Paper 3432-2015

Getting Your Hands on Reproducible Graphs

Rebecca Ottesen and Leanne Goldstein

City of Hope, Duarte, California

ABSTRACT

Learning the SAS® Graph Template Language (GTL) may seem like a daunting task. However, creating customized graphics with SAS® is quite easy using many of the tools offered with Base SAS® software. The point-and-click interface of ODS Graphics Designer provides us with a tool that can be used to generate highly polished graphics and to store the GTL-based code that creates them. This opens the door for users who would like to create canned graphics that can be used on various data sources, variables, and variable types. In this hands-on training, we explore the use of ODS Graphics Designer to create sophisticated graphics and save the template code. We then discuss modifications using basic SAS® macros in order to create stored graphics code that is flexible enough to accommodate a wide variety of situations.

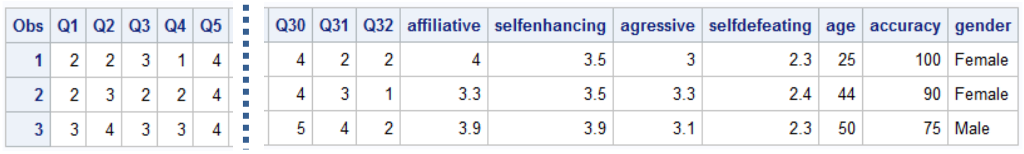
INTRODUCTION

Creating statistical graphics in SAS is easy and fun. There are many tools that can be used to make plots such as the SG procedures that come standard with Base SAS, using ODS GRAPHICS ON statements for default graphics with certain statistical procedures, and using ODS Graphics Designer. With each of these methods it is fairly simple to create graphics with a customized look and feel. In addition, SAS also has the ability to delve deeper into creating graphics through the use of GTL coding in PROC TEMPLATE.

While somewhat complicated, GTL provides complete control over graphic options and offers the advantage of being able to create reproducible graphics in a template. We will discuss the use of dynamic variables within a template and basic macro coding that can be incorporated in order to create flexible graphics for new variables and data sets. We offer a simple solution for generating GTL and template code through the use of ODS Graphics Designer. The Designer is a logical starting point for creating customized polished graphics in order to modify template code to reproduce canned graphics as desired. The examples in this workshop are all generated using SAS v 9.4.

EXAMPLES

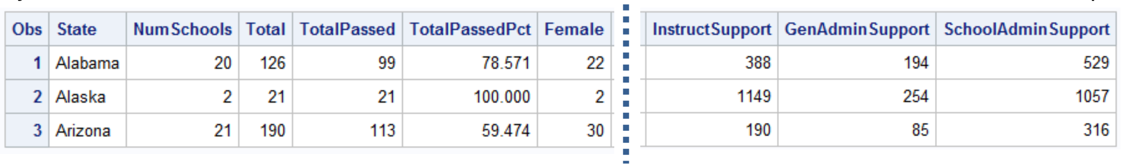
We will work with four main example data sets in this workshop. The first contains data collected from an online survey based on the Humor Styles Questionnaire form (Martin 2003). The questions on the survey, denoted as Q1 through Q32, are statements that were rated on a five point scale where -1 = did not select an answer, 1 = never or very rarely true, 2 = rarely true, 3 = sometimes true, 4 = often true, 5 = very often or always true. The questions were also scored and grouped as four types of humor styles; affiliative, self-enhancing, aggressive, and self-defeating. In addition, age, gender, and subject perception of the answer accuracy were also obtained. The first three observations from this data set are shown in Output 1.



Obs	Q1	Q2	Q3	Q4	Q5	Q30	Q31	Q32	affiliative	selfenhancing	agressive	selfdefeating	age	accuracy	gender
1	2	2	3	1	4	4	2	2	4	3.5	3	2.3	25	100	Female
2	2	3	2	2	4	4	3	1	3.3	3.5	3.3	2.4	44	90	Female
3	3	4	3	3	4	5	4	2	3.9	3.9	3.1	2.3	50	75	Male

Output 1. Data from the Humor Styles Questionnaire

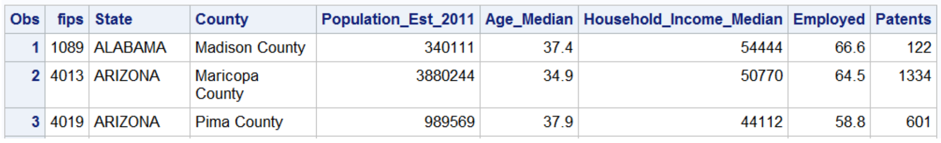
Our second example uses data compiled from the College Board (Ericson 2014) for all U.S. states participating in the 2013 High School Advanced Placement exam for Computer Science (AP CS A). These data include totals and passing rates overall as well as by minority races and gender. In addition, data from the 2012 U.S. census (Annual Survey of School System Finances 2012) regarding per pupil spending in elementary and secondary school systems by state were combined with the AP CS A exam results. The first three observations are shown in Output 2.



Obs	State	NumSchools	Total	TotalPassed	TotalPassedPct	Female	InstructSupport	GenAdminSupport	SchoolAdminSupport
1	Alabama	20	126	99	78.571	22	388	194	529
2	Alaska	2	21	21	100.000	2	1149	254	1057
3	Arizona	21	190	113	59.474	30	190	85	316

Output 2. AP Computer Science Exam and Per Pupil Support

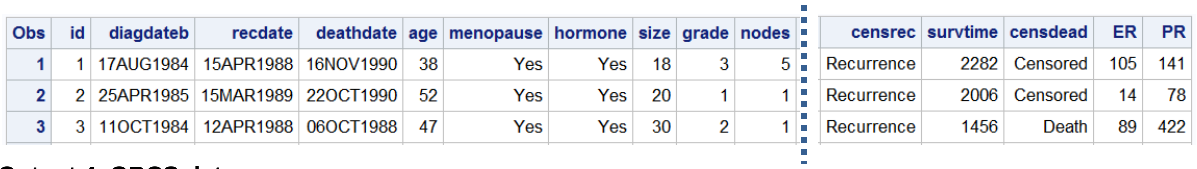
Our third example consists of data filed at the U.S. Patent and Trademark Office regarding the number of utility patent grants (patents for inventions) by U.S. county for those counties filing at least 100 patents in the year 2011 (U.S. State Patent Breakout by Regional Component 2014). In addition, this data set also includes certain demographic variables from the 1-year U.S. Census American Community Survey regarding selected social characteristics for counties with a population of 65,000 or more (American Community Survey 2014). Output 3 shows the first three observations from this data set.



Obs	fips	State	County	Population_Est_2011	Age_Median	Household_Income_Median	Employed	Patents
1	1089	ALABAMA	Madison County	340111	37.4	54444	66.6	122
2	4013	ARIZONA	Maricopa County	3880244	34.9	50770	64.5	1334
3	4019	ARIZONA	Pima County	989569	37.9	44112	58.8	601

Output 3. U.S. Patents data

Our fourth example uses the German Breast Cancer Study (GBCS) Data from Applied Survival Analysis: Regression Modeling of Time to Event Data: Second Edition (Hosmer, Lemeshow, and May 2008). Output 4 shows the first three observations from this data set.



Obs	id	diagdateb	recdate	deathdate	age	menopause	hormone	size	grade	nodes	censrec	survtime	censdead	ER	PR
1	1	17AUG1984	15APR1988	16NOV1990	38	Yes	Yes	18	3	5	Recurrence	2282	Censored	105	141
2	2	25APR1985	15MAR1989	22OCT1990	52	Yes	Yes	20	1	1	Recurrence	2006	Censored	14	78
3	3	11OCT1984	12APR1988	06OCT1988	47	Yes	Yes	30	2	1	Recurrence	1456	Death	89	422

Output 4. GBCS data

DATA BASICS 101

Suppose your boss gives you a data set to analyze. What do you do first? Do you start generating models right away? Do you calculate univariate statistics or produce graphs? Does the data need massaging? The first step in any data analysis should be to understand what data you have. In addition to gathering the proper background information about the study and main questions of interest, there are also initial coding steps that should be considered. Assuming that you have a SAS data set, running a PROC CONTENTS on the data is critical in order to provide important information including which values are character and which are numeric. Once this information is known, you can begin to examine the distributions of the variables.

For categorical variables running a PROC FREQ to obtain the frequencies and percentages will provide a useful first insight to the data. Using the _CHARACTER_ keyword allows you to specify all character type variables in the data set, if desired, without listing them individually.

```
PROC FREQ DATA = MYDATA;
  TABLE _CHARACTER_;
RUN;
```

The PROC FREQ results will help you see types of responses for each categorical variable, and which variables have missing data and any unusual responses, for example, coded missing values like “UNKNOWN”.

For quantitative data, such as continuous or large discrete data, you can generate summary statistics using PROC UNIVARIATE. UNIVARIATE will allow you to ascertain if the data are normally distributed by adding the NORMAL option. Using the _NUMERIC_ keyword tells SAS to use only the numeric variables in the data set, if desired, without listing them individually in the VAR statement. Be sure that you understand your data set before limiting the data to all numeric or all character variables. For example, sometimes categorical data can be coded numerically such as gender where 1 = female and 2 = male, so you may still want a frequency count of this numeric variable.

```
PROC UNIVARIATE DATA = MYDATA;
  VAR _NUMERIC_;
RUN;
```

The PROC UNIVARIATE results will show you a plethora of information including the minimum and maximum values to identify if there are unusual numeric data, such as 999 to represent an unknown response. The UNIVARIATE results also show you the number of null responses, measures of central tendency such as mean and median, and measures of variation such as standard deviation, variance, and range. PROC UNIVARIATE also has the ability to produce statistics on numeric variables within groups by including a CLASS statement for the grouping variable.

