Paper RIV-14

"Google-Like" Maps in SAS®

Darrell Massengill, SAS Institute Inc., Cary, NC

ABSTRACT

We are frequently asked if we can have maps similar to Google Maps in SAS®. Customers want the background image displayed behind their data so they can see where streets or other features are located. They may also want to pan and zoom the map. Unfortunately, Google has legal restrictions and limitations on the use of their maps. Now, you can have "Google-like" maps inside of SAS. You may have already seen this capability in products like SAS® Visual Analytics Explorer and other products using them will be available in future releases. This presentation will discuss and demonstrate these new capabilities in SAS Visual Analytics Explorer, SAS/GRAPH®, and other products.

INTRODUCTION

Every organization has location-based data. Maps are used to transform this data into useful information. In many cases, seeing the data in an administrative map area, such as a state or a country, is enough. In other cases, the data needs to be viewed in relationship to other map features like cities, roads, or bodies of water. These additional features provide the necessary context for evaluating your information. For example, knowing that a river separates your store from potential customers might explain why they do not shop with you. In addition to seeing these additional map features, it can be useful to zoom in and out of the map area. It is sometimes helpful to be able to back away and see a larger area or, it might be advantageous to move in closer to the area, in order to examine the details very closely.

Having 'Google-like' background maps is one way to provide this useful context information to your map. While Google Maps can meet this type of need, there are legal restrictions that prevent you from using their free maps with SAS and with typical business situations. In order to provide this capability in SAS, we have added our own background maps.

BACKGROUND MAPS IN SAS

In order to have background maps, you must have dedicated servers that can generate and return the background map. Background maps are composed of numerous map tile images that get 'glued' together to form the map view. In order to provide background maps in SAS products, we have set-up our own servers that create the images from open source map data that is available from OpenStreetMap (OSM). These servers allow SAS customers to have 'out of the box' background maps. The background images are already being used in some SAS products and are now available for SAS/GRAPH users.

SAS Products

Some products have already shipped with this new mapping capability and other products will surface the capability in future releases. Visual Analytic Explorer (VAE) in the Visual Analytics (VA) product is already using this technology and has been demonstrated at past SAS Global Forum conferences. Figure 1 shows VAE with a Bubble Map on top of the OSM map. The VAE map can be panned and zoomed. Mobile versions of the VA product also have this capability. Figure 2 shows a screen capture of the mobile map. VA will add choropleth maps in a future release.

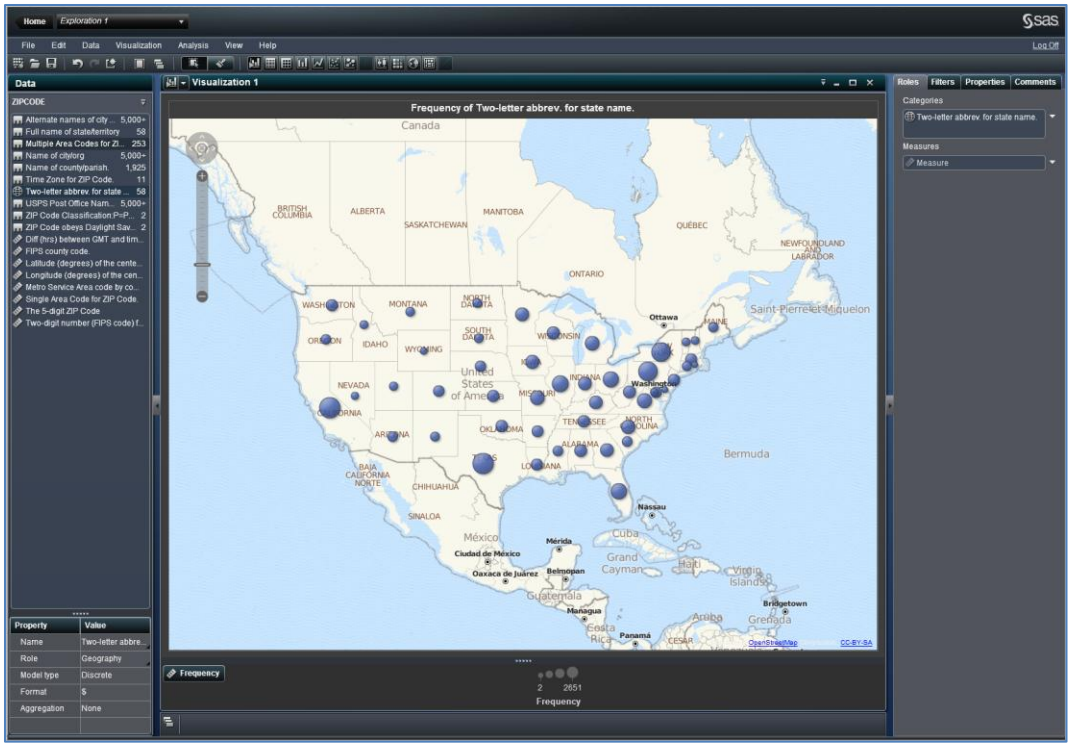

Figure 1. Bubble Map in Visual Analytic Explorer.

Figure 2. Mobile Map.

SAS/GRAPH

The background maps in VA and other SAS products are integrated directly into the functionality of that particular product. SAS/GRAPH users have requested this type of capability so they can have more control over the maps they create. SAS has introduced two tools for SAS/GRAPH; OSM Annotate Generator and the Java Map Applet. These tools enable you to add background images into your map.

OSM Annotate Generator

The OSM Annotate Generator is a tool written in SAS/AF SCL code. However, it does not require a SAS/AF license to run. This tool creates an annotate data set that contains a static background map to use with PROC GMAP. The resulting annotate data works with both older and newer versions of SAS.

The OSM Annotate Generator retrieves the required background map images from the server, stores them on your local disk, and creates an annotate data set that uses the images. You specify the information to determine the area you want to show on the map. It can also project your data into the proper form for the OSM map. PROC GMAP uses the data sets that were created. Figure 3 is an example of a choropleth map created with the tool.

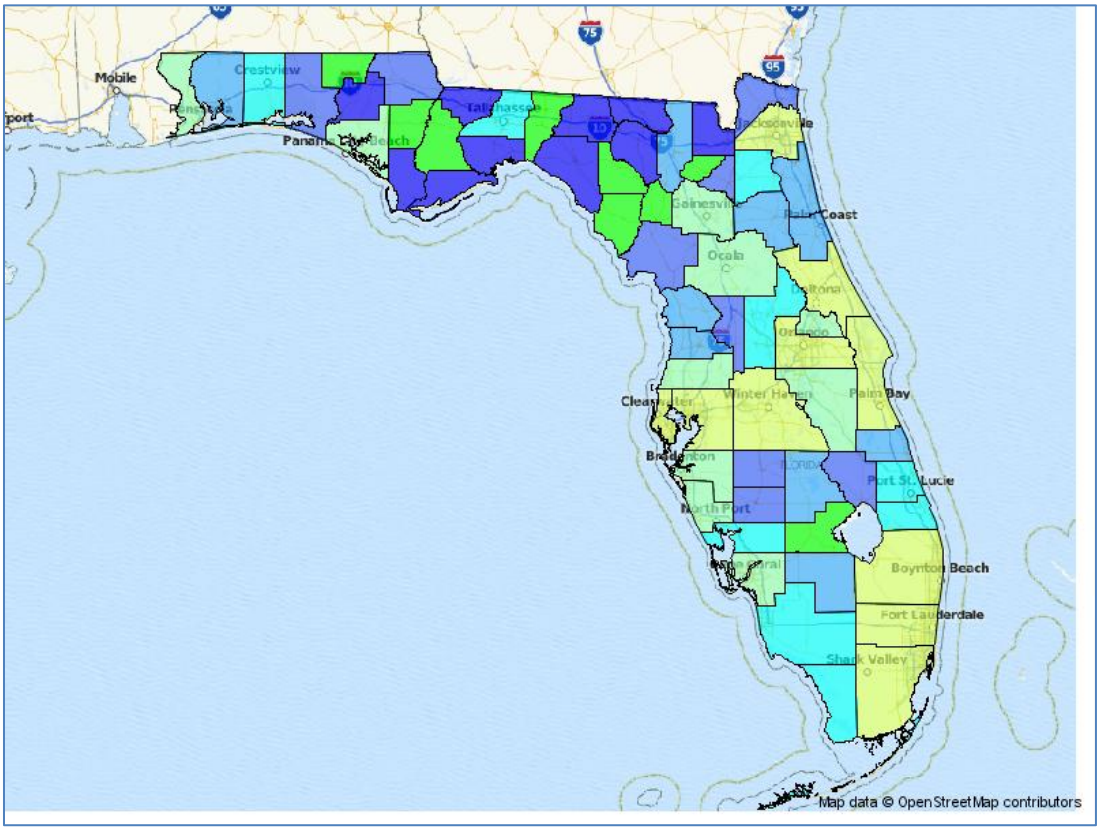

Figure 3. Choropleth map using the OSM Annotate Generator.

