How to improve your figure
An overview of annotation techniques in Graph Template Language

Konrad Żywno, inVentiv Health Clinical, Berlin, Germany
Bartosz Kutyla, SAS Institute, Warsaw, Poland

ABSTRACT
Plots are commonly used in the pharmaceutical industry to analyse data. Since SAS® 9.2 it has been possible to create high quality graphs using the newly introduced Graph Template Language (GTL). GTL is evolving with each new SAS version to encompass the expectations of users. From SAS 9.3, draw statements can be used to add graphics elements, but this involves modifying your template code. This paper presents an overview of annotation techniques in GTL, introduced with SAS 9.3 and further developed in SAS 9.4, to overcome the limitation of the draw statements by enabling you to separate the annotation graphics instructions from your template code. The whole range of annotate functions available in Maintenance One of SAS 9.4 and Statistical Graphics (SG) annotation macros available to facilitate the production of annotation datasets are discussed. Furthermore the paper contains example code which demonstrates the annotation facility together with the resulting plots.

INTRODUCTION
The paper covers the production of high quality enhanced plots using Graph Template Language introduced with SAS® 9.2 and demonstrates how to add graphics elements to your graphs using the annotation facility available in Maintenance One of SAS 9.4. Like the GTL draw statements, the annotation facility enables the user to add the following elements:

- text
- arrows and lines
- geometric shapes, such as ovals, rectangles, polylines, and polygons
- images

The whole range of annotate functions available in Maintenance One of SAS 9.4 is discussed together with the Statistical Graphics (SG) annotation macros which facilitate the production of the annotation dataset. This dataset, a required part of the annotation facility, contains "metadata" which describes the elements to be added to the plot.

The tasks which have to be completed before adding the annotations are described. Then a simple plot is produced using GTL. The main part of the paper describes the annotation functions and their main attributes. Each function is demonstrated using an SG annotation macro together with its main parameters. After describing the function and its associated macro, the code and how it impacts the example figure are presented. Also the corresponding DRAW statements, available in the TEMPLATE procedure, are described. A wide set of case scenarios is discussed and thereby a comprehensive overview of annotation techniques in Graph Template Language introduced with SAS 9.4 is given.

DATASET USED
For the purposes of this paper we have created a dummy dataset containing three variables: treatment, visit and the mean score of a dummy test. The code for creating the dataset is presented in APPENDIX 1.

SAMPLE FIGURE
The sample figure which we use in our paper is a scatter and line plot of mean score by visit within each treatment. To create the figure we first create a simple template:

```sas
PROC TEMPLATE;
    DEFINE STATGRAPH Tmp1;
    BEGINGRAPH;
        ENTRYTITLE halign=left "Figure 1 - no annotations";
        ENTRYFOOTNOTE halign=left "Authors: Konrad Zywno & Bartosz Kutyla";
        LAYOUT OVERLAY / XAXISOPTS=(LABEL="Visit" offsetmax=0.05 offsetmin=0.02 TYPE=linear
                                   linearopts=(tickvalueFormat=best12. tickvaluefitpolicy=rotate))
                        YAXISOPTS=(LABEL="Mean Score" TYPE=linear
                                   linearopts=(tickvalueFormat=best12.));
        SERIESPLOT X=VISIT Y=MEAN_SCORE / name="series1" group=TREATMENT;
        SCATTERPLOT X=VISIT Y=MEAN_SCORE / name="scatter1" group=TREATMENT;
        DiscreteLegend "series1" "scatter1" / title="Treatment" merge=true;
    ENDLAYOUT;
END TEMPLATE;
```
And afterwards we call the SGRENDER procedure (this procedure generates graphic outputs from templates created with GTL) with the TEMPLATE and DATA options.

PROC SGRENDER TEMPLATE=tmpl DATA=input;
RUN;

Figure 1: No annotations

ENHANCING THE GRAPHIC
A simple figure may not be enough for our needs. For example, we may need to add new graphic elements to the figure, mark a value in some way, or add some text, and it may not always be possible to achieve what we want without the annotation facility. Before adding annotation elements to a graph we need to do two things. Firstly we need to create an SG annotation dataset which contains the instructions needed to create the annotation elements. Secondly we need to modify the SGRENDER procedure so that it knows to use the SG annotation dataset we have created. SAS 9.4 introduced a new option, SGANNO, to the SGRENDER procedure enabling the use of annotation datasets. By using the SGANNO option, users can specify the name of a dataset which contains the annotation instructions and options:

PROC SGRENDER TEMPLATE=tmpl DATA=input sganno=anno;
RUN;

The annotation dataset is used only if the ANNOTATE statement is included within the TEMPLATE procedure; if it is, the instructions contained in the annotation dataset are read and applied to the output figure.

PROC TEMPLATE;
   DEFINE STATGRAPH Tmpl;
   BEGINGRAPH;
   LAYOUT OVERLAY;
   ...
   ANNOTATE;
   ENDLAYOUT;
   ENDFRAME;
RUN;

We also have to decide where to put the annotation. If we want it in the margin then the PAD option has to be set. The PAD option specifies the size of the margin inside the figure’s outer border.

Instead of creating an annotation dataset it is also possible to modify the template code to add annotations to a graphic by adding multiple draw statements. Both techniques are described below.

DRAWING AREA
Apart from deciding the kind of annotation we want we also need to decide where to put it. As may be expected, we can control both the position and scaling of the annotation. Annotation can be positioned in one of four so-called drawspace
areas as defined via the DRAWSPACE variable. Values for this variable take the form <area><unit>. The unit (percent, pixel or value) enables us to control the scaling of the annotation. DRAWSPACE has the following permitted values:

- GRAPHPERCENT, GRAPHPIXEL (drawspace area is the entire graph)
- LAYOUTPERCENT, LAYOUTPIXEL (drawspace area is the entire graph excluding titles and footnotes)
- WALLPERCENT, WALLPIXEL (drawspace are is the area within the axes including offsets)
- DATAPERCENT, DATAPIXEL, DATAVALUE (drawspace area is the area within the axes not including offsets)

Note that the DRAWSPACE variable is not supported by the POLYCONT and TEXTCONT functions.

