PharmaSUG 2015 – Paper PO03

A Visual Reflection on SAS/GRAPH® History: Plot, Gplot, Greplay, and Sgrender

Haibin Shu, AccuClin Global Services LLC, Wayne, PA
John He, AccuClin Global Services LLC, Wayne, PA

ABSTRACT

The authors present the novel development of SAS/GRAPH techniques including Proc Plot, Proc Gplot, Proc Greplay, and Proc Sgrender, and provide the real-life examples to explain how SAS/GRAPH as a visualization tool has indeed been evolved in all aspects over the years. We believe that this visualization tool now has become more powerful and more user-friendly.

INTRODUCTION

The authors attentively select four procedures to showcase the different stages of SAS/GRAPH development. Although there're many other ways to demonstrate this technology evolvement, these four procedures, as commonly used by programmers working over the figures, indisputably represent the most important features. These features have been developed over a decade or even longer. A visual reflection on the history will not only help one to better understand these popularly used graph procedures but also help to promote the employment of the advanced features available now and in addition, to use them effectively in presenting visual analytical results.

IN THE "OLD DAYS"

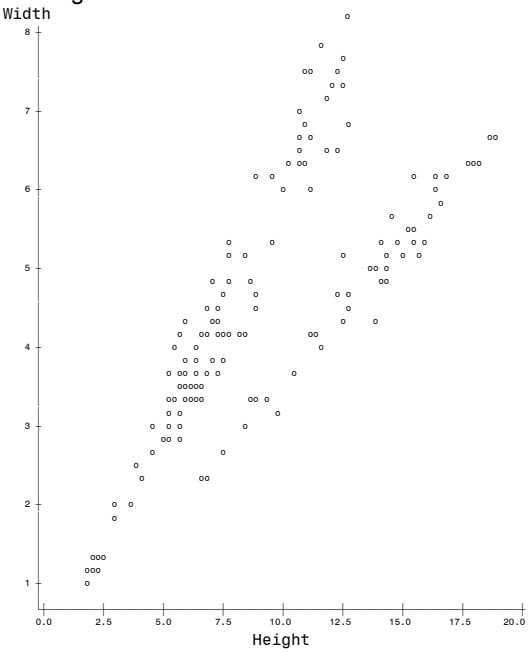
When there was nothing else available in SAS older versions (V5 or earlier), Proc Plot was widely employed to present text-type-visual effect in a limited way, which was less data discernible: strictly two-dimensional, no color, and didn't scale well. The then most commonly used Scatter Plot often faces the technical challenges.

PROC PLOT

The following code generated a Scatter Plot – illustrating a positive correlation between the height and width measurements among fish specifies such as Bream, Parkki, Perch, Pike, Roach, Smelt, and Whitefish (based on an exemplary data set built in SAS):

```
proc plot data=sashelp.fish;
     plot width*height='o';
quit;
```

The "figure" looks like this:



In the above content SAS monospace font has to be retained because the output is not based on a proportional font. It is nearly impossible to perform any post editing without compromising partial or even the entire figure.

PROC GPLOT

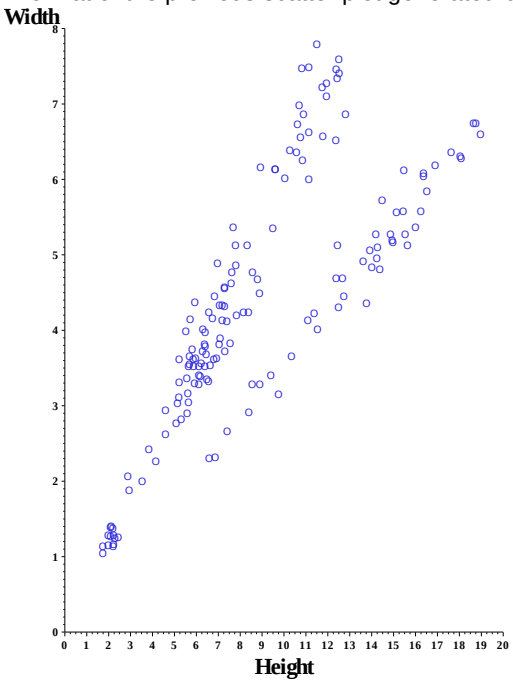
Gplot was then developed to plot real graphic figures beginning in V6. It obviously overcame the above mentioned weakness of Proc Plot; however, it has also the fetters in use as it cannot accommodate the following important presentation needs:

1. There was no easy figure format that could work smoothly with other files such as Word (back in V6 and Office 97)
2. For multiple figures generation, the presentation was largely confined to one figure a time.

