

## From Clicking to Coding: Using ODS Graphics Designer as a Tool to Learn Graph Template Language

Margaret M. Kline, Grand Valley State University, Allendale, MI  
Daniel F. Muzyka, Grand Valley State University, Allendale, MI

### ABSTRACT

ODS Graphics Designer brings simple graphics creation to SAS® platforms 9.2 and later. This application enables any novice user who can navigate an interactive point-and-click menu to generate highly customizable graphical representations. ODS Graphic Designer which functions in conjunction with the suite of SAS products can be invoked to facilitate the creation of Graph Template Language (GTL) through a non-intimidating interface. Not only can this code be edited at a subsequent time but providing novice users with the instant gratification of a striking graphic display could encourage the continued expansion of SAS skills or ease the transition from other software. There is untapped potential in ODS Graphics Designer as an educational tool which exists in its ability to acquaint users with the underlying syntax of GTL. This paper describes how to access and navigate the user interface, provides examples of generated and edited code, and discusses potential uses and limitations to showcase the ability of ODS Graphics Designer as a pedagogical tool for beginner to intermediate programmers.

### INTRODUCTION

For modern statistical analysis, graphics are critical in exploring, analyzing and presenting your data. When SAS ODS Graphics was introduced in version 9.2, its primary intention was to make the production of standard statistical graphics easier. The SAS Output Delivery System creates statistical graphics in three ways: graphs created automatically by invoking ODS Graphics, the SG procedures (SGPLOT, SGSCATTER and SG PANEL) and the Graph Template Language (GTL). Part of the third maintenance release for SAS 9.2 was the launch of a new interactive graphing application, ODS Graphics Designer. The Designer is based on GTL and easily creates graphs through a point-and-click interface while simultaneously generating GTL code that can be saved and edited at a subsequent time.

From our experience, being taught new SAS procedures in an educational setting involves a professor walking through examples of code to create output. You almost always start with simple syntax and modify and add to the code to achieve the desired curated output. Through subsequent iterations, you learn which options produce which effects. Unless taking a course specifically on data visualization, it is likely the only graphics procedures encountered in statistical coursework are the basic SG procedures along with what is produced automatically with ODS Graphics.

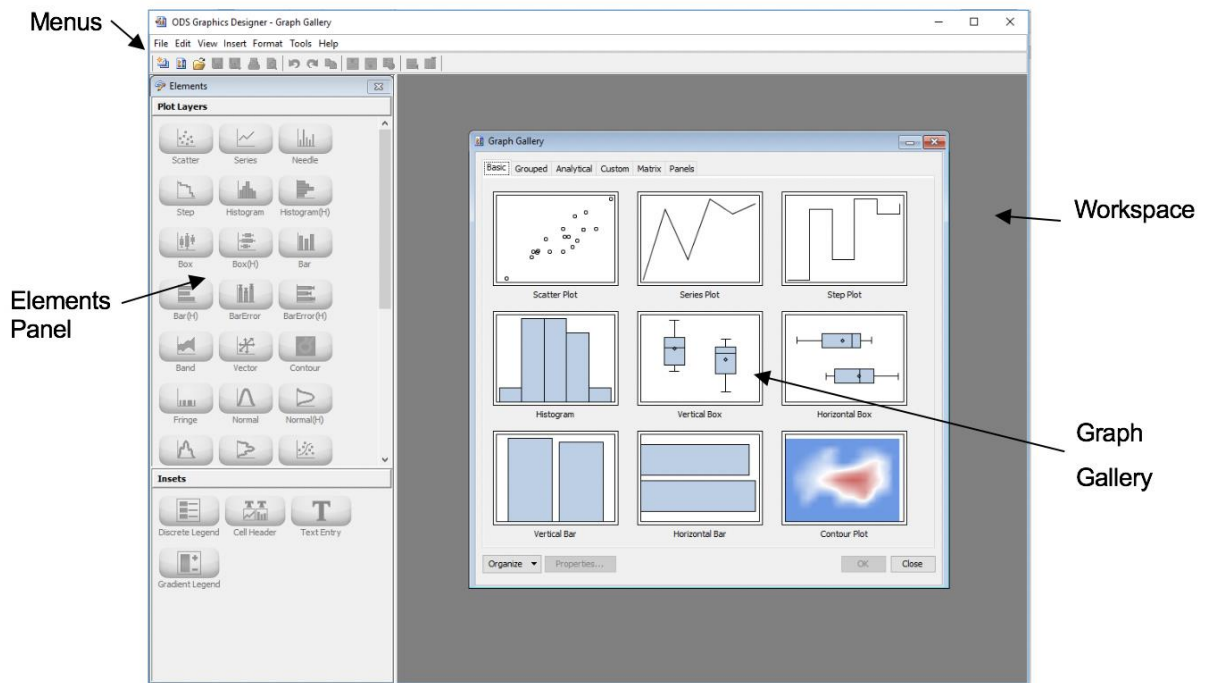
So, what is there to do if you want to self-teach a new procedure? Of course, most programmers are familiar with Google searches and the SAS help documentation. These options can provide a wealth of information but sometimes the documentation can be quite intimidating. That certainly can be the case for something like GTL which allows for a high degree of customization when rendering graphics, at the cost of requiring extensive syntax. Maybe the versatility of GTL has a potential application for your work, but you are unsure if you are up to the task. Maybe you have even attempted to use GTL in the past to your dismay. Do not fret. Sometimes all you need is a little push, or in this case, the ability to push buttons.