Instead of setting the drawing space globally for all axes, we can also control the drawing space for each axis and coordinate individually. To do so we set X1SPACE (not supported by TEXTCONT), X2SPACE (supported only by the ARROW and LINE functions), Y1SPACE (not supported by TEXTCONT) and Y2SPACE (supported only by the ARROW and LINE functions).

ANNOTATION DATASET REQUIREMENTS
After deciding what kind of annotation we want an annotation dataset containing the instructions for creating the annotation elements needs to be set up. Annotations are created in order of their appearance in the annotation dataset. The plot-creating procedure looks into each observation and searches for the variables with names predefined within the annotation facility. All other variables are ignored. There is a reserved set of keywords which is used to specify the drawing operation. Variables within an annotation dataset can be categorised into the following main types:

- function, which indicates what element should be drawn
- variables specifying the drawing space used by the annotation
- position variables, specifying the point at which the annotation element is to be drawn
- attribute variables, specifying the style of the annotation elements – e.g. colour or text font.

The most important variable in the dataset is FUNCTION, which specifies the kind of annotation to be added. The available functions are: TEXT, TEXTCONT, ARROW, LINE, OVAL, RECTANGLE, POLYGON, POLYLINE, POLYCONT and IMAGE. All of these are described in the examples below.

TEXT
The TEXT function enables the user to place text in the graph output. The only required variable in the annotation dataset apart from the FUNCTION variable is the LABEL variable which contains the text to be added.

Let's take a look at our example. Say we want to write text next to the maximum value per treatment indicating that it is the maximum value. We identify the maximum coordinates for treatment and placebo and store them in macro variables. We'll call the macro variables &max_x_trt., &max_y_trt., &max_x_plac., &max_y_plac. respectively. These macro variables will be used as coordinates in the X1 variables and Y1 annotation variables. We will position the text to the left of the datapoint on the figure and just below it (ANCHOR=TOPRIGHT). As we are using the data values to identify where the annotation should be placed we need to set DRAWSPACE to DATAVALUE. We will let SAS calculate the width of the annotation automatically so we do not set the WIDTH variable but we will give the dimension unit to be used for the automatic calculations. Therefore we set the WIDTHUNIT variable to DATA. Say we also want to display the maximum score in bold text. To do so we need to use the TEXTCONT function and change the TEXTWEIGHT to BOLD. This function needs to be added to the annotation dataset just after the preceding TEXT function.

DATA anno;
    ATTRIB FUNCTION LENGTH = $20;
    ATTRIB LABEL LENGTH = $100;

    FUNCTION = "text";
    X1 = &max_x_plac.;
    Y1 = &max_y_plac.;
    ANCHOR = "TOPRIGHT";
    WIDTHUNIT = "DATA";
    DRAWSPACE = "DATAVALUE";
    LABEL = "Maximum for Placebo:"
    OUTPUT;

    FUNCTION = "textcont";
    TEXTWEIGHT = "BOLD";
    LABEL = " &max_y_plac.";
    OUTPUT;

    FUNCTION = "text";
    X1 = &max_x_trt.;
    Y1 = &max_y_trt.;
    ANCHOR = "TOPRIGHT";
    WIDTHUNIT = "DATA";
    DRAWSPACE = "DATAVALUE";
    TEXTWEIGHT = "NORMAL";
    LABEL = "Maximum for Treatment:"
    OUTPUT;

3
Since Maintenance One of SAS 9.4 we can use SG annotation macros, which facilitate the production of annotation datasets. For example, the macro for the TEXT function is %SGTEXT and for TEXTCONT it is %SGTEXTCONT. The arguments to the macro are simply the names of the variables which relate to a given function. To compile the SG annotation macros we need to run the %SGANNO macro:

%SGANNO;
DATA anno_macro;
  %SGTEXT(
    X1 = &max_x_plac.,
    Y1 = &max_y_plac.,
    ANCHOR = "TOPRIGHT",
    WIDTHUNIT = "DATA",
    DRAWSPACE = "DATAVALUE",
    LABEL = "Maximum for Placebo:"
  )
  %SGTEXTCONT(
    TEXTWEIGHT= "BOLD",
    LABEL = " &max_y_plac."
  )
  %SGTEXT(
    X1 = &max_x_trt.,
    Y1 = &max_y_trt.,
    ANCHOR = "TOPRIGHT",
    WIDTHUNIT = "DATA",
    DRAWSPACE = "DATAVALUE",
    TEXTWEIGHT= "NORMAL",
    LABEL = "Maximum for Treatment:"
  )
  %SGTEXTCONT(
    TEXTWEIGHT= "BOLD",
    LABEL = " &max_y_trt."
  )
RUN;

Theoretically the same annotations can be added to the figure using the draw statements within the template. Usually the option names and values are the same as in the annotation facility. But this is not always the case. In our example the option names for the DRAWTEXT statement are the same as for the TEXT function in the annotation facility. To change the attributes of the appended text item (the maximum score) we need to set the TEXTATTRS prefix option which precedes the text whose attributes are to be changed. As we want to print the maximum score values in bold then we set the WEIGHT option to BOLD and enclose it in brackets within the TEXTATTRS option.

PROC TEMPLATE;
  DEFINE STATGRAPH Tmpl;
  BEGINGRAPH;
  ...  
    DRAWTEXT "Maximum for Placebo" textattrs=(weight=bold) " &max_y_plac."
    x = &max_x_plac.
    y = &max_y_plac.
    ANCHOR = TOPRIGHT
    WIDTHUNIT = DATA
    DRAWSPACE = DATAVALUE;
  ENDGRAPH;
END;
As well as the attributes set in our example we can also specify whether we want a border around the text, change the pattern, thickness and colour of the border, change the colour of the background, justify the text, rotate it, change the colour, font and size of the text and set it as italic or bold. We can also specify whether the annotation is drawn in front of or behind the graph and set its transparency level.