Often we have data that are recorded as numeric but would be better analyzed as meaningful categories. This decision might be made based on previous knowledge about the variable and how it is typically reported. Or it could be based on the violation of the assumption of normality for a continuous variable. An example of modifying a quantitative variable so that it can be treated as categorical might be if you wish to analyze age in meaningful groupings. This can be carried out easily by incorporating a PROC FORMAT to group the data and then applying the format in PROC FREQ.

```
PROC FORMAT;
VALUE agedx  0 -< 50 = '0 - <50 years'
            50 -< 70 = '50 - <70 years'
           70 - HIGH = '70+ years';
RUN;

PROC FREQ DATA = MYDATA;
  TABLE Age;
   FORMAT Age agedx.;
RUN;
```

These very basic methods of generating summary statistics of the data set are a good start. But to really understand the data, it is also important to visualize it. For many investigators and clients (who you will be presenting the data to) it may be easier to understand visual representations of the data rather than getting lost in the numbers.

However, before moving on to our focus on generating polished and reproducible graphics, it is important to note that we are bypassing some vital data analysis steps. First we assume that you understand the scope of the project, how the data were obtained and the main questions of interest. We also assume that the data are ready for analysis, which is often not the case for real life data that rarely come prepackaged and ready to go. And, of course, we assume that you have a basic understanding of statistics. These are huge topics and the focus of many books and college level courses. In this workshop we will focus mainly on the aspect of building plots that you may need to assess your data in preparation for analysis. In addition, we will explore the idea of enhancing these graphs so that they can be used in presentations and even be packaged to reproduce plots on similar variable types or other data sets.

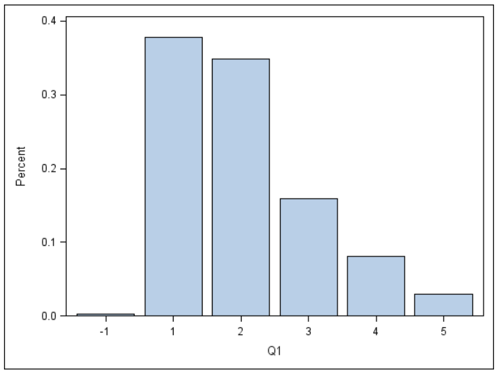


Figure 1. Plot for univariate categorical data

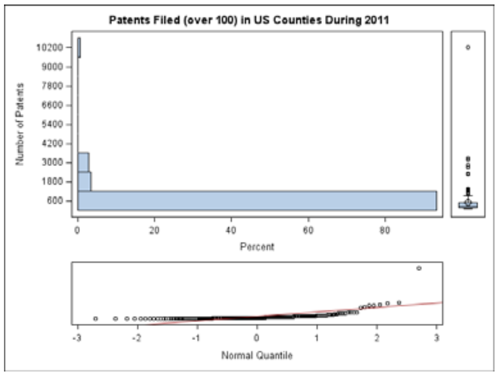


Figure 2. Plots for univariate quantitative data

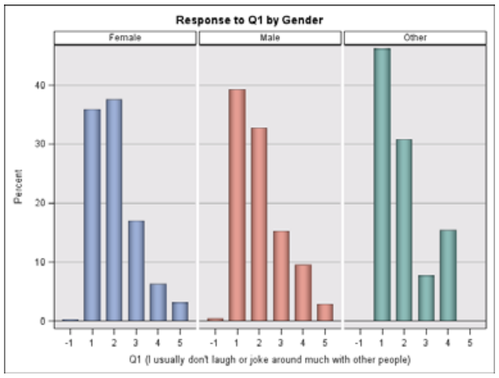


Figure 3. Grouped bar chart

PLOTS BASICS 101

UNIVARIATE DATA

An easy way to start visualizing categorical and certain discrete data is by creating a simple bar chart. Figure 1 is an example of responses for the first question in our example questionnaire data. It is evident from this bar chart that there are few -1 responses (did not select an answer) and the most frequent response for Q1 is a 1 response (never or very rarely true) representing approximately 38% of the data.

For continuous and large discrete data there are several basic plots that can be examined: a histogram, box plot, or normal quantile plot. Each of these plots makes it easy to identify unique features of the data. Figure 2 contains an example of a graphic on our example data set containing number of patents by county. Each section or cell within the plot contains a different graphic summarizing the data. The horizontal histogram in the top left cell shows that the data are positively skewed, i.e. there are several large outlier values. The corresponding box plot in the right cell incorporates measures of center (the median and mean), measures of spread (lower and upper quartiles and interquartile range), and outlier values. Finally, the normal quantile plot in the bottom cell demonstrates whether or not the data are normally distributed. We can see in this lower cell that the patent data are not normally distributed as the data do not line up exactly with the red line.

BIVARIATE DATA

Univariate plots can only take us so far. For some projects we might want to look at two variables in relationship to each other. There are several plots that can be assembled depending on the data type. If one of the variables is a categorical grouping variable, such as gender or race, we can make a grouped bar chart or a grouped histogram. Figure 3 shows the bar chart for Q1 of our example questionnaire data, but this time it is grouped by gender. This allows us to assess how gender impacts subject response to Question 1. In addition, certain graphics options have been applied to make the plot more aesthetically appealing. A quick comparison of Figure 1 to Figure 3 reveals that while overall the most frequent response to Q1 was a 1 (never or very rarely true), when grouping the responses by gender, females most often respond to Q1 with 2 (rarely true) while males and other gender most often respond with 1.

When both variables are continuous, we can make a scatter plot. Figure 4 shows a scatted plot of overall passing rate by per pupil spending for the AP Computer Science Exam example. This figure is enhanced with additional cells of univariate boxplots for each of the quantitative variables that provide important information about the underlying distribution for each individual variable.

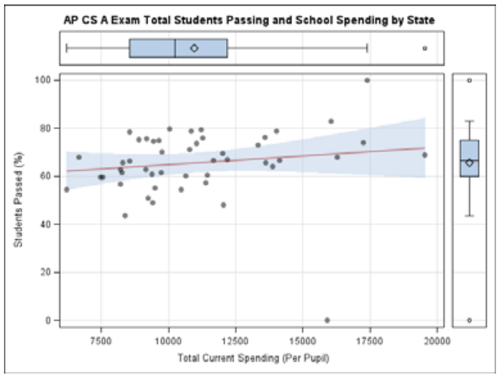


Figure 4. Scatter plot with univariate boxplots

PLOTS FOR REPORTING

SAS can also be used to create polished graphics for reports, presentations, and manuscripts. We will explore the use of various options to enhance graphics as well as combining more than one plot into a single display through the use of multi-cell plots (side by side) and by plot layers that overlay compatible plot types. In addition, we will explore how to customize graphics with the point-and-click interface of ODS Graphics Designer.

Figure 5 presents Microsoft Stock data from 2009-2010 that takes into consideration the time series aspect of the data. This graphic uses more than one cell to provide a snapshot of the price and volume data at each time point. We can see that the stock price climbs over time while volume has spikes at various time points along the way.

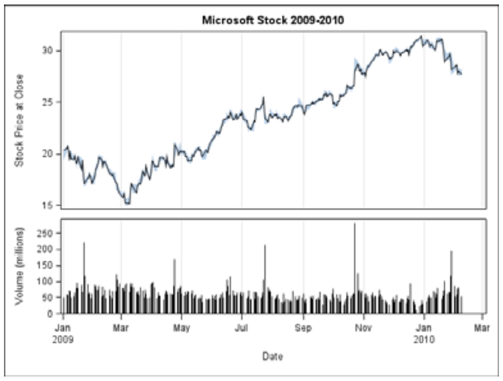
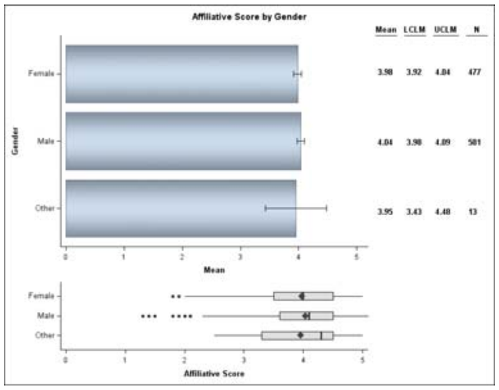


Figure 5. Multi-cell stock price and volume

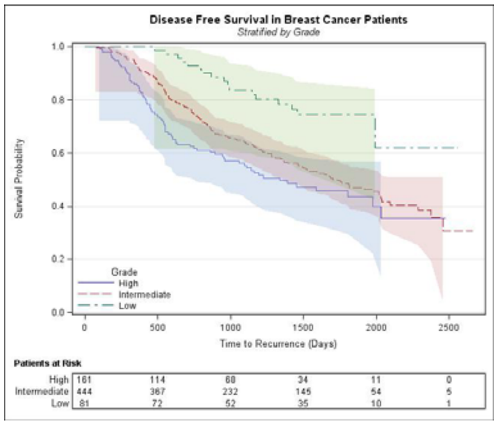
Multi-cell plots can also be used to include more information including a table of descriptive statistics as shown in Figure 6. This graphic displays the average affiliative scores by gender from the questionairre example, along with confidence limit bars and boxplots. Cells have been added to the graphic that present the descriptive statistics associated with each bar. To accomplish this with Designer, the descriptive statistics must be calculated with a procedure in SAS first, in this case PROC MEANS, and then combined back with the original raw data in order to make the plots.