Lucky for you, the point of this paper is to demonstrate that one can use the point-and-click ODS Graphics Designer to produce GTL and acquaint themselves with the syntax instead of trying to write the code from scratch. Using SAS Enterprise Guide 7.1, this paper utilizes the SASHELP Iris data set to demonstrate using the ODS Graphics Designer. We lay out our favorite tricks to learn what ODS Graphics Designer is doing behind the scenes, so we can emulate it ourselves.

## ODS GRAPHICS DESIGNER

The ODS Graphics Designer is a Java application that can be accessed in two ways: with a macro call, %sgdesign, or through the drop-down menu, Tasks → Graph → Open ODS Graphics Designer. Even though it is an external program, the Designer uses the SAS software environment to run and the data sets used to create graph templates must be stored as permanent SAS data sets.

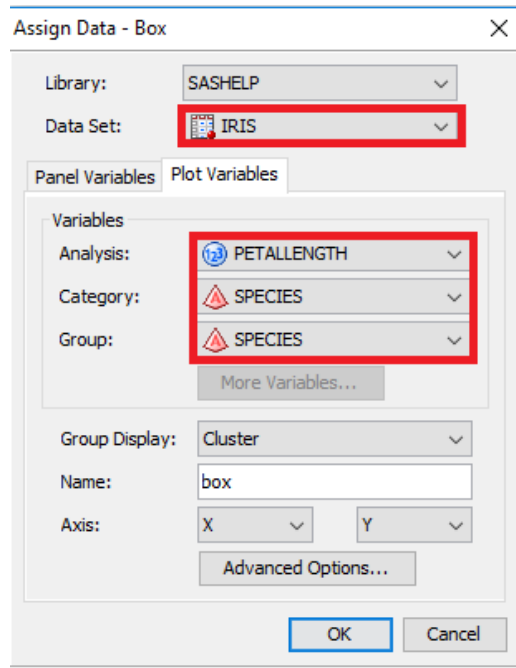
Upon launching the Designer, the initial application window will look as shown in Figure 1 below. The main features of the graphical user interface (GUI) are the menus along the top, the elements panel on the left side, the workspace on the right side and the graph gallery in the workspace.



**Figure 1 Initial ODS Graphics Designer Application Interface**

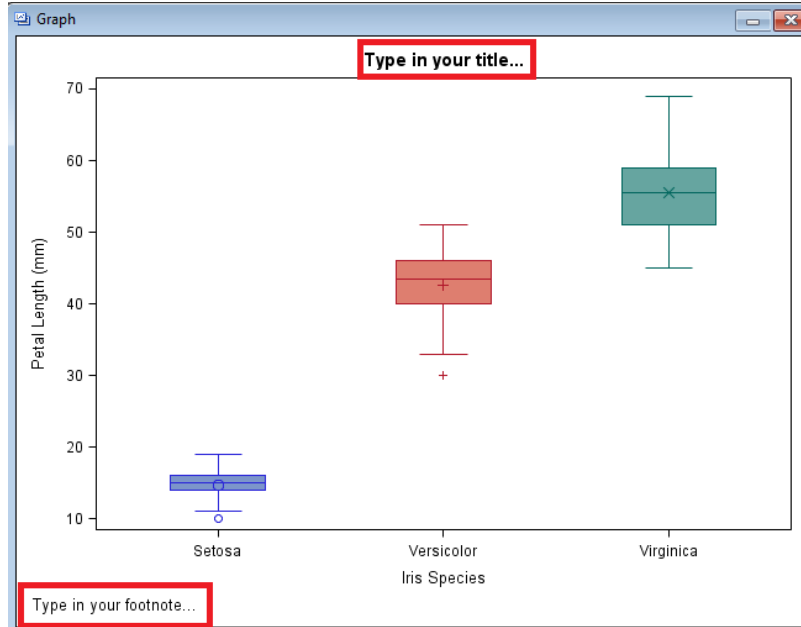
The menus contain features such as saving or opening a graph, inserting objects into the graph like titles and footnotes, and formatting visual properties of the graph. The elements panel is initially inactive but becomes active once a graph is created and allows the user to drag and drop plots and insets to the graph. The workspace can contain the graph gallery, one or more graphs and the GTL code window for the active graph.