Java Map Applet

The Java Map Applet has been modified in SAS 9.4 to support OpenStreetMap background tiles. The resulting map can be panned and zoomed if the JAVA device is specified. Or, it will create static images if JAVAIMG is specified. This extended functionality in the Map Applet is pre-production in 9.4. The new functionality is enabled with special syntax that is described later in this paper.

Figure 4 shows an example of a choropleth map combined with Annotate data. Figure 5 shows the same map that has been panned and zoomed.

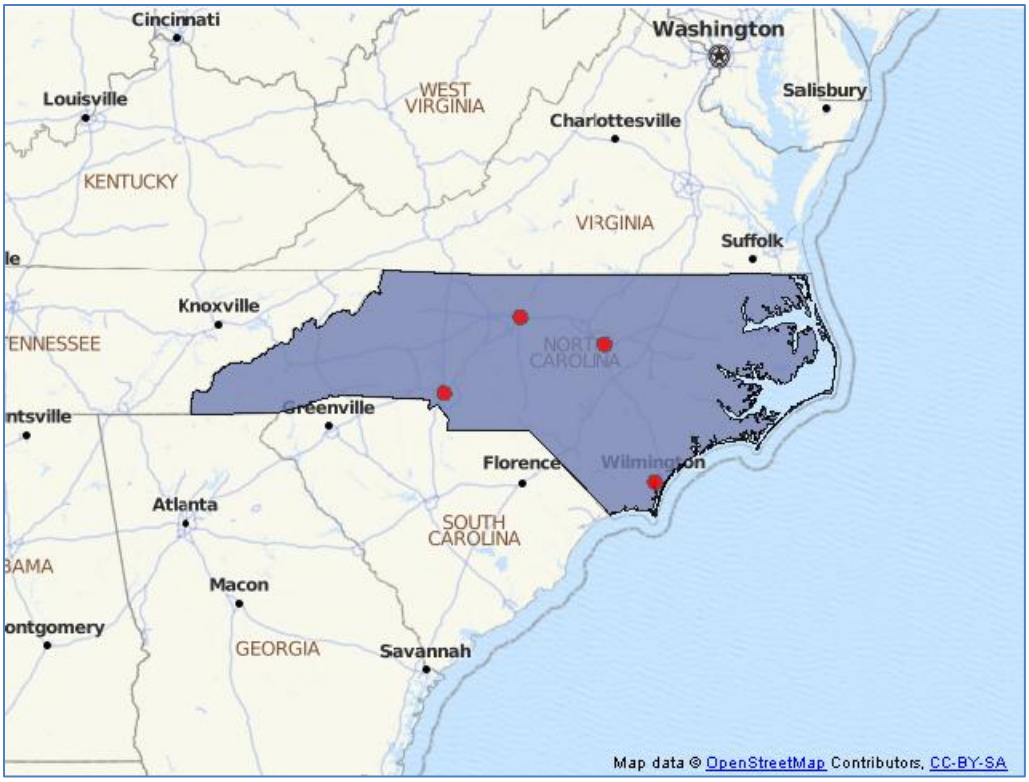

Figure 4. Map Applet with OpenStreetMap

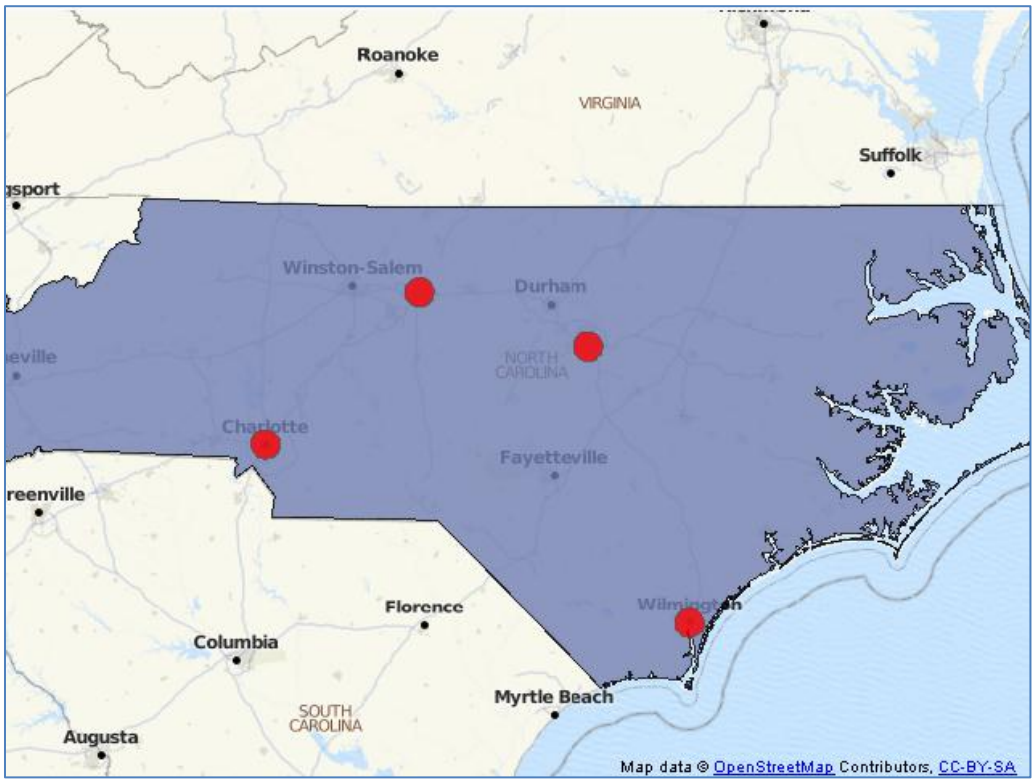

Figure 5. Same map panned and zoomed.

UNDERSTANDING THE BASICS

Before discussing how to use the new functionality and reviewing the examples, there are a few basic details that need to be understood.

Map Tiles

The background map servers return individual map tiles. Each map tile is 256 x 256 pixels. Many map tiles must be combined for each map view. These tiles are combined in the correct order to make the background map image. Figure 6 shows the outline of each tile in the whole map view. Notice that some tiles are not fully displayed.

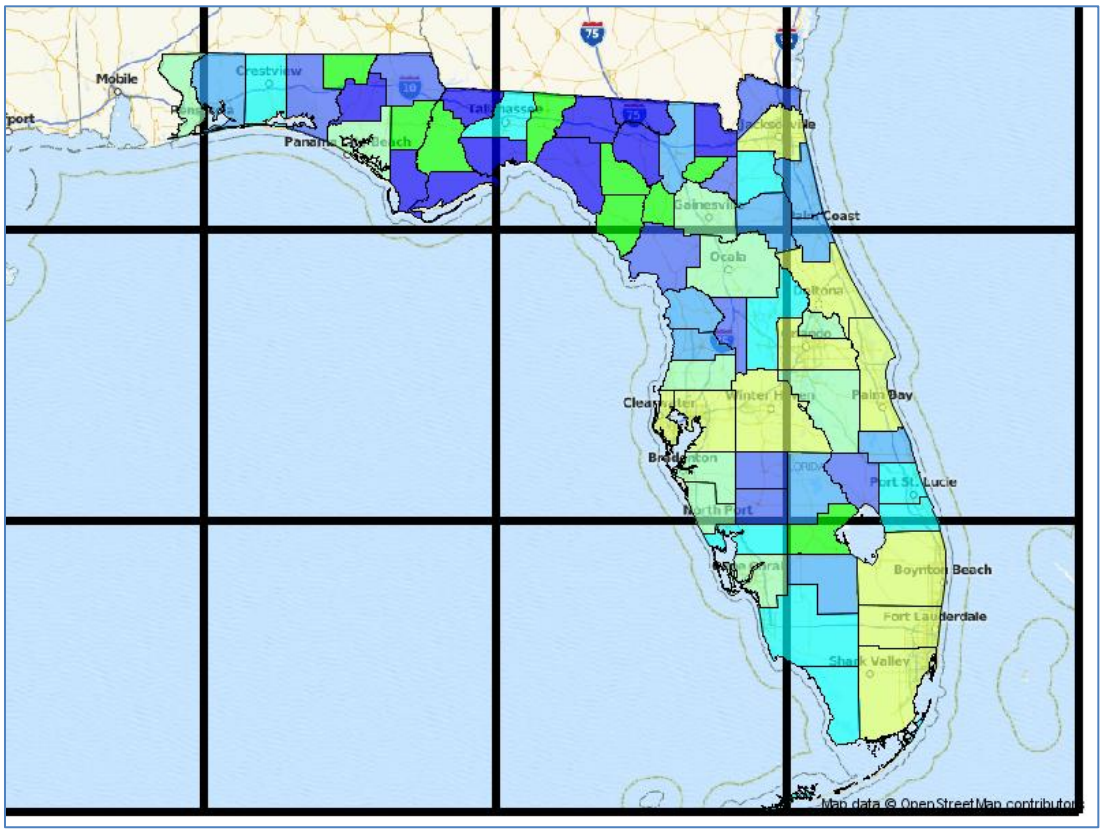

Figure 6. Illustration of MapTiles.

