PharmaSUG 2026 - Paper DV-155

A Map to Success with Data Visualization Using ODS Statistical Graphics

Richann Jean Watson, DataRich Consulting
Louise S. Hadden, Independent Consultant

ABSTRACT

Creating custom graphics does not have to be a daunting experience. Anyone who has produced a graph using Output Delivery System (ODS) Graphics has unknowingly used the Graph Template Language (GTL). We take you on a guided tour of how to create a truly custom graph. Our first stop starts with an illustration of a basic plot with little complexity produced with Statistical Graphics (SG) procedures. We then make a pit stop with the TMPLOUT option to help convert the simple plot to GTL. On our road to create a custom graph we need to get out our map to build a map. Our last stop of this adventure takes us to the combining of these two graphs to illustrate the power of GTL to truly customize your graphs.

INTRODUCTION

The focus of this paper and presentation is the procedures, tools and techniques that create data visualizations in SAS®, emphasizing opportunities for customization using ODS Graphics, SG Procedures, and GTL as a base. While a number of graphs produced from ODS Graphics are often "camera-ready", at times there may be a need for that user intervention to get the truly customized graph that is desired. Using the GTL framework of ODS style templates and destinations both enables customization and ensures consistency in production.

GATHERING ALL NECESSARY TOOLS

Our goal for this presentation is to build a pretty fabulous, complex data visualization as shown in Figure 1 with the tools previously described, and demonstrate just how easy and reusable the techniques available within the ODS Graphics system can be.

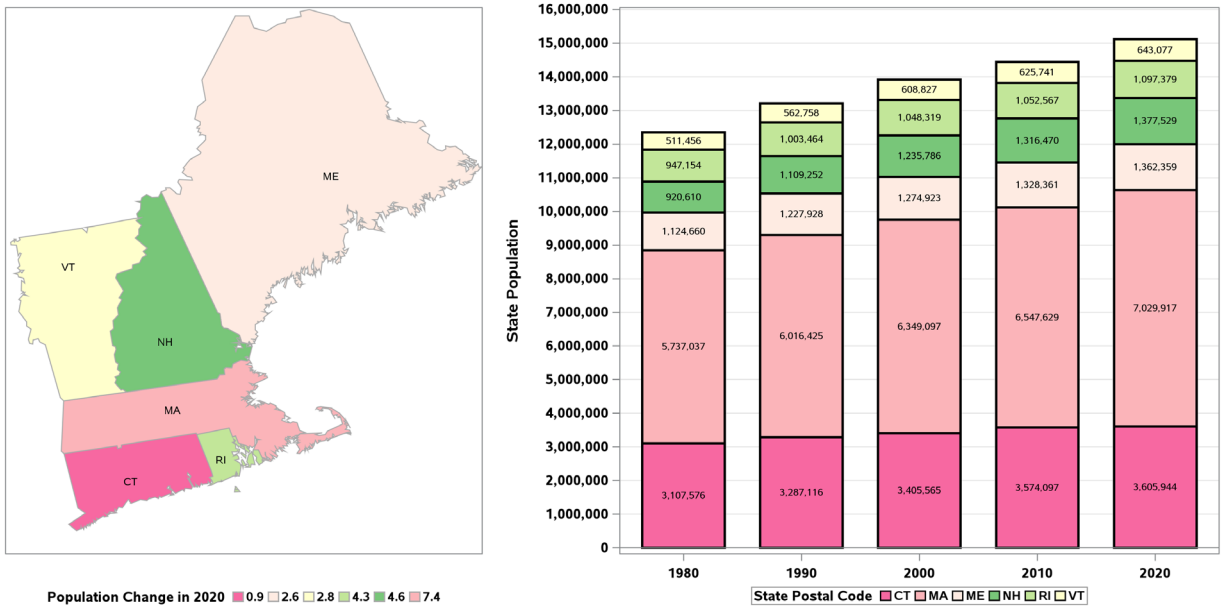

Figure 1. Final Side-by-Side GTL Plot

Before we can get started, we need to make sure we have all the necessary 'tools' gathered. That includes having access to the data, a set of specifications as well as the appropriate software. In this case, the data is the SASHELP.US_DATA data set, and the software is ODS Graphics.

Sample Data 1 shows a subset of SASHELP.US_DATA for the New England States. For each state the following information is captured every 10 years:

- people per square mile (DENSITY_YYYY)
- density rank (RANK_YYYY)
- number of representatives (REPS_YYYY)
- people per representative (PEOPLE_PER_REP_YYYY)
- seat change (SEAT_CHANGE_YYYY)
- population (POPULATION_YYYY)
- change in population expressed as a percentage (CHANGE_YYYY).

Although there are a lot of fields in the data set, for the examples in this paper we are going to focus on POPULATION from 1910 through 2020 and CHANGE in 2020.

Row	STATENAME	DENSITY_1910	...	DENSITY_2020	RANK_1910	RANK_1920	...	RANK_2020
1	Connecticut	230.2		744.7	6	6		6
2	Maine	24.1		44.2	33	34		40
3	Massachusetts	431.6		901.2	3	3		5
4	New Hampshire	48.1		153.8	20	22		23
5	Rhode Island	524.9		1,061.40	2	2		3
6	Vermont	38.6		69.8	27	30		33

Row	REPS_1910	...	REPS_2020	PEOPLE_PER_REP_1910	...	PEOPLE_PER_REP_2020
1	5		5	222,951		721,189
2	4		2	185,593		681,180
3	16		9	210,401		781,102
4	2		2	215,286		688,765
5	3		2	180,870		548,690
6	2		1	177,978		643,077

Row	SEAT_CHANGE_1910	...	SEAT_CHANGE_2020	POPULATION_1910	...	POPULATION_2020
1	0		0	*1,114,756*		*3,605,944*
2	0		0	*742,371*		*1,362,359*
3	2		0	*3,366,416*		*7,029,917*
4	0		0	*430,572*		*1,377,529*
5	1		0	*542,610*		*1,097,379*
6	0		0	*355,956*		*643,077*

Row	CHANGE_1910	...	CHANGE_2020	STATE	STATECODE	DIVISION	REGION
1	22.7		*0.9*	9	CT	New England	Northeast
2	6.9		*2.6*	23	ME	New England	Northeast
3	20		*7.4*	25	MA	New England	Northeast
4	4.6		*4.6*	33	NH	New England	Northeast
5	26.6		*4.3*	44	RI	New England	Northeast
6	3.6		*2.8*	50	VT	New England	Northeast

Sample Data 1: SASHELP.US_DATA Subset for New England States

We also need to make sure we have the specifications for the desired output. Specifications can include things such as

- Font style and size
- Colors
- Layout of the graph(s)
- Types of graph(s) to be produced

With ODS Graphics, we are going to utilize the SGPLOT Procedure, TEMPLATE Procedure as well as SGRENDER Procedure, which is all part of Base SAS, so no additional software is needed. In order to use these tools, we need to understand the basic syntax of each of these procedures.

SGPLOT PROCEDURE

PROC SGPLOT is sufficient for several graphs and is simple to use. It requires at least one plot statement. Within PROC SGPLOT, we can overlay plots and the order in which the plots are specified is important since they will be laid on top of each other. The plot(s) can be further controlled by using additional statements that are associated with the style, symbols and axes. **STYLEATTRS** allows for control of different aesthetics, such as colors and markers. **SYMBOLCHAR** and **SYMBOLIMAGE** allows for defining a special marker. **XAXIS**, **X2AXIS**, **YAXIS** and **Y2AXIS** control the axes. SAS Program 1 shows generic syntax. As mentioned previously, the only required statement is at least one plot statement. The rest is for controlling appearance of the graph. (Harris & Watson, 2020)

```
proc sgplot </options>;

   plot statement(s) / <options>;

   styleattrs </options>;
   symbolchar NAME=identifier CHAR="hex-string" | keyword </options>;
   symbolimage NAME=identifier IMAGE="image-file-specification" </options>;
   xaxis <options>;
   x2axis <options>;
   yaxis <options>;
   y2axis <options>;
run;
```

SAS Program 1: General Syntax for PROC SGPLOT

GTL (TEMPLATE AND SGRENDER PROCEDURES)

GTL is a two-step process. Per Matange (Matange, Getting Started with the Graph Template Language in SAS®, 2013)

1. First, we need to define the structure of the graph using the STATGRAPH template. In the creation of the template, no graph is actually produced.
2. Secondly, we need to associate the data in order to render the template which will produce the graph.

