PharmaSUG 2013 – Paper DG01

Managing Graphic Appearance for Grouped Data in ODS Graphics

Yunzhi Ling, Sanofi, Bridgewater, NJ
Mei Wu, Sanofi, Bridgewater, NJ

ABSTRACT

Analysis of clinical trial data typically involves graphical presentation of results by different treatment groups. Graphically displaying grouped data into desired layout with specific and consistent visual attributes such as colors, marker symbols, and line patterns for each treatment group, can sometimes be challenging. In ODS graphics, such challenges can be effectively overcome by properly leveraging four appearance-control components: attributes options, input dataset, style template, and graph template. This paper provides some general methods and techniques to manage graphic appearance, which is often desired for grouped data in clinical research.

KEYWORDS

ODS Graphics, Graph Template Language (GTL), SG Procedures, Graph template, Style template, Index option, GTL conditional logic, Library and item store.

INTRODUCTION

In clinical research, most data are collected and summarized by different treatment groups. When these treatment grouped data are presented through graphical displays, unique graphical attributes like colors, marker symbols, and line patterns, are often desired for each treatment group. Furthermore, such graphical attributes are expected to be project specific and consistent throughout the entire project.

With the production release of ODS Graphics in SAS 9.2, highly customized graphs can be created. ODS Graphics is a flexible system that generates statistical graphics automatically from a number of SAS products, as well as creating stand-alone graphics with the SG procedures and the Graph Template Language(GTL). In ODS Graphics, desired graphical appearance and attributes are mainly controlled by four key factors: attributes options, input dataset, graph template, and style template. Comparing with traditional SAS/GRAPH, ODS Graphics has a very different technique in managing graphical attributes, especially when dealing with grouped data. For a graph that can not be directly created from SAS/STAT procedures or SG procedures, customized graph template and style template written in GTL are needed to fulfill user-specific requirements.

The Graph Template Language (GTL) is incredibly powerful but also very complex. It contains a large amount of statements and options and is a relatively new language to traditional SAS users. To ease ODS graph creation and customization process, this paper shows an incremental approach, based on a prototype SG procedure, to properly address the above-mentioned challenges and enhance graphic layouts for grouped data. The approach consists of:

- Use input dataset to control graph contents and appearance
- Combine input dataset and attributes options to manage graphic attributes
- Override default style elements with a customized style
- Create a customized graph template
- Utilize index option in graph template to make graphic attributes data independent
- Develop a dynamic template for multiple graphic needs
- Store and share template through ODS PATH statement

ODS GRAPHICS – A TEMPLATE-BASED SYSTEM

SAS/GRAPH produces graphic outputs using two very different systems: a device-based system vs. a template-based system.

The traditional SAS/GRAPH procedures, including GPLOT, GCHART, GMAP, GARBLINE, GCONTOUR and G3D procedures, produce graphic outputs using a device-based system. For device-based graphics, the GOPTIONS statement controls the overall graphical environment. The global statements, such as SYMBOL, PATTERN, AXIS and LEGEND manage graphic appearance and attributes. Within the frame work of these

traditional SAS graph procedures, further customization of graphic outputs can be achieved by annotation facility, a tool that is well established yet can be cumbersome to implement.

ODS Graphics is a template-based system. In ODS Graphics, each graph is created based upon a compiled graph template which is a SAS program written in GTL. Template-based graphs include: 1) graphs created from SAS/STAT, SAS/ETS, and SAS/QC produces when ODS GRAPHICS ON statement is specified; 2) graphs produced from SGPLOT, SGPANEL, SGSCATTER procedures; 3) graphs created from customized graph template written in GTL. The first two methods surface the power of GTL and allow graph creation from a clear and concise syntax similar to conventional SAS procedures, however, they are actually based on behind-the-scene ODS templates supported by SAS.

For template-based graphics, default ODS Graphics settings from SAS Registry control the overall graphical environment. ODS GRAPHICS statement controls the runtime environment of a SAS session with options to specify paper size, threshold for anti-aliasing, image format, file name, etc. For example:

```
ods graphics on/width=6in height=3.5in antialismax=800 imagefmt=PS
imagename='mygraph';
```

In ODS Graphics, graph template and its associated input dataset control the graphical layout, type, contents, titles, footnotes, legends, lines, etc. Style template controls the graphical attributes, such as fonts, font size, markers, marker size, colors, line types and thickness.

Starting from SAS 9.3, new features of annotation and attribute mapping are added, giving users more control of graphic appearance. Annotation facility provides a mechanism for adding shapes, images and annotations to graph output. Attribute mapping provides a mechanism for controlling the visual attributes that are applied to specific grouped data. While these two new features are beyond the scope of this paper, they are certainly helpful and worth further learning.

GRAPH TEMPLATE

The STATGRAPH template, a reserved graph template from Proc TEMPLATE procedure written in GTL, describes the layout and contents of a graph. It can be defined, compiled, and associated with a dataset for graph creation.

After a template is defined, SAS stores the compiled template in a special SAS file called item store under a library. All SAS-supplied production templates are stored in SASHELP library and TMPLMST item store, SASHELP.TMPLMST in short, and it is read only. All user-defined graph and style templates can be stored in TEMPLAT item store under three different libraries, i.e., WORK.TEMPLAT, SASUSER.TEMPLAT, or TEMPLAT item store in a permanent location.