	Mean	LCLM	UCLM	N
Female	3.98	3.92	4.04	477
Male	4.04	3.98	4.09	581
Other	3.95	3.43	4.48	13

Figure 6. Descriptive statistics on a graphic

Another example of plotting output statistics is shown in the survival plot in Figure 7 (Goldstein 2013). This plot is built entirely on the estimates produced with PROC LIFETEST using the PLOTS= option with ATRISK and CB suboptions, rather than based on the raw data itself. The step plot is built with Designer and overlaid with band plots for each stratum. The atrisk table is created in a new cell and displays the corresponding number of patients for each time interval. This survival plot can be customized to be a standard template and then recreated as a graphic with any given outcomes based data set.



Patients at Risk

High	161	114	60	34	11	0
Intermediate	444	367	232	145	54	5
Low	81	72	52	35	10	1

Figure 7. Survival plot with atrisk table

A common graphic used for logistic regression is a forest plot of the odds ratios and their corresponding 95% confidence intervals (Figure 8). This type of plot can also be made with Designer by outputting estimates from PROC LOGISTIC to a data set. The estimates are plotted as a scatter layer that includes error bars to represent the confidence intervals. The estimates are represented numerically in the cells located to the right. The red reference line at 1.0 helps us visualize which odds ratios are significant. These graphics are just examples of what can be done with the ODS Graphics Designer, but possibilities are endless. It is simple to create any plot that you might need with a little playing around and a basic understanding of the various layers. Knowledge of which statistics are appropriate for a graphic is important when building plot layers that require output statistics. Each of these custom graphics can be saved in the Designer format called an SGD file. In addition, the corresponding GTL code can be viewed, saved as a template, and run later as a SAS program.

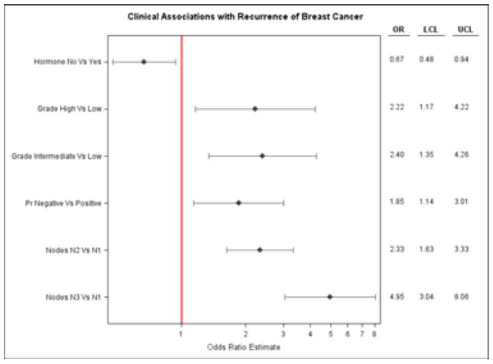
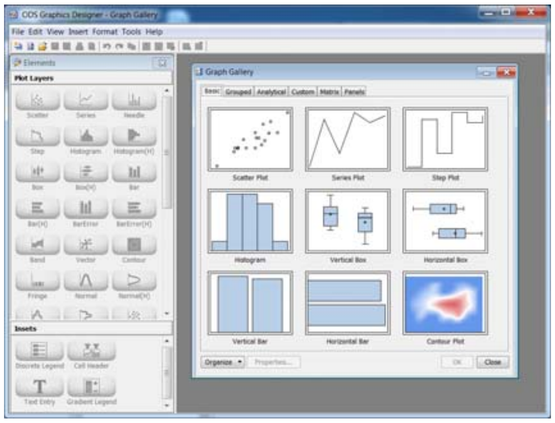


Figure 8. Forest plot for odds ratios

ODS GRAPHICS DESIGNER

Our main focus in this workshop will be using ODS Graphics Designer to create useful and polished graphics in order to generate and then tweak the TEMPLATE code behind them. The Designer tool is a point-and-click interface that makes it very easy to customize graphics. It is important to understand that when building plots with the Designer, a user is not able to rely on default graphics that are produced within a given statistical procedure. Therefore, in order to use this tool properly it is critical to know what graphic is appropriate for your analysis. In addition, some plots may require the creation of a data set with computed estimates before the appropriate graph can be produced.

Before launching the ODS Graphics Designer, we need to load the data sets by submitting LIBNAME statements in SAS so that the Designer has access to the data for the plot. Next, in SAS, go to **Tools** > **ODS Graphics Designer**. It might take a minute to launch the application and if there is a message about connecting to the server, just click retry.



Display 1. ODS Graphics Designer window

The ODS Graphics Designer opens presenting a graph gallery, as shown in Display 1, where custom graphs can be selected, including saved user created plots. The left pane contains various plot layers and inset options that can be used to define a plot. As you create plots, keep in mind that these graphics are assembled in layers and only certain layers can be used together.

CREATING A SIMPLE BAR CHART

Example 1: We begin with a simple bar chart based on the Humor Styles Questionnaire data.

1) Create a basic bar chart.
    a. On the top toolbar go to **File** > **New** > **Blank Graph**.
    b. Drag the **Bar** layer from the Plot Layers box over to the blank graph.
    c. A window named **Assign Data - Bar** will pop-up. Choose the **SASDATA Library** and **HSQ** as the **Data Set**.
    d. On the **Plot Variables** tab choose **Q1** as the **Category** variable. Change the **Statistic** to **Percent**. Click OK.
2) Modify graph properties.
    a. Right click on the plot and choose **Graph Properties**.
    b. On the **General** tab change the **Data Skin** to **Pressed**. Click OK.
3) Add variable groupings.
    a. Right click on the plot and choose **Assign Data**.
    b. On the **Plot Variables** tab choose **GENDER** as the **Group** variable. Verify that the **Statistic** selected is **Percent**.
    c. On the **Panel Variables** tab select **Data Lattice** and choose **GENDER** as the **Column** variable. Click OK.
4) Further modify plot properties.
    a. Right click on the plot and choose **Plot Properties**.
    b. On the **General** tab change the **Wall Fill** to another color. Uncheck the **Outline** box. Click OK.

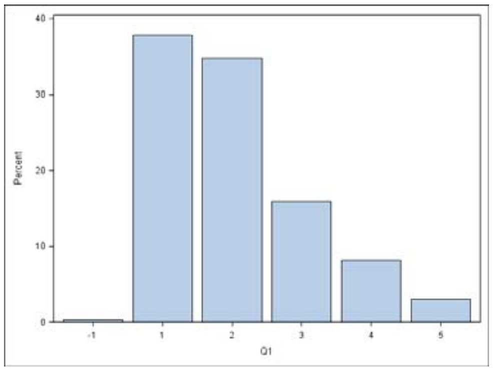


Figure 9. Default bar chart

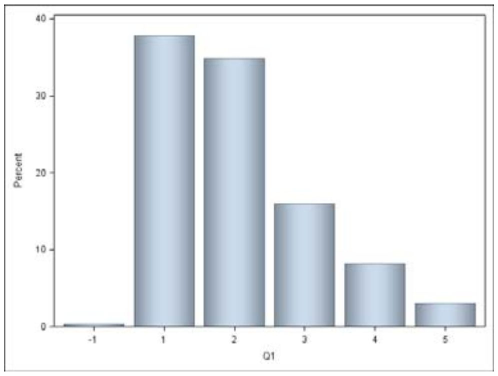


Figure 10. Bar chart with enhancements

5) Modify the axes.
   a. Right click on the X axis and choose **Axis Properties**. On the **General** tab, modify the label as appropriate in the **Label** box. Click OK.
   b. Right click on the Y axis and choose **Axis Properties**. Check the **Grid** box. Click on the **Grid** tab and choose a darker color for the gridlines. Click OK.
6) Insert a title.
   a. On the top toolbar go to **Insert** > **Title**. Double click the **Type in your title…**and modify the text as appropriate.
7) Viewing template code and saving.
   a. From the top toolbar select **View** > **Code**. This is the PROC TEMPLATE code that creates the plot. It can be copied and run directly in the SAS program editor which means that it can also be saved as a program and modified, which we will discuss later.
   b. Click on the graph window. From the top toolbar select **File** > **Save As**. This will allow saving the plot as a SGD file which can be opened in the ODS Graphics Designer later for further editing. Or the graph can be saved as many other file types including PNG, JPEG, and more.
   c. From the top toolbar select **File** > **Save in Graph Gallery**. This will allow saving as a custom made plot in the graph gallery to be used later with ODS Graphics Designer.

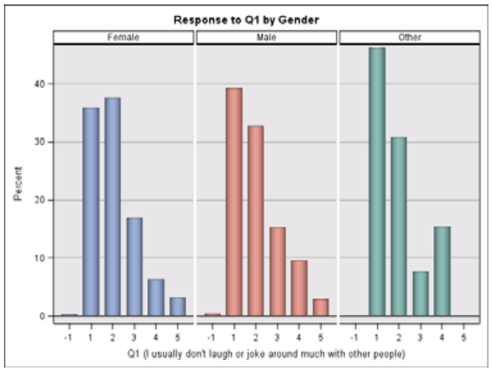


Figure 11. Grouped bar chart with enhancements

WORKING WITH A GALLERY PLOT

Example 2: Next we will build a scatter plot with univariate boxplots using the AP Computer Science data and utilizing a pre-made graphic from the Designer Graph Gallery.