1) Define the Template

The first step is to define the structure of the figure/plot using PROC TEMPLATE. All templates will have the following structure. SAS Program 2 shows the general syntax for PROC TEMPLATE.

```
proc template;
   define statgraph templatename;
      begingraph / <options>;
         layout type / <options>;

            … GTL SAS code …

         endlayout;
      endgraph;
   end;
run;
```

SAS Program 2: General Syntax for PROC TEMPLATE

a) When using PROC TEMPLATE to define a structure for a plot, the first step is to define a **STATGRAPH** template and provide it with a name that can be used for future reference. This is important since PROC TEMPLATE can also be used to create style templates. To define a **STATGRAPH** template, we need to "open" the template with the statement **DEFINE STATGRAPH *TEMPLATENAME*** (blue text in SAS Program 2). The **STATGRAPH** template must also be closed once it has been defined. This is accomplished with the **END** statement.

b) The next step is to signal the start of the various components that are used to create the custom template. **BEGINGRAPH** is used to signal the start and there is at most one **BEGINGRAPH** within the **STATGRAPH**. Like **STATGRAPH**, we need to signal the end of the definition, which is done using **ENDGRAPH** (see green text in SAS Program 2).

c) Each custom template needs to have the layout(s) specified. There are several different layouts to choose from: **OVERLAY, OVERLAYEQUATED, PROTOTYPE, REGION, OVERLAY3D, GRIDDED, LATTICE, DATAPANEL, DATALATTICE**. Each type of layout has a set of options used to control appearance. Certain layouts allow for nesting layouts: **GRIDDED, LATTICE, DATAPANEL, DATALATTICE**. For each **LAYOUT**, we need to close it by using **ENDLAYOUT** (see orange text in SAS Program 2).

d) The last required piece of creating a custom template is to define the structure of the plot. This takes place within the **LAYOUT** block(s). Within the **LAYOUT** block(s) the plot statement(s) are specified. How this portion is programmed is highly dependent on the desired outcome. (Harris & Watson, SAS® Graphics for Clinical Trials by Example, 2020)

2) Produce the Graph

Even though the structure has been defined, PROC TEMPLATE itself does not create the output. To produce an output, data needs to be associated with the structure. PROC SGRENDER is used to make this association between the template (i.e., the defined structure of the plot) and the data which then produces the plot. SAS Program 3 demonstrates the general syntax for PROC SGRENDER. Optional statements that can be used are

- BY: renders by different groups
- FORMAT: formats the data without losing order of data
- LABEL: defines x and y-axis labels if they are not defined in the template
- DATTRVAR: associates a variable in the data with an ID in an attribute map

(Harris & Watson, SAS® Graphics for Clinical Trials by Example, 2020)

```
proc sgrender data = datasetname template = templatename;
   <optional SAS statements>;
run;
```

SAS Program 3: General Syntax for PROC SGRENDER

STACKED BAR CHART

To build the desired graph, we need to start by trying to build one piece at a time. Let's start with building the stacked bar chart of the population size for the New England states. Each bar represents the total population for the year with each segment representing the population for a state. Figure 2 illustrates the desired output.

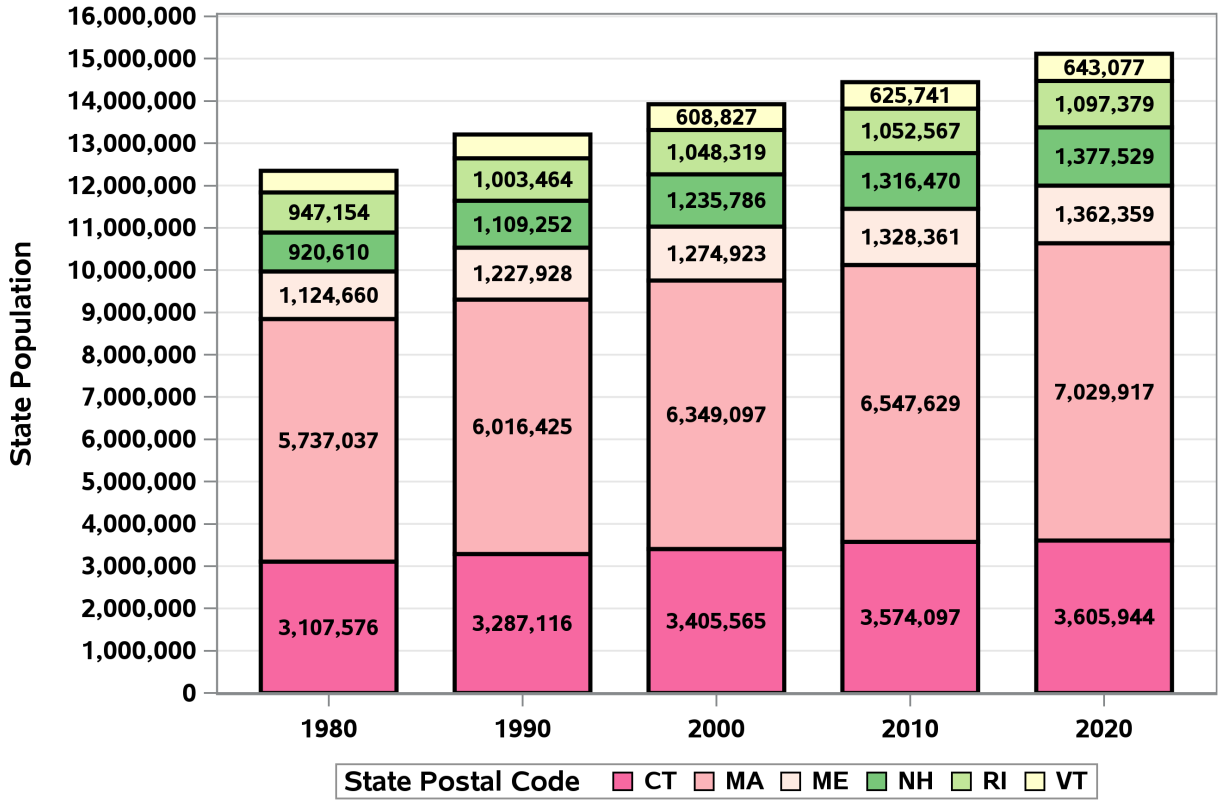

Figure 2: Stacked Bar Chart of the Population of the New England States

As mentioned previously, a subset of SASHELP.US_DATA is used for these examples. However, in order to easily produce the graph, the data needs to be transposed so that there is one record per state per year (refer to Sample Data 2).

STATENAME	STATECODE	DIVISION	REGION	POPULATION	YEAR
Connecticut	CT	New England	Northeast	3,107,576	1980
…					
Connecticut	CT	New England	Northeast	3,605,944	2020
Maine	ME	New England	Northeast	1,124,660	1980
…					
Maine	ME	New England	Northeast	1,362,359	2020
Massachusetts	MA	New England	Northeast	5,737,037	1980
…					
Massachusetts	MA	New England	Northeast	7,029,917	2020
New Hampshire	NH	New England	Northeast	920,610	1980
…					
New Hampshire	NH	New England	Northeast	1,377,529	2020
Rhode Island	RI	New England	Northeast	947,154	1980
…					
Rhode Island	RI	New England	Northeast	1,097,379	2020
Vermont	VT	New England	Northeast	511,456	1980
…					
Vermont	VT	New England	Northeast	643,077	2020

Sample Data 2: Subset of New England States Population Size Transposed

SGPLOT PROCEDURE

This graph can easily be produced with the VBAR plot statement within SGPLOT. The use of additional statements helps to control the appearance of the graph. Refer to SAS Program 4 for the full code to produce the stacked bar chart.

```
proc sgplot data = popln_eng;
   format population comma13. ;
   styleattrs datacolors=(&sixcolors);  ❷
   vbar YEAR / response = POPULATION group = STATECODE
               seglabel seglabelattrs = (size = 8 weight = bold)
               barwidth=.7
               outlineattrs = (color = black thickness = 2pt);   ❶
   xaxis display = (nolabel)
         valueattrs = (weight = bold);
   yaxis label = 'State Population'
         labelattrs = (weight = bold)                            ❸
         values = (0 to 16000000 by 1000000)
         valueattrs = (weight = bold)
         grid offsetmin = 0;
   keylegend / valueattrs = (weight = bold)
               title = "State Postal Code"                       ❹
               titleattrs = (weight = bold);
run;
```

SAS Program 4: Stacked Bar Chart Produced with SGPLOT

❶ **The VBAR** statement creates a vertical bar chart based on category, which is the only required argument. In this case, the category variable is YEAR. Anything after '/' is considered an option. The use of the **RESPONSE** option indicates the variable to use to plot. To produce a stacked bar chart, **GROUP = STATECODE** is needed. GROUPDISPLAY can also be used. However, the default value for **GROUPDISPLAY** is STACK. Recall that one of the features of the stacked bar chart is to have each of the segments labelled with the population count. To achieve this, **SEGLABEL** is used and the attributes of the segment label can be controlled by using **SEGLABELATTRS**. With **SEGLABELATTRS**, we can control the font family, font style (i.e., italic or normal), font size, font weight (i.e., bold or normal) and font color. **BARWIDTH** specifies the width of the bars as a ratio to the maximum possible width (i.e., 1). If we set **BARWIDTH** = 1, there would be no space between the bars. **OUTLINEATTRS** allows us to specify the color, thickness and pattern type for the outline around the bar.

