SESUG Paper RIV-138-2017

Building Heat Maps for Data Cleaning and Beyond

Kristen Much, Rho®, Inc.; Chapel Hill, NC; Nicole Bader, Rho®, Inc., Chapel Hill, NC

ABSTRACT

A heat map is a graphical representation of data that translates data values into colors within a matrix. This type of data visualization summarizes a vast amount of data within a single snapshot which helps to quickly communicate relationships between data values. Using SAS®, a heat map can be generated easily with the HEATMAPPARM statement in either the SGPLOT/SGPANEL procedures or Graph Template Language (GTL). The focus of this paper is to explore the syntax of the HEATMAPPARM statement within both SGPLOT/SGPANEL and GTL as well as to walk through a direct application of a heat map to be used in preparation for database lock. The example demonstrates how to develop a heat map that indicates the status of all case report forms (CRFs) within a database across all visits for all subjects. When used effectively, this tool can greatly enhance understanding within a clinical team regarding the status of requirements for a successful database lock.

INTRODUCTION

Heat maps are used across a variety of industries in order to visualize complex data very quickly. In a world where the amount of data is increasing exponentially every year, heat maps have become a powerful tool to summarize complicated data. They can be utilized to enhance our understanding of financial trends, gene expression, sports odds, crime rates, weather patterns, and even election results, just to name a few.

SAS has some great tools for constructing heat maps, particularly the HEATMAPPARM plot statement. Included below is a basic overview of the HEATMAPPARM plot statement in both SGPLOT/SGPANEL and the Graph Template Language (GTL) followed by an in-depth look at how a heat map can be used in preparation for database lock within a clinical trial. We then cover some additional examples that stretch beyond the clinical trial setting.

OVERVIEW OF THE HEATMAPPARM STATEMENT

The HEATMAPPARM plot statement shares the same syntax for both SGPLOT/SGPANEL and GTL. This allows us to switch between each method very easily. That being said, GTL offers a higher level of customization and that is why it is the preferred method used throughout this paper.

There are three required arguments for the HEATMAPPARM plot statement: X, Y, and either COLORGROUP or COLORRESPONSE variables. The X variable represents the columns and the Y variable represents the rows of the matrix. The COLORGROUP or COLORRESPONSE variable will appropriately color in the individual squares designated by the X and Y variables. The difference between COLORGROUP and COLORRESPONSE is that the COLORGROUP argument colors the squares discretely (i.e. there are a finite number of levels of the variable in the dataset and each level maps to a different color) and the COLORRESPONSE argument colors the squares continuously (i.e. using a color gradient). Another notable difference is that the COLORGROUP variable can be either character or numeric, whereas the COLORRESPONSE must be numeric.

The examples below make use of the RETAIL data set available in the SASHELP library. Two variables were added to the dataset: QUARTER, which is derived from the DATE variable to indicate the quarter of the year, and SALES2, which is derived from the SALES variable to indicate sales above/below $500 million.

USING SGPLOT/SGPANEL PROCEDURES

In Figure 1, Y represents the YEAR, X represents the QUARTER, and in this case, COLORGROUP has been selected as SALES2. Since we are using COLORGROUP, the boxes will be shaded a distinct color for each value of SALES2, which is either '>= 500 million' or '< 500 million'. The SGPLOT/SGPANEL code to produce the Figure 1 heat map is shown below:

```
proc sgplot data=retail;
   styleattrs datacolors=(cxdd6060 cx6497eb);
   heatmapparm y=year x=quarter colorgroup=sales2 / showybins outline
                                                     name='sales' ;
   keylegend 'sales';
run;
```

Aside from the HEATMAPPARM plot statement, the STYLEATTRS and KEYLEGEND statements are used. STYLEATTRS controls the coloring of each level of our COLORGROUP variable. If we omit this statement or do not select enough colors to match the number of categories in our COLORGROUP variable, SAS will select colors from its default list. KEYLEGEND displays the legend named 'sales' that is associated with the HEATMAPPARM plot statement. Within the HEATMAPPARM statement, the options SHOWYBINS and OUTLINE are selected so that the y-axis labels are displayed for each bin (i.e. each year in this example) and the boxes created have a black outline, respectively.