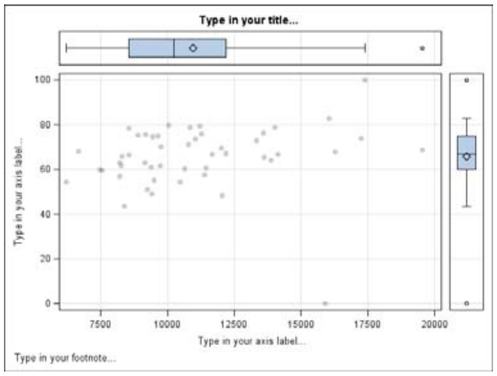


Figure 12. Scatter with boxplots

1) Choose from the Graph Gallery.
   a. On the top toolbar go to **View** > **Graph Gallery**. Click on the **Analytical** tab.
   b. Select the **Data Profile** plot and click OK. The dialog regarding the multi-cell graph tells us that each cell in the graph has its own data source. This means if we change data in one cell it doesn't change in the other cells. Click OK on the dialog.
2) Assign the AP data to the scatter plot.
   a. Right click on the scatter plot and choose **Assign Data**.
   b. Assign **SASDATA** as the **Library** and **APTEST** as the **Data Set**. Select **SPENDING** as the **X** variable and **TOTALPASSEDPCT** as the **Y** variable. Click OK.
3) Modify boxplots.
   a. In the top cell of the graphic, right click on the histogram and select **Remove Plot 'histogram'**.
   b. From the **Plot Layers** drag the **Box (H)** layer to the cell.
   c. Assign **SASDATA** as the **Library** and **APTEST** as the **Data Set**. Select **SPENDING** as the **Analysis** variable. Click OK.
   d. Remove the histogram from the right cell on the graphic by right clicking on the histogram and selecting **Remove Plot 'histogram (H)'**.
   e. Drag a **Box** layer and assign **SASDATA** as the **Library**, **APTEST** as the **Data Set**, and select **TOTALPASSEDPCT** as the **Analysis** variable. Click OK.
4) Add a fitted line to the scatter plot.
   a. From the **Plot Layers** drag a **Regression** layer to the scatter plot.
   b. Make sure that **Fit an existing plot** is checked. This will ensure that the data sources are the same for the fitted line and scatter plots.
   c. Check **CLM** for the **Model Band**. Click OK.

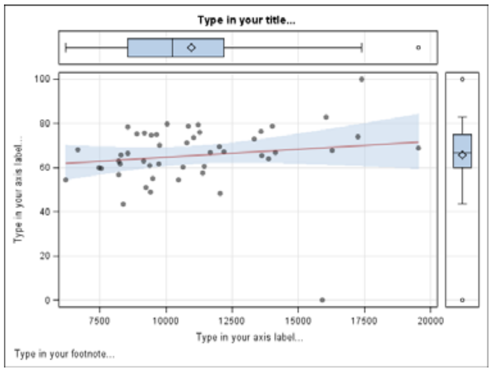


Figure 13. Adding regression information

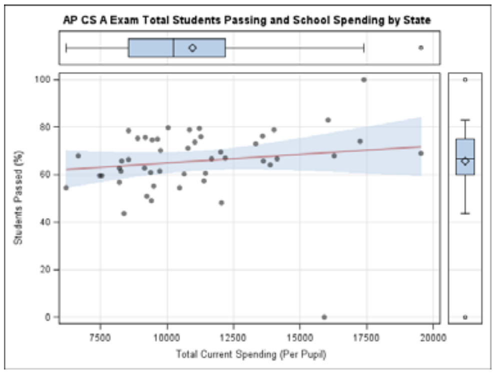


Figure 14. Enhanced bivariate graphic

5) Adjust the transparency of the scatter plot and bands.
   a. Right click on the scatter plot and choose **Plot Properties**. On the **Plots** tab set the **Plot** dropdown to **modelband**. Modify the color and/or transparency of the modelband.
   b. Change the **Plot** dropdown to **scatter**. Modify the transparency of the scatter plot.
   c. Change the **Plot** dropdown to **regression**. Modify the color and/or transparency of the regression line. Click OK.
6) Titles, axis labels and footnotes.
   a. Double click on the title and modify text as desired.
   b. Double click on the Y axis label modify the Y axis label as desired.
   c. Alternatively, right click on the X axis and choose **Axis Properties**. On the **General** tab modify the X axis label and attributes as desired. Click OK.
   d. Right click on the footnote and choose **Remove Footnote**.

CREATING A MULTI-CELL PLOT

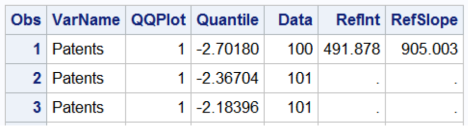
Example 3: Next we will build our own multi-cell plot based on the U.S. patents data for counties with at least 100 patents during the year. This plot will also incorporate output from a procedure to create a cell for a normal probability plot. In addition, we will explore the shared variable graph feature of the Designer.

1) Create an output data set with the statistics required to make a normal probability plot.
   a. In the SAS program editor we run a UNIVARIATE procedure with a QQPLOT statement to generate output for a normal probability plot. Adding ODS TRACE to the procedure will tell us (in the log) the name of the output object that can be captured with ODS OUTPUT.

```
ODS TRACE ON;
PROC UNIVARIATE DATA = sasdata.uscounties NOPRINT;
  ODS OUTPUT QQPLOT = MYQQPLOT;
  QQPLOT patents / NORMAL(MU = est SIGMA = est);
RUN;
ODS TRACE OFF;
```

   b. Reviewing the results of a PROC PRINT we can see that the output data set contains one observation for each of the original data points plus the corresponding quantile if this data had come from a normal distribution. In addition, there is a single record for the first observation that captures the intercept and slope for a reference line for a distribution where mu and sigma are estimated by the sample mean and standard deviation. All of these variables will be used to create our graphic that includes a normal probability plot.

```
PROC PRINT DATA = MYQQPLOT;
RUN;
```



Obs	VarName	QQPlot	Quantile	Data	RefInt	RefSlope
1	Patents	1	-2.70180	100	491.878	905.003
2	Patents	1	-2.36704	101	.	.
3	Patents	1	-2.18396	101	.	.

Output 5. Output data set from UNIVARIATE

2) Create the shared variable plot.
   a. In ODS Designer, go to **File** > **New** > **Blank Shared Variable Graph**.
   b. On the **Assign Data** box Select **WORK** for the **Library** and **MYQQPLOT** for the **Data Set**.
   c. Click on the **Shared Variables** tab. Select **DATA** as the variable for **V1** and set the **Type** to **Numeric**.
   d. Click the add a variable icon Select **QUANTILE** as the variable for **V2** and set the **Type** to **Numeric**. Click OK. In this shared variable plot we will only have access to the variables in the data set that we specified as shared.
3) Add a horizontal histogram.
   a. Drag a **Histogram (H)** layer to the plot.
   b. On the **Plot Variables** tab assign **V1 (DATA)** to the **Analysis** variable. Click OK.
   c. Right click on the histogram and choose **Plot Properties**. On the **Plots** tab check **Prefer Binned Axis**. Click OK.
4) Add a boxplot in another cell.
   a. Right click on the histogram and select **Add a Column**.
   b. Drag a **Box** layer to the empty second column.

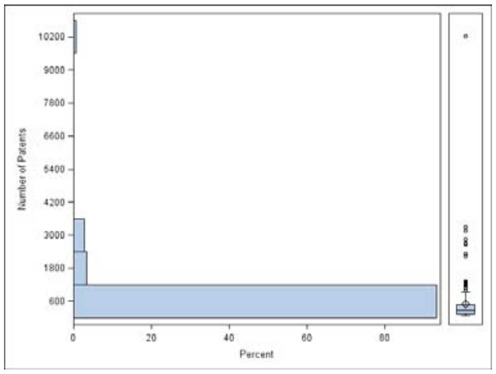


Figure 15. Default histogram and boxplot

   c. On the **Plot Variables** tab select **V1 (DATA)** for the **Analysis** variable. Click OK.
   d. Right click on the <u>Y axis of the boxplot</u> and select **Common Row Axis**.
   e. Right click on the Y axis of the histogram and select **Axis Properties**. Change the **Label** to **Number of Patents**. Click OK.
5) Add a normal plot in another cell.
   a. Right click on the graph and select **Add a Row**.
   b. Drag a Scatter layer to the empty cell below the histogram.
   c. On the **Plot Variables** tab select **V2 (QUANTILE)** for the **X** variable and **V1 (DATA)** for the **Y** variable. Click OK.
   d. Drag a **Line** layer to the scatter plot cell.
   e. Review the numbers in the QQPLOT output for the y intercept and slope values. Type **0** for **X**, **491.878** for **Y**, and **905.003** for **Slope**. Click OK.
   f. Position the mouse between the histogram and quantile plot. Resize the graph so that the normal plot is slightly smaller.
6) Enhance the features of the normal plot.
   a. Right click the normal plot and select **Plot Properties**.
   b. On the **Plots** tab change the **Plot** dropdown to line. Change the color of the line.
   c. Click on the **Axes** tab. Change the X axis **Label** to **Normal Quantile**. Click OK.
   d. Right click on the Y axis of the normal plot and select **Axis Properties**. Uncheck **Label**, **Value**, **Grid**, and **Tick**. Click OK.
7) Insert a Title.
   a. On the top toolbar go to **Insert** > **Title**.
   b. Add an appropriate title for the graphic.
8) Change a shared variable.
   a. To understand the usage of shared variables right click on the histogram and go to **Assign Data**.
   b. On the **Shared Variables** tab change **V1** to **Quantile**. Click OK. Notice that each cell in the plot changes accordingly. While this graphic may not be useful for this combination of variables, there is utility in being able to update the entire graphic at once.
   c. On the top toolbar go to **Edit** > **Undo** to go back to the previous version.

