

Macros for creating a custom report of figures

Laura Williams, CROS NT LLC

ABSTRACT

Often, with clinical studies, a report that includes both tables and figures can be preferred. PROC REPORT can be used to place multiple images in a document, such as an RTF file, in combination with summary tables or other text. First, a set of macros have been developed to control the appearance of figures within our organization, using a single template with GTL (Graph Template Language). Plus, the macros also control the file type, size, and resolution of the ODS output. Second, another set of macros have been developed to join these figures into an RTF file and populate the appropriate titles and footnotes. This second set of macros is a joint set of macros that can also produce tables. Therefore, we are able to create reports in a certain style that have a consistent appearance regardless of who programs the analysis output.

INTRODUCTION

Large clinical studies may have multiple statisticians and statistical programmers working to create statistical analysis output. In this situation, it is helpful to use macros to create output with a consistent appearance. The first macros presented in this paper use PROC TEMPLATE to create a custom style to be used on all figures for a specific clinical report. The user can customize certain aspects of the individual figures (e.g. color versus black and white image, marker types, line types), while others settings should be standardized for the study (e.g. font, font size, maximum image size, file type, file naming convention). The second set of macros presented in this paper use PROC REPORT to take the image files of the figures and print them into an RTF report. This allows both table or text output and figure output to be combined in the same file. The macro prints single page or multipage figures and adds titles and footnotes to the corresponding page(s) in the output document. The macros are intended to be used in two programs: one to create a figure or figures and another to join several figures together.

CREATING IMAGE FILES

The first step is to create image files of graphics that are to be included in a report. For this paper, macros are used to ensure that all figures have a consistent style and appearance, as well as the appropriate dimensions and file type (i.e. jpg or png). The figure programs follow this general layout:

1. Prepare the data to be graphed
2. Set up the style using PROC TEMPLATE
3. Specify the ODS destination
4. Produce the figure
5. Close the ODS destination

CREATING A STYLE WITH PROC TEMPLATE

SAS® has many built in styles which may be sufficient for most reports, however custom styles can be created with PROC TEMPLATE, using the Graph Template Language (GTL). One reason to create a custom style is to control the colors, marker attributes, and line attributes SAS uses in graphics. For example, the default colors can be difficult to distinguish when an image is printed in black and white, therefore a custom template can assign different colors to be used. Another benefit to creating your own custom style, is that rather than cycling through different marker and line styles for different groups, you can assign the same marker and line styles for each group. These, and many other options, can be accomplished by defining a custom style with PROC TEMPLATE. The macro **%FigTemplate** defines the style. In this macro, the options that can be controlled by the user are:

1. Black & white vs color figure

2. All the lines with the same attributes vs different attribute per line (grouped data)
3. All the markers with the same attributes vs different attribute per marker (grouped data)

The code for **%FigTemplate** is given below:

```
/*-----*\
| macro: FigTemplate                                     |
| usage: Set up the style for a figure                   |
| parameters: color = (set to Y for color, N for B&W)   |
|               sameline = (set to Y for all groups to  |
|                           have the same line style, N  |
|                           for different line styles per |
|                           group)                       |
|               samemark = (set to Y for all groups to   |
|                           have the same marker style,  |
|                           N for different marker styles |
|                           per group)                   |
\*-----*/

/*set the defaults as N*/
%macro FigTemplate(color=N, sameline=N, samemark=N);

/*line and marker attributes for non-grouped graph*/
%let line_style=%str(LineStyle=1 LineThickness=1px);
%let mark_style=%str(MarkerSymbol="CircleFilled" MarkerSize=5px);

/*line attributes for a grouped graph*/
%let line_style1=%str(LineStyle=1 LineThickness=1px);
%let line_style2=%str(LineStyle=2 LineThickness=1px);
/*etc. up to line 12*/

/*marker attributes for a grouped graph*/
%let mark_style1=%str(MarkerSymbol="CircleFilled" MarkerSize=5px);
%let mark_style2=%str(MarkerSymbol="DiamondFilled" MarkerSize=5px);
/*etc. up to marker 12 */

/*Use ODS path statement if you want to save your template to a
directory besides the default - in this case, we are saving to
work.fig */
ODS path sasuser.template(UPDATE) work.fig(UPDATE);

proc template;
define style fig / STORE = work.fig;
/*Use an existing style as a starting point*/
parent = styles.printer;

/*Assign attributes in color*/
%if &color=Y %then %do;
/*For a non-grouped graph*/
class GraphDataDefault /
    Color=cx13478C
    ContrastColor=cx4D7EBF
    &line_style &mark_style
    StartColor=cx90B0D9 /** light color **/
    NeutralColor=cxFFFFF ** white **/
    EndColor=cx4F4F4F /** dark color **/
;

