

Bordering on Success with PROC GMAP in SAS®: Utilizing Annotate Datasets to Enhance Your Maps

Kathryn Schurr, M.S., Spectrum Health-Healthier Communities, Grand Rapids, MI
Jonathan Wiseman, Spectrum Health-Healthier Communities, Grand Rapids, MI

ABSTRACT

PROC GMAP is a valuable tool for visualizing data. GMAP gives users the means to geographically represent their data so that audiences can relate to the data on multiple levels at once. One limitation of PROC GMAP that has been identified is its inability to allow for multiple geographic regions (such as zip code areas, census tracts and counties) to be plotted all at once, especially if they do not have the shape of a polygon. This paper will discuss how PROC GMAP in conjunction with the %MAPLABEL and %ANNOTATE macros (1) creates a map in SAS® that annotates various locations on the map as well as providing an (x,y)-coordinate grid corresponding to the shapefile being used, (2) gives the user the capability to move singular labels to a more desirable location, and (3) draws borders of differing map regions regardless of shape.

INTRODUCTION

Utility of Maps

Maps have been used for all of recorded history. Maps are the epitome of representing data for their sheer ability to capture one's attention and to divulge copious amounts of information in a single glance. By looking at the map given below, an individual is able to take in information within the first few seconds of viewing it.

Figure 1



The information in this map that someone first registers might be any one of the following:

- The United States of America
- Bordering Countries
- Larger bodies of water including size and depth
- Significant mountain ranges including directions and size
- State borders and names
- Important roadways including direction and distance

All the information above was viewed, taken in, and registered within a very small slice of time. After looking at the map further, one may notice that there are state capitals designated by a small red star and also that larger cities are labeled. Maps are so intricate and can contain so many different levels of data that the capability of mapping data for an enterprise or other type of organization is almost a necessity.

Spectrum Health

Spectrum Health is a not-for-profit health system located in western Michigan. It is presently comprised of 10 hospitals, 190 other service sites and employs over 20,000 people including 975 advanced practice providers and physicians. Its mission is to improve the health of the communities that it serves. The Healthier Communities Department of Spectrum Health is dedicated to operating or providing funding and technical support for a variety of programs that seek to improve the health of individuals with chronic diseases, to prevent and detect illness, especially in school-age populations, to serve the health needs of the underserved population of Kent County, Michigan, and to eliminate or at least mitigate in a multitude of ways barriers to effective health care.

Given the relatively large geographic area that Spectrum Health serves, it is important to be able to identify areas where individuals are at elevated risk of disease within the community so that the organization is better able to plan and offer assistance. PROC GMAP in SAS® allows Healthier Communities to look at various sub-populations based on zip codes, census tracts, or counties to determine where certain at-risk populations reside.

The programs that Healthier Communities supports are operated across western Michigan with a reach far beyond the brick and mortar walls that house the department. Because individuals readily identify with zip code boundaries rather than with census tracts it is best to view the data in the most accessible manner possible. Because Healthier Communities' programs span many zip codes and multiple counties, it was deemed necessary to develop a technique to present both zip code boundaries and county boundaries on a single map. This would allow viewers to understand the reach of the department's programs and to view multiple levels of the data.

GETTING THE SHAPEFILES

In order to create a map using PROC GMAP in SAS®, two shapefiles containing the required geographic regions are needed. There are many different sources for accessing shapefiles. We were able to retrieve our shapefiles from the US Census Website and the Michigan Department of Technology, Management & Budget (DTMB) website. The census tracts are from the 2012 TIGERLINE® shapefiles located within the Census website and the zip code shapefile was found on the DTMB website.

On the US Census website, the shapefiles that are available contain many different geographic representations. Choosing the correct shapefile for your project is very important. Because we want to create a zip code map with a county border annotation, we need to make sure that the two shapefiles we have chosen are compatible. This is somewhat time-consuming and takes some digging; however, the end result is well worth it. Because census tracts are divided by counties, a census tract shapefile allows for a bit more flexibility than solely a county map. Also, singular county maps pose a problem if there are some counties that extend into oceans or lakes that span more than one state or country (Lake Michigan for example). This will be explained further in the **DIFFICULTIES** section. In the next section we will go over the MAPIMPORT procedure and how to determine if two different types of shapefiles are compatible.

CREATING THE BASE MAP

Importing and Evaluating the Shapefiles

Once the shapefiles have been downloaded to a computer and saved in a folder, they must be imported into SAS®. The code below shows the PROC MAPIMPORT procedure which recognizes that the data being imported is a shapefile and handles it accordingly.

```
PROC MAPIMPORT  
    DATAFILE = "C:\Documents and  
                Settings\Katie\Desktop\zt26_d00.shp"  
    OUT = Zip_Codes;  
RUN;
```

The DATAFILE = statement specifies the path and the file name of the shapefile that is of interest. The OUT = statement tells SAS® the name of the dataset where the imported shapefile should be saved.

This same code will be applied to the census tract shapefile. Once both shapefiles have been imported into SAS® we can look at the variables that correspond to each dataset. This can be accomplished by running a PROC CONTENTS procedure on both datasets. Having an

understanding of these variables will allow us to restrict the shapefile to only the section of the map that we are interested in. After performing the PROC CONTENTS, we find that both data sets contain the variables X and Y. These variables are used in plotting the maps that were imported. The entire variable list from PROC CONTENTS is located in the Appendix to this paper.

Now that both the shapefiles contain the same plotting variables, it is time to compare them to see if the shapefiles are compatible. We can do this by applying a PROC MEANS to the two datasets to inspect and compare the X and Y variables. We want to look at the spread of the X and Y variables and see if they are similar between the two datasets. The code and output are given below.

```
PROC MEANS DATA = Zip_Codes  MIN P10 Q1 MEDIAN Q3 P90 MAX;
  VAR x y;
RUN;

PROC MEANS DATA = Census_Tracts  MIN P10 Q1 MEDIAN Q3 P90 MAX;
  VAR x y;
RUN;
```

Zip Codes

Variable	Minimum	Lower Quartile	Median	Upper Quartile	Maximum
X	-90.4183920	-86.4450810	-85.3061835	-84.1107130	-82.4099770
Y	41.6961180	42.9524600	44.3667010	45.9636170	47.4845870

Census Tracts

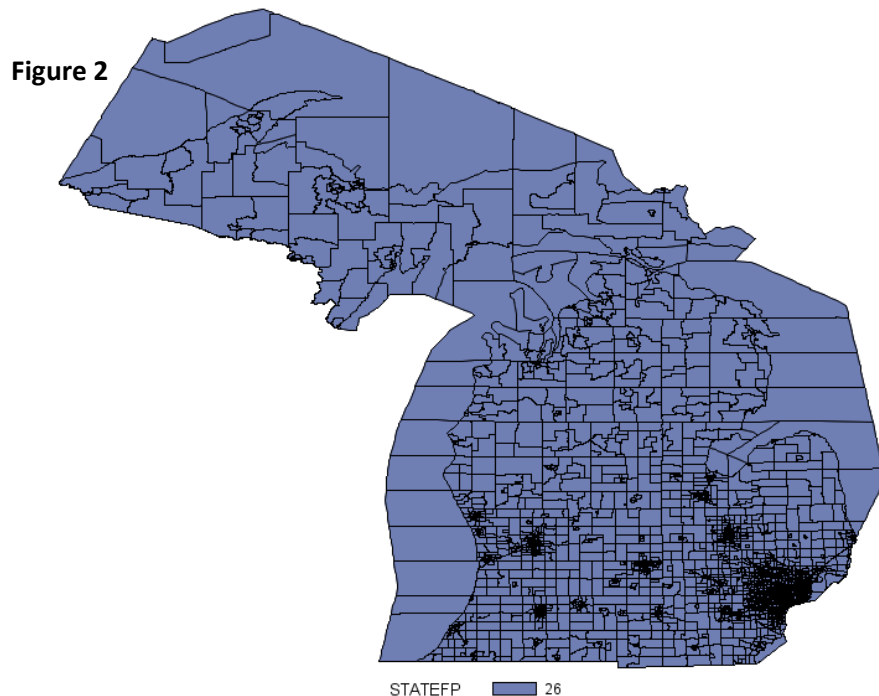
Variable	Minimum	Lower Quartile	Median	Upper Quartile	Maximum
X	-90.4183920	-85.8649030	-84.6009980	-83.4942240	-82.1229710
Y	41.6961180	42.4767380	43.0782620	44.8111560	48.3060630

By reviewing the above output, it can be seen that the values of X and Y between the two variables are very close. This is what we would expect whenever two datasets are compatible. There are many different ways that shapefiles can be plotted. Some of the shapefiles scale down or scale up their X and Y coordinates. We just need to ensure that the two shapefiles we have chosen are roughly on the same scale; otherwise, combining two geographic regions would not work well.

