**Paper SAS3460-2016**

# Creating Custom Map Regions in SAS® Visual Analytics

Angela Hall, SAS Institute Inc., Cary, NC

## ABSTRACT

Discover how to answer the “Where?” question of data visualization by leveraging SAS® Visual Analytics. Geographical data elements within SAS Visual Analytics provides users the capability to quickly map data by countries and regions, by states or provinces, and by the centroid of US ZIP codes. This paper demonstrates how easy it is to map by these elements. Of course, once your manager sees your new maps they will ask for more granular shapes (such as US counties or US ZIP codes). Respond with “Here it is!” Follow the steps provided to add custom boundaries by parsing the shape files into consumable data objects and loading these custom boundaries into SAS Visual Analytics.

## INTRODUCTION

SAS Visual Analytics provides geographical mapping for country and state polygons (shapes) as well as the centroid points for US ZIP codes. With a few modifications, SAS Visual Analytics can provide more granular level mapping.

First, you will learn the simple steps in creating geographical maps in SAS Visual Analytics. Anyone can do this! After getting the mapping experience you will require the step-by-step instructions and tips to load custom regions into your installation of this solution.

Skills required to complete the custom mapping set up include SAS programming and access to the underlying physical folder structure of SAS Visual Analytics. The examples are based on changes required for the United States counties and postal (or ZIP) codes. However, this can be modified to address any custom or country’s geographical data.

## SECTION 1: CREATING GEOGRAPHICAL MAPS

### CHANGING DATA CATEGORIES IN SAS VISUAL ANALYTICS TO GEOGRAPHICAL

Within SAS Visual Analytics, data elements can be classified as Geography. In the following example we convert FIPS code, which is a common US county identifier, to “Subdivision (State, Province) SAS Map ID Values” Geography Type.

Right-click a data element, (#1 in Figure ), click Geography, and select the associated Geography Type (#2 in Figure ).

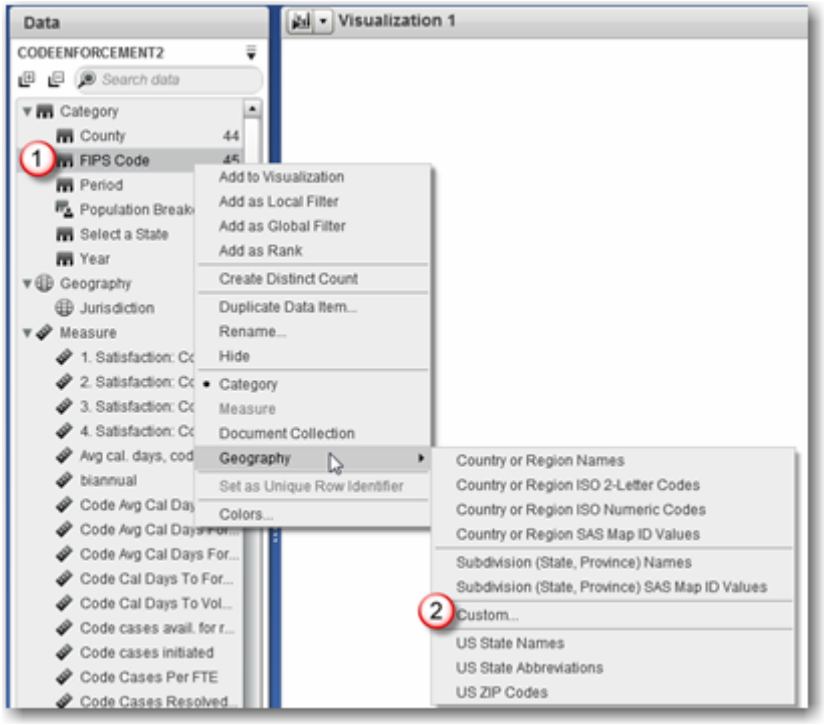


**Figure 1: Changing Data Categories: FIPS Code Example**

Examples of all Geographical Data Elements are provided in the data table below. Additional information about Geographical data elements is available with each version of the SAS Visual Analytics documentation.

