PharmaSUG 2014 - Paper PO07

SAS Can Automatically Provide GTL Templates for Graphics in Three Ways

David Shen, Li Zhang, Independent Consultant, Chesterbrook, PA

Ben Adeyi, Dapeng Zhang, Shire Pharmaceuticals, Chesterbrook, PA

ABSTRACT

User-defined GTL templates are needed for the complicated statistical graphics which can not be obtained directly from the SG procedures. However, GTL is a relatively new language to many SAS users and it encompasses a large amount of syntax, statements and options. Actually there are three ways SAS can automatically provide the GTL templates. This presentation will show you how we may leverage this SAS feature to obtain the desired GTL templates, and then tailor these templates to create the customized statistical graphics, without writing GTL codes from scratch.

INTRODUCTION

Started with SAS 9.2, ODS Graphics introduces new SAS/GRAPH SG procedures and Graph Template Language (GTL) to generate high-quality graphs. SG procedures (SGPLOT, SGPANEL and SGSCATTER) provide an extensive set of plots and supporting statements to offer a wide variety of stand-alone graphs using a concise syntax. However, they can not cover all of the options to control the properties of the graphs, and completely meet the users' specific requirements. There would be times that you cannot make a customized graph by using the SG procedures only. For a graph that is beyond the scope of the SG procedures, a designed graph template written in GTL syntax should be used. SAS/GRAPH includes GTL syntax for defining sophisticated statistical graphics using the TEMPLATE procedure, while SGRENDER procedure creates the expected plots by executing a user-defined GTL template with a dataset.

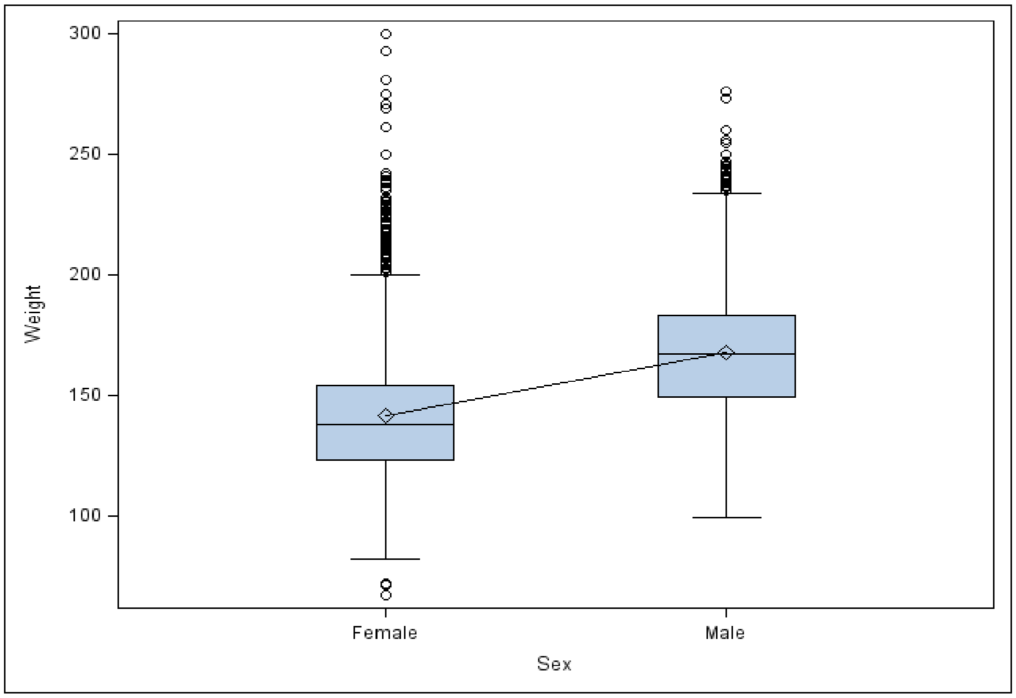
GTL, as a very powerful language to create complex statistical graphics, specifies graph layouts (lattices, overlays), types (scatter plots, histograms), titles, footnotes, insets, colors, symbols, lines, and other graph elements. You may image that each graph is controlled by a template, which is an actual SAS program written in GTL syntax. Therefore, you can create the templates to generate complicated or customized single plots and multi-celled plots with SGRENDER once you have learned GTL. But GTL syntax seems totally different from that of traditional SAS/GRAPH code. Many SAS users are not comfortable with GTL syntax since it requires more advanced knowledge of template language and the ability to write and debug codes.