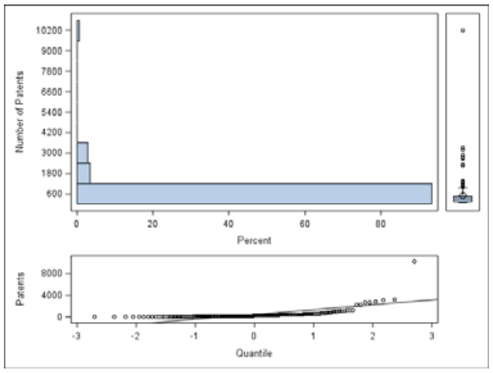


Figure 16. Histogram, boxplot, and normal plot

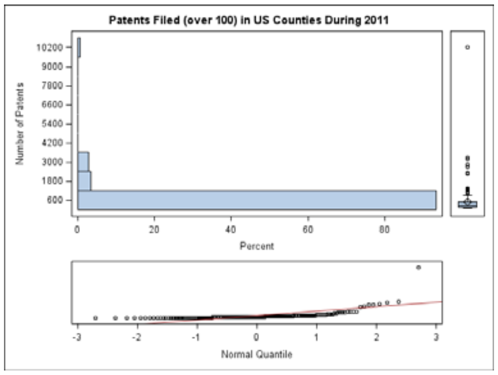


Figure 17. Enhanced patents graphic

CREATING A FOREST PLOT

Example 4: We will now apply what we have learned from working with ODS Graphics Designer to create a publication ready graphic for a statistical analysis. The data from the GBCS study is meant for time to event analysis of survival or breast cancer recurrence; however, we will use this data in a simple logistic regression evaluating predictors of breast cancer recurrence.

1) Create an output data set with the parameter estimates required to make a forest plot.
   a. In the SAS program editor we run a LOGISTIC procedure with an ODS TRACE to tell us the name of the output object (in the log) that captures the odds ratio estimates.

```
PROC FORMAT;
      VALUE rec 0 = 'Censored'
                1 = 'Recurrence';
      VALUE dead 0 = 'Censored'
                 1 = 'Death';
      VALUE ynf  1 = 'Yes'
                 2 = 'No';
      VALUE nodes 1-3 = 'N1'
                  4-9 = 'N2'
                10-HIGH = 'N3';
      VALUE posneg 0 = 'Negative'
               1-HIGH = 'Positive';
      VALUE grd 1 = 'Low'
                2 = 'Intermediate'
```

```
                    3 = 'High';
                 RUN;
   ODS TRACE ON;
   ODS OUTPUT OddsRatios = OddsRatios;
   PROC LOGISTIC DATA = sasdata.gbcs DESCENDING;
     CLASS censrec hormone (REF = 'Yes') grade (REF = 'Low') PR
           nodes (REF = 'N1') / PARAM = REF;
     MODEL censrec = hormone grade PR nodes;
     FORMAT PR ER posneg. nodes nodes. grade grd.;
   RUN;
   ODS TRACE OFF;
```

b. A quick review of the listing for the OddsRatios data set verifies that the parameter estimates and the corresponding 95% confidence intervals are stored in this data set (Output 6).

```
   PROC PRINT DATA = OddsRatios;
   RUN;
```

Obs	Effect	OddsRatioEst	LowerCL	UpperCL
1	hormone No vs Yes	0.671	0.478	0.942
2	grade High vs Low	2.218	1.167	4.216
3	grade Intermediate vs Low	2.395	1.346	4.263
4	PR Negative vs Positive	1.853	1.141	3.010
5	nodes N2 vs N1	2.327	1.628	3.326
6	nodes N3 vs N1	4.949	3.038	8.064

Output 6. Output data set from LOGISTIC

c. We can use a DATA step to clean up the Effect variable in the output data by applying proper case and removing multiple blanks within the category descriptions. We can also format the estimates so that they are ready for presentation on our graphic (Output 7).

```
   DATA OddsRatiosClean; SET OddsRatios;
     Category = COMPBL(PROPCASE(Effect));
     FORMAT oddsratioest lowercl uppercl 4.2;
   RUN;

   PROC PRINT DATA = OddsRatiosClean;
   RUN;
```

Obs	Effect	OddsRatioEst	LowerCL	UpperCL	Category
1	hormone No vs Yes	0.67	0.48	0.94	Hormone No Vs Yes
2	grade High vs Low	2.22	1.17	4.22	Grade High Vs Low
3	grade Intermediate vs Low	2.40	1.35	4.26	Grade Intermediate Vs Low
4	PR Negative vs Positive	1.85	1.14	3.01	Pr Negative Vs Positive
5	nodes N2 vs N1	2.33	1.63	3.33	Nodes N2 Vs N1
6	nodes N3 vs N1	4.95	3.04	8.06	Nodes N3 Vs N1

Output 7. Cleaned version of output data set

2) To create a basic forest plot.
   a. In ODS Designer, go to **File** > **New** > **Blank Graph**.
   b. Drag a **Scatter** layer from the **Plot Layers** box over to the new graph.
   c. Select **WORK** for the **Library** and **ODDSRATIOCLEAN** for the **Data Set**.
   d. On the **Plot Variable** tab, select **ODDSRATIOEST** as the **X** variable and **CATEGORY** as the **Y** variable.
   e. Click on the **More Variables** button. Select **UPPERCL** as **X Error Upper** and **LOWERCL** as **X Error Lower**. Click OK. Click OK again.
3) Enhance the forest plot.
   a. Right click on the scatter plot and choose **Plot Properties**.
   b. On the **Plots** tab change the scatter **Symbol** to another shape. Adjust the transparency.
   c. On the **Axes** tab select **X** from the **Axis** dropdown and choose **Log10** for the **Type**.

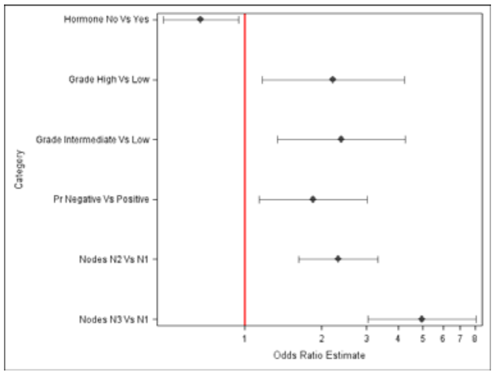


Figure 18. Basic forest plot

   d. On the **Axes** tab select **Y** from the **Axis** dropdown. Uncheck **Label** (Value and Tick should remain checked). On the **General** subtab check the **Reverse** box. Click OK.

4) Add a reference line.
   a. Drag a **Ref(V)** layer to the scatter plot. Enter **1.0** for **X**. Click OK.
   b. Right click on the line and choose **Plot Properties**. On the **Plots** tab choose **vref** from the **Plot** dropdown and change the color and thickness of the reference line. Click OK.
5) Add parameter estimates table
   a. Right click on the plot and choose **Add a Column**. Drag a **StackBlock** layer to the new column cell.
   b. Select **WORK** for the **Library** and **ODDSRATIOCLEAN** for the **Data Set**.
   c. Choose **ODDSRATIOEST** for the **X** variable and also for the **Block** variable. Choose **CATEGORY** for the **Group** variable. Change **Axis** to **X2**. Click OK.
   d. Right click on the table and choose **Plot Properties**. On the **Plots** tab and **Display** subtab uncheck **Fill**, **Outline**, and **Label**.
   e. On the **General** tab uncheck **Fill** and **Outline**.
   f. On the **Axes** tab choose **X2** from the **Axis** dropdown. Uncheck **Value** and **Tick** (leave Label checked). On the **General** subtab type **OR** in the **Label** box (you may need to uncheck and re-check label to make it appear). On the **Label** subtab change the **Font Style** to **Bold**. Click OK.
   g. Position the cursor between the two columns and resize so that the **StackBlock** is narrower.
   h. Repeat steps 5a) (each time right click on the most recent StackBlock to add a new column) to 5g) for the LOWERCL (label as LCL), and again for the UPPERCL (label as UCL) variables.
6) Insert a Title.
   a. On the top toolbar go to **Insert** > **Title**.
   b. Add an appropriate title for the graphic.
7) Align the forest plot and estimates table.
   a. Right click on the Y axis of the forest plot and select **Axis Properties**.
   b. On the **Axes** tab choose **Y** from the **Axis** dropdown. On the **General** subtab enter **0.09** for the **Min Offset**, and **0.09** for the **Max Offset**. Click OK.

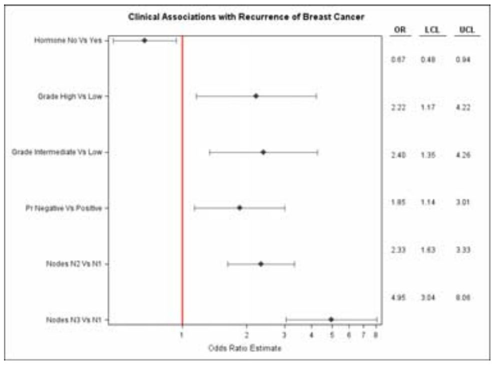


Figure 19. Forest plot with odds ratio table

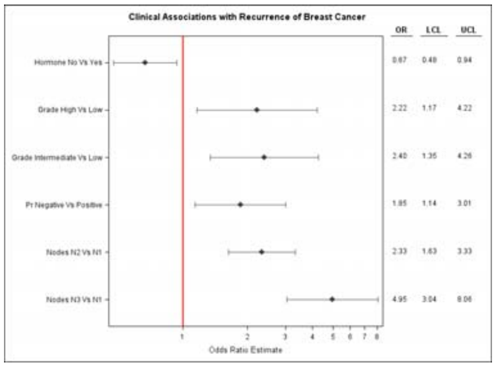


Figure 20. Enhanced forest plot

WORKING WITH GTL AND TEMPLATES

Working with GTL code can be all consuming at first, in fact many books and user group papers have been written about this topic. However, knowing some basic GTL concepts can help take your graphics even further. The purpose of this workshop is not to learn every aspect of the TEMPLATE procedure and GTL. Rather, it is to understand that using ODS Graphics Designer as a launching point to generate GTL code is extremely helpful in creating customized plots. From there, the code can be copied and saved as a SAS program and then modified for later use, even with other variables or data sets.

