

Using the SAS Annotate Facility for Creating Custom Graphs

Patrick M. McGown, FSD Data Services, Inc., Winston-Salem, NC

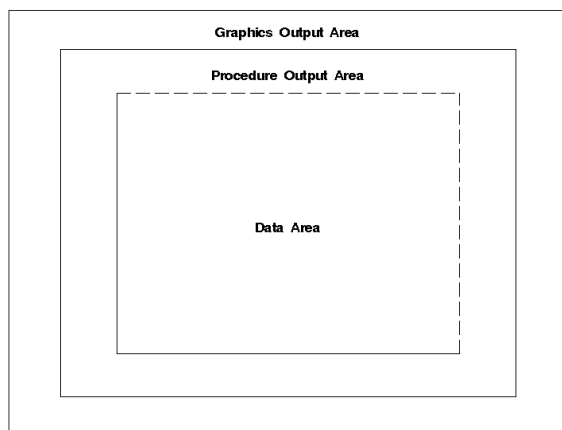
Introduction

With survey reporting, it is often necessary to produce graphical reports with pages of horizontal or vertical bar charts. Originally, these report applications used Proc Gchart to produce the graphs. However, when the customer changed the number of groups for a report, the application had to be modified to accommodate the new specifications. It became obvious that time was being wasted having to modify the report programs for different specifications. For that reason, it was decided to program the report applications using SAS Annotate. Combined with a macro loop, the system provided flexibility in placing different numbers of items and groups on any given page and being able to control all aspects of the formatting and presentation of the data.

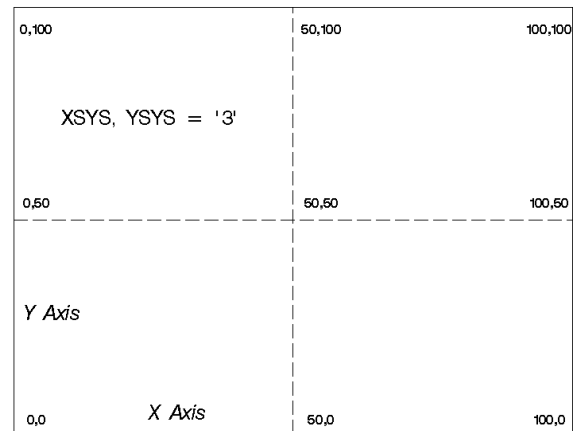
This paper will provide a brief introduction to the SAS Annotate Facility and some of its key features. The rest of the paper will work through building a sample graph from generating the summary data set to the final result.

Defining the Work Space

The features provided in the SAS Annotate Facility can be used either to modify graphics produced using the SAS Graph procedures or by itself to create custom graphics. This paper focuses on using Annotate as a separate tool. SAS defines three different regions on the page when producing graphics: the data area, the procedure output area and the graphics output area (pg. 476.) For the purpose of creating custom graphics, the graphics output area provides access to the entire page.



In addition, there is an absolute and relative coordinate system for each of the three areas. Within each area by system combinations, the coordinates can be percentage based or cell based. This paper utilizes the absolute percentage coordinate system in the graphics output area. In SAS code, this is specified by setting the coordinate system variables, xsys and ysys, to '3' (pp. 476-477.)



The numbers in the figure are percentages of the graphic output area of a landscape 8.5 x 11 page.

Starting the SAS Annotate Macros

Using the SAS Annotate Facility as a separate tool utilizes annotate macros. In order to use the macros, they must first be compiled. This is done by executing the following statement:

```
%annomac;
```

Once the macros have been compiled, the annotate data set can be created from the summary data set created earlier. A couple of statements need to be included in the data step, namely, the length and the retain statements. The text, function, color and style variables are used by the annotate macros and the retain statement sets the coordinate system to use for the macros.

```
data slide;  
  set sketch;  
  length text $200 function color style $8;  
  retain xsys ysys '3';
```

The data step above doesn't contain any instructions for the Annotate Facility and would not produce any graphics. The next section will begin to add the macros that produce the graphics.