AN ARROW

An arrow can be annotated to the output using the arrow function and providing the arrow’s coordinates via the \texttt{X1, Y1, X2} and \texttt{Y2} variables. These define where the arrow starts and ends. By default the arrowhead is located at \texttt{X2, Y2} (\texttt{DIRECTION = OUT}) but this can easily be changed by setting the \texttt{DIRECTION} variable to \texttt{IN}.

Say we want an arrow to point to the maximum of each treatment. First, though, we must make space for the arrows by moving the text away from the points a little. The text and arrow end are located between the maximum score’s visit and the previous visit. We will change the position of the arrowhead by setting the \texttt{DIRECTION} variable to \texttt{IN}.

\begin{verbatim}
DATA anno;
...(TEXT FUNCTION)... 
FUNCTION = "arrow";
X1 = &max_x_plac.;
Y1 = &max_y_plac.;
X2 = %sysevalf(&max_x_plac. - 0.5);
Y2 = %sysevalf(&max_y_plac. - 0.5);
DRAWSPACE = "DATAVALUE";
DIRECTION = "IN";
OUTPUT;

FUNCTION = "arrow";
X1 = &max_x_trt.;
Y1 = &max_y_trt.;
X2 = %sysevalf(&max_x_trt. - 0.5);
Y2 = %sysevalf(&max_y_trt. - 0.5);
DRAWSPACE = "DATAVALUE";
DIRECTION = "IN";
OUTPUT;
RUN;
\end{verbatim}
Table 2: arrow annotations

As with **TEXT** function we can facilitate the creation of the annotation dataset using the `%SGARROW` macro.

```plaintext
DATA anno_macro;

...(TEXT FUNCTION)...

%SGARROW(
  X1 = &max_x_plac.,
  Y1 = &max_y_plac.,
  X2 = %sysevalf(&max_x_plac. - 0.5),
  Y2 = %sysevalf(&max_y_plac. - 0.5),
  DRAWSPACE = "DATAVALUE",
  DIRECTION = "IN"
)

%SGARROW(
  X1 = &max_x_trt.,
  Y1 = &max_y_trt.,
  X2 = %sysevalf(&max_x_trt. - 0.5),
  Y2 = %sysevalf(&max_y_trt. - 0.5),
  DRAWSPACE = "DATAVALUE",
  DIRECTION = "IN"
)

RUN;

The same results can be achieved using the **DRAWARROW** statements within the template. Please note that the name of the option responsible for the direction of arrowhead is different. What we achieved with the **DIRECTION** variable within the annotation facility we now achieve using the **ARROWHEADDIRECTION** option within the **DRAWARROW** statement.

```plaintext
PROC TEMPLATE;
  DEFINE STATGRAPH Tmpl;
  BEGINGRAPH;
    LAYOUT OVERLAY;

    ...(DRAWTEXT)...
      DRAWARROW X1 = &max_x_plac. Y1 = &max_y_plac.
      X2 = %sysevalf(&max_x_plac. - 0.5) Y2 = %sysevalf(&max_y_plac. - 0.5)
      /ARROWHEADDIRECTION = IN DRAWSPACE = DATAVALUE;
    
    DRAWARROW X1 = &max_x_trt. Y1 = &max_y_trt.
    X2 = %sysevalf(&max_x_trt. - 0.5) Y2 = %sysevalf(&max_y_trt. - 0.5)
    /ARROWHEADDIRECTION = IN DRAWSPACE = DATAVALUE;

    ENDLAYOUT;
  ENDDOCUMENT;
RUN;
```
Arrow annotation also allows us to change the pattern, thickness and colour of the arrow, change the direction of the arrow or set the arrow head on both ends of the line. We can also change the shape of the arrowhead and specify whether the annotation is drawn in front of or behind the graph and set its transparency level.

**A LINE**

To annotate the output with a line we need to use the `LINE` function and set the $X1$, $Y1$, $X2$ and $Y2$ functions. Just like with an arrow, these are the coordinates of the line and define where the line starts and ends.

In our example we will draw a line between the mean score values for treatment and placebo per each visit. And here once again we can see that the annotation facility is better than the draw statements. To create the annotation dataset we can use the input dataset and easily calculate the coordinate variables. To use the draw statements we have to put all coordinates to macro variables and then use them in the template. Doing it like this certainly requires more effort. A faster way is to calculate the coordinates for the line and then put text next to the line saying what the difference in mean score between placebo and treatment is. The text is written to the right of each line so we need to add some space to the right of the graph to be able to fit the whole text next to the last line. To do so within a template we need to specify the `PAD` option. We will set the right margin to be 10% of the whole graph.

``` SAS 
PROC TEMPLATE; 
   DEFINE STATGRAPH Tmpl; 
   BEGINGRAPH; 
      LAYOUT OVERLAY / ... PAD=(RIGHT=10%); 
      ... 
      ENDLAYOUT; 
   ENDFRAME; 
RUN; 
```

We will create the coordinates by setting the dataset twice and renaming the mean score for placebo to `mean_score_plac` and for the treatment to `mean_score_trt`:

``` SAS 
DATA anno; 
   SET input(WHERE=(treatment = 0)  RENAME=(mean_score = mean_score_plac)); 
   SET input(WHERE=(treatment_trt = 1) KEEP=mean_score treatment_RENAME=(mean_score = mean_score_trt treatment=treatment_trt)); 
RUN; 
```

We will change the line pattern to a dashed line by setting the `LINEPATTERN` variable, and the line colour to black by setting the `LINECOLOR` variable.