- SAS Visual Analytics 6.3: http://support.sas.com/rnd/datavisualization/va63geo/VA63LookupValues.html
- SAS Visual Analytics 7.1: http://support.sas.com/rnd/datavisualization/vageo/71/VA71LookupValues.html

**Table 1: Geographical Data Elements**

| Geography Type | Sample | Additional Information |
|---|---|---|
| Country or Region Names | United States | |
| Country or Region ISO 2-Letter Codes | US | |
| Country or Region ISO Numeric Codes | 840 | |
| Country or Region SAS Map ID Values | US | |
| *Subdivision (State, Province Names)** | Prince William | County Name |
| *Subdivision (State, Province) SAS Map ID Values** | US-51153 | FIPS Code |
| | 20110 | ZIP Code |
| Custom | 38.750949<br>-77.475267 | Latitude<br>Longitude |
| U.S. State Names | Virginia | |
| U.S. State Abbreviations | VA | |
| U.S. ZIP Codes | 20110 | This maps to the centroid (center of the polygon) rather than display the polygon (shape). |

* Both Subdivision Categories require completing the steps in Section 2: Loading Custom Regions.

## MAPPING GEOGRAPHICAL DATA ELEMENTS

To generate a map display in SAS Visual Analytics Explorer, simply drag the geographical data element from the data pane to the visualization window.

Then from the Roles tab (far right window pane), select the **Use Geo Map** link (Figure 2).

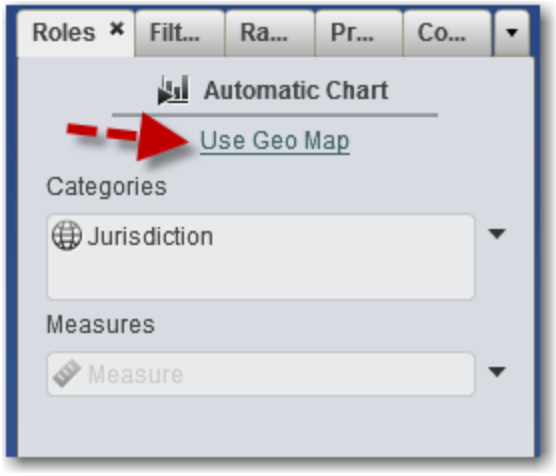


**Figure 2: Roles Tab**

You can now change the display from one Geographical category to another. See Figure .


**Figure 3: Geographic Categories**

You can also modify the map style from Bubbles to Coordinates or Regions (Figure 4).


**Figure 4: Map Style**

## SECTION 2: LOADING CUSTOM REGIONS

### SET UP A CUSTOM MAP LIBRARY

Before beginning this section, set up and add to your autoexec a library for holding custom map data sets.

For this example, the author created a single maps library:

```
libname MAPCUSTM "/customer/warehouse/project/maps";
```

And added it to the application server autoexec program found at:

../Lev1/SASApp/appserver_autoexec_usermods.sas

Note that the SAS Workspace Server must have access to read this custom map library. Also, a restart of the SAS Object Spawner is required to pick up this autoexec modification.

### BACKUP SAS VISUAL ANALYTICS CONFIGURATION

SAS Visual Analytics is shipped with configuration data. Back it up before beginning (or continuing with) any modifications. It is much easier to restore a file than to reconfigure/reinstall the solution.

Copy the SAS Visual Analytics configuration files found at …/940/Lev1/SASApp/Data/valib:

```
libname valib "../940/Lev1/SASApp/Data/valib";

libname bkup "/home/me/valib";

proc datasets library=valib;
   copy out=bkup;
run;
```

Of course, you can also use other tools at the physical level to copy and paste these tables into a backup folder. The important item to remember is that a backup is a best practice and is highly recommended by the author.

### ENABLING COUNTY MAPPING

As mentioned previously, country and state are available within the product. Country is noted as Level 0, while State is Level 1. In order to complete county-level mapping and use the ‘Subdivision ..’ geographical data types, county must be moved up to Level 1.

BEFORE YOU RUN THIS CODE: Back up the SAS Visual Analytics data tables found in ../Lev1/SASApp/Data/valib.