In order to grasp the basics of GTL syntax it is important to understand how GTL works. Put simply, the TEMPLATE procedure uses GTL syntax to create a template to define a graph. Then the SGRENDER procedure can be used to assign a data set and render the graphic. The main building blocks of PROC TEMPLATE are the DEFINE statement that specifies type and template name, DYNAMIC, MVAR, and NVAR statements that identify variables to be used in the plot, and the begin/end graph block which includes the GTL syntax for the plot. Within the begin/end graph block the layout and structure of the graph is specified with PLOT statements, LAYOUT statements, accessory statements, and additional syntax for expressing things such as conditional logic and functions (Matange 2008). Multi-cell plots will have multiple LAYOUT statements. Basically, the more detailed the graphic, the more complicated the TEMPLATE and GTL code will be in order to generate it.

NAMING STORING TEMPLATES

SAS has templates for every procedure that creates an ODS graphic. Users have the ability to find these templates with ODS trace, and even copy, edit, and save them. Templates can be modified or created, and then compiled and stored in an item store; however, working with the SAS defined templates is tricky. There are many great resources for learning about editing, naming, and storing templates if these actions are desired. In this workshop we will focus on generating templates and GTL code via the Designer, making modifications, and then saving the code as a SAS program that can be called or modified later.

AUTOMATION

Now that we have generated a template to make our graphics we can tweak the code to make them more flexible and reusable. There are several ways to do this: via dynamic variables or macro variables created with PROC TEMPLATE, via shared variables created with ODS Graphics Designer, or via traditional macro variables created in SAS code. Setting up the template code to take advantage of any of these types of variables gives us even more power in creating customized templates for making graphics.

DYNAMIC VARIABLES

The ODS Graphics Designer defines variables used in the plot with the DYNAMIC statement in PROC TEMPLATE. This statement must appear after the DEFINE statement and before the begin/end block. In addition, this statement can define multiple variables. Fortunately the Designer based TEMPLATE code takes care of this for us by creating dynamic variables using an underscore naming convention. These dynamic variables are defined in and used throughout the template; and then set and called in PROC SGRENDER with a DYNAMIC statement. Dynamic variables can be user defined or SAS defined. Taking advantage of these variables is one way to make it easier to automate your graphics.

MACRO VARIABLES

Macro variables can also be utilized with PROC TEMPLATE. They can be created and used within the template by defining them with a MVAR statement (for strings) or NMVAR (for numeric). Similar to the DYNAMIC statement the MVAR and NMVAR statements go after the DEFINE statement and before the begin/end block, they can define multiple variables, and they can be used throughout the template. Assignments for the macro variables can be defined before the PROC SGRENDER code using either %LET statements or as parameters within a %MACRO statement. Both dynamic and macro variables can be used to define data variables, to define labels, and to adjust settings such as transparency or bin width. There are subtle differences in how these variables assign values and how they are treated at runtime. Macro variables have the advantage that they can also refer to SAS automatic macro variables and they can be used with conditional macro logic. One important note about these types of macro variables is that they are not referenced with ampersand (&) inside PROC TEMPLATE.

In most situations traditional SAS macro variables are not necessary for TEMPLATE code. They resolve at compile time which may or may not be what is desired. However, SAS macros can be useful for passing certain constraints into SGRENDER or for running graphics in a loop, which may be helpful, for example, when needing to generate multiple graphics for many variables.

SHARED VARIABLES

As previously discussed creating shared variables with a Designer plot allowed for flexibility in being able to update the graphic with a single change of a shared variable specification. As we will see in the following template examples the dynamic variable names that are created in a Designer shared variable graph are even more flexible in that they are not named according to the original data set for which Designer plot was built. This means less coding changes need to be made when we automate the graphic.

AUTOMATING THE BAR CHART

Let's revisit our first example with the grouped bar chart graphic for the Humor Styles Questionnaire data. Now we would like to modify the code so that the graph can be flexible with respect to the variables, labels, and title. The code below is the template code that was copied from the Designer. We can see that dynamic variables have already been created and referenced in the template.

```
proc template;
define statgraph Graph;
dynamic _Q1A _GENDER _GENDER2;
dynamic _panelnumber_;
begingraph / dataskin=pressed;
   entrytitle halign=center 'Response to Q1 by Gender';
   layout datalattice columnvar=_GENDER2 / cellwidthmin=1 cellheightmin=1 rowgutter=3
columngutter=3 rowdatarange=unionall row2datarange=unionall columndatarange=unionall
column2datarange=unionall headerlabeldisplay=value columnaxisopts=( label=('Q1 (I
usually don''t laugh or joke around much with other people)') discreteopts=(
tickvaluefitpolicy=splitrotate)) rowaxisopts=( griddisplay=ON
gridattrs=(color=CXA4A5A4 ));
      layout prototype /  wallcolor=CXE8E6E8 walldisplay=(FILL);
         barchart category=_Q1A / group=_GENDER name='bar' stat=pct barwidth=0.85
groupdisplay=Cluster clusterwidth=0.85 grouporder=data;
      endlayout;
   endlayout;
endgraph;
end;
run;

proc sgrender data=SASDATA.HSQ template=Graph;
dynamic _Q1A="Q1" _GENDER="GENDER" _GENDER2="GENDER";
run;
```

❶ To run this plot on any question in the survey we can easily replace the original variable reference of Q1 to another variable, for example Q5, in SGRENDER as shown below. This changes the data for the plot but not the labels or title. An unfortunate side effect of using the Designer to create the TEMPLATE code is that in unshared variable graphs the dynamic variables are named according to the original data, in this case _Q1A and _GENDER (shown in the preceding DYNAMIC statement of the original Designer TEMPLATE code). As a matter of good housekeeping we will also modify the variable names in the dynamic statement of PROC TEMPLATE and their references to allow more flexibility.

❷ To modify the title we can change the ENTRYTITLE statement to use the dynamic variables for the question and gender as shown below.

❸ To modify the x-axis label we can add a new dynamic variable called _Xlabel and refer to it in SGRENDER as shown below.

❹ Last we might wish to limit the data displayed by only eliminating unknown responses (-1) and only including the females and males. This can be accomplished with a simple WHERE statement in the SGRENDER procedure. All the changes to the template and SGRENDER are shown below.

```
proc template;
define statgraph Graph;
dynamic _CatVar _GroupVar _GroupVar2 _Xlabel;  ❶ ❸
dynamic _panelnumber_;
begingraph / dataskin=pressed;
   entrytitle halign=center 'Response to ' _CatVar ' by ' _GroupVar;      ❷
   layout datalattice columnvar=_GroupVar2 / cellwidthmin=1 cellheightmin=1 ❶
rowgutter=3 columngutter=3 rowdatarange=unionall row2datarange=unionall
columndatarange=unionall column2datarange=unionall headerlabeldisplay=value
columnaxisopts=( label=(_Xlabel) discreteopts=( tickvaluefitpolicy=splitrotate))
rowaxisopts=( griddisplay=ON gridattrs=(color=CXA4A5A4 )); ❸
      layout prototype /  wallcolor=CXE8E6E8 walldisplay=(FILL);
         barchart category=_CatVar / group=_GroupVar name='bar' stat=pct barwidth=0.85 ❶
groupdisplay=Cluster clusterwidth=0.85 grouporder=data;
      endlayout;
   endlayout;
endgraph;
end;
run;

proc sgrender data=SASDATA.HSQ template=Graph;
WHERE Q5~=-1 and Gender in('Female','Male'); ❹
dynamic _CatVar="Q5" _GroupVar="GENDER" ❶
        _GroupVar2="GENDER"
        _Xlabel="Q5 (I seem to be a naturally ❸
```

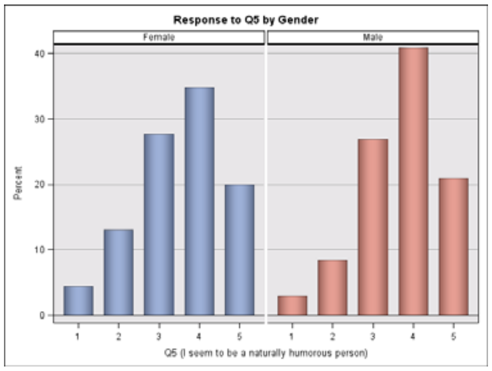


Figure 21. Q5 bar chart with a subset

```
                humorous person)";
run;
```

In order to make our code even more flexible we will incorporate SAS macro programming. The main reason for this is to cut back on the number of times we have to specify the SGRENDER code. In addition, the WHERE statement in SGRENDER above does not work with the dynamic variables from the template; therefore, macro programming will allow us to modify the variables names in that statement as well.

❺ We create the macro called Qplot around the SGRENDER code and define macro variables for the specific question and corresponding label.

❻ We replace the question variable in the WHERE statement with a reference to the macro variable called Q and we make a similar replacement for the dynamic categorical variable that is plotted by _CatVar.

❼ We replace the label value in the dynamic statement for _Xlabel, to be the macro variable called Qlabel.

❽ Last we end and then call the macro setting the desired question variable and appropriate label.

```
%macro Qplot(Q,Qlabel); ❺
proc sgrender data=SASDATA.HSQ template=Graph;
WHERE &Q~=-1 and Gender in('Female','Male'); ❻
dynamic _CatVar="&Q" _GroupVar="GENDER" ❻
        _GroupVar2="GENDER"
        _Xlabel="&Qlabel"; ❼
run;
%mend;

%qplot(Q1,Q1 (I usually don’t laugh or joke around much with other people));   ❽
%qplot(Q2,Q2 (If I am feeling depressed, I can usually cheer myself up with humor));
%qplot(Q3,Q3 (If someone makes a mistake, I will often tease them about it));
%qplot(Q4,Q4 (I let people laugh at me or make fun of me more than I should));
%qplot(Q5,Q5 (I seem to be a naturally humorous person));
```

AUTOMATING THE SCATTER PLOT

Suppose that we would like to create a version of the scatter plot from our second example on AP test data. We've discussed dynamic variables and SAS macro programming previously so these concepts will be inserted in the example below without detailed comment. The Designer template code that was created for our scatter plot is shown below. We notice right away that this template has more sections that correspond to the multi-cell layout of the graphic. However, there are only four dynamic variables that were utilized throughout in order to make this plot.

```
proc template;
define statgraph sgdesign;
dynamic _SPENDING _TOTALPASSEDPCT _SPENDING2 _TOTALPASSEDPCT2;
begingraph;
   entrytitle halign=center 'AP CS A Exam Total Students Passing and School Spending
by State';
   layout lattice / rowdatarange=union columndatarange=union rows=2 columns=2
rowgutter=10 columngutter=10 rowweights=(preferred 0.54) columnweights=(0.74
preferred);
      layout overlay / yaxisopts=( discreteopts=( tickvaluefitpolicy=none));
         boxplot y=_SPENDING2 / name='box_h' groupdisplay=Cluster orient=horizontal
clusterwidth=1.0;
      endlayout;
      layout overlay;
         entry _id='dropsite4' halign=center '(drop a plot here...)' / valign=center;
      endlayout;
      layout overlay;
         modelband 'CLM' / name='modelband' datatransparency=0.5;
         scatterplot x=_SPENDING y=_TOTALPASSEDPCT / name='scatter'
datatransparency=0.5 markerattrs=(symbol=CIRCLEFILLED size=9 );
         regressionplot x=_SPENDING y=_TOTALPASSEDPCT / name='regression'
datatransparency=0.5 clm='CLM' lineattrs=(color=CXA52829 );
```

```
      endlayout;
      layout overlay / xaxisopts=( discreteopts=( tickvaluefitpolicy=splitrotate));
         boxplot y=_TOTALPASSEDPCT2 / name='box' groupdisplay=Cluster
clusterwidth=1.0;
      endlayout;
      rowaxes;
         rowaxis;
         rowaxis / griddisplay=ON label=('Students Passed (%)');
      endrowaxes;
      columnaxes;
         columnaxis / griddisplay=ON label=('Total Current Spending (Per Pupil)');
         columnaxis;
      endcolumnaxes;
   endlayout;
endgraph;
end;
run;

proc sgrender data=SASDATA.APTEST template=sgdesign;
dynamic _SPENDING="SPENDING" _TOTALPASSEDPCT="TOTALPASSEDPCT" _SPENDING2="SPENDING"
_TOTALPASSEDPCT2="TOTALPASSEDPCT";
run;
```

Our first step is to rename the dynamic variables and their references to more generic versions that we will call _X and _Y. We can also create new dynamic variables called _Xlabel, _Ylabel, and _Title to represent the axis labels and group being presented (total, females, Hispanics…) in the title.

```
proc template;
define statgraph sgdesign;
dynamic _X _Y _X2 _Y2 _Xlabel _Ylabel _Title;
begingraph;
   entrytitle halign=center 'AP CS A Exam ' _Title ' Passing and School Spending by
State';
   layout lattice / rowdatarange=union columndatarange=union rows=2 columns=2
rowgutter=10 columngutter=10 rowweights=(preferred 0.54) columnweights=(0.74
preferred);
      layout overlay / yaxisopts=( discreteopts=( tickvaluefitpolicy=none));
         boxplot y=_X2 / name='box_h' groupdisplay=Cluster orient=horizontal
clusterwidth=1.0;
      endlayout;
      layout overlay;
         entry _id='dropsite4' halign=center '(drop a plot here...)' / valign=center;
      endlayout;
      layout overlay;
         modelband 'CLM' / name='modelband' datatransparency=0.5;
         scatterplot x=_X y=_Y / name='scatter' datatransparency=0.5
markerattrs=(symbol=CIRCLEFILLED size=9 );
         regressionplot x=_X y=_Y / name='regression' datatransparency=0.5 clm='CLM'
lineattrs=(color=CXA52829 );
      endlayout;
      layout overlay / xaxisopts=( discreteopts=(
tickvaluefitpolicy=splitrotate));
         boxplot y=_Y2 / name='box' groupdisplay=Cluster
clusterwidth=1.0;
      endlayout;
      rowaxes;
         rowaxis;
         rowaxis / griddisplay=ON label=(_Ylabel);
      endrowaxes;
      columnaxes;
         columnaxis / griddisplay=ON label=(_Xlabel);
         columnaxis;
      endcolumnaxes;
```

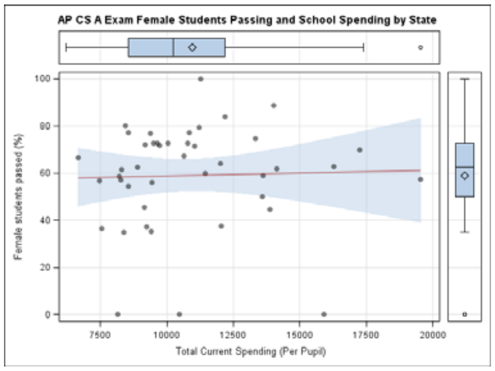


Figure 22. Automated fitted line plot for females

```
   endlayout;
endgraph;
end;
run;

%macro Splot(X,Xlabel,Y,Ylabel,Title);
proc sgrender data=SASDATA.APTEST template=sgdesign;
dynamic _X="&X" _Y="&Y" _X2="&X" _Y2="&Y"
        _Xlabel="&Xlabel"
        _Ylabel="&Ylabel"
        _Title="&Title";
run;
%mend;

%Splot(Spending,Total Current Spending (Per Pupil),TotalPassedPct,
       Students passed (%),Total Students);
%Splot(Spending,Total Current Spending (Per Pupil),FemalePassedPct,
       Female students passed (%),Female Students);
%Splot(Spending,Total Current Spending (Per Pupil),AfricanAmPassedPct,
       African Am students passed (%),African American Students);
%Splot(Spending,Total Current Spending (Per Pupil),HispanicPassedPct,
       Hispanic students passed (%),Hispanic Students);
```

AUTOMATING THE NORMAL PLOT

Suppose that we would like to create a version of the normality graphic for any of the variables in the U.S. patent example. Because we created the template as a shared variable graph the dynamic variables have less data dependent names; however, the underscore naming convention is gone. Ultimately, this means less work for us to transform this into a macro. The resulting Designer code is show below.

```
proc template;
define statgraph Graph4;
dynamic V1 V2;
begingraph;
   entrytitle halign=center 'Patents Filed (over 100) in US Counties During 2011';
   layout lattice / rowdatarange=union columndatarange=data rows=2 columns=2
rowgutter=10 columngutter=10 rowweights=(1.38 0.62) columnweights=(1.0 preferred);
      layout overlay;
         histogram V1 / name='histogram_h' orient=horizontal;
      endlayout;
      layout overlay / xaxisopts=( discreteopts=( tickvaluefitpolicy=splitrotate));
         boxplot y=V1 / name='box' groupdisplay=Cluster clusterwidth=1.0;
      endlayout;
      rowaxes;
         rowaxis / label=('Number of Patents');
         rowaxis / display=none;
      endrowaxes;
      layout overlay / xaxisopts=( label=('Normal Quantile'));
         scatterplot x=V2 y=V1 / name='scatter';
         lineparm x=0.0 y=491.878 slope=905.003 / name='line' extend=true clip=true
lineattrs=(color=CXA52829 );
      endlayout;
      layout overlay;
         entry _id='dropsite4' halign=center '(drop a plot here...)' / valign=center;
      endlayout;
   endlayout;
endgraph;
end;
run;

proc sgrender data=WORK.MYQQPLOT template=Graph4;
dynamic V1="DATA" V2="QUANTILE";
run;
```

❶ We want to avoid the redundancy of re-running the TEMPLATE each time the macro is called, so we place the PROC UNIVARIATE and ODS OUTPUT code with SGRENDER into one macro that appears after PROC TEMPLATE. Within this macro we specify the variable of interest and corresponding label.

❷ We add a dynamic variable for the y-axis label called _label and make sure that it is specified in SGRENDER by the macro variable called varlabel. We do not need to change the variable names being called for V1 and V2 in the DYNAMIC statement in SGRENDER because they refer to the variables DATA and QUANTILE that were created in the ODS OUTPUT data set.

❸ We include an MVAR statement to define macro variables for the y-intercept and slope which will change depending on the variable that we are analyzing. These macro variables are created with CALL SYMPUT in a DATA step to assign the current values of the y intercept and slope to macro variables that can be used in the TEMPLATE code for the reference line in the normal plot. This way the line can be constructed with a more data-driven approach.

