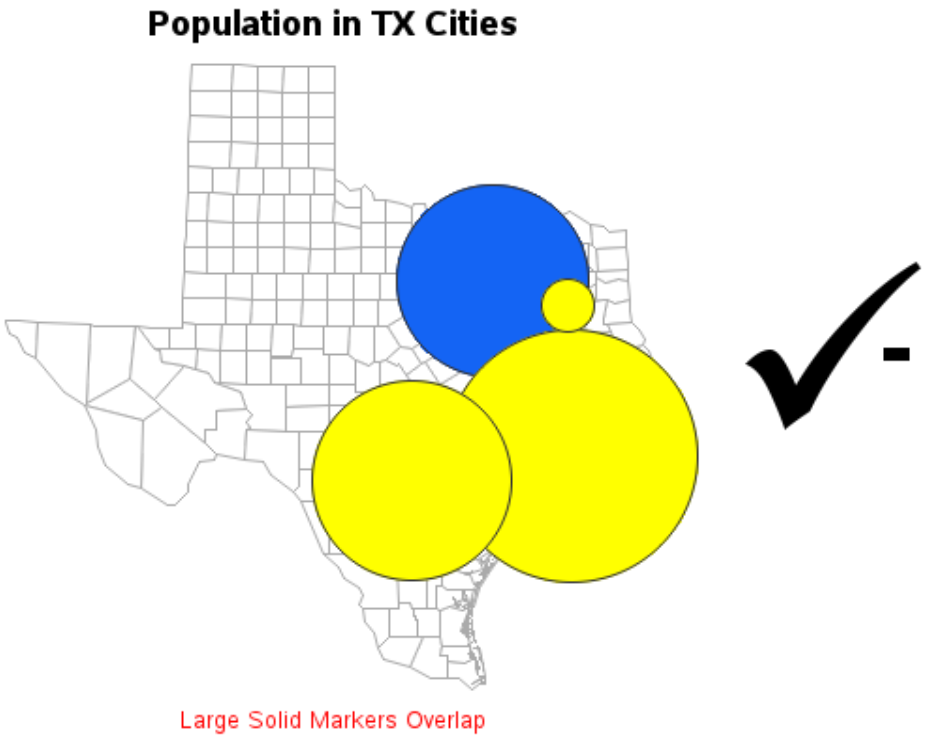
```
data anno_pies; set anno_pies;
xsys='2'; ysys='2'; hsys='3'; when='a';
length function style $8 color $20;
function='pie'; rotate=360; style='psolid';
run;

data anno_pie1; set anno_pies;
size=sqrt(population_2013/3.14)*.007;
if city='Dallas' then color='cx1464F4';
else color='cxFFFF00'; output;
style='pempty'; color='gray33'; output;
run;
```



Small Solid Color Markers

The resulting map looks pretty good – there is a bubble marker at each city, with the size is proportional to the population and the color controlled by the city (let's say, for example, we want Dallas to be blue because it's the distribution hub for our imaginary company). But what if we wanted more visual 'impact' from the bubbles, and therefore we wanted them to be bigger? That's easy – we can just increase the scaling factor (currently .007) in the size equation: size=sqrt(population_2013/3.14)***.007**. Let's increase it to .02 …

```
data anno_pie2; set anno_pies;
size=sqrt(population_2013/3.14)*.02;
if city='Dallas' then color='cx1464F4';
else color='cxFFFF00'; output;
style='pempty'; color='gray33'; output;
run;
```

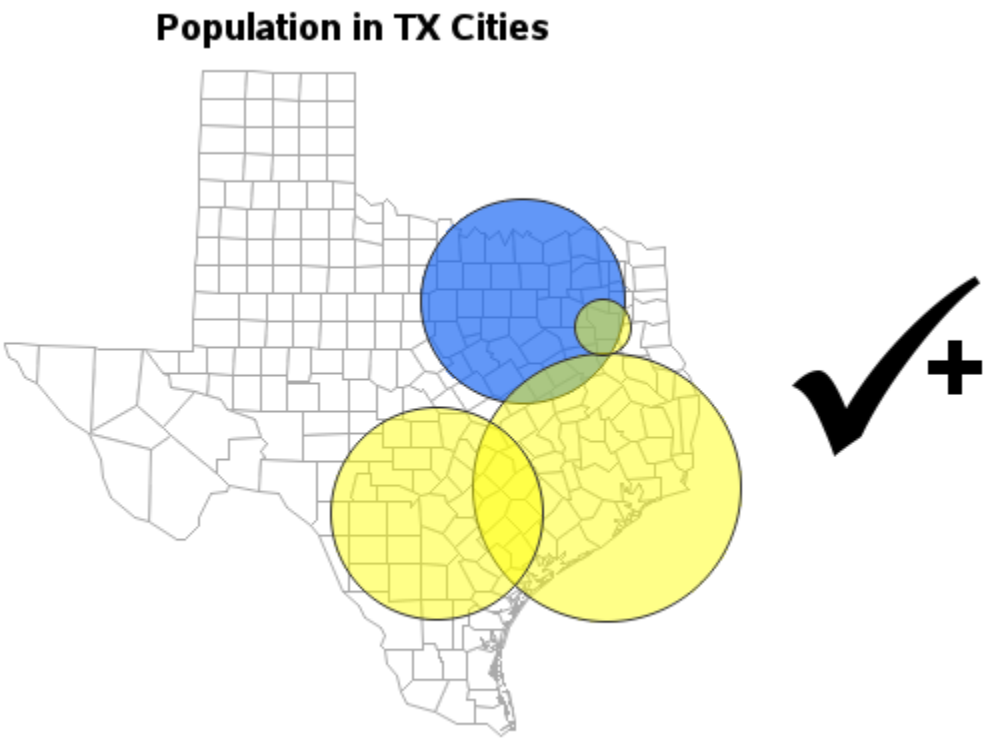


Large Solid Markers Overlap

Larger bubbles can be problematic though – they obscure the map, and could even obscure other bubbles. In situations like this, my favorite work-around is to use transparent colors to fill the bubbles. Whereas you specify a regular RGB color using cxRRGGBB, you specify a transparent color as aRRGGBBnn (where nn is a hexadecimal number that controls the amount of transparency, with a low nn being more transparent, and a high nn being more solid). In this case, instead of using cxFFFF00 (yellow), simply use aFFFF0077 (transparent yellow), then you can better see the state borders and overlapping bubbles:

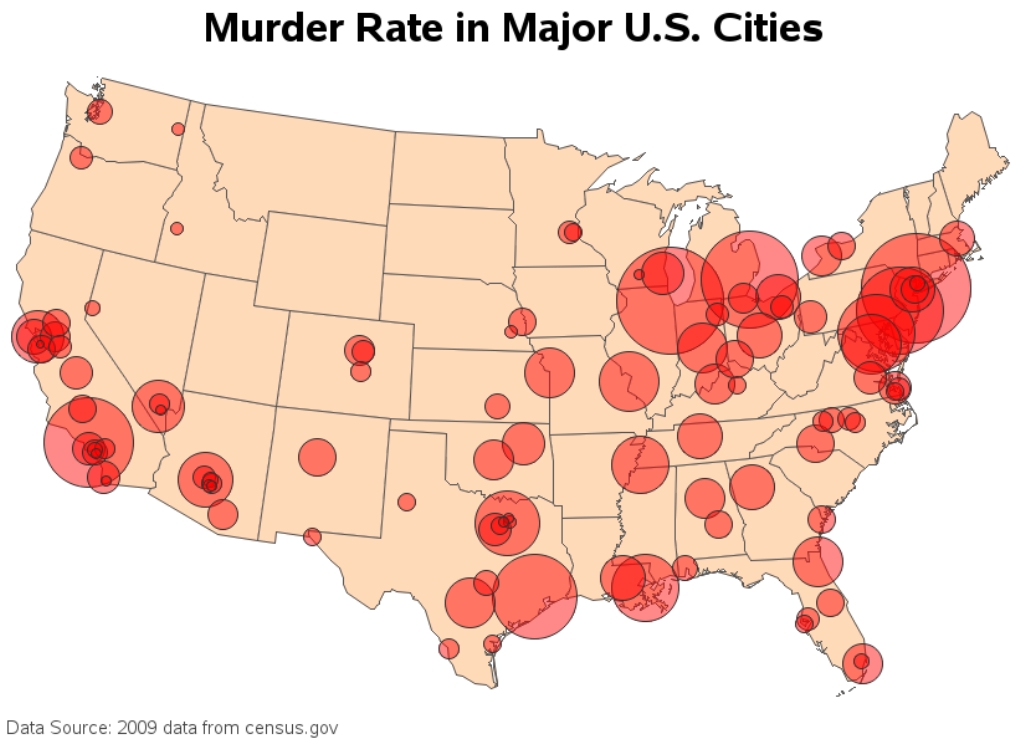
```
data anno_pie3; set anno_pies;
size=sqrt(population_2013/3.14)*.02;
if city='Dallas' then color='a1464F4aa';
else color='aFFFF0077';
output;
style='pempty'; color='gray33'; output;
run;
```

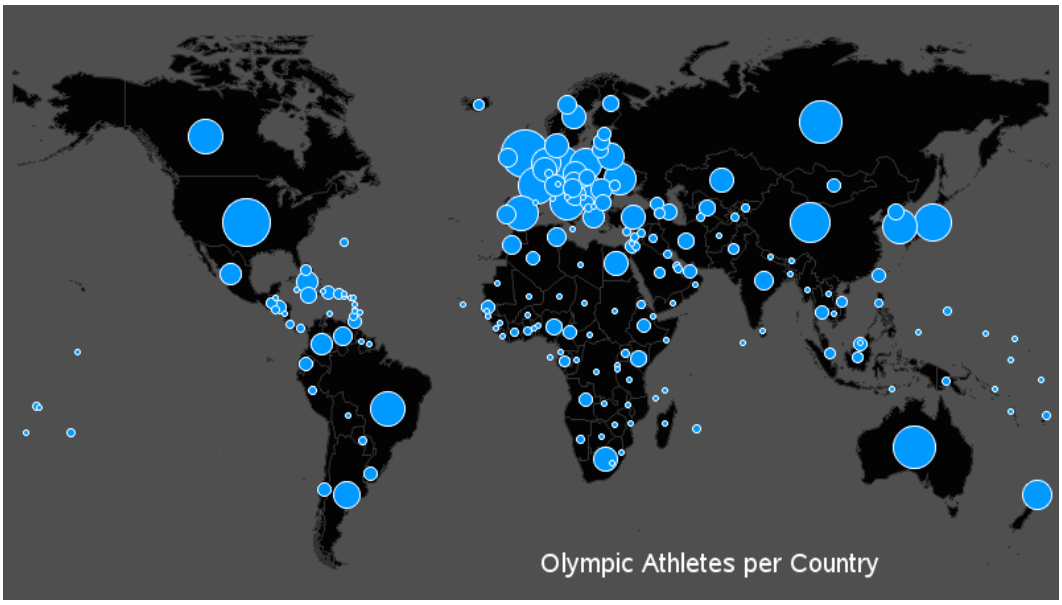
And there you have it – large bubbles, that don't obscure the map or the other bubbles!