The easiest way to start making a graph in the Designer is to pick a premade template from the graph gallery, although you can start with a blank template and drag-and-drop items from the elements panel to start making your graph. The vertical box plot template was chosen and automatically a window pops up, as shown in Figure 2 that has you assign the data set you want to use. We selected the Iris data set and wanted to look at the petal length for each species, so we chose the variable SPECIES for the category and group and PETALLENGTH for the response.



**Figure 2 Data Assignment Window**

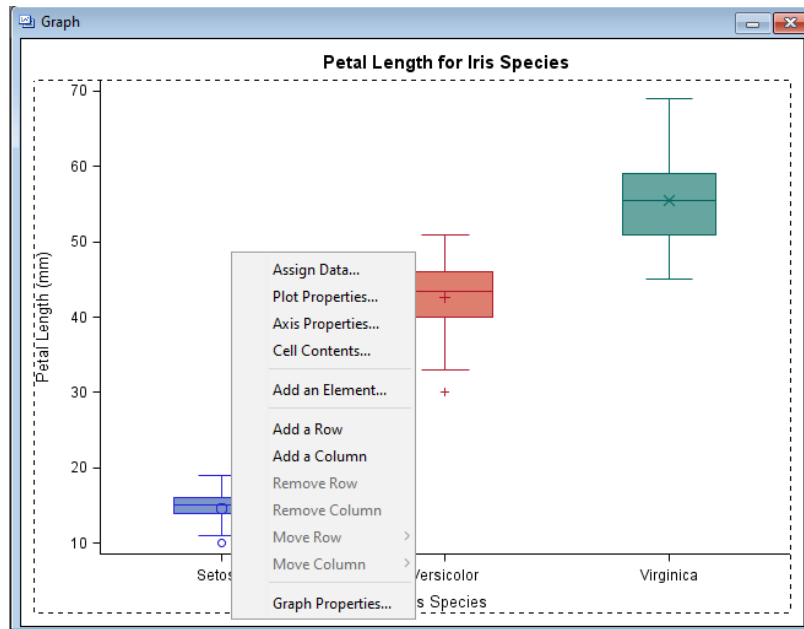
After clicking OK, the plot is updated to display the mean petal length for each species shown in Figure 3 below. Notice the title and footnote on the graph, which you can edit the text by clicking on it or delete them by right clicking and removing them. We updated the title to “Petal Length for Iris Species” and removed the footnote.



**Figure 3 First View of Box Plot**

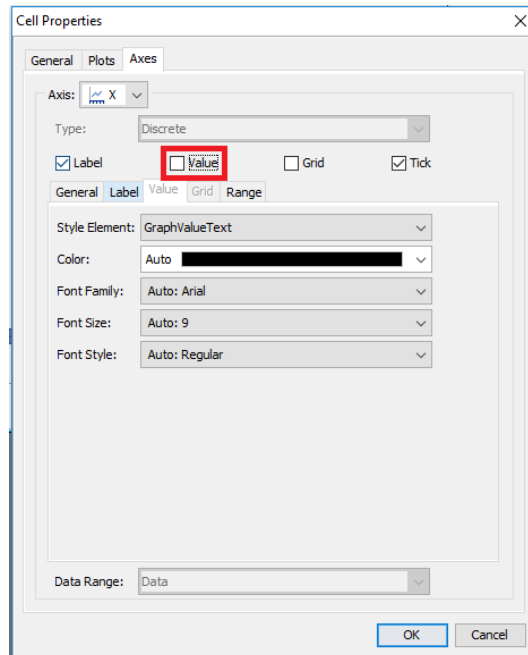
Now, we have a fairly standard box plot that looks as though it was outputted via PROC SGPLOT—in fact, we’ll get to that a little later. Anyways, this is where the fun begins. If you right click on the graph, Figure 4 below, you will find a pop-up menu for different options to customize your graph, which can also be found under the “Format” tab on the menu. One option is “Graph Properties” and it allows you to

change the style, background and image size of your graph, which is where we changed the style of our graph to “statistical.”