The time taken to learn GTL and spent on coding is likely to be long. This presentation will show you that SAS can automatically provide GTL sample templates as desired in three ways. All you have to do is isolate the relevant parts that you want to tailor, and update those parts with just a little knowledge of GTL. While ignoring the surrounding complexity, you can edit these templates to create your own highly customized graphs by combining language elements, building panels that contain multiple graphs, managing plot axes and legends, modifying style elements to control appearance characteristics, and using functions, expressions, and conditional processing. These operations leverage the power of GTL, and can save you time and effort currently required in creating the templates from scratch.

APPROACHES

1. TMPLOUT Option

SGPLOT, SGSCATTER and SGPANEL procedures (introduced in SAS 9.2) have an option called TMPLOUT on the procedure statement that writes out the GTL templates for the graphs you create into a file, which can be useful for re-building graphs by SGRENDER. For example, suppose you'd like to create a graph containing a box plot with a line plot which connecting the boxes through the mean points, see the figure below.



If submit the following SGPLOT statements directly:

```
proc sgplot data=sashelp.heart;
  vbox weight / category = sex;
  vline sex   / response = weight stat=mean ;
run;
```

There will be an Error message:

```
ERROR: Attempting to overlay incompatible plot or chart types
```

In this case, we can create each of these plots individually using PROC SGPLOT and use the TMPLOUT option to obtain their GTL templates. Then combine the relevant parts together and use SGRENDER to create the plots as designed.

1). Run SGPLOT separately to sketch out the plots, output the underlying GTL templates;

.

```
proc sgplot data=sashelp.heart tmplout="c:\temp\a1.sas";
  vbox weight / category = sex;
run;

proc sgplot data=sashelp.heart tmplout="c:\temp\a2.sas";
  vline sex  / response = weight stat=mean ;
run;
```

Two templates will then be output as follows:

```
proc template;              **** a1.sas ***;
define statgraph sgplot;
dynamic _ticklist_;
begingraph;
layout overlay / xaxisopts=(type=Discrete discreteOpts=(tickValueList=_ticklist_));
   BoxPlot X=Sex Y=Weight / SortOrder=Internal primary=true LegendLabel="Weight" NAME="VBOX";
   ;
endlayout;
endgraph;
end;
run;
```

```
proc template;             **** a2.sas ***;
define statgraph sgplot;
dynamic _ticklist_;
begingraph;
layout overlay / xaxisopts=(type=Discrete discreteOpts=(tickValueList=_ticklist_));
   SeriesPlot X=Sex Y=_Mean1_Weight_ / primary=true LegendLabel="Weight (Mean)" NAME="VLINE";
   ;
endlayout;
endgraph;
end;
run;
```

2). Customize the graph template to the desired layout. Here we combine two GTL templates selectively by copying the **SeriesPlot** statement from a2 into a1 and delete the irrelevant options, to form a new template called BoxPlotLink.

```
proc template;
define statgraph BoxPlotLink;
dynamic _ticklist_;
begingraph;
layout overlay / xaxisopts=(type=Discrete discreteOpts=(tickValueList=_ticklist_));
   BoxPlot X=Sex Y=Weight /SortOrder=Internal primary=true LegendLabel="Weight" NAME="VBOX";
   SeriesPlot X=Sex Y=_Mean1_Weight_ ;
   ;
endlayout;
endgraph;
end;
run;
```

3). SGRENDER to associate the template with a dataset for graph creation. A new dataset which containing the new summary mean variable of _Mean1_Weight_ should be derived before SGRENDER operation.

```
proc means data=sashelp.heart nway;
  class sex;
  var weight;
  output out=mean1_weight mean=_mean1_weight_;
run;

data heart;
  set sashelp.heart mean1_weight;
run;

proc sgrender data=heart template=boxplotlink;
run;
```