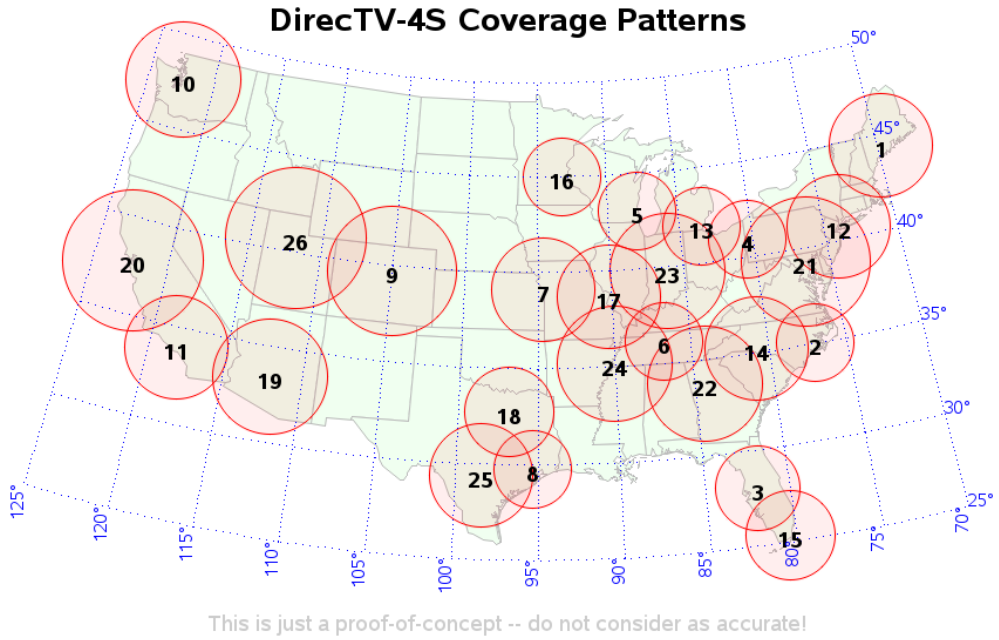


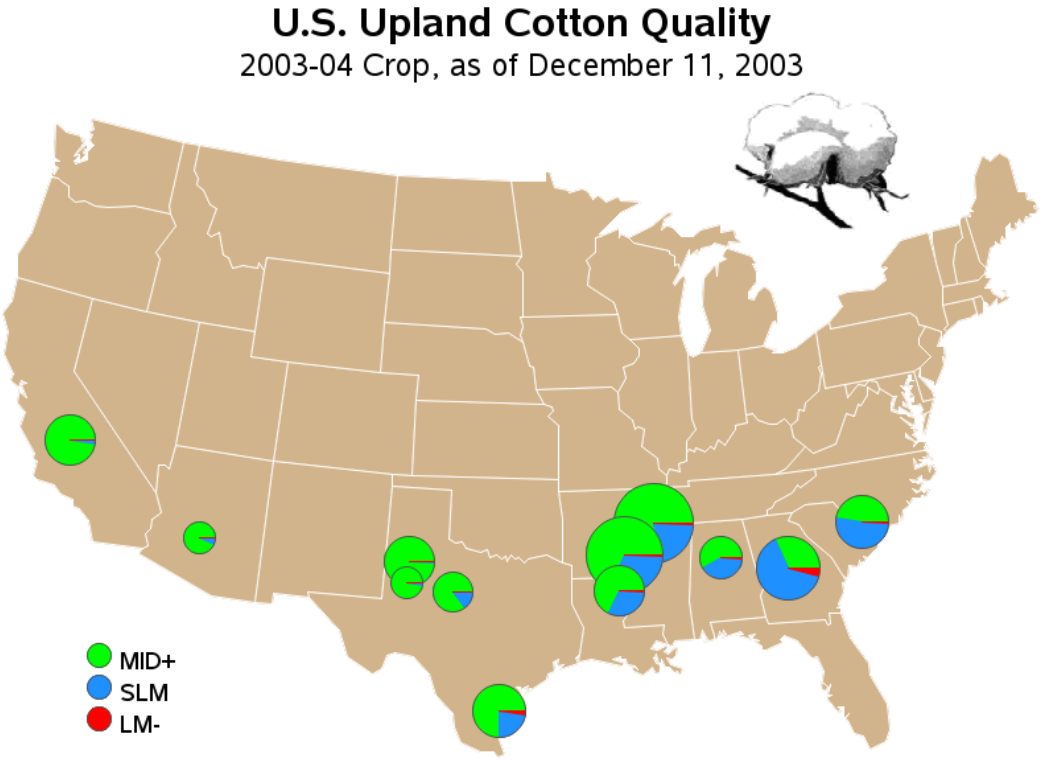
Transparent Markers

The following are some real-world examples of maps with bubble markers, of varying sizes, annotated at specific locations on maps …






DirecTV-4S Coverage Patterns
10
26
20
9
16
5
13
4
12
1
21
23
7
17
11
19
6
14
2
24
22
18
25
8
3
15
50°
45°
40°
35°
30°
25°
125°
120°
115°
110°
105°
100°
95°
90°
85°
80°
75°
70°
This is just a proof-of-concept -- do not consider as accurate!


U.S. Upland Cotton Quality
2003-04 Crop, as of December 11, 2003
MID+
SLM
LM-

Annotating Bubbles at Map Polygon Centroids

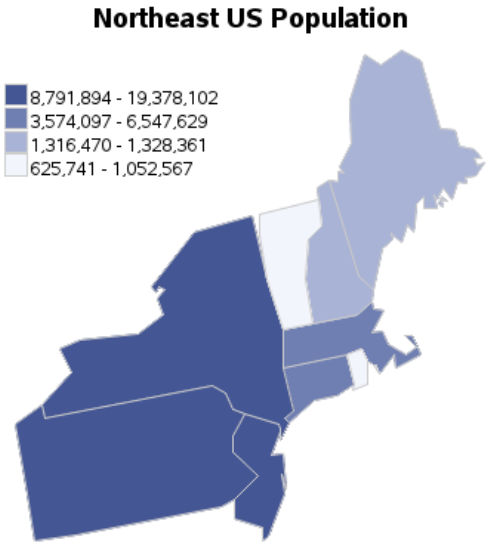
With a traditional **choropleth** map, you can represent the amount (or magnitude) of the data values with the colors or shades of the areas in the map. For example, let's create a map showing the population of the states in the northeast US, from the sashelp.us_data dataset. Below is the code that subsets the data, and plots it on a simple choro map:

```
data my_data;
set sashelp.us_data (where=(region='Northeast'));
run;
```

my_data

STATECODE	POPULATION_2010
CT	3,574,097
ME	1,328,361
MA	6,547,629
NH	1,316,470

```
title1 ls=1.5 "Northeast US Population";
legend1 label=none across=1 position=(top left) mode=share
 order=descending shape=bar(.15in,.15in) offset=(8,-8);
proc gmap map=maps.us data=my_data;
id statecode;
choro population_2010 / coutline=graycc legend=legend1;
run;
```



Simple Choropleth Map

A choro map is often a good way to plot data on a map, but there are several limitations. Since not all map areas are the same size, it is often difficult to easily compare the values. For example, what if a very small state has a very large population? … The state would be in the brightest color, but it wouldn't make a very big visual impact because of the small size of the state. One alternative that makes it easier to compare the values is to use bubbles to represent the data values. Then the varying physical sizes of the states are not an issue, because you're visually comparing the sizes of the bubbles.

Unlike the gmap block map (which plots a bar in the middle of each map polygon – in this case states), there is no built-in option to plot a bubble in the middle of each state. We have to calculate the x/y location where we want the bubble, and then annotate a pie (bubble). The following code subsets the map to get just the states we want, and then uses the SAS %centroid macro to estimate the x/y center of each state:

```
data my_map; set maps.us (where=(statecode in
 ('CT' 'ME' 'MA' 'NH' 'NJ' 'NY' 'PA' 'RI' 'VT')));
run;

%annomac;
%centroid(my_map,state_centroids,state);
```

state_centroids

STATE	x	y
9	0.29835	0.11580
23	0.32954	0.18871
25	0.30370	0.13298
33	0.30226	0.16135

Now that we've got the centroid for each state, let's merge that data with our response data (in this case the population), and then we'll have the basis for our annotate dataset (anno_bubbles). There are several ways to merge data in SAS – my favorite is using SQL. Note that you don't need to have the numeric state fips code (state) or the 2-character state abbreviation (statecode), but I like to include such variables because they make it easier to verify the results …

```
proc sql;
create table anno_bubbles as
select unique state_centroids.*,
 my_data.statecode, my_data.population_2010
from state_centroids left join my_data
on state_centroids.state=my_data.state
```

```
order by population_2010 descending;
quit; run;
```

anno_bubbles

STATE	x	y	STATECODE	POPULATION_2010
36	0.26377	0.13596	NY	19,378,102
42	0.23706	0.09073	PA	12,702,379
34	0.28237	0.08632	NJ	8,791,894
25	0.30370	0.13298	MA	6,547,629
9	0.29835	0.11580	CT	3,574,097

Now it's just a simple matter of adding a few more variables to the dataset so that it can be used as an annotate dataset to draw the bubbles. The variables are pretty much the same as the ones in the previous example. Here's the data step code, followed by the first few observations of the dataset:

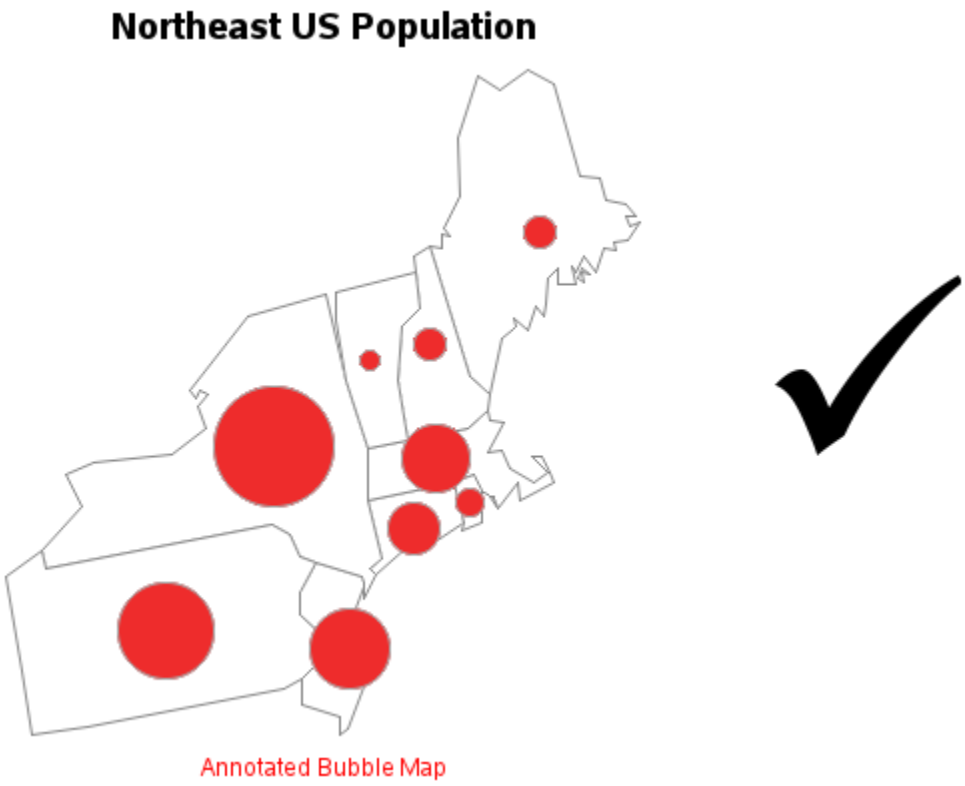
```
data anno_bubbles; set anno_bubbles;
length color $9;
xsys='2'; ysys='2'; hsys='3'; when='a';
function='pie'; rotate=360;
size=sqrt(population_2010/3.14)*.003;
style='psolid'; color='cxEE2C2C'; output;
style='pempty'; color='grayaa'; output;
run;
```

anno_bubbles

x	y	xsys	ysys	hsys	when	function	rotate	size	style	color
0.26377	0.13596	2	2	3	a	pie	360	7.45267	psolid	cxEE2C2C
0.26377	0.13596	2	2	3	a	pie	360	7.45267	pempty	grayaa
0.23706	0.09073	2	2	3	a	pie	360	6.03391	psolid	cxEE2C2C
0.23706	0.09073	2	2	3	a	pie	360	6.03391	pempty	grayaa
0.28237	0.08632	2	2	3	a	pie	360	5.01993	psolid	cxEE2C2C
0.28237	0.08632	2	2	3	a	pie	360	5.01993	pempty	grayaa

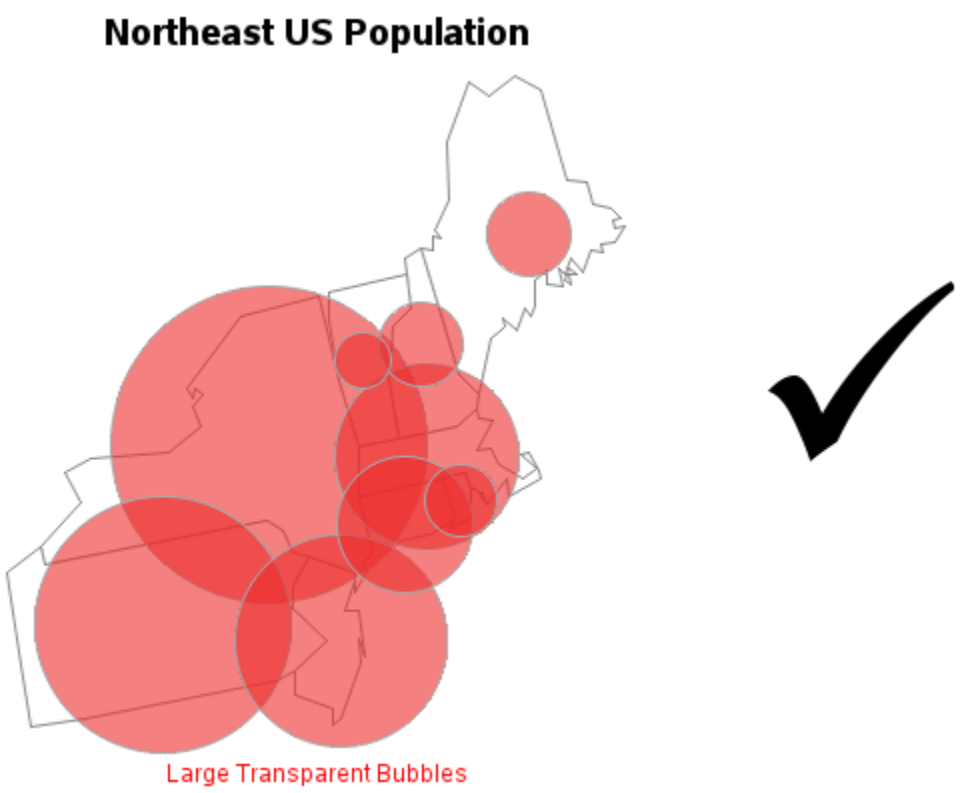
And now it's a simple matter of plotting a blank choro map, and annotating the bubble markers on it!

```
title1 ls=1.5 "Northeast US Population";
pattern1 v=s c=white;
proc gmap map=my_map data=my_map anno=anno_bubbles;
id statecode;
choro segment / levels=1 nolegend coutline=gray88;
run;
```