**Figure 4 Menu to Start Customizing Your Graph**

“Plot Properties” allows you to change aspects about your box plot, such as setting the box width or changing the symbol for the mean. “Axis Properties” allows to customize your axes with options such as removing the variable labels or reordering the appearance of the values allow the axis. Here we unchecked the variable labels on the x-axis, as shown in Figure 5.



**Figure 5 Axis Properties**

You can drag-and-drop elements into your graph using the Element Panel. Because we removed the values of Iris species from the X-axis, we chose to insert a “Discrete Legend” from the Insets on Elements Panel, shown in Figure 6 below, so we can display which color goes with what specific Iris species.

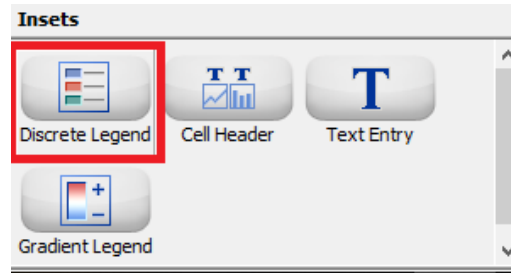


Figure 6 Inset Options

At last, we are satisfied with our final product. This is the time to save the graph. You can do this through the “File” tab and clicking “Save as.” Saving the graph as a .SGD file will allow you to open it in the Designer at a future time if you need to change the format or update the data. You can see the final customizations to our box plots and the saved final product in Figure 7 below:

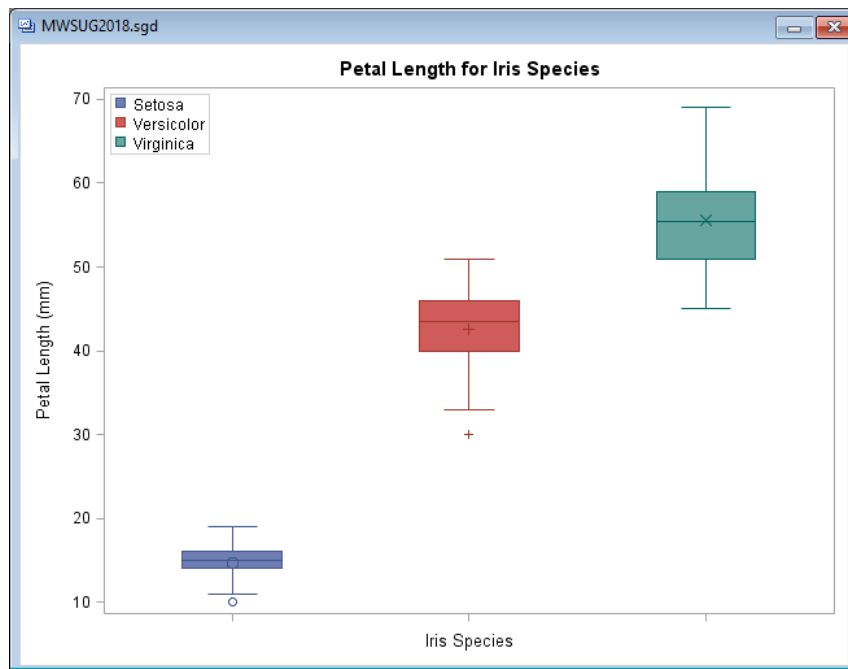
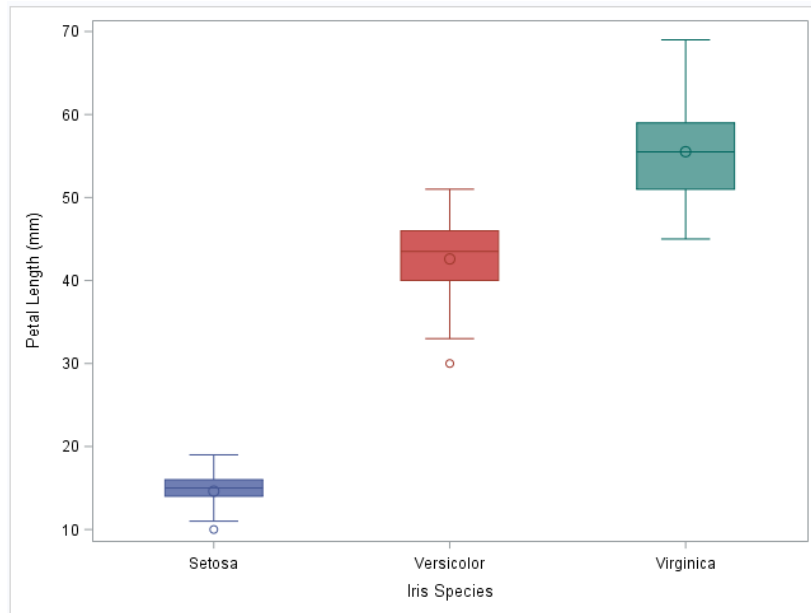


