Paper RF-011-2014

A Simplified Approach to Add Shading to your Graph using GTL

Jennifer L Yen, Abbott Laboratories, Abbott Park, IL

ABSTRACT

You have mastered the skill of using SAS annotate facility along with ODS statistical graph procedures to customize your graphs. You know how to use the annotate function to add color shading anywhere on your graph, and you have figured out how to customize your axis labels using title and footnote statements. If you have always hoped for a simpler way to do all of this, Graphic Template Language (GTL) is the answer! This paper will show you how to leverage the features of GTL to add color shading to your graph and customize your axis labels.

INTRODUCTION

Graph Template Language (GTL) introduces powerful, yet simple ways to produce customized graph. This paper will show you step by step how to customize a simple scatter plot with upper and lower boundaries and shading in between. It will also show you how to put subscript and superscript in axis labels with ENTRY statements.

BASIC SYNTAX OF PROC TEMPLATE WITH GTL

For a refresher, here is the skeleton syntax for a GTL graph.

```
proc template;
    define statgraph TEMPLATE.NAME;
        begingraph /options;
        entrytitle “TITLE”;
                layout statement(s) / options;
                        plot statement(s) / options;
                endlayout statement(s);
        endgraph;
    end;
run;
```

After defining the template, PROC SGRENDER is used to render the graph

```
proc sgrender data = <DATA SET> template = “TEMPLATE.NAME”;
run;
```

FIGURE 1: A SIMPLE SCATTER PLOT WITH GTL

1. BEGINGRAPH AND ENDGRAPH BLOCK

All template definitions in the Graphics Template Language (GTL) must start with a BEGINGRAPH statement and end with an ENDGRAPH statement. The options for BEGINGRAPH are used to control the size and appearances of the graph. There are many options available. Three of them are used in this paper.

- DESIGNWIDTH and DESIGNHEIGHT options are used to change the output size for a single graph. By default, graphs are rendered at 640px by 480px (4:3 aspect ratio).
- BORDER option specifies whether a border is drawn around the graph. ON is the default setting.
- DRAWSPACE specify a global drawing space and drawing units for all of the draw statements within the BEGINGRAPH block. LAYOUTPERCENT is the default setting.

2. ENTRYTITLE STATEMENT

ENTRYTITLE statement adds text to the title area of the graph. HALIGN and TEXTATTRS can be used to define the alignment and properties of the text.

- HALIGN = RIGHT|LEFT|CENTER specify the alignment of the text.
- TEXTATTRS = (WEIGHT=, FAMILY=, SIZE=) specify the font weight, type, and size respectively. When TEXTATTRS is used as a prefix option, the text properties stay in effect for subsequent text items unless changed by another TEXTATTRS prefix option.
- {UNICODE}, {SUB}, and {SUP} text commands can also be used with ENTRYTITLE statement to display special characters in the graph title.

3. XAXISOPTS AND YAXISOPTS, OPTIONS FOR LAYOUT OVERLAY STATEMENT

XAXISOPTS and YAXISOPTS are options for the LAYOUT OVERLAY statement. They are used to specify features for X axes and Y axes.

- DISPLAY = (LABEL LINE TICKVALUES TICKS) control how the labels, line, tick values, and ticks will be displayed along the axis.

- OFFSETMIN= specifies the distance from the beginning of the axis to the first tick and OFFSETMAX= specifies the distance from the last tick to the end of the axis. The values can range from 0 to 1.

- VIEWMIN= and VIEWMAX= specify the data range for the axis.

- TICKSTYLE and TICKVALUEATTRS specify how the ticks and tick values will be displayed.

- LABEL and LABELATTRS specify how the labels for the axis will be displayed

DEFINE GTL TEMPLATE

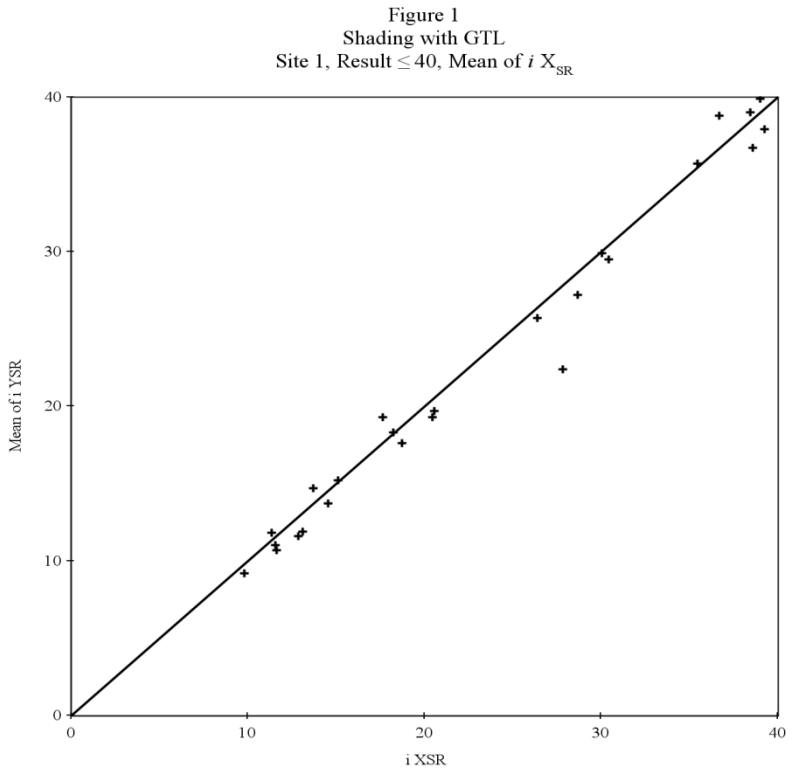
```
Proc Template;
 Define statgraph scatteratd;

 BeginGraph / designwidth = 650px designheight = 700px border = off drawspace = datavalue;

  ***************************************************************************
  Define title of the graph with ENTRYTITLE statement
  ***************************************************************************;
```

```
   EntryTitle halign=center 'Figure 1'
              / textattrs=(family = "Times New Roman" weight = normal size=12pt);

   EntryTitle halign=center 'Shading with GTL'
              / textattrs=(family = "Times New Roman" weight = normal size=12pt);

   EntryTitle halign=center
          textattrs=(family = "Times New Roman" weight = normal  size=12pt)
                       'Site 1, Result ' {unicode "2264"x} ' 40, Mean of'
             textattrs=(family = "Times New Roman" weight = normal  style = italic size=12pt) ' i'
             textattrs=(family = "Times New Roman" weight = normal  size=12pt) 'X' {sub 'SR'};

  Layout overlay /
               xaxisopts=(
                     display=(label line tickvalues ticks)
                     OFFSETMIN = 0 OFFSETMAX = 0
                     LINEAROPTS=(viewmin=0 viewmax=40)
                     tickstyle=across
                     TICKVALUEATTRS = (family = "Times New Roman" size=9pt)
                     labelattrs=(family = "Times New Roman" size=9pt)
                     label='i XSR')
               yaxisopts=(
                     display=(label line tickvalues ticks)
                     LINEAROPTS=(viewmin=0 viewmax=40)
                     tickstyle=across
                     OFFSETMIN = 0 OFFSETMAX = 0
                     TICKVALUEATTRS = (family = "Times New Roman" size=9pt)
                     labelattrs=(family = "Times New Roman" size=9pt)
                     label='Mean of i YSR');


        scatterplot x = x y = y /yaxis = y name = "ATD"
              markerattrs = (color = black symbol = plus size = 2.5pt weight = bold);



         beginpolyline x=0 y=0 /xaxis = x yaxis = y  xspace = datavalue yspace = datavalue
                  LINEATTRS = (color = black pattern = 1);
            draw x=40 y=40;

         endpolyline ;
     Endlayout;
  Endgraph;
End;
run;
```

RENDER GTL TEMPLATE WITH PROC SGRENDER

```
proc sgrender data=plotdata1 template=scatteratd; run;
```

Figure 1 shows the scatter plot with the identity line produced by the above template.



Here we started with a basic scatter plot with identity line. In the following examples, I will show you step by step how to add color shading to the graph and customize the labels for X and Y axis to show italic *i* and subscript $_{SR}$. In addition, example about how to use macro variables with GTL template will also be shown.

FIGURE 2: SCATTER PLOT WITH LOWER AND UPPER BOUNDARY

To add the lower and upper boundary for the above scatter plot, BEGINPOLYGON and ENDPOLYGON block along with DRAW statements are used. Options specified in BEGINPLYGON statement apply to all DRAW statements in the block.

- XAXIS specifies whether the X value is interpreted using the primary X axis scale or the secondary X (X2) axis scale.
- YAXIS specifies whether the X value is interpreted using the primary Y axis scale or the secondary Y (Y2) axis scale.
- XSPACE and YSPACE specify the drawing space and drawing units for interpreting the axis values. When DATAVALUE is used, the actual value of x and y are used to draw the graph. You can also use GRAPHPERCENT, GRAPHPIXEL, LAYOUTPERCENT, LAYOUTPIXEL, WALLPERCENT, WALLPIXEL, DATAPERCENT, and DATAPIXEL.

BEGINPOLYGON x=0 y=0/ xaxis = x yaxis = y xspace = datavalue yspace = datavalue;

draw x=3 y=0;
draw x=10 y=8;
draw x=40 y=28;
draw x=40 y=40;
draw x=28 y=40;
draw x=10 y=12;
draw x=0 y=3;

ENDPOLYGON;

The input values for the X and Y in BEGINPOLYGON statement and each DRAW statement are the coordinates for each vertex of the below polygon counter clockwise.

Figure 2 shows the graph with lower and upper boundary added.

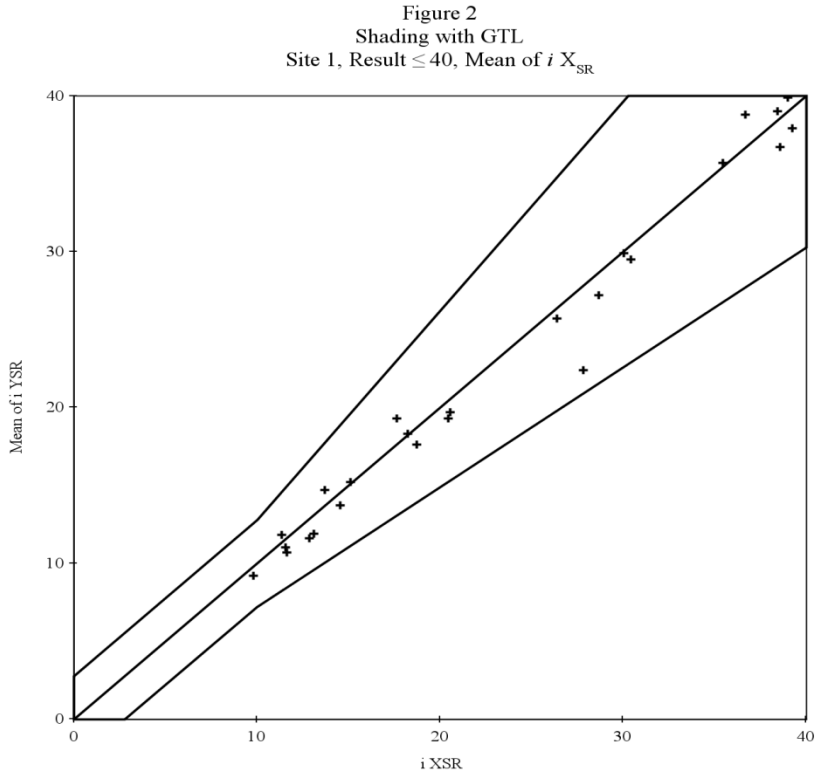


FIGURE 3: ADD SHADING TO THE AREA BETWEEN LOWER AND UPPER BOUNDARY AND CHANGE OUTLINE PATTERN TO DASHED LINE

To add shading and change the line type to the area between lower and upper boundary, more options are specified for BEGINPOLYGON statement.

- DISPLAY= (FILL OUTLINE) specifies that fill color and outline will be displayed for the polygon.

- FILLATTRS = (COLOR=GRAY TRANSPARENCY = 0.6) specifies the color GRAY will be use for the shading and transparency for the fill is 0.6.
- OUTLINEATTRS specifies the outline type is dashed and color is black.

```
BEGINPOLYGON  x=0 y=0/ xaxis = x yaxis = y xspace = datavalue yspace = datavalue
                          display=(fill outline)
                          fillattrs=(color=gray transparency=0.6)
                          outlineattrs=( pattern=shortdash color=black);
        draw x=3 y=0;
        draw x=10 y=8;
        draw x=40 y=28;
        draw x=40 y=40;
        draw x=28 y=40;
        draw x=10 y=12;
        draw x=0 y=3;
ENDPOLYGON;
```

Because options specified for BEGINPOLYGON statement apply to the whole block, the dashed lines also show up around the border of the graph where the outline of the polygon overlaying the border.

Figure 3 shows the graph with shading added to the area between lower and upper boundary and dashed outline for the polygon

Figure 3
Shading with GTL
Site 1, Result ≤ 40, Mean of *i* X_{SR}

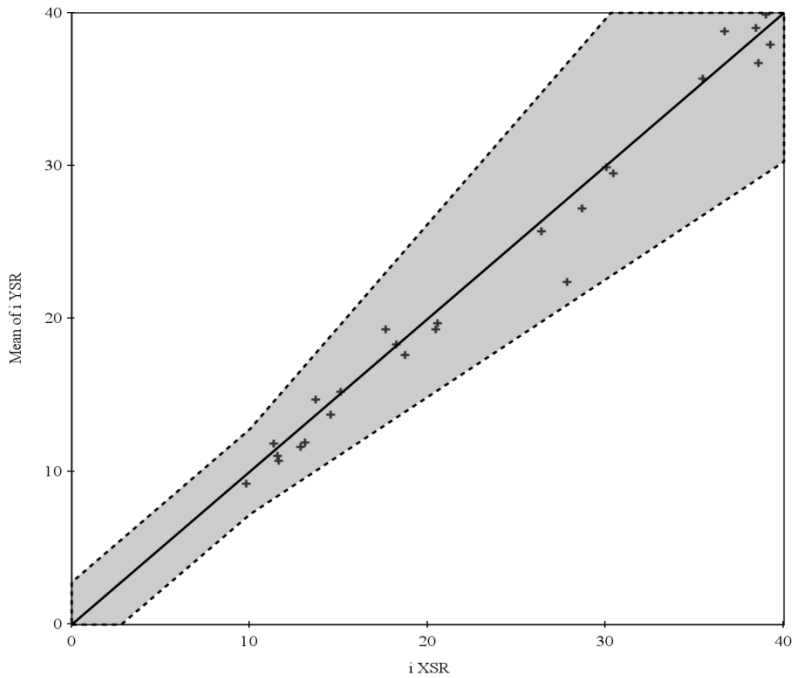
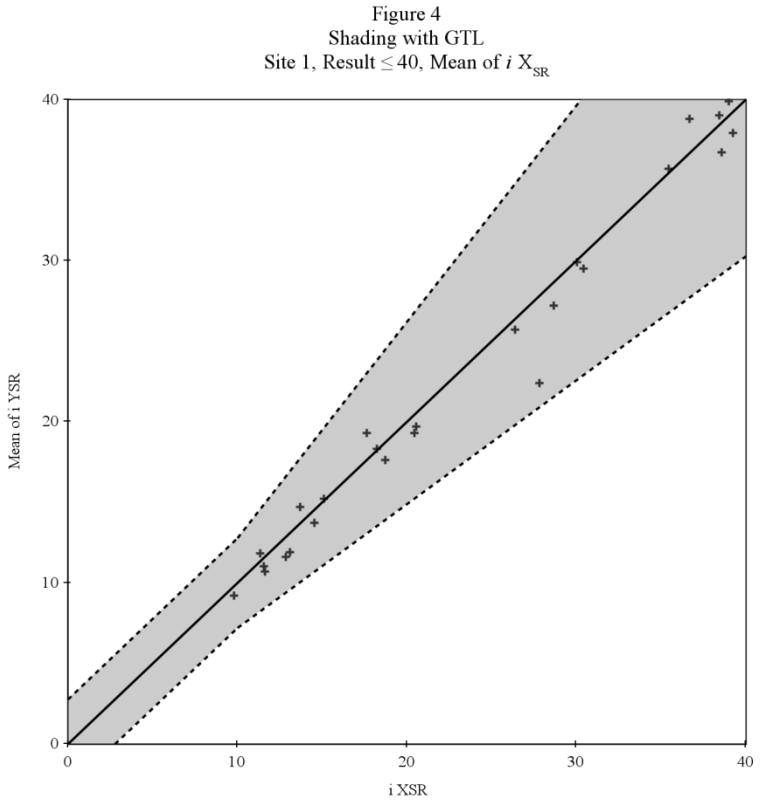


FIGURE 4: DRAW DASHED LINES ONLY FOR THE BOUNDARY, NOT ALL OUTLINES OF THE POLYGON

In this example, the codes are updated to only draw dashed line on the upper and lower boundary, not around the border of the graph. First, update the DISPLAY option for BEGINPOLYGON to suppress the outline and only show shading for the polygon by using DISPLAY = (FILL). Second, BEGINPOLYLINE and ENDPOLYLINE blocks are introduced to draw the two dashed lines for lower and upper boundaries. The BEGINPOLYLINE and ENDPOLYLINE block works in similar fashion as BEGINPOLYGON and ENDPOLYGON block.

```
BEGINPOLYGON  x=0 y=0/ xaxis = x yaxis = y xspace = datavalue yspace = datavalue
                          display=(fill)  fillattrs=(color=gray transparency=0.6) ;
        draw x=3 y=0;
        draw x=10 y=8;
        draw x=40 y=28;
        draw x=40 y=40;
        draw x=28 y=40;
        draw x=10 y=12;
        draw x=0 y=3;
  ENDPOLYGON;

  BEGINPOLYLINE x=3 y=0 /xaxis = x yaxis = y
                          xspace = datavalue yspace = datavalue
                          LINEATTRS = (color = black pattern = 2);
            draw x=10 y=8;
            draw x=40 y=28;
  ENDPOLYLINE;

  BEGINPOLYLINE x=28 y=40 /xaxis = x yaxis = y
                          xspace = datavalue yspace = datavalue
                          LINEATTRS = (color = black pattern = 2);
            draw x=10 y=12;
            draw x=0 y=3;
  ENDPOLYLINE;
```

Figure 4 shows the graph with area shaded inside the boundary and dashed line shown only on the lower and upper boundary lines



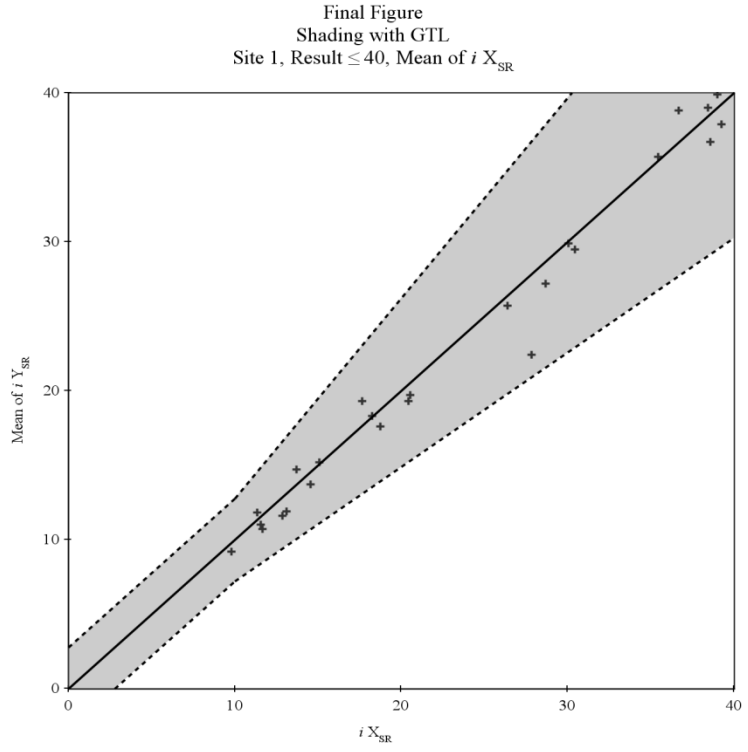
FINAL FIGURE: CUSTOMIZE X AND Y AXIS LABELS

The {UNICODE}, {SUB}, and {SUP} text commands apply only to the ENTRY, ENTRYTITLE, and ENTRYFOOTNOTE statements. For axis labels, unicode characters can be used with "in-line formatting", in which the unicode text command is preceded by an ODS escape character, e.g. ^{unicode "2264"x}. Currently, only the {UNICODE} text command is recognized, not {SUB} or {SUP} for axis labels. In order to customize the X and Y labels to show italic *i* and subscript $_{SR}$, the DISPLAY option for XAXISOPTS and YAXISOPTS statements are modified to suppress labels by using DISPLAY = (LINE TICKVALUES TICKS), LAYOUT LATTICE and ENTRY statements are also used to customize labels for axis.

```
Layout lattice / rows=2 columns=1 rowweights=(.95 .05) rowgutter=1;
   Layout lattice / rows=1 columns=2 columnweights=(0.05 0.95) columngutter=1;

    /***Customize Y axis label with ENTRY statement***/

     Entry halign=left textattrs=(family = "Times New Roman" size=9pt) 'Mean of'
                  textattrs=(family = "Times New Roman" style = italic size=9pt) ' i'
                  textattrs=(family = "Times New Roman" size=9pt) 'Y' {sub 'SR'}/ROTATE = 90;
          Layout overlay /
            xaxisopts=(
                  display=(line tickvalues ticks)
                  OFFSETMIN = 0 OFFSETMAX = 0
                  LINEAROPTS=(viewmin=0 viewmax=40)
                  tickstyle=across
```

```
                TICKVALUEATTRS = (family = "Times New Roman" size=9pt))
            yaxisopts=(
                display=(line tickvalues ticks)
                LINEAROPTS=(viewmin=0 viewmax=40)
                tickstyle=across
                OFFSETMIN = 0 OFFSETMAX = 0
                TICKVALUEATTRS = (family = "Times New Roman" size=9pt));
        Endlayout;
    Endlayout;

    Layout gridded/columns=1  border=false;

      /***Customize X axis label with ENTRY statement***/

      Entry halign=center
                    textattrs=(family = "Times New Roman" style = italic size=9pt) ' i'
                    textattrs=(family = "Times New Roman" size=9pt) 'X' {sub 'SR'};
    Endlayout;
  Endlayout;
```

Final figure shows the customized X and Y axis labels with italic '*i*' and subscript '$_{SR}$'



USING MACRO VARIABLES WITH PROC TEMPLATE

For BEGINPOLYGON and BEGINPOLYLINE statements, user needs to specify the X and Y value for each draw satement. Macro variables can be used to eliminate the need to update the value manually for different graph. To use macro variable in PROC TEMPLATE with GTL, MVAR statement is used.

```
proc template;
  Define statgraph scatteratd;
  mvar x1 y1 x2 y2 x3 y3 x4 y4 x5 y5 x6 y6 x7 y7 x8 y8;
  layout overlay;
    beginpolygon x="&x1" y="&y1" / xaxis = x yaxis = y  xspace = datavalue yspace = datavalue
                display=(fill) fillattrs=(color=gray transparency=0.6);
                draw x="&x2" y="&y2";
                draw x="&x3" y="&y3";
                draw x="&x4" y="&y4";
                draw x="&x5" y="&y5";
                draw x="&x6" y="&y6";
                draw x="&x7" y="&y7";
                draw x="&x8" y="&y8";
    endpolygon;

    beginpolyline x="&x2" y="&y2" /xaxis = x yaxis = y  xspace = datavalue yspace = datavalue
                        LINEATTRS = (color = black pattern = 2);
                  draw x="&x3" y="&y3";
                  draw x="&x4" y="&y4";
     endpolyline;
  endlayout;
 end;
run;
```

CONCLUSION

Graphic Template Language (GTL) might seem daunting at first glance. It is actually a very structured and comprehensive language and easy to grasp once you give it a chance. It provides ways to customize graph beyond the cabalities of SAS graph procedures.

REFERENCES

Statistical Graphics in SAS, Warren F. Kuhfeld

SAS(R) 9.3 Graph Template Language: Reference, Third Edition

ACKNOWLEDGMENTS

Special thanks to Rose Grandy for her encourgement and review, and Shawn Du and Hope Knuckles for their support.

CONTACT INFORMATION

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

Name: Jennifer L Yen
Enterprise: Abbott Laboratories
Address: 100 Abbott Park Road
City, State ZIP: Abbott Park, IL 60064
Work Phone: 224-668-2942

E-mail: jennifer.L.yen@abbott.com