The Map Applet, the OSM Annotate Generator, and the map components used by VA and other products, must make the requests for the proper map tiles and put them together in the proper order. Depending on the display size and the map area to be shown, a particular zoom level is chosen and the tiles that are needed to fill that area are requested. Since the OSM Annotate Generator does not know the display size, it is more difficult for it to select tiles and fit them together. This can result in scaling of the image, truncation of tiles, and not completely filling the display area. In Figure 6 above, notice the partial tiles on the top and left. The truncated tiles will cause this type of message in your program:

```
NOTE: PROBLEM IN OBSERVATION 1 -
            DATA SYSTEM REQUESTED, BUT VALUE IS NOT ON GRAPH 'X'
```

For more information on Map Tiles, see: http://wiki.openstreetmap.org/wiki.

Map Data

Map Data is the polygonal data that defines countries, states, counties, and other similar areas. Historically, SAS shipped the MAPS data set library that contained this data. Starting in SAS 9.3M2, SAS licensed new polygonal data from GfK GeoMarketing and ships this data in the MAPSGFK library.

The MAPSGFK data sets will eventually replace the MAPS data as there are issues with the MAPS data sets. A few important items are:

- There were multiple sources for this data and some of it was not accurate enough.
- The sources for this data are no longer available to us, so we cannot make corrections and changes.

The MAPSGFK data sets come from a single source that have updates. These data sets are very accurate. However, the MAPSGFK data is not "plug and play" compatible with the MAPS data. Slight changes in the programs are required when you switch between data.

Issues to be aware of include the following:

- Data set names may be different.
- Some variables may be different in the data sets.
- All MAPSGFK data sets use X and Y for projected data and LONG and LAT for unprojected data. Note: You must start with unprojected data for the OSM background maps.
- All MAPSGFK data are in degrees instead of radians. Coordinates are no longer based on Western Hemisphere.

For more complete information on the GfK maps and differences, see the paper "Tips and Tricks: Using the new SAS Map data sets" at http://support.sas.com/rnd/papers under SAS Global Forum 2013.

For the examples in this paper, the following code is typical for processing each of the two types of map data sets. Note that if you are using MAPS data, it may not always align properly on the OSM background maps due to the accuracy issues.

Using MAPSGFK

```
/* Subset map for Florida counties only */
data mymap; set mapsgfk.us_counties (where=(statecode='FL'));
    /* lat, long are unprojected in MAPSGFK data */
    /* Save as y, x to make easier to use        */
    x=long; y=lat;
run;
```

Using MAPS

```
/* Subset map for Florida counties only */
/* in MAPS data sets, x and y are unprojected radians */
data rmymap; set maps.counties (where=(state=(stfips('FL'))));
run;

/* change data from radians to degrees */
%to_degrees(mymap,rmymap);
```

Geocoding

Some of the examples use PROC GEOCODE to convert addresses into map coordinates. This paper does not go into detail on using this PROC. For more complete information on PROC GEOCODE, see the paper "PROC GEOCODE: Finding locations outside the US" at http://support.sas.com/rnd/papers under SAS Global Forum 2013.

Macros

In order to simplify some of the tasks in the examples and keep the focus on what is new, a set of macro programs are provided in order to execute many functions. The source code is also provided if you want to understand how the code works. The provided macros are:

- %to_mercator - Convert unprojected data to Mercator projection as required for OSM background maps.
- %to_degrees – Convert radian coordinate data to degrees.

- %to_radians – Convert degree coordinate data to radians.
- %make_dots – Create an annotate data set of dots when a data set of coordinates is provided.
- %make_circle – Create an annotate data set or map data set of circles when provided coordinates.
- %draw_map – Run PROC GMAP to draw a map.

OSM ANNOTATE GENERATOR

Syntax

The OSM Annotate Generator is a SAS/AF SCL application. Values are passed-in by setting special macro variables with %let. For example:

```
%let _inmapds=work.mymap;
```

- Each of the macro variables are described below. However, their use is easier to understand in the example code. You must specify either `_inmapds` or `_inannods`, but not both. This data set is used to determine the location and size of the background map so that all of the polygons are shown or all of the annotate data is shown.

  _inmapds – Input map data set (polygonal data)

  _inannods – Input annotate data set

- You must specify the location where the map tile images are to be saved on disk. This location is then used in the annotate data set that is created. Note that this location must have the final slash since it is a directory path rather than a filename.

  _imageloc – Location where the map tile images will be downloaded and saved.

- You should specify the name of the annotate data set that is created with the background image. If not specified, it will default to 'work.anno_osm';

  _outannoOSMds – Output annotate data set containing the background map.

- You should specify the size of your map. Since the OSM Annotate Generator is not part of PROC GMAP, you need to specify this information:

  _xpixels – xpixel size of the background image. Defaults to 800.

  _ypixels – ypixel size of the background image. Defaults to 600.

- You should turn on auto projection so that all data is projected to the Mercator projection. If you do not, you will need to project it yourself. To turn it on, you specify `%let _autoproject="Y";`

  _autoproject – Turn on the projection to Mercator to match what is needed for OSM.

- In addition, you need to specify one of the following in order to save the projected values:

  _outmapds – Output map data set

  _outannods – Output annotate data set.

- If you wish to only create a background map of a particular area, you can specify the _backgroundonly option. This will create a rectangular map based on either _inmapds or _inannods and output to _outmapds. To turn backgroundonly on, you specify %let _backgroundonly="Y";

  _backgroundonly – Turn on the background only option.

Other options include:

- **_tileserv** – Location of the map tile server being used if other than the default server.
- **_copyright** – Set the text of the copyright string for alternate _tileserv settings.
- **_proxy** – Set your proxy server if needed.

To run the OSM Annotate Generator, you must specify the following:

```
proc display cat=OAG.osmmapanno.osm_anno_gen.scl; run;
```

Since the values are set with macro variables, these remain set until you change or clear them. If you forget to clear them, you may get unexpected results or errors. It is strongly recommended that you run OAG_reset.sas before each program.

Examples

All example code that is discussed in this paper can be downloaded from http://support.sas.com/rnd/papers. Click on the link for SAS Global Forum 2013. In that section, look for this paper, "Google-like maps in SAS". There are two links; one to this paper and another to a zipfile containing all examples.

The OSM Annotate Generator examples all follow a similar layout with the following sections.

- LIBNAMES and FILENAMES are defined and macro code is included. Here is the example code:

```
/* Location for HTML output */
filename odsout 'c:\';

/* included macro programs */
%include "c:\SGF2013\macros\mapmacros.sas";

/*location of the SCL catalog containing the OSM Annotate Generator */
libname OAG 'C:\SGF2013\AFpgm';
```

- The map data set or the point data set is created and/or processed. Below is example code for creating map data from the MAPS library. As explained previously, the code is slightly different for MAPSGFK data.

```
/* Subset map for Florida counties only */
/* in MAPS data sets, x and y are unprojected Radians */
data rmymap; set maps.counties (where=(state=(stfips('FL'))));
run;

/* change data from radians to degrees */
%to_degrees(mymap,rmymap);
```

  If point data is being used, it is converted into an annotate data set. To simplify this process of creating the annotate data set, a %make_dots macro program is included. The point data may be created by using PROC GEOCODE to convert address information into coordinates.