❹ We specify the macro variable called var in the QQPLOT statement in PROC UNIVARIATE. The resulting macro code is shown below.

```
proc template; ❶
define statgraph Graph4;
dynamic V1 V2 _label; ❷
mvar _B0 _B1; ❸
begingraph;
   entrytitle halign=center _label ' in US Counties During 2011'; ❷
   layout lattice / rowdatarange=union columndatarange=data rows=2 columns=2
rowgutter=10 columngutter=10 rowweights=(1.38 0.62) columnweights=(1.0 preferred);
      layout overlay;
         histogram V1 / name='histogram_h' orient=horizontal;
      endlayout;
      layout overlay / xaxisopts=( discreteopts=( tickvaluefitpolicy=splitrotate));
         boxplot y=V1 / name='box' groupdisplay=Cluster clusterwidth=1.0;
      endlayout;
      rowaxes;
         rowaxis / label=(_label); ❷
         rowaxis / display=none;
      endrowaxes;
      layout overlay / xaxisopts=( label=('Normal Quantile'));
         scatterplot x=V2 y=V1 / name='scatter';
         lineparm x=0.0 y=_B0 slope=_B1 / name='line' extend=true clip=true ❸
lineattrs=(color=CXA52829 );
      endlayout;
      layout overlay;
         entry _id='dropsite4' halign=center '(drop a plot here...)' / valign=center;
      endlayout;
   endlayout;
endgraph;
end;
run;

%macro nrmplt(var,varlabel); ❶
ODS TRACE ON;
PROC UNIVARIATE DATA = sasdata.uscounties NOPRINT;
  ODS OUTPUT QQPLOT = MYQQPLOT;
  QQPLOT &var / NORMAL(MU = est SIGMA = est); ❹
RUN;
ODS TRACE OFF;

DATA _NULL_; SET MYQQPLOT;     ❸
  IF _N_=1 THEN DO;
    CALL SYMPUT("_B0",PUT(RefInt,7.3));
    CALL SYMPUT("_B1",PUT(RefSlope,7.3));
  END;
```

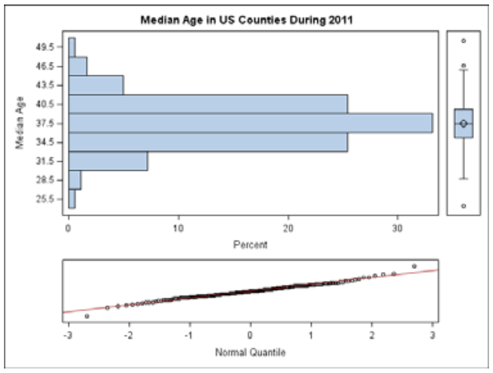


Figure 23. Automated normal probability plot

```
  ELSE STOP;
RUN;

proc sgrender data=WORK.MYQQPLOT template=Graph4;
dynamic V1="DATA" V2="QUANTILE" _label="&varlabel"; ❷
run;
%mend;
%nrmplt(patents,Number of Patents);
%nrmplt(Age_Median,Median Age);
```

We can further modify this template and macro so that the graphic can be run on any continuous variable from any given data set. As an example we will run this normality graphic on any continuous variable from our other examples.

❶ First we add two new macro variables to the macro call, one for the data set and one for a title. Because the title is referenced in the mvar statement in PROC TEMPLATE we will want to refer to it in the macro call with the same name, _title.

❷ The reference to the macro variable called dsn is used to correspond to the input data set for PROC UNIVARIATE.

❸ Create a macro variable reference called title in MVAR statement and then pass it to the entrytitle statement.

❹ Finally the new macro, named nrmplt2, is called with references to new variables and data sets. The appropriate label and title are also passed to the macro.

```
proc template;
define statgraph Graph4;
dynamic V1 V2 _label;
mvar _B0 _B1 _title;   ❸
begingraph;
   entrytitle halign=center _title;  ❸
   layout lattice / rowdatarange=union columndatarange=data rows=2 columns=2
rowgutter=10 columngutter=10 rowweights=(1.38 0.62) columnweights=(1.0 preferred);
      layout overlay;
         histogram V1 / name='histogram_h' orient=horizontal;
      endlayout;
      layout overlay / xaxisopts=( discreteopts=( tickvaluefitpolicy=splitrotate));
         boxplot y=V1 / name='box' groupdisplay=Cluster clusterwidth=1.0;
      endlayout;
      rowaxes;
         rowaxis / label=(_label);
         rowaxis / display=none;
      endrowaxes;
      layout overlay / xaxisopts=( label=('Normal Quantile'));
         scatterplot x=V2 y=V1 / name='scatter';
         lineparm x=0.0 y=_B0 slope=_B1 / name='line' extend=true clip=true
lineattrs=(color=CXA52829 );
      endlayout;
      layout overlay;
         entry _id='dropsite4' halign=center '(drop a
plot here...)' / valign=center;
      endlayout;
   endlayout;
endgraph;
end;
run;

%macro nrmplt2(var,varlabel,dsn,_title); ❶
ODS TRACE ON;
PROC UNIVARIATE DATA = &dsn NOPRINT;  ❷
  ODS OUTPUT QQPLOT = MYQQPLOT;
  QQPLOT &var / NORMAL(MU = est SIGMA = est);
RUN;
```

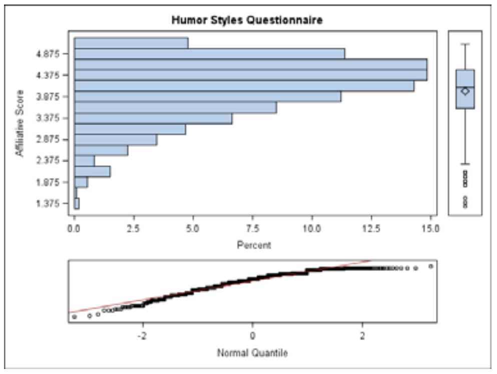


Figure 24. Fully automated normality graphic

```
ODS TRACE OFF;

DATA _NULL_; SET MYQQPLOT;
  IF _N_=1 THEN DO;
    CALL SYMPUT("_B0",PUT(RefInt,7.3));
    CALL SYMPUT("_B1",PUT(RefSlope,7.3));
  END;
  ELSE STOP;
RUN;

proc sgrender data=WORK.MYQQPLOT template=Graph4;
dynamic V1="DATA" V2="QUANTILE" _label="&varlabel";
run;
%mend;

%nrmplt2(affiliative,Affiliative Score,sasdata.hsq,Humor Styles Questionnaire); 4
%nrmplt2(TotalPassedPct,Total Passed (%),sasdata.APTest,AP CS A Exam);
```

AUTOMATING THE FOREST PLOT

We leave the automation of the forest plot to your imagination. Because the cleaned version of the output data is virtually ready to be put directly into the graphic only the title and possible the Y axis offsets would need to be adjusted. If the Y axis labels need further polishing, this can be accomplished easily in the oddsratioclean DATA step.

CONCLUSION

Using the ODS Graphic Designer to create customized graphics provides us with a window into the complicated programming required for PROC TEMPLATE and GTL. The benefit of using the Designer is that the interface is designed so that plots can be created and tweaked with ease. Once the desired graphic is created with Designer the TEMPLATE code can be copied into a SAS program and modified further. The most important requirement for working with Designer is an understanding of basic statistics and of the data required for a fundamentally sound graphic. In addition, a basic knowledge of PROC TEMPLATE coding and the SAS macro language can be leveraged in order to make the Designer graphics automated for use with other data sets and variables. Combining these tools will help you take your customized graphics skills to a new level.

REFERENCES

Ericson B. "Detailed data on pass rates, race and gender for 2013." *Gatech.edu*. June 3, 2014. Available at: http://home.cc.gatech.edu/ice-gt/556.

Goldstein L. and Ottesen R. 2013. "Survival 101 – Just Learning to Survive." Proceedings of the Western Users of SAS Software 2013 Conference. Las Vegas, CA. Available at: http://wuss.org/Proceedings13/113_Paper.pdf.

Hosmer, D.W. and Lemeshow, S. and May, S. (2008). Applied Survival Analysis: Regression Modeling of Time to Event Data: Second Edition, John Wiley and Sons Inc., New York, NY. Data available at: ftp://ftp.wiley.com/public/sci_tech_med/survival.

Infochimps.“NASDAQ Exchange Daily 1970-2010 Open, Close, High, Low and Volume.” 2013. Available at http://www.infochimps.com/datasets/nasdaq-exchange-daily-1970-2010-open-close-high-low-and-volume.

Martin R, Puhlik-Doris P, Larsen G, Gray J, Weir K. 2003. "Individual differences in uses of humor and their relation to psychological well-being: Development of the Humor Styles Questionnaire." *Journal of Research in Personality*, Volume 37, 48-75.

Martin R, et al. "Raw data from online personality tests." 2012. Available at: http://personality-testing.info/_rawdata/HSQ.zip.

Matange S. 2008. "Introduction to the Graph Template Language." Proceedings of the SAS Global Forum 2008 Conference. Cary, NC: SAS Institute. Available at: http://support.sas.com/resources/papers/sgf2008/gtl.pdf.

Matange S. 2009. "ODS Graphics Designer An Interactive Tool for Creating Batchable Graphs." Proceedings of the SAS Global Forum 2009 Conference. Cary, NC: SAS Institute. Available at: https://support.sas.com/resources/papers/proceedings09/331-2009.pdf.

U.S. Census. "Public Elementary-Secondary Education Finance Data." *Census.gov*. 2012. Available at: https://www.census.gov/govs/school/.

U.S. Census. "American Community Survey". 2014. Available at: http://factfinder2.census.gov/.

U.S. Patent and Trademark Office, Electronic Information Products Division - PTMT. "U.S. State Patent Breakout by Regional Component." 2014. Available at: http://www.uspto.gov/web/offices/ac/ido/oeip/taf/countyall/usa_county_gd.htm.

ACKNOWLEDGMENTS

Thank you to Sanjay Matange for personally answering our questions about ODS Graphics Designer.

CONTACT INFORMATION

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

Rebecca Ottesen
City of Hope, Department of Information Sciences, Division of Biostatistics
1500 East Duarte Road
Duarte, CA 91010
Tel: (626) 256-4673
Email: rottesen@coh.org

Leanne Goldstein
City of Hope, Department of Information Sciences, Division of Biostatistics
1500 East Duarte Road
Duarte, CA 91010
Tel: (626) 256-4673
Email: lgoldstein@coh.org