Figure 7 Customized Box Plot

## GRAPH TEMPLATE LANGUAGE

We know we told you that GTL could be a little intimidating, but how intimidating? We mentioned earlier our original graph looked like something PROC SGPLOT would output. Figure 8 shows an almost identical box plot (just minus the title) that was rendered using the following code:

```
proc sgplot data = SASHELP.Iris noautolegend;
    vbox PETALLENGTH / category = species group = species;
run;
```



**Figure 8 Box Plot via PROC SGPLOT and PROC TEMPLATE**

Short and sweet, PROC SGPLOT created the basic box plot above. The equivalent code in GTL to produce that exact Figure 8 box plot is shown below:

```
proc template;
  define statgraph boxplot;
    begingraph;
      layout overlay;
        boxplot x = SPECIES y = PETALLENGTH / group =
          SPECIES;
      endlayout;
    endgraph;
  end;
run;

proc sgrender data = SASHELP.Iris template = boxplot;
run;
```

It is clear to see that GTL required more coding involving additional syntax, but for good cause. This was for a basic box plot, to truly demonstrate the abilities of GTL will take more time. To help you understand why GTL is so powerful, let us take a brief look at the underlying syntax of the GTL.

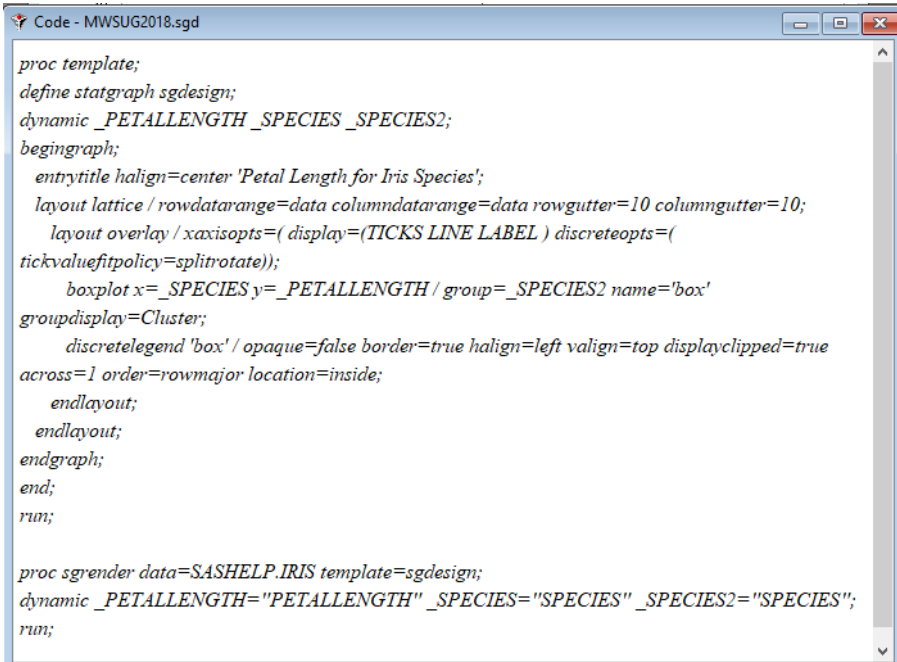
PROC TEMPLATE constructs the graph template using GTL syntax and PROC SGRENDER calls forward that template to be used with the data set in producing the output. The syntax elements of GTL can be categorized into four main groups:

1. Plot statements for specifying the type of graph
2. Layout statements to create paneled or comparative graphs
3. Accessory statements to place titles, footnotes, legends and other text
4. Conditionals, expressions, functions and more

GTL statements are used as building blocks to create the template, where specifying more statements with more options increases the customization of your graph. For a beginner or even intermediate SAS user who probably has graphed typically in SG procedures, the difference in syntax requirements to

produce the same “basic” box plot might be enough to deter them from exploring all the customization possibilities that GTL offers.

The Designer is based on GTL and it allows the user to view the GTL code it generated by going to the “View” tab on the menu and selecting “Code.” A separate window will pop up in your workspace that displays the code for the active graph. Now it’s the moment of truth. We made some modifications to our original graph, like changing the title, adding a legend and removing a value label. Could our Designer outputted GTL be much more extensive than the above GTL code that produced the original graph? Let’s take a look at Figure 9 below which shows the code for our customized box plot.



```
proc template;
define statgraph sgdesign;
dynamic _PETALLENGTH _SPECIES _SPECIES2;
begingraph;
  entrytitle halign=center 'Petal Length for Iris Species';
  layout lattice / rowdatarange=data columndatarange=data rowgutter=10 columngutter=10;
  layout overlay / xaxisopts=( display=(TICKS LINE LABEL) discreteopts=(
tickvaluefitpolicy=splitrotate));
  boxplot x=_SPECIES y=_PETALLENGTH / group=_SPECIES2 name='box'
groupdisplay=Cluster;
  discretelegend 'box' / opaque=false border=true halign=left valign=top displayclipped=true
across=1 order=rowmajor location=inside;
  endlayout;
  endlayout;
endgraph;
end;
run;

proc sgrender data=SASHELP.IRIS template=sgdesign;
dynamic _PETALLENGTH="PETALLENGTH" _SPECIES="SPECIES" _SPECIES2="SPECIES";
run;
```

**Figure 9 Code Viewing Window**

Okay, there seems to be a little more syntax just to move around a few things. How could you ever make the transition from clicking to coding?

## USING THE DESIGNER TO LEARN GTL

We already showed you how to view the code generated by the Designer, but what we did not mention is that you can see the code update one option at a time. When you add an option to the template, such as a fill color, the code will change accordingly in the open code panel. Sometimes this means the code will change slightly, other times new options will be added to the code. Each optional statement has a default state, so if that option is not mentioned, it can default. Just think, if you do not specify a font for your title, is the title not going to appear? Of course, it will. You can see above in Figure 9 on the sixth line that there is no font specified for our title. Look below in Figure 10 to see the change in code when we set the font to Calibri.

```

Code - MWSUG2018.sgd

proc template;
define statgraph sgdesign;
dynamic _PETALLENGTH _SPECIES _SPECIES2;
begingraph / designwidth=311 designheight=480;
entrytitle halign=center 'Petal Length for Iris Species' / textattrs=(family='Calibri');
layout lattice / rowdatarange=data columndatarange=data rowgutter=10 columngutter=10;
layout overlay / xaxisopts=( display=(TICKS LINE LABEL) discreteopts=( tickvalueleftpolicy=splitrotate));
boxplot x=_SPECIES y=_PETALLENGTH / group=_SPECIES2 name='box' groupdisplay=Cluster;
discretelegend 'box' / opaque=false border=true halign=left valign=top displayclipped=true across=1
order=rowmajor location=inside;
endlayout;
endlayout;
endgraph;
end;
run;

proc sgrender data=SASHELP.IRIS template=sgdesign;
dynamic _PETALLENGTH="PETALLENGTH" _SPECIES="SPECIES" _SPECIES2="SPECIES";
run;