❷ In this example, **STYLEATTRS** is used to set the colors that will be used. In order to set the colors the **DATACOLORS** option is used. Here the colors are stored as a macro variable. Although in this example, only the colors are being specified. **STYLEATTRS** can also be used to set the fill patterns, line patterns, markers if applicable

❸ Various other features of the graph can be controlled using the other statements within PROC SGPLOT. **XAXIS** and **YAXIS** allow us to specify things such as the label and its attributes, values of the tick marks and their attributes. We can control what is displayed or not displayed by using the **DISPLAY** option. For the X axis, the label is not to be displayed, but for the Y axis, all features of the axis will be displayed. **LABEL** allows us to indicate how to label each axis and **LABELATTRS** controls the attributes associated with the labels. The Y axis has specific tick marks at specific intervals and those are indicated with the **VALUES** option and the appearance of those tick mark values are controlled by **VALUEATTRS**.

❹ The **KEYLEGEND** statement allows us to define a legend for the graph and where it should be placed. By default, the legend is **LOCATION = OUTSIDE** and **POSITION = BOTTOM**. Since these are the default values, they do not need to be specified. However, using **LOCATION** and **POSITION** allows us to change the position of the legend. In addition, we can set other features such as a title and attributes associated with the text using **TITLE** and **TITLEATTRS** respectively.

CREATING TEMPLATES

Although using PROC SGPLOT can produce the desired stacked bar chart, the goal is to have it nestled next to a map of the New England states. To achieve this, GTL would be a better option. To use GTL, we need to create a template using PROC TEMPLATE. While creating a template can be daunting, there is a quick way to get started. Using the same code we used to produce the stacked bar chart; we can convert that to template with a simple option.

```
proc sgplot data = input-data-set
             TMPLOUT = "filename";

    … PLOT STATEMENTS …

run;
```

SAS Program 5: Produce a Template from PROC SGPLOT

In SAS Program 5, the option **TMPLOUT** is added to the PROC SGPLOT statement. The **TMPLOUT** option takes the statements defined with PROC SGPLOT and converts it to the equivalent in PROC TEMPLATE and saves it to the file specified. Note that the filename includes the full path and filename with extension. It is important to point out that **TMPLOUT** only works with SGPLOT and SGSCATTER.

While the code that is produced from **TMPLOUT** is executable, it is an eyesore to look at as shown in SAS Program 6.

```
proc template;
define statgraph sgplot;
dynamic _NEGATIVE_;
dynamic _ticklist_;
begingraph / collation=binary dataColors=( CXF768A1 CXFBB4B9 CXFEEBE2 CX78C679 CXC2E699
CXFFFFCC );
EntryTitle "Stacked Bar Chart - Produced with SGPLOT" /;
layout overlay / xaxisopts=( display=( ticks tickvalues line ) TickValueAttrs=(
Weight=bold) type=discrete discreteopts=( TickValueFitPolicy=SplitRotate
tickValueList=_ticklist_ ) ) y2axisopts=(labelFitPolicy=Split) yaxisopts=( Label="State
Population" labelFitPolicy=Split offsetmin=0 LabelAttrs=( Weight=bold) TickValueAttrs=(
Weight=bold) type=auto linearopts=( tickvaluelist=( 0 1000000 2000000 3000000 4000000
5000000 6000000 7000000 8000000 9000000 10000000 11000000 12000000 13000000 14000000
15000000 16000000 ) viewmin=0 viewmax=16000000 ) griddisplay=on )
x2axisopts=(type=Discrete discreteOpts=(tickValueList=_ticklist_
tickvaluefitpolicy=SplitRotate tickValueListPolicy=Union))
y2axisopts=(labelFitPolicy=Split);
   BarChartParm X='YEAR'n Y='_Sum1_POPULATION_'n / primary=true Group='STATECODE'n
OutLineAttrs=( Color=CX000000 Thickness=2pt) SegmentLabelType=auto SegmentLabelAttrs=(
Size=8 Weight=bold ) barwidth=0.7 LegendLabel="POPULATION (Sum)" NAME="VBAR";
   DiscreteLegend "VBAR" / Location=Outside Title="State Postal Code" TitleAttrs=(
Weight=bold) ValueAttrs=( Weight=bold);
endlayout;
endgraph;
end;
run;
```

SAS Program 6: Template Produced from SGPLOT for Stacked Bar Chart

GTL (TEMPLATE AND SGRENDER PROCEDURES)

The code in SAS Program 6 is a good starting point but there are a lot of unnecessary code. There is the use of an option **COLLATION = BINARY** that is undocumented. The code produced by **TMPLOUT** includes **X2AXISOPTS** and **Y2AXISOPTS** which are not needed since the X2 and Y2 axes are not used. In addition, **DYNAMIC** is not needed unless we need to make the template flexible, which allows the variable to be resolved when executed. For this example, it is a one-time use thus DYNAMIC is not needed. For more information on **DYNAMIC** and other Macro Variables in GTL refer to Flexible Templates (SAS Institute Inc., 2026). The use of **ENTRYTITLE** forces the title within the graph area. Because the preference is to have the title and footnotes outside of the graph area, **ENTRYTILE** is not needed. Another thing that SAS did when converting from SGPLOT to GTL is it used **VALIDVARNAME = ANY**. Notice that for **BARCHARTPARM** the **X** and **Y** arguments and the **GROUP** option are in single quotation marks with a 'n' following (see red text in SAS Program 6). Lastly, SAS used **BARCHARTPARM** indicating that the data has already been pre-summarized. If the data is not pre-summarized, we would need to use **BARCHART**. In this case since there is only one record per STATECODE and YEAR, both **BARCHART** and **BARCHARTPARM** work. However, in this example to switch from **BARCHARTPARM** to **BARCHART** instead of using **SEGMENTLABELTYPE = AUTO**, we need to use **SEGMENTLABEL = TRUE**. SAS Program 7 shows the cleaned-up version of the code generated using **TMPLOUT** on PROC SGPLOT.

```
proc template;
   define statgraph ne_bar_ttl;
      begingraph / datacolors = (&sixcolors);
         layout overlay /
            xaxisopts = (display = (ticks tickvalues line)
                         labelattrs = (weight = bold)
                         tickvalueattrs = (weight = bold))
            yaxisopts = (label = "State Population"
                         labelattrs = (weight = bold) offsetmin = 0
                         tickvalueattrs = (weight = bold)
                         linearopts = (tickvaluesequence = (start = 0
                                       end = 16000000 increment = 1000000)
                                       viewmin = 0 viewmax = 16000000)
                         griddisplay = on);
            barchart x = year y = population / group = STATECODE
                         barwidth = 0.7
                         outlineattrs = (color = CX000000 thickness = 2pt)
                         segmentlabel = true segmentlabelformat = comma15.
                         segmentlabelattrs = (size = 6)
                         name = "VBAR"; ❶
            discretelegend "VBAR" / location = outside
                         title = "State Postal Code"
                         titleattrs = (weight = bold)
                         valueattrs = (weight = bold); ❷
         endlayout;
      endgraph;
   end;
run;
```

SAS Program 7: Stacked Bar Chart Produced with GTL

❶ While most of the options on **BARCHART** plot statement are similar to the statements in SGPLOT, there is an option that needs to be explained. **NAME** assigns a name to the plot that allows the aesthetics of the plot to be referenced outside of the plot statement and within another template statement (e.g., **DISCRETELEGEND**).

❷ The **DISCRETELEGEND** statement creates a legend for the plot(s) within the template. **DISCRETELEGEND** requires either the name of the graph, a legend item or a discrete attribute name. In this example, the name of the graph "VBAR" is provided. Note that the name is case sensitive. If we specify "vbar" instead, a legend would not be produced. In addition, other options can be used to control the appearance of the legend. **TITLE** allows us to provide a specific title for the legend. Without TITLE option, no title will be displayed. **TITLEATTRS** controls the appearance of the title. In this case, the title is bold. In order to change the attributes for the values of legend items, **VALUEATTRS** is used. If the attributes are not specified, then GraphLabelText and GraphValueText associated with the style template currently in effect is used for legend titles and values respectively.

Now we have our template, if we decide that we want to save it to a permanent location for future use, we need to update ODS PATH. By default, all templates are saved to SASUSER.TEMPLAT (Harris & Watson, SAS® Graphics for Clinical Trials by Example, 2020).

```
libname ADAM "location-to-save-template";
ods path ADAM.TEMPLAT (update) SASUSER.TEMPLAT (read)
         SASHELP.TMPLMST (read);
```

SAS Program 8: Using ODS Path

MAP OF NEW ENGLAND STATES

The next piece of the desired out is the map of the change in population for the New England states. Figure 3 illustrates the desired output.