```

```

/*For a grouped graph, SAS cycles through GraphData1,
GraphData2, etc. By default, styles have 12 GraphData
classes and if you have more than 12 groups SAS will
return to GraphData1. You can set more than 12 GraphData
classes in your custom style. A %do loop can be used here.*/
class GraphData1 /
    Color=cx13478C
    ContrastColor=cx4D7EBF
    %if &sameline=Y %then %do;
        &line_style
    %end;
    %else %do;
        &line_style1
    %end;
    %if &samemark=Y %then %do;
        &mark_style
    %end;
    %else %do;
        &mark_style1
    %end;
;

/* etc. up to GraphData12 */
%end;

/*Assign attributes in black & white */
%else %do;

    /*Repeat the previous section, except with
    greyscale colors*/

%end;

/*The fonts of the graph: if you prefer, these can also be
controlled with a macro parameter*/
style graphfonts from graphfonts /
    'GraphAnnoFont' = ("Times, <MTserif>",10pt)
    'GraphTitle1Font' = ("Times, <MTserif>",10pt)
    'GraphTitleFont' = ("Times, <MTserif>",10pt)
    'GraphFootnoteFont' = ("Times, <MTserif>",10pt)
    'GraphLabelFont' = ("Times, <MTserif>",10pt)
    'GraphLabel2Font' = ("Times, <MTserif>",10pt)
    'GraphValueFont' = ("Times, <MTserif>",9pt)
    'GraphUnicodeFont' = ("<MTserif-unicode>",9pt)
    'GraphDataFont' = ("Times, <MTserif>",7pt);

end;
run;
%mend FigTemplate;

```

This macro represents a relatively simple style, with many attributes copied from a pre-existing SAS style. The SAS ODS Graphics: Procedure's Guide gives some recommended styles for statistical graphics (see the section "Controlling the Appearance of Your Graphs"). Also, see the recommended reading for documentation useful for creating custom styles with GTL.

PRODUCING AN IMAGE FILE WITH ODS

The next step to create the image file is to set up the ODS destination. In this example, we want to create a png file, so the listing destination is used in conjunction with setting the file type in the ODS GRAPHICS statement. A macro is also used within this step to keep the file type, size, naming conventions, etc. consistent between outputs. This macro is called **%StartFig** and the code is below:

```

/*-----*\
| macro: StartFig                                     |
| usage: Set up the ODS destination for a figure     |
| parameters: path = (path of the output location)   |
|              name = (filename of the image)        |
|              height = (image height)               |
|              width = (image width)                 |
|              units = (units for height and width)  |
|              page = (the current page number, for  |
|                      multipage figures)            |
|-----*\
%macro StartFig(path=, name=, height=, width=, units=, page=);
    /* Note: you may want path= and name= to be pre-determined
    by your organization's file structure and/or the Statistical
    Analysis Plan*/

    /* if height, width, or units are not assigned, assign some defaults */
    %if &units= or &height= or &width= %then %do;
        /* assign the defaults */
    %end;

    /** maximum dimensions *****/
    /*
    /* It's a good idea to check the height and width values the user
    /* inputs to make sure they will fit on a page of your document,
    /* considering things like margins, titles, footnotes, orientation.
    /*
    /*
    /*****

        /*Use the style FIG we defined in %FigTemplate*/
    ods listing gpath = "&path" image_dpi=300 style=FIG;
    %if &page ne %then %do;
        ods graphics / reset height=&height &units width=&width &units
            noborder imagename="&name.__&page" imagefmt=png;
    %end;
    %else %do;
        ods graphics / reset height=&height &units width=&width &units
            noborder imagename="&name" imagefmt=png;
    %end;

%mend StartFig;

```

The **%StartFig** macro sets some parameters that the user cannot change (e.g. image format will be png). To create the image files using **%FigTemplate** and **%StartFig**, the program will generally follow this convention:

```

/* some code to prepare the data for plotting */

%FigTemplate(color=Y, sameline=N, samemark=N);

%StartFig(path=%bquote(C:\SESUG 2017), name=F1, height=4, width=6,