Figure 1 summarizes sales from 1980 to 1990 at each quarter using SGPLOT/SGPANEL. With this heat map, it is easy to see that the highest sales years were 1988-1990, regardless of quarter.

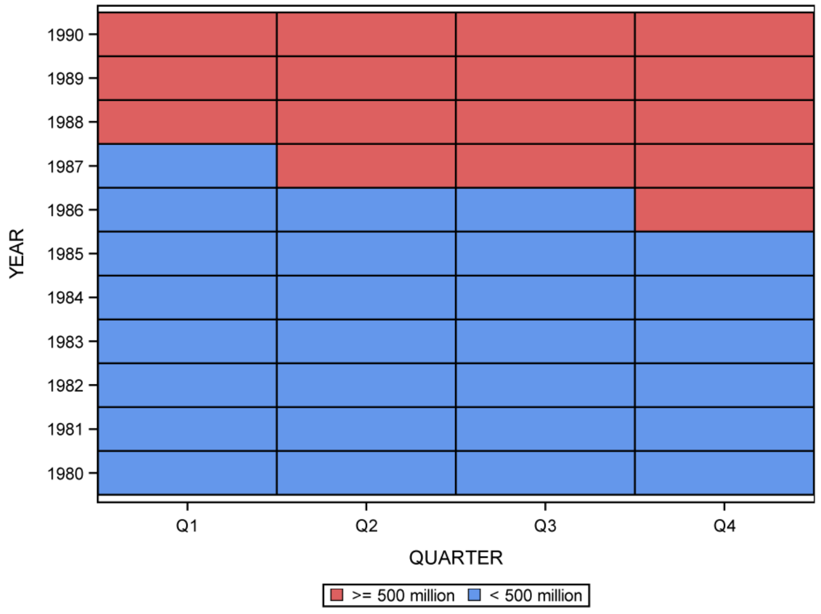

Figure 1. Example of HEATMAPPARM Plot using SGPLOT/SGPANEL Procedures

USING GRAPH TEMPLATE LANGUAGE

Using the same data from Figure 1, Figure 2 is created using GTL. Just like in Figure 1, the Y represents the YEAR, the X represents the QUARTER, and in this case, COLORRESPONSE has been selected as our continuous variable, SALES. Since COLORRESPONSE is selected and not COLORGROUP, the boxes will be shaded in a gradient fashion for each value of SALES. The GTL code to produce the Figure 2 heat map is shown below:

```
proc template;
   define statgraph heatmap;
      begingraph;

      layout overlay;
         heatmapparm y=year x=quarter colorresponse=sales / display=all
                                                             name='sales';
         continuouslegend 'sales';
      endlayout;

      endgraph;
   end;
run;

proc sgrender data=retail template=heatmap;
run;
```

If you are unfamiliar with GTL, the code outside of the bracket is standard code that you must always use. If you want some additional guidance on using GTL, consult the recommended reading “Creating Sophisticated Graphics with GTL”.

Within the bracket, you will notice that the CONTINUOUSLEGEND statement is also used to show the legend named ‘sales’, associated with the HEATMAPPARM plot statement, to show how the color gradient translates to sales. Within the HEATMAPPARM plot statement, the DISPLAY option is selected to show the black outline around the each box.