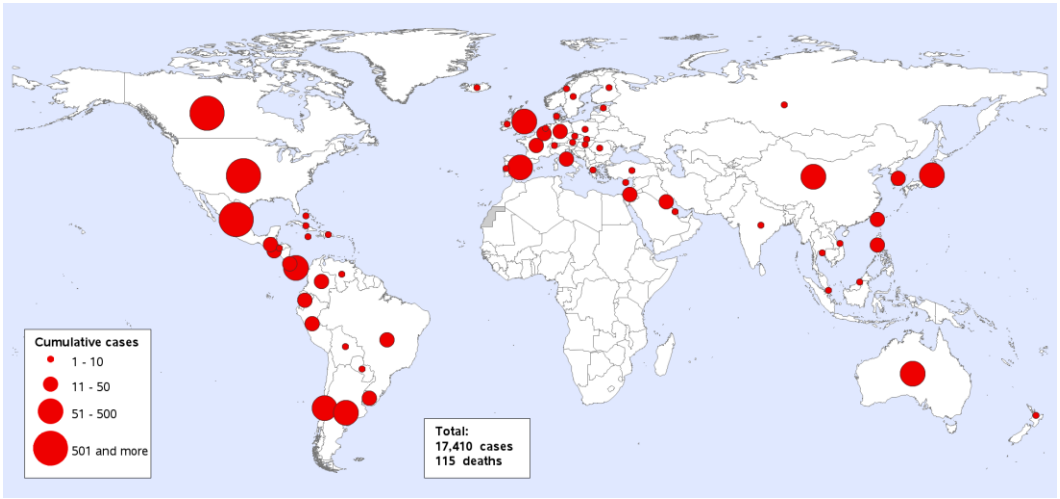


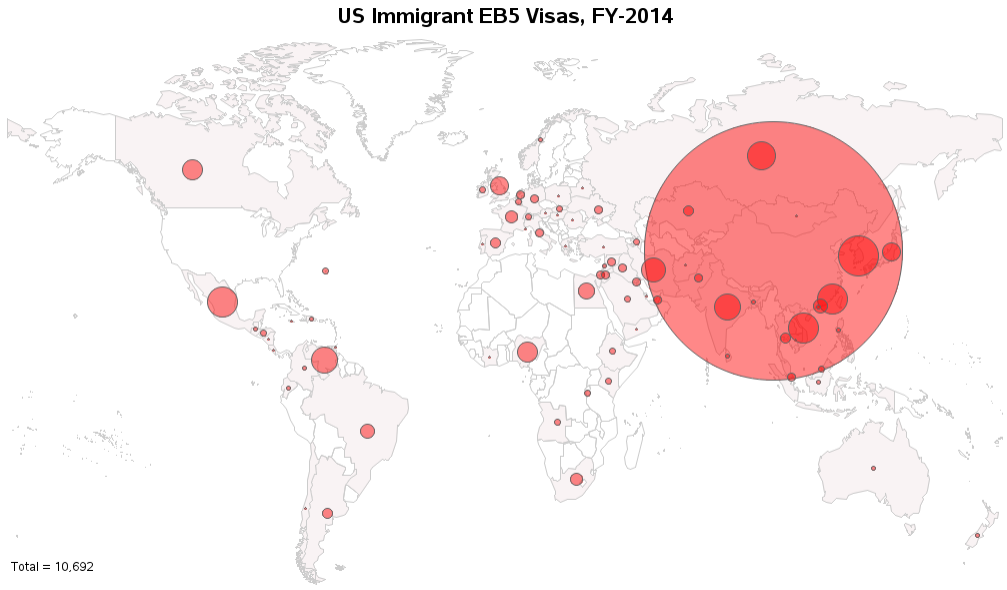
Annotated Bubble Map

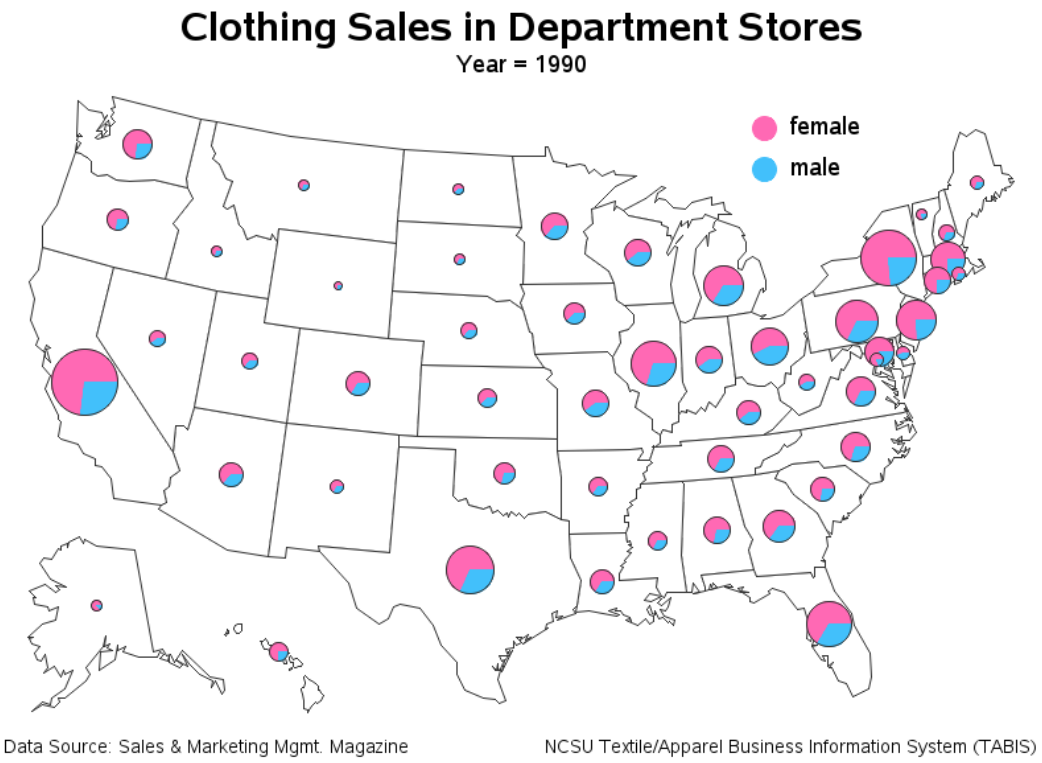
And, similar to the previous example, you can use bigger bubbles (multiply the size by .008 instead of .003), and a transparent color (aEE2C2C99 instead of cxEE2C2C) to get a slightly different look …



Here are a couple of real-world examples, where markers were annotated at the centroids of the map areas …




US Immigrant EB5 Visas, FY-2014
Total = 10,692


Clothing Sales in Department Stores
Year = 1990
female
male
Data Source: Sales & Marketing Mgmt. Magazine
NCSU Textile/Apparel Business Information System (TABIS)

Annotating Paths

Sometimes you don't want to annotate individual markers on a map, but rather a series of connected locations. Perhaps you want a line representing locations where a person, vehicle, or animal have been. Or perhaps you're plotting a highway or railroad tracks on a map. This is another situation where annotate comes in handy.

All you need is some latitude/longitude points along the path, and then you can connect those points with an annotated line.

For this example, I'll use a simplified version of the proposed Atlantic Coast Pipeline route. I'm only including a few of the data points here, to make it easier for you to copy-n-paste (if you download my code from the samples page, you'll get extra points along the route).

```
data anno_route;
input lat long;
flag=1;
datalines;
39.17 -80.57
39.15 -80.55
39.14 -80.45
39.10 -80.41
39.02 -80.32
38.83 -80.11
38.50 -79.68
38.27 -79.41
38.21 -79.16
37.91 -78.80
37.41 -78.42
37.13 -77.96
36.92 -77.86
36.47 -77.55
35.62 -78.12
35.19 -78.68
34.74 -79.19
;
run;
```

Here's what the first few lines of data look like:

anno_route (unprojected)

lat	long	flag
39.17	-80.57	1
39.15	-80.55	1
39.14	-80.45	1
39.10	-80.41	1
39.02	-80.32	1

Now you'll want to get the map, combine it with the point data, and then project them. In this case, since the pipeline will be going through three states, we'll want those three states (NC, VA, and WV) in our map. Let's subset the map based on the density values, so that fewer points will be used, and the borders will look slightly less busy.

```
data my_map; set mapsgfk.us_counties
 (where=(statecode in ('NC' 'VA' 'WV')
   and density<=2) drop=resolution);
run;

data combined; set my_map anno_route;
run;
proc gproject data=combined out=combined
 latlong degrees eastlong dupok;
id statecode county;
run;
data my_map anno_route; set combined;
if flag=1 then output anno_route;
else output my_map;
run;
```

Now that the lat/long coordinates in our map and our annotated route have been projected into X/Y coordinates that will look good on a flat page, let's add some annotate commands to tell SAS what to do with the X/Y points in the annotated route dataset. First, so you can 'see' where the points will show up on the map, let's plot them as red dots, using the annotate pie function.

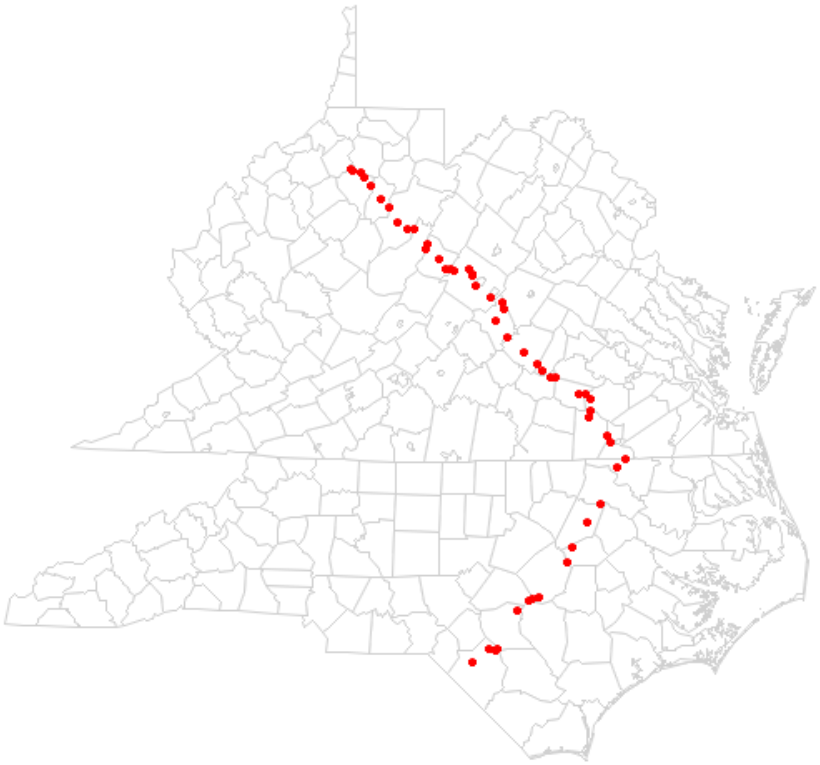
```
data anno_route; set anno_route;
length function $8 color $8 style $35;
xsys='2'; ysys='2'; hsys='3'; when='a';
function='pie'; style='psolid'; rotate=360;
color='red'; size=.5;
run;
```

anno_route red dots

X	Y	xsys	ysys	hsys	function	color	style	rotate	size	when
-0.010664	0.033788	2	2	3	pie	red	psolid	360	0.5	a
-0.010396	0.033437	2	2	3	pie	red	psolid	360	0.5	a
-0.009044	0.033252	2	2	3	pie	red	psolid	360	0.5	a
-0.008507	0.032550	2	2	3	pie	red	psolid	360	0.5	a
-0.007296	0.031147	2	2	3	pie	red	psolid	360	0.5	a

And now we can easily plot a simple/blank map, and specify the above data using the anno= option, and get the following map:

```
pattern1 v=s c=white;
proc gmap data=my_map map=my_map all anno=anno_route;
id statecode county;
choro segment / levels=1 nolegend coutline=grayd0;
run;
```



Points along the path

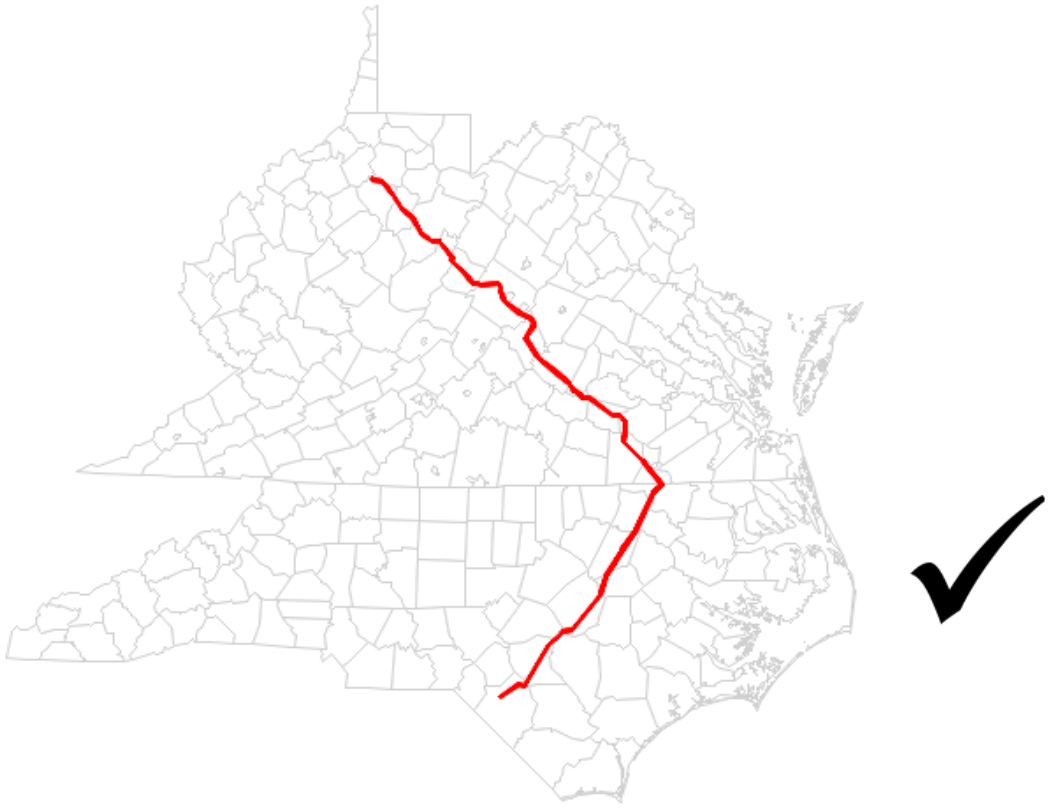
But what we really want to do is connect the points with a line, therefore let's modify the annotate dataset and use the annotate **move** and **draw** functions, rather than the pie function. To connect the points with a line, you'll want to move to the first observation (which is at the beginning of the route), and then you'll want to draw to each of the following points.

```
data anno_route; set anno_route;
length function $8 color $8;
xsys='2'; ysys='2'; hsys='3'; when='a';
if _n_=1 then function='move';
else function='draw';
color='red'; size=.5; style=''; rotate=.;
run;
```

anno_route line

X	Y	xsys	ysys	hsys	function	color	size	when
-0.010664	0.033788	2	2	3	move	red	0.5	a
-0.010396	0.033437	2	2	3	draw	red	0.5	a
-0.009044	0.033252	2	2	3	draw	red	0.5	a
-0.008507	0.032550	2	2	3	draw	red	0.5	a
-0.007296	0.031147	2	2	3	draw	red	0.5	a

```
proc gmap data=my_map map=my_map all anno=anno_route;
id statecode county;
choro segment / levels=1 nolegend coutline=grayd0;
run;
```