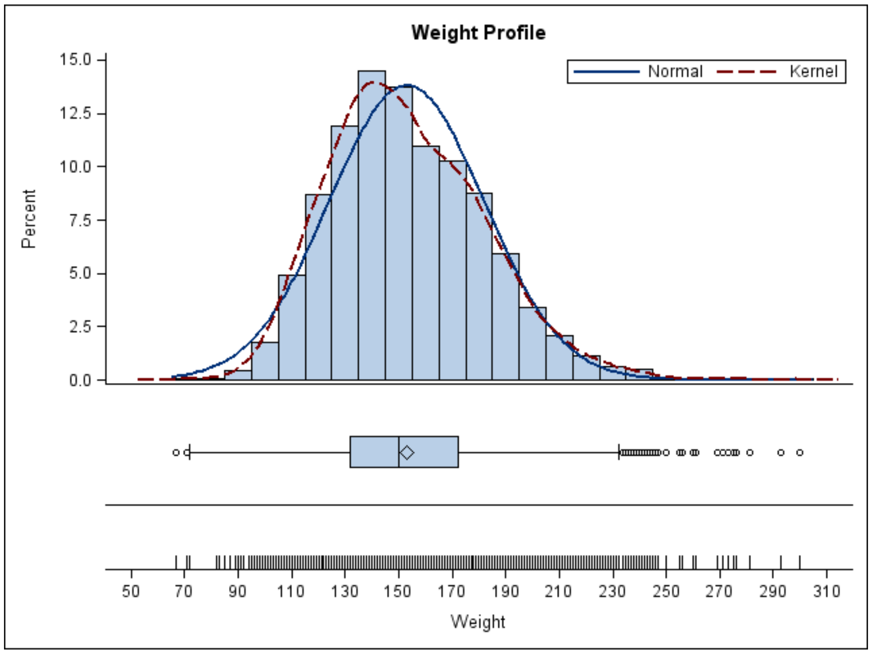
One more example, see data profile plot below. Please note that Fringe plots in PROC SGPLOT are available in SAS 9.4, but not available in SAS 9.2. Fortunately, we may use GTL templates in SAS 9.2 to make fringe plots.

```
      layout overlay / xaxisopts=(label="Weight") WALLDISPLAY=none;
          fringeplot weight;
      endlayout;
```

There are some restrictions for TMPLOUT option:

- The GTL output created by the TMPLOUT option in proc SGPANEL is not always correct, especially when using statistical plots like BOX, HISTOGRAM, etc. SAS 9.3 has discontinued supporting this option for SGPANEL onwards for just this reason.

- Graphs that contain summarized data (for example, bar, line, and dot) do not generate a directly useable template. For summarized data, the procedure sets internal values for one or more variables, and these internal values do not exist in the data set that is used with the procedure.



2. Template Library

SAS has created a large number of default templates to make the graphs for many procedures. The default templates for graphs are stored in SASHELP library. Although these templates often seem large and complicated, you can modify them into the customized templates with a little knowledge of GTL and without understanding the details.

To view the templates in the SAS template library, follow the steps below:

1). Open the Templates window. Two ways to open the Templates window in:

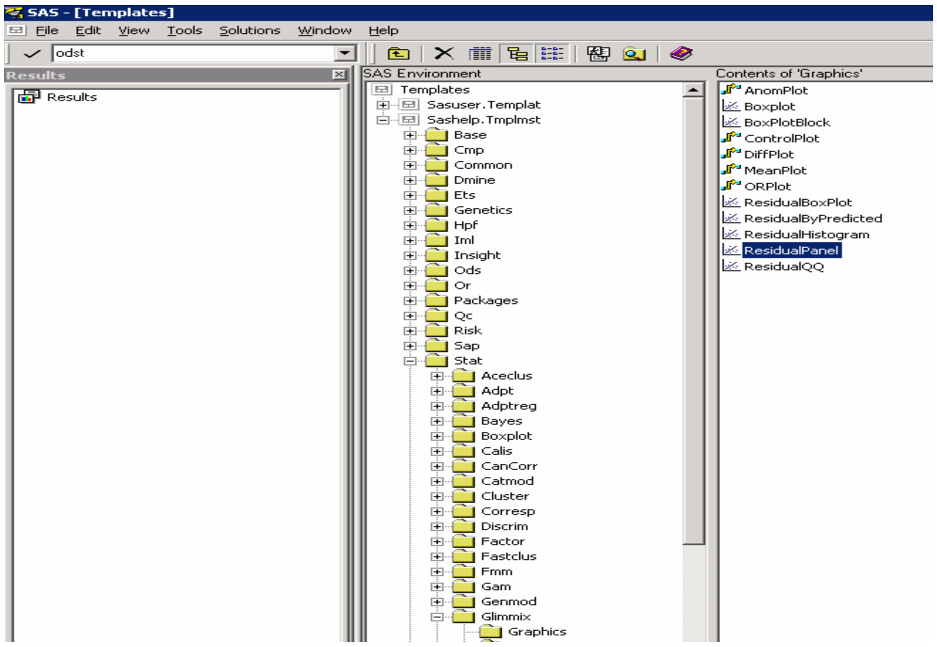
- Enter the **odstemplates** or **odst** command on the SAS command line.
- In the Results window, select the Results folder. Right-click and select Templates to open the Templates window.