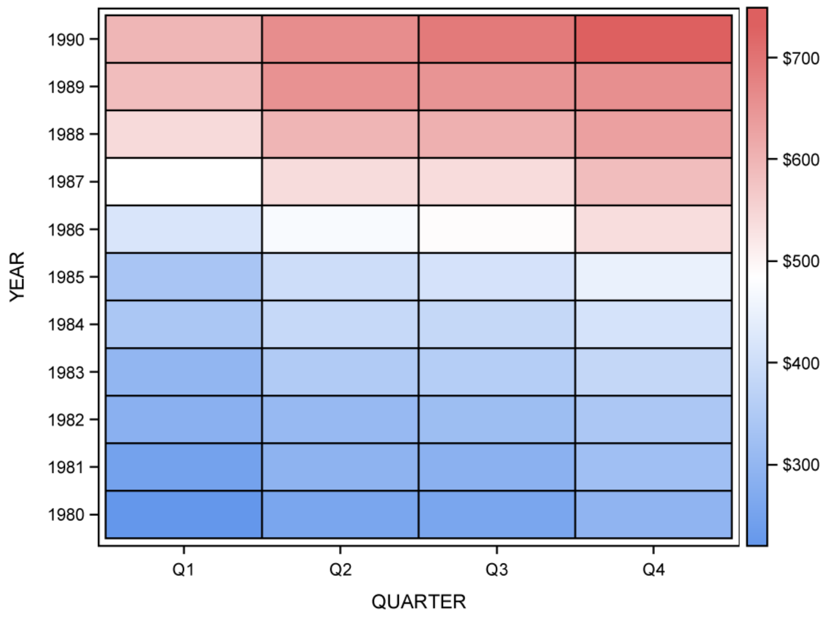

Figure 2. Example of HEATMAPPARM Plot using Graph Template Language

HEAT MAP FOR DATABASE LOCK EXAMPLE

Medidata Rave is a popular electronic data capture (EDC) system used throughout the clinical trial industry. For any database created within Rave, a data set containing case report form (CRF) metadata can be exported. This data set contains detailed information (e.g. form completion status, the existence of an outstanding query, etc.) about each unique CRF collected at each visit. This data set makes it possible to create a heat map that visually represents the overall status of each requirement for database lock.

The resulting heat map, Figure 3, has one row per subject and one column for each CRF collected at each visit. This results in one ‘box’ per subject per visit per CRF. Each box is colored and/or annotated to indicate the current status of each and every CRF in the database.

A quick study of this graphic can provide the clinical study team with a high level overview of how many CRFs have yet to be completed, where queries have not yet been closed, which CRFs have been source data verified, and whether or not an individual CRF has been locked. In addition, all of this detailed information can be identified for a specific subject at a specific visit for a specific CRF.

