Paper DV-03 2015 MWSUG

Sales Force Alignment Visualization with SAS®

Yu(Daniel) Wang, Experis, Cincinnati, OH

ABSTRACT

SAS 9.4, OpenStreetMap(OSM) and JAVA APPLET provide tools to generate professional Google like maps. The zip code boundary data files from U.S. Census Bureau can be freely downloaded and imported into SAS by PROC DATAIMPORT. PROC GEOCODE, with the STREET method, can get the longitude and latitude of a street level address from the data files USM, USS, and USP downloaded from the SAS Maps Online website. A data set of sales force alignment is created after running the sales force optimization model implemented with SAS OPTMODEL. This paper demonstrates the use of all these data sets with SAS to exhibit sales force alignment and target locations on the Google like maps with cities, highways, roads, bodies of water and forests in the background. Each alignment defined area, territory, has its own color. Each territory ID is labelled at its 'center' location. Each sales people office location is marked with his/her name underneath. The boundary of each zip code in a territory is displayed. Each zip code and the number of targets in the zip code are labelled. Different targets could be at the same location. Each target location is dotted with different colors to reflect the different number range of targets at the same address.

INTRODUCTION

Sales organizations often conduct sales force alignment in order to improve sales productivity and revenue, reduce sales effort and cost. The alignment optimization goals include maximizing sales, balancing workload among sales representatives, minimizing travel time, etc. There always have some kind of assumptions or simplifications in the optimization modeling due to complicities of the work such as geography, regulations, local knowledge and data accuracy. In the alignment process, geographical boundaries, territories, are created to assign work to sales representatives. It will be convenient to visualize the alignment on the map for reviewing and making adjustment. This paper uses SAS and data from a pharmaceutical company to display the alignment on a Google like map with cities, highways, roads, bodies of water and forests in the background.

A typical sales force hierarchical structure is territory, district, region and nation. This paper shows a territory level alignment map including a district and all its territories. Other level maps can be generated similarly.

SAS 9.4 provides powerful capabilities in generating maps with PROC GMAP, OpenStreetMap(OSM) and JAVA APPLET. A Google like map will be imported via OSM and JAVA APPLET, and displayed in the background. This paper displays the coloring of user defined territories and labeling at the center of each area. Figure 1 shows a North Carolina district map that includes 6 territories.

On the map, the map title and the legend of number of target in each location is presented. Target office locations are dotted with different colors according to the number of targets in the same location. Sales representative home office locations are marked in blue star with name underneath. Each territory has different colors. Each zip code boundary is displayed. Each zip code with the number of targets in the zip code is labelled. The golden zip code is the territory 'center' location from the optimal design algorithm, which generated balanced territory coverages. Each Territory_ID is labelled at the 'center' location, i.e. the golden zip code. With all above information on the map, it will be very easy to review the alignment and make appropriate adjustments for any factors hard to be included in the optimization model. The typical adjustments include: exchange zip codes in neighboring territories due to the actual sales representative home office locations or the convenience of road access and traffics; reassign a zip code to a neighboring territory because of local knowledge or regulations; find and remove isolated targets or not easily accessible targets; etc. A detailed high resolution territory map could be used by sales representative in the call routing design as well.