Using the Annotate Macros

This paper will discuss three of the annotate macros, %label, %line and %bar. There are several more that can be used but are not presented in this paper. The syntax for the three macros is:

```
%label(x1,y1,text,color,ang,rot,ht,font,pos);  
%line(x1,y1,x2,y2,color,linetype,linewidth);  
%bar(x1,y1,x2,y2,color,bartype,pattern);
```

The x and y values used in the macros will be absolute percentages of the graphics output area. These can be

expressed explicitly or stored in variables. The details of the syntax for the three macros follows.

```
%label(x1,y1,text,color,ang,rot,ht,font,pos);
```

The text field contains the text to be printed on the page with the formatting specified by the rest of the fields. 'Ang' specifies the angle of the text string from horizontal and the 'rot' field slants the string from vertical. 'Ht' defines the size of the text and font specifies the font. The 'pos' or position is the placement of the text relative to the x,y coordinates specified. There are 15 possible placements for the text (pg. 522.)

```
%line(x1,y1,x2,y2,color,linetype,linewidth);
```

The line macro is fairly straightforward with the x1,y1 values providing the starting point of the line and the x2,y2 values defining the end point. There are 46 different line types including solid, dashed, dotted and combinations of dashes and dots (pg. 429.)

```
%bar(x1,y1,x2,y2,color,bartype,pattern);
```

As with the line macro, the x1,y1 values give the starting point of the bar and the x2,y2 values provide the end point. In the case of the bar, the x1,y1 would be the lower left hand corner of the bar and the x2,y2 would be the upper right. There are four different bar type options, 0-3, with 0 providing a solid bar, 1 giving a vertically adjusted bar, 2 outputs a horizontally adjusted bar and 3 creates a bar without a border. The pattern variable can be solid, empty or crosshatched. The crosshatching is defined by R|X|L and the density of the hatching goes from 1-5 with 5 being most dense.

The %label, %line and %bar macros can be used to create custom bar charts and displayed using Proc Gslide.

Creating the Summary Data

In report applications, the summary data sets are generated either using SAS procedures or in data steps. The following code creates the summary data set used to create the sample graphs. Perfav and perunf represent percent favorable and percent unfavorable respectively with perneu being percent neutral. N represents the sample size for each group.

```
data sketch;
length group $15;
input n perfav perunf group $;
perneu=100-(sum(of perfav,perunf));
cards;
112 45 22 Professional
105 23 40 Administration
74 39 50 Manufacturing
68 47 15 Executive
215 36 40 Warehouse
98 25 38 Other
;
```

```
run;
```

One Time Details

In generating graphs using the SAS Annotate Facility, there are items that need to be created once and items that will need to be created for every observation in the summary data set. Features such as titles and page numbers need to be drawn only once while group labels and bars need to be drawn for every observation in the summary data set. To create the one time details, the program executes those statements for only the first observation in the summary data set. See the example program below.

```
data slide;
set sketch;
length text $200 function color style $8;
retain xsys ysys '3';
if _n_=1 then do;
    *Include one time statements here;
end;
    *Rest of annotate statements here;
run;
```

For example, if a title is needed for the graph, the following code can be used:

```
data slide;
set sketch;
length text $200 function color style $8;
retain xsys ysys '3';
if _n_=1 then do;
    %label(50,95,'Title 1',black,0,0,3.5,swissb,5);
end;
run;
```

This code produces a data set that includes the commands from the macro statement to plot the title. To display the title, use Proc Gslide (pp.1261-1268) with the following syntax:

```
proc gslide annotate=slide; run;
```

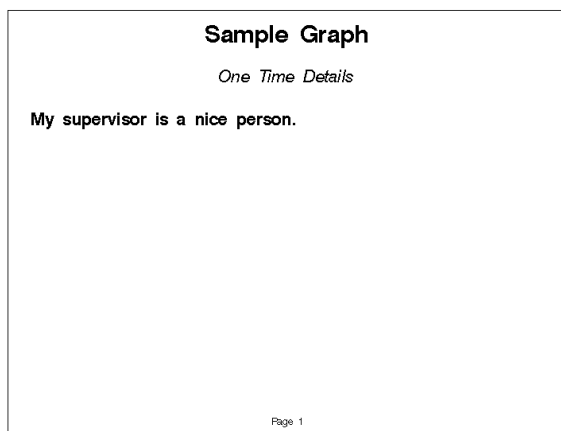
This statement performs all of the commands listed in the annotate data set and produces a slide based on those commands.