Proc TEMPLATE LIST statement displays all complied templates in specified library and item store in OUTPUT window:

```
proc template;
     list/ store=sasuser.templat;
     list/ store=SASHELP.TMPLMST ;
     list/ store=work.templat;
run;
```

There are mainly three different ways to obtain source code of a compiled template:

1) Use Source statement in Proc TEMPLATE to write source code of specified template into LOG window:

```
Proc template;
     Source Stat.KDE.Graphics.ScatterPlot;
     Source styles.default;
run;
```

2) Use TMPLOUT=FILENAME option that sends graph template code from SGPLOT, SGPANEL, SGSCATTER procedure to a SAS file:

```
proc sgplot data=a tmplout="G:\myplot.sas";
     reg y=bmi x=weight;
run;
```

3) Use Template Browser to review template code, as in the following steps:
   a. Type odst in the command field
   b. Open available item store, e.g., SASHELP.TEMLMST
   c. Click the Styles folder
   d. Double-click the LISTING style

A customized graph template can be created either starting from scratch using Proc TEMPLATE procedure or by modifying any SAS provided graph template into a customized version, and the later proved to be faster and more practical. Section “Enhancement 4: Create a customized graph template” provides some details on how to modify a SAS-provided graph template into a user-specific one.

STYLE TEMPLATE

ODS styles control general appearance and consistency of all graphs and tables. In fact, SAS provides more than fifty ODS styles for use with ODS graphics, and about one third of all GTL syntax addresses graphic appearance. Within ODS graphics, six ODS styles, LISTING, DEFAULT, STATISTICAL, ANALYSIS, JOURNAL and JOURNAL2, are the recommended styles for common use. All examples in this paper use the LISTING style, which is almost identical to the DEFAULT style.

The following SOURCE statements write the DEFAULT and LISTING style definition to LOG window:

```
Proc template;
     source styles.default;
     source styles.listing;
run;
```

Large amount of the DEFAULT and LISTING style definitions are produced from the above step. To save space, only essential code for graphic attributes, called style element, in the DEFAULT style are displayed below. The beginning part of the LISTING style definition is also displayed, showing that it inherits most style elements definition from the DEFAULT style.

```
*** the DEFAULT style definition;
Proc template;
   define style Styles.Default;
   ...
    class GraphDataDefault /
         endcolor = GraphColors('gramp3cend')
         neutralcolor = GraphColors('gramp3cneutral')
         startcolor = GraphColors('gramp3cstart')
         markersize = 7px
         markersymbol = "circle"
         linethickness = 1px
         linestyle = 1
         contrastcolor = GraphColors('gcdata')
         color = GraphColors('gdata');
      class GraphData1 /
         markersymbol = "circle"
         linestyle = 1
         contrastcolor = GraphColors('gcdata1')
         color = GraphColors('gdata1');
      class GraphData2 /
         markersymbol = "plus"
         linestyle = 4
         contrastcolor = GraphColors('gcdata2')
         color = GraphColors('gdata2');
      class GraphData3 /
         markersymbol = "X"
         linestyle = 8
         contrastcolor = GraphColors('gcdata3')
         color = GraphColors('gdata3');
```

```
...

*** the LISTING style definition, using the DEFAULT style as parent style;
proc template;
   define style Styles.Listing;
      parent = styles.default;
      more style statements;
run;
```

Style element GraphDataDefault defines the default marker, marker size, line style, line thickness, marker and line colors, and other color for non-grouped data. In the DEFAULT and LISTING style, the marker size is 7 pixels, the marker is a circle, the line style is 1 (solid), the color is light blue, and the contrast color is black. Colors apply to filled area like bar charts, while contrast colors apply to markers and lines.

Style element GraphData1 – GraphDataN defines graphic attributes for grouped data. When GROUP= option on a plot statement is used, SAS automatically assigns GraphData1 – GraphDataN to each distinct group in the order as they occur in the dataset to ensure different graphic attributes for unique group display. GraphData1 defines marker, line style, color and contrast color for the first group data, GraphData2 defines the attributes for the second group data, and so on, with N up to 12. In the DEFAULT and LISTING style, blue circle and solid line are assigned to the first group, red plus and MedianDash line are set for the second group, and green cross and MedianDashShortDash line are assigned to the third group.

Similar to graph template practice, a user-specific style template based on Proc TEMPLATE procedure can be created, either as a completely new style or as a modification from an existing style. Following SAS approach of defining the LISTING style using the DEFAULT style as parent style, a quick modification of SAS-provided style template into a customized version is explained in section “Enhancement 3: Override default style element with a customized style”.

INPUT DATASET AND TARGET GRAPH

For simplicity and demonstration purpose, all examples in this paper use the briefest code and one dataset, Demographic data ADDM. This dataset contains Weight, Height and Body Mass Index (BMI) for patients from three treatment groups: Placebo, Treatment A and Treatment B, with key variables shown below:

SUBJID	TRTN	TRT	WEIGHT	HEIGHT	BMI
1001/0001	0	Placebo	60.0	160	23.4375
1001/0002	2	Trt B	69.5	165	24.9202
1001/0003	2	Trt B	92.0	172	29.0367
1001/0004	1	Trt A	56.2	177	20.6428
1201/0006	0	Placebo	55.0	161	20.2020
1201/0007	1	Trt A	70.0	179	25.7117
1201/0008	2	Trt B	81.0	162	28.6990
...					

The goal is to create a scatter plot plus a regression line following the identified regression equation, BMI=2.4+0.325*Weight, with colors and symbols of the scatter plot following project convention: black dot for Placebo, blue triangle for Treatment A, and red star for Treatment B.