- If a choropleth map is being drawn, then the response data is created and/or processed. Here is example code for reading in crime data for Florida.

```
/* Process response data - crime data */
data crimeFL;
    infile crimedat;
    input countynm $1-13 arrests county;
run;
```

- Code for the OSM Annotate Generator is then run: This is the most important section and is discussed for each example. Here is example code for producing a choropleth map:

```
%let _inmapds=mymap;
%let _outmapds=work.mmymap;
%let _outannoOSMds=work.mannoosm;
%let _imageloc='c:\OSM\'; /*must have final slash*/
```

```
%let _xpixels=800;
%let _ypixels=600;
%let _autoproject='Y';

goptions reset;
/* Set xpixels, ypixels to the size to be used */
goptions xpixels=&_xpixels ypixels=&_ypixels;

/* Generate background map image */
proc display cat=OAG.osmmapanno.osm_anno_gen.scl; run;
```

- PROC GMAP is used to display the resulting map. The examples all create ODS HTML output. The example below is for a choropleth map and uses transparent choropleth colors (availiable in SAS 9.3 and SAS 9.4).

```
/*************************************************************
 * Create an HTML file output.
 *************************************************************/
GOPTIONS DEVICE=png;
ODS LISTING CLOSE;
ODS HTML path=odsout body="ChoroFl94_png.html";
pattern; /*reset pattern statements */

/* Set transparent colors - works in 9.3 and 9.4 */
pattern1  v=solid repeat=1 c=A00ff00aa;
pattern2  v=solid repeat=1 c=A0000feaa;
pattern3  v=solid repeat=1 c=A3661feaa;
pattern4  v=solid repeat=1 c=A38acffaa;
pattern5  v=solid repeat=1 c=A00ffffaa;
pattern6  v=solid repeat=1 c=A91ffb4aa;
pattern7  v=solid repeat=1 c=Ad2fe69aa;
pattern8  v=solid repeat=1 c=Affff00aa;
pattern9  v=solid repeat=1 c=Affb700aa;
pattern10 v=solid repeat=1 c=Aff6f00aa;
pattern11 v=solid repeat=1 c=Afe0000aa;

/*Draw _outmapds map dataset that was projected to mercator, */
/*use the _annooutOSMds created that has the background map  */
/*images and color the choropleth with the crime data        */
proc gmap map=&_outmapds anno=&_outannoOSMds data=crimefl all;
     id county;
     choro arrests /nolegend name="ChoroFl94";
run;
quit;
ODS HTML CLOSE;
ODS LISTING;
```

OAG_ChoroMapGfK.sas and OAG_ChoroMap.SAS

This example draws a choropleth map of Florida, showing crime rates per county with an OSM background map. The difference between these two programs is that one uses MAPSGFK map data sets and the second uses MAPS data sets.

Figure 7 shows the output from these programs. Notice that the transparent colors on each county.

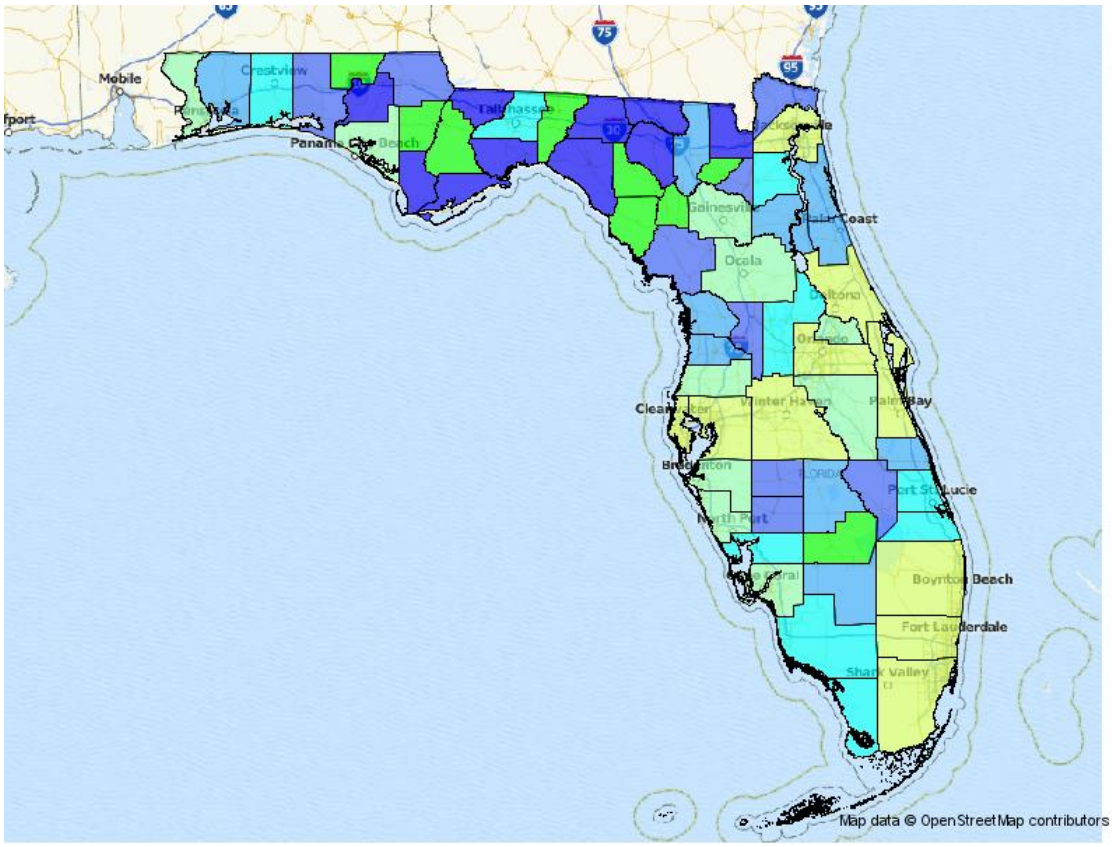

Figure 7. Choropleth Map

The code first creates the map data set, `mymap`, with only the Florida counties and assigns it to the `_inmapds` macro variable. Because we want the data set values automatically converted to Mercator projection, the code sets `_autoproject` to 'Y' and specifies a data set name for the resulting data. This is specified by `_outmapds` and it will be used by Proc GMAP.

Next, we want our map output to be about 800x600 pixels. These values are specified with `_xpixels` and `_ypixels` and then those values are used to set the GOPTIONS xpixels and ypixels values.

Finally, the annotate data that is set to be created is specified by `_outannoOSMds`. This annotate data set name is used by Proc GMAP. In addition, the map tile images must be downloaded to disk to be used by the annotate code. This location is specified by `_imageloc`. Notice that the path used with `_imageloc` must contain the final slash since it is a path rather than a file name.

After the macro variables are setup, Proc DISPLAY is called to run the OSM Annotate Generator. The code is shown below:

```
%let _inmapds=mymap;
%let _outmapds=work.mmymap;
%let _autoproject='Y';
%let _outannoOSMds=work.mannoosm;
%let _imageloc='c:\OSM\'; /*must have final slash*/
%let _xpixels=800;
%let _ypixels=600;

goptions reset;
/* Set xpixels, ypixels to the size to be used */
goptions xpixels=&_xpixels ypixels=&_ypixels;

/* Generate background image */
proc display cat=OAG.osmmapanno.osm_anno_gen.scl; run;
```

OAG_EmptyMapGfK.sas and OAG_EmptyMap.sas

These examples draw a Florida map with only the outlines of the counties. There are a number of ways that this can be done. These examples are implemented similarly to the choropleth examples using map data sets, except they are drawn as empty. Like the choropleth examples, the difference between these two programs is that one uses MAPSGFK map data sets and the second uses MAPS data sets.

The output from the map will look like Figure 8.

Figure 8. Empty outline map

In these examples, the code for calling the OSM Annotate Generator is identical to the choropleth examples. The difference is in the Proc GMAP code. Instead of defining a list of PATTERN statements for each of the colors, one completely transparent pattern is defined for all colors. This makes all the polygonal areas transparent or empty. Note that this works in SAS 9.3 and SAS 9.4 code. In addition, the crime data set is not needed. The example sets the response data set (DATA=) to be the map data set since the polygonal areas are not colored.

The pattern statement used is:

```
pattern v=me c=A00000000 r=100;
```

OAG_AnnoMapAD.sas and OAG_AnnoMap.sas

These examples draw annotate points on the map. The difference between these two examples is that the area of the map displayed is determined with either the annotate points (OAG_AnnoMapAD.sas) or by a map area (OAG_AnnoMap.sas). OAG_AnnoMapAD.sas output is shown in Figure 9 and shows only the area of the points on the map. OAG_AnnoMap.sas output in Figure 10 shows the whole US map.