Title 1

If this statement was executed for all observations in the summary data set, the 'Title 1' data would be in six observations in the annotate data set and would be drawn onto the graph six times. This can take up space in the data set and time in resolving the graph.

The following example provides a more complete program displaying features that only need to be processed once such as item text and page number.

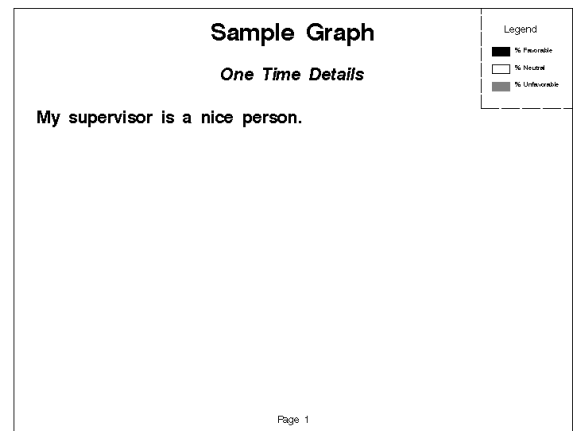
```
data slide;
set sketch;
length text $200 function color style $8;
retain xsys ysys '3';
if _n_=1 then do;
%label(50,95,'Sample Graph',black,0,0,3.5,swissb,5);
%label(50,85,'One Time Details',black,0,0,2.5,swiss,5);
%label(5,75,'My supervisor is a nice
person.',black,0,0,2.5,swissb,6);
%label(50,5,'Page 1',black,0,0,1.5,swiss,5);
%line(1,99,99,99,black,1,5);
%line(1,1,1,99,black,1,5);
%line(1,1,99,1,black,1,5);
%line(99,1,99,99,black,1,5);
end;
run;
```



In addition to titles, page numbers and frame lines, items like legends can also be included in this block of code. Inserting the following code into the program in the do-loop above results in the legend in the upper right hand corner of the page.

*Legend Definition;

```
%label(90,95,'Legend',black,0,0,1.5,swiss,5);
%bar(85,90,88,88,black,0,solid);
%bar(85,86,88,84,black,0,empty);
%bar(85,82,88,80,gray,0,solid);
%label(89,90,'% Favorable',black,0,0,1,swiss,6);
%label(89,86,'% Neutral',black,0,0,1,swiss,6);
%label(89,82,'% Unfavorable',black,0,0,1,swiss,6);
%line(83,76,99,76,black,5,3);
%line(83,76,83,99,black,5,3);
```



Depending on the type of presentation desired, more or fewer details can be used.

Multi-Observation Statements

The macro statements used to create the bars need to be executed for each observation in the summary data set. There are several components that make up the bars: the group label, the three bar components and the numeric labels for each part of the bar. Since these need to be run for each observation, the statements will follow the do-loop containing the one time details. One important part of this process is retaining the value of y as the statements are processed. To do this, the value of y is retained using a retain statement and a starting point is set using the y=75 statement prior to leaving the do-loop. The following code places the group labels down the left hand side of the page.

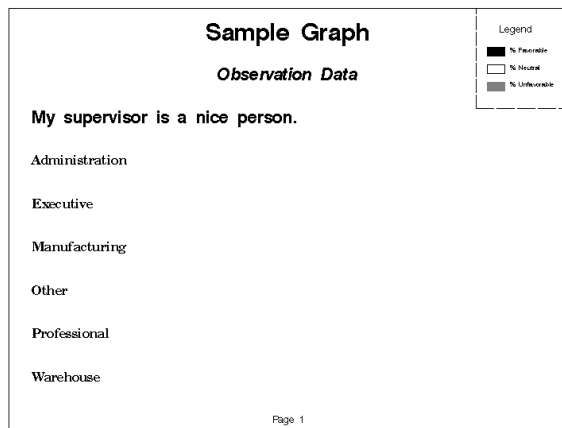
```
data slide;
set sketch;
length text $200 function style color $8;
retain xsys ysys '3';
if _n_=1 then do;
%label(50,95,'Sample Graph',black,0,0,3.5,swissb,5);
%label(50,85,'Observation Data',black,0,0,2.5,swissbi,5);
%label(5,75,'My supervisor is a nice
person.',black,0,0,2.5,swissb,6);
%label(50,5,'Page 1',black,0,0,1.5,swiss,5);
%line(1,99,99,99,black,1,5);
%line(1,1,1,99,black,1,5);
%line(1,1,99,1,black,1,5);
%line(99,1,99,99,black,1,5);
```