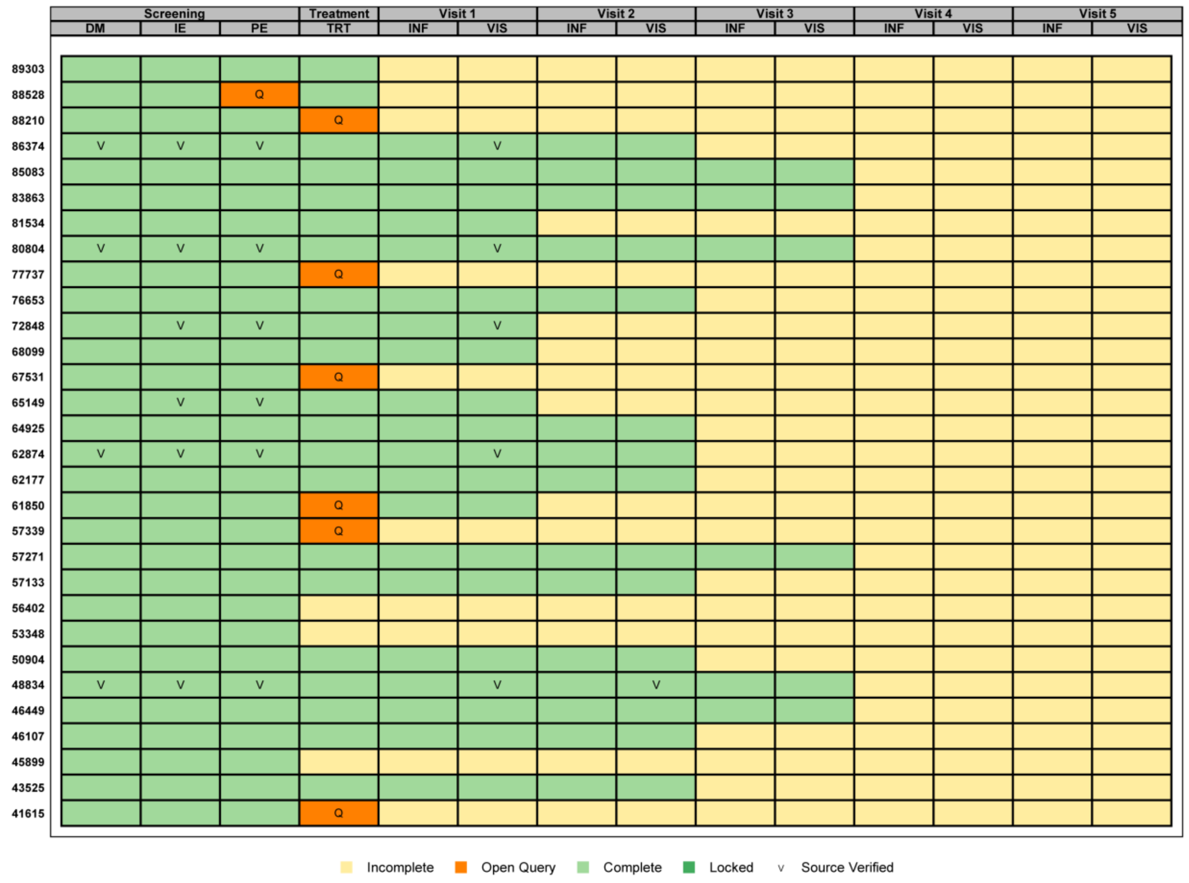

Figure 3. Heat Map for Database Lock

This is the code used to produce Figure 3:

```
proc template;
   define statgraph crfstat;
      begingraph;

      symbolchar name=sym1 char='0056'x;

      legenditem type=marker name='c1' / <LEGENDITEM OPTIONS>;
      legenditem type=marker name='c2' / <LEGENDITEM OPTIONS>;
      legenditem type=marker name='c3' / <LEGENDITEM OPTIONS>;
      legenditem type=marker name='c4' / <LEGENDITEM OPTIONS>;
      legenditem type=marker name='t1' / <LEGENDITEM OPTIONS>;

      discreteattrmap name='colors' / ignorecase=true;
         value '0' / fillattrs=(color=cxffeda0);
         value '1' / fillattrs=(color=cx41ab5d);
         value '2' / fillattrs=(color=cxa1d99b);
         value '3' / fillattrs=(color=cxff8000);
      enddiscreteattrmap;

      discreteattrvar attrvar=flagcolors var=flag attrmap='colors';

      layout overlay / <LAYOUT OVERLAY OPTIONS>;
         heatmapparm x=column y=id colorgroup=flagcolors /<HEATMAPPARM
                                                           OPTIONS>;
         annotate;

         innermargin / align=top;
            blockplot x=column block=visitname / <BLOCKPLOT OPTIONS>;
            blockplot x=column block=crf / <BLOCKPLOT OPTIONS>;
         endinnermargin;

         discretelegend 'c1' 'c2' 'c3' 'c4' 't1' / <DISCRETELEGEND
                                                    OPTIONS>;
      endlayout;

      endgraph;
   end;
run;

proc sgrender data=rave4 template=crfstat sganno=anno;
run;
```

LEGENDITEM OPTIONS

The <LEGENDITEM OPTIONS> control the attributes (size, color, and shape) and label for each item included within the legend. The code for this is displayed below:

```
label="Incomplete" lineattrs=(color=black)
markerattrs=(symbol=squarefilled color=cxffeda0 size=10pt)
labelattrs=(size=10pt);

label="Open Query" lineattrs=(color=black)
markerattrs=(symbol=squarefilled color=cxff8000 size=10pt)
labelattrs=(size=10pt);
```

```
label="Complete" lineattrs=(color=black) markerattrs=(symbol=squarefilled
color=cxa1d99b size=10pt) labelattrs=(size=10pt);
```

```
label="Locked" lineattrs=(color=black) markerattrs=(symbol=squarefilled
color=cx41ab5d size=10pt) labelattrs=(size=10pt);
```

```
label="Source Verified" lineattrs=(color=black) markerattrs=(symbol=sym1
color=black size=10pt) labelattrs=(size=10pt);
```