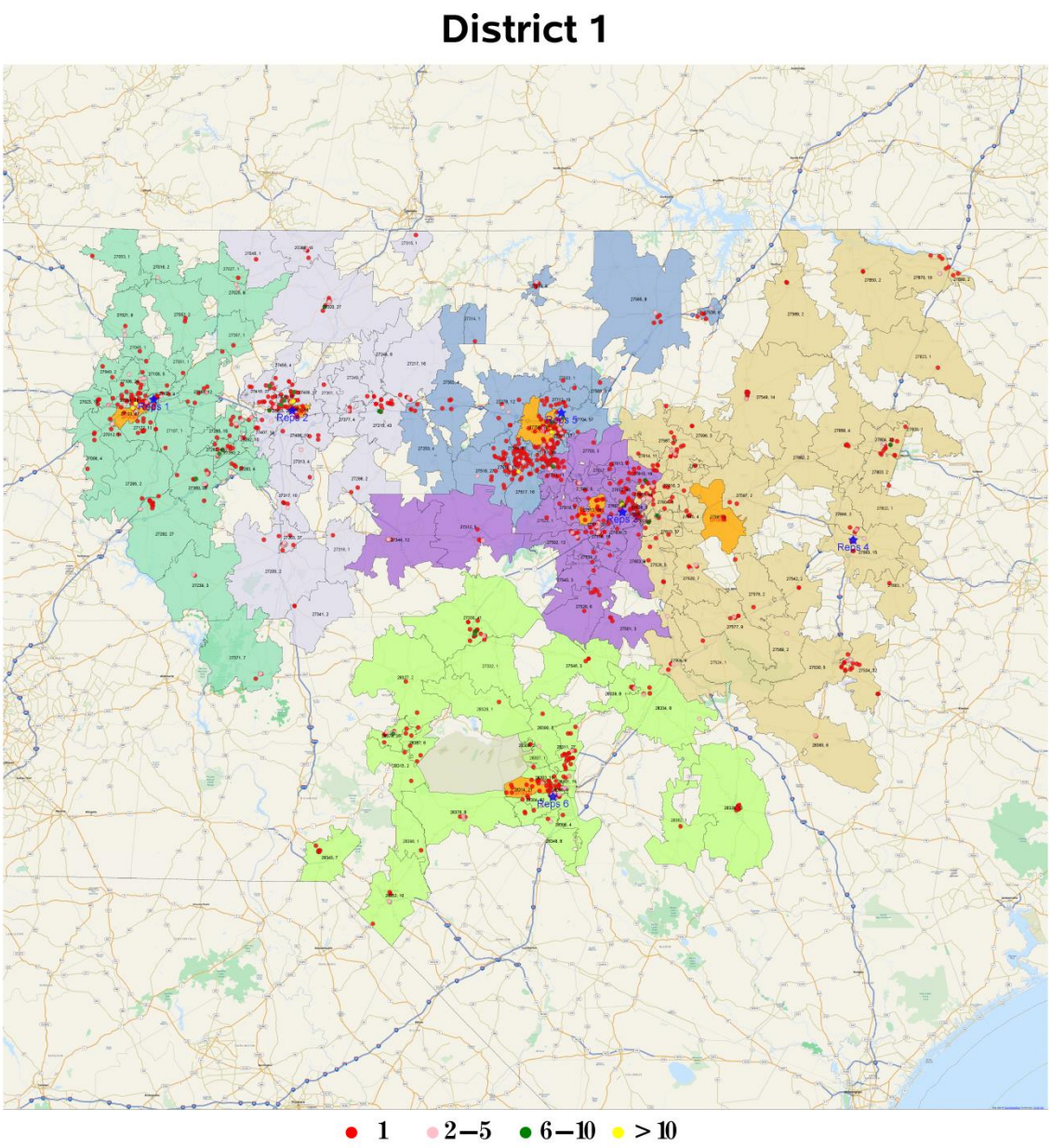
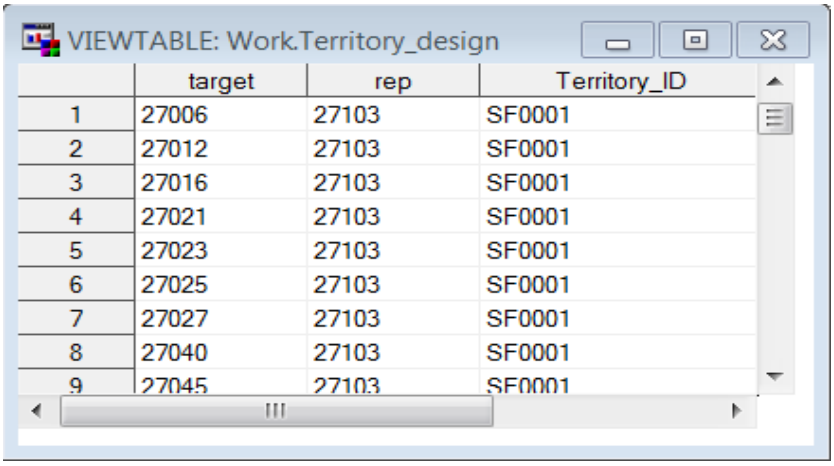


Figure 1

DATA

This paper will focus on the visualization of the optimal sales force alignment. PROC OPTMODEL was used to generate the optimal result. After running the PROC OPTMODEL, the output data set Territory_design will include 3 columns: Target, Rep, Territory_ID, which is looked as Figure 2.