*Legend Definition;

```
%label(90,95,'Legend',black,0,0,1.5,swiss,5);
%bar(85,90,88,88,black,0,solid);
%bar(85,86,88,84,black,0,empty);
%bar(85,82,88,80,gray,0,solid);
%label(89,90,'% Favorable',black,0,0,1,swiss,6);
%label(89,86,'% Neutral',black,0,0,1,swiss,6);
%label(89,82,'% Unfavorable',black,0,0,1,swiss,6);
%line(83,76,99,76,black,5,3);
%line(83,76,83,99,black,5,3);
y=75;
end;
retain y;
y=y-10;
%label(5,y.group,black,0,0,2,centb,6);
run;
```

The code added to the previous program (underlined) sets the starting y value at 75 before ending the do-loop.

When SAS processes the next observation, the value of y goes to 65 and the label is printed on the page. The next observation will have a y value of 55 and continues until all of the observations in the summary data set are processed. The added code results in this graph:

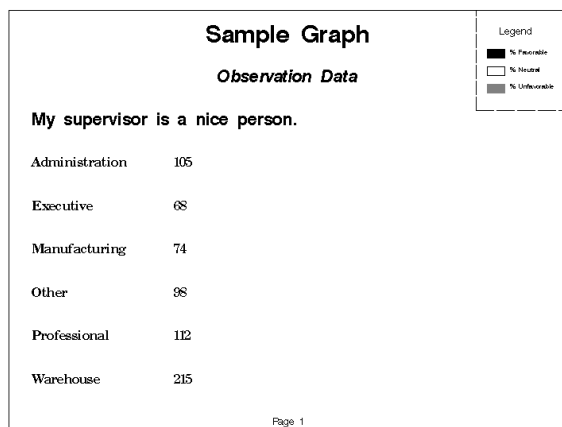


By using the vertical coordinate value stored in the y variable, it takes only one label command to put the label for each of the six groups on the page. Likewise, by adding the following code to the above program after the last label macro statement, it will add the sample size for each group to the graph.

```
%label(30,y,left(put(n,3.0)), black,0,0,2,centb,6);
```

The text field is 200 characters long and without the 'left' function, the numbers would be plotted off the page to the right.

This results in the following graph:



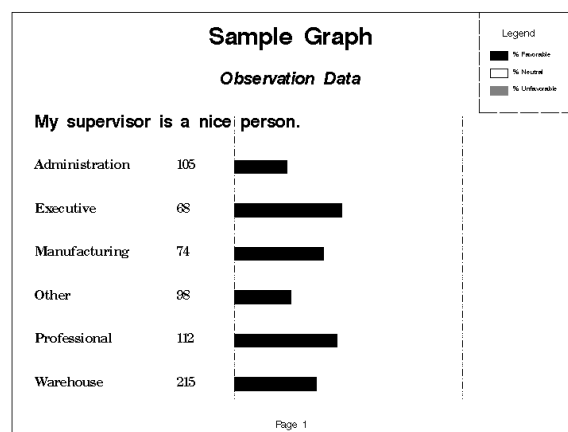
for each segment. The statement for plotting the percent favorable bar is:

```
%bar(40,y,40+(40*(perfav/100)),  
      y-3,black,0,solid);  
y=y+3;
```

The first x value, 40, is the starting value of the plotting area. The second x value is the proportion of the plotting area for the percent favorable value. The equation for this value is:

$$40+(40*(\text{perfav}/100))$$

This equation places the perfav value on the scale of the plotting area and then adds that value to the starting value of the plotting area. The y=y+3; is added to maintain the spacing between the groups on the page. If this value was not added back in, the lower groups would plot off the page and the spacing between them would increase by 3%. The above statements added to the program result in:



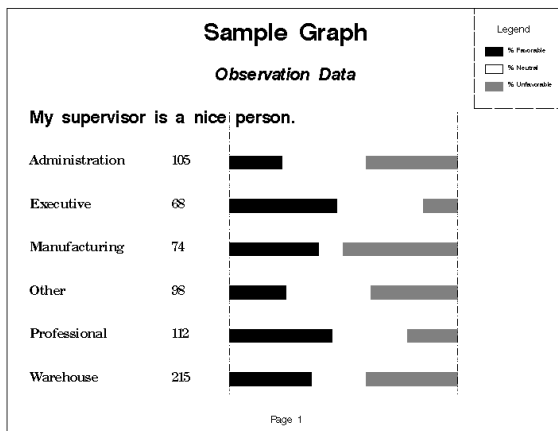
The vertical dotted lines represent x=40 and x=80 and are put there for reference only. The bar for percent unfavorable is plotted in the same fashion but instead of starting at x=40 and adding some value, the bar will start at x=80 and subtract the scaled percentage of the plotting area. The statement for the percent unfavorable bar is:

```
%bar(80,y,80-(40*(perunf/100)),y+3, gray,0,solid);  
*y=y+3;
```

The y=y+3; is commented out since the 3 % is added back during this %bar statement. Adding this statement produces the following:

Creating the Bars

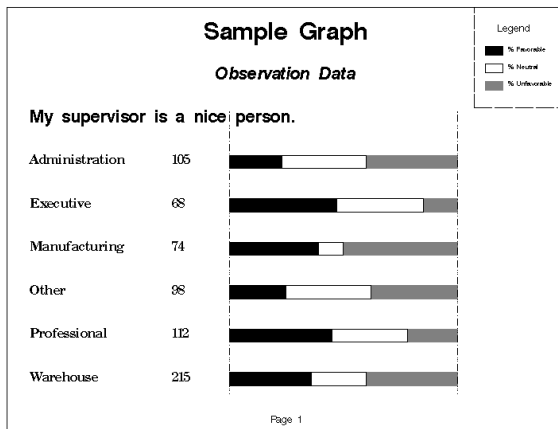
The first step in creating the bars is defining a plotting area for the bars. For this graph, the starting point will be x=40 and the finishing point will be x=80. This provides the system with 40% of the page width for plotting the bars. Determining the length of the bar segments is a matter of calculating the percentage of the plotting area



The final step in completing the bars is plotting the percent neutral bar in the middle. This statement uses both equations in determining the right and left hand side of the bar. The code for the percent neutral bar is:

```
%bar(40+(40*(perfav/100)),y,  
      80-(40*(perunf/100)), y-3,black,0,empty);  
y=y+3;
```

The first x value is the same as the x value calculated for the right hand side of the percent favorable (black) bar. The second value is the same as the x value calculated for the left hand side of the percent unfavorable (gray) bar. The 'empty' option will give a white box with black line borders. Once again, the y=y+3; must be used to maintain the spacing between the groups. The graph appears like this:



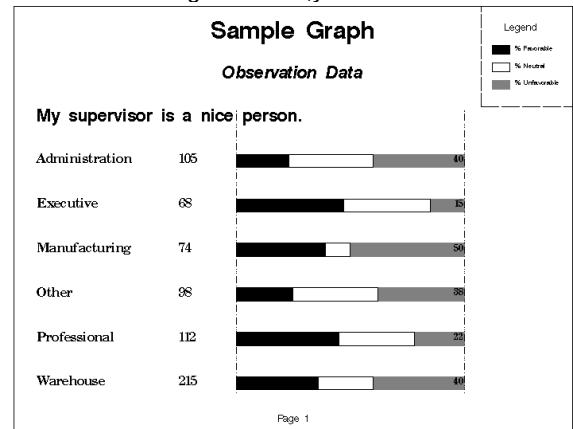
Adding the Labels for the Bars

The last step for completing the graph is inserting the percentage values into the bars. This is done in the same manner as calculating the start and end points for the bars. The easiest to plot is the percent unfavorable (gray) value. This value will be plotted to the right of the left hand edge of the gray bar. The statement is:

```
%label(80,y-.25,left(put(perunf,3.0)),  
        black,0,0,1.5,centb,4);
```

The .25 is subtracted from y to bring the number down a little from the top of the bar. Once again, without the 'left'