LAYOUT OVERLAY OPTIONS

The <LAYOUT OVERLAY OPTIONS> control what is displayed on the X and Y axis and their associated visual attributes. This code is shown below:

```
yaxisopts=(tickvalueattrs=(size=8pt weight=bold) display=(line
tickvalues))
xaxisopts=(display=(line) tickvalueattrs=(size=6pt));
```

HEATMAPPARM OPTIONS

The <HEATMAPPARM OPTIONS> control the outline of each box. By selecting DISPLAY equal to ALL, each box is outlined. OUTLINEATTRS controls the coloring and thickness of the outline. This code is shown below:

```
outlineattrs=(color=black thickness=2) display=all
```

BLOCKPLOT OPTIONS

The <BLOCKPLOT OPTIONS> control the headers above the heat map. There is a BLOCKPLOT statement for each row of the header. The top contains the visit name and the bottom contains the form name. The REPEATVALUES=FALSE ensures that the visit name spans across visits with multiple forms instead of it repeated above each form. This maintains header readability. The <BLOCKPLOT OPTIONS> also control the appearance of the header, in this case shading it grey and center-aligning the bolded text. This code is displayed below:

```
repeatedvalues=false display=(values outline fill)
altfillattrs=(color=cxbdbdbd) fillattrs=(color=cxbdbdbd)
filltype=alternate valuehalign=center valueattrs=(weight=bold)
outlineattrs=(thickness=2);

repeatedvalues=false display=(values outline fill)
altfillattrs=(color=cxbdbdbd) fillattrs=(color=cxbdbdbd)
filltype=alternate  valuehalign=center valueattrs=(weight=bold)
outlineattrs=(thickness=2);
```

DISCRETELEGEND OPTIONS

The <DISCRETELEGEND OPTIONS> control the location and size of the legend. The DISPLAYCLIPPED option specifies that the legend will be displayed even if any portion of the legend cannot entirely display. This option is particularly useful when working with legends that contain numerous entries or long labels. By default, if there is not enough space to write all legend entries without overlapping the text, the entire legend will be omitted. The DISPLAYCLIPPED option forces the legend to be printed, which allows you to troubleshoot the issue. The ACROSS option specifies the number of legend entries to be placed horizontally before moving to the next row. This code is displayed below:

```
across=5 displayclipped=true border=false
```

KEY FEATURES OF HEAT MAP FOR DATABASE LOCK EXAMPLE

USE OF SPECIAL SYMBOLS

The SYMBOLCHAR statement allows symbols to be defined using any Unicode character, which means symbols are no longer limited to the 30 marker options available within SAS. In the code above, the letter V ('0056'x) has been selected. To define a symbol within the SYMBOLCHAR statement, simply specify a NAME for the symbol and set the Unicode specification (4 digit alphanumeric code within single quotes followed by an ‘x’) to CHAR. That NAME can then later be used as the SYMBOL within the MARKERATTRS.

CONTROLLING THE LEGEND

The LEGENDITEM statement defines each element to be included in the legend independent of what is actually present in the data. If LEGENDITEM had not been utilized in Figure 3, the ‘Locked’ status would not have appeared on the legend because no forms have yet been locked. TYPE and NAME are the two required arguments where TYPE specifies a type (fill, marker, markerline, line, or text) for the legend item and NAME assigns a name to the legend item to be called within the DISCRETELEGEND statement.

ATTRIBUTE MAPPING

The DISCRETEATTRMAP assigns graphical attributes to specified data values. In this case, a fill color has been assigned to each of the possible values for the variable FLAG. To achieve this, the DISCRETEATTRMAP must be assigned a NAME (in this case, ‘colors’) and the values must be mapped to their graphical attributes. That NAME is then set to ATTRMAP in a DISCRETEATTRVAR statement. VAR is assigned to the variable in the data set that contains the mapped values (in this case, FLAG), and a new ATTRVAR name is assigned (in this case, FLAGCOLORS). This ATTRVAR is then used within the plot statement.