The Target column is the target zip code of the territory. The Rep column is the 'center' zip code of the territory. The Territory_ID column is the sales territory ID.



	target	rep	Territory_ID
1	27006	27103	SF0001
2	27012	27103	SF0001
3	27016	27103	SF0001
4	27021	27103	SF0001
5	27023	27103	SF0001
6	27025	27103	SF0001
7	27027	27103	SF0001
8	27040	27103	SF0001
9	27045	27103	SF0001

Figure 2

Zip Code Data with Boundary

Okerson's paper briefed the history of U.S. zip code, and where and how to get the zip code data with boundary. The U.S. Census Bureau website (www.census.gov) provided TIGERLINE zip code level shapefiles, which renewed every year. The current 2014 zip code data file (t1_2014_us_zcta510.shp) of the entire U.S. country was downloaded from its ftp site (ftp://ftp2.census.gov/geo/tiger/TIGER2014/ZCTA5/), and imported as a SAS data set.

```
PROC MAPIMPORT
DATAFILE = "C:\Documents\t1_2014_us_zcta510.shp"
OUT = uszip;
RUN;
```

The zip code boundary data (segment =1) will be used. In order to create the zip code boundary map correctly, a map index is created to keep the original data order.

```
data uszip;
set uszip(where = (segment = 1));
retain map_index 0;
map_index+1;
run;
```

The Shapefile also includes the longitude and latitude (i.e. the (x, y) coordinate) of an interior point (`intptlon10`) of each zip code, which will be used for the labels of the zip code and the number of targets in the zip code.

```
proc sql;
create table zip_intpt as
select distinct zcta5ce10 as zip, input(intptlon10, 16.) as x,
    input(intptlat10, 16.) as y
from uszip
order by zip;
quit;
```

Street Level Geocode

From SAS 9.2 third maintenance release, PROC GEOCODE has the STREET method to get the longitude and latitude of a street level address. The STREET method tries to match the street name and zip code first. If there is no match, then it attempts to match the street name, city name and two-character zip code. If it fails again, then the ZIP and the CITY method are used. If a street match is found, LONG (or X) and LAT (or Y) coordinate values are interpolated along the street by using the house number. The default street matching SAS data sets (USM, USS, and

USP) are not installed with SAS/GRAPH. These data sets are created from U.S. Census Bureau TIGER/Line shapefiles, and contain address lookup data for the entire United States. After the annual release of new TIGER/Line data, the updated version of USM, USS, and USP data sets can be downloaded from the SAS Maps Online website (http://support.sas.com/rnd/datavisualization/mapsonline/html/home.html). Starting from SAS 9.4, the format of the lookup data sets varies from that of previous releases. PROC GEOCODE in SAS 9.4 cannot read the earlier versions of the lookup data sets. Likewise, PROC GEOCODE in releases prior to 9.4 cannot read the newer lookup data sets.

The Doctors data set includes each physician name, address and other related information such as specialty, rating and ranking.

```
proc geocode
   method=STREET
   data=doctors
   out= doctors
   lookupstreet=usm;
run;
```

The Doctors data set now includes the longitude and latitude of each target after running the PROC GEOCODE. Similarly, the Sales_rep data set contains the longitude and latitude of each sales representative.

SAS 9.4 PROC GMAP

SAS online document (http://support.sas.com/documentation/94/) has the detail explanation of PROC GMAP. The basic syntax of the procedure used in this paper is as follows:

```
proc gmap map=map-data-set
data=response-data-set
anno = annotate-data-set;
id id-variable;
choro response-variable/discrete nolegend;
run;
quit;
```

SAS has set up its own servers to allow its users to create the Google like maps from OpenStreetMaps(OSM). The JAVA MAP APPLET in SAS 9.4 supports OSM background map tiles. To enable the OSM capability, the options are added to the PROC GMAP CHORO statement:

```
choro response-variable/showosm;
```

The syntax works with both JAVA and JAVAIMG devices.

Massengill's paper described in detail how to use PROC GMAP to create the background maps, dots and area colors on the maps. We will use the macro provided by the paper. The following implementation will create the data sets needed for PROC GMAP.