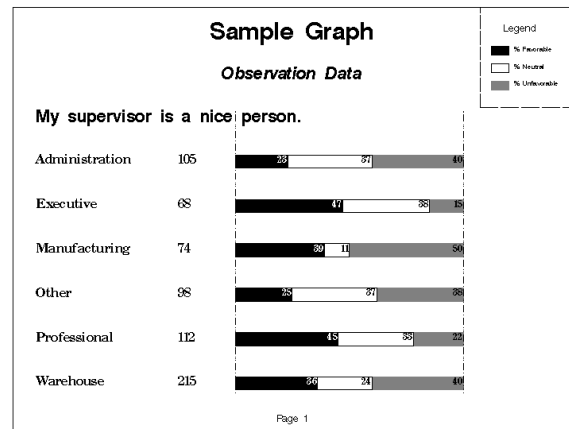
statement, the text would print off the page since the text field is 200 characters long. The '4' position option places the label to the right of the x,y coordinate. The results is:



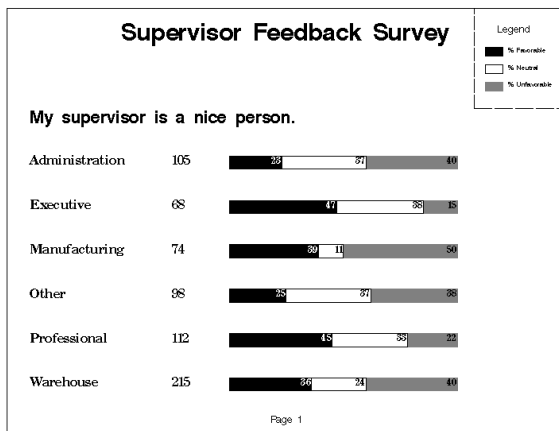
Adding the labels for percent favorable and percent unfavorable is done in the following code:

```
%label(40+(40*(perfav/100)),y,left  
        (put(perfav,3.0)),white,0,0,1.5,centb,4);  
%label(80-(40*(perunf/100)),y,left  
        (put(perneu,3.0)),black,0,0,1.5,centb,4);
```

The x value for the percent favorable label uses the value for the right hand side of the bar and then the '4' position option used before plots the text to the left. The percent neutral statement uses the same syntax but is plotted at the left hand side of the percent unfavorable (gray) bar. The two statements added produces:



After removing the reference lines from the program and a title change, the final graph appears as:



The code to produce this final graph using the data set created earlier is:

```
data slide;
set sketch;
length text $200 function style color $8;
retain xsys ysys '3';
if _n_=1 then do;
  %label(50,95,'Supervisor Feedback
    Survey',black,0,0,3.5,swissb,5);
  %label(5,75,'My supervisor is a nice
    person.',black,0,0,2.5,swissb,6);
  %label(50,5,'Page 1',black,0,0,1.5,swiss,5);
  %line(1,99,99,99,black,1,5);
  %line(1,1,1,99,black,1,5);
  %line(1,1,99,1,black,1,5);
  %line(99,1,99,99,black,1,5);

  *Legend Definition;

  %label(90,95,'Legend',black,0,0,1.5,swiss,5);
  %bar(85,90,88,88,black,0,solid);
  %bar(85,86,88,84,black,0,empty);
  %bar(85,82,88,80,gray,0,solid);
  %label(89,90,'% Favorable',black,0,0,1,swiss,6);
  %label(89,86,'% Neutral',black,0,0,1,swiss,6);
  %label(89,82,'% Unfavorable',black,0,0,1,swiss,6);
  %line(83,76,99,76,black,5,3);
  %line(83,76,83,99,black,5,3);
  y=75;

end;
retain y;
y=y-10;
%label(5,y,group,black,0,0,2,centb,6);
%label(30,y,left(put(n,3.0)),black,0,0,2,centb,6);
%bar(40,y,40+(40*(perfav/100)),y-3,black,0,solid);
%bar(80,y,80-(40*(perunf/100)),y+3,gray,0,solid);
%bar(40+(40*(perfav/100)),y,80-(40*(perunf/100)),y-
  3,black,0,empty);
y=y+3;
%label(80,y-.25,left(put(perunf,3.0)),black,0,0,1.5,centb,4);
%label(40+(40*(perfav/100)),y,left(put(perfav,3.0)),white,0,0,1.5,centb,4);
%label(80-
  (40*(perunf/100)),y,left(put(perneu,3.0)),black,0,0,1.5,centb,4);
run;

proc gslide anno=slide;
run;
```

Adding Flexibility to the System

In this example, the six bars for the one item fit on one page, but in actual applications, most of the reports either put multiple items on one page or there are so many groups for each item, the items have to be split between multiple pages. Using a macro variable to designate the total number of groups and a macro loop, it is possible to dynamically calculate the number of items and/or groups to put on each page.