MORE EXAMPLES

The main example of this paper focuses on data collected in a clinical trial, but the use of heat maps can extend beyond clinical data. The next two examples will show heat maps beyond the clinical trial setting.

TEMPERATURE DATA EXAMPLE

This example uses public data of average temperature from three North Carolina airports provided by the National Weather Service. In this Figure 4, Y represents the name of the AIRPORT, X represents the DAY of the month, and COLORRESPONSE is selected as the average daily temperature variable, AVG.

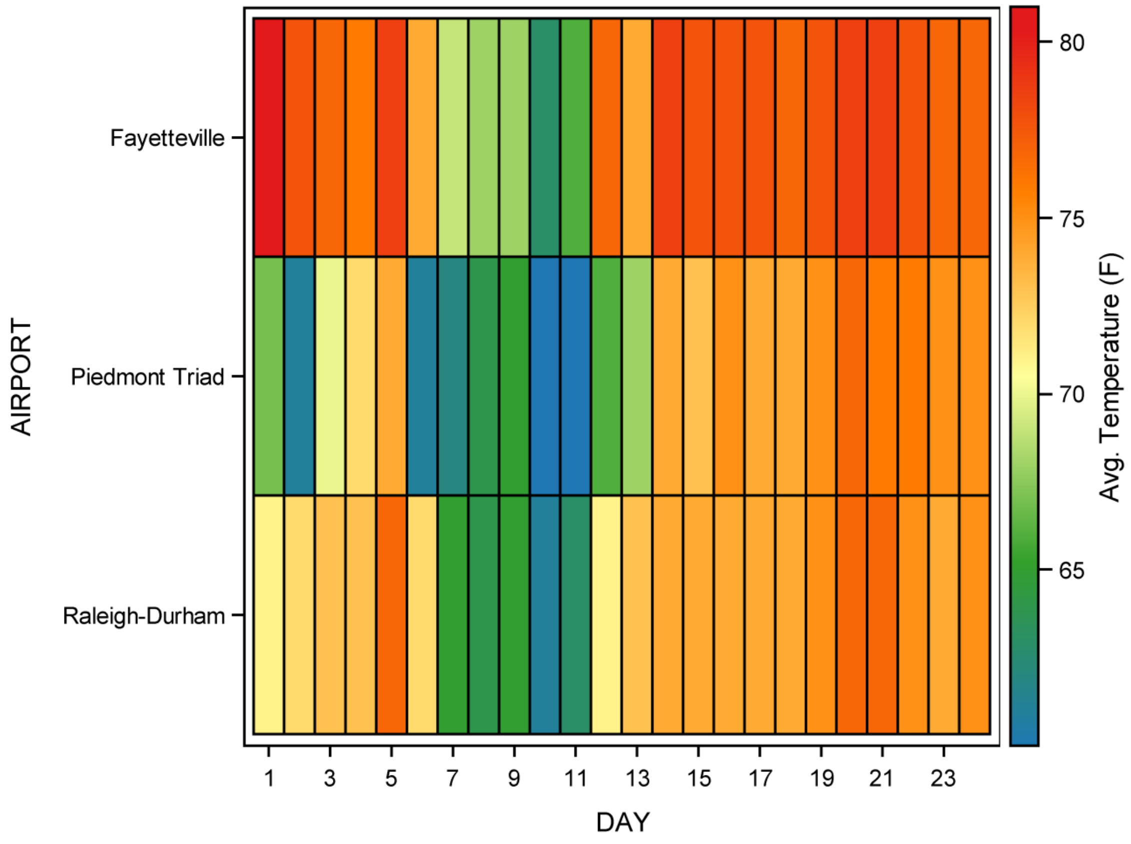

Figure 4. Heat Map of Average Temperature at NC Airports across 24 Days

The GTL code to produce the Figure 4 heat map is shown below:

```
proc template;
   define statgraph heatmap;
   begingraph;

      layout overlay;
         heatmapparm y=airport x=day colorresponse=avg / <HEATMAPPARM
                                                        OPTIONS>;
         continuouslegend 'temps' / title = 'Avg. Temperature (F)';
      endlayout;

   endgraph;
   end;
run;

proc sgrender data=temp template=heatmap;
run;
```