USE SG PROCEDURES TO CREATE A PROTOTYPE FOR FURTHER IMPROVEMENT

A scatter plot of BMI vs. Weight with regression line can be quickly created using concise code of SG procedure. Two Proc SGPLOT programs are provided below as examples. The first produces a scatter plot of BMI on Y axis and WEIGHT on X axis, along with one linear regression line, as shown in Figure 0.1:

- REG statement has built-in regression features, it create a scatter plot of all data plus a regression line.
- DEGREE= option specifics the degree of the polynomial fit. Default degree=1 (can be omitted as convention) specifies a linear fit, you can also set degree=2 for a quadratic fit and degree=3 for a cubic fit.
- LEGENDLABEL option is used to label legend as ‘Predicted BMI’, to surpass default “Regression” legend label.
- The visual attributes like color, symbol of scatter plot and line pattern and color of regression line follow the LISTING style element GraphDataDefault definition.

The second Proc SGPLOT creates similar graph as shown in Figure 0.2:

- Using GROUP=TRTN option, a scatter plot and three regression lines are created for three treatment groups. The graphic attributes of each treatment group automatically follow the default LISTING style element GraphData1, GraphData2 and GraphData3 definition that SAS provides.
- Due to GROUP= option, LEGENDLABEL is not functioned correctly even though no error message appears in log file. Instead, treatment group values and variable label are displayed for regression lines legend.
- TMPLOUT=FILENAME option sends GLT code of the Proc SGPLOT to a SAS program, which can be modified into a customized graph template.

```
ods rtf style=listing;
proc sgplot data=addm;
     reg y=bmi x=weight/degree=1 legendlabel="Predicted BMI";
run;

proc sgplot data=addm tmplout="G:\temp.sas";
     reg y=bmi x=weight/group=trt legendlabel="Predicted BMI";
run;
ods rtf close;
```

Figure 0.1

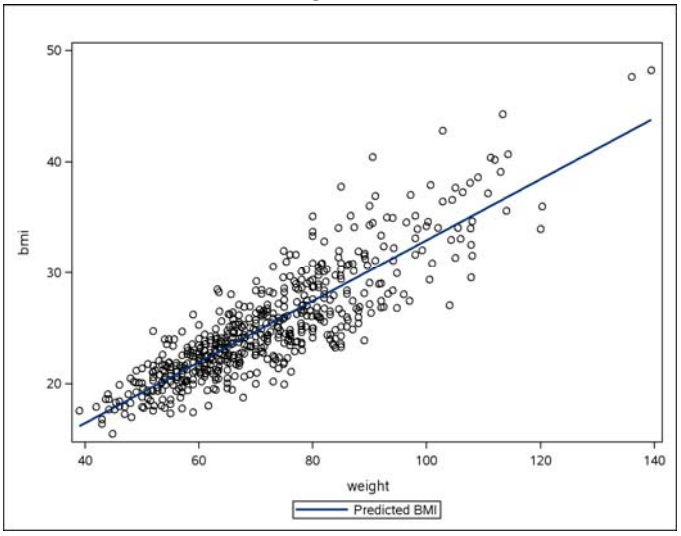
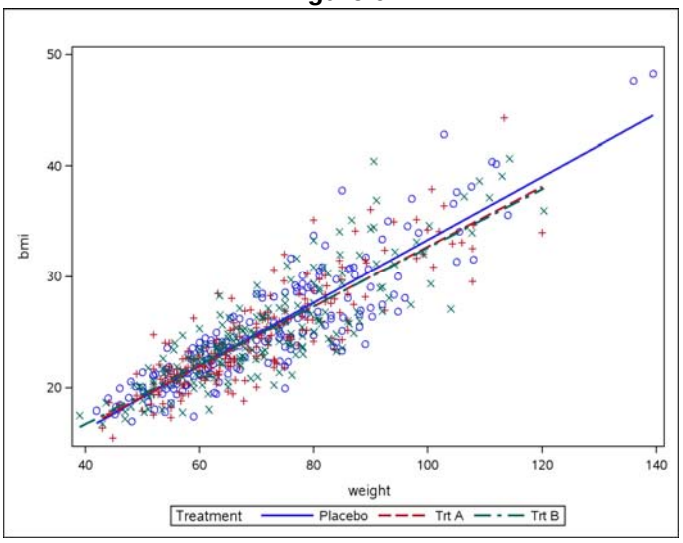


Figure 0.2



Neither Figure 0.1 nor Figure 0.2 satisfies the goal, which is to create a scatter plot with different symbols and colors for 3 treatment groups, and one regression line following the identified regression equation, BMI=2.4+0.325*Weight. This can be achieved through the enhancements shown in the following sections.

ENHANCEMENT 1: USE INPUT DATASET TO CONTROL GRAPH CONTENTS AND APPEARANCE

Through input dataset, we can add data points to draw regression line, as in Program 1:

- Create REG dataset and then append it to original ADDM for regression line points. This REG dataset creates WEIGHT2 and BMI2 coordinates following regression function, BMI=2.4+0.325*Weight, with WEIGHT2 containing integers from 40 to 140, and BMI2 calculated based on regression function. BMI2 variable label is set to blank to conceal legend label.
- Set TRTN and TRT to an existing group. Otherwise, a new symbol or missing symbol will show up in legend.

- Modify Weight and BMI label to control x-axis and y-axis label description.
- In PROC SGPLOT, SCATTER statement creates a scatter plot of BMI vs. WEIGHT, and GROUP=TRT option is used to plot data using TRT as a classification. By default, this option uses the LISTING style elements GraphData1 to GraphData3 for three different treatment groups. Given that the input dataset is sorted by TRT in creasing order, Placebo is the first group to be associated with GraphData1, and has blue circle as the symbol for scatter plot and a solid regression line; Trt A is the second group with GraphData2 assigned, and has red plus for scatter plot and a MedianDash regression line; Trt B is the third group with GraphData3 assigned, and has green cross for scatter plot and a MedianDashShortDash regression line.
- SERIES statement draws a straight line using WEIGHT2 and BMI2 as x, y coordinates. Without GROUP= option, LINEATTRS can be used to set color to blue and thickness to 2 for regression line.
- Figure 1 is created from the program and is displayed below. It is close to the target plot except that colors and symbols of the scatter plot still do not fit project needs.