Connect points with line

And now that we've got the line annotated on the map, let's annotate some title text (we're annotating, rather than using the built-in title statements, so that the titles can share the map space instead of being at the top of the page).

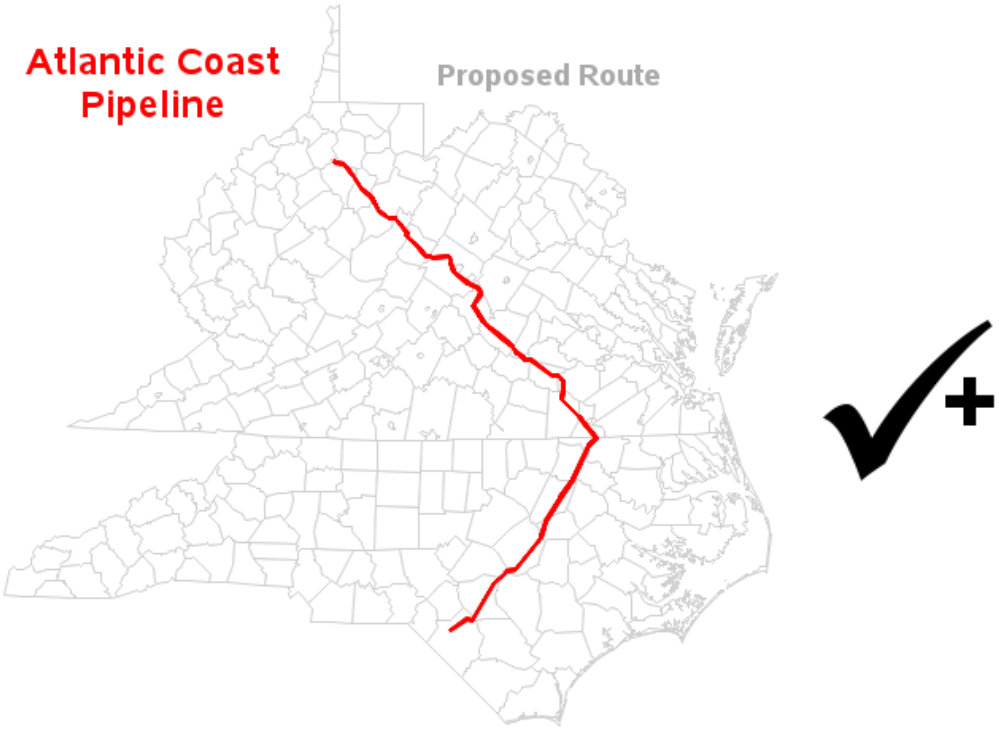
```
data anno_text;
length function $8 color $8 style $35 text $100;
xsys='3'; ysys='3'; hsys='3'; when='a';
function='label'; position='5'; style='albany amt/bold';

color="red"; size=4.7;
x=20; y=92; text="Atlantic Coast"; output;
y=y-5.5; text="Pipeline"; output;

color="grayaa"; size=3.5;
x=70; y=90; text="Proposed Route"; output;

run;
```

anno_text

function	color	style	text	xsys	ysys	hsys	when	position	size	x	y
label	red	albany amt/bold	Atlantic Coast	3	3	3	a	5	4.7	20	92.0
label	red	albany amt/bold	Pipeline	3	3	3	a	5	4.7	20	86.5
label	grayaa	albany amt/bold	Proposed Route	3	3	3	a	5	3.5	70	90.0

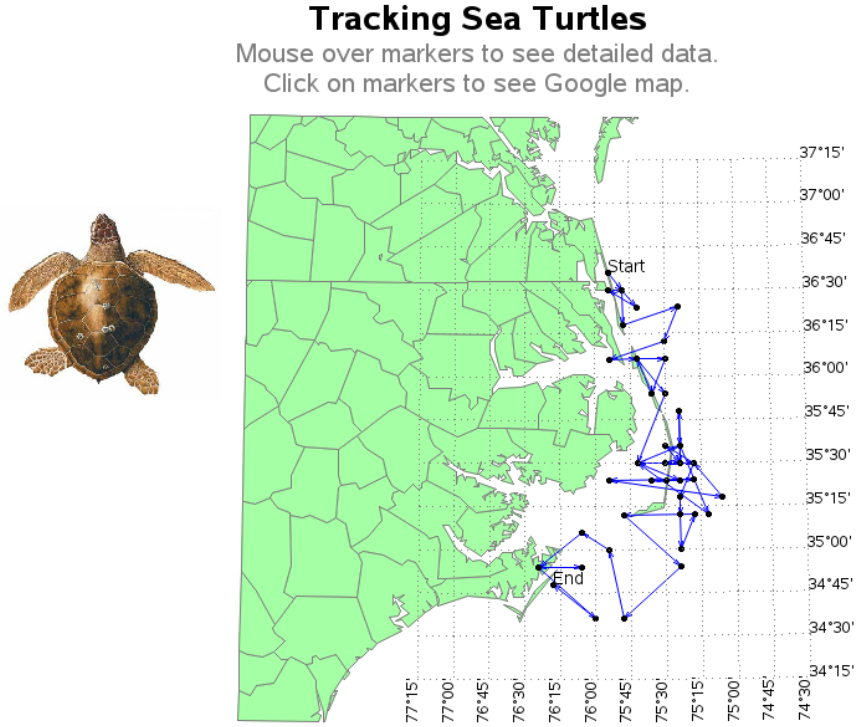
And now we can produce the final map! …

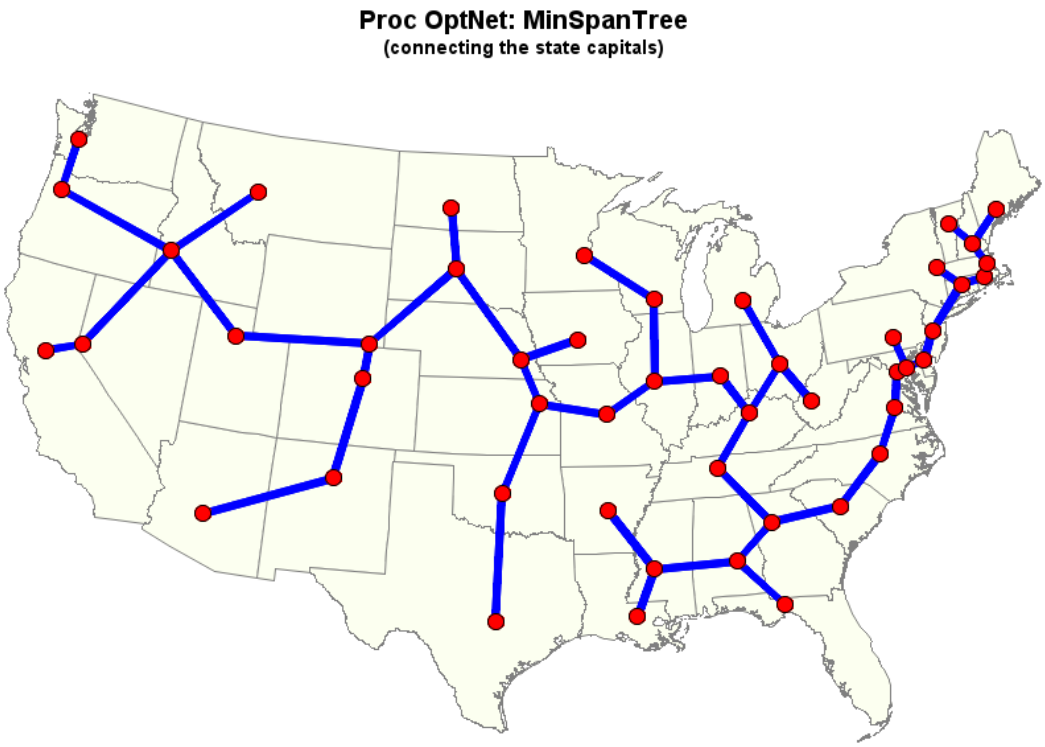
```
proc gmap data=my_map map=my_map all anno=anno_route;
id statecode county;
choro segment / levels=1 nolegend coutline=grayd0
 anno=anno_text;
run;
```

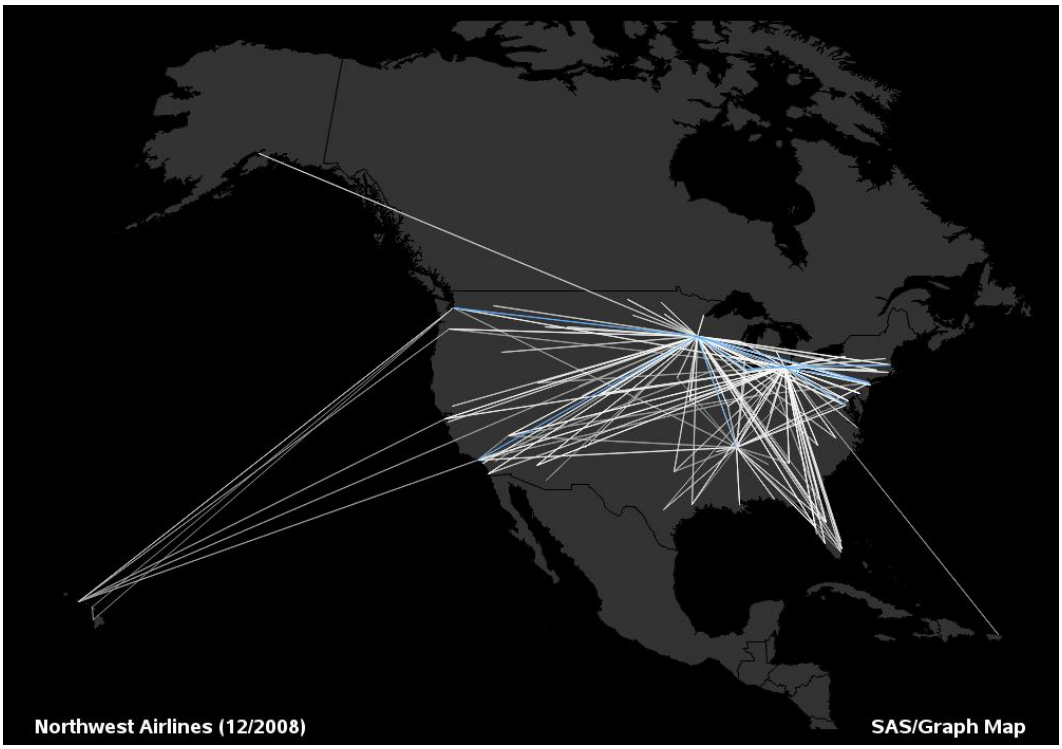


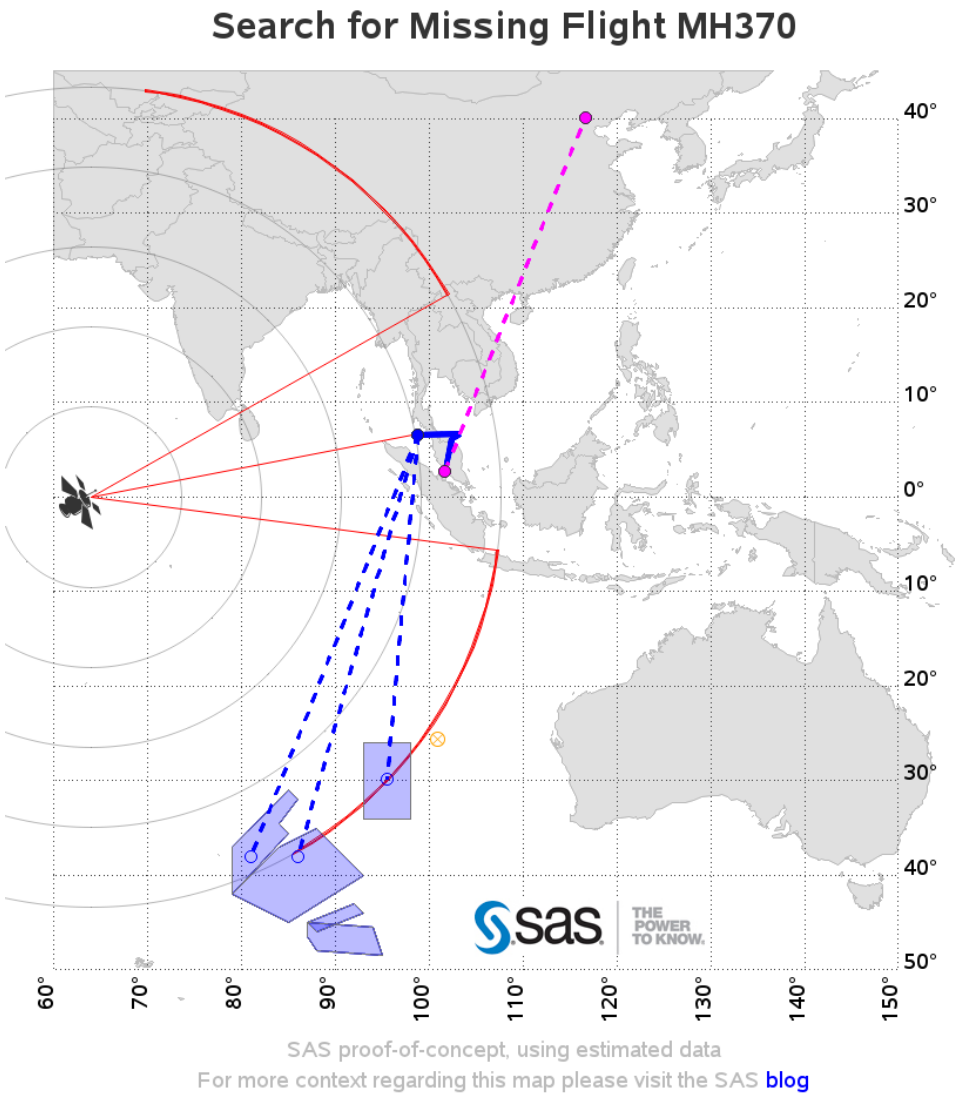
Add annotated titles

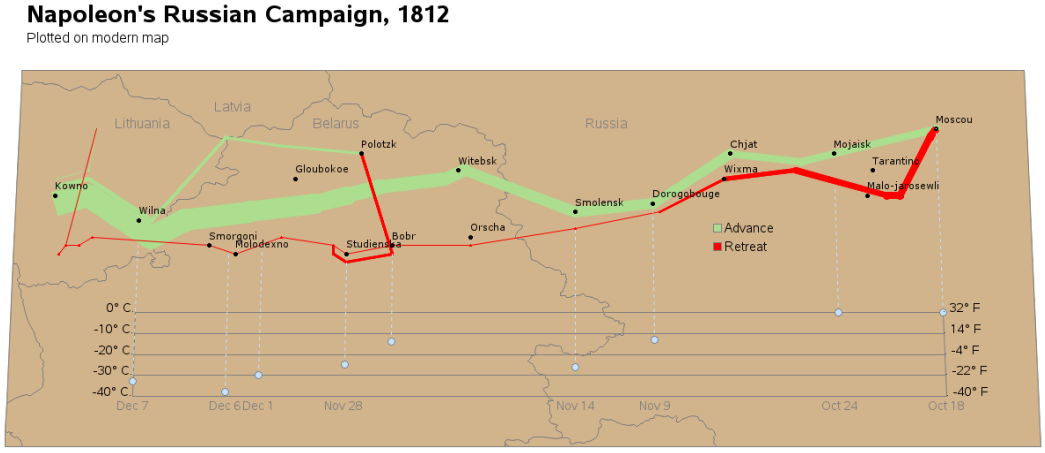
And now, here are a few real-world examples of paths annotated on a map …




Proc OptNet: MinSpanTree
(connecting the state capitals)


Northwest Airlines (12/2008)
SAS/Graph Map


Search for Missing Flight MH370
40°
30°
20°
10°
0°
10°
20°
30°
40°
50°
60°
70°
80°
90°
100°
110°
120°
130°
140°
150°
sas
THE POWER TO KNOW.
SAS proof-of-concept, using estimated data
For more context regarding this map please visit the SAS blog

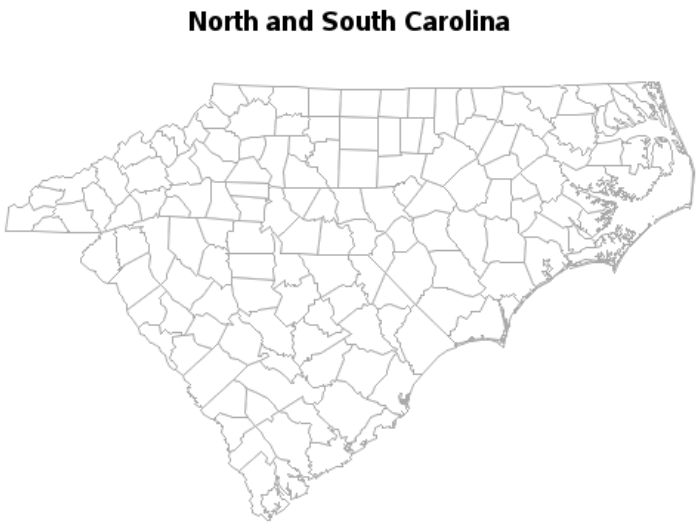

Napoleon's Russian Campaign, 1812
Plotted on modern map
Latvia
Lithuania
Belarus
Russia
Kowno
Wilna
Smorgoni
Molodexno
Polotzk
Gloubokoe
Studienska
Bobr
Witebsk
Orscha
Smolensk
Dorogobouge
Chjat
Wixma
Mojaisk
Tarantino
Malo-jaroswli
Moscou
Advance
Retreat
0° C
-10° C
-20° C
-30° C
-40° C
32° F
14° F
-4° F
-22° F
-40° F
Dec 7
Dec 6
Dec 1
Nov 28
Nov 14
Nov 9
Oct 24
Oct 18

Annotating Borders

SAS/Graph Proc GMap is only designed to represent 1 level of borders on a map. For example, if you're plotting a county map, GMap only shows the county borders (not the state borders). Therefore if you want to show extra levels of borders, you'll need to do a little extra work. As with most customizations, we can use annotate to add the extra borders.

Let's start with a county map of North and South Carolina. Since we're subsetting this out of the US map, we'll need to project it (otherwise it will look 'crooked' since it was projected based on the center of the US rather than the center of the Carolinas).