```

```

units=in);

proc sgplot data=mydata;
  scatter x=tpt y=mean / group=grpvar yerrorupper=upper yerrorlower=lower
                                name='scatter' groupdisplay=cluster
                                clusterwidth=0.5;
  series x=tpt y=mean / group=grpvar markers name='series'
                                groupdisplay=cluster clusterwidth=0.5;
  xaxis values=(0 to 4 by 1);
  keylegend 'series' / title='Treatment Group';
run;

```

This example creates a grouped series plot using the SGPLOT procedure. The image is 4" x 6", in color, with different line styles and different markers for each group. The output is shown in Figure 1 below.

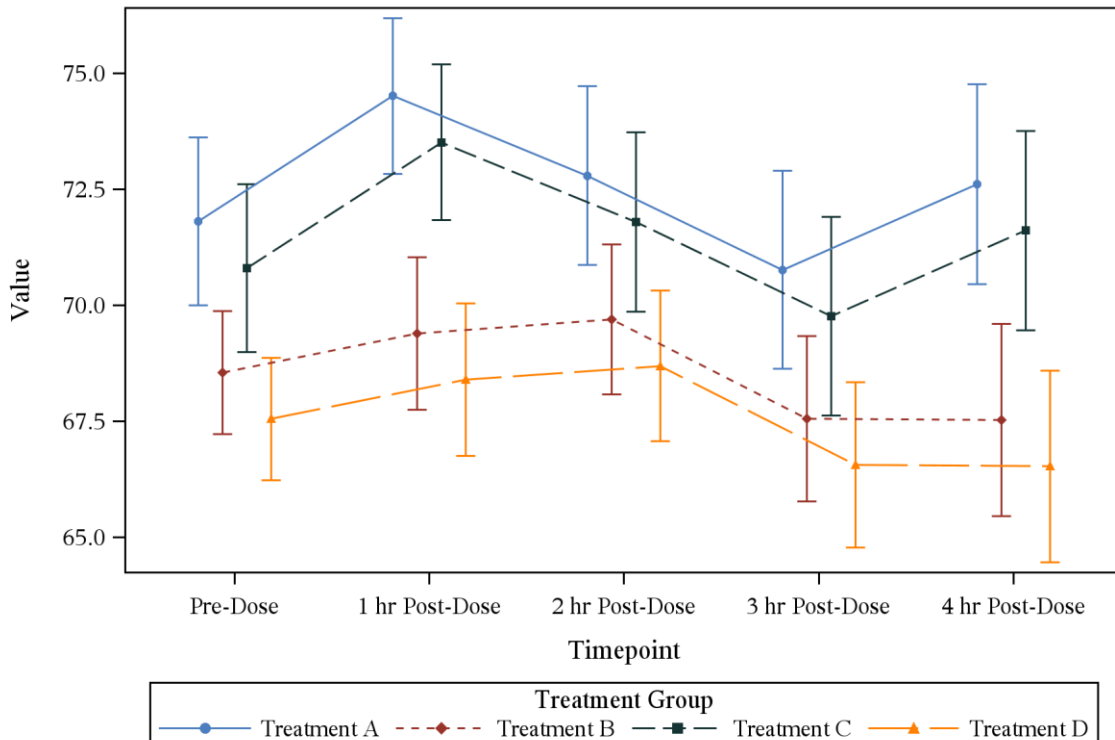


Figure 1. Example of a figure produced using the **%FigTemplate** and **%StartFig** macros.

After the plotting procedure, you can (optionally) close ODS graphics with the following statement: `ods graphics off;`. This example shows a single page figure. If you want a multipage figure, you only need to call **%FigTemplate** once, but call **%StartFig** before each output.

