Paper 72-26

Reduction of Symbols in Plots and Generation of Accompanying Legend Using Annotate

Dean Clous, Trilogy Consulting, Kalamazoo, MI

ABSTRACT

You've been given a large amount of data to be displayed graphically using Proc GPLOT and you've successfully accomplished this. You're told that the resulting graph looks "cluttered". Various basic methods of changing the graph's appearance are judged as inadequate. The suggestion is made to display each plotline and only have every nth symbol on each plotline; this will satisfy your client's need to "unclutter" the graph. This paper will attempt to detail the process of generating a graph with the reduced symbols accompanied by an appropriate legend. It is hoped that this presentation will benefit the novice and intermediate SAS user on all platforms.

INTRODUCTION

Generation of graphs using Proc GPLOT is a tool frequently employed to display data in reports. Data can be displayed across time in an X-Y graph using the PLOT statement. In some cases due to the data values involved, the graph may appear congested and little seems to be able to clear up this problem. The appearance of the overall graph may be improved by reducing the number of symbols displayed on each plot line. Generation of a corresponding legend using ANNOTATE will complete the display.

ANALYSIS OF REDUCTION

Should this be the route taken to improving the appearance of the graph, the first step is to determine how many plot symbols to remove from the plot line. My personal preference is to keep around 20% of the symbols but your mileage may vary; some users may prefer up to 30% or even 40% retention. The endpoints of each line also seem to be logical choices for retention. Additional symbols will have to be assigned to deal with the additional data points.

PROCESS DATA

Using by-group processing, subset the data keeping the first, last, and every nth data point where n is directly based on the retention percentage. This subsetted data needs to be differentiated from the original as it will be recombined with the original dataset. If the by-variable is a numeric variable, it is suggested that it be converted to a character version to aid in the differentiation process. In the example, the by-variable is category (CAT). The subsetted data is differentiated by appending an asterisk to the category value. The subsetted data is then recombined with the original data and sorted on the character version of the by-variable, category. The result is a complete copy of the data supplemented by select data points that exactly correspond to other records; the only difference is a single character in the by-variable. This is important as it allows us to systematically deal with the symbols for the plot.

```
proc sort data=temp;
  by cat rdate;
run;

data temp1;
  set temp;
  by cat rdate;
  substr(cat,5,1)='*';
  if (first.cat OR last.cat OR mod(_n_,4)=0);
run;

data temp2;
  set temp temp1;
run;

proc sort data=temp2;
  by cat rdate;
run;
```

ASSIGNMENT OF SYMBOL DEFINITIONS

With the data arranged in this order, the next step is to assign the appropriate symbols. Two sets of symbols will be assigned for each virtual plotline. A virtual plot line will consist of two physical plot lines: one line based on the original data and one based on the subsetted data for that year. Each physical line needs to have a symbol defined for it. Using the subsetting/recombination process described above, the odd numbered symbol statements (Group A) will be associated with the original data and the even numbered symbol statements (Group B) will be associated with the subsetted data. Group A is given an Interpolation value of "Join" (I=join) and a Value of "None" (V=none). Group B is given an Interpolation value of "None" (I=none) and a Value of "Dot" (V=dot). (Dot could be replaced by any other valid symbol value.) Line type, Width, and Height are given the same value in each pair of symbol definitions per one's personal preference for the graph's appearance or requirements as the case may be. Differentiation between the virtual lines in this case is based on the Line Type. An alternate scheme would use a constant Line Type across virtual lines and vary the symbol Value for Group B. This is the method used in the example.

```
symbol1 I=join v=none     c=black;
symbol2 I=none v=dot      c=black;
symbol3 I=join v=none     c=black;
symbol4 I=none v=circle   c=black;
symbol5 I=join v=none     c=black;
symbol6 I=none v=triangle c=black;
```

GENERATE ANNOTATE DATASET

Using two physical lines to generate a single virtual line leads to a problem with the legend. Each by-group should have one entry that includes both the line type and the symbol that represents that particular plot line. The legend generated by proc GPLOT will instead contain an entry for twice as many lines as there are by-groups: one each for the original by-groups and the subsetted by-groups. Needless to say, this is not an acceptable result. This situation will be rectified using ANNOTATE to generate a custom legend that has the appropriate placement and the correct line descriptions.