Territory Color

To identify the territory in the map after the optimal territory design output, each territory will have its own color, which is defined by `color_index`, in the macro ***territory_color***. The `color_index` will be used as a response variable in PROC GMAP. All the zip codes in the same territory will have the same `color_index`, except the Rep zip code. The Rep zip code will have the `color_index = 0`, which keeps the same color (golden) for all Rep zip codes. The macro ***territory_color*** is as follow.

```
%macro territory_color(dst);

proc sort data = &dst; by rep target; run;
```

```
data &dst(drop = temp_color_index);
set &dst;
by rep ;
retain color_index 0;
retain temp_color_index 0;

if first.rep then do;
   color_index +1;
   temp_color_index = color_index;
   end;
if target = rep then do;
   color_index = 0; output;
   color_index = temp_color_index;
   end;
else output;
run;
%mend;
```

Annotation

The locations of targets are dotted on the map. The numbers of target at the same address are formed in 4 groups: 1 target, 2 – 5 targets, 6 – 10 targets and more than 10 targets. The corresponding colors are Red, Pink, Green and Yellow respectively. The macro ***target_dots*** has two inputs and one output. The `dst_targ` data set includes the target address (street name and number, city, state, zip).The `dot_size` controls the size of the dot of the targets in the map. The macro output data set, `out_dst`, is a projected annotation data set of targets used by PROC GMAP. The macros ***%to_mercator*** and ***%make_dots*** used in the paper are from Massengill's paper.

```
%macro target_dots(dst_targ, dot_size, out_dst);

data &dst_targ;
set &dst_targ;
retain temp_index 0;
temp_index+1;
run;

proc sql;
create table &dst_targ._cnt as
select distinct y, x, address, city, state, zip, count(distinct temp_index) as
count
from &dst_targ
group by y, x, address, city, state, zip
order by y, x, address, city, state, zip;
quit;

data &dst_targ._1 &dst_targ._2 &dst_targ._3 &dst_targ._4;
set &dst_targ._cnt;
if count = 1 then output &dst_targ._1;
else if 1 < count <=5 then output &dst_targ._2;
else if 5 < count <=10 then output &dst_targ._3;
else if count > 10 then output &dst_targ._4;
run;

%to_mercator(proj_&dst_targ._1,&dst_targ._1, 0);
%to_mercator(proj_&dst_targ._2,&dst_targ._2, 0);
%to_mercator(proj_&dst_targ._3,&dst_targ._3, 0);
%to_mercator(proj_&dst_targ._4,&dst_targ._4, 0);

%make_dots(Anno_1,proj_&dst_targ._1,'RED', &dot_size);
%make_dots(Anno_2,proj_&dst_targ._2,'PINK', &dot_size);
%make_dots(Anno_3,proj_&dst_targ._3,'GREEN', &dot_size);
%make_dots(Anno_4,proj_&dst_targ._4,'YELLOW', &dot_size);
```

```
data &out_dst;
set Anno_1 Anno_2 Anno_3 Anno_4;
run;
%mend;
```

Label

The Territory_ID will be labeled at Rep zip code. Each zip code and the number of targets in the zip code separated by comma will be labeled at the interior point of the zip code by the data set `zip_intpt`. A blue star denotes the sales representative home location with his/her name below. Figure 3 shows a zoomed map of Figure 1 as a part of a territory alignment.

The macro ***territory_label*** generates the data sets needed for PROC GMAP that shows the zip code boundary, the label of zip code and the number of targets in the zip code separated by a comma. The input data sets `dst_terr` and `dst_targ` are for Territory_design and Targets data sets correspondingly. The parameters `zip_label_size` and `territory_label_size` determine the label sizes of zip code and territory_id respectively. The position of a label is controlled by the POSITION option. The `position = '5'` places the zip code label at the interior point of the zip code, while `position = '8'` locates the territory_id label under the interior point (see Figure 3). The `out_dst` is the output data set which will be included in anno data set in PROC GMAP. The `proj_territory` is the response data set used in PROC GMAP.

