Put Presentation Graphics in Your Applications  
Anthony L. Friebel, SAS Institute Inc., Cary, NC  
Jade Walker, SAS Institute Inc., Cary, NC

ABSTRACT
In developing applications using SAS/GRAPH software, perhaps you have come up with some clever tricks to enhance the appearance of your graphs. This paper presents a mixed bag of tricks collected over the years. The authors have tried to streamline the code examples and to show how to generalize display enhancements so that you do not have to code a separate program for every input data set you use. Techniques employed in this paper include the use of other parts of the SAS System to gather information from your input data set, the use of MACRO variables to make global changes over a program, and the building and subsequent including of external files containing SAS statements with parameters that are automatically based on your input data set. Using these generalized algorithms reduces the amount of SAS source code that you have to maintain, and, who knows, this paper may even include as an example the very graph you have been trying to create!

INCREMENTAL IMPROVEMENT OF A GRAPH
The default graph is not always the best graph, but through successive refinements, you can produce a presentation-quality graph. Here are the default statements used to produce a vertical bar chart:

```sas
data alpha; length xc 8 region 16; label region = 'Region'; input region S xc;' y; Southern Alpha 4 Southern Beta 6 Southern Complex 10 Southern Delta 3 Southern Epee 7 ;
```

The pattern selected as a default pattern dates the design of SAS/GRAPH software. Back when this pattern was selected as the default, very few devices had hardware rectangle fill. Now that most modern devices handle this quickly and easily, solid patterns make SAS/GRAPH software. Back when this pattern was selected as the default, very few devices had hardware rectangle fill. Now that most modern devices handle this quickly and easily, solid patterns make the graph more pleasing to view. In addition, if you add space around the graph by using blank TITLE and FOOTNOTE statements, the graph can retain a more aesthetic shape closer to the golden mean rather than compressing or elongating itself to fit the page. Use the AXIS statements to remove the minor tick marks to make the graph look cleaner. The following example code shows these improvements to the chart.

```sas
/* good */ title h=3 f=none ' '; /* get enough space reserved */
title2 m=(25,100) pct w=(0,-2) cells h=1 f=none 'TITLE is aligned with axis frame';
title m=(25,100) pct w=(0,-3) cells h=1 f=none 'DECOR TITLE is aligned under first TITLE';
footnote m=(25,3) pct w=(0, +2) cells h=1 f=none 'FOOTNOTE is also aligned with axis frame';
axis origin=(25,25) pct minor=cnone label=none; axis1 origin=(25,25) pct minor=cnone label=none;
proc gchart data=alpha; vbar xc / freq=y frame raxis=axis1 maxis=axis2;
run;
```

The final example shows you how to control the output provided by using the BY variable. Since the BY line is produced automatically by the software and you can exercise little control over it, you eliminate using the BY line by specifying OPTIONS HBY=0 (or OPTIONS NOBYLINE) and by using the BY variable and BY value substitution available in TITLE, FOOTNOTE, and NOTE statements to format the information usually printed in the BY line. The #BYVAL substitution uses the current value of the BY variable, and the #BYVARN substitution uses the label or the name of the BY variable. Labeling the variable region as 'Region' allows you to display a mixed-case label in your title. Moreover, this final example code draws the frame background in a slightly different color to pull the graph away from the background and make it stand out more. Through gradual improvement, you can turn a graph from an ugly duckling into a swan. Here is the final example code:

```sas
/* best */ options hby=0; /* disable BY line */
title h=1 f=none ' '; /* get enough space reserved */
title2 m=(25,100) pct w=(0,-3) cells h=1 f=none 'COMBINED 1991 SALES FIGURES';
title m=(25,100) pct w=(0,-3) cells h=1 f=none 'BYVAL(1991) REVISED!';
footnote m=(25,3) pct w=(0, +3) cells h=1 f=none 'Product sales in millions of units';
axis1 origin=(25,25) pct minor=cnone label=none; axis2 origin=(25,25) pct minor=cnone label=none;
proc gchart data=alpha; by region;
axis1 origin=10' axis1 maxis=axis2;
run;
```