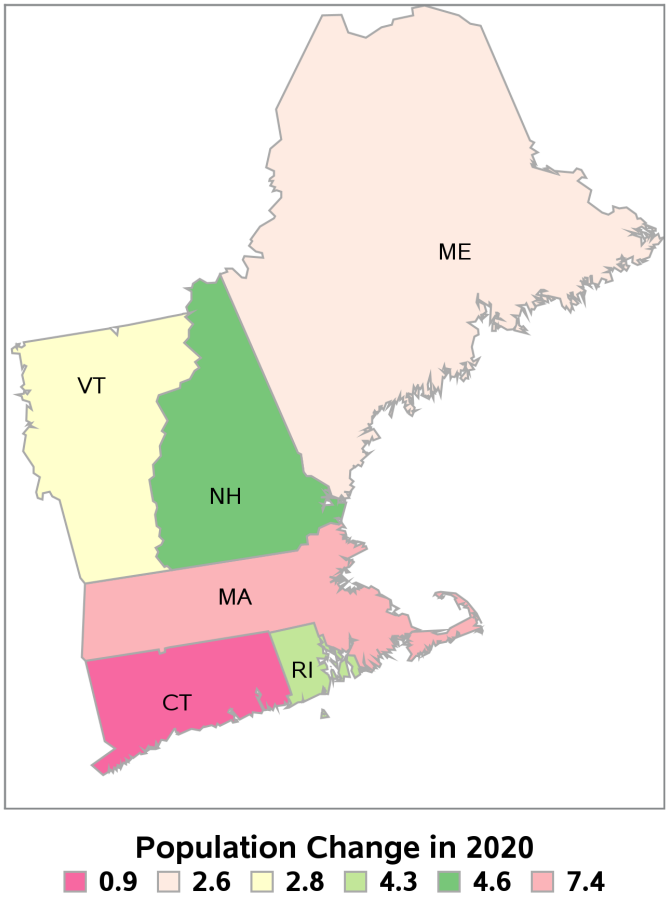

Figure 3: Map of the Population Change of the New England States

Continuing with the subset of SASHELP.US_DATA we extract the change in population for 2020 (CHANGE_2020). In order to produce a geographic visualization, some additional inputs are required to draw the outline of each state (LAT and LONG coordinates) and to place the state abbreviations in the center of each state's polygon. These inputs are linked via the MAPID variable. In our use case, we need to obtain geographic data (LAT and LONG representing latitude and longitude) from a SAS map database. Sample Data 3 incorporates information from SASHELP.US_DATA, calculated centroids, which are the latitude and longitude coordinates (XCEN and YCEN) for the state labels which are calculated for map areas using a SAS-provided macro (%centroid, part of a suite of geographic macros provided by SAS enabled by %SGANNO), and geographic data from the SAS-provided map MAPSGFK.US.

The display in Sample Data 3 below shows six rows at the top which provide information that will be annotated onto the polygon for each state (the state abbreviation, LABEL) in addition to centroid information (XCEN, YCEN) that tells SAS where to place the label information. Although the "stacking" of the data set looks a little unusual, the template for the POLYGON statement in PROC SGPLOT knows exactly what to do with the LAT, LONG, XCEN, YCEN, response variable (CHANGE_2020), and LABEL variable, in conjunction with color lists. The important thing to remember is that a map, in this case a

polygon plot, is still just a plot to SAS, and follows all the rules of the graph template language. For an excellent explanation of SAS-produced POLYGON plots please see "Combining Functions and the POLYGON Plot to Create Unavailable Graphs Including Sankey and Sunburst" (Meyers, 2024).

Row	MAPID	LAT	LONG	STATE	STATECODE	STATENAME
1	US-09	.	.	.		
2	US-25	.	.	.		
3	US-23	.	.	.		
4	US-33	.	.	.		
5	US-44	.	.	.		
6	US-50	.	.	.		
7	US-09-1	0.29513	0.12434	9	CT	Connecticut
8	...					
9	US-09-1	0.28661	0.12251	9	CT	Connecticut
10	US-25-1	0.31606	0.14456	25	MA	Massachusetts
11	...					
12	US-25-2	0.32756	0.13358	25	MA	Massachusetts
13	US-23-1	0.31642	0.22689	23	ME	Maine
14	...					
15	US-23-1	0.31418	0.22684	23	ME	Maine
16	US-33-1	0.30226	0.18493	33	NH	New Hampshire
17	...					
18	US-33-1	0.3018	0.18362	33	NH	New Hampshire
19	US-44-1	0.31318	0.12859	44	RI	Rhode Island
20	...					
21	US-44-9	0.31494	0.12257	44	RI	Rhode Island
22	US-50-1	0.28953	0.17625	50	VT	Vermont
23	...					
24	US-50-1	0.27943	0.17396	50	VT	Vermont

Row	CHANGE_2020	DIVISION	REGION	XCEN	YCEN	LABEL
1	.			0.297336	0.115669	CT
2	.			0.304032	0.132749	MA
3	.			0.329596	0.188531	ME
4	.			0.302927	0.149041	NH
5	.			0.311886	0.120933	RI
6	.			0.287356	0.166911	VT
7	0.9	New England	Northeast	.	.	
8						
9	0.9	New England	Northeast	.	.	
10	7.4	New England	Northeast	.	.	
11						y

Row	CHANGE_2020	DIVISION	REGION	XCEN	YCEN	LABEL
12	7.4	New England	Northeast	.	.	
13	2.6	New England	Northeast	.	.	
14						
15	2.6	New England	Northeast	.	.	
16	4.6	New England	Northeast	.	.	
17						
18	4.6	New England	Northeast	.	.	
19	4.3	New England	Northeast	.	.	
20						
21	4.3	New England	Northeast	.	.	
22	2.8	New England	Northeast	.	.	
23						
24	2.8	New England	Northeast	.	.	

Sample Data 3: Subset of New England States with Coordinates and Change in Population for 2020

SGPLOT PROCEDURE

To produce the map of the New England states, we can use **POLYGON** plot statement to draw each state and **SCATTER** to place the state abbreviation in the center of the polygon. Similar to the stacked bar chart, we can use additional statements to help control the appearance of the graph. Refer to SAS Program 9 for the full code to produce the map.

```
proc sgplot data = ne_map;
   xaxis offsetmin = 0.01 offsetmax = 0 display = none;
   yaxis offsetmin = 0.01 offsetmax = 0 display = none;          ❶
   polygon x = LAT y = LONG id = MAPID / group = CHANGE_2020
         fill outline lineattrs = (color = grayaa);              ❷
   styleattrs datacolors = (&sixcolors);
   scatter x = XCEN y = YCEN / markerchar = LABEL;               ❸
   keylegend / position = bottom location = outside
         across = 6 noborder
         sortorder = ascending
         valueattrs = (weight = bold) titleattrs = (weight = bold)   ❹
         title = 'Population Change in 2020'
         exclude = ('.' 'YCEN');
run;
```

SAS Program 9: Map of New England States Produced with SGPLOT

❶ As mentioned previously, **XAXIS** and **YAXIS** statements control both the *features* of the axis and the structure of the axis. For map visualizations, X and Y axes are rarely required, so the **DISPLAY = NONE** option is used for both axes. **OFFSETMIN** and **OFFSETMAX** options control the minimum and maximum space reserved from the edge of the axes (i.e., how big the gaps are from the axes to the plot); although the axes won't be visible, these options ensure that the visualization will take up an appropriate amount of space.

❷ The **POLYGON** statement in PROC SGPLOT creates filled or outlined shapes, such as custom polygons or maps, using user-provided latitude and longitude (LAT and LONG, respectively). It requires a

data set structured with LAT and LONG variables and a map-id code (MAPID) variable to define separate polygons, mapping data points in order to form closed shapes. SAS allows the user to specify the variables to be used for **X**, **Y**, and **ID**, whether **GROUP** is needed to display colors by each unique value for the indicated variable (e.g., CHANGE_2020). Furthermore, using the **FILL** option indicates that the polygons should be "filled" with color. By default, polygons use **NOFILL**, which hides the fill color. In addition, the **OUTLINE** option indicates outlines of the polygons are desired, and if so, the color can be specified using **LINEATTRS** with the **COLOR** option. In other words, the polygon statement allows fine control of every aspect of visualizing the polygon shape(s).

❸ The **SCATTER** statement in PROC SGPLOT creates high-quality scatter plots to analyze relationships between numeric variables. It can also be used to "SCATTER" annotations such as character labels and images over another plot. Like the **POLYGON** statement, the **SCATTER** statement allows the user to specify the variables used for **X** and **Y**. For **SCATTER** markers, MAPID is irrelevant as SAS relies on the LAT and LONG coordinates to place markers, and the values of variable LABEL are associated with specific LAT and LONG coordinates. The **MARKERCHAR** option in SG procedures (e.g., **SCATTER**, **SERIES**) replaces standard scatter plot symbols with characters from a data variable. It is used for labeling individual data points with text as opposed to a marker (in our case, the variable LABEL which contains state abbreviations) and can be used with **MARKERCHARATTRS** (not shown) to customize text color, size, and weight. If this option is omitted, standard markers will be used instead of the text label.