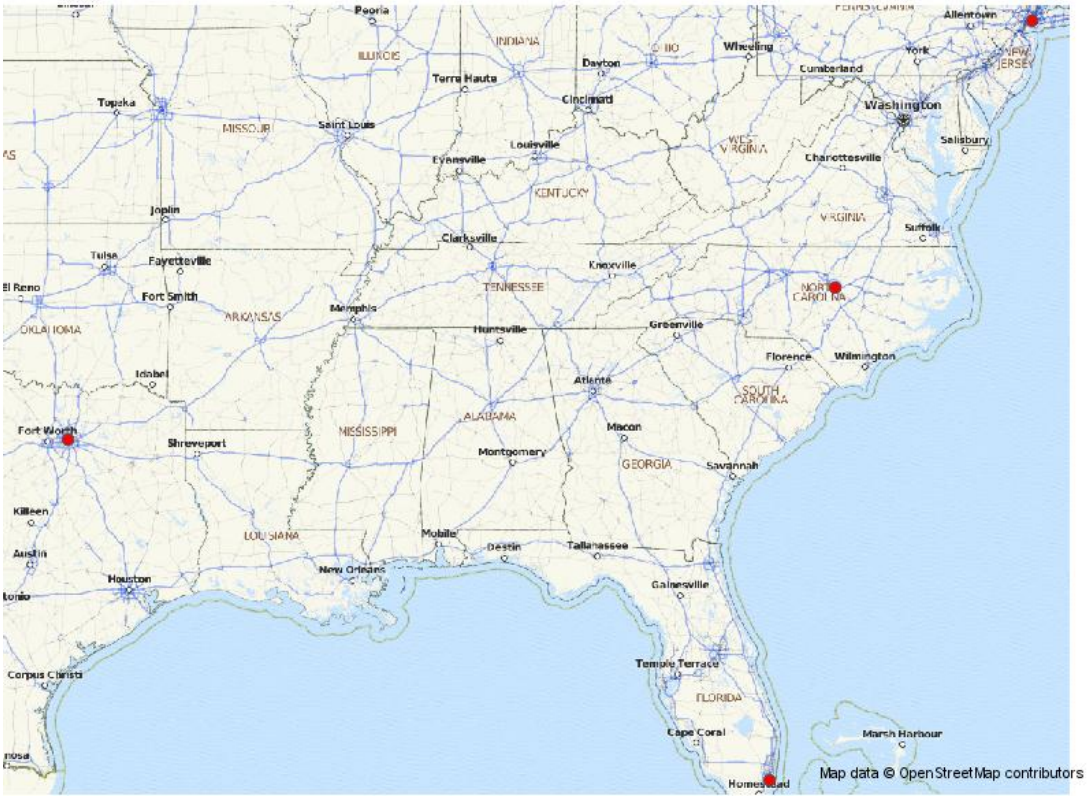

Figure 9. Output from OAG_AnnoMapAD.sas

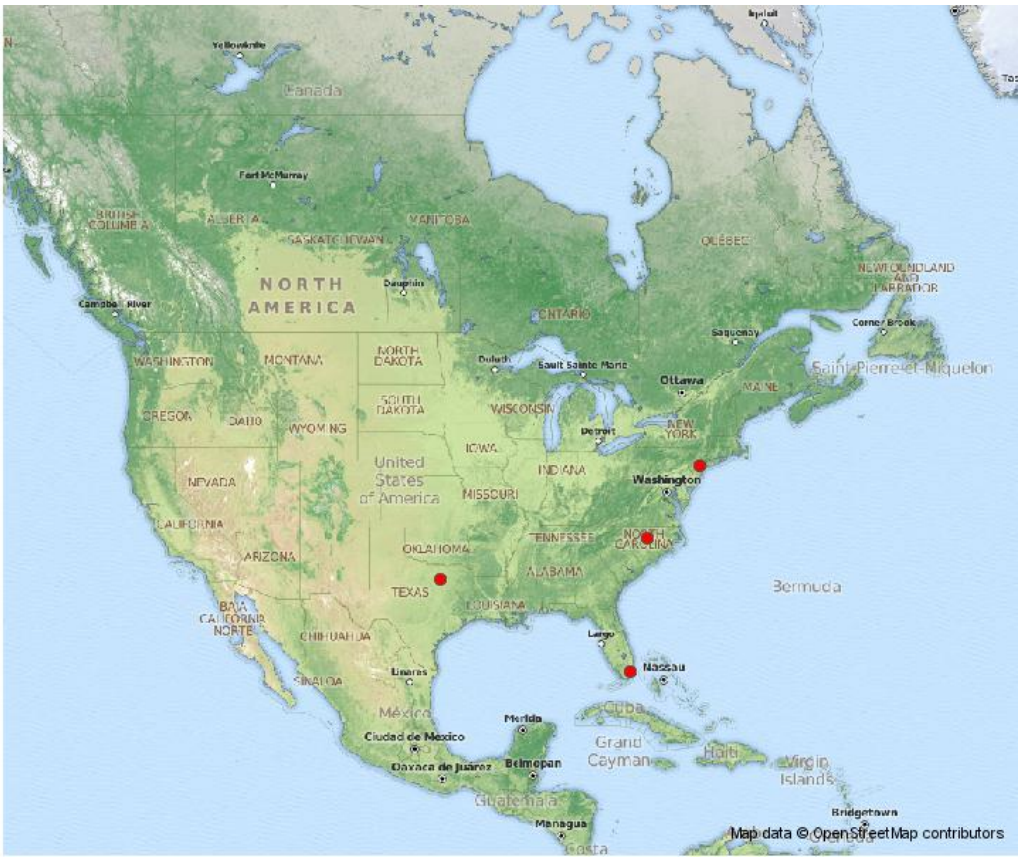

Figure 10. Output from OAG_AnnoMap.sas

These examples draw four city locations in the US. These city locations are created as an annotate data set containing 'dots' to be drawn. OAG_AnnoMapAD.sas uses this annotate data set to determine the size and area of the map drawn. In Figure 9, the resulting map is 'zoomed' into the four annotate points.

The code first sets the `_inannods` macro variable to the name of the annotate data set of 'dots'. Note that we do NOT use the `_inmapds` in this case. Since we do want the data set values automatically converted to Mercator projection, `_autoproject` is set to 'Y' and a data set name is specified for the resulting data. This is specified by `_outannods` and is used by PROC GMAP as the background annotate data set along with the `_outannoOSMds`. Also, `_backgroundonly` is set to "Y" to indicate that we want a background image with no map polygons drawn. This causes an output map data set to be created that is a bounding box of the area you want to display. The value of `_outmapds` specifies the value of this bounding box data set but it is not drawn on top of the background map.

Next, we want the map output to be about 800x600 pixels. These values are specified with `_xpixels` and `_ypixels`. Those values are then used to set the GOPTIONS xpixels and ypixels values.

Finally, the annotate data set that is created is specified by `_outannoOSMds`. This annotate data set name is used by Proc GMAP. In addition, the map tile images must be downloaded to disk in order to be used by the annotate code. This location is specified by `_imageloc`. Notice that the path used with `_imageloc` must contain the final slash since it is a path rather than a file name.

After the macro variables are setup, PROC DISPLAY is called to run the OSM Annotate Generator. Here is the code:

```
%let _backgroundonly='Y';
%let _inannods=annocity;
%let _outannods=mannocity;
%let _outmapds=mmap;
%let _outannoOSMds=work.mannoosm;
%let _imageloc='c:\OSM\'; /*must have final slash*/
%let _xpixels=800;
%let _ypixels=600;
%let _autoproject='Y';
```

```
goptions reset;
/* Set xpixels, ypixels to the size to be used */
goptions xpixels=&_xpixels ypixels=&_ypixels;

/* Generate background image */
proc display cat=OAG.osmmapanno.osm_anno_gen.scl; run;
```

OAG_AnnoMap.sas code only differs slightly from OAG_AnnoMapAD.sas code. In this case, the resulting map shows all 48 states. To do this, first create a map data set containing only those 48 states. This value is set to `_inmapds`. Note that `_inannods` is not specified in this case because the annotate data set is not used. This also means that the annotate data is not converted to the Mercator projection, so you do not specify `_outannods` either. You must convert the annotate data to the Mercator projection since it never passes through the OSM Annotate Generator. This is illustrated in the example code with a call to the `%to_mercator` macro provided with the examples:

```
/* Convert to mercator projection*/
%to_mercator(mannocity, annocity, 0);
```

This means you remove `_inannods` and `_outannods` and add `_inmapds`. In addition, add the `%to_mercator` call to the annotate data. You then get the following code instead:

```
%let _backgroundonly='YES';
%let _inmapds=mymap;
%let _outmapds=mmap;
%let _outannoOSMds=work.mannoosm;
%let _imageloc='c:\OSM\'; /*must have final slash*/
%let _xpixels=800;
%let _ypixels=600;
%let _autoproject='Y';
```

OAG_AltServers.sas

It is possible that you may need to use a different set of background map tiles other than the set provided by SAS. In order to do this, you must specify a different map tile server. Before using any map tile server, you must read the Terms of Use documentation to make sure you have a legal right to use the tiles. In addition, you should apply the proper copyright message to your map. This example shows how to redirect to an alternate server. However, the use of a tile server in this example does not mean that you are allowed to use it; you must verify this for yourself. Note that other tile servers may be extremely slow and therefore the program may run extremely slow.

This example is similar to OAG_AnnoMapAD.sas except that it introduces two additional macro variables: `_tileserv` and `_copyright`. By default, `_tileserv` is set to the SAS provided tile server. To point it to an alternate server, you can specify something like the following:

```
%let _tileserv=otile1.mqcdn.com/tiles/1.0.0/sat;
```

Figure 11 shows the output from the default SAS tile server. Figure 12 shows the output from Open Aerial Map. Figure 13 shows an example of output from a terrain tile server.