HEATMAPPARM OPTIONS

The <HEATMAPPARAM OPTIONS> options are the same as the options from Figure 2, with the addition a COLORMODEL statement. COLORMODEL specifies the colors to be used within the heat map, and is set equal to either a style element or a color list. In Figure 4, a color list of five different colors has been selected. At least two colors should be listed for COLORMODEL to produce a color gradient. The first color listed within the color list corresponds to the smallest data value of the COLORRESPONSE variable, whereas the last color listed corresponds to the largest data value. The code is shown below:

```
display=all name='temps' colormodel=(cx1f78b4 cx33a02c cxffff99 cxff7f00
cxe31a1c)
```

POPULATION DATA EXAMPLE

One specific extension of a heat map is called a choropleth map. Instead of a matrix representation with boxes, geographical regions are used. Figure 5 below illustrates how a map of the United States can be colored to visually represent a relationship between each state and a measure of interest, in this case the 2010 population.

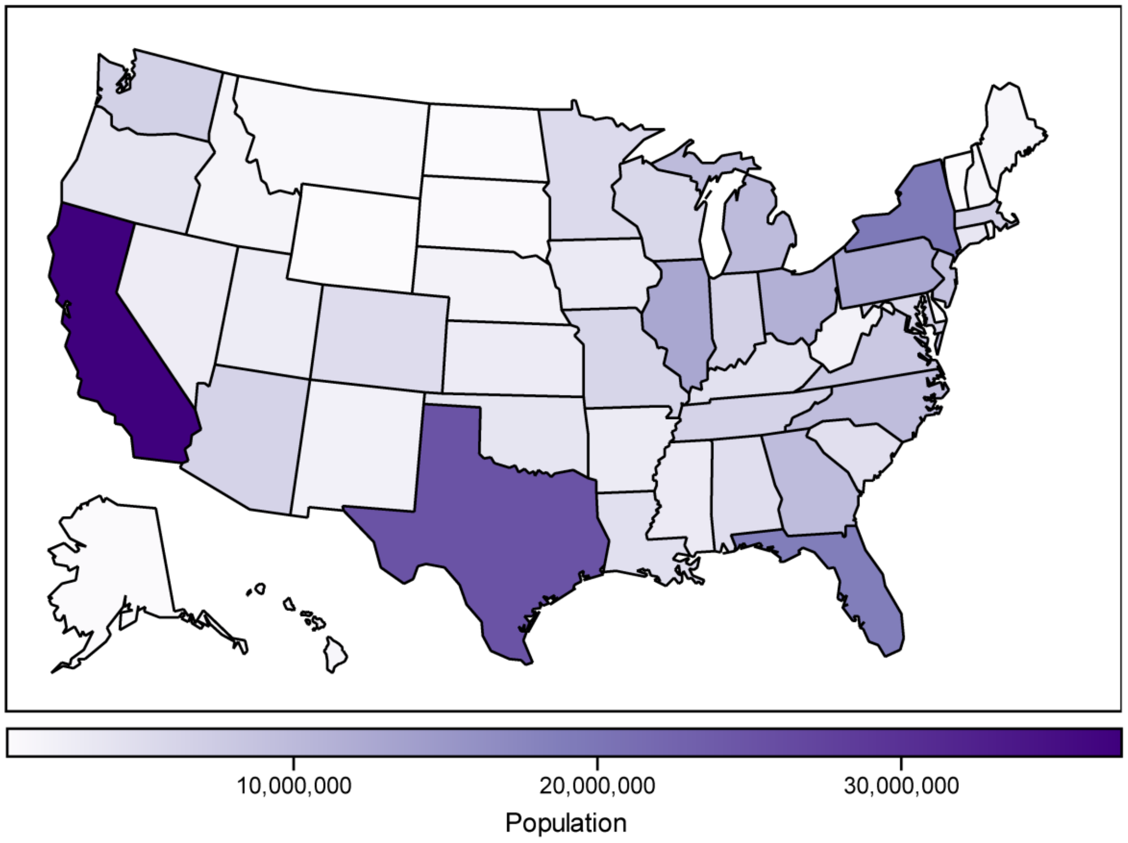

Figure 5. Choropleth map displaying the 2010 population counts for each state.