GRAPHS PRODUCED FROM SAS PROCEDURES

You can also use PROC TEMPLATE to alter the default appearance of graphics produced automatically from statistical procedures. Rather than running the procedure, saving the output in a dataset, possibly manipulating that dataset, and running a second procedure to create a customized plot, you can get the plot exactly as you want it from the first procedure. For example, the macro **%KMFigTemplate** is used to adjust the default appearance of the survival plot from PROC TEMPLATE. The usage of **%KMFigTemplate** follows:

```

%KMFigTemplate(xlab=Time to Event, ylab=Survival Probability,
               grouplab=Treatment Group, fontsz=9);

```

```

%FigTemplate(color=Y, sameline=N, fill=N);

%StartFig(path=%bquote(C:\SESUG 2017), name=F2, height=4, width=6,
           units=in);

ods noptitle;
ods select survivalplot;

proc lifetest data=mydata
           plots=survival(atrisk(maxlen=20 outside)=0 to 150 by 50);

           time tte*censor(1);
           strata treat;
           format treat trt.;
run;

```

The macro **%KMFigTemplate** includes the default template for the survival plot created by PROC LIFETEST, with some alterations. Here is a snippet of the code contained in this macro:

```

%macro KMFigTemplate(xlab=Time to Event, ylab=Survival Probability,
                    grouplab=Treatment Group);
proc template;
  define statgraph Stat.Lifetest.Graphics.ProductLimitSurvival2;

    /*...*/

    layout overlay / xaxisopts=(shortlabel=XNAME label="&xlab"
                               /*...*/) yaxisopts=(label="&ylab" /*...*/);

    /*...*/

    if (PLOTCEASURED=1)
      scatterplot y=CENSORED x=TIME / group=STRATUM index=STRATUMNUM
                                tiplabel=(y="&ylab")
                                markerattrs=(symbol=circle);
    endif;

    DiscreteLegend "Survival" / title="&grouplab" location=inside
                                autoalign=(Bottom BottomLeft
                                BottomRight);

    /*...*/

  EndGraph;
end;
run;
%mend KMFigTemplate;

```

Some changes to the default template are highlighted. Three macro parameters (xlab, ylab, grouplab) are used to change the default axis and group labels on the plot. Additionally, the default symbol is changed from plus to circle and the default alignment of the legend is changed to the bottom of the plot. For any procedure, you can get the code of the template printed to the log by running the following code:

```

proc template;
  source <template name>;
run;

```

To determine the templates used in any procedure, run your procedure with ODS trace on. A record of each output created, including the template used, is written to the log. For example, Figure 2 shows a

portion of the log from the PROC LIFETEST code above, when ODS trace is on. The survival plot (ODS name is "SurvivalPlot") uses the template called "Stat.Lifetest.Graphics.ProductLimitSurvival2", which is the template we altered in the **%KMFigTemplate** macro.

```
Output Added:
-----
Name:      SurvivalPlot
Label:     Survival Curves
Template:  Stat.Lifetest.Graphics.ProductLimitSurvival2
Path:     Lifetest.SurvivalPlot
-----
NOTE: PROCEDURE LIFETEST used (Total process time):
      real time          0.95 seconds
      cpu time           0.26 seconds
```

Figure 2. A section of the SAS log when PROC LIFETEST is run with ODS trace on.

JOINING IMAGES WITH PROC REPORT

The macro **%PrintFig** is used to print the figures to your output destination using PROC REPORT (SAS Institute Inc., 2012). Each figure is printed on a separate page of the document. PROC REPORT also is used to add titles and footnotes to the figures. This way, if you want to use the images for other purposes, titles and footnotes are not embedded in the image file.

```
/*-----*\
| macro: PrintFig                                     |
| usage: Print figure(s) to your ODS destination     |
| parameters: path = (path of the output location)   |
|              name = (filename of the image)         |
|              pages = (the total number of pages, for |
|                      multipage figures; leave missing |
|                      for single page figures)       |
\*-----*/
```

```
%macro PrintFig(path=, name=, pages=);
```

```
/* Creating a multi-page figure (one image per page, with common
title/footnotes)*/
%if &pages ne %then %do;
```

```
/* Create a dataset to be used to make a format which will contain the
path and name of each figure. The variables are: start (page
number or order of the figures), label (path to each figure in
sequence), and fmtname (format name). Here, I call our format "imgs".
Then use proc format with the cntlin option to create the format. See
the SAS sample referenced below (SAS Institute Inc., 2012). */
```

```
/* A format to blank out the values in the report so only our images
appear */
proc format;
  value blank other=" ";
run;
```

```
/* Create a dataset to be used in the proc report. The variable pag
represents the page number and also the image number.*/
data total;
%do i=1 %to &pages;
  pag=&i; output;
```

```

%end;
run;

proc report data=total nocenter missing nowindows split='#' contents=""
            style(report)=[rule=none frame=void vjust=m];
    column  pag image;

    define pag    / order  order = internal noprint ;
    /* The variable "image" is a computed variable. Note that it takes
    the value of "pag" and formats it with our "imgs" format. This
    passes the path where your image is saved to the postimage= option.
    The format "blank" keeps the values of pag from printing */
    define image  / display ' ' computed format=blank.
                    style(column)=[postimage=imgs. cellwidth=9.0 in
                    just=CENTER];

    break after pag / page; /* Create a page break after each figure */

    compute image;
        image = pag;
    endcomp;

    /* Here you can use compute blocks to add the figure titles and
    footnotes before and after the automatic variable "_page_" which
    indicates the page breaks*/

    run;
%end;

/* Creating a single page figure. Similar to the previous block, but you
don't need the formats since you only have one path and image name*/
%else %do;

    /* In this case, the dataset "total" has a single record where pag=1 */

    proc report data=total nocenter missing nowindows split='#' contents=""
            style(report)=[rule=none frame=void vjust=m];
    column  pag image;

    define pag    / order  order = internal noprint ;
    /* Here the variable "image" is computed as nothing, but an
    image is placed */
    define image  / display ' ' computed
                    style(column)=[postimage="&path\&name..png"
                    cellwidth=9.0 in just=CENTER];

    break after pag / page;

    /* Here you can use compute blocks to add the figure titles and
    footnotes before and after the automatic variable "_page_" which
    indicates the page breaks*/

    run;
%end;

%mend PrintFig;