Creating the base map

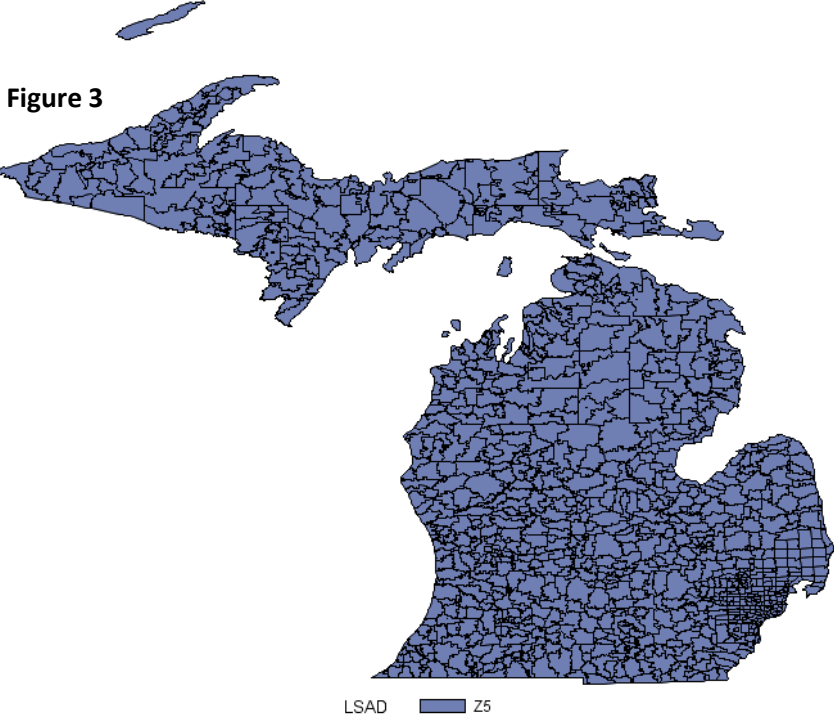
Now that we know the maps are compatible we should graph them to view their layout and identify any problems that they may have such as incomplete portions or unexpected lines. To make a base map we can use PROC GMAP and specify a choropleth option that is constant throughout the entire dataset. Below is the code and output regarding the census tract dataset.

```
PROC GMAP DATA = Census_Tracts MAP = Census_Tracts;  
  ID namelsad;  
  CHORO statefp;  
RUN;
```



In the previous example the DATA = statement instructs SAS® to use that dataset in conjunction with the MAP = dataset to create the map. These datasets do not have to be the same; however, they both need to include the ID variable. The ID variable given is the identification variable by which the data will be plotted. The CHORO statement identifies which levels of data are to be represented in the map via a choropleth. In this case, to simplify the output, we chose the State FIPS code that is constant throughout the data set so that the base

map would be created. We can follow the same methodology to plot the zip code map as well which is given below.



Sub-Setting the Area of Interest

Suppose we want to limit the map to only the zip codes that lie partially or completely within Kent County. To restrict the zip codes we must first find out which zip codes are located in the Kent County borders. This can be done via internet search. We find that the zip codes located within Kent County are the following:

- 48809, 48838, 49301, 49302, 49306, 49315, 49316, 49319, 49321, 49330, 49331, 49341, 49343, 49345, 49418, 49428, 49501, 49502, 49503, 49504, 49505, 49506, 49507, 49508, 49509, 49512, 49514, 49519, 49525, 49534, 49544, 49546, 49548

We can limit the output of SAS® to the zip codes listed above via a simple data step. The code below limits first the Zip_Codes dataset to the Zip codes of interest, and then the Census_Tracts dataset to the county of interest.

```

DATA Kent_Zips;
  SET Zip_Codes;
  IF ZCTA in ('48809', '48838', '49301', '49302', '49306',
'49315', '49316', '49319', '49321', '49330', '49331',
'49341', '49343', '49345', '49418', '49428', '49501',
'49502', '49503', '49504', '49505', '49506', '49507',
'49508', '49509', '49512', '49514', '49519', '49525',
'49534', '49544', '49546', '49548');

```

```

RUN;

```

```

DATA Kent_Census;
  SET Census_Tracts;
  IF COUNTYFP = '081';

```

```

RUN;

```

The variable COUNTYFP pertains to the County FIPS Code that is used to identify counties within states. In order to see which FIPS Code pertains to Kent County, Michigan we used the following website: <http://www.epa.gov>. Now that we have restricted the data to only include areas of interest, we apply the same methodology as before and graph the basic map with only one choropleth option. The resulting maps are below.

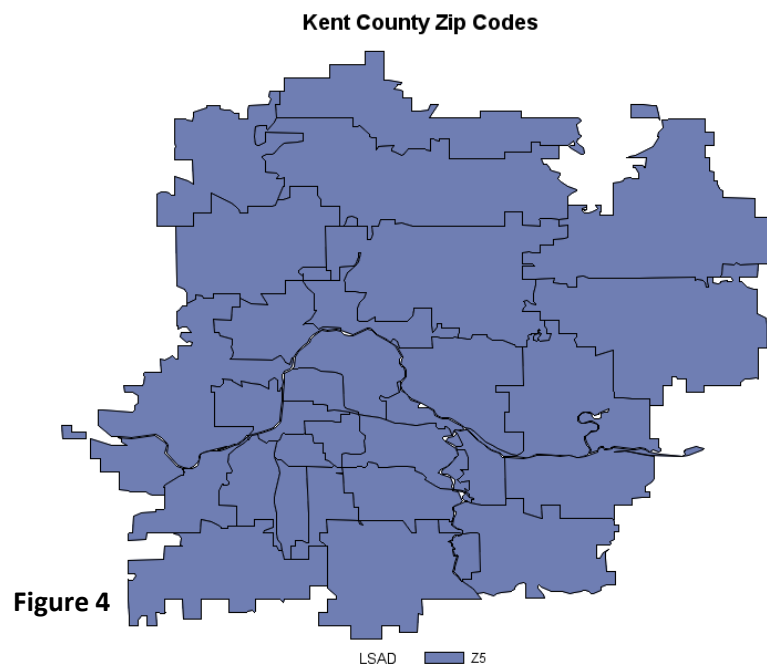


Figure 4

Kent County Census Tract

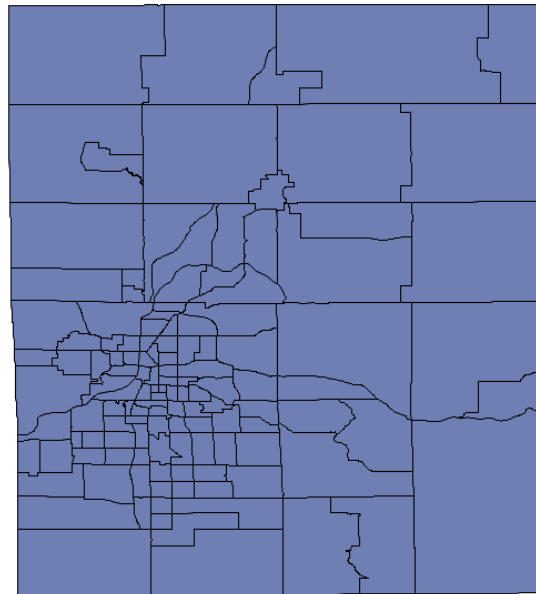


Figure 5

Using PROC GREMOVE

As previously mentioned we only require a map with the borders of Kent County. In order to accomplish this, we will use PROC GREMOVE on the subsetted Kent County Census Tract data used in the previous map. This will create a dataset that contains only the information to graph the outermost border of Kent County. The code and output are below.

```
PROC GREMOVE DATA = Kent_Census OUT = Kent_Border;  
  BY countyfp;  
  ID tractce;  
RUN;  
  
PROC GMAP DATA = Kent_Border MAP = Kent_Border;  
  ID countyfp;  
  CHORO countyfp;  
  TITLE "Kent County Border";  
RUN;
```


Kent County Border

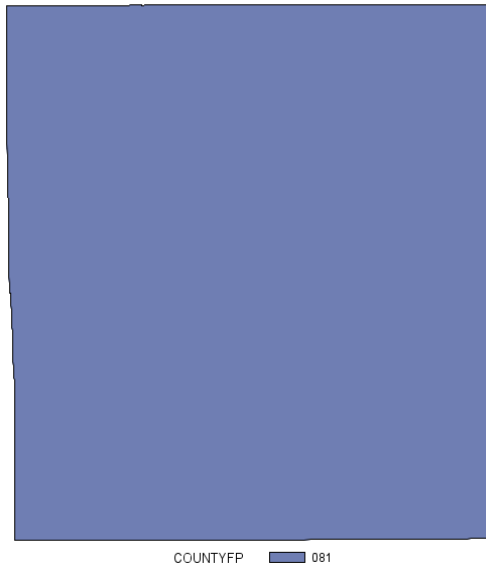


Figure 6

Mapping Our Data

The dataset that we need to map consists of the patients enrolled in one of the programs of Healthier Communities. There are 1,299 patients enrolled in the Core Health Program. This program is designed to help uninsured and underserved patients in Kent County manage their chronic disease. The patients' information is stored in a dataset called "Core_Health_Data," including the zip code corresponding to each patient's residence. More specifically, we would like to graph the percentage of our program participants by zip code. We will use PROC GMAP to create a choropleth map which will shade each zip code region a specific color based on the percentage of patients enrolled.