The templates that SAS provides are in the item store Sashelp.Tmplmst.

2). Double-click **Sashelp.Tmplmst**, to expand the list of directories where ODS templates are stored.

3). If want to view the statistical graph templates, double-click **Stat** and select the statistical procedure of interest, such as **Glimmix**, then select **Graphics** to list the templates.

4.Double-click the template of interest, such as **ResidulePanel** to view the template definitions.



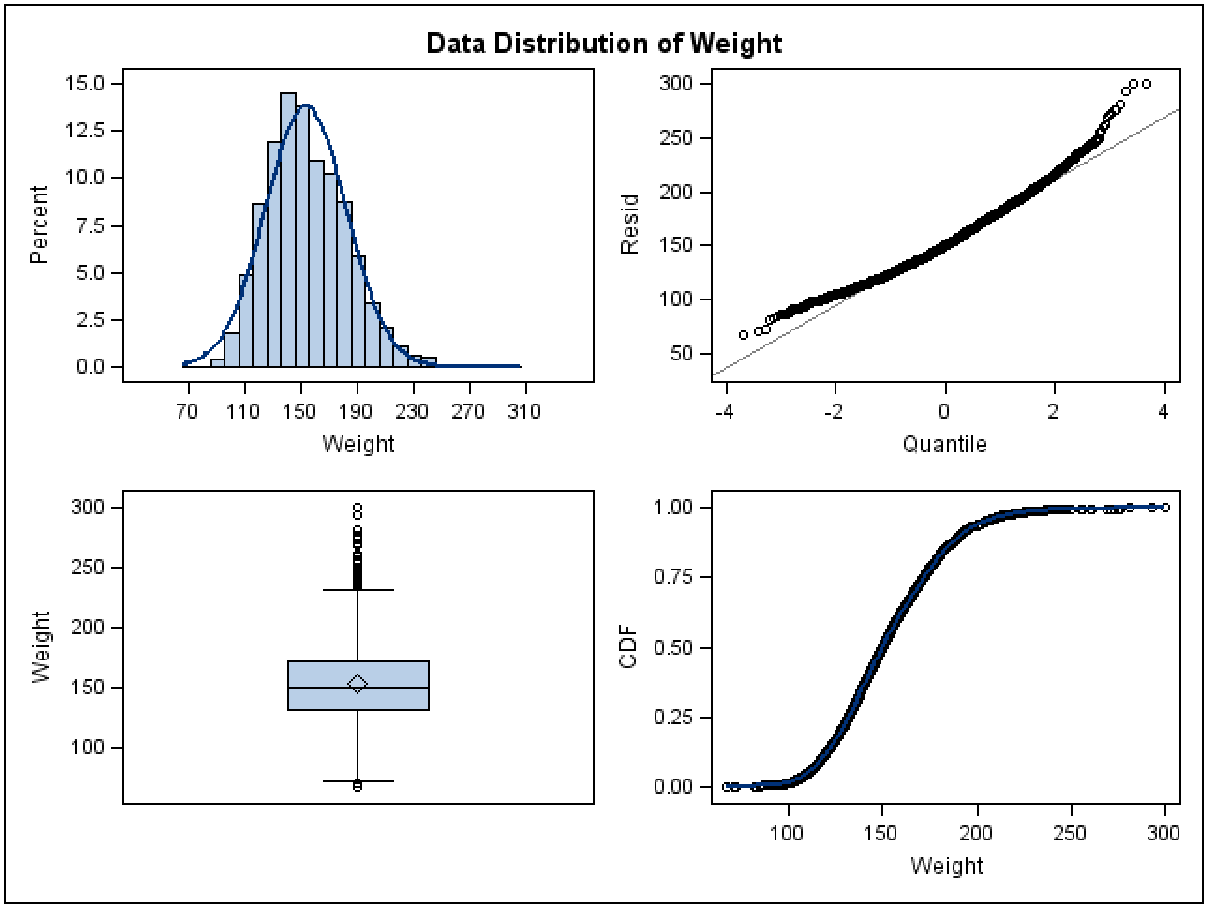
As an example, if we'd like to make a data distribution panel which consists of histogram plot, density plot, Q-Q plot, boxplot and CDF plot, we may copy the codes from **Stat.Glimmix.Graphics.ResidualPanel** template, then make two modifications:

1. Rename template name of Stat.Glimmix.Graphics.ResidualPanel to DistPanel.
2. Remove the first block of codes (which is for Residual vs. Predicted plot), and add a new block of codes (for new CDF plot) at the bottom.

```
proc template;
   define statgraph Stat.Glimmix.Graphics.ResidualPanel;       → DistPanel
      dynamic _TITLE _LABEL _SLABEL _RESIDUAL _PREDICTED;
      BeginGraph;
         entrytitle _TITLE;
         layout lattice / rows=2 columns=2 rowgutter=15 columngutter=10
            shrinkfonts=true;

         layout overlay / xaxisopts=(shortlabel=_SLABEL);
            referenceline y=0;
            scatterplot y=_RESIDUAL x=_PREDICTED / markerattrs=
                               GRAPHDATADEFAULT primary=true rolename=(_tip1=OBS _id1=L1
                               _id2=L2 _id3=L3 _id4=L4 _id5=L5) tip=(y x _tip1 _id1 _id2
                               _id3 _id4 _id5);
         endlayout;                         ;

            layout overlay / xaxisopts=(label=_LABEL) yaxisopts=(label=
               "Percent");
               histogram _RESIDUAL / primary=true;
               densityplot _RESIDUAL / name="Normal" legendlabel="Normal"
                  lineattrs=GRAPHFIT;
            endlayout;

            layout overlay / yaxisopts=(label=_LABEL shortlabel="Resid")
               xaxisopts=(label="Quantile");
               lineparm slope=eval (STDDEV(_RESIDUAL)) y=eval (MEAN(_RESIDUAL)
                  ) x=0 / extend=true lineattrs=GRAPHREFERENCE;
               scatterplot y=eval (SORT(DROPMISSING(_RESIDUAL))) x=eval (
                  PROBIT((NUMERATE(SORT(DROPMISSING(_RESIDUAL))) -0.375)/(0.25
 + N(_RESIDUAL)))) / markerattrs=GRAPHDATADEFAULT primary=true rolename=(s=
                  eval (SORT(DROPMISSING(_RESIDUAL))) nq=eval (
                  PROBIT((NUMERATE(SORT(DROPMISSING(_RESIDUAL))) -0.375)/(0.25
 + N(_RESIDUAL))))) tiplabel=(nq="Quantile" s="Residual") tip=(nq s);
            endlayout;

            layout overlay / yaxisopts=(gridDisplay=auto_off);
               boxplot y=_RESIDUAL / labelfar=on datalabel=OBS;
            endlayout;

         ; layout overlay / yaxisopts=(label="CDF") xaxisopts=(label=_Residual);
                        scatterplot x=eval (SORT(DROPMISSING(_RESIDUAL))) y=eval
         (NUMERATE(SORT(DROPMISSING(_RESIDUAL))) / N(_RESIDUAL))
                        / markerattrs=GRAPHDATADEFAULT primary=true ;
                        loessplot   x=eval (SORT(DROPMISSING(_RESIDUAL))) y=eval
         (NUMERATE(SORT(DROPMISSING(_RESIDUAL))) / N(_RESIDUAL)) ;
           endlayout;

         endlayout;
      EndGraph;
   end;
run;
```

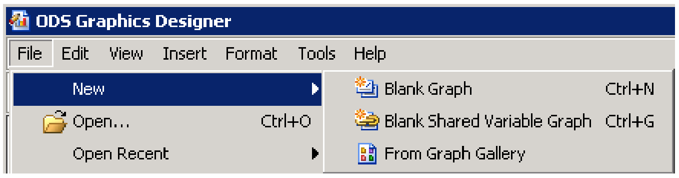
After the TEMPLATE above has been edited and compiled, SGRENDER is used to associate the template with a dataset and relevant variable for graph creation.

```
proc sgrender data=sashelp.heart template=distpanel ;
  dynamic _RESIDUAL="Weight" ;
run;
```