When specifying a `_tileserv`, you must also specify `_copyright` to set the proper copyright message that is required by the owner of those map tiles. The specified message appears in the bottom right of the map.

```
%let _copyright="Data, imagery and map information provided by MapQuest, Open
Street Map and contributors, CC-BY-SA.";
```

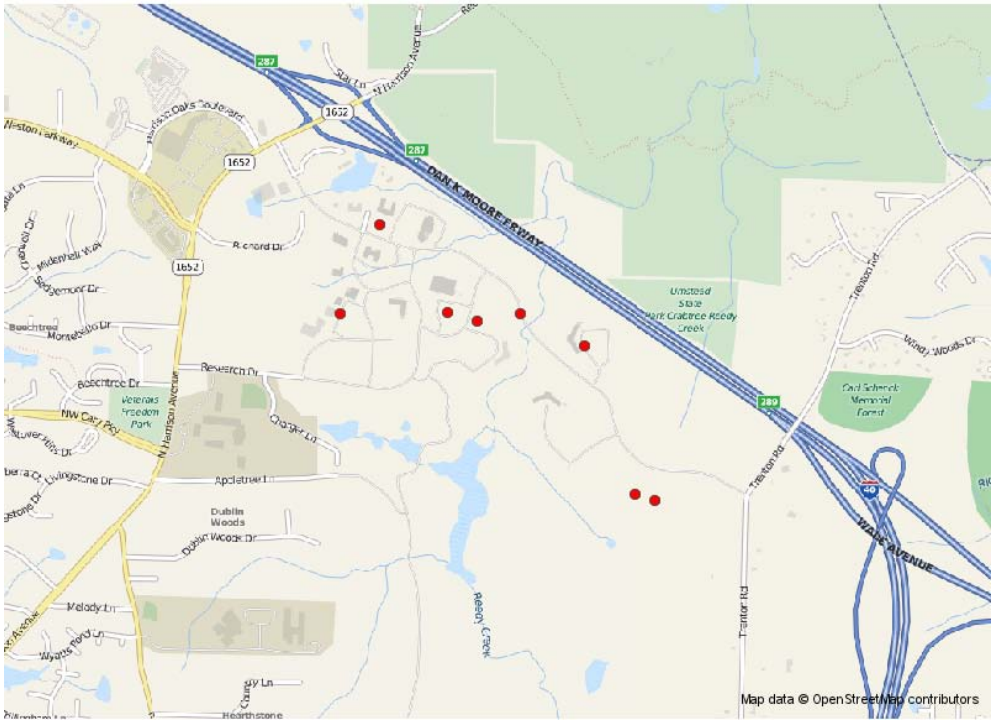

Figure 11. SAS tile server

Figure 12. Open Aerial Map tile server on MapQuest

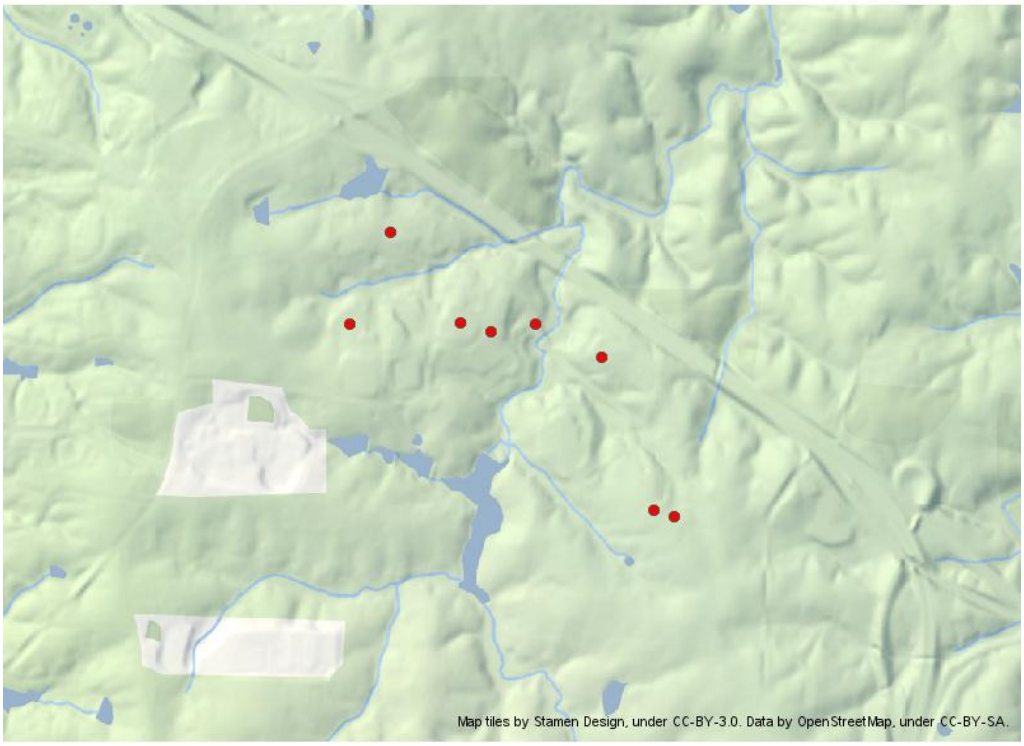

Figure 13. Terrain tile server hosted by Stamen Design

OAG_Test_Connect.sas

OAG_Test_Connect.sas is a small test program that can be used to insure that you are properly configured to access a tile server. It can also be used to determine the time it takes to acquire one or more map tiles from that server. The example is self-explanatory and no map is displayed.

JAVA MAP APPLET

Syntax

The JAVA Map Applet is an extension of the existing Map Applet. By default, the old map applet functionality runs. The map applet is pre-production and will be introduced with SAS 9.4. There are syntax changes to Proc GMAP in SAS 9.4 for using it. However, it is possible to use it with earlier versions of SAS. A slightly updated version of the Applet can be downloaded and used with SAS 9.4 and earlier versions of SAS. It is located on the MapsOnLine website, in the Tools section under Resources: http://support.sas.com/rnd/datavisualization/mapsonline/html/tools.html. The syntax below is used to enable the extended functionality, depending on which version of SAS you are using.

With SAS 9.4, there are options that are added to the PROC GMAP CHORO statement that enable the OpenStreetMap capability.

This syntax works with both the JAVA device and the JAVAIMG device.

CHORO variable / **SHOWOSM<=(<STYLE=osmstyle> <autoproject>)>;**

osmstyle can be SASMAPNIK or SASMAPNIK_LITE;

Here are some example cases:

- Turn on OSM to the default style of SASMAPNIK.

```
CHORO ID / showosm;
```

- Turn on OSM to use the lite style.

```
CHORO ID / showosm=(style=SASMAPNIK_LITE);
```

- Turn on OSM to use autoprojection to Mercator.

```
CHORO ID / showosm=( autoproject);
```

- Turn on OSM to use the lite style and use autoprojection.

```
CHORO ID / showosm=(style=SASMAPNIK_LITE autoproject);
```

Before SAS 9.4, you must enable the OSM capability with the 'parameters' option on the ODS HTML statement. This syntax will work only with the JAVA device.

ODS HTML …. **parameters=("OSMSTYLE"="style" <"AUTOPROJECT"="TFvalue">);**

where:

style is SASMAPNIK or SASMAPNIK_LITE

TFvalue is TRUE or FALSE (Converts unprojected lat/long to Mercator projection)

Here are some example cases:

- Turn on OSM to the SASMAPNIK style.

```
ODS HTML … parameters=("OSMSTYLE"="SASMAPNIK");
```

- Turn on OSM to use the lite style.

```
ODS HTML … parameters=("OSMSTYLE"="SASMAPNIK_LITE");
```

- Turn on OSM to use lite style and autoproject to Mercator.

```
ODS HTML … parameters=("OSMSTYLE"="SASMAPNIK_LITE"  "AUTOPROJECT"="TRUE");
```

Examples

All example code can be downloaded from http://support.sas.com/rnd/papers. Click on the link for SAS Global Forum 2013. In that section, look for this paper, "Google-like maps in SAS". There are two links: one to this paper and another to a zipfile containing all examples.

The Map Applet examples are all written in the way that you would have written the code before the addition of OSM background maps, except that data must be in Mercator projection and there is new syntax to enable the functionality.

The new functionality was written for SAS 9.4 with the use of the new MAPSGFK maps. SAS 9.4 examples end in 94 and can produce both JAVA and JAVAIMG output. SAS also provided some alternate syntax to enable most of the functionality to work in earlier versions of SAS. Examples that use the alternative syntax and use the MAPS data sets end in 93. These examples can produce only JAVA output. Note that OSM extended functionality is pre-production.

The JAVA Map Applet examples all follow a similar layout with the following sections.