For those who may be unfamiliar with the ANNOTATE facility, there are three steps to modifying a graph using Annotate:

1. Determine what you want to draw and the location in the plot where the object is to be placed.
2. Generate a dataset composed of specific variables for use by the ANNOTATE facility. Each observation specifies what graphics element to draw, where to draw it, and how to draw it. (It's not as hard as it sounds.)
3. Submit a SAS/GRAPH procedure to produce the desired graph using the ANNOTATE= option.

Determination of what is to be drawn is fairly basic. The legend needs to have a title, an example segment of each virtual line, a corresponding label, and a box to contain the legend. A little quick sketching will give the user a layout of these items to be generated.

Now that the appearance of the legend has been determined, a dataset containing the observations for use with the ANNOTATE facility needs to be generated. There are three methods to handle this process:

1. Use a DATA step, generating the observations directly with assignment statements or INPUT and CARDS statements.

2. Use the FSEDIT or FSVIEW procedures in SAS/FSP software.

3. Use Annotate macros in a DATA step to assign values to Annotate variables.

For the purpose of this paper, the method for generating this dataset will be by means of a DATA step. There are many options available to the user who is delving into the ANNOTATE facility. This paper will deal specifically with those items needed to complete this task. For further information on the other options available in Annotate, the reader is directed to the chapters on the Annotate Facility in the SAS/GRAPH reference manual.

The first item to deal with is where the legend will be located. There are three drawing areas defined for use with graphics in SAS: the Data Area, the Procedure Output Area, and the Graphics Output Area. The coordinate system for each area is defined differently with each having a different origin and a different range of values for X and Y coordinates. The Data Area is the area bounded by the X and Y axes in the plot. The Procedure Output Area contains the Data Area and also includes the area where the titles, footnotes, and axis labels would be placed. The Graphics Output Area contains the Procedure Output Area plus a buffer zone outside of this area, effectively the maximum possible accessible area. Annotate handles this process through the use of two variables, XSYS and YSYS. These variables take on a character value of 1-9 or A, B, or C depending on which of the three areas the object is to be placed in, whether the coordinates are expressed as absolute or relative values, and whether the units are to be expressed in percentages or values/cells. Once again the reader is directed to the SAS/GRAPH reference manual for a more in depth investigation of this topic. For the purpose of this paper, XSYS and YSYS will be given a value of “1" which indicates placement in the Data Area using absolute percentage references.

A related variable that needs to be assigned is HSYS. This variable defines the area and coordinate system used by Annotate’s SIZE variable. The values that can be assigned to this variable are identical to those available to XSYS and YSYS. The assigned value in this case is also “1".

The WHEN variable is used to control when a function is executed or a graphics item is generated in relation to the generation of other graphics output. (Remember, each record in the annotate dataset is a graphics object.) The WHEN variable has a value of “A” (for After the graph is drawn), “B” (for Before the graph is drawn), or missing. “B” is the default and a missing value assumes the default. Our legend will be drawn after the graph is drawn.
Placement of the legend now becomes an issue as the references to individual items are going to be based on an absolute percentage of the Data Area. It is assumed that the person generating the code for a graph has a very rough idea of what the output is going to look like: overall positive or negative slope, expected white space locations, etc... so that a decision can be made as to where to place the legend. This is assuming that the location of the legend is to be in the Data Area. If the legend is to be placed outside the Data Area but inside the Procedure Output Area, this information is not an issue. For this example the data has an overall negative slope so the legend can be placed in the upper right corner of the Data Area. The rough placement of the legend can be estimated but expect to “tweak” the actual X-Y coordinates to achieve the desired final results.