Program 1

```
data reg;
     label bmi2='00'x;
     trtn=0; trt='Placebo';
     do weight2=40 to 140 by 1;
        bmi2=2.4+0.325*weight2;
        output;
     end;
run;
data addm_reg;
     label weight='Weight in kg'
           bmi='Body Mass Index (kg/m2)';
     set addm reg;
proc sort; by trtn; run;

ods rtf style=listing file="G:\Figure1";
proc sgplot data=addm_reg tmplout;
     scatter x=weight y=bmi /group=trt name="scatter";
     series  x=weight2 y=bmi2 /lineattrs=(color=blue thickness=2) name="series"
                              curvelabel="BMI=2.4+0.325*Weight";
     keylegend "scatter"/ title="Observed BMI:" position=bottomleft noborder;
     keylegend "series"/ title="Predicted BMI:" position=bottomright noborder;
 run;
ods rtf close;
```

Figure 1

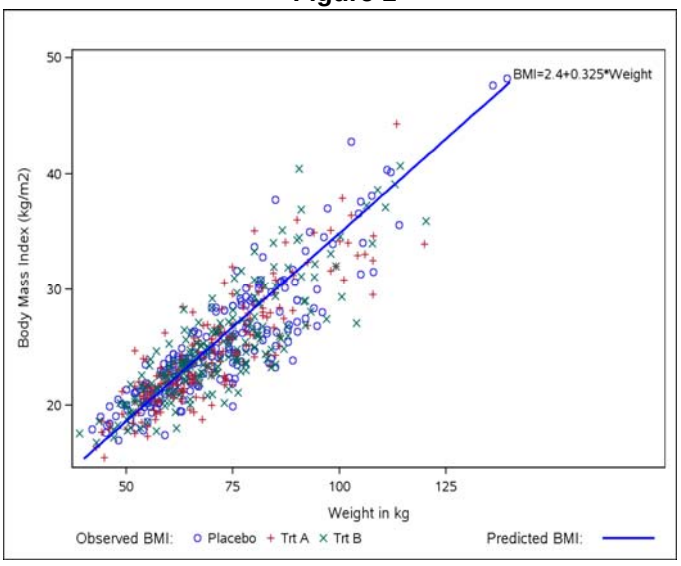
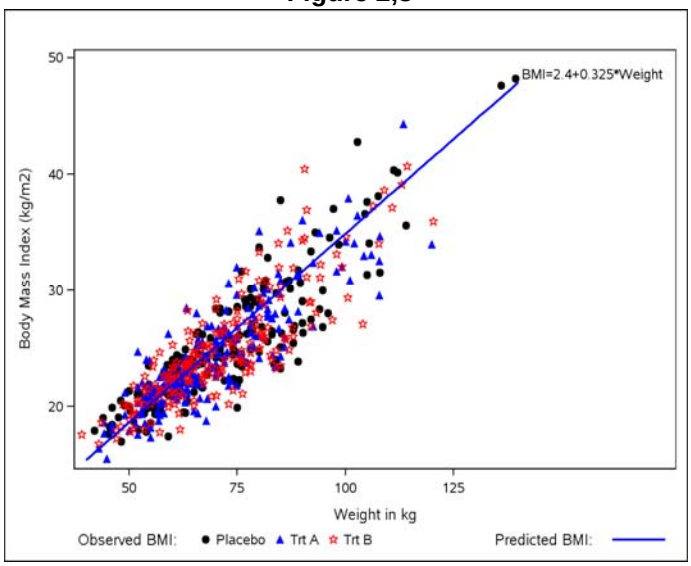


Figure 2,3



ENHANCEMENT 2: COMBINE INPUT DATASET AND ATTRIBUTES OPTIONS TO MANAGE GRAPHIC ATTRIBUTES

In the previous program, due to GROUP=TRT usage in scatter statement, the graphical attributes, such as colors and marker attributes, are predefined through the default LISTING style in SAS registry and can not be user-specified. Instead, they are automatically assigned to graphical style elements GraphData1 to GraphData3 in the underlying GTL syntax.

To specify attributes options of the plot statement to control scatter plot's color and symbol, Program 2 assigns different weight variable for each treatment group so that a separate scatter statement can be used. This approach avoids GROUP= option in order to control graphic attributes directly.

- Three weight variables wt_0, wt_1 and wt_2 are created for different treatment groups. Set wt_0 variable label to 'Weight in kg' to control x-axis label appearance.
- Three scatter statements are used with desired colors and symbols specified in MARKERATTRS= option.
- Assign desired LEGENDLABEL of three scatter plots to control the legend label description.
- Figure 2 created from Program 2 meets the goal, with desired graphic attributes for three treatment groups and one regression line.

Program 2

```
data addm_reg2;
     label wt_0='Weight in kg';
     set addm_reg;
     if trtn=0 then wt_0=weight;
        else if trtn=1 then wt_1=weight;
        else if trtn=2 then wt_2=weight;
run;
proc sort; by trtn; run;

ods rtf style=listing file="G:\Figure2";;
proc sgplot data=addm_reg2;
     scatter x=wt_0 y=bmi /markerattrs=(color=black symbol=circlefilled)
                           name="scatter1" legendlabel="Placebo";
     scatter x=wt_1 y=bmi /markerattrs=(color=blue symbol=TriangleFilled)
                           name="scatter2" legendlabel="Trt A";
     scatter x=wt_2 y=bmi /markerattrs=(color=red symbol=Star) name="scatter3"
                           legendlabel="Trt B";
     series  x=weight2 y=bmi2 / lineattrs=(color=blue thickness=2) name="series"
                              curvelabel="BMI=2.4+0.325*Weight" ;
     keylegend "scatter1" "scatter2" "scatter3"/ title="Observed BMI: "
                                                 position=bottomleft noborder;
     keylegend "series"/   title="Predicted BMI:" position=bottomright noborder;
run;
ods rtf close;
```

ENHANCEMENT 3: OVERRIDE DEFAULT STYLE ELEMENTS WITH A CUSTOMIZED STYLE

The hard-coded approach in previous section uses three scatter statements. Although very straight-forward and controls colors and symbols on the spot, it is obviously not an efficient and flexible approach to plot grouped data, and it does not utilize GROUP= option even thought data are collected by different treatment groups. The program has to be significantly modified when number of classifications or style preference changes. In comparison, using customized style to control graphic attributes for grouped data will be a better approach.

With DEFINE STYLE statement in Proc TEMPLATE procedure, a customized style template can be defined. In stead of creating a completely new style template from scratch, which requires lots of GTL syntax and knowledge, the most efficient approach is to derive a new style template from an existing style which is meticulously designed and provided by SAS. This approach resets the attributes that need to be changed, and any untouched attribute settings will be inherited from the parent style.

Program 3 creates a new style template called PROJ_101, which takes over all style elements and attributes from the LISTING style, and only changes key style elements GraphData1 - GraphData3 following project convention:

- DEFINE STYLE statement defines PROJ_101 as the new style template, PARENT= option sets the

LISTING style as parent style.

- Three Style statements define GraphData1, GraphData2, and GraphData3 style elements for three treatment groups, respectively. CONTRASTCOLOR sets symbol and line color; COLOR defines the color attribute applied to grouped filled areas such as grouped bar charts; MARKERSYMBOL defines the marker symbols; LINESTYLE defines the line patterns.
- Rerun Program 1 and specify STYLE= PROJ_101 instead of STYLE=LISTING to apply the customized style template; output Figure 3 from this rerun is exact the same as Figure 2 shown before.

Program 3

```
Proc template;
     Define style proj_101;
     Parent=styles.listing;
     style GraphData1/ contrastcolor=black color=grey markersymbol='Circlefilled'
                       linestyle=1;
     style GraphData2/ contrastcolor=blue color=lightblue
                      markersymbol='TriangleFilled' linestyle=2;
     style GraphData3/ contrastcolor=red color=lightred markersymbol='Star'
                       linestyle=5;
     end;
Run;

ods rtf style=proj_101 file="G:\Figure3";
    Proc SGPLOT code from Program 1;
ods rtf close;
```

ENHANCEMENT 4: CREATE A CUSTOMIZED GRAPH TEMPLATE

Using input datasets to draw regression line requires additional variables to be added. To avoid adding new variables to input dataset, a customized graph template can be created to reach the same goal. Program 4 shows how to create a customized graph template by modifying the graph template TEMP.SAS obtained from the SGPLOT procedure which creates Figure 0.2. The key changes are:

- Instead of REGRESSIONPLOT statement, using LINEPARM statement to create a straight line based on a point (x=0, y=2.4) and the slope of 0.325. In fact, LINEPARM statement can also generate multiple lines if GROUP= option is used, with all required arguments, slope, x and y specified through numeric variables.
- After template MYPLOT is defined, use Proc SGRENDER to associate the template with original dataset ADDM for Figure 4 creation. Style template PROJ_101 from previous section is also applied to override the LISTING style.

Program 4

```
proc template;
     define statgraph myplot;
       begingraph;
       layout overlay;
          ScatterPlot X=weight Y=bmi / primary=true Group=trt LegendLabel="Body
                                       Mass Index (kg/m2)" NAME="scatter";
          lineparm slope=0.325 x=0 y=2.4/extend=true Lineattrs=( Color=blue
                    Thickness=2) legendlabel="Predicted BMI"
                   curvelabel="BMI=2.4+0.325*Weight" NAME="series";
          DiscreteLegend "scatter" / Location=Outside halign=left valign=bottom
                                     Title="Observed BMI:" Border=false;
          DiscreteLegend "series" / Location=Outside halign=right valign=bottom
                                     Title="Predicted BMI:" Border=false;
        endlayout;
        endgraph;
      end;
run;
ods rtf style=proj_101 file="G:\Figure4";
proc sgrender data=addm template=myplot;
ods rtf close;
```

Figure 4

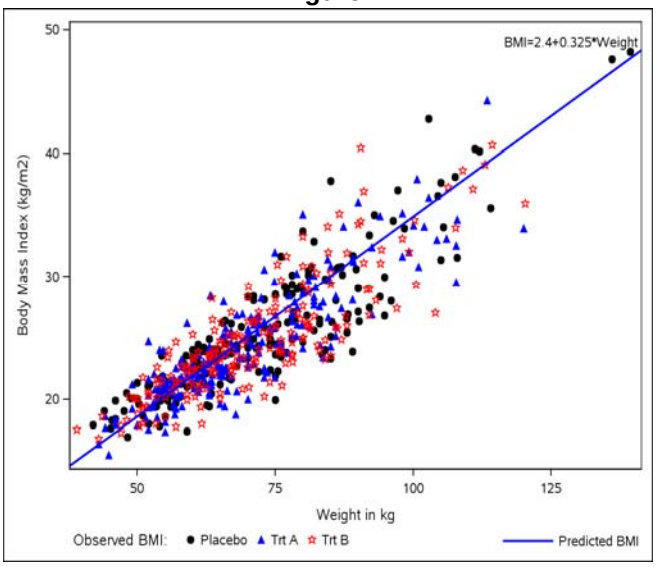
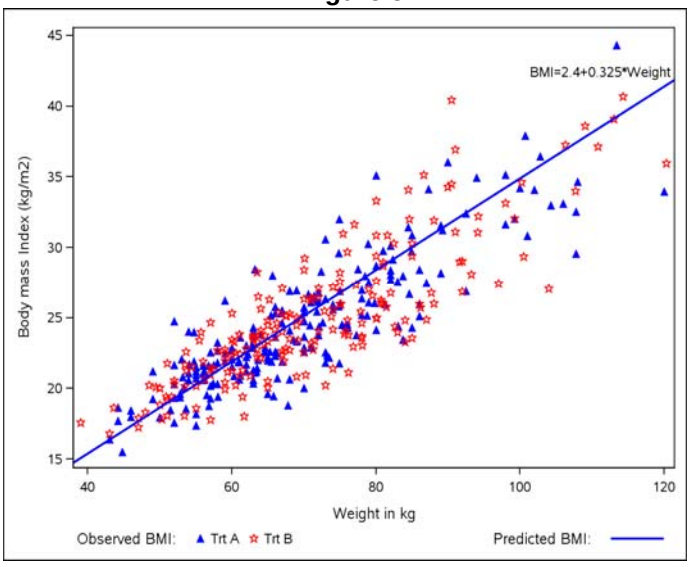


Figure 5



ENHANCEMENT 5: UTILIZE INDEX OPTION IN GRAPH TEMPLATE TO MAKE GRAPHIC ATTRIBUTES DATA INDEPENDENT

The previous approaches are all data driven. When the input dataset is modified, sorted, or filtered, the order of treatment groups and their associations with GraphData1–GraphDataN may change. To guarantee alignment of style elements and classification variables, INDEX= column option can be added to the plot statement in graph template, as shown in program 5.

- Add a numeric index that maps all group values to a positive integer that corresponds to one of the GraphData1-GraphDataN style elements. By doing this, GraphData1 is always associated with the first group Placebo, GraphData2 is associated with the second group Trt A, and GraphData3 with Trt B.
- In graph template, add INDEX=ID option to GROUP=TRT to permanently associate grouped data with GraphData1-GraphData3.
- When use Proc SGRENDER to create Figure 5, Placebo group is purposely excluded from input dataset. Because of INDEX=ID option, graphic attributes for Trt A and Trt B remain unchanged as they are displayed in Figure 4.

Program 5

```
data addm2;
   set addm;
     if trtn=0 then id=1;
     else if trtn=1 then id=2;
     else if trtn=2 then id=3;
run;

proc template;
     define statgraph myplot2;
     begingraph;
       layout overlay;
         ScatterPlot X=weight Y=bmi / primary=true Group=trt index=id
                                      NAME="scatter";
         lineparm slope=0.325 x=0 y=2.4/extend=true Lineattrs=( Color=blue
                  Thickness=2) legendlabel=" " curvelabel="BMI=2.4+0.325*Weight"
                  NAME="SERIES1";
         DiscreteLegend "scatter" / Location=Outside valign=bottom halign=left
                                    title="Observed BMI:" border=no;
```

```
        DiscreteLegend "SERIES1" / Location=Outside valign=bottom halign=right
                                   title="Predicted BMI:" border=no;
      endlayout;
    endgraph;
end;

ods rtf style=proj_101 file="G:\Figure5";
proc sgrender data=addm2(where=(trtn ne 0)) template=myplot2;
run;
ods rtf close;
```

ENHANCEMENT 6: DEVELOP A DYNAMIC TEMPLATE FOR MULTIPLE GRAPH NEEDS

GTL supports conditional logic that allows inclusion or exclusion of one or more GTL statements at runtime, as shown in this basic syntax:

```
IF ( condition )
   GTL statement(s);
ELSE
   GTL statement(s);
ENDIF ;
```

Adding conditional logic and nested IF statements to graph template allows multiple graphs creation from one template. After a dynamic template is created and complied, it can be associated with different datasets and conditions for various plots creation. The following Program 6 creates a generalized scatter plot that conditionally adds regression line and reference lines:

- DYNAMIC statement identifies dynamic variables REF_REG, _x, _y.
- The first IF block specifies an ENTRYTITLE statement that is conditionally executed based on dynamic variable REF_REG value.
- The second IF block defines two OVERLAY layouts based on REF_REG. When REF_REG is initialized to 'Y', OVERLAY layout defines a scatter plot with regression line, plus a set of reference lines of BMI equals 25 (kg/m2) and 20 (kg/m2) respectively, and two reference lines of Weight at 55 and 65 Kg. When REF_REG equals to 'N', OVERLAY layout defines a simple scatter plot.
- Figure 6.1 is created with REF_REG set to 'Y', as a scatter plot of BMI vs. Weight with corresponding title, regression line, and reference lines for all three treatment groups.
- Figure 6.2 is created when REF_REG is set to 'N', as a scatter plot of Height vs. Weight with only two treatment groups Trt A and Trt B included.