Another presentation-quality layout aligns the titles and footnotes with the vertical axis of the graph. In order to align all these elements to the same horizontal position, you need to use the MOVE—parameter in the TITLE and FOOTNOTE statements and the ORIGIN—parameter in the AXIS statements to refer to the same point in screen percentage. Once you use the MOVE—parameter in the TITLE and FOOTNOTE statements, however, you give up the space-reserving algorithm that these statements perform by default. You need to make a tall blank title to reserve room at the top; the ORIGIN—parameter in the AXIS statement prevents any collision between the graph and the footnote. For the titles and footnotes, once you determine at what point the text and graphics should line up, you need to move the text in the vertical direction to prevent collision of the titles. Since the font used in the following code has a height of one (1) cell, you can adjust the title in the vertical direction by a multiple of one cell to get the correct vertical position.

```sas
data alpha; length xc 8 region 16; label region = 'Region'; /* see 'BEST' example */
input region S xc;' y;
```

The default graph is not always the best graph, but through successive refinements, you can produce a presentation-quality graph. Here are the default statements used to produce a vertical bar chart:

```sas
data alpha; length xc 8 region 16; label region = 'Region'; /* see 'BEST' example */
input region S xc;' y;
```

Another presentation-quality layout aligns the titles and footnotes with the vertical axis of the graph. In order to align all these elements to the same horizontal position, you need to use the MOVE—parameter in the TITLE and FOOTNOTE statements and the ORIGIN—parameter in the AXIS statements to refer to the same point in screen percentage. Once you use the MOVE—parameter in the TITLE and FOOTNOTE statements, however, you give up the space-reserving algorithm that these statements perform by default. You need to make a tall blank title to reserve room at the top; the ORIGIN—parameter in the AXIS statement prevents any collision between the graph and the footnote. For the titles and footnotes, once you determine at what point the text and graphics should line up, you need to move the text in the vertical direction to prevent collision of the titles. Since the font used in the following code has a height of one (1) cell, you can adjust the title in the vertical direction by a multiple of one cell to get the correct vertical position.
GENERALIZING A PROGRAM FOR ANY INPUT DATA SET

This next program using the GPLOT procedure has a few clever tricks to produce output with better date handling on AXIS statements and to produce annotation which changes as the input data changes. The example incorporates a generalized algorithm to annotate the minimum and the maximum of the dependent variable with text and to signify the mean values by a reference line. The example uses a method of specifying the increment on an axis without knowing either the minimum or maximum values of the data. It also shows how to create a month/year axis without using SAS time/date values. The example also uses the same display enhancements shown in the GCHART procedure vertical bar chart example. This example uses MACRO variables so that it is easier to extract the code and use it in creating other charts.

The data used here represent rainfall over time, but month and year are recorded as two separate values rather than two components of a SAS date/time value. You ensure order by using the SORT procedure by year and month.

The following DATA step adds a variable n to the SAS data set to help this example generate for you. You must initialize n to 1 because axis tag values are 1-based, and the default behavior of the variable in the sum statement is to initialize the value at 0.

```
data rain;
  length station $ 16;
  input station $ year month predp;
  retain n ,;
  run;
```

The MEANS procedure uses the SAS data set as input and creates an output data set that contains the number of non-missing values and the minimum, maximum, and mean values for the dependent value. This output data set is used later in this paper to create an Annotate = data set that labels the maximum and minimum values.

```
/* generate controlling axis statements */
data _NULL_; file '~/sugilgplot.inc'; length text $ 40; retain ; retain i i lastyr 0; if _N_ = 1 then do; get stats; /* MACRO use for procedure options. It is often easier to use a short macro sequence for specifying things like process options which need to be inserted with $include processing. For example, just a single reference line & mean. */ text = 'VREF=P' &HVAR=WHITE VREF = /* trim(left('put(stat(pmean, best, t1)'); put option values into macro var VREF (see gplot use below) */ put option values into macro var VREF (see gplot use below) */ call symput('VREF', text);
```

This DATA step creates the AXIS statement for the dependent variable. In the following example code, you know you want the data increment between major tick marks to be three (3) units (inches here), but you do not necessarily know the range of the data, and you do not want to exclude any of the data points by hard-coding an AXIS statement. This DATA step formats the AXIS statement by using PUT statements and getting the data range variables from the output data set created by PROC MEANS to make a list using the ORDER= option.

```
/* perform effectual BY 3 order list on vertical */ inc = 3; /* set increment */ /* find nearest multiple of inc less than or equal min */ amn = pmin - mod(pmin, inc); /* find nearest multiple of inc greater than or equal max */ amx = floor(max / inc) * inc; if amx <= max then amn = amx + inc; /* construct vertical axis statement */ put "axisl origin=("&HVAR.25") put offset=(3,2) cells =; put " labelnone order="; do x = amx to amx by inc; / range of our list */ put x best.; /* each value along the way */ end;
```