Generation of graph items is very simple. First the X-Y coordinates to be operated on are specified. Then a text item or function is specified to operate from that point. Then output a record to the dataset. The first step is to draw a box to put the legend in. The box for the legend needs to be a solid object (opaque) as it is being generated after the graph has been created. Should the graph contain grid lines or calculations have gone awry and data lines intersect the area of the legend box, they will be visible through the legend. As might be imagined, this is a most distracting effect. To avoid this effect the Style is given a value of “solid” (STYLE=”solid”).

Two functions are involved in generating a rectangle to put the legend in (or any other polygon for that matter): POLY and POLYCONT. The POLY function specifies the starting point for a polygon. The POLYCONT function specifies the continuation of the current polygon. Note that the last POLYCONT has to have operating coordinates equal to the starting point of the polygon (i.e., the X-Y coordinates associated with the POLY function). The color of the polygon is determined by the value of the COLOR variable in the record with function equal to “POLY”. The color of the outline of the polygon is determined by the value of the COLOR variable in the first record with function equal to “POLYCONT”.

At this point we will be generating a solid, outlined rectangle of whatever colors have been specified. The next step is placing the sample lines into the legend. This process also involves two functions: MOVE and DRAW. Annotate tracks two sets of internal (non-user modifiable) coordinates: the coordinates of the last graphics element drawn (XLAST, YLAST) and the coordinates of the last text drawn (XLSTT, YLSTT). Many Annotate functions use these points as the starting point and the X-Y coordinates in the same record as the function as the end point. DRAW is one of these functions. The MOVE function is used to set the internal coordinates to the starting point of the line segment that needs to be drawn. The DRAW function then draws a line from that point to the X-Y coordinates associated with the DRAW function. We specify a color for the line, clear the style, and set the size for the line. Specify the starting coordinate for the line, set the line type to 0, and set the function to “move” in order to reset the internal coordinates. Specify the ending coordinate for the line, set the line type to the value appropriate for the line being drawn, and set the function to “draw” . This is repeated one time for each by-group value shifting the vertical coordinate for each line.

Once the lines for each by-group have been generated, the appropriate symbols must be added to each line. It is suggested that two or three examples of the symbol be placed on each sample line segment in the legend but, the exact number will be a matter of personal preference. The symbols are placed on the line segments as text items. Specify the coordinates for the location of the symbol and assign the variable TEXT the appropriate value.

The only remaining activity needed for the layout of the legend is the placement of the actual text items labeling the by-groups and the title. The process is basically the same as for the placement of the sample symbols. The only difference is that instead of assigning TEXT a value that converts to a special character, an actual label that identifies the particular by-group or title is assigned instead.

```
data anno;
  length text $9. color style function $8.;
  retain xsys '1' ysys '1' hsys '1' when 'a';
  color='white';
```

```
 style='solid';
 x=77; y=73; function='poly';     line=1; output;
 x=99; y=73; function='polycont';         output;
 x=99; y=93; function='polycont';         output;
 x=77; y=93; function='polycont';         output;
 x=77; y=73; function='polycont';         output;
 color='black';
 style=' ';
 size=.1;
 x=78; y=84; function='move';     line=0; output;
 x=86; y=84; function='draw';     line=1; output;
 x=78; y=80; function='move';     line=0; output;
 x=86; y=80; function='draw';     line=1; output;
 x=78; y=76; function='move';     line=0; output;
 x=86; y=76; function='draw';     line=1; output;
 function='symbol';
 size=2;
 x=80; y=84; text='dot';                output;
 x=80; y=80; text='circle';             output;
 x=80; y=76; text='triangle';           output;
 x=84; y=84; text='dot';                output;
 x=84; y=80; text='circle';             output;
 x=84; y=76; text='triangle';           output;
 function='label';
 size=4;
 x=85; y=90; text='Category';         output;
 x=93; y=85; text='Cat1';             output;
 x=93; y=81; text='Cat2';             output;
 x=93; y=77; text='Cat3';             output;
run;
```