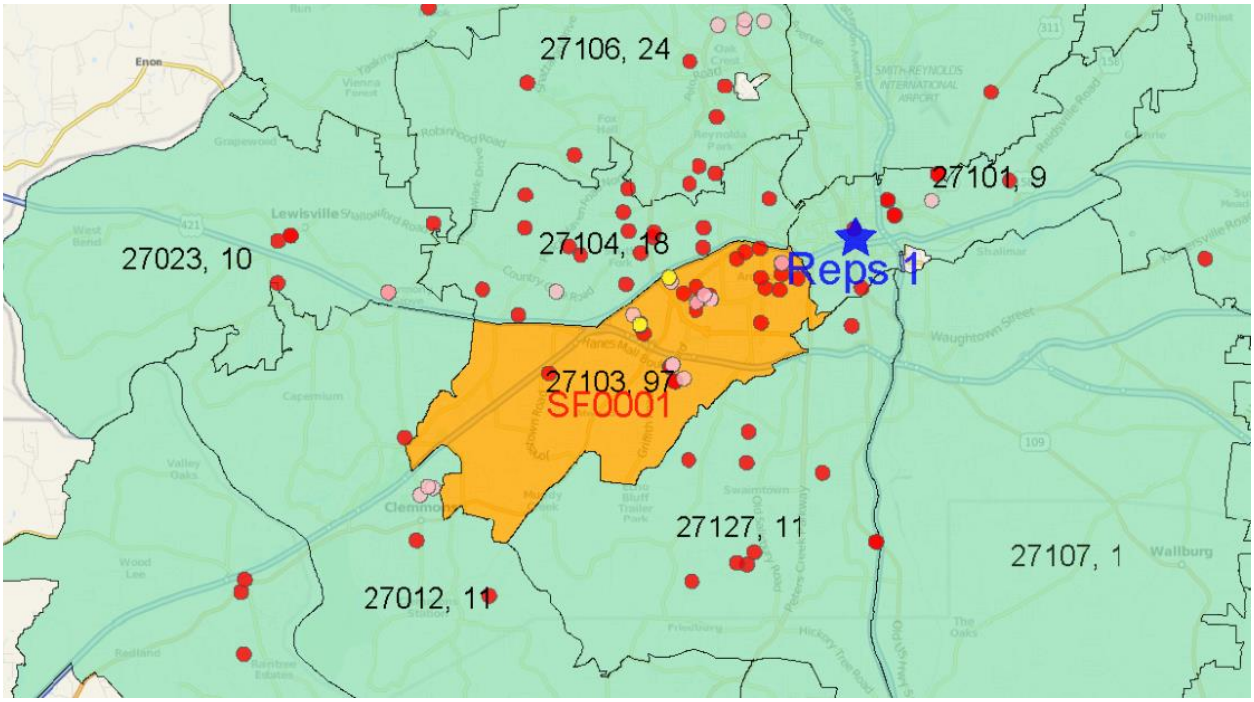


Figure 3

```
%macro territory_label(dst_terr, dst_targ, zip_label_size, territory_label_size,
out_dst);

%territory_color(&dst_terr);

proc sql;
create table territory as
select a.*, b.zip, b.x, b.y
from &dst_terr a, zip_intpt b
where a.zip = b.zip
```

```
order by a.rep, a.target;
quit;

%to_mercator(proj_territory, territory,0);

data &dst_targ;
set &dst_targ;
retain temp_index 0;
temp_index+1;
run;

proc sql;
create table zip_targ_cnt as
select a.target as zip,
      count(distinct b.temp_index) as targ_cnt
from &dst_terr a, &dst_targ b
where a.target = b.zip
group by a.target
order by a.target;
quit;

proc sort data = proj_territory; by zip; run;

data &out_dst(drop = color_index rep);
merge proj_territory (keep = x y color_index zip rep) zip_targ_cnt;
by zip;
text = trim(zip)|| ', ' || strip(put(targ_cnt, 8.));
function = 'label';
xsys = '2'; ysys ='2'; when = 'a'; hsys = '3'; position = '5';
color = 'black'; style = 'centbi'; size = &zip_label_size;
run;

proc sort data = proj_territory; by rep zip; run;

data territory_label(drop = color_index);
set proj_territory (where = (color_index = 0)
                    keep = x y color_index zip territory_id
                    rename = (territory_id = text));

function = 'label'; xsys = '2'; ysys ='2';
when = 'a'; hsys = '3'; position = '8';
color = 'black'; style = 'centbi';
size = &territory_label_size;
run;

data &out_dst;
set &out_dst territory_label;
run;
%mend;
```