``` SAS 
DATA anno; 
   SET anno(IN= a) anno(IN= _b); 
   IF _a THEN DO; 
      ...(TEXT FUNCTION)... 
   END; 
   IF _b THEN DO; 
      FUNCTION = "line"; 
      Y1 = mean_score_plac; 
```

Figure 3: arrow annotations
Y2 = mean_score_trt;
X1 = visit;
X2 = visit;
DRAWSPACE = "DATAVALUE";
LINEPATTERN = "2"; /*dash line*/
LINECOLOR = "black";
END;
RUN;

FUNCTION LABEL Y1 X1 ANCHOR DRAWSPACE WIDTH TEXTSIZE Y2 X2 LINEPATTERN LINECOLOR
  text Placebo - Treatment equals: 1.6 8.9 1 left DATAVALUE 20 8 . .
  text Placebo - Treatment equals: -2.6 6.5 2 left DATAVALUE 20 8 . .
  text Placebo - Treatment equals: 4.7 7.75 3 left DATAVALUE 20 8 . .
  text Placebo - Treatment equals: -0.6 9.2 4 left DATAVALUE 20 8 . .
  text Placebo - Treatment equals: -2.3 9.85 5 left DATAVALUE 20 8 . .
  
line 9.7 1 DATAVALUE . . 8.1 1 2 black
line 5.2 2 DATAVALUE . . 7.8 2 2 black
line 10.1 3 DATAVALUE . . 5.4 3 2 black
line 8.9 4 DATAVALUE . . 9.5 4 2 black
line 8.7 5 DATAVALUE . . 11 5 2 black

Table 3: line annotations

The same can be achieved by using the %SGLINE macro while creating the annotation dataset:

DATA anno_macro;
  SET anno_macro(IN=_a) anno_macro(IN=_b);
  IF _a THEN DO;
  ...(TEXT FUNCTION)...
  END;
  IF _b THEN DO;
  %SGLINE(
    Y1 = mean_score_plac,
    Y2 = mean_score_trt,
    X1 = visit,
    X2 = visit,
    DRAWSPACE = "DATAVALUE",
    LINEPATTERN = "2", /*dash line*/
    LINECOLOR = "black")
  END;
RUN;

The same can be done within the template using a DRAWLINE statement for each line. In our example we need a separate DRAWLINE for each visit. Note here also that the syntax is different. And for instance to change the line pattern or colour we do it all within the LINEATTRS option. Say we have five visits then we need to loop 5 times and insert the DRAWLINE statement to the template. Say &&mean_score_plac_&i.. is the mean score for placebo on i-th visit and &&mean_score_trt_&i.. is the mean score for treatment on i-th visit, then the template creation looks like the below:

%MACRO _tmpl;
PROC TEMPLATE;
  DEFINE STATGRAPH Tmpl;
  BEGINGRAPH;
    LAYOUT OVERLAY;
      %DO i = 1 %TO 5;
      DRAWLINE Y1 = &&mean_score_plac_&i.. Y2 = &&mean_score_trt_&i.. X1 = &i.
      X2 = &i. / DRAWSPACE = DATAVALUE 
      LINEATTRS = (PATTERN = "2" /*dash line*/ COLOR = "black")
    %END;

    ...(DRAWTEXT)...

%END;
Figure 4: line annotations

Using the line annotation we can also change the pattern, thickness and colour of the line. We can also specify whether the annotation is drawn in front of or behind the graph and set its transparency level.

AN OVAL

If you want to draw an oval on your figure you just need to use the OVAL function. This function requires four variables, \textit{X1} and \textit{Y1} which specify the coordinates around which the oval is going to be drawn, and \textit{HEIGHT} and \textit{WIDTH}, which specify the height and width of the oval.

In our example we will highlight the maximum values for treatment and placebo drawing a solid oval over the scatter points which represent these values. For the placebo we will draw a blue oval and for the treatment a red oval. To do so we need to specify the \textit{FILLCOLOR} variable to define the desired colours. We will display solid ovals without borders so we need to set the \textit{DISPLAY} variable to \textit{FILLED}.

DATA anno;

*(TEXT FUNCTION)*...

\begin{verbatim}
    FUNCTION = "oval";
    X1 = &max_x_plac.;
    Y1 = &max_y_plac.;
    WIDTH=2;
    HEIGHT=2.5;
    DRAWSPACE = "DATAVALUE";
    FILLCOLOR="BLUE";
    DISPLAY = "FILLED";
    OUTPUT;

    FUNCTION = "oval";
    X1 = &max_x_trt.;
    Y1 = &max_y_trt.;
    WIDTH=2;
    HEIGHT=2.5;
    DRAWSPACE = "DATAVALUE";
    FILLCOLOR="RED";
    DISPLAY = "FILLED";
    OUTPUT;

RUN;
\end{verbatim}

\begin{tabular}{|l|l|l|l|l|l|l|l|l|l|l|}
\hline
FUNCTION & LABEL & X1 & Y1 & WIDTH & HEIGHT & DRAWSPACE & FILLCOLOR & DISPLAY & ANCHOR & WIDTHUNIT \\
\hline
oval & & 3 & 10.1 & 2 & 2.5 & DATAVALUE & BLUE & FILLED & & \\
oval & & 5 & 11 & 2 & 2.5 & DATAVALUE & RED & FILLED & & \\
\hline
\end{tabular}
Table 4: oval annotations

The same can be achieved by using the `%SGOVAL` macro while creating the annotation dataset:

```plaintext
DATA anno_macro;

... (TEXT FUNCTION) ...

%SGOVAL(
   X1 = &max_x_plac.,
   Y1 = &max_y_plac.,
   WIDTH=2,
   HEIGHT=2.5,
   DRAWSPACE = "DATAVALUE",
   FILLCOLOR="BLUE",
   DISPLAY = "FILLED")

%SGOVAL(
   X1 = &max_x_trt.,
   Y1 = &max_y_trt.,
   WIDTH=2,
   HEIGHT=2.5,
   DRAWSPACE = "DATAVALUE",
   FILLCOLOR="RED",
   DISPLAY = "FILLED")

RUN;
```

We can also change the template using the `DRAWOVAL` statement. Here the syntax is a bit different. The fill colour of an oval is defined via the `FILLATTRS` option enclosed in brackets. To specify that we want to display the oval without border we also need to specify the `DISPLAY` option enclosed in brackets.

```plaintext
PROC TEMPLATE;
   DEFINE STATGRAPH Tmpl;
   BEGINGRAPH;
      LAYOUT OVERLAY;

      ... (DRAWTEXT) ...

      DRAWOVAL x= &max_x_plac. y= &max_y_plac. width=2 height=2.5 /
                     DRAWSPACE=DATAVALUE
                     FILLATTRS=(COLOR=BLUE)
                     DISPLAY = (FILL);

      DRAWOVAL x= &max_x_trt. y= &max_y_trt. width=2 height=2.5 /
                     DRAWSPACE=DATAVALUE
                     FILLATTRS=(COLOR=RED)
                     DISPLAY = (FILL);

      ENDLAYOUT;
   ENDGRAPH;
RUN;
```
Using the oval annotation we can also change the pattern, thickness and colour of the border of an oval. We can also specify how the oval is rotated, whether the annotation is drawn in front of or behind the graph and set its transparency level.

A RECTANGLE

A rectangle can be drawn in a similar way to an oval. Again there are four required variables: X1 and Y1 for the coordinates and HEIGHT and WIDTH for the height and width. The only difference is that we can specify the ANCHOR variable which gives the anchor position of the annotation. CENTER is the default for ANCHOR.

Once again in our example we will highlight the maximum values for treatment and placebo but this time we will draw rectangles over the scatter points which represent these values. For the placebo we will draw a blue rectangle and for the treatment a red rectangle. To do so we need to specify the FILLCOLOR variable to define the colours we want. We will set the DISPLAY variable to FILLED to display a solid rectangle without border. We will leave the ANCHOR option with the default value assigned, so the given coordinates will be assigned to the middle of the rectangle.

DATA anno;
...(TEXT FUNCTION)...

    FUNCTION = "rectangle";
    X1 = &max_x_plac.;
    Y1 = &max_y_plac.;
    WIDTH=2;
    HEIGHT=2.5;
    DRAWSPACE = "DATAVALUE";
    FILLCOLOR="BLUE";
    DISPLAY = "FILLED";
    OUTPUT;

    FUNCTION = "rectangle";
    X1 = &max_x_trt.;
    Y1 = &max_y_trt.;
    WIDTH=2;
    HEIGHT=2.5;
    DRAWSPACE = "DATAVALUE";
    FILLCOLOR="RED";
    DISPLAY = "FILLED";
    OUTPUT;
RUN;

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>LABEL</th>
<th>X1</th>
<th>Y1</th>
<th>WIDTH</th>
<th>HEIGHT</th>
<th>DRAWSPACE</th>
<th>FILLCOLOR</th>
<th>DISPLAY</th>
<th>anchor</th>
<th>WIDTHUNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>rectangle</td>
<td>Maximum for Placebo</td>
<td>3</td>
<td>10.1</td>
<td>2</td>
<td>2.5</td>
<td>DATAVALUE</td>
<td>BLUE</td>
<td>FILLED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rectangle</td>
<td>Maximum for Treatment</td>
<td>5</td>
<td>11</td>
<td>2</td>
<td>2.5</td>
<td>DATAVALUE</td>
<td>RED</td>
<td>FILLED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>text</td>
<td>Maximum for Placebo</td>
<td>3</td>
<td>10.1</td>
<td>2</td>
<td>2.5</td>
<td>DATAVALUE</td>
<td>RED</td>
<td>FILLED</td>
<td>TOPRIGHT</td>
<td>DATA</td>
</tr>
<tr>
<td>text</td>
<td>Maximum for Treatment</td>
<td>5</td>
<td>11</td>
<td>2</td>
<td>2.5</td>
<td>DATAVALUE</td>
<td>RED</td>
<td>FILLED</td>
<td>TOPRIGHT</td>
<td>DATA</td>
</tr>
</tbody>
</table>

Table 5: rectangle annotations
The same can be achieved by using the %SGRECTANGLE macro while creating the annotation dataset:

DATA anno_macro;

...(TEXT FUNCTION)...

%SGRECTANGLE(
   X1 = &max_x_plac.,
   Y1 = &max_y_plac.,
   WIDTH=2,
   HEIGHT=2.5,
   DRAWSPACE = "DATAVALUE",
   FILLCOLOR="BLUE",
   DISPLAY = "FILLED")

%SGRECTANGLE(
   X1 = &max_x_trt.,
   Y1 = &max_y_trt.,
   WIDTH=2,
   HEIGHT=2.5,
   DRAWSPACE = "DATAVALUE",
   FILLCOLOR="RED",
   DISPLAY = "FILLED")

RUN;

If we want to change the template instead of creating an annotation dataset then we can use the DRAWRECTANGLE statement with a similar syntax to the DRAWOVAL statement. The fill colour of a rectangle is defined via the FILLATTRS option enclosed in brackets. To specify that we want to display a solid rectangle without border we also need to specify the DISPLAY option enclosed in brackets.

PROC TEMPLATE;
   DEFINE STATGRAPH Tmpl;
      BEGINGRAPH;
         LAYOUT OVERLAY;

            ...(DRAWTEXT)...

            DRAWRECTANGLE x= &max_x_plac. y=&max_y_plac. width=2 height=2.5 /
               DRAWSPACE=DATAVALUE
               FILLCOL=(COLOR=BLUE)
               DISPLAY = (FILL);

            DRAWRECTANGLE x= &max_x_trt. y=&max_y_trt. width=2 height=2.5 /
               DRAWSPACE=DATAVALUE
               FILLCOL=(COLOR=RED)
               DISPLAY = (FILL);

         ENDLAYOUT;
      ENDDGRAPH;
   END;
RUN;
Figure 6: rectangle annotations

Just like with the oval annotation we can also change the pattern, thickness and colour of the border of a rectangle. We can also specify how the rectangle is rotated, whether the annotation is drawn in front of or behind the graph and set its transparency level.

A POLYGON
The POLYGON function can be used if we want to create any geometric shape which has vertices. We need to specify X1 and Y1 which define the starting point of the polygon and at least two POLYCONT functions which define the location of the remaining vertices. The starting point and vertices are linked automatically to form a closed geometric shape. The visual attributes of the polygon are given in the POLYGON function and POLYCONT just inherits them.

In our example, as in the two previous examples, we will highlight the maximum values for treatment and placebo. This time we will draw triangles. For the placebo we will draw a blue triangle and for the treatment a red triangle. To do so we need to specify the FILLCOLOR variable which defines the colours. For a solid triangle without border we will set the DISPLAY variable to FILLED. We do not need to close the shape on our own by defining the last vertex with the coordinates of the starting point as this is done automatically.

DATA anno;
...(TEXT FUNCTION)...

FUNCTION = "polygon";
X1 = %sysevalf(&max_x_plac. - 0.05);
Y1 = %sysevalf(&max_y_plac. - 0.1);
DRAWSPACE = "DATAVALUE";
FILLCOLOR="BLUE";
DISPLAY = "FILLED";
OUTPUT;

FUNCTION = "polycont";
X1 = %sysevalf(&max_x_plac.);
Y1 = %sysevalf(&max_y_plac. + 0.1);
OUTPUT;

FUNCTION = "polycont";
X1 = %sysevalf(&max_x_plac. + 0.05);
Y1 = %sysevalf(&max_y_plac. - 0.1);
OUTPUT;

FUNCTION = "polygon";
X1 = %sysevalf(&max_x_trt. - 0.05);
Y1 = %sysevalf(&max_y_trt. - 0.1);
DRAWSPACE = "DATAVALUE";
FILLCOLOR="RED";
DISPLAY = "FILLED";
OUTPUT;

FUNCTION = "polycont";
X1 = %sysevalf(&max_x_trt.);
Y1 = %sysevalf(&max_y_trt. + 0.1);
OUTPUT;

FUNCTION = "polycont";
X1 = %sysevalf(&max_x_trt. + 0.05);
Y1 = %sysevalf(&max_y_trt. - 0.1);
OUTPUT;
RUN;

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>LABEL</th>
<th>X1</th>
<th>Y1</th>
<th>DRAWSPACE</th>
<th>FILLCOLOR</th>
<th>DISPLAY</th>
<th>ANCHOR</th>
<th>WIDTHUNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>polygon</td>
<td></td>
<td>2.95</td>
<td>10</td>
<td>&quot;DATAVALUE&quot;</td>
<td>&quot;BLUE&quot;</td>
<td>&quot;FILLED&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>polygon</td>
<td></td>
<td>3.05</td>
<td>10</td>
<td>&quot;DATAVALUE&quot;</td>
<td>&quot;BLUE&quot;</td>
<td>&quot;FILLED&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>polygon</td>
<td></td>
<td>4.95</td>
<td>10.2</td>
<td>&quot;DATAVALUE&quot;</td>
<td>&quot;RED&quot;</td>
<td>&quot;FILLED&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>polygon</td>
<td></td>
<td>5.05</td>
<td>10.9</td>
<td>&quot;DATAVALUE&quot;</td>
<td>&quot;RED&quot;</td>
<td>&quot;FILLED&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>text</td>
<td>Maximum for Placebo</td>
<td>3</td>
<td>10.1</td>
<td>&quot;DATAVALUE&quot;</td>
<td>&quot;RED&quot;</td>
<td>&quot;FILLED&quot;</td>
<td>TOPRIGHT</td>
<td>DATA</td>
</tr>
<tr>
<td>text</td>
<td>Maximum for Treatment</td>
<td>5</td>
<td>11</td>
<td>&quot;DATAVALUE&quot;</td>
<td>&quot;RED&quot;</td>
<td>&quot;FILLED&quot;</td>
<td>TOPRIGHT</td>
<td>DATA</td>
</tr>
</tbody>
</table>

Table 6: polygon annotations

The same dataset can be created using the %SGPOLYGON and %SGPOLYCONT macros.

DATA anno_macro;

...(TEXT FUNCTION)...

%SGPOLYGON(
    X1 = %sysevalf(&max_x_plac. - 0.05),
    Y1 = %sysevalf(&max_y_plac. - 0.1),
    DRAWSPACE = "DATAVALUE",
    FILLCOLOR="BLUE",
    DISPLAY = "FILLED"
)

%SGPOLYCONT(
    X1 = %sysevalf(&max_x_plac.),
    Y1 = %sysevalf(&max_y_plac. + 0.1)
)

%SGPOLYCONT(
    X1 = %sysevalf(&max_x_plac. + 0.05),
    Y1 = %sysevalf(&max_y_plac. - 0.1)
)

%SGPOLYGON(
    X1 = %sysevalf(&max_x_trt. - 0.05),
    Y1 = %sysevalf(&max_y_trt. - 0.1),
    DRAWSPACE = "DATAVALUE",
    FILLCOLOR="RED",
    DISPLAY = "FILLED"
)

%SGPOLYCONT(
    X1 = %sysevalf(&max_x_trt.),
    Y1 = %sysevalf(&max_y_trt. + 0.1)
)

%SGPOLYCONT(
    X1 = %sysevalf(&max_x_trt. + 0.05),
    Y1 = %sysevalf(&max_y_trt. - 0.1)
)
RUN;

Within the template the polygon is created by using the BEGINPOLYGON statement. Within that statement we specify the coordinates of the starting point and then specify the further vertices by nesting a series of DRAW statements within the BEGINPOLYGON block. Specification of the colour for the inside of a triangle is done via the FILLATTRS option enclosed in brackets. To specify that we want to display a solid triangle we also need to set the DISPLAY option to FILL enclosed in brackets. As we can see, the options for the BEGINPOLYGON statement are similar to those used for DRAWOVAL or DRAWRIRCTANGLE.

PROC TEMPLATE;
    DEFINE STATGRAPH Tmpl;
    BEGINGRAPH;
        LAYOUT OVERLAY;

    %SGPOLYGON(
        X1 = %sysevalf(&max_x_trt.);
        Y1 = %sysevalf(&max_y_trt. + 0.1);
        OUTPUT;
    )
    %SGPOLYCONT(
        X1 = %sysevalf(&max_x_trt. + 0.05);
        Y1 = %sysevalf(&max_y_trt. - 0.1);
        OUTPUT;
    )
    %SGPOLYCONT(
        X1 = %sysevalf(&max_x_trt. + 0.05);
        Y1 = %sysevalf(&max_y_trt. - 0.1);
        OUTPUT;
    )
    %SGPOLYCONT(
        X1 = %sysevalf(&max_x_trt. + 0.05);
        Y1 = %sysevalf(&max_y_trt. - 0.1);
        OUTPUT;
    )
    %SGPOLYCONT(
        X1 = %sysevalf(&max_x_trt. + 0.05);
        Y1 = %sysevalf(&max_y_trt. - 0.1);
        OUTPUT;
    )
    %SGPOLYCONT(
        X1 = %sysevalf(&max_x_trt. + 0.05);
        Y1 = %sysevalf(&max_y_trt. - 0.1);
        OUTPUT;
    )
    %SGPOLYCONT(
        X1 = %sysevalf(&max_x_trt. + 0.05);
        Y1 = %sysevalf(&max_y_trt. - 0.1);
        OUTPUT;
    )
RUN;

Within the template the polygon is created by using the BEGINPOLYGON statement. Within that statement we specify the coordinates of the starting point and then specify the further vertices by nesting a series of DRAW statements within the BEGINPOLYGON block. Specification of the colour for the inside of a triangle is done via the FILLATTRS option enclosed in brackets. To specify that we want to display a solid triangle we also need to set the DISPLAY option to FILL enclosed in brackets. As we can see, the options for the BEGINPOLYGON statement are similar to those used for DRAWOVAL or DRAWRIRCTANGLE.
Using the polygon annotation we can also change the pattern, thickness and colour of the border of a geometric shape. We can also specify whether the annotation is drawn in front of or behind the graph and set its transparency level.

A POLYLINE

The POLYLINE function is very similar to the POLYGON function except that we do not have to close the shape as a POLYLINE is just a series of connected lines. We need to specify X1 and Y1 which define the coordinates of the start of the polyline. As with the POLYGON function we need to specify at least one succeeding POLYCONT function which defines the location of the subsequent point, then a straight line is drawn from the previous point to that point. The visual attributes of the polyline are given in the POLYLINE function and POLYCONT just inherits them.

Say we had a washout period in our study between visits two and four. We will draw a brace above the visits 2, 3 and 4 as a polyline. Say &max_mean_score_washout is the maximum score for both placebo and treatment in the washout period. We will increase the thickness of the brace by setting the variable LINETHICKNESS to 5.

DATA anno;

... (TEXT FUNCTION) ...

FUNCTION = "polyline";
X1 = 2;
Y1 = %sysevalf(&max_mean_score_washout + 0.1);
DRAWSPACE = "DATAValue";
LINETHICKNESS = 5;
OUTPUT;
FUNCTION = "polycont";
X1 = 2;
Y1 = %sysevalf(&max_mean_score_washout. + 0.1);
OUTPUT;

FUNCTION = "polycont";
X1 = 3;
Y1 = %sysevalf(&max_mean_score_washout. + 0.3);
OUTPUT;

FUNCTION = "polycont";
X1 = 3;
Y1 = %sysevalf(&max_mean_score_washout. + 0.5);
OUTPUT;

FUNCTION = "polycont";
X1 = 3;
Y1 = %sysevalf(&max_mean_score_washout. + 0.3);
OUTPUT;

FUNCTION = "polycont";
X1 = 4;
Y1 = %sysevalf(&max_mean_score_washout. + 0.3);
OUTPUT;

FUNCTION = "polycont";
X1 = 4;
Y1 = %sysevalf(&max_mean_score_washout. + 0.1);
OUTPUT;

RUN;

Table 7: polyline annotations

Using the %SGPOLYLINE and %SGPOLYCONT macros we can create the annotation dataset more easily.

DATA anno_macro;

...(TEXT FUNCTION)...

%SGPOLYLINE(
   X1 = 2,
   Y1 = %sysevalf(&max_mean_score_washout. + 0.1),
   DRAWSPACE = "DATAVALUE",
   LINETHICKNESS = 5)

%SGPOLYCONT(
   X1 = 2,
   Y1 = %sysevalf(&max_mean_score_washout. + 0.3))

%SGPOLYCONT(
   X1 = 3,
   Y1 = %sysevalf(&max_mean_score_washout. + 0.3))

%SGPOLYCONT(
   X1 = 3,
   Y1 = %sysevalf(&max_mean_score_washout. + 0.5))

%SGPOLYCONT(
   X1 = 3,
   Y1 = %sysevalf(&max_mean_score_washout. + 0.3))
A polyline can also be created via the template by specifying the `BEGINPOLYLINE` statement. Within that statement we specify the coordinates of the starting point. The `DRAW` statements nested within the `BEGINPOLYLINE` block specify the further points. The thickness of the line is specified in the `LINEATTRS` option via the `THICKNESS` option enclosed in brackets.