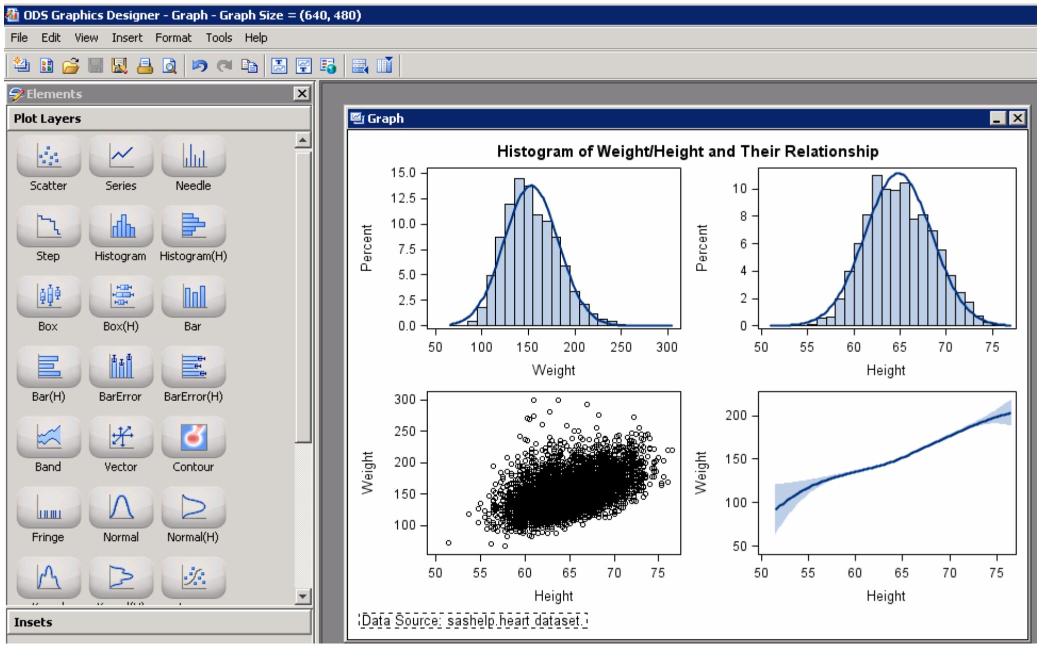
3. ODS Graphics Designer

SAS ODS Graphics Designer provides an interactive GUI of a visual panel and gallery of commonly used graphs for creating statistical graphs. With this application, you can design and create the custom graphs using its point-and-click interaction and drag-and-drop plots from the “Elements Panel” without any programming.
To creating a graph from scratch, you may select Blank Graph or From Graph Gallery.



Let’s start from Blank Graph as an example:

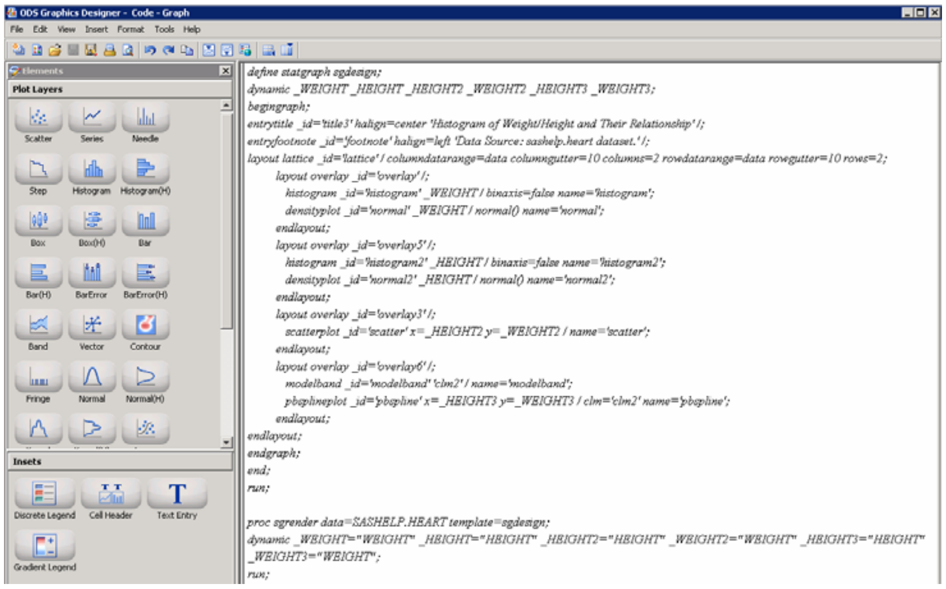
- Select File > New > Blank Graph, a new blank graph is displayed
- Drag and drop plot icons from the Plots Panel, to the Blank Graph, and select the variables needed following the instructions.
- Insert Columns and Rows for multiple cells, title/footnote, and other insets, when needed.