Determining the Number of Groups

The first step in this process is to find out how many groups there are for each item. By running a frequency of the group variable and outputting the results to a dataset, the value of `_N_` for the last observation will indicate how many groups there are and put this value to a macro variable.

```
proc freq data=sketch noprint;
tables group/out=counts;
run;
```

```
data _null_;
set counts end=final;
if final=1 then do;
  call symput('ngroups',_n_);
end;
run;
```

The macro variable `NGROUPS` now has a value of 6, representing the total number of groups for each item.

Adding an Index Variable

In order to utilize the `NGROUPS` variable, a unique index value for each group is needed so that the macro loop can select the groups for each page. A quick way to accomplish this is to sort the data by the group variable and then create a variable called `order` to store the unique identifier. This will also result in the data being ordered alphabetically by group.

```
proc sort data=sketch;
by group;
run;

data sketch;
set sketch;
by group;
retain order;
if _n_=1 then order=0;
if first.group then order=order+1;
run;
```

Creating the Macro Loop

By utilizing a macro and a macro do loop along with the macro variables, `MIN`, `MAX`, `NUMBER`, `NGROUPS`, and `PAGE`, the system can dynamically put different bars on separate pages. `NGROUPS` is the number of groups determined by the frequency procedure in an earlier step and is equal to 6. The `MIN` variable stores the first observation in the range of observations to keep for each page. For the first page, `MIN=1` and in the second page, `MIN=4`. `MAX` stores the last observation in the range of observations to keep for each page. For the first page,

MAX=3 and MAX=6 for the second page. The MIN and MAX values provide the range of valid values of the order variable to select for each page. The first page presents the groups where order equals 1, 2, or 3. The second page presents the groups where order equals 4, 5, or 6. NUMBER represents the number of bars per page minus 1. In this example, it is set to 2, resulting in three bars per page. PAGE stores a count of the pages and is used to put the page number at the bottom of the page using the Label macro in the program below.

```
%macro pages;
%let min=1;
%let max=;
%let number=2;
%let page=0;
%do %until(&max>=&ngroups);
data _null_;
call symput('max',&min+&number);
run;
data temp;
set sketch(where=(&min<=order<=&max));
call symput('page',&page+1);
run;
data slide;
set temp;
length text $200 function style color $8;
retain xsys ysys '3';
if _n_=1 then do;
%label(50,95,'Supervisor Feedback Survey',black,0,0,3.5,swissb,5);
%label(5,75,'My supervisor is a nice person.',black,0,0,2.5,swissb,6);
%label(50,5,'Page '||left(trim("&page"))',black,0,0,1.5,swiss,5);
%line(1,99,99,99,black,1,5);
%line(1,1,1,99,black,1,5);
%line(1,1,99,1,black,1,5);
%line(99,1,99,99,black,1,5);

*Legend Definition;

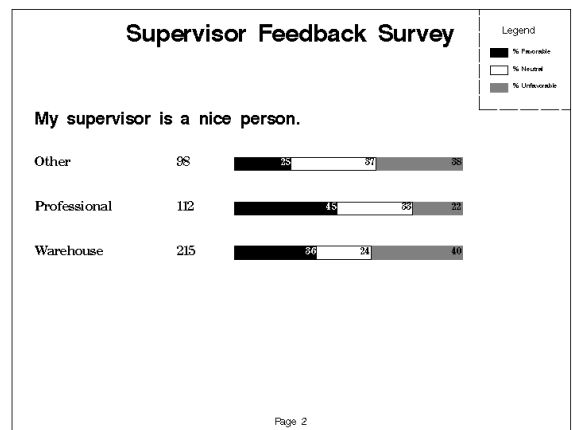
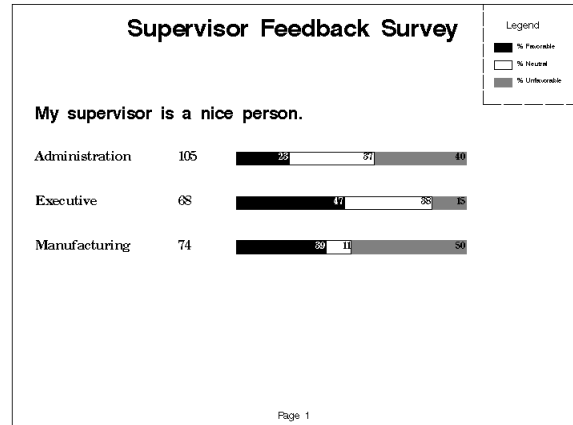
%label(90,95,'Legend',black,0,0,1.5,swiss,5);
%bar(85,90,88,88,black,0,solid);
%bar(85,86,88,84,black,0,empty);
%bar(85,82,88,80,gray,0,solid);
%label(89,90,'% Favorable',black,0,0,1,swiss,6);
%label(89,86,'% Neutral',black,0,0,1,swiss,6);
%label(89,82,'% Unfavorable',black,0,0,1,swiss,6);
%line(83,76,99,76,black,5,3);
%line(83,76,83,99,black,5,3);
y=75;
end;
retain y;
y=y-10;
%label(5,y,group,black,0,0,2,centb,6);
%label(30,y,left(put(n,3.0)),black,0,0,2,centb,6);
%bar(40,y,40+(40*(perfav/100)),y-3,black,0,solid);
%bar(80,y,80-(40*(perunf/100)),y+3,gray,0,solid);
%bar(40+(40*(perfav/100)),y,80-(40*(perunf/100)),y-3,black,0,empty);
y=y+3;
%label(80,y-.25,left(put(perunf,3.0)),black,0,0,1.5,centb,4);
%label(40+(40*(perfav/100)),y,left(put(perfav,3.0)),white,0,0,1.5,centb,4);
%label(80-(40*(perunf/100)),y,left(put(perneu,3.0)),black,0,0,1.5,centb,4);
run;

proc gslide anno=slide;
run;
```