❹ The **KEYLEGEND** statement in PROC SGPLOT allows similar control over how a legend appears. **VALIGN** controls alignment of text within the legend "box", **BORDER/NOBORDER** controls whether or not a border surrounds the legend, **ACROSS** option controls how many legend items to display across before wrapping, and various attribute (e.g., **TITLEATTRS**, **VALUEATTRS**) statements control the appearance of respective element. The **SORTORDER** options allow us to specify the order in which different legend items appear. For example, Figure 4 below demonstrates that the default sort order places the boxes in alphabetic order by state name (because the data is sorted by state), when it makes more sense to see the legend boxes ordered by increasing value (i.e. ascending). Similarly, the **EXCLUDE** option prevents variables and values that weren't meant to be mapped but are necessary to the production of the visualization from appearing in the legend. In this case, the stacking of the data set causes some variables to have missing values, and an extraneous LONG value (renamed from YCEN) is created for the SCATTERPLOT. These are excluded from the legend.

Additionally, order is important – the **SCATTER** statement must follow the **POLYGON** statement, or the state abbreviations will not appear.

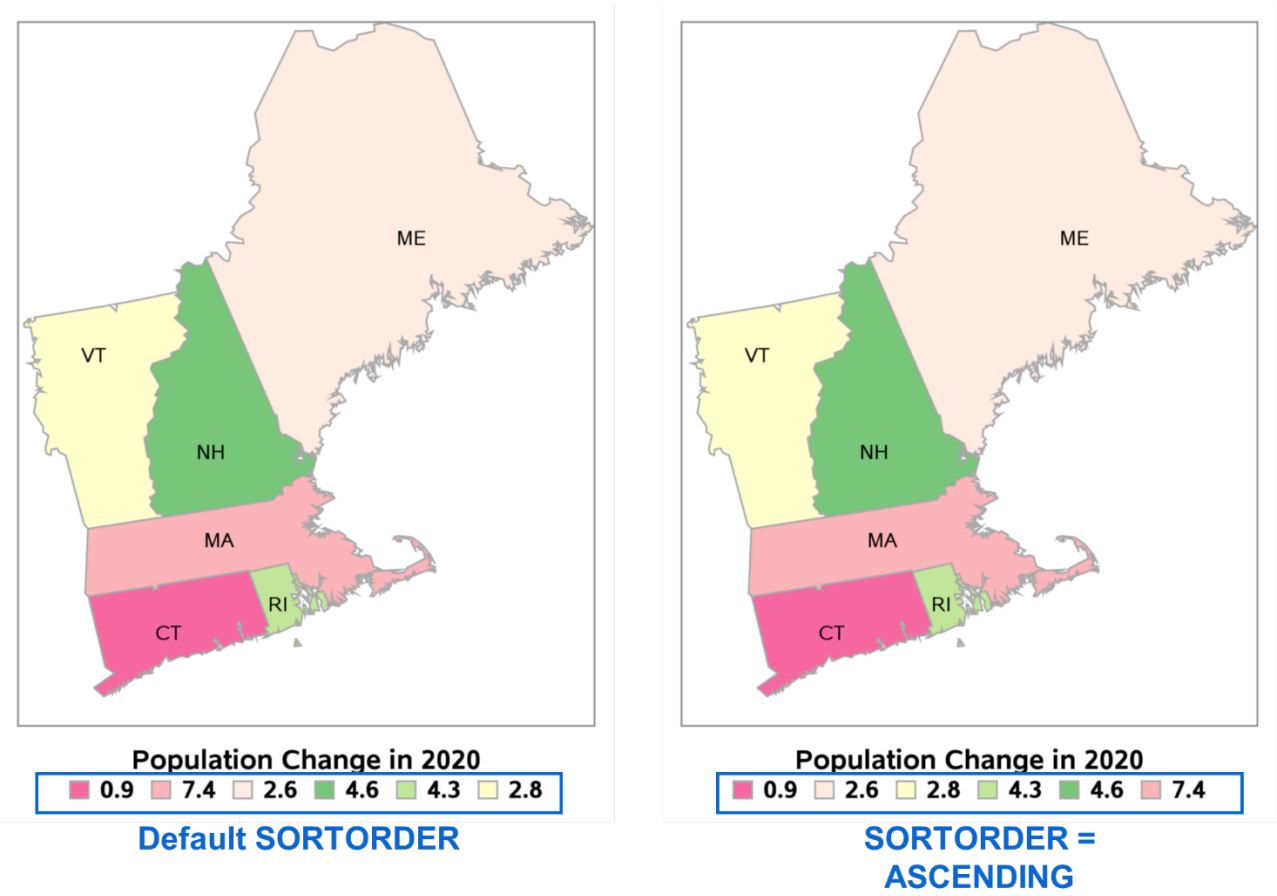

Figure 4: SORTORDER for Legend Items

GTL (TEMPLATE AND SGRENDER PROCEDURES)

We can use the **TMPLOUT** option on PROC SGPLOT statement in SAS Program 9 to generate the GTL version of the program. As with the stacked bar chart, the code is executable, but it is not easy to read and there is a lot of unnecessary code (see SAS Program 10).

```
proc template;
define statgraph sgplot;
begingraph / collation=binary dataColors=( CXF768A1 CXFBB4B9 CXFEEBE2
CX78C679 CXC2E699 CXFFFFCC );
EntryTitle "Map of New England Stats - Produced with SGPLOT" /;
layout overlay / x2axisopts=(labelFitPolicy=Split) xaxisopts=( display=none
labelFitPolicy=Split offsetmin=0.01 offsetmax=0 type=linear ) yaxisopts=(
display=none offsetmin=0.01 offsetmax=0 type=linear )
x2axisopts=(labelFitPolicy=Split);
   PolygonPlot X='LAT'n Y='LONG'n ID='MAPID'n / Group='CHANGE_2020'n
OutLineAttrs=( Color=CXAAAAAA) Display=( Outline Fill )
LegendLabel="Projected Latitude: Albers" NAME="POLYGON";
   ScatterPlot X='XCEN'n Y='YCEN'n / subpixel=off primary=true
MarkerCharacter='LABEL'n LegendLabel="YCEN" NAME="SCATTER";
   DiscreteLegend "POLYGON" "SCATTER" / Location=Outside across=6
valign=bottom Title="Population Change in 2020" TitleAttrs=( Weight=bold)
ValueAttrs=( Weight=bold) Exclude=( "." "YCEN") sortorder=ascendingFormatted
Border=false;
endlayout;
endgraph;
end;
run;
```

SAS Program 10: Template Produced from SGPLOT for Map of New England States

Things like **COLLATION = BINARY**, **ENTRYTITLE**, **X2AXISOPTS**, **Y2AXISOPTS**, **LEGENDLABEL**, as well as other statements and options can be removed. SAS Program 11 shows the cleaned-up code.

```
proc template;
   define statgraph ne_poly;
      begingraph / datacolors = (&sixcolors);
         layout overlay / xaxisopts = (display = none
                          offsetmin = 0.01 offsetmax = 0)
                          yaxisopts = (display = none
                          offsetmin = 0.01 offsetmax = 0);
            polygonplot x = LAT y = LONG id = MAPID / group = CHANGE_2020
                          outlineattrs = (color = CXAAAAAA)
                          display = (outline fill)
                          name = "polygonplot";
            scatterplot x = XCEN y = YCEN / markercharacter = LABEL;
            discretelegend "polygonplot" / location = outside
                          across = 6 valign = bottom
                          sortorder = ascendingformatted
                          title = "Population Change in 2020"
                          titleattrs = (weight = bold)
                          valueattrs = (weight = bold)
                          exclude = ("." "YCEN")
                          border = false;
         endlayout;
      endgraph;
   end;
run;
```

❶

SAS Program 11: Map of New England States Produced with GTL

❶ With the **LAYOUT OVERLAY**, the **XAXISOPTS** and **YAXISOPTS** options are used to specify axes options. These options have their own options that allow for controlling what is to be displayed and whether or not the plot needs to be moved off the axes. **DISPLAY = NONE** indicates that no axis features

are to be displayed for either the X or Y axis. The default is to display the label, line, ticks and tickvalues, but with a map (i.e., **POLYGON** plot) it doesn't make sense to display the axes. To reserve space near the axes (i.e., to not draw the graph up against the axes), the **OFFSETMIN** and **OFFSETMAX** options can be used. **OFFSETMIN** reserves space at the minimum end of the axis while **OFFSETMAX** reserves the space at the maximum end. The allowed values are a decimal portion of the full axis length (i.e., 0 to 1). In our example, only space on the minimum end is being reserved.

SIDE-BY-SIDE PLOTS

Now that we have the stacked bar chart and the map as separate graphs, it is time to put them together. There are two approaches within GTL that can be used to put these two graphs together. To build the desired output, all the data must reside within one SAS data set. Sample Data 4 shows the combined date from Sample Data 2 and Sample Data 3.

Row	MAPID	LAT	LONG	STATECODE	CHANGE_2020
1	US-09				
2	US-25				
3	US-23				
4	US-33				
5	US-44				
6	US-50				
7	US-09-01	0.29513	0.12434	CT	0.9
8	US-09-01	0.29525	0.12367	CT	0.9
9	...				
10				VT	
11				VT	

Row	XCEN	YCEN	LABEL	YEAR	POPULATION
1	0.297336	0.115669	CT		
2	0.304032	0.132749	MA		
3	0.329596	0.188531	ME		
4	0.302927	0.149041	NH		
5	0.311886	0.120933	RI		
6	0.287356	0.166911	VT		
7					
8					
9					
10				2010	625,741
11				2020	643,077

Sample Data 4: Combination of Sample Data 2 and Sample Data 3 Stacked

GUTTERS, HEADERS AND SIDEBARS

Before proceeding with putting the two graphs together, we need to understand the different areas of the graphs. As a graph becomes more and more complex, we need to consider things like gutters, headers

(cell, row, column) and sidebars (top, bottom, left, right) (Harris & Watson, 2020). Figure 5 illustrates where each of these could reside within a complex graph.

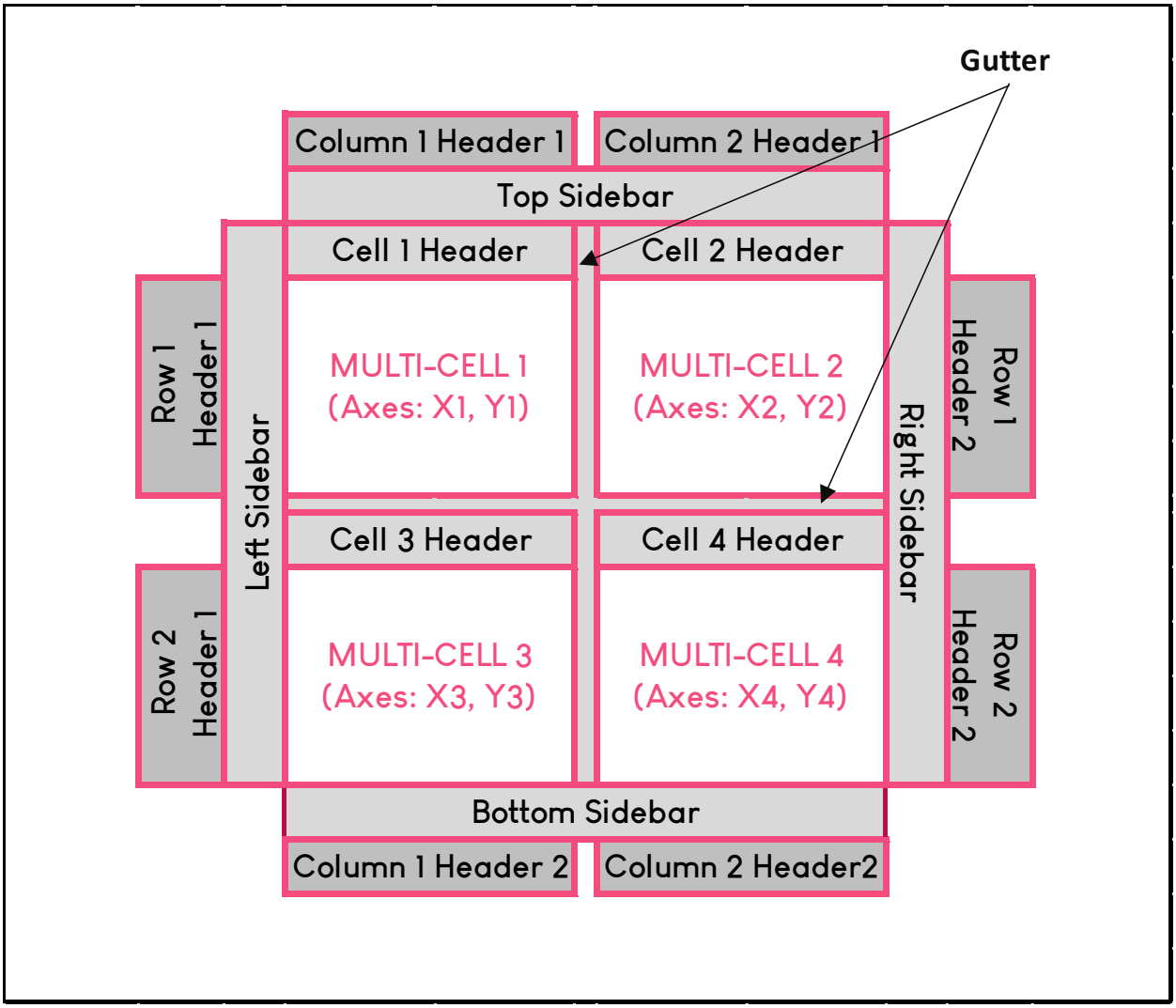

Figure 5: Complex Graph Area – Dealing with Gutters, Headers, and Sidebars

LAYOUT GRIDDED

To produce a side-by-side plot, we need to break the graph into two cells. This can be done using **LAYOUT GRIDDED**. Within **LAYOUT GRIDDED**, we nest the layouts for each cell. Refer to SAS Program 12 for illustration of **LAYOUT GRIDDED**.

```
proc template;
   define statgraph sbysgrid;
      begingraph;
         layout gridded / columns = 2 rows = 1;    ❶
            /** create the right side of the graph **/
            layout overlay / << axes options >>;
               << POLYGON STATEMENT >>
               << SCATTER PLOT STATEMENT >>          ❷
               << DISCRETELEGEND STATEMENT >>
            endlayout; /** overlay for left side **/
            /* create the right side of the graph */
            layout overlay / << axes options >>;
               << BARCHART STATEMENT >>
               << DISCRETELEGEND STATEMENT >>        ❸
            endlayout; /** overlay for right side **/
         endlayout; /** end of GRIDDED layout **/    ❶
      endgraph;
   end;
run;
```

SAS Program 12: Side-by-Side Plots Produced with LAYOUT GRIDDED

❶ **LAYOUT GRIDDED** produces a multi-cell plot. The number of cells in the plot is determined by the **COLUMNS** and **ROWS** options. In this example, there are 2 columns and one row (i.e., 2 cells). If **COLUMNS = 3** and **ROWS = 2**, then there would be 6 cells. Each cell is defined within the template and defined within the **LAYOUT GRIDDED**. Be sure to 'sandwich' the nested layouts within the **LAYOUT GRIDDED**. We can further control the cells by indicating the order in which they are populated **ORDER** = **ROWMAJOR** or **COLUMNMAJOR** (not shown). There are other options available which are not discussed in this paper but can be viewed at LAYOUT GRIDDED Statement webpage on SAS website.

❷ Because there are two cells each cell has its own layout and definition. The left side of the graph is a map of the New England states. Thus, the first cell is a copy of the **LAYOUT OVERLAY** block found in SAS Program 11.

❸ The right side of the graph is the stacked bar chart of the population by years. The second cell is a copy of the **LAYOUT OVERLAY** block found in SAS Program 7 with the only difference being that in the **DISCRETELEGNED** statement, the **EXCLUDE** option is used. **EXCLUDE = (" ")** is added so that records with missing STATECODE are not added to the legend. Remember that we are stacking the latitude and longitude data with the change in population data and those records have missing STATECODE.

Order is important when defining the cells, if the **LAYOUT OVERLAY** blocks within **LAYOUT GRIDDED** are switched then the stacked bar chart would be on the left side instead of the right side.

Figure 6 shows the output produced by SAS Program 12. This isn't quite right. Notice that the map is 'stretched' and the states look distorted while the stacked bar charts looked 'squished'. This is due to the fact that with **LAYOUT GRIDDED** each cell is the same size.

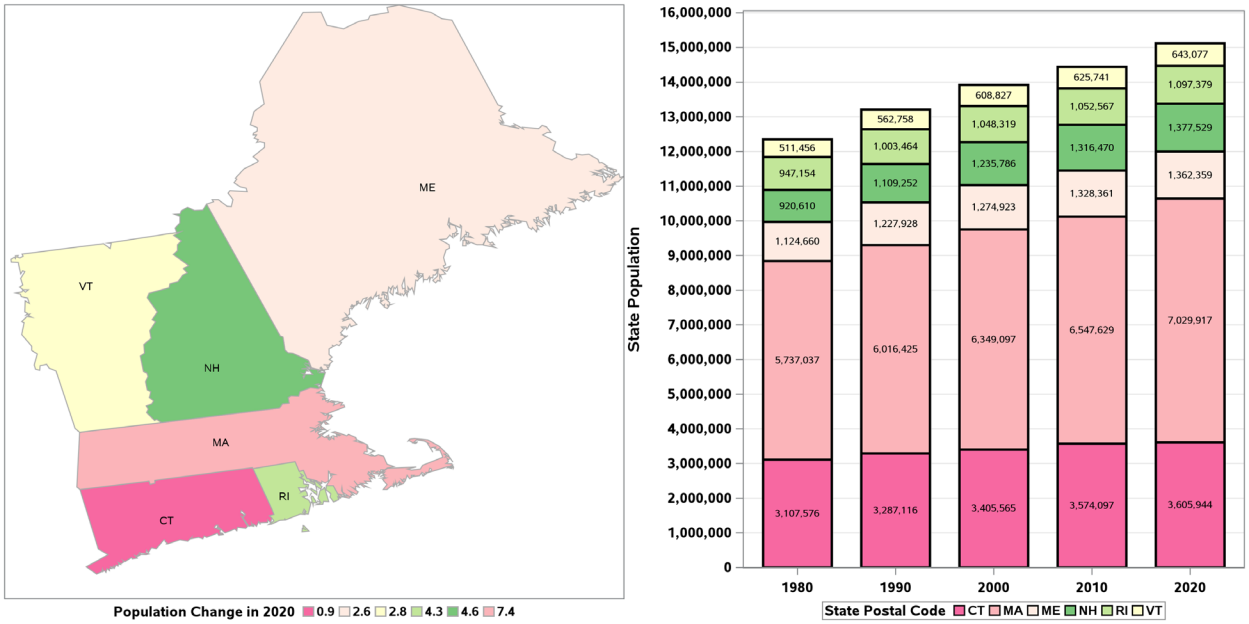

Figure 6: Side-by-Side Plot Produced Using LAYOUT GRIDDED

LAYOUT LATTICE

To fix the distortion of the graph so that it looks like the desired output, we can use **LAYOUT LATTICE**. **LAYOUT LATTICE** allows us to control the size of each cell. SAS Program 13 shows the syntax needed to produce the side-by-side plot using a lattice. Everything else in the program is the same as SAS Program 12.

```
proc template;
   define statgraph sbyslat;
      begingraph;
         layout lattice / columns = 2 rows = 1
                columnweights = (0.4 0.60)
                columngutter = 50;
            /** create the right side of the graph **/
            layout overlay / << axes options >>;
               << POLYGON STATEMENT >>
               << SCATTER PLOT STATEMENT >>
               << DISCRETELEGEND STATEMENT >>
            endlayout; /** overlay for left side **/
            /* create the right side of the graph */
            layout overlay / << axes options >>;
               << BARCHAT STATEMENT >>
               << DISCRETELEGEND STATEMENT >>
            endlayout; /** overlay for right side **/
         endlayout; /** end of LATTICE layout **/
      endgraph;
   end;
run;
```

❶

SAS Program 13: Side-by-Side Plots Produced with LAYOUT LATTICE

❶ Similar to **LAYOUT GRIDDED**, we need to specify the number of columns and rows in **LAYOUT LATTICE**. While there are a number of differences between **GRIDDED** and **LATTICE** such as the union

of axes and adding side bars, for this example, we are only focusing on controlling the size of the cells and the space between the cells. To control the size of the cells when we have multiple columns we use **COLUMNWEIGHTS** and specify the size of each cell as a percentage of the grid (in this case the row). If we need to control the size of the rows, we use **ROWWEIGHTS**. **COLUMNGUTTER = 50** indicates we want 50 pixels between the two graphs. The default is 0 pixels. If there were multiple rows and we wanted space between the rows, we use **ROWGUTTER**.

Figure 7 shows the results of SAS Program 13 if we change **COLUMNGUTTER = 0** or if we remove **COLUMNGUTTER** option. Figure 8 shows the results with **COLUMNGUTTER = 50**. Notice the space between the map and the label for the stacked bar chart.

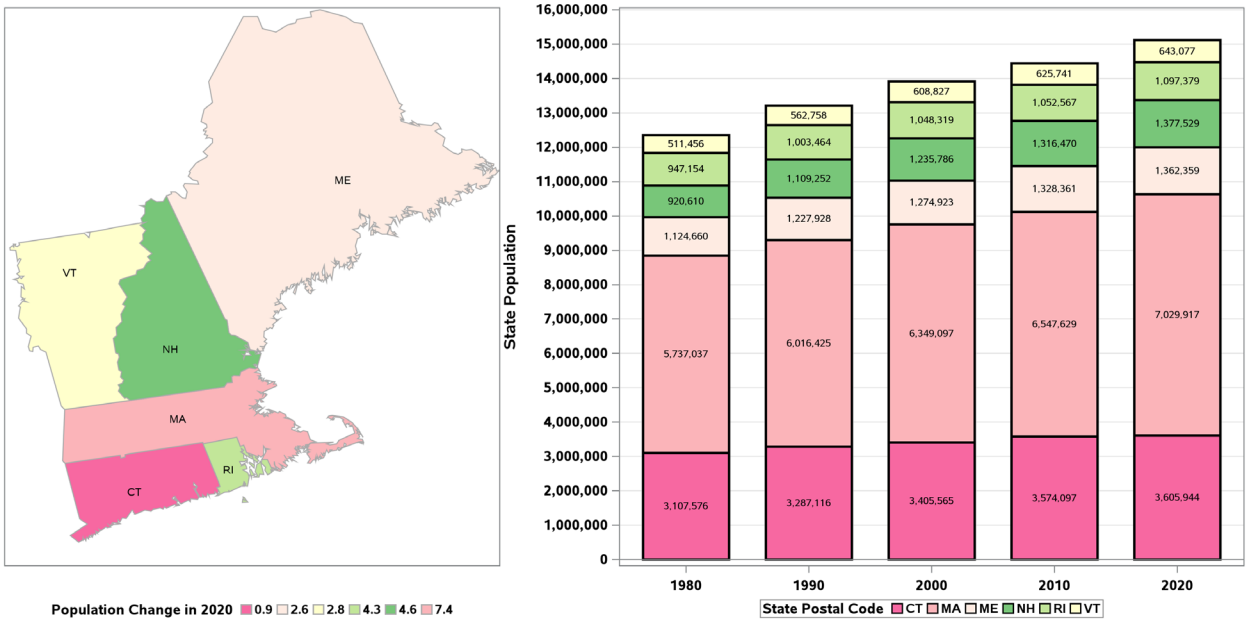

Figure 7: Side-by-Side Plot Produced Using LAYOUT LATTICE with GUTTER = 0

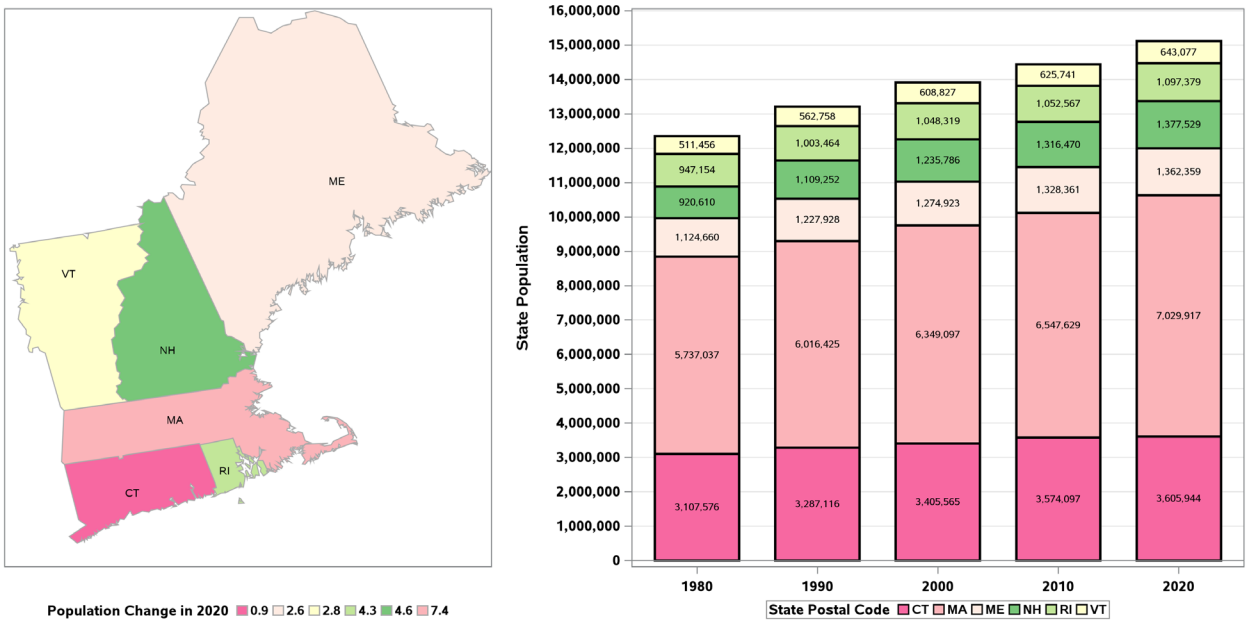

Figure 8: Side-by-Side Plot Produced Using LAYOUT LATTICE with GUTTER = 50

CONCLUSION

SAS provides myriad opportunities and tools to create complex and camera-ready visualizations of statistical data. We recommend that you start with what you know when building visualizations. It is helpful to use TMPLOUT with SG procedures to give you a head start on coding graphs. Additionally, build each component separately; then, piece all components together and modify as necessary. Once a visualization has been built in GTL, you can easily reproduce graphics with revised data and new studies. Producing graphs does not have to be a daunting task with the SAS ODS Graphics Toolkit! **G**et **T**he **L**ove with SAS Graphics Template Language.