ODS Graphics Designer - Graph - Graph Size = (640, 480)
File Edit View Insert Format Tools Help
Elements
Plot Layers
Scatter
Series
Needle
Step
Histogram
Histogram(H)
Box
Box(H)
Bar
Bar(H)
BarError
BarError(H)
Band
Vector
Contour
Fringe
Normal
Normal(H)
Insets
Graph
Histogram of Weight/Height and Their Relationship
Percent
Weight
Height
Data Source: sashelp.heart dataset.

As you build a graph, the necessary STATGRAPH template is created using the appropriate GTL syntax step by step during the whole process behind the scenes. To view the STATGRAPH template code for this graph, select the graph, and then select **View > Code**.

The code for the graph is displayed in the Code window.

You can copy and paste this template into a SAS Program Editor window, then edit the titles and footnotes, reposition the legends, and customize the visual properties of the plots and axes. Finally, submit the template using the SGRENDER procedure to create the graph.



```
define statgraph sgdesign;
dynamic _WEIGHT _HEIGHT _HEIGHT2 _WEIGHT2 _HEIGHT3 _WEIGHT3;
begingraph;
entrytitle _id='title3' halign=center 'Histogram of Weight/Height and Their Relationship' /;
entryfootnote _id='footnote' halign=left 'Data Source: sashelp.heart dataset.' /;
layout lattice _id='lattice' / columndatarange=data columngutter=10 columns=2 rowdatarange=data rowgutter=10 rows=2;
      layout overlay _id='overlay' /;
        histogram _id='histogram' _WEIGHT / binaxis=false name='histogram';
        densityplot _id='normal' _WEIGHT / normal() name='normal';
      endlayout;
      layout overlay _id='overlay5' /;
        histogram _id='histogram2' _HEIGHT / binaxis=false name='histogram2';
        densityplot _id='normal2' _HEIGHT / normal() name='normal2';
      endlayout;
      layout overlay _id='overlay3' /;
        scatterplot _id='scatter' x=_HEIGHT2 y=_WEIGHT2 / name='scatter';
      endlayout;
      layout overlay _id='overlay6' /;
        modelband _id='modelband' 'clm2' / name='modelband';
        pbsplineplot _id='pbspline' x=_HEIGHT3 y=_WEIGHT3 / clm='clm2' name='pbspline';
      endlayout;
endlayout;
endgraph;
end;
run;

proc sgrender data=SASHELP.HEART template=sgdesign;
dynamic _WEIGHT="WEIGHT" _HEIGHT="HEIGHT" _HEIGHT2="HEIGHT" _WEIGHT2="WEIGHT" _HEIGHT3="HEIGHT"
_WEIGHT3="WEIGHT";
run;
```

CONCLUSION

Customized graphics are generally complicated: multiple graph elements, multiple panels, statistics, legends, titles, footnotes, axis labels, colors, lines, markers, ticks, grids, axes, reference lines, and more. Complicated graphs would definitely need complicated templates using GTL with complex syntax, many statements and options.

Many users who are not familiar with the GTL syntax would prefer to use an interactive tool to create their graphs. The ODS Graphics Designer allows SAS users to create many kinds of graphs, and it is a very good learning tool to generate GTL templates for review and tailor.

This presentation reveals the techniques to enable users to create the specialized graphs needed by requesting SAS to provide GTL templates. Use of these blended approaches can shorten the GTL learning curve and obtains flexibility and simplicity to create the customized graphs with GTL templates.

REFERENCES

SAS/GRAPH® 9.2: ODS Graphics Designer User's Guide
SAS/GRAPH® 9.2: ODS Graphics Designer Help

CONTACT INFORMATION

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

Ben Adeyi, Director of Biostatistics
Shire Pharmaceuticals
735 Chesterbrook Boulevard
Chesterbrook, PA 19087
Work Phone: 484-595-8752
E-mail: badeyi@shire.com
Web: www.shire.com