- LIBNAMES and FILENAMES are defined and macro code is included. Example code:

```
/* Location for HTML output */
filename odsout 'c:\';

/* included macro programs */
%include "c:\SGF2013\macros\mapmacros.sas";
```

- The map data set or the point data set is created and/or processed. Below is example code for creating map data from the MAPS library. As explained previously, the code is slightly different for MAPSGFK data.

```
/* Subset map for Florida counties only */
/* in MAPS data sets, x and y are unprojected Radians */
data rmymap; set maps.counties (where=(state=(stfips('FL'))));
run;

/* change data from radians to degrees */
%to_degrees(mymap,rmymap);
```

  If point data is used, it is converted into an annotate data set. To simplify this process of creating the annotate data set, a %make_dots macro program is included. The point data may be created by using PROC GEOCODE to convert address information into coordinates.

- The map data must be converted to Mercator projection. These examples use the %to_mercator macro program.

```
/* Convert to mercator projection*/
%to_mercator(proj_map, mymap, 0);
```

- If a choropleth map is being drawn, then the response data is created and/or processed.
- PROC GMAP is used to display the resulting map. The examples all create ODS HTML output.

```
/*************************************************************
 * Create an HTML file output.
 *************************************************************/
GOPTIONS DEVICE=JAVA;
ODS LISTING CLOSE;

/** Syntax for 9.4 **/
ODS HTML path=odsout body="Choro_png.html";
/** Syntax for 9.3 **/
ODS HTML path=odsout body="Choro_png.html"
         parameters=("OSMSTYLE"="SASMAPNIK");


/*Draw _outmapds map dataset that was projected to mercator, */
/*use the _annooutOSMds created that has the background map  */
/*images and color the choropleth with the crime data        */
proc gmap map=proj_map data=proj_map;
     id id;
     /** Syntax for 9.4 **/
     choro id /nolegend name="Choro" showosm;
     /** Syntax for 9.3 **/
     choro id /nolegend name="Choro";

run;
quit;
ODS HTML CLOSE;
ODS LISTING;
```

MA_ChoroMap94.sas and MA_ChoroMap93.sas

These examples create a choropleth map on top of the OSM background. Figure 14 shows the output from these examples.

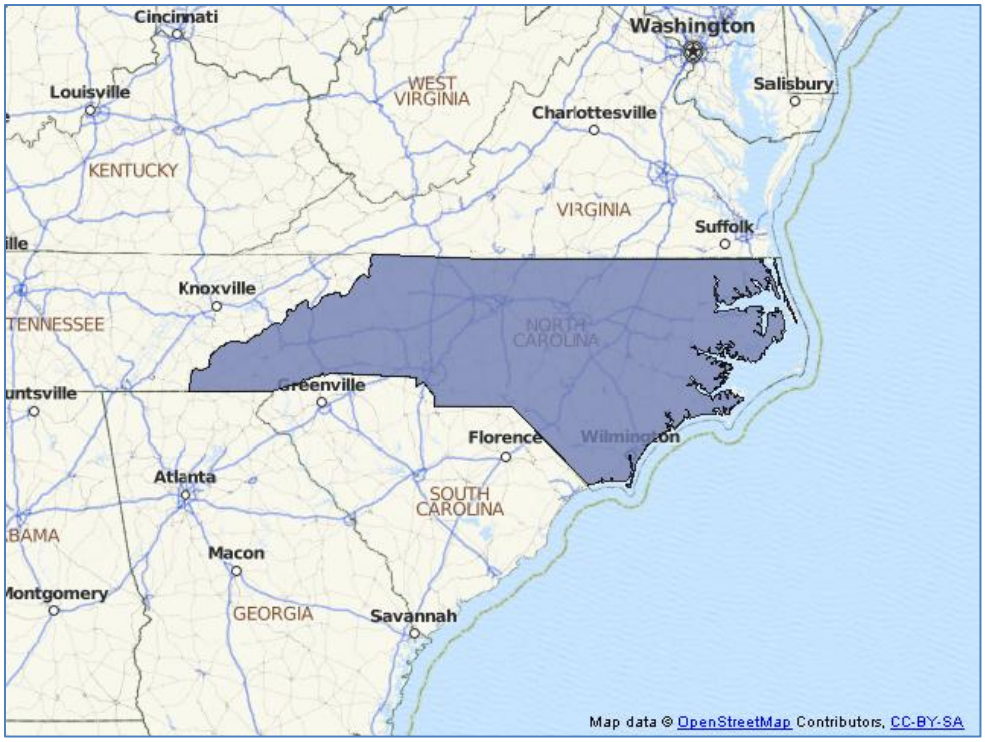

Figure 14. Output from MA_ChoroMap examples

The code for this example is identical to the previously discussed code. The differences between the two programs are:

MA_ChoroMap94:

- uses the MAPSGFK data sets.
- uses the following syntax:

```
ODS HTML path=odsout body="Choro_png.html";
choro id /nolegend name="Choro" showosm;
```

MA_ChoroMap93:

- uses the MAPS data sets.
- uses the following syntax:

```
ODS HTML path=odsout body="Choro_png.html"
      parameters=("OSMSTYLE"="SASMAPNIK");
choro id /nolegend name="Choro";
```

MA_ChoroAnno94.sas and MA_ChoroAnno93.sas

These examples create a choropleth map like the previous example, but add annotate points on top. Figure 15 shows the output from these examples.

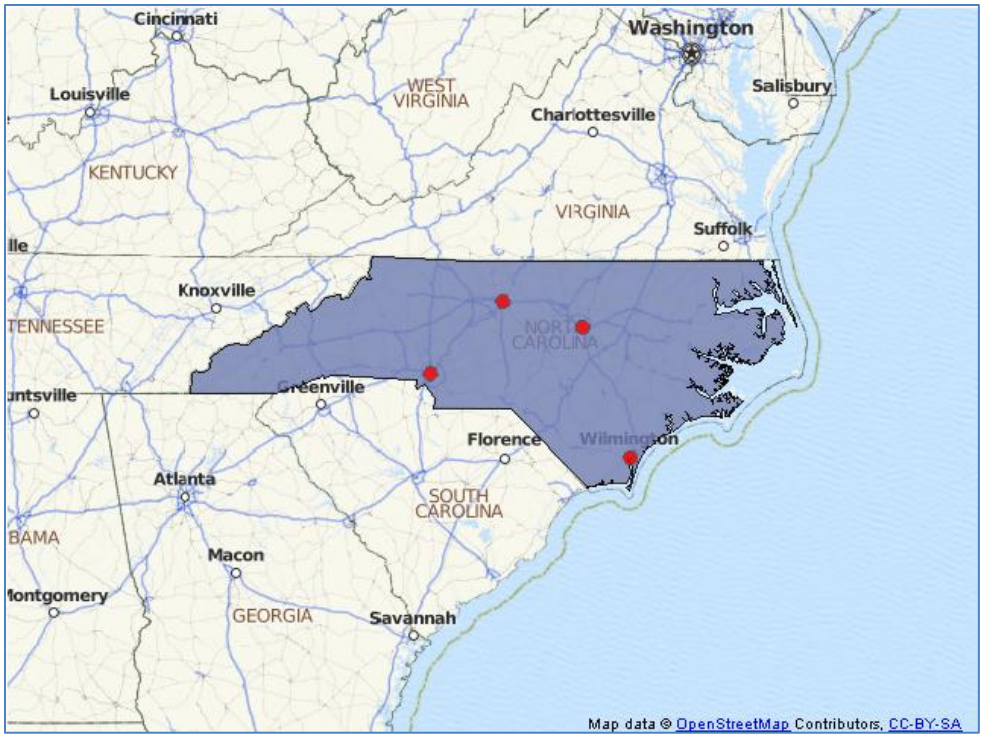

Figure 15. Output from MA_ChoroAnno examples

These programs are identical to the MA_ChoroMap programs except that they add annotate points. This is done by inputting four city names, using PROC GEOCODE to convert these into map coordinates, using the %to_mercator macro to change the projection, and using the %make_dots macro to create an annotate data set with dots for these locations. This annotate data set is then passed to PROC GMAP.

```
/* Convert to mercator projection*/
%to_mercator(proj_map,mymap, 0);

/* Cities you want to show on the map */
data mycities;
  input state $ 1-2 city $ 4-20;
cards;
NC RALEIGH
NC GREENSBORO
NC CHARLOTTE
NC WILMINGTON
;
run;

/* Get the lat/long location of each city */
Proc geocode
  method=city
  data=mycities
  out=gmycities;
run;

/* Set x and y for later use */
data gmycities;
  set gmycities;
  x=long; y=lat;
run;

/* Convert to mercator projection*/
```

```
%to_mercator(proj_anno,gmycities, 0);

/* Create annotate data set of dots at locations */
%make_dots(annocity,proj_anno,'RED',3);
```

MA_PointMap94.sas and MA_PointMap93.sas

These examples create a point map on top of the OSM background without a polygonal map area. Figure 16 shows the output from these examples.