Program 6

```
proc template;
  define statgraph plot_1;
     dynamic ref_reg _x _y;
     begingraph;
     if (ref_reg='Y')
         EntryTitle "Scatter plot of " _y " vs. " _x " with regression line";
     else
         EntryTitle "Scatter plot of " _y " vs. " _x;
     endif;
     if (ref_reg='Y')
        layout overlay / xaxisopts=(type=linear linearopts=( tickvaluelist=( 40 60
              80 100 120 140 ) viewmin=40 viewmax=140 ) )
              yaxisopts=( type=linear linearopts=( tickvaluelist=( 10 20 30 40 50 )
              viewmin=10 viewmax=50 ) );
        ReferenceLine x=55 / clip=true;
        ReferenceLine x=65 / clip=true;
        ReferenceLine y=20 / clip=true curvelabel="BMI=20 (kg/m2)"
                             curvelabellocation=inside;
        ReferenceLine y=25 / clip=true curvelabel="BMI=25 (kg/m2)"
                             curvelabellocation=inside;
```

```
        ScatterPlot X=_x Y=_y / primary=true Group=trt index=id NAME="scatter";
        lineparm slope=0.325 x=0 y=2.4/extend=true Lineattrs=( Color=blue
                Thickness=2) curvelabel="BMI=2.4+0.325*Weight" NAME="SERIES1"
                legendlabel=" ";
        DiscreteLegend "scatter" / Location=Outside valign=bottom halign=left
                                   title="Observed BMI:" border=no;
        DiscreteLegend "SERIES1" / Location=Outside valign=bottom halign=right
                                   title="Predicted BMI:" border=no;
        endlayout;
     else
        layout overlay;
        scatterPlot X=_x Y=_y / primary=true Group=trt index=id NAME="scatter";
        DiscreteLegend "scatter" / Location=Outside valign=bottom border=no;
        endlayout;
     endif;
     endgraph;
end;

ods rtf style=proj_101 file="G:\Figure6.1";
proc sgrender data=addm2 template=plot_1;
     dynamic ref_reg='Y' _x="Weight" _y="BMI";
run;

ods rtf style=proj_101 file="G:\Figure6.2";
proc sgrender data=addm2(where=(trtn ne 0)) template=plot_1;
     dynamic ref_reg='N' _x="Weight" _y="Height";
run;
ods rtf close;
```

Figure 6.1

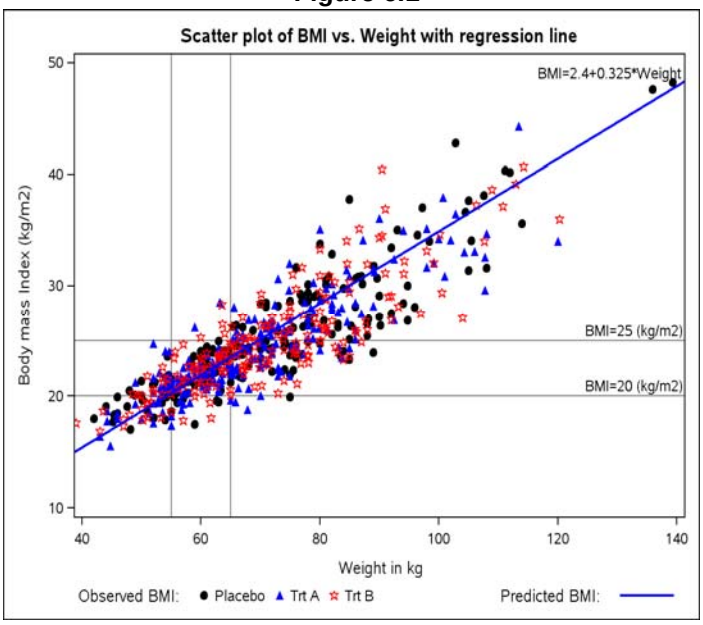
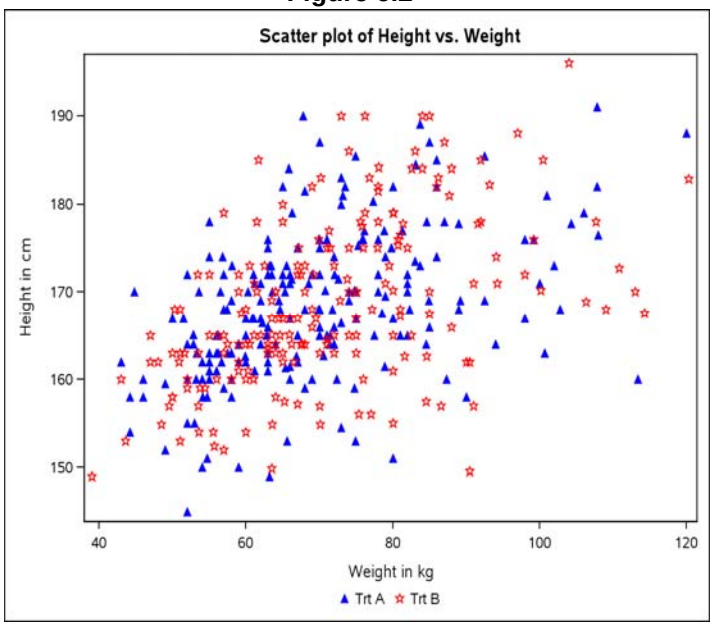


Figure 6.2



STORE AND SHARE TEMPLATE THROUGH ODS PATH STATEMENT

After a graph template or style template is developed and complied, it has to be stored into an item store in order to be used for graph creation and shared with other users. User-defined templates can be saved into three different template libraries: SASUSER.TEMPLAT, WORK.TEMPLAT, or a template library in a permanent location. The templates stored in SASUSER.TEMPLAT are only accessible for personal use. They are hanging around and can only be deleted when SAS is re-installed or by using Proc TEMPLATE DELETE statement. WORK.TEMPLAT is a good place for template development and temporary storage. This item store is automatically deleted when a SAS session is terminated, and it prevents failed attempts and bad templates to be stored into SASUSER.TEMPLAT. The template library in a permanent location is a preferred place to store well-developed templates to avoid deletion when SAS is re-installed, and most importantly, to give shared access to other users.