The horizontal axis is formatted using the number of non-missing data values of the dependent variable. This works only if there is a 1-to-1 correspondence between the independent variable and the
dependent variable; otherwise, you have to create another PROC MEANS output data set on the independent variable. In the VALUE= clause, the numbers of the axis tag values are generated from the observation count, and the text strings are generated from a subroutine that formats the months. If a change in the variable year occurs, the code adds a second level to the axis tag to display the new value for the year variable.

```sas
/* construct horizontal axis statement */
put *axis1 origin=(XRANGE,5) pt minor-none label=none *;
put order 1 to nobs; /* from MEANS step above */
put value= ; /* text is gleaned from actual data */
end;

/* bring in data, one obs at a time */
set rain end-eof;
/* bring in data, one obs at a time */
put "T" order 1 to nobs ; /* which tick position we are working with */
if (IVAR = lastyr);
  /* just the month */
  link FMTMON ;
  put ' j"c "Jan" ' ;
  end;
else do;
  /* month and year */
  link FMTMON ;
  put trim(left(put(iYVAR, best));
end;

i = 1; /* next tick */
lastyr = IVAR ; /* track change */
/complete horizontal axis statement */
if i then put 'j"c "Jan" ' ;
return;

/namespace ;
/* generate text and justification control for individual values */
select (IVAR) ;
when ( 1 ) do; put ' j"c "Jan" ' ;
when ( 2 ) do; put ' j"c "Feb" ' ;
when ( 3 ) do; put ' j"c "Mar" ' ;
when ( 4 ) do; put ' j"c "Apr" ' ;
when ( 5 ) do; put ' j"c "May" ' ;
when ( 6 ) do; put ' j"c "Jun" ' ;
when ( 7 ) do; put ' j"c "Jul" ' ;
when ( 8 ) do; put ' j"c "Aug" ' ;
when ( 9 ) do; put ' j"c "Sep" ' ;
when (10) do; put ' j"c "Oct" ' ;
when (11) do; put ' j"c "Nov" ' ;
when (12) do; put ' j"c "Dec" ' ;
end;
return;
run;
```

The next DATA step creates an Annotate= data set that contains labels for the minimum and maximum values of the dependent variable. The DATA step compares the statistics from the PROC MEANS output data set to the original data set. If it finds a match, it uses the step data to place the label, selecting the variable position based on whether it labels the maximum or the minimum formats the text, and stores the information in the Annotate= data set.

```sas
/* get our generated GLOBAL statements into effect */
include '-/sugi/gplot.inc';
symbol join vnone c=green;
goptions hby='O; /* disable BY line */
title h=1 f=none ; /* get enough space reserved */
title2=\(\text{Recorded Rainfall Amounts}\); /* use ANNOTATE in PROC stmt to negate BY variable requirements */
footnote=\(\text{Preliminary Analysis}\); plot IVAR * / VREF from 'string' gsys=hspline axis1=axis2 ;
run;
```

Next, the external file containing the AXIS statements is included using the %INC command. The OPTIONS, TITLE, and FOOTNOTE statements show the same display enhancements that the PROC GCHART example shows. However, if you assign the origin of x to the variable XORG, the titles, footnotes, and axis will still line up no matter where you set your x origin.

```sas
/* construct horizontal axis statement */
data labels(keep=x yys y text position when);
length text $ 16;
retain;
text yys yys '2' when 'A';
if _N_ = 1 then set stats;
end retain;

/* special process for min or max of data */
if IVAR = min or IVAR = max then do;
  x = 0;
  if precip = precip then do;
    text = 'LEAST';
    position = 'B';
    /put text 'under' actual coordinate */
  else do;
    text = 'MOST';
    position = 'B';
    /put text 'above' actual coordinate */
  end;
end;
```

Creating a Pie Chart with a Legend

Often when you use pie charts as output in production jobs, you do not know the number of midpoints that will end up in your graph. Because PROC GCHART resizes room for the text that labels the midpoints, it may not leave enough room to draw a pie of a reason­able size. In these circumstances labels may collide, or positioning of the labels with respect to the pie slices becomes ambiguous. Using a legend with a pie chart allows you to have more room for the pie, making it easier to read and compare pie slices. The text labels stay with the legend; this removes labeling ambiguity because the legend and its label line up much better than the pie slice and its label. The midpoint and statistics are produced off to the right in a neat tabular format and do not interfere with the appearance of the pie.

The following example gives tighter control of the use of pattern and color for the pie chart. The PATTERN statement, if not fully qualified, cycles through pattern values and pattern colors. This example uses the Annotate facility to set the color and pattern values for the legend bars, then builds and submits fully-qualified PATTERN state-
ments that match the legend bars. You can customize this section of the example to include your favorite patterns and colors.