In the dataset Kent_Zips the zip code is stored in the variable ZCTA as a five-digit character variable. In the Core_Health_Data the zip code variable, Zip_Code, is stored as a numeric variable. In order to use PROC GMAP, we need these two variables to be of the same format, and this conversion is accomplished in a DATA step shown below.

```
DATA Core_Health_Data;  
  SET Core_Health_Data;  
  FORMAT ZCTA $5.;  
  ZCTA = Zip_Code;  
  KEEP CPI DIAGNOSIS ZCTA;  
RUN;
```

Once we have the zip code variables in the same format, we can begin to prepare our data for analysis. Because we would like to look at the percentage of participants in each zip code, we can use a PROC FREQ procedure to calculate these values for us and output them into a dataset. The code is below.

```

PROC FREQ DATA = Core_Health_Data NOPRINT;
    TABLES ZCTA / NOCUM OUT = Zip_Freq;
RUN;

```

The first five observations of the resulting dataset are as follows.

Obs	ZCTA	COUNT	PERCENT
1		8	.
2	48809	1	0.07716
3	48838	1	0.07716
4	49301	6	0.46296
5	49302	1	0.07716

Note that these data contain the percentage of the total patients in each zip code, so we will be using this dataset to create the choropleth map in conjunction with the dataset Kent_Zips.

Now that we have created the data we would like to map, we must devise a format that we would like the data to appear in. The following code will create a format “percentgroup” which will separate the percentage into five groups. In addition, the PATTERN option will specify the color to be used in the map for each level.

```

PROC FORMAT;
    VALUE PercentGroup Low - 1    = 'Less than 1%'
    1 <- 5                    = 'Between 1% and 5%'
    5 <- 10                   = 'Between 5% and 10%'
    10 <- 15                  = 'Between 10% and 15%'
    15 <- HIGH                = 'Over 15%';

```

```
RUN;
```

```

PATTERN1 C = LightGreen;
PATTERN2 C = Cyan;
PATTERN3 C = Yellow;
PATTERN4 C = Orange;
PATTERN5 C = Red;

```

Once the formats and patterns have been created, we must sort the data prior to creating the map. Both the datasets Kent_Zips and Zip_Freq must be sorted by the ZCTA variable. The code below does this.

```

PROC SORT DATA = Kent_Zips;
    BY ZCTA;
RUN;

```

```

PROC SORT DATA = Zip_Freq;
  BY ZCTA;
RUN;

```

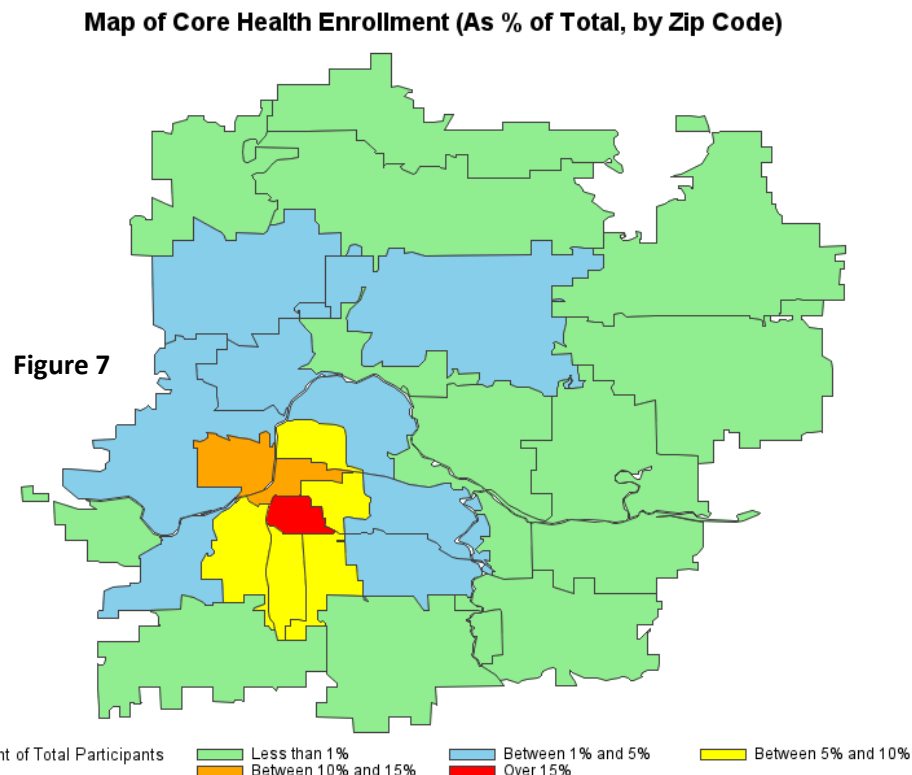
We are now ready to create the Core Health Program map with PROC GMAP using the following code.

```

PROC GMAP DATA = Zip_Freq MAP = Kent_Zips;
  ID ZCTA;
  FORMAT Percent PercentGroup.;
  CHORO Percent / DISCRETE;
  TITLE 'Map of Core Health Enrollment (As % of Total, by Zip Code)';
RUN;

```

The variable used in the ID statement specifies how the map area is to be split up (i.e., whether we are using zip codes, counties, states, countries, etc.). As mentioned previously, this variable must be formatted identically in both the dataset and shapefile being used. The CHORO statement specifies the type of map to be created, and the variable specified tells SAS® which variable is to be represented. The DISCRETE option on this statement ensures that every level in our formatted response (in this case, “percent”) is created, even if the dataset contains no observations at this particular level. The resulting map from this code is shown below.



Now that the first data map is created, we can see that this instance depicts the desired information; however, we require that the map show sufficient detail so that this information is as readable and comprehensive as possible. The next section describes how using the %MAPLABEL and %ANNOMAC macros can make a good map better.

USING THE ANNOTATE DATA SET

Once the data map has been created, we can use the %MAPLABEL macro (available on the SAS® website) to add labels to the zip codes. While the previous map is functional and displays the desired information, we will now present some alternative ways to enhance this graphic using an annotate dataset.

First, we will need a dataset containing a sorted list of the zip codes. The Zip_Freq data set created in the previous section will suffice. We also need to make sure our shapefile is sorted by the variable we wish to label.

```
PROC SORT DATA = Kent_Zips;  
  BY ZCTA;  
RUN;
```

Now that we have the needed elements, we can properly call the %MAPLABEL macro using the following syntax.

```
%ANNOMAC;  
%MAPLABEL (enrollment_zip_codes, zip_freq, anno_zip_label, zcta,  
  zcta, font=Arial Black, color=grey, size=2, hsys=3);
```

The arguments needed for this macro are a shapefile dataset (enrollment_zip_code), a dataset (zip_freq), the name of the output dataset (anno_zip_label), the ID variable (zcta), the variable containing the text of the label (also zcta in this case), the font of the label, and the color of the label. Values for size and hsys can also be specified, but they are not necessary and will default to 2 and 3, respectively. We have shown them here with their default values only to illustrate that they may be used as arguments for the macro as well. Note that we must first call the %ANNOMAC macro. This makes the annotate macros available for use in SAS® and must be run prior to using any of the annotate macros (such as %MAPLABEL). The outputted dataset “anno_zip_label” contains an observation for each unique zip code found in “zip_freq”. The first five observations are shown below for reference.

Obs	position	xsys	ysys	when	hsys	text	function	style	color	size	x	y	ZCTA
1	5	2	2	a	3	48809	label	Arial Black	grey	2	-85.2486	43.0579	48809
2	5	2	2	a	3	48838	label	Arial Black	grey	2	-85.2524	43.2088	48838
3	5	2	2	a	3	48851	label	Arial Black	grey	2	-84.9558	42.9495	48851
4	5	2	2	a	3	49301	label	Arial Black	grey	2	-85.5291	42.9641	49301
5	5	2	2	a	3	49302	label	Arial Black	grey	2	-85.3944	42.8237	49302