| | |
|---|---|
| Create a custom library 'MapCustm'. | `libname MAPCUSTM "/customer/warehouse/project/maps";` |
| Open a connection to the SAS Visual Analytics map library 'valib'. | `libname valib "../940/Lev1/SASApp/Data/valib";` |
| Move the US Counties (ISO = 840 and Level = 2) up to Level =1.<br><br>This enables the use of the "Subdivision (State, Province) SAS Map ID Value" Geographical Data Element. | `proc sql;`<br>`  update valib.attrlookup`<br>`  set`<br>`  LEVEL=1,`<br>`  ID2="",`<br>`  ID2NAME=""`<br>`where ISO="840" and LEVEL=2;`<br>`quit;` |
| Copy the boundary data sets into our custom map library. | `proc datasets library=MAPSGFK noprint;`<br>`  copy out=MAPCUSTM;`<br>`  select NAMERICA1;`<br>`quit;` |
| Merge NAMERICA Level=1 and Level=2 boundary data sets into the custom map library Level=1 table. | `proc append base=MAPCUSTM.NAMERICA1`<br>`data=MAPSGFK.NAMERICA2;`<br>`run;` |
| Update the centlookup table to point to Level=1, and Level=2 NAMERICA to the custom merged table. | `proc sql;`<br>`update valib.centlookup`<br>`set mapname="MAPCUSTM.NAMERICA1"`<br>`where mapname in`<br>`("MAPSGFK.NAMERICA1","MAPSGFK.NAMERICA2");`<br>`quit;` |

Note that the names of the boundary data sets matche the level of the attrlookup table. This becomes important when loading custom regions.

After completing the steps above, the SAS Visual Analytics equipment will need to be restarted because these lookup tables are loaded during system start up. An example of this command on the author's equipment is:

../940/Lev1/Web/WebAppServer/SASServer12_1/bin/tcruntime-ctl.sh restart

Then verify the use of the "Subdivision (State, Province) SAS Map ID Values" geography type by following the steps in Section 1: Creating Geographical Maps.

**CUSTOM SHAPE FILE PREPARATION**

Shape files are readily available from the US Census. However, they require manipulation prior to loading into SAS Visual Analytics. Shape files can also be derived from tools such as ESRI. However, consider the level of granularity of the display and the ID variables used to join the shape to your data source.

For the US Census, shape files for US ZIP codes can be downloaded directly from: https://www.census.gov/geo/maps-data/data/cbf/cbf_zcta.html. For this example, we will use the US ZIP codes found in cb_2013_us_zcta510_500k.shp to create a SAS data set.

| | |
|---|---|
| The MAPIMPORT procedure is used to convert SHP files into data sets. | `proc mapimport`<br>`datafile="&fileloc./cb_2013_us_zcta510_500k.shp"`<br>`out=zipshp;`<br>`  ID ZCTA5CE10;`<br>`run;` |
| The GREDUCE procedure creates various levels of detail (density) to create the shapes. | `proc greduce data=zipshp out=zip_polygon1;`<br>`  ID ZCTA5CE10;`<br>`run;` |

| | |
|---|---|
| Reduce the level of detail (density) by only including density of 2 or less. This isn't required. However, higher density levels have been known to slow performance of the interface. | `data zip_polygon1;`<br>`set zip polygon1;`<br>`if density <= 2;`<br>`run;` |

Next we will need to add required fields in order for the SAS Visual Analytics geographical mapping capability to work correctly. This must be saved as a new data set in our custom map library MapCustm. Note that since we are adding a Level 1 to the map, this new data set must be suffixed with the same number 1; in this example we named it USZIPPOLY1:

```
%let REGION_LABEL=US Zipcodes;
%let REGION_PREFIX=ZI;
%let REGION_ISO=000;
%let PROVINCE_LABEL=Custom Zipcode;
%let PROVINCE_DATASET=MAPCUSTM.USZIPPOLY1;

data MAPCUSTM.USZIPPOLY1;
set zip_polygon1;
 ID = compress(ZCTA5CE10);
 IDNAME = compress("&PROVINCE_LABEL. " || compress(ZCTA5CE10));
 LONG = X;
 LAT = Y;
 ISO = "&REGION ISO.";
 RESOLUTION = 1;
 LAKE = 0;
 ISOALPHA2 = "&REGION_PREFIX.";
 AdminType = "zippolygons";
  keep ID SEGMENT IDNAME LONG LAT X Y ISO DENSITY RESOLUTION LAKE ISOALPHA2 AdminType;
run;
```

## UPDATING SAS VISUAL ANALYTICS CONFIGURATION

Again, back up the valib SAS library before continuing with this modification.