REFERENCES AND RECOMMENDED READINGS

Hadden, L. (2013). Where in the World are SAS/Graph Maps? An Exploration of the Old and New SAS Mapping Capacities. Burlington, VT: NESUG. Retrieved from https://www.lexjansen.com/nesug/nesug13/124_Final_Paper.pdf

Hadden, L. (2016). Red Rover, Red Rover, Send Data Right Over: Exploring External Geographic Data Sources with SAS®. Cincinnati, OH: MWSUG. Retrieved from https://www.lexjansen.com/mwsug/2016/DV/MWSUG-2016-DV05.pdf

Hadden, L. (2018). Wow! You Did That Map With SAS®?! Round II. Sacramento, CA: WUSS. Retrieved from https://www.lexjansen.com/wuss/2018/32_Final_Paper_PDF.pdf

Hadden, L. (2021). Visually Exploring Proximity Analyses Using SAS PROC GEOCODE and SGMAP and Public Use Data Sets. PharmaSUG. Retrieved from https://www.lexjansen.com/pharmasug/2021/DV/PharmaSUG-2021-DV188.pdf

Hadden, L. (2023). SAS® PROC GEOCODE by Example: A Case Study. Las Vegas, NV: SAS Explore. Retrieved from https://communities.sas.com/t5/SASExplore-Presentations/SAS-PROC-GEOCODE-by-Example-A-Case-Study/ta-p/896670

Hadden, L. (2024). A Deep Dive into Enhancing SAS/GRAPH® and SG Procedural Output with Templates, Styles, Attributes, and Annotation. Baltimore, MD: PharmaSUG. Retrieved from https://www.lexjansen.com/pharmasug/2024/PO/PharmaSUG-2024-PO-128.pdf

Harris, K. (2017). Hands-on Graph Template Language: Part B. *SAS Global Forum.* Orlando, FL. Retrieved from http://support.sas.com/resources/papers/proceedings17/0864-2017.pdf

Harris, K., & Watson, R. (2018). Animate Your Data! *SAS Global Forum.* Denver, CO. Retrieved from Publications: https://support.sas.com/resources/papers/proceedings18/1817-2018.pdf

Harris, K., & Watson, R. (2018). Great Time to Learn GTL. *PharmaSUG.* Seattle, WA. Retrieved from https://pharmasug.org/proceedings/2018/EP/PharmaSUG-2018-EP18.pdf

Harris, K., & Watson, R. (2019). Interactive Graphs. *SAS Global Forum.* Dallas, TX. Retrieved from https://support.sas.com/resources/papers/proceedings19/3261-2019.pdf

Harris, K., & Watson, R. (2020). *SAS® Graphics for Clinical Trials by Example.* Cary: SAS Institute Inc.

Heath, D. (2016). Now You Can Annotate Your GTL Graphs! *PharmaSUG.* Denver, CO. Retrieved from https://www.lexjansen.com/pharmasug/2016/DG/PharmaSUG-2016-DG01.pdf

Heath, D. (2018). Diving Deep into SAS® ODS Graphics Styles. *Proceedings of PharmaSUG.* Seattle: PharmaSUG. Retrieved from https://www.pharmasug.org/proceedings/2018/DV/PharmaSUG-2018-DV02.pdf

Heath, D. (2024). Building Complex Graphics from Simple Plot Types. *PharmaSUG.* Baltimore: MD. Retrieved from https://www.lexjansen.com/pharmasug/2024/HT/PharmaSUG-2024-HT-197.pdf

Heath, D. (2024). Building Complex Graphics from Simple Plot Types. Baltimore, MD: PharmaSUG. Retrieved from https://pharmasug.org/proceedings/2024/HT/PharmaSUG-2024-HT-197.pdf

Kuhfeld, W. (2015). *Advanced ODS Graphics Examples.* Cary: SAS Institute Inc. Retrieved from https://support.sas.com/documentation/prod-p/grstat/9.4/en/PDF/odsadvg.pdf

Kuhfeld, W. (2016). *Basic ODS Graphics Examples.* Cary: SAS Institute Inc. Retrieved from http://support.sas.com/documentation/prod-p/grstat/9.4/en/PDF/odsbasicg.pdf

Kuhfeld, W., & So, Y. (2013). Creating and Customizing the Kaplan-Meier Survival Plot in PROC LIFETEST. *SAS*

Global Forum. San Francisco, CA. Retrieved from https://support.sas.com/resources/papers/proceedings13/427-2013.pdf

Matange, S. (2013). *Getting Started with the Graph Template Language in SAS®.* Cary: SAS Institute Inc.

Matange, S. (2014). Up Your Game with Graph Template Language Layouts. *PharmaSUG.* San Diego, CA. Retrieved from https://pharmasug.org/proceedings/2014/DG/PharmaSUG-2014-DG14-SAS.pdf

Matange, S. (2016). *Clinical Graphs Using SAS®.* Cary: SAS Institute Inc.

Matange, S. (2018). Advanced Graphs using Axis Tables. *SAS Global Forum.* Denver, CO. Retrieved from https://support.sas.com/resources/papers/proceedings18/2180-2018.pdf

Matange, S., & Heath, D. (2019). Create A Combined Graph of Tumor Data. *SAS Global Forum.* Dallas, TX. Retrieved from https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2019/3143-2019.pdf

Meyers, J. (2024). Combining Functions and the POLYGON Plot to Create Unavailable Graphs Including Sankey and Sunburst Charts. Baltimore, MD: PharmaSUG. Retrieved from https://pharmasug.org/proceedings/2024/DV/PharmaSUG-2024-DV-155.pdf

SAS Institute Inc. (2023, Jul 19). *SGPLOT Procedure*. Retrieved Dec 2023, from SAS® 9.4 and SAS® Viya® 3.5 Programming Documentation: https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/grstatproc/n0yjdd910dh59zn1toodgupaj4v9.htm

SAS Institute Inc. (2025, May 5). *SAS® 9.4 Graph Template Language: User's Guide, Fifth Edition*. Retrieved Mar 2026, from SAS® 9.4 and SAS® Viya® 3.5 Programming Documentation: https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/grstatug/titlepage.htm

SAS Institute Inc. (2026, Feb 13). *Components of a Graph*. Retrieved Mar 2026, from SAS® 9.4 and SAS® Viya® 3.5 Programming Documentation: https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/grstatgraph/p1wj7tdd218vndn1lxd6r9ug6xxe.htm

SAS Institute Inc. (2026, Feb 13). *Flexible Templates*. Retrieved Mar 2026, from SAS® 9.4 and SAS® Viya® 3.5 Programming Documentation: https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/grstatgraph/p046os95mnigznn18xf9b9ik77jj.htm

SAS Institute Inc. (2026, Feb 13). *LAYOUT GRIDDED Statement*. Retrieved Mar 2026, from SAS® 9.4 and SAS® Viya® 3.5 Programming Documentation: https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/grstatgraph/p1h7wd5z8ihewzn1vy7htk5tu7nr.htm

SAS Institute Inc. (2026, Feb 13). *SAS® 9.4 Graph Template Language: Reference, Fifth Edition*. Retrieved Mar 2026, from SAS® 9.4 and SAS® Viya® 3.5 Programming Documentation: https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/grstatgraph/titlepage.htm

SAS Institute Inc. (n.d.). *Graph Template Language Tip Sheet*. Retrieved from https://support.sas.com/rnd/app/ODSGraphics/TipSheet_GTL.pdf

SAS Institute Inc. (n.d.). *Graph Template Modification Tip Sheet*. Retrieved from https://support.sas.com/rnd/app/ODSGraphics/TipSheet_GraphTemplateModification.pdf

Watson, R. (2019). Great Time to Learn GTL: A Step-by-Step Approach to Creating the Impossible. Dallas, TX: SAS Global Forum. Retrieved from https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2019/3170-2019.pdf

Watson, R. (2020). What's Your Favorite Color? Controlling the Appearance of a Graph. *SAS Global Forum.* Retrieved from https://support.sas.com/resources/papers/proceedings20/4660-2020.pdf

Watson, R., & Hadden, L. (2021). "Bored"-Room Buster Bingo - Create Bingo Cards Using SAS® ODS. PharmaSUG. Retrieved from https://pharmasug.org/proceedings/2021/AP/PharmaSUG-2021-AP-030.pdf

Watson, R., & Horstman, J. (2024). Complex Custom Clinical Graphs Step by Step with SAS® ODS Statistical Graphics. Baltimore, MD: PharmaSUG. Retrieved from https://pharmasug.org/proceedings/2024/HT/PharmaSUG-2024-HT-413.pdf

ACKNOWLEDGMENTS

We would like to express our gratitude to Dan Heath of SAS and Sanjay Matange (formerly of SAS) who are our ODS Graphics idols and mentors!

CONTACT INFORMATION

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

Richann Jean Watson
richann.watson@datarichconsulting.com
http://www.datarichconsulting.com/
https://www.linkedin.com/in/richann-watson-31435422/

Louise S. Hadden
louisesquibbhadden@gmail.com
https://github.com/TheGirlWiththeSASTattoo
linkedin.com/in/louisehadden