quit;

```
data _null_;
call symput('min',&max+1);
run;
%end;
%mend;
%pages;
```

The above program results in the following two graphs.



As specified using the macro variable, the system plots three groups per page and the page number automatically increments by one. If the value of NUMBER were set to 1, the system would generate three pages with two groups per page.

New Feature in V8: Including an Image

In version 8, the Annotate Facility has added an image function, allowing for the inclusion of an image on SAS Graphs. The following code places an image called 'fsdlogo.gif' on the center of the page. The two x and y values define the area for the image to be placed. The image file is fit into the space defined by the x,y pairs.

```
x=40; y=65; function='move'; output;
x=60; y=35; imgpath='fsdlogo.GIF';
style = 'fit';
function='image'; output;
```



This slide was produced using the following program:

```
data slide;
  set sketch;
  length text $200 function style color $8;
  retain xsys ysys '3';
  if _n_ = 1 then do;
    %label(50,95,'Sample of Image File',black,0,0,2,swiss,5);
    x=40; y=65; function='move'; output;
    x=60; y=35;
    imgpath="d:\projects\ssu2001_anno\fsdlogo.GIF";
    style = 'fit';
    function='image'; output;
    %line(1,1,99,1,black,1,5);
    %line(1,1,1,99,black,1,5);
    %line(1,99,99,99,black,1,5);
    %line(99,1,99,99,black,1,5);
  end;
run;
```

Limitations

While this technique is very useful in creating graphs that are completely customized, programming annotate graphs can be a time consuming process. The placement of objects and text may take many iterations of trying different values. This technique is also susceptible to exceptions that don't work with the above code like zero percentage values and text strings that overlap bars. These exceptions can be solved through more detailed programming.

Conclusions

This paper examined a technique using the SAS Annotate macros for creating custom graphs. While the programming can be time consuming, once the code is finalized, you can have a powerful tool for creating graphs that are completely customizable and can end the frustration of trying to label, place titles and create legends in the SAS Graph Procedure output.

References

SAS Institute Inc. (1990), *SAS/GRAPH® Software: Reference, Version 6, First Edition, Volumes 1 & 2*, Cary, N.C.: SAS Institute

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

Contacting the Author

Please direct any questions or feedback to the author at:

FSD Data Services, Inc.
1001 S. Marshall Street
Suite 125, Box 25
Winston-Salem, NC 27101

E-mail: patrickm@fsddatasvc.com