SAS automatically saves complied template in the first available item store in current SAS session. By default, all user defined templates are stored in SASUSER.TEMPLAT, which is always at the beginning of ODS Path of a new SAS session given no ODS path manipulation. Submit ODS PATH SHOW statement, the default ODS Path list is displayed in LOG window:

```
Current ODS PATH list is:
1. SASUSER.TEMPLAT(UPDATE)
2. SASHELP.TMPLMST(READ)
```

TEMPLATE STORAGE

There are two ways to change template storage location: 1) using Proc TEMPLATE STORE option to specify which library and item store to save the template; 2) using ODS PATH statement to change ODS Path setting. They are illustrated below.

1) Use Proc TEMPLATE STORE option to specify which library and item store to save the template

```
proc template;
     define statgraph plot_1/store=work.templat;
       more graph statements;
end;
```

Graph template PLOT_1 is saved to WORK.TEMPLAT, proved by the message in LOG window:

NOTE: STATGRAPH 'Plot_1' has been saved to: WORK.TEMPLAT

```
libname public " G:\pub";
proc template;
     define style proj_101 /store=public.templat;
       more style statements;
end;
```

Style template PROJ_101 is saved into PUBLIC.TEMPLAT, as shown in LOG window:

NOTE: STYLE 'Proj_101' has been saved to: PUBLIC.TEMPLAT

2) Use ODS PATH statement to change ODS Path setting

ODS PATH statement specifies the locations to write to or read from when creating or using the complied templates, as well as the order in which to search for them. PREPEND adds one or more locations to the beginning of a path. An item store in a shared drive can be inserted in front of the default ODS Path in either of the following equivalent ways:

```
libname public " G:\pub";
ods path (prepend) public.templat(update);
ods path public.templat(update) sasuser.templat sashelp.tmplmst;
```

The modified ODS PATH list is displayed in LOG window:

```
Current ODS PATH list is:
1. PUBLIC.TEMPLAT(UPDATE)
2. SASUSER.TEMPLAT(UPDATE)
3. SASHELP.TMPLMST(READ)
```

With this ODS PATH setting, re-run style template definition without STORE= option. Style PROJ_101 is then saved to PUBLIC.TEMPLAT, because PUBLIC.TEMPLAT is the first item store in current ODS Path list.

```
Proc template;
    Define style proj_101;
       more style statements;
end;
```

TEMPLATE ACCESS

Changing ODS Path list does not provide much advantage in template storage. However, it is an essential way to gain read access to any template that is stored in WORK.TEMPLAT or PUBLIC.TEMPLAT.

After graph template PLOT_1 and style template PROJ_101 have been developed, complied and stored into PUBLIC.TEMPLAT, other users can obtain read access and use them directly, as following:

```
libname public "G:\pub";
ods path (prepend) public.templat;

ods rtf style=proj_101 file="G:\...";
proc sgrender data=addm2 template=plot_1;
     dynamic ref_reg='Y' _x="Weight" _y="BMI";
run;
ods rtf close;
```

TEMPLATE CLEANNING

Lastly, there is a need to detect failed or obsolete templates and remove them from an item store. Proc TEMPLATE LIST statement displays the contents of an item store into OUTPUT window; Proc TEMPLATE DELETE statement allows the elimination of a complied template from an item store; Proc DATASETS DELETE statement can be used to delete an entire item store from a library:

```
*** statement to display complied templates of an item store in OUTPUT window:
Proc template;
     list/ store=sasuser.templat;
run;

*** statement to delete template MYPLOT2 from SASUER.TEMPLAT:
proc template;
     delete myplot2;
run;

*** statement to delete the whole PUBLIC.TEMPLAT item store:
proc datasets library=public;
     delete templat (memtype=itemstor);
run;
```

CONCLUSIONS

ODS Graphics provides an efficient way of creating graphs as compared with traditional SAS/GRAPH procedures. Depending on costumer needs, graphic programs with different levels of complexity and flexibility can be developed. User specific graph template and style template written in GTL are usually required to properly control graphic appearance, especially when handling grouped data. Because GLT is a very comprehensive language and differs from conventional SAS syntax, it is not easy to debug. The incremental method outlined in this paper discloses exactly when and where a problem occurs. The same approach can be adapted to develop highly customized graph template and style template for common project needs.

REFERENCES

SAS Institute (2009d), *SAS/GRAPH® 9.2 Statistical Graphics Procedures Guide*, SAS Institute, Inc., Cary, NC.

SAS Institute (2011), *SAS/GRAPH® 9.3 Graph Template Language User's Guide*, SAS Institute, Inc., Cary, NC.

Kuhfeld, Warren F. 2010. *Statistical Graphics in SAS®: An Introduction to the Graph Template Language and the Statistical Graphics Procedures*. Cary, NC: SAS Institute Inc.

CONTACT INFORMATION

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

Yunzhi Ling
Sanofi
55C-330A, 55 Corporate Drive
Bridgewater, NJ 08807
908-981-6169
Yunzhi.ling@sanofi.com

Mei Wu
Sanofi
55C-330A, 55 Corporate Drive
Bridgewater, NJ 08807
908-981-6073
mei.wu@sanofi.com