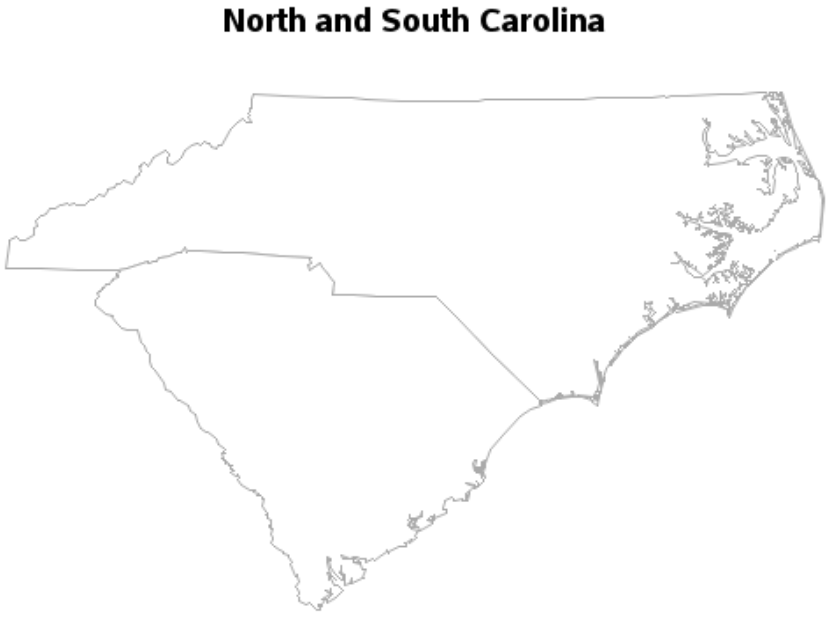
```
data my_map; set mapsgfk.us_counties (where=(statecode in
('NC' 'SC') and density<=2) drop=resolution);
run;

proc gproject data=my_map out=my_map
 latlong eastlong degrees;
id state county;
run;

title1 ls=1.5 "North and South Carolina";
pattern1 v=s c=white;
proc gmap map=my_map data=my_map;
id state county;
choro segment / levels=1 nolegend coutline=grayaa;
run;
```



Simple Choropleth Map of Counties

There are two states in this map, but it is difficult to quickly tell where one stops and the other starts, without the state borders shown. The first step to annotating the state borders is to create an actual map of the state borders, using Proc GRemove. We could hypothetically use the state map, but we would have to take special care to project it in exactly the same way the county map was projected … and even then there's no guarantee that the state map would be showing the *exact* same areas as the county map (especially when it comes to showing river & coastal borders, islands, etc). Therefore I always like to use the exact same map, and GRemove the unwanted internal borders – this guarantees the borders in the state map will match the county map! (You don't really need to plot this intermediate map, but I'm plotting it so you can see that it is indeed a map dataset.)

```
proc gremove data=my_map out=anno_outline;
by state notsorted;
id county;
run;

proc gmap map=anno_outline data=anno_outline;
id state;
choro segment / levels=1 nolegend coutline=grayaa;
run;
```



Proc Gremove Internal Boundaries

And now the final step is to convert the state outline map dataset into an annotate dataset that will simply draw the state outlines. The first observation of each state will become the annotate 'poly' function, and all subsequent observations will be 'polycont'. Poly tells annotate to start drawing a new polygon, and polycont tells annotate to continue drawing the polygon. When annotate encounters the last polycont observation, it then connects that point back to the first one. You might be asking why we didn't use "by state" – we could do that in most cases, but since some states could have multiple segments, it's best to use "by state segment". This code can be re-used with most any map that has a compound id, by simply changing the 'state' and 'county' variables to the variables used in your map.

```
data anno_outline; set anno_outline;
by state segment notsorted;
length function $8 color $8;
color='gray33'; style='mempty'; when='a'; xsys='2';
ysys='2';
if first.segment then function='poly';
else function='polycont';
run;
```

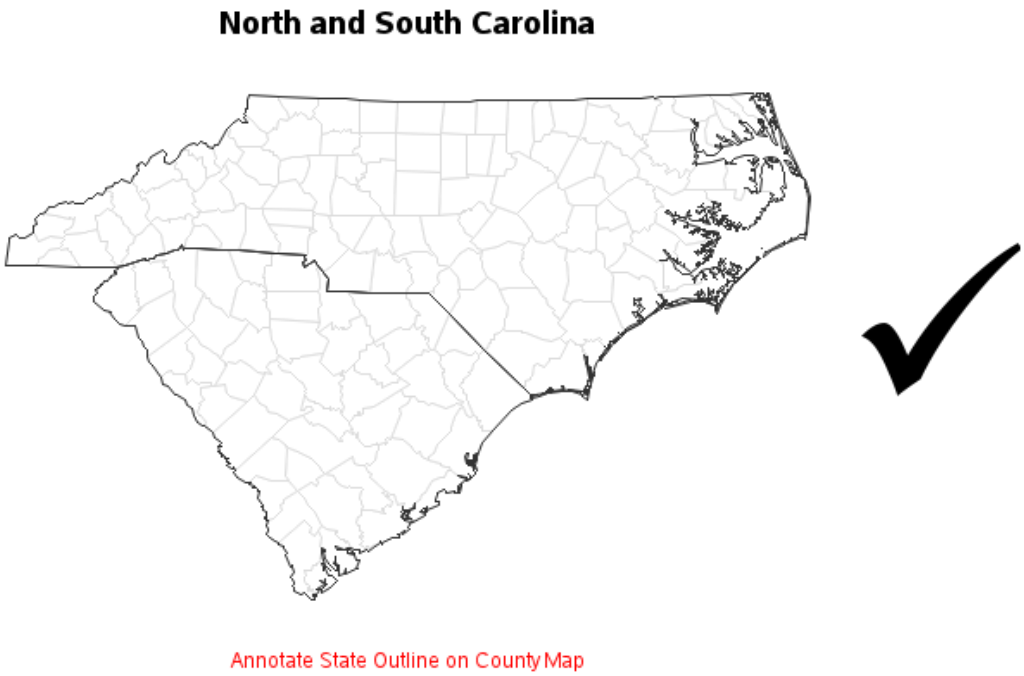
Here are the first few observations in the anno_outline dataset (notice that I'm making the annotated state outline a little darker than the county borders):

anno_outline

xsys	ysys	X	Y	function	color	style	when
2	2	-0.063323	0.013196	poly	gray33	mempty	a
2	2	-0.060575	0.013064	polycont	gray33	mempty	a
2	2	-0.058803	0.012985	polycont	gray33	mempty	a
2	2	-0.057821	0.012951	polycont	gray33	mempty	a
2	2	-0.052285	0.012828	polycont	gray33	mempty	a

And now we can annotate the state outlines on the county map, using the following code:

```
proc gmap map=my_map data=my_map anno=anno_outline;
id state county;
choro segment / levels=1 nolegend coutline=graydd;
run;
```

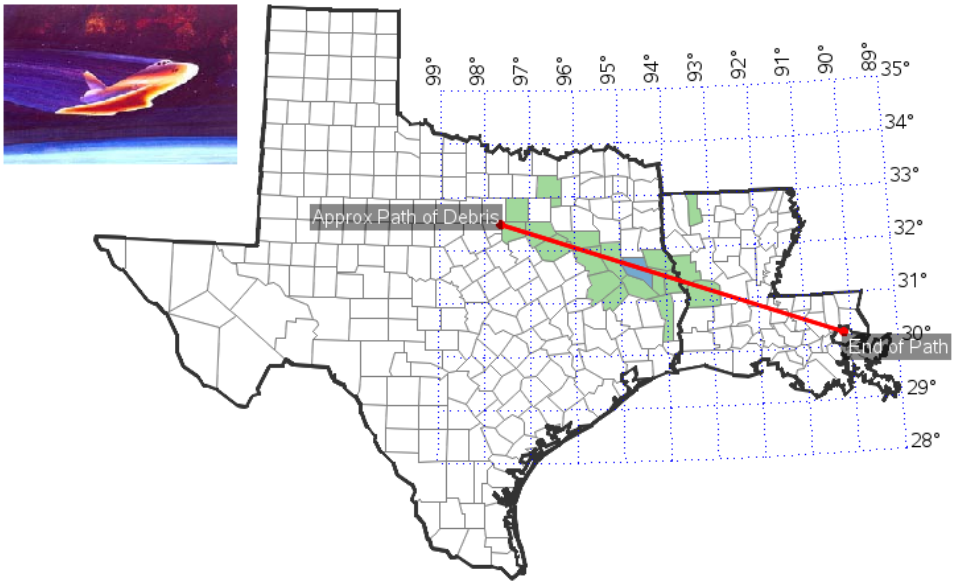


Annotate State Outline on County Map

On the following pages are some examples of fancier maps created using this technique. Note that in the above simple example I'm not actually plotting any response data on the map (all the map areas are the same color), but of course you can also use this technique on a colored choro map where you're plotting response data.