The macro ***salesman_label*** will include all the sales representatives in the given radius of the territories in determining if the territory is properly designed, and assigning the right sales representative to an appropriate territory. The input `salesman_addr` data set includes the sales representative name, address, city, state, zip code and the longitude and latitude of the address. The parameters `salesman_label_size` and `salesman_name_size` determine the size of the blue star and sales representative name respectively. The radius controls only the representatives within the radius of the nearest zip code of the territory showing up on the map. For example, radius = 50, means all the sales representatives within the 50 miles of the radius of the closest zip code of the territory will show up on the map.

```
%macro salesman_label(salesman_addr, dst_terr, radius, salesman_label_size,
salesman_name_size, out_dst);

data &salesman_addr;
```

```
set &salesman_addr;
retain sales_index 0;
sales_index+1;
run;

proc sql;
create table sales_rep as
select distinct a.*, min(ZIPCITYDISTANCE(a.zip, b.target)) as distance
from &salesman_addr a, &dst_terr b
group by a.index
having min(ZIPCITYDISTANCE(a.zip, b.target)) <= &radius
order by a.index;
quit;

%to_mercator(proj_sales_rep, sales_rep, 0);

data &out_dst;
set proj_sales_rep (keep = x y name address city state zip);
length function style color $ 8 position $ 1 text $ 20;
retain xsys ysys "2" hsys "3" when "a";

function="symbol"; style="marker"; text="V"; color="BLUE";
size= &salesman_label_size;
output;

function = 'label'; text = trim(name);
xsys = '2'; ysys ='2'; when = 'a'; hsys = '3'; position = '8';
color = 'BLUE'; style = 'SWISSXB'; size = &salesman_name_size; output;
run;

proc sort data = &out_dst; by name; run;
%mend;
```

Map

The macro ***map_dst*** generates the map data set needed in PROC GMAP.

```
%macro map_dst(dst_terr, out_dst);

proc sql;
create table mkt_zip as
select a.*, b.color_index
from uszip a, &dst_terr b
where a.zcta5ce10 = b.target
order by a.map_index;
quit;

data mkt_zip(keep = long lat x y zip color_index);
set mkt_zip ;
long = x; lat = y;
format Zip $5.;
zip = left(zcta5ce10);
run;

%to_mercator(&out_dst, mkt_zip, 1);

proc sort data = &out_dst; by zip ; run;
%mend;
```

There are two different ways to generate a map with PROC GMAP in SAS 9.4 from Massengill's paper. JAVA Map Applet method is used in this implementation. The `output_path` is the path of the output map file. The `file_name`

and png_name are the html and .png name of the map file. The proj_map, proj_target and proj_anno are the data sets used in PROC GMAP. The xpls and ypls are x and y pixels of the map.

```
%macro map_output(output_path, file_name, png_name, proj_map, proj_target,
proj_anno, xpls, ypls);

filename odsout "&output_path";

goptions reset=all;
goptions gunit=pct htitle=16pt ftitle="albany amt/bold"
         htext=25pt ftext="albany amt/bold";
goptions xpixels=&xpls ypixels=&ypls;
ods html path=ODSOUT file = "&file_name..html";
goptions device  = JAVAIMG;

pattern;

pattern1 v=me repeat=1 c=CXFFAA00;   *Gold*;
pattern2 v=me repeat=1 c=CX99E5BC;   *Very Light Green*;
pattern3 v=me repeat=1 c=CXDEDDED;   *Bluish White*;
pattern4 v=me repeat=1 c=CXAC74D9;   *Very Light Violet*;
pattern5 v=me repeat=1 c=CXE8D898;   *Cream*;
pattern6 v=me repeat=1 c=CX90B0D9;   *Very Light Greenish Blue*;
pattern7 v=me repeat=1 c=CXC0FF81;   *Lime*;
pattern8 v=me repeat=1 c=CXAEADD9;   *Very Pale Blue*;
pattern9 v=me repeat=1 c=CXCB74D9;   *Very Light Purple*;
pattern10 v=me repeat=1 c=CXD8BFD8;  *Thistle*;
pattern11 v=me repeat=1 c=CXE0A860;  *Tan*;
pattern12 v=me repeat=1 c=CX7FFDD4;  *Aquamarine*;
pattern13 v=me repeat=1 c=CXFFB6C1;  *Light Pink*;

proc gmap map=&proj_map data=&proj_target anno = &proj_anno all;
id zip;
choro color_index/ discrete nolegend name = "&png_name" showosm;
run;
quit;

ods html close;
%mend;
```