The two control tables from SAS Visual Analytics (centlookup and attrlookup) must be modified to include these ZIP codes and the reference to this new data set USZIPPOLY1:

| | |
|---|---|
| Included for reference are the macro values that were run in the prior step. They must match what you run in this step. | `%let REGION_LABEL=US Zipcodes;`<br>`%let REGION_PREFIX=ZI;`<br>`%let REGION_ISO=000;`<br>`%let PROVINCE_LABEL=Custom Zipcode;`<br>`%let PROVINCE_DATASET=MAPCUSTM.USZIPPOLY1;` |
| Remove any data in these tables that was previously loaded and will be replaced/added through the following steps. | `proc sql;`<br>`delete * from valib.attrlookup`<br>`where ISONAME = "&REGION_LABEL.";`<br>`delete * from valib.centlookup`<br>`where MAPNAME = "&PROVINCE_DATASET";`<br>`quit;` |
| Add a master record into attrlookup:<br>ID Label = US Zipcodes<br>ID = ZI<br>IDName = US Zipcodes<br>ID1Name = null<br>ID2Name = null<br>ISO = 000<br>ISONAME = US Zipcodes<br>Key = US Zipcodes<br>ID1 = null<br>ID2 = null<br>ID3 = null<br>ID3Name = null<br>Level = 0 | `proc sql;`<br>`insert into valib.attrlookup`<br>`values (`<br>`"&REGION_LABEL.",`<br>`"&REGION_PREFIX.",`<br>`"&REGION_LABEL.",`<br>`"",`<br>`"",`<br>`"&REGION_ISO.",`<br>`"&REGION_LABEL.",`<br>`"&REGION_LABEL.",`<br>`"",`<br>`"",`<br>`"",`<br>`"",`<br>`0 );`<br>`; quit;` |

| | |
|---|---|
| Now insert all the boundary records into attrlookup. | `proc sql;`<br>`insert into valib.attrlookup`<br>`  select distinct`<br>`    IDNAME, /* IDLABEL=State/Province Label */`<br>`    ID,     /* ID=SAS Map ID Value */`<br>`    IDNAME, /* IDNAME=State/Province Name */`<br>`    "&REGION_LABEL.",/* ID1NAME=Country Name */`<br>`    "",             /* ID2NAME */`<br>`    "&REGION_ISO.",  /* ISO=Country ISO Numeric Code */`<br>`    "&REGION_LABEL.",/* ISONAME */`<br>`    trim(IDNAME) \|\| "\|&REGION_LABEL.",   /* KEY */`<br>`    "&REGION_PREFIX.", /* ID1=Country ISO 2-Letter Code */`<br>`    "",     /* ID2 */`<br>`    "",     /* ID3 */`<br>`    "",     /* ID3NAME */`<br>`    1       /* LEVEL (1=state level) */`<br>`from &PROVINCE_DATASET.;`<br>`quit;` |
| Create a master record for the MAPCUSTM.USZIPPOLY1 data table. | `proc sql;`<br>`  insert into valib.centlookup`<br>`   select distinct`<br>`    "&PROVINCE_DATASET." as mapname,`<br>`    "&REGION_PREFIX." as ID,`<br>`    avg(x) as x,`<br>`    avg(y) as y`<br>`from &PROVINCE_DATASET.;`<br><br>`quit;` |
| Insert a record for each unique value of ID (ZIP code) from the MAPCUSTM.USZIPPOLY1 data table. | `proc sql;`<br><br>`insert into valib.centlookup`<br><br>`select distinct`<br>`    "&PROVINCE_DATASET." as mapname,`<br>`    ID as ID,`<br>`    avg(x) as x,`<br>`    avg(y) as y`<br>`from &PROVINCE_DATASET.`<br>`   group by ID;`<br>`quit;` |

After completing the steps above, the SAS Visual Analytics equipment will need to be restarted because these lookup tables are loaded during system start up. An example of this command on the author's equipment is:

../940/Lev1/Web/WebAppServer/SASServer12_1/bin/tcruntime-ctl.sh restart

Then verify the use of the "Subdivision (State, Province) SAS Map ID Values" geography type for ZIP codes by following the steps in Section 1: Creating Geographical Maps. It is also recommended that you retest any other modifications (such as the county-level mapping) to ensure there are no regressions.

## CONCLUSION

Point-and-click mapping provides easy and snazzy reports, while adding custom or more granular polygons (shapes) exponentially increases your ability to answer the specific "Where?" question posed by your boss.

## REFERENCES

Okerson, Barbara. 2013. "Creating Zip Code-Level Maps with SAS®," *Proceedings of the SAS Global Forum 2013 Conference.* Cary, NC. SAS Institute Inc. Available at: http://support.sas.com/resources/papers/proceedings13/214-2013.pdf

## CONTACT INFORMATION

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

Angela M. Hall
Director, Analytic Integration Consulting
SAS Solutions OnDemand
SAS Institute Inc.
100 SAS Campus Drive
Cary, NC 27513
angela.hall@sas.com
http://www.sas.com