```

Figure 10

When clicking through the designer, if you check a box or select an item in a drop-down list other than the default, that option line is added to the code. For some options, if you uncheck the boxes or return the drop-down menus to the default, the option will be written in the code specifying the default value. Look at Figure 11 to see what the code looks like when we set the font to Arial, the default font.

```

Code - MWSUG2018.sgd

proc template;
define statgraph sgdesign;
dynamic _PETALLENGTH _SPECIES _SPECIES2;
begingraph / designwidth=311 designheight=480;
entrytitle halign=center 'Petal Length for Iris Species' / textattrs=(family='Arial');
layout lattice / rowdatarange=data columndatarange=data rowgutter=10 columngutter=10;
layout overlay / xaxisopts=( display=(TICKS LINE LABEL) discreteopts=( tickvalueleftpolicy=splitrotate));
boxplot x=_SPECIES y=_PETALLENGTH / group=_SPECIES2 name='box' groupdisplay=Cluster;
discretelegend 'box' / opaque=false border=true halign=left valign=top displayclipped=true across=1
order=rowmajor location=inside;
endlayout;
endlayout;
endgraph;
end;
run;

proc sgrender data=SASHELP.IRIS template=sgdesign;
dynamic _PETALLENGTH="PETALLENGTH" _SPECIES="

```

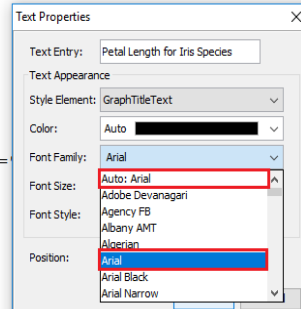


Figure 11

Take notice in Figure 11 that there are two Arial fonts, one labeled “Auto”. If you select an option from a drop-down menu labeled auto, the code will actually be removed. If you check and uncheck a box, the code will remain.

Clicking through the menu options and observing the code in a side by side manner can help you familiarize yourself with where the options go and how they are formatted. If you take the code into SAS and comment well, you can leave those options in the code with instructions of how to edit them when needed.



## CONCLUSION

You might ask, if the Designer is so fast and easy to use, why would users want to learn the GTL behind it? Not all features of GTL are available in the Designer GUI. While using GUI may be more comfortable to novice coders, there is much more potential with direct coding. Therefore, we recommend using the Designer to learn the basic structure of GTL code which allows you to save time in advancing your familiarity with GTL. Once you become comfortable, you can begin to add additional features by coding.

Intermediate SAS users who are familiar with macros can save the template code and make macros to easily replicate the same style graph on a reoccurring basis with new data. Another way direct coding could be more powerful is through efficiency. If a user explores many options in the Designer customization abilities, the code can become bulky with unnecessary code. Learning the underlying syntax will enable users to trim code to just the essentials.

If users want to explore and become familiar with the different types of graphs available, the Designer can help with that as well. If a user selects a plot type from the graph gallery, the Designer will automatically default to a SAS dataset that makes sense for that type of plot. This is helpful in exploring visually what type of plots are available and how customizing them works. For visual learners, this is much more intuitive than reading a list of plot options and trialing them.

GTL syntax can be intimidating at first glance, and because of this many beginner or even intermediate SAS users probably will stick with the basic SG procedures or export their data and make graphics in another program. The Designer presents users with an easy way to compose a graph with their data. The Designer will give the satisfaction of creating a high quality graphic faster than learning GTL through a trial and error process. Since the graphic is accompanied by the code, you can learn the components of GTL while using a dataset you are familiar with. This is how the Designer can take you from clicking to coding.

## REFERENCES

- Holland, Philip. 2010. "Using the ODS Graphics Designer to Create Your Own Templates." *SAS Global Forum 2010*, Seattle, WA. Available at <http://support.sas.com/resources/papers/proceedings10/034-2010.pdf>.
- Kuhfeld, Warren. 2010. "The Graph Template Language and the Statistical Graphics Procedures: An Example-Driven Introduction." *PharmaSUG 2010*. Orlando, FL. Available at: <https://www.lexjansen.com/pharmasug/2010/SAS/SAS-TU-SAS01.pdf>.
- Mantage, Sanjay. 2009. "ODS Graphics Designer An Interactive Tool for Creating Batchable Graphs." *SAS Global Forum 2009*. Washington, DC. Available at: <http://support.sas.com/resources/papers/proceedings09/198-2009.pdf>.
- Muller, Roger. 2016. "Get a Quick Start with SAS® ODS Graphics By Teaching Yourself." *PharmaSUG 2016*. Denver, CO. Available at: <http://support.sas.com/resources/papers/proceedings16/11881-2016.pdf>.

## CONTACT INFORMATION

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

Margaret M. Kline  
Grand Valley State University  
klinem@mail.gvsu.edu

Daniel F. Muzyka  
Grand Valley State University  
muzykad@mail.gvsu.edu

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies.