Paper 284-2011

Creating a half error bar using Graph Template Language

Dongsheng Yang, Cleveland Clinic Foundation, Cleveland, OH

ABSTRACT

Many statistical ODS graphics in SAS® 9.2 are generated by the Graph Template Language (GTL) using the predefined graph templates in the SAS® 9.2. With GTL, we can also create our own graph template. The half error bar (mean ± standard error) is a popular plot in displaying statistics. However, the major problems in creating a standalone half bar are to dynamically change the direction of the half bar to avoid overlay, display the number of subjects over time at the bottom of the plot, and minimize user inputs. In this paper, we show how to use GTL to create a standalone half error bar plot. We created a macro called Halfbar(). The user provides the Halfbar() macro with the data set name, group, x, y, and graph parameters such as axis labels and ticks, etc. The macro will output a nice-looking half bar plot. We also show if you specify a customized ODS style such as the journal style and call the macro, you can easily generate a high-quality plot for publication.

INTRODUCTION

Different from the traditional SAS/GRAPH® in SAS® 9.2, Graph Template Language (GTL) is a powerful tool to customize and build standalone ODS statistical graphs. The GTL syntax uses a well-structured “building-block” approach to define a graph: The DEFINE STATGRAPH statement in PROC TEMPLATE assigns a name to the new graph template definition; The DYNAMIC statement defines one dynamic variable which creates the title of the graph. The whole GTL syntax block expands from BEGINGRAPH to ENDGRAPH. The LAYOUT block expands from LAYOUT to ENDLAYOUT to define overlaid in the graph), and it is the main component of a graph template. Couple of things that we could do within the LAYOUT block include creating a graph overlaid by another graph, changing attributes of AXISOPTS according to our needs, and specifying the color and size of a symbol, text, and legend. Furthermore, the style templates control the overall appearance of an ODS graph such as graph elements (lines, markers, fonts and colors, fitted lines, and confidence bands etc) and attributes (line style, marker sizes etc). The graph templates control the layout and details of an ODS graph such as graph elements, attributes, etc.

An error bar (means ± Std or SE) is a popular statistic plot to show how the mean of a continuous outcome changes over time. It is very easy to use the SAS procedures to create a graph. However, as we known, it is always difficult to offer a generally flexible graph template to capture different requests such as fonts, colors or symbols etc from different users. One way to do this is to define a lot of input parameters in a SAS macro. But the disadvantages are: 1) for developers, SAS codes will become very long and complicated; 2) for users, it is hard to remember many macro parameters, so that users might can lose their interests to learn it.

In the SAS® 9.2, Graph Template Language (GTL) has provided new ways to overcome the above disadvantages. Firstly, with rich layout and plot types in the GTL, we can easily create desired graphs. Secondly, since both style and graph templates can be used to control the appearance of the graph, we can build a standalone graph template and let users choose their graph styles to modify appearances of the graph.

CREATING A HALF ERROR BAR

In this paper, we show how to use GTL to create a standalone half error bar plot step by step. We created a macro called Halfbar(). The user provides the HalfBar() macro with the dataset name, group, x, y and graph parameters such as axis labels and ticks, etc. We also show if you specify a customized ODS style such as the journal style, and call the macro, you can easily generate a high-quality plot for publication.

In the %HalfBar(), we have minimized input parameters from users. A sample call would look somehow like the following:

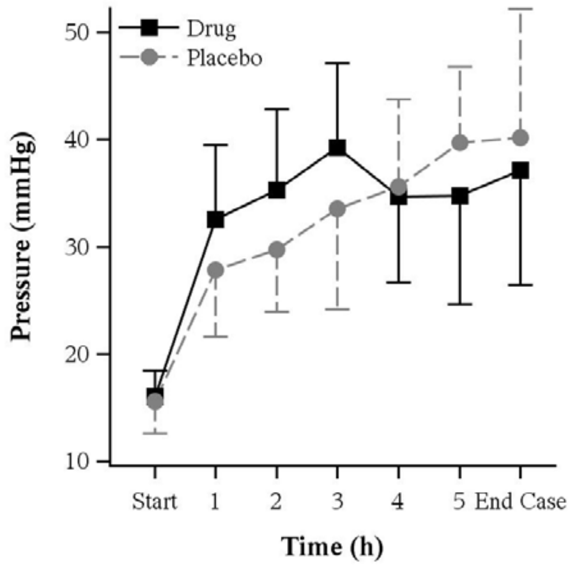
```
%HalfBar(     ds = _last_,          /* a data set                               */
              GroupVar = ,          /* a group variable: must be 0 and 1        */
              Xvar = ,              /* x variable usually your time variable    */
              Yvar = ,               /* y variable                              */
              Stat = std,           /* statistics: STD or Stderror              */
              Title = ,             /* figure title                             */
              XaxisList = ,         /* x axis options                           */
              YaxisList = ,         /* y axis options                           */
```

```
            DisplayN = No,        /* display number of subjects at time         */
            SubTitle = Number of Patients, /* sub-title for display N           */
            FigProp = 0.85 0.15,  /* the proportions of a plot and a table of N */
            FigStyle = Journal,   /* ODS style                                  */
            FigHeight = 480,      /* figure height                              */
            FigWidth   = 640,     /* figure width                               */
            FigDpi     = 150,     /* figure resolution                          */
            FigType    = JPG,     /* figure type                                */
            FigPath   = ./,       /* the folder to store a figure               */
            FigName   = MeanStd   /* figure name                                */
      );
```

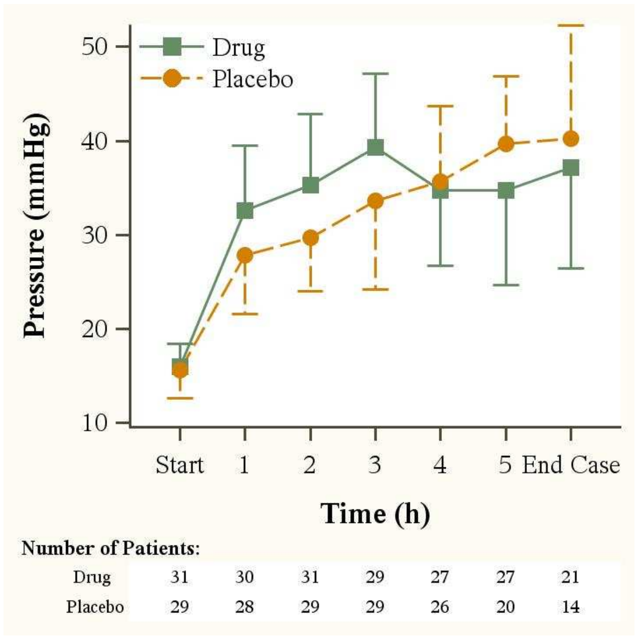
After called the macro, the mean ± Std plot has been generated (Figure 1). By changing the macro parameters (i.e. styles, display N or not), users can easily get a desired graph.

Figure 1. a half bar plot

(a) Journal style



(b) Analysis style



PROGRAMMING STRATEGY STEP-BY-STEP

The main steps include:

1. Checking the macro parameters.
2. Manipulate the raw data set to get a final data with statistics or other generated variables.
3. Create the standalone graph template.
4. Modify the style template to control the overall appearance of a graph.
5. Set up the ODS Destination and ODS graphic statement.
6. Run the SGRENDER procedure to associate the appropriate data with the template.

1 CHECKING THE MACRO PARAMETERS

It is always good to check user's inputs and returned warning messages in the log window in order to prevent unexpected errors. To achieve this, we first define a Boolean macro variable which has value FALSE if there are no errors and TRUE if there are error messages. If an error is found, the programming will be stopped and the error message will print to the log window.

For example:
```
%let __Err=0;
%if %length(&GroupVar)=0 %then %do;
       %put ERROR: NO GROUP VARIABLE SUPPLIED;
       %let __Err=1;
%end;
%if &__Err %then %do;
       data _null_;
              abort;
       run;
%end;
```

2. MANIPULATE DATA

Our raw repeated measurements included the treatment or placebo, the continuous outcome pressure, and time during surgery for the above figure 1. Here the group variable must be coded as 0 or 1. Since only one dataset is allowed to pass onto GTL, if we want to add more details such as points, lines, arrows and plots to a ODS statistical graph, we always need to manipulate our data object in the data step. To create a half bar, we have to get statistics for the treatment and placebo groups first (i.e. mean, Std, Stderror, or N). Another very important thing is to create a flag variable having values 1 or -1 to indicate larger group means at each point. If up = 1, the direction will be up. If flag = -1, then the direction will be down. By doing this, we can dynamically change the direction of the half bar to avoid overlay in the graph template. The following SAS codes were used to create a final data which would be passed to the SGSENDER procedure.

```
proc summary data = &ds. nway;
       class &groupvar. &xvar.;
       var &yvar.;
       output out = _temp(drop = _type_ _freq_)
                    mean = mean_y &stat. = y_&stat. n = N;
run;

(omit some SAS codes ··········)
```

Below is a portion of the final data we generated from the data step. The new variable gp1_mean is used for the treatment group and gp2_mean for the placebo group in the scatter plots, respectively.

```
  Treat   Time     Mean_y     y_std    N      gp1_mean      gp2_mean    Up

   Drug      1    16.0323    2.4150   31       16.0323             .     1

Placebo      1    15.5862    2.9462   29             .       15.5862    -1
```

3. BUILD STANDALONE GRAPH TEMPLATE

We used the graph template language (GTL) to create our standalone graph template by giving specific plot layouts (such as lattice or overlays), plot types (such as scatter plots and series plots), text elements (such as titles, footnotes, and insets), and other plot features (such as lines, marker symbols, and colors). However, the rule to build a standalone graph template is to avoid hard-coding attributes of lines, marker symbols or fonts (i.e. font sizes = 14pt, line thickness = 2 px) in it. Alternatively, the best choice is always to control the attributes of a graph by using the style template. We explain the whole graph template structure here in details:

For the layout of a figure, we use a logical condition to control whether we want to display number of subjects over time in the figure or not. Otherwise, only one layout is used. If it is true, there will be two panels in which the information of number of subjects will be combined into a table below the plot (Figure 1). Then we use a new layout—a LAYOUT LATTICE to stack two panels vertically, one for the plot and one for the number information. Using ROWWEIGHTS = (.85 .05 .10), the plot on top occupies 85% of the display, the sub title and number information in the second panel occupy 5% and 10%, respectively. The option COLUMNDATARANGE=UNIONALL is used to create a common axis across the two panels.

```
%if ("%upcase(&displayn.)" = "Y") %then %do;
layout lattice /
             rows=3 columns=1
             columndatarange=unionall
```

```
                    rowweights=(&__prop1 .05 &__prop3.);
        %end;
```

For the plot (in the upper panel), we use an overlay layout to draw two scatter plots and one series plot in it.

```
        layout overlay/ xaxisopts = ( &xaxislist.) yaxisopts = ( &yaxislist.); ❶

                scatterplot x = &xvar y = gp1_mean/

                        markerattrs   = (color = white )
                        errorbarattrs = graphdata1 ❷
                        yerrorlower   = eval(gp1_mean + up*y_&stat); ❸

                scatterplot x = &xvar y = gp2_mean/
                        markerattrs   = (color = white )
                        errorbarattrs = graphdata2 ❷
                        yerrorlower   = eval(gp2_mean + up*y_&stat); ❸

                seriesplot x = &xvar y = mean_y / ❹

                        group = &groupvar. display = all
                        name = "series";

                discretelegend "series"/ ❺

                        across=1 location=inside
                        autoalign=( topleft topright) border = false;
        endlayout;
```

❶ Use macro parameters to assign attributes of the axis such as label, fonts, ticks etc.

❷ Use the elements of the style template — GraphData1 or GraphData2 to control attributes of error bars in the two groups.

❸ Control the bar direction up or down at each time point.

❹ Display lines and markers for two groups.

❺ Show legends

For the table (the lower panel), we use two overlay layouts to draw the title and table. The text attributes have been controlled by the GraphFootnoteText and GraphDataText elements of the style template.

```
        %if ("%upcase(&DisplayN.)" = "Y") %then %do;
                layout overlay;
                        entry halign=left textattrs=GraphFootnoteText "&subtitle.";
                endlayout;

                layout overlay / xaxisopts=(display=none);

                        blockplot x= &xvar block=N /
                                class=&groupvar.
                                repeatedvalues=true display=(label values)
                                valuehalign=start valuefitpolicy=truncate
                                labelposition=left labelattrs=GraphDataText
                                valueattrs=GraphDataText
                                includemissingclass=false;
                endlayout;
        endlayout;
        %end;
```

As you see, we did not assign any attributes of lines, symbol, or fonts by hard-coding inside our standalone template. We tried to control them by external style template which can be customized by a user.

4. CUSTOMIZE ODS STYLE TEMPLATE

The following PROC TEMPLATE code shows the style template that we defined based on the default SAS journal style template. We controlled graph attributes such as font types, font sizes, axis lines, boundary, and lines etc. by the style. You can read SAS Graphical style elements in the SAS/GRAPH(R) 9.2: Graph Template Language User's

Guide, Second Edition to learn the detailed definitions of elements and their attributes as well as how to set up them. One thing we need to point out here is that, for group data, sometimes ❾ and ❿ have to be used together to fully control line types and thickness, marker size and types since each one controls parts of graph attributes. After you set up your desired template, you can save the customized style template into your own library.

```
ods path work.grafttemp(update)  ❶
         sashelp.tmplmst(read); ❷
proc template ;
define style styles.myErrorBarstyle; ❸
       parent = styles.Journal;       ❹
       style  GraphFonts from GraphFonts / ❺
              'GraphDataFont'      = ("<MTserif>, Times New Roman", 10pt)
              'GraphUnicodeFont'   = ("<MTserif>, Times New Roman", 14pt)
              'GraphValueFont'     = ("<MTserif>, Times New Roman", 14pt)
              'GraphLabelFont'     = ("<MTserif>, Times New Roman", 16pt, bold)
              'GraphFootnoteFont'  = ("<MTserif>, Times New Roman", 11pt, bold)
              'GraphTitleFont'     = ("<MTserif>, Times New Roman", 18pt, bold)
                    ;

       style GraphAxisLines       /LineThickness = 2px; ❻
       style GraphWalls           /LineThickness = 2px FrameBorder = off; ❼
       style GraphError           /LineThickness = 2px; ❽
       style GraphDataDefault     /LineThickness = 2px MarkerSize =14px; ❾
       style graphdata1    / markersymbol = "squarefilled" LineStyle = 1 ❿
                                color = black ContrastColor = black;
       style graphdata2    / markersymbol = "circlefilled" LineStyle = 4 ❿
                                Color = gray ContrastColor = gray;
end;
run;
```

We will make the following changes:

❶ Save the customized style template into work library

❷ Default template library for ODS statistical graphics. This template will be invoked if the first one does not exist

❸ Assign a name to the new style

❹ Use SAS default Journal style as its parent style

❺ Control fonts or font sizes for axis labels, title, footnote, text, axis tick values and legend entries, etc.

❻ Control the line thickness of axis lines

❼ Control graph boundaries

❽ Control error bar line thickness for two groups

❾ Control series line thickness and marker sizes

❿ Control group 1 and 2 marker types, line styles and colors

5.ODS DESTINATION STATEMENT

The ODS destination statements will indicate output directory and the type of an image. ODS HTML, ODS LISTING, ODS PDF, ODS PS, and ODS RTF are often used here.

```
ods   listing
             style     = &figstyle    ❶
             image_dpi =  &figDpi     ❷
             gpath     = "&FigPath"; ❸
```

❶ Specify the style we defined

❷ Specify the dots per inch (DPI), which is the image resolution for a graphical output

❸ Specify the directory where the graphics files are saved

6. ODS GRAPHICS STATEMENT

The ODS GRAPHICS statement enables ODS to create graphics

```
ods graphics on/reset                                       ❶
                  imagefmt  = jpg                           ❷
                  imagename = "myhalfbar"                   ❸
                  border    = off                           ❹
                  height    = 6.5in                         ❺
                  width     = 6.5in                         ❻
                  scale     = on;                           ❼
```

❶ Reset ODS GRAPHICS options to their default settings

❷ Specify type of the image file

❸ Specify the image file name

❹ Specify whether to draw the graph with a border

❺ Specify the height of the graph

❻ Specify the width of the graph

❼ Specify whether the fonts and symbol markers are scaled proportionally with the size of the graph

7.ODS RUN THE SGRENDER PROCEDURE

The SGRENDER procedure produces a graph from an input SAS data set and an ODS graph template:

```
proc sgrender data = _temp5 template = "mygraph.Errorbar"; ❶
       dynamic       title="&title."; ❷
run;
```

❶ Call the graph template we defined

❷ Pass dynamical parameters

CONCLUSION

Graph template language in SAS 9.2 provides rich layouts and plots. By controlling graph elements and their attributes externally by the style template, it is very easy to build a standalone graph template to create high quality of graphs for different users.

REFERENCES

■ Cartier, J. (2006), “A Programmer’s Introduction to the Graphics Template Language,” Proceedings of the Thirty-first Annual SAS Users Group International Conference. Cary, NC: SAS Institute Inc.

■ SAS Institute Inc. SAS/GRAPH SAS®9.2: Graph Template Language User's Guide

■ SAS Institute Inc. SAS/GRAPH ® 9.2: Graph Template Language Reference

■ SAS Institute Inc. SAS/STAT® 9.2 User's Guide

CONTACT INFORMATION

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

Name: Dongsheng Yang
Enterprise: Cleveland Clinic
City, State ZIP: Cleveland, OH 44195
Work Phone: (216) - 6365418
E-mail: yangd@ccf.org

Web: http://www.lerner.ccf.org/qhs/

APPENDIX

```
%macro HalfBar( ds = _last_,      /* a data set                                  */
             GroupVar = ,          /* a group variable: must be 0 and 1           */
             Xvar = ,              /* x variable usually your time variable       */
             Yvar = ,              /* y variable                                  */
             Stat = std,           /* statistics: STD or Stderror                 */
             Title = ,             /* figure title                                */
             XaxisList = ,         /* x axis options                              */
             YaxisList = ,         /* y axis options                              */
             DisplayN = No,        /* display number of subjects at time          */
             SubTitle = Number of Patients, /* sub-title for display N            */
             FigProp = 0.85 0.15,  /* the proportions of a plot and a table of N */
             FigStyle = Journal,   /* ODS style                                   */
             FigHeight = 480,      /* figure height                               */
             FigWidth   = 640,     /* figure width                                */
             FigDpi     = 150,     /* figure resolution                           */
             FigType    = JPG,     /* figure type                                 */
             FigPath   = ./,       /* the folder to store a figure                */
             FigName   = MeanStd   /* figure name                                 */
       );
%*-------------------------------------------------------;
%* part 1:                                               ;
%*     (a) Check for proper SAS version 9.2 for a macro  ;
%*     (b) check imput parameters                        ;
%*-------------------------------------------------------;

%* title: Check for proper SAS version for a macro;
%if %sysevalf(&sysver < 9.2) %then %do;
    %put ERROR: This program requires SAS Version 9.2 or later. It cannot run in
&sysver.;
    data _null_;
       abort;
    run;
       %end;

%*Parameter Checking the Macro;
   %let __Err=0;

   %if %length(&GroupVar)=0 %then %do;
       %put ERROR: NO GROUP VARIABLE SUPPLIED;
       %let __Err=1;
   %end;
   %if %length(&xvar)=0 %then %do;
       %put ERROR: NO X VARIABLE SUPPLIED;
       %let __Err=1;
   %end;

    %if %length(&yvar)=0 %then %do;
       %put ERROR: NO Y VARIABLE SUPPLIED;
       %let __Err=1;
   %end;

   %if %upcase(&stat) ne STD and %upcase(&stat) ne STDERR %then %do;
       %put ERROR: Statistics must be Std or Stderr;
       %let __Err=1;
   %end;
```

```
    %if %upcase(&DisplayN) ne Y and %upcase(&DisplayN) ne N %then %do;
        %put ERROR: DisplayN must be Y or N;
        %let __Err=1;
    %end;

    %if %upcase(&DisplayN) eq Y %then %do;
              %let __prop1 = %scan(&figprop.,1,' ');
              %let __prop2 = %scan(&figprop.,2, ' ');
              %let __sumprop = %sysevalf(&__prop1. + &__prop2.);
              %if %sysevalf(&__sumprop ^= 1) %then %do;
              %put &__prop1. + &__prop2. = &__sumprop. ERROR: Sum of FigProp must be 1;
              %let __Err=1;
              %end;
    %end;

    %if &__Err %then %do;
        data _null_;
           abort;
        run;
    %end;

%*-----------------------------------------------------------------;
%* part 2:                                                         ;
%*     (a) summary statistics based on group and time var          ;
%*     (b) create an indicator to detect bar should be up or down ;
%*-----------------------------------------------------------------;

proc summary data = &ds. nway;
       class &groupvar. &xvar.;
       var &yvar.;
       output out = _temp(drop = _type_ _freq_)
                    mean = mean_y &stat. = y_&stat. n = N;
run;

proc sort data = _temp;
       by &xvar &groupvar;
run;

proc sql;
       create table _temp_1 as
       select a.*,b.mean_y as mean_2,b.&groupvar as treat_1
       from _temp as a, _temp as b
       where a.&xvar = b.&xvar
       order by &xvar, &groupvar.;
quit;

data _temp_2;
       set _temp_1;
       if &groupvar = treat_1 then delete;

       if &groupvar = 1 then gp1_mean = mean_y;
       else if &groupvar = 0 then gp2_mean = mean_y;
       if round(mean_y,0.000001) >= round(mean_2,0.000001) then up= 1;
       else up = -1;
       keep &groupvar &xvar mean_y y_&stat. gp1_mean gp2_mean up n;
run;

proc sort data = _temp_2;
       by descending &groupvar &xvar;
run;


%*-----------------------------------------------------------------;
%* part 3:                                                         ;
%*     (a) bulid a standalone graph template                       ;
%*-----------------------------------------------------------------;
```

```
ods path work.grafttemp(update) sashelp.tmplmst(read);
proc template ;
       define statgraph mygraph.errorbar;
              begingraph;
              dynamic title;
       entrytitle title ;

              %if ("%upcase(&displayn.)" = "Y") %then %do;
                     %let __prop3 = %sysevalf(&__prop2 - 0.05);
                     layout lattice /  rows=3 columns=1 columndatarange=unionall
                            rowweights=(&__prop1 .05 &__prop3.);
              %end;

              layout overlay/ xaxisopts = ( &xaxislist.) yaxisopts = ( &yaxislist.);
                     scatterplot x = &xvar y = gp1_mean/
                            markerattrs  = (color = white )
                            errorbarattrs      = graphdata1
                            yerrorlower  = eval(gp1_mean + up*y_&stat);

                     scatterplot x = &xvar y = gp2_mean /
                            markerattrs  = (color = white )
                            errorbarattrs   = graphdata2
                            yerrorlower  = eval(gp2_mean + up*y_&stat);

                     seriesplot x = &xvar y = mean_y /
                            group = &groupvar. display = all  name = "series";

                     discretelegend "series"/
                            across=1 location=inside autoalign=( topleft topright)
                     border = false;
              endlayout;

              %if ("%upcase(&DisplayN.)" = "Y") %then %do;
                     layout overlay;
                            entry halign=left textattrs=GraphFootnoteText "&subtitle.";
                     endlayout;

                     layout overlay / xaxisopts=(display=none);
                       blockplot x= &xvar block=N / class=&groupvar.
                            repeatedvalues=true display=(label values)
                            valuehalign=start valuefitpolicy=truncate
                            labelposition=left labelattrs=GRAPHDATATEXT
                            valueattrs=GraphDataText
                            includemissingclass=false;
              endlayout;
              endlayout;
%end;
       endgraph;
end;
run;
quit;

%*-----------------------------------------------------------------;
%* part 4:                                                         ;
%*     (a) run a graph template and output a figure                ;
%*-----------------------------------------------------------------;

ods listing     style      = &figstyle
                image_dpi =  &figDpi
                 gpath      = "&FigPath";

ods graphics on/ reset border= off
                 imagefmt = &Figtype.
                  imagename = "&FigName"
                 HEIGHT = &FigHeight
                 width  = &FigWidth
                 scale = on;
```

```
proc sgrender data = _temp_2 template = "mygraph.Errorbar";
  dynamic title="&title.";
run;

ods graphics off;
ods path reset;

%mend;

*----------------------------------------------------------------;
* part 5:                                                        ;
*      (a) users defined their own style template                ;
*----------------------------------------------------------------;
ods path work.grafttemp(update) sashelp.tmplmst(read);

proc template ;
define style styles.myErrorBarstyle;
       parent = styles.Journal;
       style  GraphFonts from GraphFonts /
                     'GraphDataFont'     = ("<MTserif>, Times New Roman", 10pt)
                     'GraphUnicodeFont'  = ("<MTserif>, Times New Roman", 14pt)
                     'GraphValueFont'    = ("<MTserif>, Times New Roman", 14pt)
                     'GraphLabelFont'    = ("<MTserif>, Times New Roman", 16pt, bold)
                     'GraphFootnoteFont' = ("<MTserif>, Times New Roman", 11pt, bold)
                     'GraphTitleFont'    = ("<MTserif>, Times New Roman", 18pt, bold)
                     ;

       style GraphAxisLines      / linethickness = 2px;
       style GraphWalls          / LineThickness = 2px FrameBorder = off;
       style GraphError          / LineThickness = 2px;
       style GraphDataDefault    / LineThickness = 2px MarkerSize =14px;
       style graphdata1          / Markersymbol = "squarefilled" LineStyle = 1
                                   Color = black ContrastColor = black;

       style graphdata2          / Markersymbol = "circlefilled" LineStyle = 4
                                   Color = gray ContrastColor = gray;
end;
run;


*----------------------------------------------------------------;
* part 6:                                                        ;
*      (a) a sample to call macro                                ;
*----------------------------------------------------------------;

%halfbar(    Ds = mydata,
            GroupVar = treat,
             Xvar = time,
             Yvar = y,
             Stat = std,
             Title = ,
             XaxisList =  offsetmin = 0.1 offsetmax = 0.08
                           display = (label line ticks tickvalues)
                          label = 'Time (h)'
                          type = linear
                          LINEAROPTS = ( viewmin = 1 viewmax= 7
                                        tickvaluelist = (1 2 3 4 5 6 7)
                          tickdisplaylist = ('Start' '1' '2' '3' '4' '5' 'End Case'))
                          ,

            YaxisList =  offsetmin = 0.01 offsetmax = 0.01
                          display = (label line ticks tickvalues)
                          label = 'Pressure (mmHg)'
                          type = linear
                          LINEAROPTS = (viewmin = 10  viewmax= 52
                          tickvaluelist = (10 20 30 40 50))
                          ,
             DisplayN = y,
             SubTitle = Number of Patients:,
             FigProp = 0.85 0.15,
             FigStyle = myErrorBarstyle,
             FigHeight =  650,
             FigWidth   = 650,
             FigDpi     = 150,
             FigType    = JPG,
             FigPath   = H:\Temp,
             FigName   = MeanStd_n
);
```