You use PROC SORT and PROC SUMMARY in the following example code to order the legend and to gather summary statistics for each midpoint so that you can report them in the legend. To track the composition of the other slice, use a DATA step to collect the statistics for all the discrete midpoints if the percentage of the slice does not exceed the percentage allowed to the other slice. The statistics are stored in two places — one to keep information on each component of the other slice and one to keep a running total of the information to be used in constructing statistics for the other slice. Information on midpoints included in the other slice are represented in the legend by a single bar recorded beneath the legend. You designate the midpoints in the other slice in this DATA step by setting the value of the variable rot to MISSING.

To synchronize the patterns and colors in the pie legend with the patterns and colors used in the pie chart, you need to submit properly constructed PATTERN statements before running PROC GCHART. The example program builds these statements for you. You also need to reserve a certain amount of room on the right for the execution of PROC GCHART. This is the equivalent of submitting these statements manually.

After you synchronize the colors and patterns, you read the midpoint data set and generate an Annotate= data set that produces a bar and some text containing statistics for each midpoint whose response exceeds the OTHER= value and just the text if the midpoint’s response is less than the OTHER= value in PROC GCHART. The pattern and color of the OTHER slice in the legend has to wait until last in order to match the order of the slices set up by PROC GCHART. To convert the values of the pie patterns to the bar patterns, use the BARPATT subroutine.

You include the file containing PATTERN and FOOTNOTE statements, then you invoke PROC GCHART with no labeling of the slices using the Annotate= data set that was created. The resulting pie is as large as the device allows, unrestricted by labeling orientation, you may want to reserve room for and print the legend at the bottom of the page in order not to restrict the room allotted for the pie.

**FINAL EXAMPLE**

```plaintext
/* Solves some common problems with GCHART pie statements */
/* Pie gets smaller as midpoints increase */
/* Labels can bump into each other */
/* What midpoints went into the "OTHER" slice */
/* Trouble controlling the use/order of color/pattern in pie */

/* Limit of twenty-one (21) discrete midpoints. */

/* Some data to work with */
/* (I think this was color distribution on a scanned EGA image) */

data freq;
  input pel count;
  cards;
  0  2
  1  6
  2  907
  3  650
  4  793
run;

/* Make this application a bit more generic */
%let MIDPOINT = pel;
%let RESPONSE = count;
%let OTHER = 4;
%let CTXT = WHITE;
%let FTXT = HEN;
%
include "&sysdir\annome."

/* If it wasn't already(same as GCHART) */
%MACRO bar( x1, y1, x2, y2, color, bartyp, pattern );
  DO;
    IF color = "NONE" THEN;
    ELSE color = "icolor" ;
    IF bartyp = " " THEN;
    ELSE style = "tpattern" ;
    IF pattern = " " THEN;
    ELSE FUNCTION = "BAR" ;
    output;
  ENDS;
%
%END;

/* Gather midpoint response statistics (frequency) */
proc summary data = freq; by MIDPOINT; run;
%
/* Gather midpoint response statistics (frequency) */
proc summary data = freq;
var RESPONSE;
output out=stat( keep= _fre4- total otherpct otherfrq otherval othernum )
  / , +------ -- - -- --- -- 
  includ"ivar\annomac";
/ * make this application a bit more generic */
%include "&sysdir\annome."
%
/* Order data by midpoint, if it wasn't already(same as GCHART) */
proc sort data=freq; by MIDPOINT; run;
%
/* Determine composition of "OTHER" category as GCHART would */
data stat( keep= _freq _otherperc _otherfrq _otherval _othernum )
  / , +------ -- - -- --- -- 
  input pel count;
  cards;
  0  2
  1  6
  2  907
  3  650
  4  793
run;
%
/* Determine composition of "OTHER" category as GCHART would */
data stat( keep= _freq _otherperc _otherfrq _otherval _othernum )
  / , +------ -- - -- --- -- 
  input pel count;
  cards;
  0  2
  1  6
  2  907
  3  650
  4  793
run;
%
/* Determine composition of "OTHER" category as GCHART would */
data stat( keep= _freq _otherperc _otherfrq _otherval _othernum )
  / , +------ -- - -- --- -- 
  input pel count;
  cards;
  0  2
  1  6
  2  907
  3  650
  4  793
run;
```