Table 4

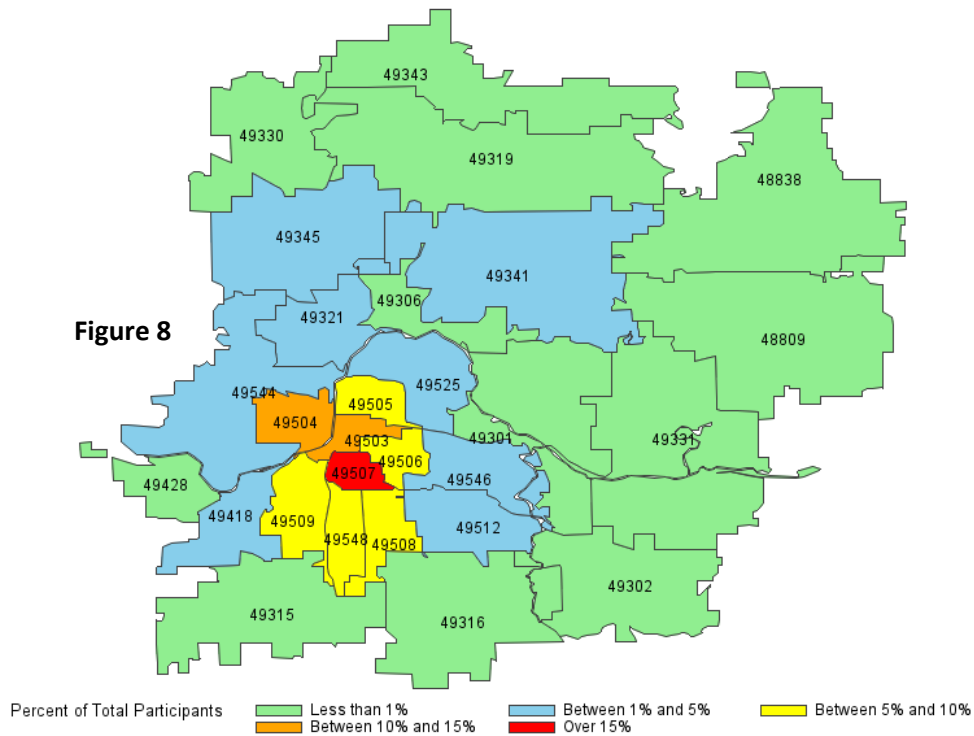
Here we see that each observation contains a value of “label” for the variable “function” signifying what action is to be carried out, and a pair of X and Y coordinates signifying where the label will go. Other functions such as “draw” and “move” can be used in the annotate data set, and we will show how these are used in the APPLYING THE COUNTY BORDER section. The variable “text” determines what the label says, while the variables “style”, “color”, and “size” dictate how it will appear. The “hsys” variable specifies the units for the “size” variable, “xsys” and “ysys” specify the coordinate system for X and Y, “position” specifies the placement and alignment for the label, and “when” determines whether the label will be drawn before or after the map graphics are output. Using the same syntax as we did for the base choropleth map, we add the ANNOTATE option on the CHORO statement to apply our labels, and the resulting map is shown below.

```

PATTERN1 C = lightgreen;
PATTERN2 C = cyan;
PATTERN3 C = yellow;
PATTERN4 C = orange;
PATTERN5 C = red;

PROC GMAP DATA = zip_freq MAP = enrollment_zip_codes ALL;
  ID zcta;
  FORMAT percent percentgroup.;
  CHORO percent / DISCRETE ANNOTATE = anno_zip_label;
  TITLE 'Map of Core Health Enrollment (As % of Total, by Zip
  Code)';
RUN;
```

Map of Core Health Enrollment (As % of Total, by Zip Code)



Applying Gridlines

For the most part, the %MAPLABEL macro does a good job producing clear and readable labels. However, where there are areas dense with numerous zip codes, the labels can be obstructed by neighboring labels or border lines (e.g., the yellow, orange, and red portions of the above map). Waterways and other geographic features can also obscure the labels. For example, we see that the label for zip code 49331 is blocked by the Grand River in the above map. We can use a DATA step on the annotate data set to manually change the X and Y coordinates for that particular label so it shows up more clearly. To do this, we require a better feel for the ranges of the coordinates in our map, and this necessity is precisely what led us to create the %GRIDLINES macro. This macro will essentially take a shapefile, determine the minimum and maximum coordinates for both X and Y, and split this range into the number of equal-sized intervals specified. At each of these intervals, a line will be drawn. The full code for the macro and the proper syntax to call the macro are given below.

```

%MACRO gridlines(dataset, levels, line_size, line_style,
    line_color, text_size, text_style, text_color);

PROC MEANS DATA = &dataset NOPRINT;
    VAR x y;
    OUTPUT OUT = means MAX = x_max y_max MIN = x_min y_min;
RUN;

DATA _NULL_;
    SET means;
    CALL SYMPUT('xmax', x_max);
    CALL SYMPUT('ymax', y_max);
    CALL SYMPUT('xmin', x_min);
    CALL SYMPUT('ymin', y_min);
    CALL SYMPUT('x_incr', ROUND((x_max - x_min) / &levels,
    0.01));
    CALL SYMPUT('y_incr', ROUND((y_max - y_min) / &levels,
    0.01));
RUN;

DATA draw_x_lines;
    LENGTH function text $10 color $10 position $ 1 line 8.
    size 8. style $20;
    RETAIN xsys ysys '2' hsys '3' when 'a' position '5' line
    &line_style style &text_style;
    current_x = ROUND(&xmin, &x_incr) + &x_incr;
    DO WHILE (current_x <= &xmax);
        function = 'move';
        x = current_x;
        y = &ymin;
        OUTPUT;
        y = &ymax;
        function = 'draw';
        color = &line_color;
        size = &line_size;
        OUTPUT;
        y = y + &y_incr / 10;
        function = 'label';
        color = &text_color;
        size = &text_size;
        text = current_x;
        OUTPUT;
        current_x + &x_incr;
    END;

```

```

KEEP function color line size position xsys ysys hsys when
    x y text style;

RUN;

DATA draw_y_lines;
LENGTH function text $10 color $10 position $ 1 line 8.
size 8. style $20;
RETAIN xsys ysys '2' hsys '3' when 'a' position '5' line
&line_style style &text_style;
current_y = ROUND(&ymin, &y_incr) + &y_incr;
DO WHILE (current_y <= &ymin);
    function = 'move';
    y = current_y;
    x = &xmin;
    OUTPUT;
    x = &xmax;
    function = 'draw';
    color = &line_color;
    size = &line_size;
    OUTPUT;
    x = x - &x_incr / 4;
    y = y - &y_incr / 10;
    function = 'label';
    color = &text_color;
    size = &text_size;
    text = current_y;
    OUTPUT;
    current_y + &y_incr;
END;

KEEP function color line size position xsys ysys hsys when
    x y text style;

RUN;

DATA gridlines;
SET draw_x_lines draw_y_lines;
RUN;

%MEND;

%gridlines(enrollment_zip_codes, 10, 0.4, 1, "lightgrey", 2,
    "Arial", "grey");

```

From this example, we see that the needed elements to call %GRIDLINES are a shapefile, the number of levels to split the map into, the size of the lines, the style of the lines, the color

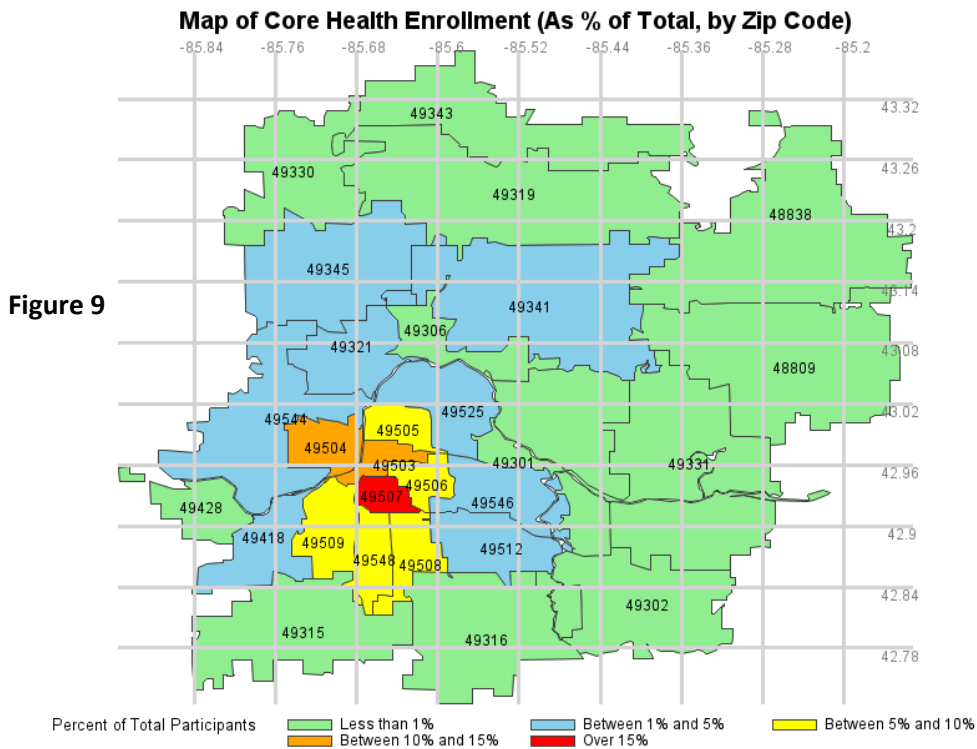
of the lines, the size of the text, the font of the text, and the color of the text (in that order). A list of line and font styles, as well as what colors are available in SAS® can be found on the SAS® website (links to the specific pages can be found in the References section). The annotate dataset “gridlines” created from this macro can then be combined with the previous annotate dataset. We can then use this in PROC GMAP as before, and the resulting map is shown below.

```

DATA anno_label_grid;
    SET gridlines anno_zip_label;
RUN;

PROC GMAP DATA = zip_freq MAP = enrollment_zip_codes ALL;
    ID zcta;
    FORMAT percent percentgroup.;
    CHORO percent / DISCRETE ANNOTATE = anno_label_grid;
    TITLE 'Map of Core Health Enrollment (As % of Total, by Zip Code) ';
RUN;

```



MOVING INDIVIDUAL LABELS

We can now better estimate the X and Y coordinates, and thus straightforwardly adjust specific labels. Again looking at the label for zip code 49331, we will consider moving this label to the left so that it is not blocked by the path of the river. Using the newly made gridlines, we can now see that an X-coordinate of about -85.375 degrees would be a sufficient move. We use a DATA set to change this.