```

In the example below, the output destination is RTF. A file called "figures.rtf" is created. The title and footnote statements put text into the header and footer of the document. The **%PrintFig** macro places the image and puts the titles and footnotes related to that image directly above and below it. The following code shows how you would use all these elements together to create a report:

```
title "The Document Title";
footnote "A Document Footnote";
options orientation=landscape leftmargin=1in rightmargin=1in topmargin=1in
      bottommargin=1in;
ods rtf style=pearl file="C:\SESUG 2017\figures.rtf" ;

%PrintFig(path=%bquote(C:\SESUG 2017), name=F1, pages=);
%PrintFig(path=%bquote(C:\SESUG 2017), name=F2, pages=);
%PrintFig(path=%bquote(C:\SESUG 2017), name=F3, pages=);

ods rtf close;
```

The first page of the RTF file can be seen in Figure 3 below. This basic example shows the functionality of the **%PrintFig** macro. These macros can be enhanced to control the overall style of the rtf document, such as standardizing the titles/footnotes, adding a table of contents, adding page numbers, etc.

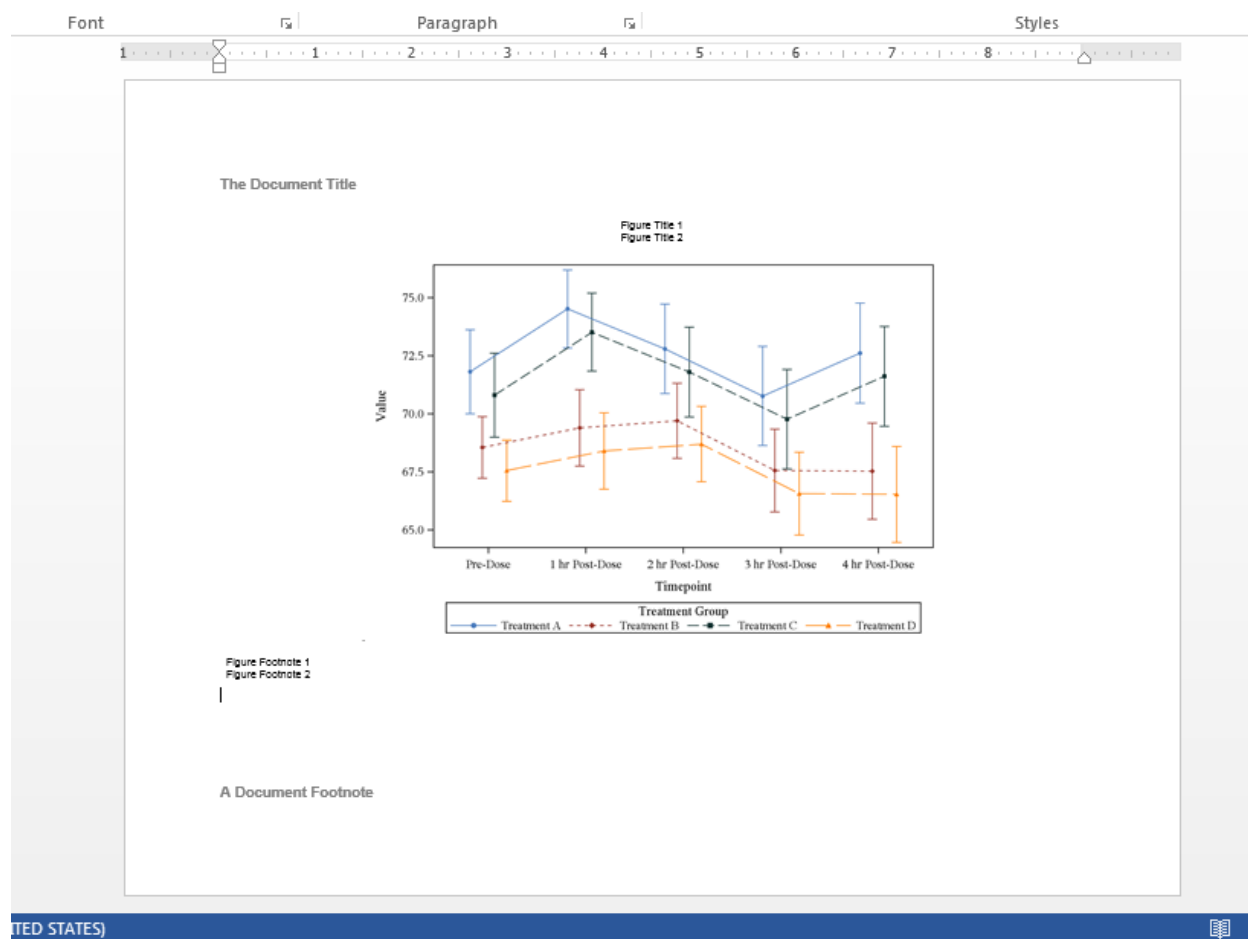


Figure 3. RTF output using the %PrintFig macro

The example above would produce a report of only figures, however it is quite simple to create a report that would combine figures and tables/text. The program would need to be altered to include additional code to produce tables or other text (presumably also with PROC REPORT) along with the figures as follows:

```

title "The Document Title";
footnote "A Document Footnote";
options orientation=landscape leftmargin=1in rightmargin=1in topmargin=1in
        bottommargin=1in;
ods rtf style=pearl file="C:\SESUG 2017\figures.rtf" ;

/* some code to produce a table*/
%PrintFig(path=%bquote(C:\SESUG 2017), name=F1, pages=, label=Figure 1);