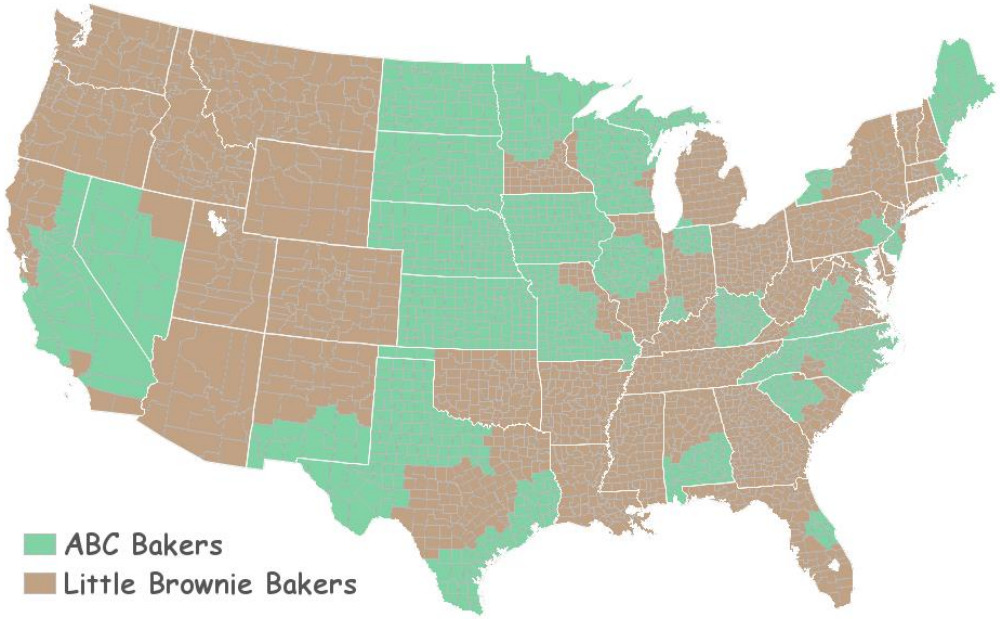
Space Shuttle Debris Field

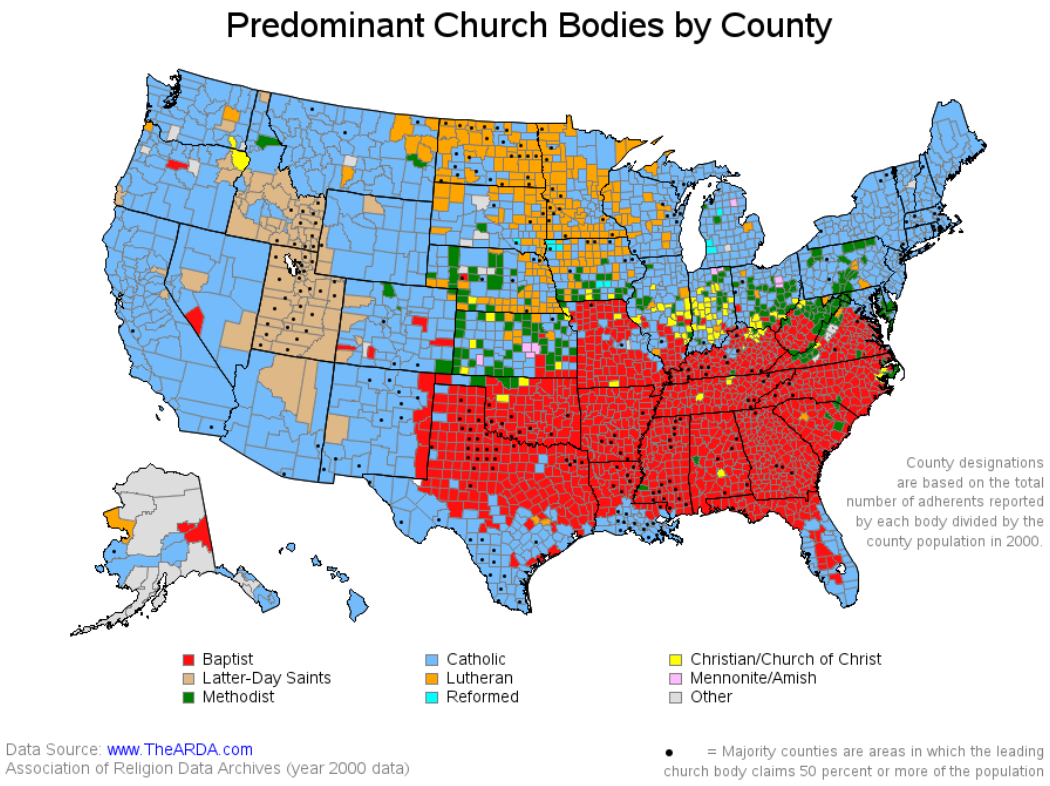
Mouse over counties to see debris count data.
Click on green counties to see 'mapquest' roadmap of that area.
Click on blue counties to see detailed map of debris in that county.

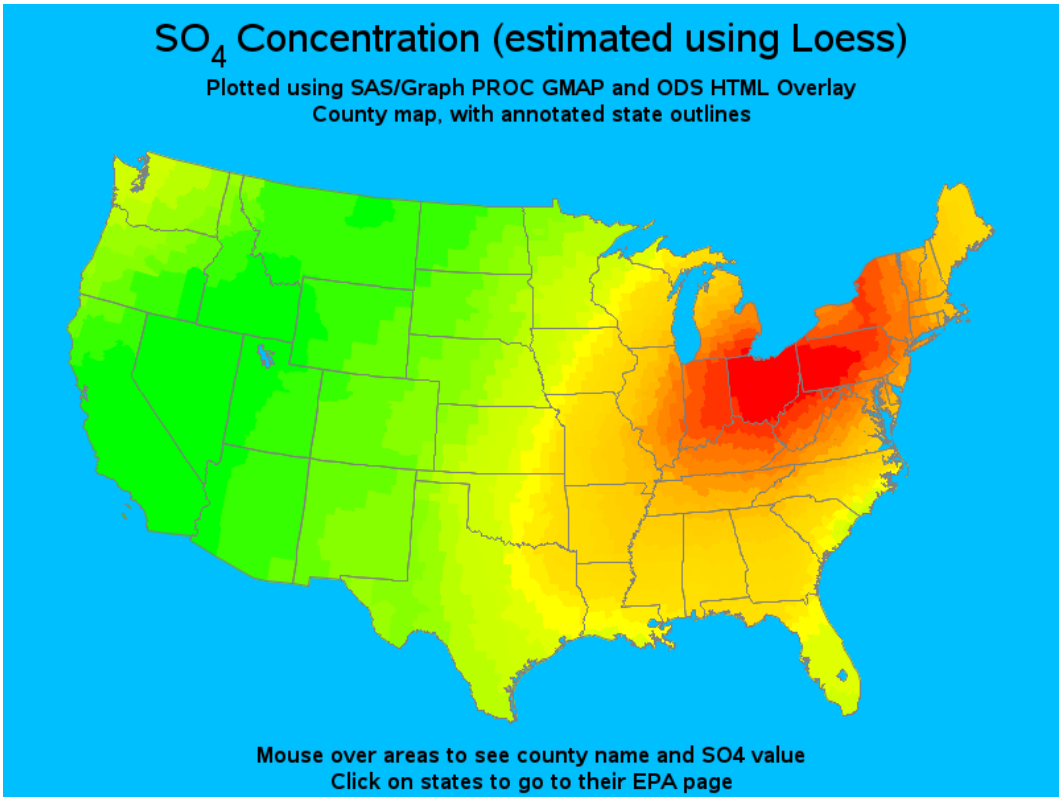


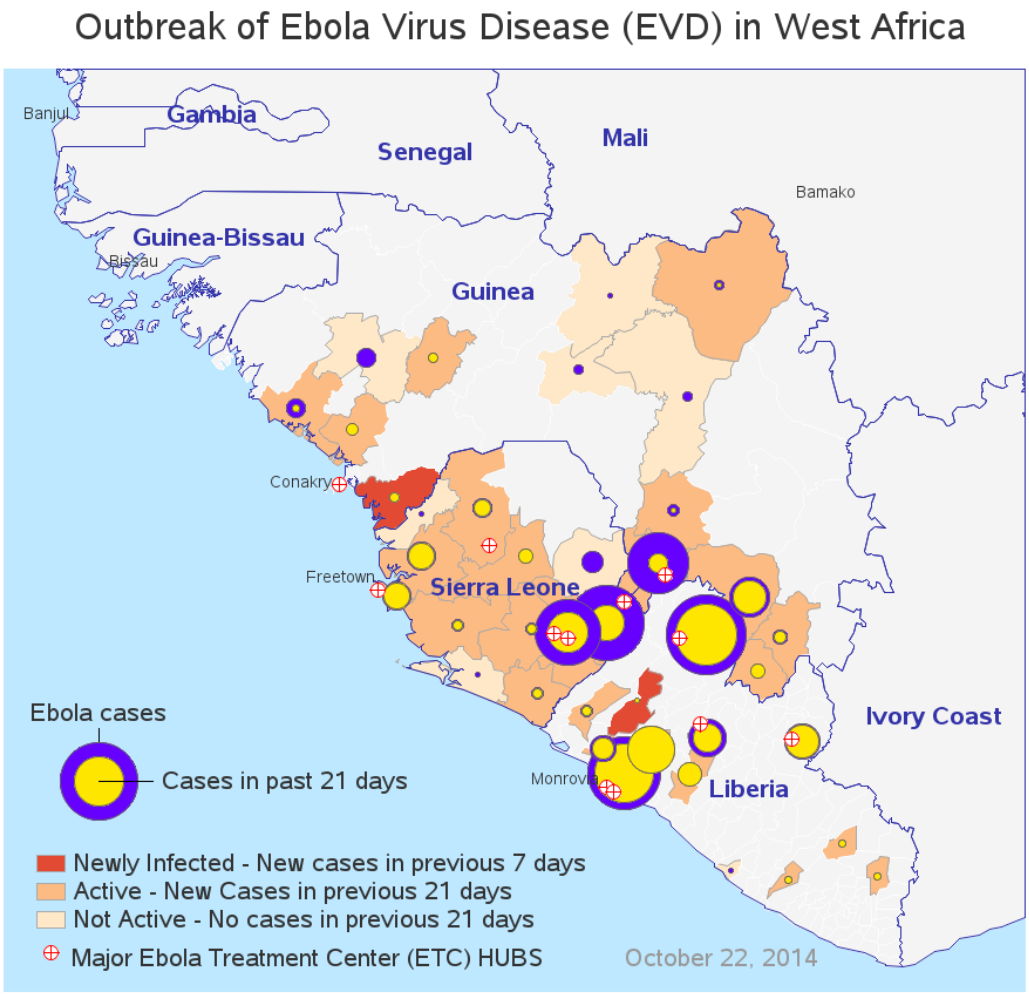
SAS Proof-of-Concept (using contrived data) somewhat based on [click here]

Who Makes Your Girl Scout Cookies?




Predominant Church Bodies by County
County designations are based on the total number of adherents reported by each body divided by the county population in 2000.
Baptist
Catholic
Christian/Church of Christ
Latter-Day Saints
Lutheran
Mennonite/Amish
Methodist
Reformed
Other
Data Source: www.TheARDA.com
Association of Religion Data Archives (year 2000 data)
= Majority counties are areas in which the leading church body claims 50 percent or more of the population


SO4 Concentration (estimated using Loess)
Plotted using SAS/Graph PROC GMAP and ODS HTML Overlay
County map, with annotated state outlines
Mouse over areas to see county name and SO4 value
Click on states to go to their EPA page


Outbreak of Ebola Virus Disease (EVD) in West Africa
Banjul
Gambia
Senegal
Mali
Bamako
Guinea-Bissau
Bissau
Guinea
Conakry
Freetown
Sierra Leone
Ivory Coast
Monrovia
Liberia
Ebola cases
Cases in past 21 days
Newly Infected - New cases in previous 7 days
Active - New Cases in previous 21 days
Not Active - No cases in previous 21 days
Major Ebola Treatment Center (ETC) HUBS
October 22, 2014
Proof-of-Concept Using Estimated Data