Title and Legend

Finally, the legend of dots of target and the title of the map will be added by using PROC GSLIDE. The input_path and output_path are the input and output of the image files. The image and name are the input image and output files name. The title is the title of the map. The xpls and ypls are the x and y pixels of the image.

```
%macro Map_TitleLegend(input_path, image, output_path, name, title, xpls, ypls);

data legend; length function text $24; xsys='3'; ysys='3';
  hsys='3'; when='a';
  function='PIE'; line=0; angle=0.0; rotate=360.0; size=0.5;
  style='SOLID';

  color='RED    '; x=35; y=3; output;
  color='PINK   '; x=42; y=3; output;
  color='GREEN  '; x=50; y=3; output;
  color='YELLOW '; x=58; y=3; output;

  function='LABEL'; size=2; style='centbi'; position='1';
  color='BLACK ';
  text='1    '; x=38; y=1.2; output;
```

```
  text='2-5  '; x=47; y=1.2; output;
  text='6-10 '; x=56; y=1.2; output;
  text='>10  '; x=63; y=1.2; output;
run;

data image;
   length function $ 8 style $ 8;

   function = 'move';
   xsys = '3'; ysys = '3';
   x = 5; y = 5; output;

   function = 'image';
   imgpath  = "&input_path.\&image..png";
   x = 95; y = 95; style = 'centbi'; output;
 run;

data anno_dst;
set image legend;
run;

ODS LISTING CLOSE;
ODS HTML  path="&output_path" body="&name..html" style=sasweb;
goptions device  = png;
goptions xpixels=&xpls ypixels=&ypls;
goptions gunit=pct ftitle="albany amt/bold"
         ftext="albany amt/bold";

title h= 2 "&title";

proc gslide anno=anno_dst  imagestyle = tile
name = "&name";
run;
quit;

ods html close;
title;
%mend;
```

MAIN PROGRAM

The following program is used to call all the macros mentioned above to generate the map shown in Figure 1.

```
%target_dots(target, 0.2, anno);
%territory_label(territory_design, target, 0.4, 0.4, terr_label);
%salesman_label(sales_rep, terr, 50, 0.5, 0.6, rep_label);

data anno;
set anno terr_label rep_label;
run;

%map_dst(territory_design, proj_map);

%map_output(C:\Documents, District_1, dist_1_png, proj_map, proj_territory, anno,
10000, 10000);
%Map_TitleLegend(C:\Documents, dist_1_png, C:\Documents, District_1_Title,
District 1, 11000, 11000);
```

The resolution can be enlarged by choosing large x and y pixels, and zoom in the map to see the very detail of the map. The PNG image file format is easily distributed among the sales representatives, management team and website.

CONCLUSION

The techniques used in this paper are applicable to zip code data files for any U.S. places. The SAS code in this paper can be modified to create Google like maps that show the area of interest, dot the point of interest with label and useful data, and color the user defined area such as territory, district and region.

REFERENCES

Darrell Massengill, "'Google-Like' Maps in SAS®", SAS Global Forum 2013, paper 377-2013.
http://support.sas.com/resources/papers/proceedings13/377-2013.pdf

Barbara Okerson, "Creating ZIP Code-Level Maps with SAS®", SAS Global Forum 2013, paper 214-2013.
http://support.sas.com/resources/papers/proceedings13/214-2013.pdf

Ted Conway, "Get to Your Points: Using SAS® to Build Google Maps", SAS Global Forum 2010, paper 052-2010.
http://support.sas.com/resources/papers/proceedings10/052-2010.pdf

CONTACT INFORMATION

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

Yu (Daniel) Wang, Ph.D.
Experis
4445 Lake Forest Drive, Suite 470
Cincinnati, OH 45242
Phone: 513-808-9077
Email: yu.wang@experis.com