/* some code to produce a table*/
%PrintFig(path=%bquote(C:\SESUG 2017), name=F2, pages=, label=Figure 2);
/* some code to produce a table*/
%PrintFig(path=%bquote(C:\SESUG 2017), name=F3, pages=, label=Figure 3);

ods rtf close;

```

The important thing to note is that any output can be added to the report by putting the code in between the statements that open and close the ODS destination.

CONCLUSION

The macros presented in this paper create a default appearance to be used throughout a project or study, which is applied to all figure output. Since setting up the style or template for figures is a repetitive task, which often needs to be consistent, it is a good idea to use a macro to accomplish this. This paper also demonstrates how to print multiple figures in a single report file using PROC REPORT. This method allows for many image files to be consolidated to a single document as well as permitting tables and figures to be included in the same report.

REFERENCES

SAS Institute Inc. 2012. "Sample 46552: Embed images in a PROC REPORT table." Accessed January 19, 2017. <http://support.sas.com/kb/46/552.html>

Complete code can be received by contacting the author.

ACKNOWLEDGMENTS

I would like to thank Gianluca Mortari for providing PROC REPORT code. I would like to acknowledge Martina Garlet, who developed and maintains our current standard macro system for tables and listings. I would also like to thank Hunter Vega and Jo Marshall for their helpful reviews on this paper.

RECOMMENDED READING

- SAS® *Graph Template Language: User's Guide*
- SAS® *Output Delivery System: User's Guide*
- SAS® *ODS Graphics: Procedure's Guide*

CONTACT INFORMATION

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

Laura Williams
CROS NT, LLC
(919) 929-5015
laura.williams@crosnt.com
www.crosnt.com