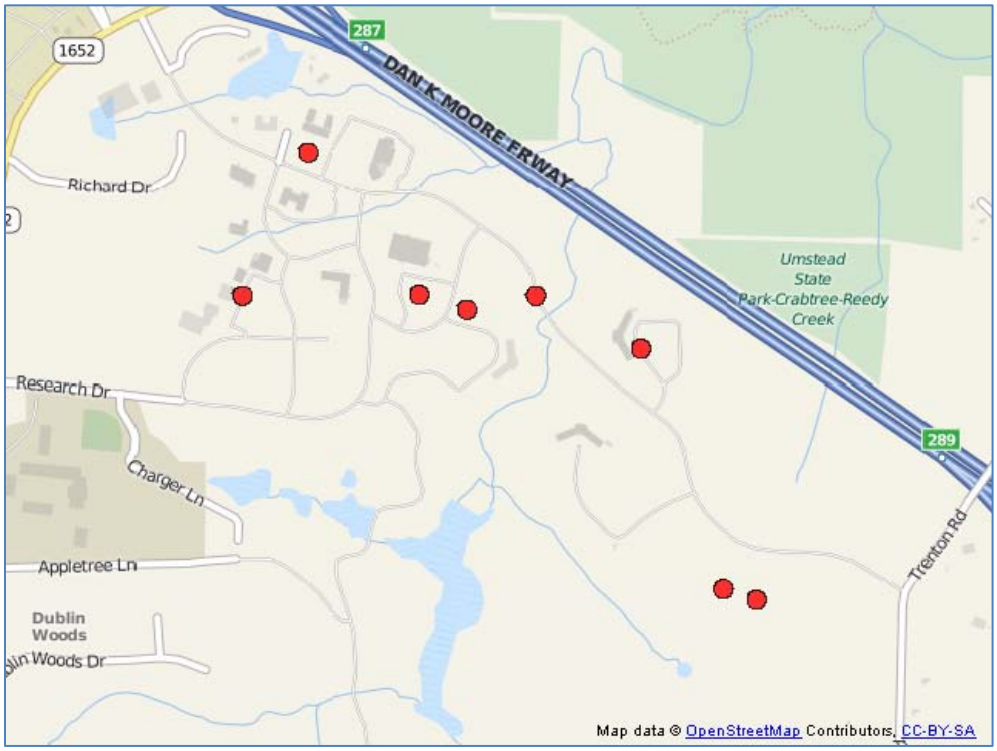

Figure 16. Output from MA_PointMap examples

The MA_ChoroAnno examples create points on the map with annotate data, However, this cannot be done in the MA_PointMap examples. PROC GMAP is designed to always draw a polygonal map and the annotate data is drawn on top of the polygon. In this case, you only want to see points on the OSM background map. In order to meet the requirement to have a polygonal map, the dots for the points must be created as polygonal map data. To do this, read in the point locations and make circle polygons for each point with the %make_circle macro. Because this macro expects coordinates in radians, use the %to_radians macro before calling it and %to_degrees after calling it. The resulting circle polygons are converted to the Mercator projection with the %to_mercator macro. Here is the example DATA step:

```
/* Points for events on campus */
data myspots;
   input lat long incident $21-40 date $;
cards;
35.824302 -78.75533  FLAT TIRE           01Jun12
35.82399  -78.757253 Parking Violation   05Jun12
35.82432  -78.758583 Parking Violation   14Jun12
35.824303 -78.763497 Accident            18Jun12
35.827539 -78.761673 Car theft           22Jun12
35.823102 -78.752403 Break/enter Car     25Jun12
```

```
35.817674 -78.750107 Parking Violation   01Aug12
35.81743  -78.749185 Parking Violation   02Aug12
;
run;

/* add some details */
data dmyspots;
   set myspots;
   x=long; y=lat;
   details=trim(left(date))|| ": " || trim(left(incident));
run;

/* Instead of annotate, draw map circles for each location.*/
/* To do this, convert the degrees to radians as needed for */
/* the making the circles, then convert them back to degrees.*/
%to_radians(rspots, dmyspots);
%make_circle(rmymap,rspots,x,y,.015,'red');
%to_degrees(mymap,rmymap);

/* Convert data to mercator */
%to_mercator(proj_map, mymap,  0);
```

This data set is used by PROC GMAP for the MAP data.

MA_ChoroMapAdv94.sas and MA_ChoroMapAdv93.sas

These examples create a more advanced choropleth map on top of the OSM background. Figure 17 shows the output from these examples.

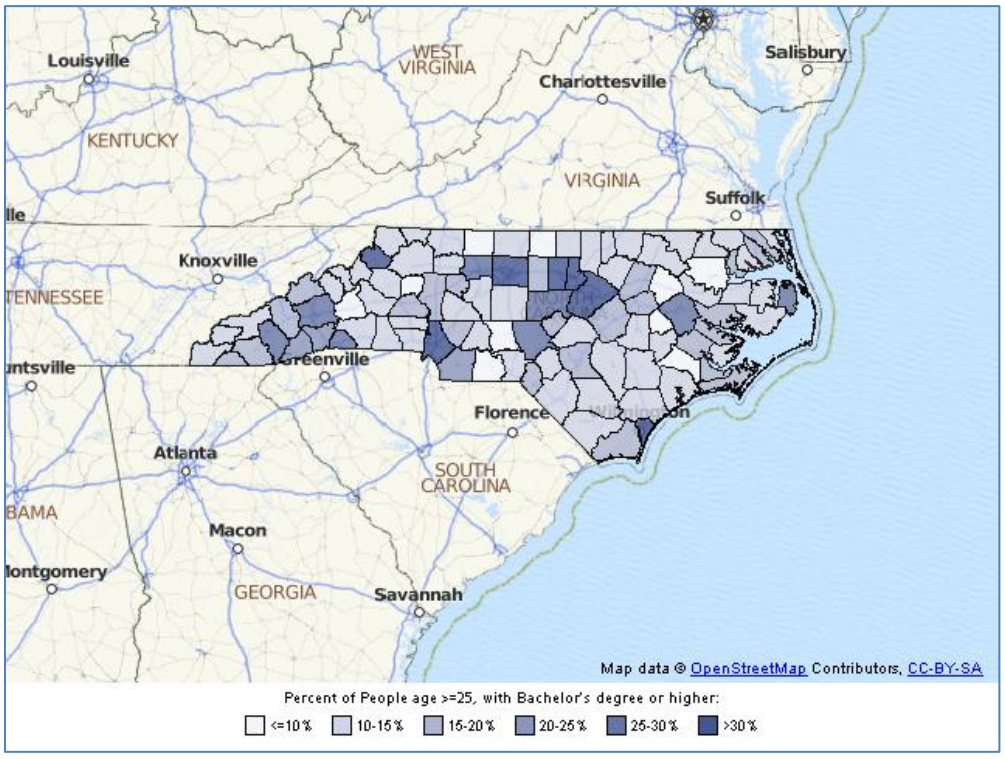

Figure 17. Output from MA_ChoroMapAdv examples

These examples are similar to the MA_ChoroMap examples except that they show data at the county level and add a legend.

CONCLUSION

"Google-like" maps will be used throughout SAS Products. The OSM Annotate Generator and the enhanced JAVA Map Applet provide the tools that enable you to add this functionality to your existing SAS/GRAPH PROC GMAP programs with only simple modifications. You can create interactive maps that pan and zoom or static maps with background maps containing map features.

RESOURCES

Paper and example code downloads.

"Google-like maps in SAS." SAS Presentations at SAS Global Forum 2012, Cary, NC: SAS Institute Inc. Available at http://support.sas.com/rnd/papers

"Proc Geocode: Finding locations outside the US." SAS Presentations at SAS Global Forum 2012, Cary, NC: SAS Institute Inc. Available at http://support.sas.com/rnd/papers

"Tips and Tricks: Using the new SAS Map data sets." SAS Presentations at SAS Global Forum 2012, Cary, NC: SAS Institute Inc. Available at http://support.sas.com/rnd/papers

"Together at Last: Spatial Analytics and SAS® Mapping." SAS Presentations at SAS Global Forum 2012, Cary, NC: SAS Institute Inc. Available at http://support.sas.com/rnd/papers

"SAS Mapping: Technologies, Techniques, Tips and Tricks." Papers from SUGI 28, Cary, NC: SAS Institute Inc. Available at http://support.sas.com/rnd/papers

"Tips and Tricks II: Getting the most from your SAS/GRAPH maps." Papers from SUGI 29, Cary, NC: SAS Institute Inc. Available at http://support.sas.com/rnd/papers

"Tips and Tricks III: More Unique SAS/GRAPH Maps." Papers from SUGI 30, Cary, NC: SAS Institute Inc. Available at http://support.sas.com/rnd/papers

"Tips and Tricks IV: More SAS/GRAPH Map Secrets." SAS Presentations at SAS Global Forum 2009. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/rnd/papers

CONTACT INFORMATION

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

A. Darrell Massengill
SAS Institute Inc.
100 SAS Campus Drive
Cary, NC, 27513
919-677-8000
Darrell.Massengill@sas.com