This example uses two data sets that are available within SAS—the US data set from the MAPS library and the US_DATA data set from the SASHELP library. The US data set will provide the relevant information needed to draw the boundaries of each state, whereas the US_DATA data set will give us the population count for each state.

The SAS code to produce Figure 5 can be found below:

```
proc template;
   define statgraph choropleth;
      begingraph;

      layout overlayequated / <LAYOUT OVERLAYEQUATED OPTIONS>;
         polygonplot x=x y=y id=polyid / <POLYGONPLOT OPTIONS>;
         continuouslegend 'pop' / <CONTINUOUSLEGEND OPTIONS>;
      endlayout;

      endgraph;
   end;
run;

proc sgrender data=us_all template=choropleth;
run;
```

Instead of using the HEATMAPPARM plot statement, this figure uses the POLYGONPLOT statement. This plot statement has three required arguments—X, Y, and ID. The X variable is the longitude and the Y variable is the latitude. These variables together define the boundaries of each state. The ID variable is an identifier that is associated with each distinct polygon. ID is not equal to STATE because states like Hawaii and Michigan are made up of more than one polygon. If STATE is selected instead, random connecting lines appear.

LAYOUT OVERLAYEQUATED OPTIONS

The <LAYOUT OVERLAYEQUATED OPTIONS> control the attributes of the X and Y axis. In this case, the only item displayed on each axis is the axis line. Otherwise, the x and y variable values and labels would display, and in this case would make the overall message less clear. The <LAYOUT OVERLAYEQUATED OPTIONS> used in Figure 5 are shown below:

```
xaxisopts=(display=(line)) yaxisopts=(display=(line))
```

POLYGONPLOT OPTIONS

The <POLYGONPLOT OPTIONS> control the appearance of the polygons. COLORRESPONSE works just as it did in the HEATMAPPARM statement, and in this case, colors each polygon in a gradient fashion based on the value of POPULATION_2010, the 2010 population for the state. COLORMODEL selects the color gradient, in this case shades of red. DISPLAY allows for each polygon to have a FILL color and an OUTLINE. OUTLINEATTRS specifies the coloring of the outline of each polygon, in this case black. The <POLYGONPLOT OPTIONS> to produce Figure 5 are shown below:

```
colorresponse=population_2010 colormodel=(cxfcfbfd cx807dba cx3f007d)
name='pop' display=(fill outline) outlineattrs=(color=black)
```

CONTINUOUSLEGEND OPTIONS

The <CONTINUOUSLEGEND OPTIONS> control the various characteristics of the legend. The placement of the legend is controlled by the LOCATION and VALIGN options. In this case, the legend has been placed outside the figure on the bottom. Additionally, using the TITLE option, a label for the legend has been assigned. The <CONTINUOUSLEGEND OPTIONS> used in Figure 5 are shown below:

```
location=outside valign=bottom title='Population'
```

CONCLUSION

In a world where the amount of data is exponentially increasing every year, finding ways to summarize a vast amount of data simultaneously is now very critical. Heat maps make this possible and SAS provides a convenient platform to create them. Whether this is your first crack at producing a heat map within SAS or you're looking to create a more customized version of a heat map, the examples provided in this paper should steer you in the right direction.

REFERENCES

McConville, Kaitlyn, and Kristen Much. 2015. "Creating Sophisticated Graphics with GTL." Proceedings of the PharmaSUG 2015 Conference. Available at http://www.pharmasug.org/proceedings/2015/DV/PharmaSUG-2015-DV02.pdf.

RECOMMENDED READING

- Graphically Speaking Blog (http://blogs.sas.com/content/graphicallyspeaking/)

CONTACT INFORMATION

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

Kristen Much
Rho®, Inc.
919-408-8000
kristen_much@rhoworld.com

Nicole Bader
Rho®, Inc.
919-408-8000
Nicole_Bader@rhoworld.com