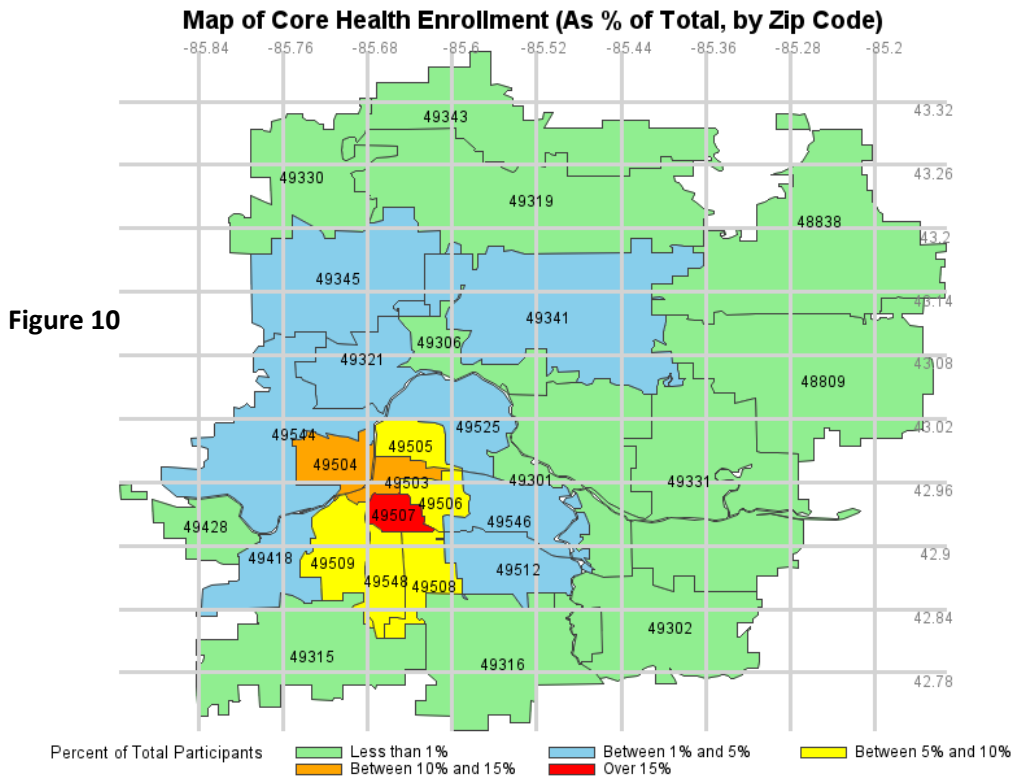
```

DATA anno_zip_label;
  SET anno_zip_label;
  IF text = '49331' THEN x = -85.375;
RUN;

DATA anno_label_grid;
  SET gridlines anno_zip_label;
RUN;

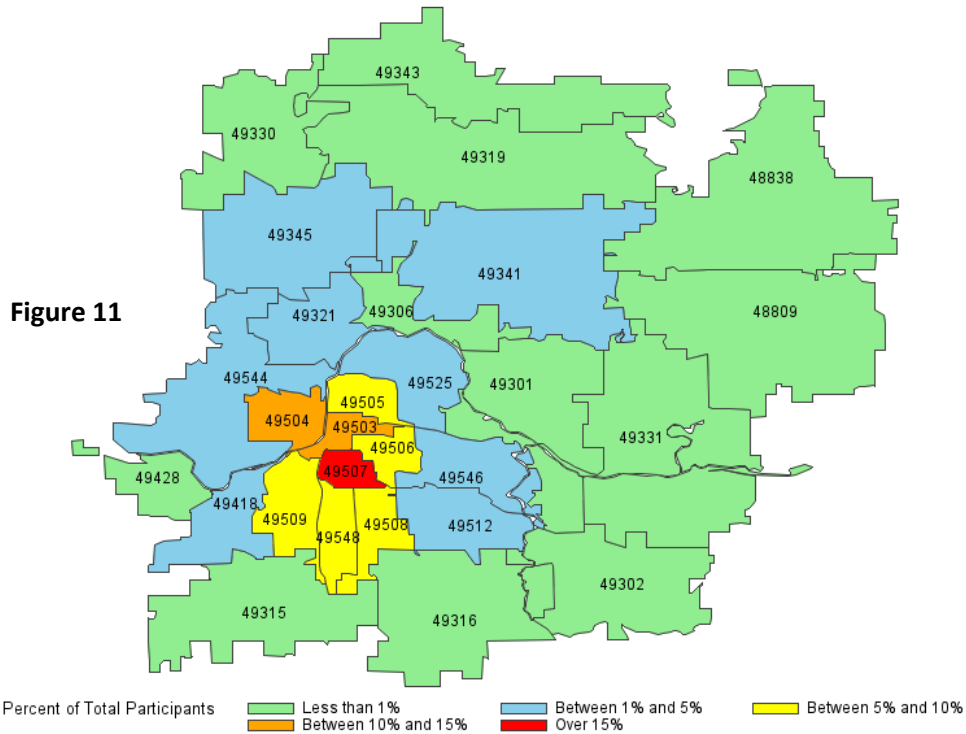
PROC GMAP DATA = zip_freq MAP = enrollment_zip_codes ALL;
  ID zcta;
  FORMAT percent percentgroup.;
  CHORO percent / DISCRETE ANNOTATE = anno_label_grid;
  TITLE 'Map of Core Health Enrollment (As % of Total, by Zip
  Code) ';
RUN;

```



The result is that the label has indeed been moved and is easier to read now. We proceed in a similar fashion moving the labels to the best positions for clarity. Note that since the edits were made to the annotate data set “anno_zip_label” as opposed to “anno_zip_grid”, we are able to produce the following map with the labels altered and without gridlines.

Map of Core Health Enrollment (As % of Total, by Zip Code)



Applying the County Border

The boundaries for zip code regions do not typically align well with county borders. Therefore, in a map such as the one above, it may be beneficial to include the county’s border for reference. We will now discuss how to do this by incorporating the Kent County outline created previously and using that outline to create another annotate data set.

In the annotate dataset created by the %ANNOMAC macro and %MAPLABEL macro, an observation with “draw” as a value for the “function” variable will draw a line of the specified style from the previous observation’s X and Y coordinates to the coordinates of the current observation. Using “move” as the value for the “function” variable in the first observation will position us at the proper (X,Y) coordinates to make sure the subsequent lines are being drawn to and from the correct locations. Note that we have used similar variables before when the annotate data set was created for the zip code labels, except that in this case we included a “line” variable to specify the type of line that is to be drawn. Also, we do not have to specify values for “style” or “text” when using the “move” or “draw” functions. The code below will create the required annotate dataset from the dataset for Kent County created previously.

```

DATA countyline;
LENGTH function color $ 8 position $ 1 line 8.;
RETAIN xsys ysys '2' hsys '3'
      when 'a' position '5' line 33 size 0.6 color 'black';
SET Kent_Census;
IF _N_ = 1 THEN function = 'move';
IF _N_ NE 1 THEN function = 'draw'; OUTPUT;
KEEP function color line size position xsys ysys hsys when
x y;
RUN;