```sas
PROC TEMPLATE;
  DEFINE STATGRAPH Tmpl;
  BEGINGRAPH;
  LAYOUT OVERLAY;
  ...(
      DRAW TEXT
  )...
  BEGINPOLYLINE x = 2 y = %sysevalf(&max_mean_score_washout. + 0.1) /
                  DRAWSPACE=DATAVALUE,
                  LINEATTRS=(THICKNESS=5);
  DRAW x = 2 y = %sysevalf(&max_mean_score_washout. + 0.3);
  DRAW x = 3 y = %sysevalf(&max_mean_score_washout. + 0.3);
  DRAW x = 3 y = %sysevalf(&max_mean_score_washout. + 0.5);
  DRAW x = 3 y = %sysevalf(&max_mean_score_washout. + 0.3);
  DRAW x = 4 y = %sysevalf(&max_mean_score_washout. + 0.3);
  DRAW x = 4 y = %sysevalf(&max_mean_score_washout. + 0.1);
  ENDPOLYLINE;
  ENDLAYOUT;
  ENDPGRAPH;
END;
RUN;
```

Figure 8: polyline annotations

Using the polyline annotation we can also change the pattern and colour of the line. We can also specify whether the annotation is drawn in front of or behind the graph and set its transparency level.

**AN IMAGE**

To add an image to the figure we need to specify the entire path to the image file as well as the file name. We do this by specifying the path within the `IMAGE` variable. Although this is the only required variable for the `IMAGE` function, this results in the image being displayed in the centre of the figure. To specify a location for the image we can provide coordinates using the `X1` and `Y2` variables. By default, the image is drawn in its actual size, however this can be changed by setting the `WIDTH` and/or `HEIGHT` variables.
Say we want to add images to label the treatment and placebo plot lines. We identify the values of the mean score at the last visit for placebo and treatment and store these values in the &mean_score_plac_5. and &mean_score_trt_5. macro variables respectively. To be sure that the image fits the graph we will increase the maximum offset in the template slightly and change the width of the figure and the dimension unit for that width. This we do via the WIDTH and WIDTHUNIT variables respectively. The figure is located outside of the range of values on the x-axis. Instead of specifying the DRAWSPACE variable for our coordinate we can easily specify the drawing space by defining it separately for each axis. We set X1SPACE to DATAPERCENT and, as in our previous examples, we keep Y1SPACE as DATAVALUE.

DATA anno;
  ATTRIB IMAGE length=$100;
  FUNCTION = "image";
  IMAGE = "C:\placebo.jpg";
  X1 = 107;
  Y1 = &mean_score_plac_5.;
  Y1SPACE = "DATAVALUE";
  X1SPACE = "DATAPERCENT";
  WIDTH = 14;
  WIDTHUNIT = "pixel";
  OUTPUT;

  FUNCTION = "image";
  IMAGE = "C:\treatment.jpg";
  X1 = 107;
  Y1 = &mean_score_trt_5.;
  Y1SPACE = "DATAVALUE";
  X1SPACE = "DATAPERCENT";
  WIDTH = 14;
  WIDTHUNIT = "pixel";
  OUTPUT;
RUN;

<table>
<thead>
<tr>
<th>IMAGE</th>
<th>FUNCTION</th>
<th>X1</th>
<th>Y1</th>
<th>Y1SPACE</th>
<th>X1SPACE</th>
<th>WIDTH</th>
<th>WIDTHUNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>C:\placebo.jpg</td>
<td>image</td>
<td>107</td>
<td>8.7</td>
<td>DATAVALUE</td>
<td>DATAPERCENT</td>
<td>14</td>
<td>pixel</td>
</tr>
<tr>
<td>C:\treatment.jpg</td>
<td>image</td>
<td>107</td>
<td>11</td>
<td>DATAVALUE</td>
<td>DATAPERCENT</td>
<td>14</td>
<td>pixel</td>
</tr>
</tbody>
</table>

Table 8: image annotations

As expected, there is an %SGIMAGE macro which can be used for the purposes of creating the annotation dataset.

DATA anno_macro;
  %SGIMAGE(IMAGE = "C:\placebo.jpg",
            X1 = 107,
            Y1 = &mean_score_plac_5.,
            Y1SPACE = "DATAVALUE",
            X1SPACE = "DATAPERCENT",
            WIDTH = 14,
            WIDTHUNIT = "pixel")
  %SGIMAGE(IMAGE = "C:\treatment.jpg",
            X1 = 107,
            Y1 = &mean_score_trt_5.,
            Y1SPACE = "DATAVALUE",
            X1SPACE = "DATAPERCENT",
            WIDTH = 14,
            WIDTHUNIT = "pixel")
RUN;

The syntax for the DRAWIMAGE statement is similar to that of the annotation for displaying an image. The difference is that instead of specifying the X1SPACE and Y1SPACE variables we specify the XSPACE and YSPACE options.

PROC TEMPLATE;
  DEFINE STATGRAPH Tmpl;
  BEGINGRAPH;
    LAYOUT OVERLAY;
    DRAWIMAGE "C:\placebo.jpg" /
      X = 107
      Y = &mean_score_plac_5.
      YSPACE = DATAVALUE
      XSPACE = DATAPERCENT
Figure 9: Image annotations

Using the image annotation we can also specify how the image is rotated, whether the annotation is drawn in front of or behind the graph and set its transparency level. We can also decide whether to draw a border around the image, and change the pattern, thickness and colour of the border.

CONCLUSION
Sometimes it is very difficult to enhance your graphic just by using proc template or the SG procedures. We may not want or be able to modify the template, so we cannot use the draw statements introduced in SAS 9.3 to add graphics elements. The annotation facility provides an easy way to add graphics elements to our figures which are not available using the SG procedures. Furthermore, the SG annotation macros introduced with Maintenance One of SAS 9.4 make it very easy to create an annotation dataset. Reading this paper and looking into the examples it introduces will help you become familiar with all the annotation functions and how to implement them.

REFERENCES

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

Konrad Zywno, M.Sc.
inVentiv Health Clinical
Joachimstaler Str. 10-12
10719 Berlin Germany
Email: konrad.zywno@inventivhealth.com

Authors: Konrad Zywno & Bartosz Kutyła
APPENDIX 1: Dummy dataset used for the examples in the paper

PROC FORMAT;
  VALUE _treat
  0   = 'placebo'
  1   = 'treatment' ;
RUN;

DATA work.input;
  ATTRIB treatment LENGTH = 8 FORMAT = _TREAT.;
  ATTRIB visit LENGTH = 8 FORMAT = best12.;
  ATTRIB mean_score LENGTH = 8 FORMAT = best12.;
  INPUT treatment 1-12 visit 14-25 mean_score 27-38;
DATALINES4;
  0   1   9.7
  0   2   5.2
  0   3  10.1
  0   4   8.9
  0   5   8.7
  1   1   8.1
  1   2   7.8
  1   3   5.4
  1   4   9.5
  1   5  11.0
;;;
RUN;