(*Note: certain items are unnecessarily repeated between output statements (i.e., repeated function assignments of "polycont". This is for clarity and readability purposes and is not actually needed in the code.)*

GENERATE GRAPH

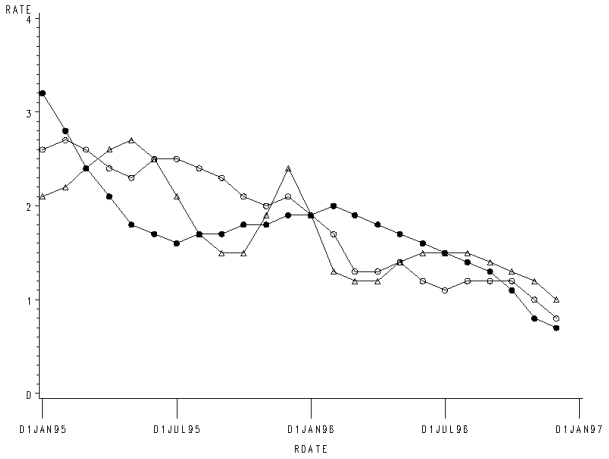
Once the annotate dataset has been created, the plot is ready for generation. This is probably the simplest step in the process. The only change to the call to proc GPLOT is to add two options: NOLEGEND and ANNOTATE=<datasetname>. Proc GPLOT will suppress the automatic generation of the legend and use each record from the annotate dataset to generate a graphic item. When all processing is done, the graph will have a custom legend that corresponds to the modified data.

```
proc gplot data=temp2;
  plot rate*rdate=cat / nolegend anno=anno;
run;
```

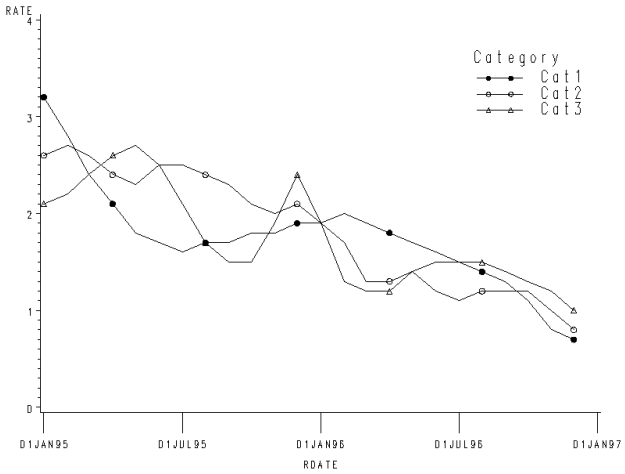
CONCLUSION

This paper has demonstrated one of many potential methods for improving the appearance of basic output from GPLOT. A secondary purpose of this paper was to illustrate how easy ANNOTATE can be to use. Many users are somewhat leery of using this product; there is no reason to be afraid of ANNOTATE. This paper barely scratches the surface of the capabilities of GPLOT and ANNOTATE. The graph in the example can be made far more appealing with a minimum of effort. Hopefully this work will prompt some users to investigate the options available in both these products regarding modification of their basic graphs.

Before:



After:



REFERENCES

References go at the end of your paper. This section is not required.

ACKNOWLEDGMENTS

Acknowledgments go after your references. This section is not required.

CONTACT INFORMATION

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

Dean Clous
Trilogy Consulting
5278 Lovers Lane
Kalamazoo, MI 49002
Work Phone: 616.344.4100, ext. 573
Fax: 616.344.6849
Email: ddclous@trilogyusa.com
Web: www.trilogyusa.com