```

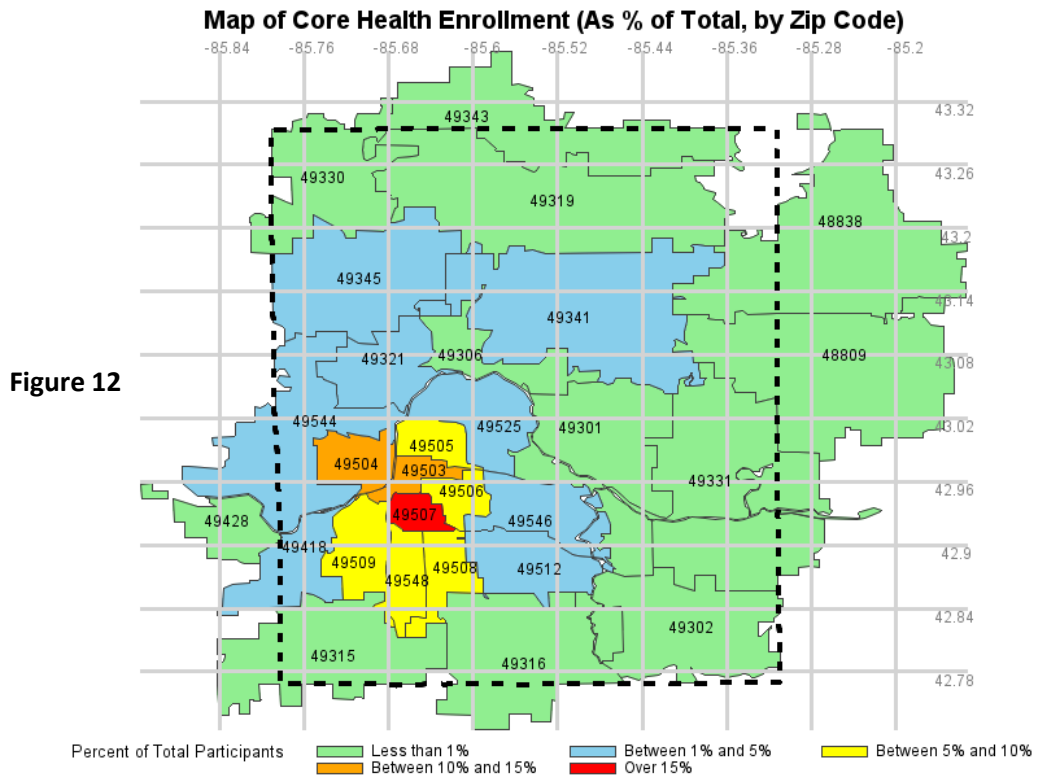
We can now affix this to our other annotate dataset and use PROC GMAP to produce a similar graph as before, only now with the Kent County border (shown with the gridlines):

```

DATA anno_zip_county;
SET gridlines countyline anno_zip_label;
RUN;

PROC GMAP DATA = zip_freq MAP = enrollment_zip_codes ALL;
ID zcta;
FORMAT percent percentgroup.;
CHORO percent / DISCRETE ANNOTATE = anno_zip_county;
TITLE 'Map of Core Health Enrollment (As % of Total, by Zip Code)';
RUN;

```



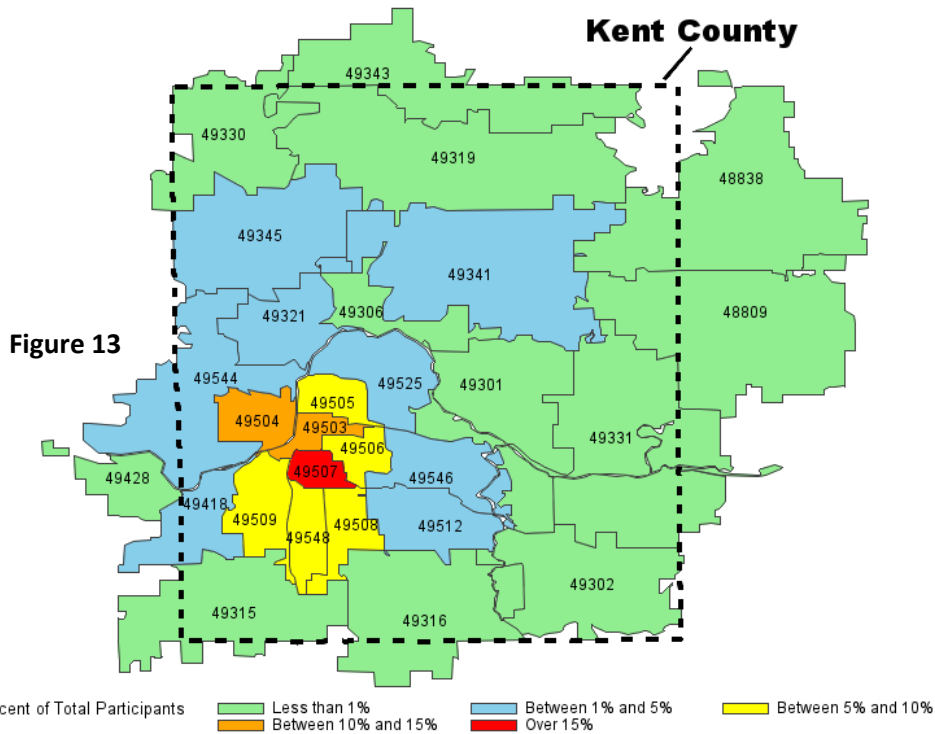
This map can be taken one step further by adding a label to the Kent County border (with a line connecting the label to the actual border). This can be done by making another annotate data set with three observations: one to “move” to the correct coordinates, a second to “draw” the line connecting the border and label, and a third to make the actual “label”. Using the gridlines provided to estimate which coordinates should be used, the DATA step to create the county label is shown below.

```
DATA countylabel;  
  LENGTH function color $ 8 position $ 1;  
  RETAIN xsys ysys '2' hsys '3' when 'a' position '5';  
  function = 'move'; size = 2; x = -85.325; y = 43.3; OUTPUT;  
  function = 'draw'; size = 0.5; color = 'Black';  
  x = -85.3; y = 43.33; line = 1; OUTPUT;  
  function = 'label'; size = 4; color = 'Black';  
  style = 'Arial Black'; x = -85.3; y = 43.355; text = 'Kent  
  County'; OUTPUT;  
RUN;
```

Finally, we can concatenate this to the annotate data sets for the zip code labels and the county border data to produce our finished graphic.

```
DATA anno_all;  
  SET countyline countylabel anno_zip_label;  
RUN;  
  
PROC GMAP DATA = zip_freq MAP = enrollment_zip_codes ALL;  
  ID zcta;  
  FORMAT percent percentgroup.;  
  CHORO percent / DISCRETE ANNOTATE = anno_all;  
  TITLE 'Map of Core Health Enrollment (As % of Total, by Zip  
  Code)';  
RUN;
```

Map of Core Health Enrollment (As % of Total, by Zip Code)



DIFFICULTIES

Borders Over Water

Michigan is different from many states because a majority of its boundaries lie across water instead of land. This is where using a census tract shapefile as opposed to a county shapefile is helpful. When importing the shapefile for the census tracts, the areas of the counties that cross bodies of water are given different census tract designations. To produce the map below, we first restrict the map to a county that contains a water boundary. For this example, we chose Ottawa County which happens to be adjacent to Kent County. Then we had PROC GMAP do a choropleth map of the census tracts for Ottawa County. The code and the resulting map are as follows.

```

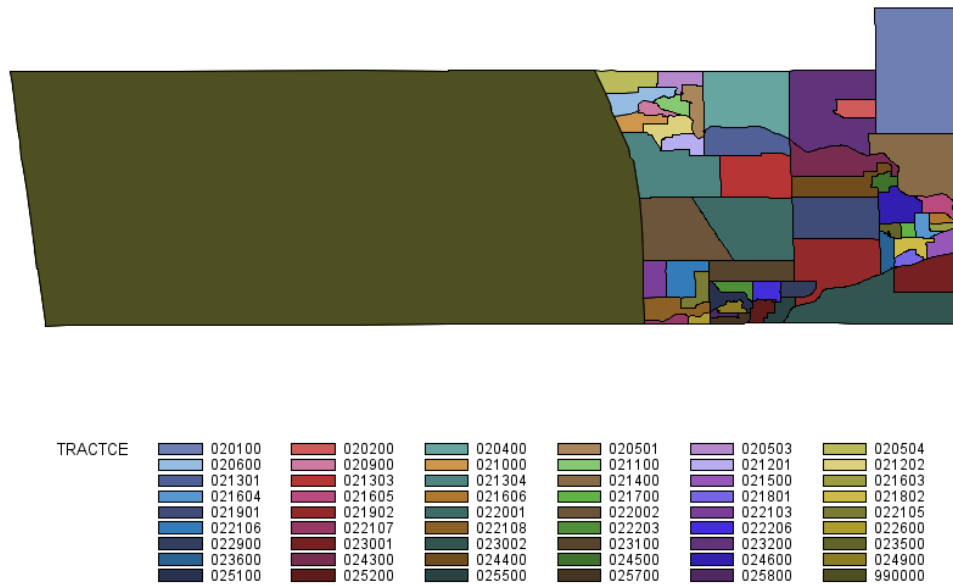
DATA Ottawa;
  SET Census;
  IF countyfp in ('139');
RUN;

PROC GMAP DATA = Ottawa map = Ottawa;
  ID tractce;
  CHORO tractce;
  TITLE "Ottawa County";
RUN;