The first issue above is a non-issue nowadays thanks to a variety of advanced graphic formats such as jpeg; but back in V6, the specific Computer Graphics Metafile (CGM) format was adopted by SAS to produce scalable figures with manageable size that could be inserted into WORD without compromising the quality of pictures. See codes below to specify the CGM graphic format through the device option:

```
filename outf "c:\temp\test.cgm";
goptions reset=all device=CGMOF97P rotate=portrait display gsfname=outf;
proc gplot
…
filename outf clear;
```

Generally speaking, one needed to first insert the CGM file into WORD then view the image – displayed below is the CGM format of the previous scatter plot generated by proc gplot:



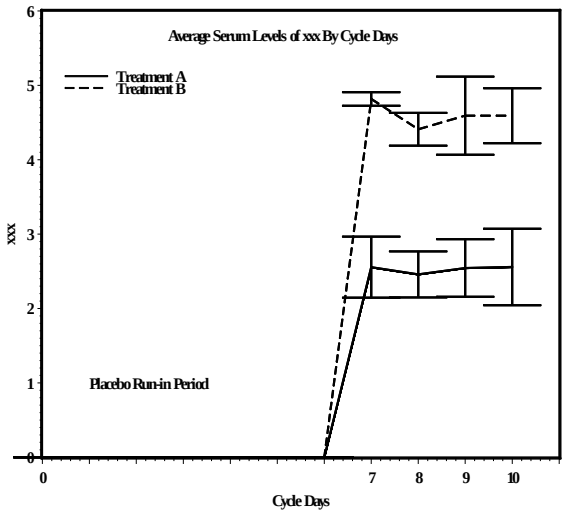
Offset=(0) was used in the axis statements so the origins of the horizontal axis and the vertical axis were the same in the above figure; thanks to Gplot, different font/size/highlights can be specified as indicated above to achieve desirable visual effects.

Regarding the 2nd limitation of Proc Gplot, it was almost impossible without using Proc Greplay, and still challenging even with the aid of Proc Greplay, to display multiple figures on the same page and arrange their individual positions by customization. Details about Proc Greplay will be discussed in the later section. The following syntax does produce multiple plots with default setting, including the positions for individual figures etc.:

```
plot (y1 y2)*(x1 x2);
```

This multiplicity feature only applies to plot-type figures granted the variables from the same input data set.

Of note as a useful option in the symbol statement i(interpol)=std#mtj (where #=1,2,3) specifies that a solid line connect the mean value with ±# standard deviations with tops and bottoms and mean values be connected with a line from bar to bar, see below illustration based on a hypothetical test data set from a serum chemistry lab value which was collected during the placebo run-in period and on Cycle Days 7 to 10:



PROC GREPLAY

Now we arrived at a point of SAS graph history, the Greplay came to play to facilitate arrangement of multiple figures on one page; however, the programming effort was rather sophisticated involving several steps to set up:

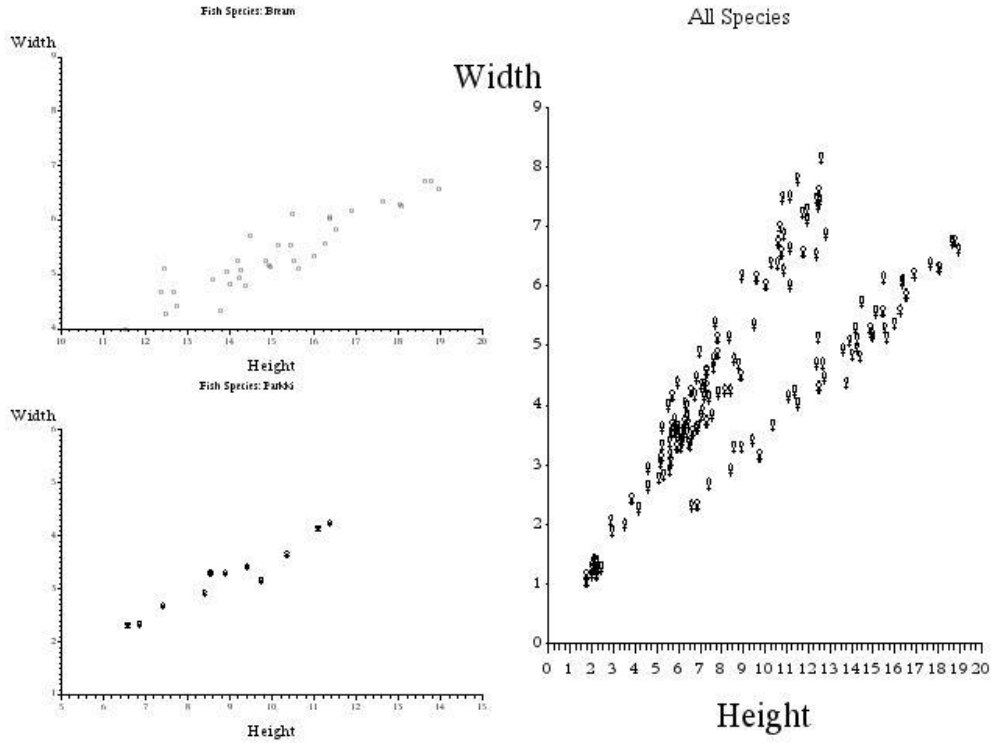
1. Specify locations of each individual figures by 4 pairs of coordinates, i.e., (llx, lly), (lrx, lry), (ulx, uly), and (urx, ury), which correspond to the four corners of each graphic area. It's achievable to allocate the graphic areas with overlapping if necessary. In the following example, the whole page is divided into three sub-areas with the left two areas for two smaller figures and the right area for a large figure:

```
proc greplay nofs tc=work.graph;
     tdef templat1
          /*Graphic Area #1 - upper left*/
          1/llx=1 lly=40
            lrx=40 lry=40
            ulx=1 uly=80
            urx=40 ury=80
          /*Graphic Area #2 - lower left*/
          2/llx=1 lly=1
            lrx=40 lry=1
            ulx=1 uly=40
            urx=40 ury=40
          /*Graphic Area #3 - right side*/
          3/llx=40 lly=1
            lrx=80 lry=1
            ulx=40 uly=80
            urx=80 ury=80;
run;
quit;
```

2. Generate individual output objects separately, e.g., by Proc Gplot, Proc Glide etc. In this example, we will have 3 individual output objects ready to display.
3. Replay (like showing movie!) the individual objects into the final figure using Proc Greplay according to the graphic areas 1, 2, and 3 previously defined:

```
proc greplay igout=work.graph tc=work.graph template=templat1 nofs;
     treplay 1:1 2:2 3:3;
     delete 1 2 3;
run;
```

See below a replay of the previous scatter plot according to the three graphic areas defined at Step 1. The left two figures were subset for fish specifies Bream (Graphic Area #1) and Parkki (Graphic Area #2) respectively, while the right side was the overall scatter plot (Graphic Area #3) combing all species from the sample data set built in SAS:



ANNOTATIONS

When the programmers intended to further enhance their figure outputs, the SAS institute introduced the annotation tool. It is an important tool to add customized features into a figure. It could be a textual string, a line with an angle, a piece of object or specific color etc. These features can be added from the annotation data set. In the figure of page 3 of this paper, the text string “Placebo Run-in Period” was added through an annotation data set:

```
data _anno;
     retain xsys ysys '2';
     function='label';
     position='>';
     x=1;
     y=1;
     text='Placebo Run-in Period';
     output;
     …
run;
```

Once this annotation data set is created and ready to use, we can deploy the annotation in the graph procedure by using the relevant option to include this data set _anno.