527
data pielgnd;
test freq;
put = RESPONSE / total;
if put < otherval then do;
otherval = otherval + 1;
otherpct = otherpct + pct;
otherfreq = otherfreq + count;
put ^=;
end;
else rot = put * 566;
output freq;
if ^=freq then output stat;
test;
/*
 Construct legend for PIE midpoints
 */
data pielgnd; /* File we write SAS global stmts to (included in-line later) */
/* This includes OPTIONS, FOOTNOTE, PATTERN etc. */
FILE ="/stlgilpielgnd.inc";
/* Initialization */
length statline $ 20;
length tnt $ 20;
array colors(21) $ 8;
array patt(21) $;
retain ystart pattl-patt21 colors1-colors21;
/* annotate variable definitions */
/* Variables we actually need for annotation */
keep ystart patt1-patt21 colors1-colors21;
/* First observation processing */
if ^=1 then do;
/* Put expected defaults into global file */
put OPTIONS LEGEND=NO VPOS=10;
/* Reserve space on the right for the legend */
put FOOTNOTE PATTERN=A50 XPOS=15 "";
put = ";
set stat; /* Composition of "OTHER" data */
legstart = 5 + otherval; /* Count in "OTHER" plus spacing */
/* Pattern statement control (control GCHART use of color) */
/* You can customize the entire display with changes here */
do i=1 to 7; patt(i)='GSOILD'; end;
do i=8 to 14; patt(i)='PINK'; end;
do i=15 to 21; patt(i)='PINK0'; end;
color; 1 = 'BLUE';
color; 2 = 'RED';
color; 3 = 'WHITE';
color; 4 = 'GREEN';
color; 5 = 'CHRM';
color; 6 = 'YELLOW';
color; 7 = 'WHITE';
do i=9 to 14; colors(i)=colors(1-7); end;
do i=15 to 21; colors(i)=colors(1-7); end;
/* Fabricate the appropriate PATTERN statements to file */
do i=1 to 21;
text = PATTERN [ leyieldput(i, t2.1) ]
'C' = [ leyieldput(i, t1.1) ]
'C' = [ leyieldput(i, t1.1) ]
'C' = [ leyieldput(i, t1.1) ]
'C' = [ leyieldput(i, t1.1) ]
'C' = [ leyieldput(i, t1.1) ]
'C' = [ leyieldput(i, t1.1) ];
put text;
end;
/* First observation processing */
/* Get midpoint data observation */
set freq;
/*
 Histogram grid

 Determination of location for box and stats text

 The f. format includes space between stats fields

 */
/**
 /**
 **/
else style = 'EMPTY';
end;
return;
run;

/* Let GCHART do the pie, and annotate our legend */
/* get our generated GLOBAL statements into effect */
%include '~/sugi/pielgend.inc';
proc gchart data=freq;
pie MIDPOINT / discrete freq=RESPONSE
  other=OTHER
  slice=none value=none /* turn off labels */
  annotate=pielgend;
run; quit;

SAS and SAS/GRAPH are registered trademarks of SAS Institute Inc. in the USA and other countries. © indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.
Figure 1 Default Bar Chart

Figure 2 Good Bar Chart
Figure 3  Better Bar Chart

Figure 4  Best Bar Chart
Precipitation measured in inches

Figure 5 PROC GPLOT Example
FREQUENCY of PEL

Statistics

<table>
<thead>
<tr>
<th>Mid</th>
<th>Resp</th>
<th>Pct</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>1514</td>
<td>4.67</td>
</tr>
<tr>
<td>14</td>
<td>2376</td>
<td>7.32</td>
</tr>
<tr>
<td>13</td>
<td>2656</td>
<td>8.19</td>
</tr>
<tr>
<td>12</td>
<td>3048</td>
<td>9.40</td>
</tr>
<tr>
<td>11</td>
<td>3418</td>
<td>10.54</td>
</tr>
<tr>
<td>10</td>
<td>3442</td>
<td>10.61</td>
</tr>
<tr>
<td>9</td>
<td>3038</td>
<td>9.37</td>
</tr>
<tr>
<td>8</td>
<td>2982</td>
<td>9.19</td>
</tr>
<tr>
<td>7</td>
<td>3226</td>
<td>9.95</td>
</tr>
<tr>
<td>6</td>
<td>2913</td>
<td>8.98</td>
</tr>
<tr>
<td>5</td>
<td>1473</td>
<td>4.54</td>
</tr>
</tbody>
</table>

OTHER composition

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>793</td>
<td>2.44</td>
</tr>
<tr>
<td>3</td>
<td>650</td>
<td>2.00</td>
</tr>
<tr>
<td>2</td>
<td>907</td>
<td>2.80</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0.01</td>
</tr>
</tbody>
</table>