```

Ottawa County

Figure 14



When viewing the map above, we can see that the areas we know to be water are in fact shaded differently from the other census tracts. To determine which census tracts belong to water areas we can look at the legend that was given and we can see that the one census tract that is different from the rest is census tract number 990000. Following the manner in which we restricted the entire Michigan map to only that of Ottawa County we can restrict the above map to only areas of land. The code and output is:

```

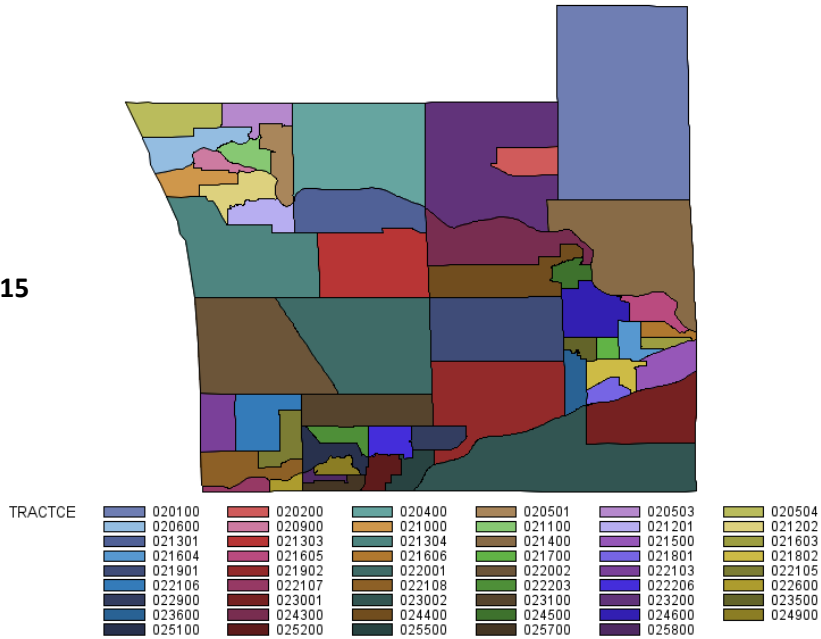
DATA OttawaLand;
  SET Ottawa;
  IF tractce =: "99" THEN DELETE;
RUN;

PROC GMAP DATA = OttawaLand MAP = OttawaLand;
  ID tractce;
  CHORO tractce;
  TITLE "Ottawa County - Only Land";
RUN;

```

Ottawa County - Only Land

Figure 15



Once we have restricted the county to only land, we can apply PROC GREMOVE as we did for Kent County. It is important to first eliminate the water areas before applying PROC GREMOVE otherwise the outline that is created will contain the body of water. The remaining code and final Ottawa County map are given below.

```

PROC GREMOVE DATA = OttawaLand OUT = OttawaBorder;
  BY countyfp;
  ID tractce;
RUN;

PROC GMAP DATA = OttawaBorder MAP = OttawaBorder;
  ID countyfp;
  CHORO countyfp;
  TITLE "Ottawa County Border";
RUN;

```


Ottawa County Border

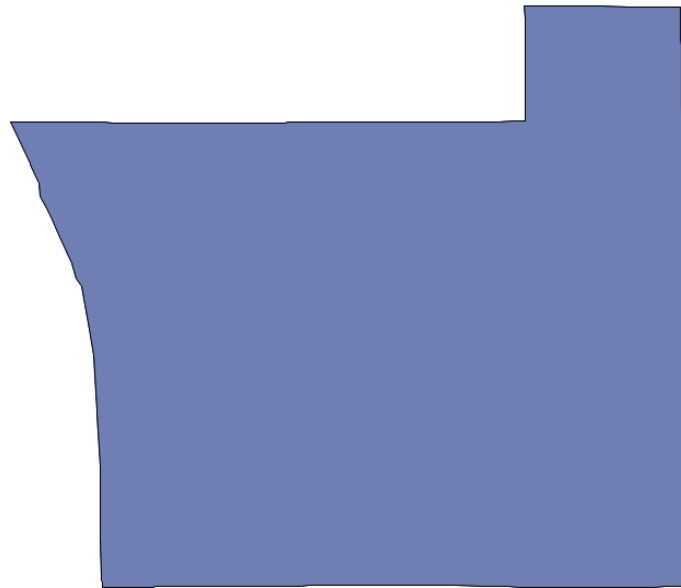


Figure 16

COUNTYFP 139

Now that the appropriate parts of the final map have been created, we can apply the same methodology as before and can create a similar map that shows both zip code and county borders. The map below represents fabricated data pertaining to Ottawa County.

Map of Ottawa County

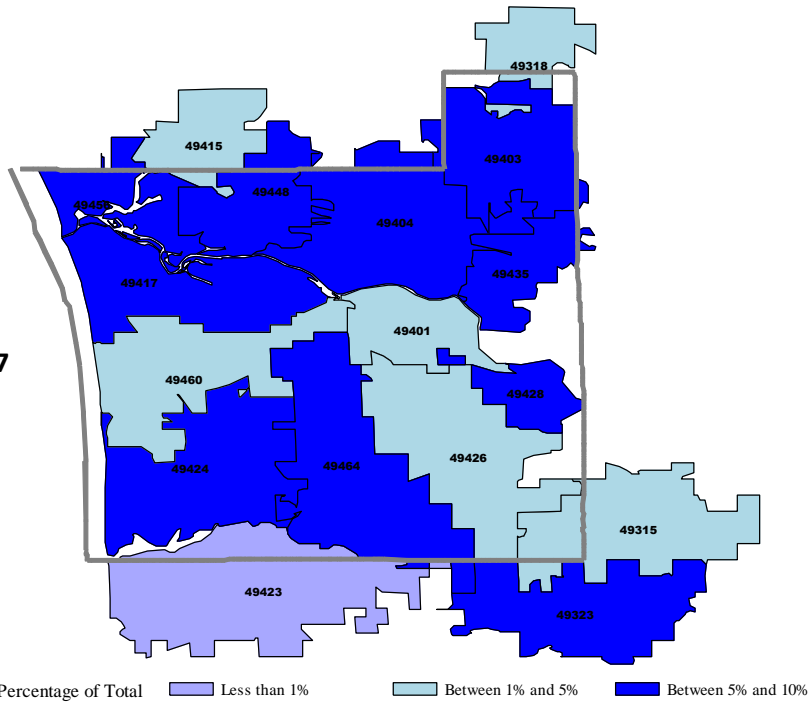


Figure 17

Percentage of Total Less than 1% Between 1% and 5% Between 5% and 10%

Excess Lines in Annotating Borders

Now that we know that water boundaries and curved lines are able to be mapped, we can extend the PROC GMAP capabilities one step further. There is the potential for organizations to have programs that extend beyond a single county. In that case, when we want to annotate a map with multiple county borders the easiest way to create the correct annotate dataset is to merge datasets that contain the coordinates for the outlines of multiple counties. Once those different border datasets have been merged and the draw function has been implemented as in previous examples, we can apply the PROC GMAP code as before to get the resulting output and code that is below.

```
DATA CountyLine_WestMI;
    SET OttawaBorder KentBorder MontcalmBorder MuskegonBorder
        NewaygoBorder;
RUN;

DATA CountyLine_WestMI2;
    LENGTH function color $ 8 position $ 1 line 8.;
    RETAIN function 'label' xsys ysys '2' hsys '3'
        when 'a' position '5' line 1;
    SET CountyLine_WestMI;
    IF _N_ = 1 THEN function = 'move';
    IF _N_ NE 1 THEN function = 'draw';
    IF _N_ NE 1 THEN size = 0.6;
    IF _N_ NE 1 THEN color = 'Red'; OUTPUT;
    KEEP function color line size position xsys ysys hsys when
        x y;
RUN;

DATA Anno_WestMI;
    LENGTH function $8;
    SET CountyLine_WestMI2 Anno_Label;
RUN;

PROC GMAP DATA = WestMI_Data MAP = WestMI_Zip;
    ID zcta;
    FORMAT percent percentgroup.;
    CHORO PERCENT / DISCRETE ANNOTATE = Anno_WestMI;
    TITLE 'Map of West Michigan County Borders';
    TITLE2 'Before Applying the Countyline Fix';
RUN;
```

As can be seen from the image below, a problem arises when drawing multiple county borders. This comes from the draw function within SAS.

**Map of West Michigan County Borders
Before Applying the Countyline Fix**

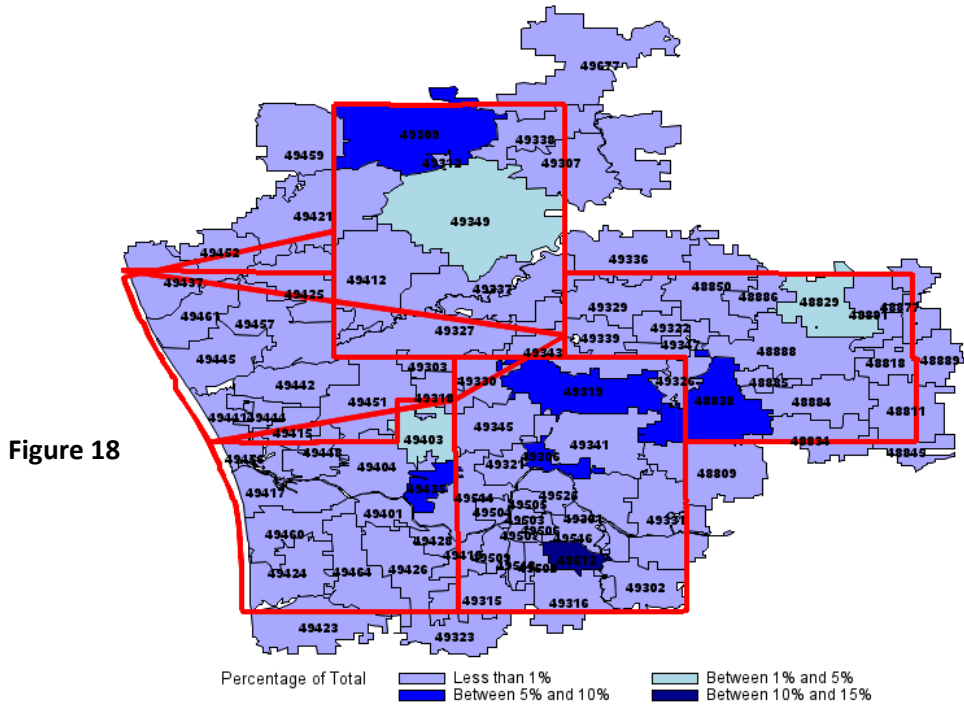


Figure 18

The DRAW function in SAS acts like a pencil touching a piece of paper and trying to draw multiple objects without every lifting off the paper. Because of the draw function’s behavior when drawing each county on this map, extra lines are created when the DRAW function ends for one county and goes on to draw another.

To correct for the superfluous lines, we can apply a fix that evaluates the difference in coordinate location (either X or Y) between one point and its predecessor. We must evaluate the merged annotate dataset and look for extreme differences between two subsequent points. When an annotate dataset contains thousands of data points, these differences may not be much however. An extreme difference can be anything greater than 0.2 (what is considered extreme will be dependent on the scale of the shapefile that is being utilized). When a large difference between points is found, the code will instruct SAS to change the function from DRAW to MOVE. These differences are calculated for both X and Y coordinates. As previously stated, we used the cutoff point of an absolute difference greater than the value of 0.2; this was chosen arbitrarily to obtain the output we desired. The code is below.

```

DATA Previous;
  SET CountyLine_WestMI2 nObs = _nObs;
  lead_x = .;
  lead_y = .;
  _point = _N_ - 1;
  IF _point <= _nObs THEN DO;
    SET CountyLine_WestMI2 (KEEP = x y RENAME=(x=lead_x y
      = lead_y)) point = _point;
  END;
  diff = abs(lead_x - x);
  diff2 = abs(lead_y - y);
RUN;

```

```

DATA CountyLine_WestMI3;
  LENGTH function color $ 8 position $1 line 3.;
  RETAIN function 'label' xsys ysys '2' hsys '3'
    when 'a' position '5' line 1;
  SET Previous;
  IF _N_ = 1 or diff > .2 or diff2 > .2 THEN function =
    'move';
  ELSE DO;
    function = 'draw';
    size = 0.6;
    color = 'Red';
  END;
  OUTPUT;
  KEEP function color line size position xsys ysys hsys when
  x y;
RUN;

```

```

DATA Anno_WestMI2;
  SET CountyLine_WestMI3 Anno_Label;
RUN;

```

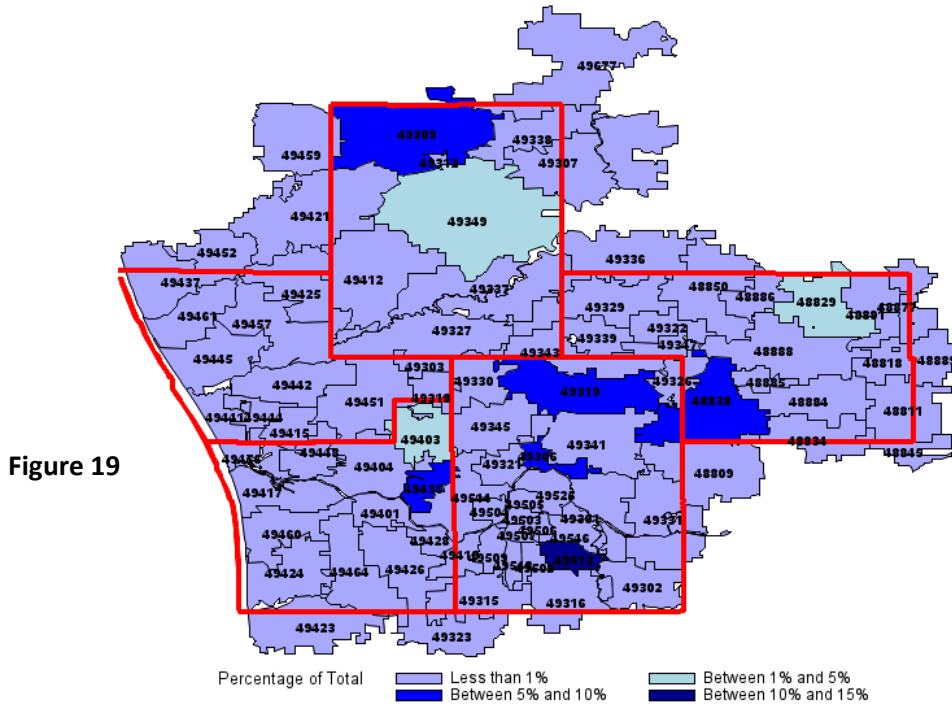
Now that the annotate dataset has had the extra lines removed, we can apply the new Annotate dataset to the map. The code and final map are given below.

```

PROC GMAP DATA = WestMI_Data MAP = WestMI_Zip;
  ID zcta;
  FORMAT percent percentgroup.;
  CHORO PERCENT / DISCRETE ANNOTATE = Anno_WestMI2;
  TITLE 'Map of West Michigan County Borders';
  TITLE2 'After Applying the Countyline Fix';
RUN;

```

**Map of West Michigan County Borders
After Applying the Countyline Fix**



The final map shows multiple county borders with zip codes identified and labeled. With the extra lines removed, the visual appeal of the map has increased and PROC GMAP has gained yet another facet of capabilities to add to its repertoire.

CONCLUSION

PROC GMAP has a great deal of potential in describing data. By using PROC GMAP in conjunction with the %ANNOTATE macro, more and more users will begin to identify some of its shortcomings and figure out ways to overcome and resolve them. Combining both zip code boundaries and county borders has long been a mystery for GMAP users and this paper offers a solution to that enigma. The annotate dataset can be utilized far more extensively than might have heretofore been thought; and therefore enhance maps produced in SAS when applied effectively.

REFERENCES

- "Base SAS - ODS Statistical Graphics (Preproduction) and ODS Styles: Usage and Reference." SAS. SAS, 2013. Web. 1 June 2013. <<http://support.sas.com>>.
- "County FIPS Code Listing for the State of Michigan." *U.S. Environmental Protection Agency*. N.p., 10 Apr. 2009. Web. 1 June 2013. <<http://www.epa.gov/enviro/html/codes/mi.html>>.
- "Creating an Annotate Data Set." SAS. SAS, 2013. Web. 1 May 2013. <<http://support.sas.com>>.
- Michigan Department of Technology, Management & Budget*. N.p., 2002. Web. 1 June 2013. <<http://www.mcgi.state.mi.us/mgdl/?rel=ext&action=sext>>.
- MONGABAY.COM. Ed. Rhett Butler. N.p., 2012. Web. 1 June 2013. <http://www.mongabay.com/igapo/zip_codes/counties/alpha/Kent%20County-Michigan1.html>.
- "PROC GMAP Statement." SAS. SAS, 2013. Web. 1 June 2013. <<http://support.sas.com>>.
- "Specifying Colors in SAS/GRAPH Programs." SAS. SAS, 2013. Web. 1 June 2013. <<http://support.sas.com>>.
- "TIGER Products." *United States Census Bureau*. N.p., 2013. Web. 1 June 2013. <<http://www.census.gov/geo/maps-data/data/tiger.html>>.
- "TrueType Fonts that Are Supplied by SAS." SAS. SAS, n.d. Web. 1 June 2013. <<http://support.sas.com>>.
- United States Map*. Web. 1 June 2013. <<http://www.caribemedianetwork.com/wp-content/uploads/2011/03/United-States-Map-3.gif>>.

CONTACT INFORMATION

Your comments and questions are much appreciated. Contact the authors at:

Kathryn Schurr, M.S.

Statistical Database Analyst

Spectrum Health Corporate

Healthier Communities

665 Seward Avenue NW, Suite 110

Grand Rapids, MI 49504

Work Phone: 616.391.2983

Work E-Mail: kathryn.schurr@spectrumhealth.org

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® Indicates USA registration.

Other brand and product names are trademarks of their respective companies.

APPENDIX

PROC CONTENTS

WORK.CENSUS

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
4	ALAND	Num	8
5	AWATER	Num	8
6	COUNTYFP	Char	3
7	FUNCSTAT	Char	1
8	GEOID	Char	11
9	INTPTLAT	Char	11
10	INTPTLON	Char	12
11	MTFCC	Char	5
12	NAME	Char	7
13	NAMELSAD	Char	20
3	SEGMENT	Num	8
14	STATEFP	Char	2
15	TRACTCE	Char	6
1	X	Num	8
2	Y	Num	8

WORK.ZCTA

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
4	AREA	Num	8
5	LSAD	Char	2
6	LSAD_TRANS	Char	50
7	NAME	Char	90
8	PERIMETER	Num	8
3	SEGMENT	Num	8
1	X	Num	8
2	Y	Num	8
9	ZCTA	Char	5
10	ZT26_D00_	Num	8
11	ZT26_D00_I	Num	8