NOW EVERYTHING ODS

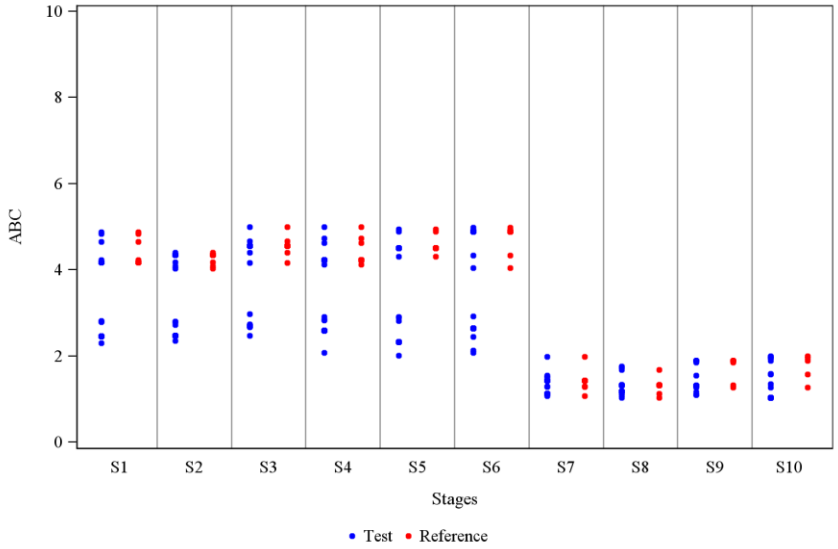
Output Delivery System (ODS) ushers in a new era in SAS Graph programming. The Proc Sgrender is able to produce sophisticated figures easily and the outputs are fully compatible with Word. With ODS, the graph procedures are dramatically simplified while output figures are flexible and discernible.

POWER OF PROC SGRENDER

The coding of procedure SGRENDER could be simply like this:

```
Proc sgrender data=_input_data_ template=…
run;
```

The SGRENDER procedure can generate a graph from data and a template that is formulated in the program. With Proc SGRENDER and the template, we can create highly customized graphs. The above simple SGRENDER will generate the following individual stage profile figure from in vivo data (WORD/RTF format!):



From the above graph we can see the usage of display format, color, font and reference note to represent the data. The achieved simple coding and customized graph are possible thanks to the technology of SGRENDER procedure and ODS template.

ODS AND TEMPLATE

In order to automatically generate graphic output in WORD or HTML or other ODS destination format, the ODS Template should include both graph definition as well as style definition for the particular ODS destination. The code embraced by “begin graph” and “endgraph” in the graph template specifies how the graphic output will be deployed; the code following “define style” is the regular style definition for ODS output. Here is what the template structure looks like:

```
proc template;
     define statgraph sp/store=work.templ;
            begingraph;
              ...
            endgraph;
     end;
     define style tStyle/store=work.templ;
            parent=styles.rtf;
            ...
     end;
run;
```

The above procedure defines the style for the graph within the bracket of BEGINGRAPH- ENDGRAPH. After the generation of the defined style, you can also update the styles accordingly whenever needed.

WHY THIS IS USER FRIENDLY

First of all generating sophisticated figures becomes a task that is quite straightforward. Both graph and ODS templates have abundant well-defined and easy-to-use features such as layout (including lattice, overlay etc.) and a variety of figure types such as scatterplot, sidebar etc. Following the corresponding manuals to use these features is not only intuitive for a programmer with graph experience but also involving less programming efforts because the syntax is established at a high level language style.

Compatibility and full integration with WORD is another important advantage to make using Proc Sgrender user friendly. The graph in ODS can seamlessly integrate with output tables/listings and the result makes it unnecessary for any extra effort to import figures.

DISCUSSION AND CONCLUSION

Generally it might be a desirable strategy to have a dedicated resource to maintain global templates that are used across team and across projects. As shown after the global templates (both graph and ODS) are established or maintained up-to-date (e.g. due to a new therapeutic area), the programming effort of applying the global templates to study specific data to generate the figures in a clinical trial study will be pragmatically straightforward.

The benefits of SGRENDER and ODS can be expandable with the usage of Annotation. We can use option of SGANNO to specify annotation data set to make further customizations.

With all these advantages said, a few suggestions can be made for the future employment of SAS/GRAPH. We may need more online reference materials such as: how to make it easy to learn and effectively use SAS/GRAPH technology for beginners? Due to its high-level syntax style, it might be challenging for a programmer who doesn't have any graph experience. It can also be helpful to consolidate and centralize the relevant contents so people can pick up the complete knowledge and practice.

Overall, the leading technology from integration of ODS and Proc Sgrender marks a BIG step forward to have greatly empowered SAS/GRAPH and become an easy-to-use and value-added visualization tool for programmers.

REFERENCES

SAS Institute Inc. SAS Manuals Version 6
SAS Institute Inc. SAS Online Doc 8.2
SAS Institute Inc. SAS Online Doc 9.3
SAS Institute Inc. SAS Online Doc 9.4

CONTACT INFORMATION

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

Haibin Shu
AccuClin Global Services
P.O. Box 1491
1000 W. Valley Rd
Southeastern, PA 19399-1491
haibin@accuclinglobal.com