

Making SAS Graphs Presentable

Andrew Beggs, Amadeus Software Limited



ABSTRACT

The SAS/GRAPH® product is a formidable piece of software that allows you to generate more or less any type of chart that you can think of. However, the procedure is somewhat disjointed, meaning that many of its strengths and versatility are often overlooked or even shied away from. As a result, graphs are left looking rather plain and lacking in features that we take for granted when producing graphs using other software. This paper aims to show you the additional tools that are available for making SAS graphs more presentable in terms of their general appearance, axis labelling options and some of the features relating to specific types of graph.

INTRODUCTION

Making a SAS graph look sharp and easy to interpret can be accomplished by making appropriate use of the wide range of SAS/GRAPH® statements and options. Being aware of what is available is half the battle and what follows aims to highlight the key points. Although SAS/GRAPH® may not be your first choice for generating charts, it offers a lot of control over the final output and is a sound logical choice for generating graphical output from a SAS based application.

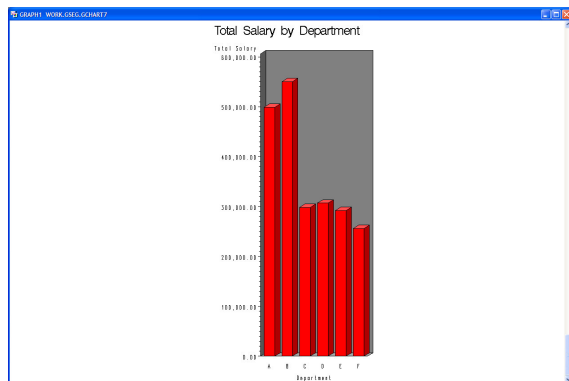
AXIS LABELLING

The basis of any meaningful graph lies in setting up appropriate axis scales and labels. Each axis of a chart is defined separately via an Axis statement. Axis definitions are stored in memory and once defined, are available to all types of graph procedure. The axis statement can take a number of forms:

To simply define some axis labels:

```
axis1 label=('Department');
axis2 label=('Total Salary');

title 'Total Salary by Department';
proc gchart data=views.demog;
  vbar3d dept / sumvar=salary type=sum
  maxis=axis1 raxis=axis2;
run;
```

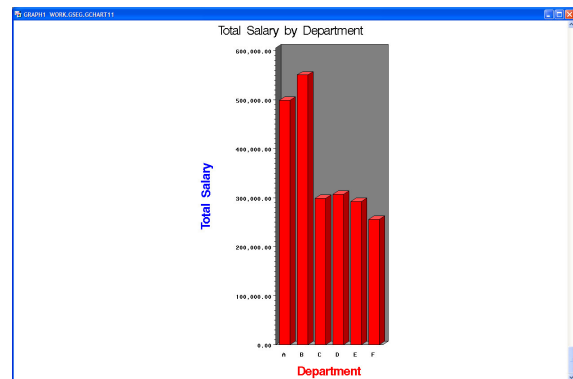


Note that you need to increment the axis number for each successive definition.

The axis label for the vertical axis can be aligned vertically by including the ANGLE= option in the axis definition. You can also set the font, height and colour of the axis label:

```
axis1 label=(font=SwissB height=2 color=red
'Department');
axis2 label=(angle=90 font=SwissB height=2
color=blue 'Total Salary' );
```

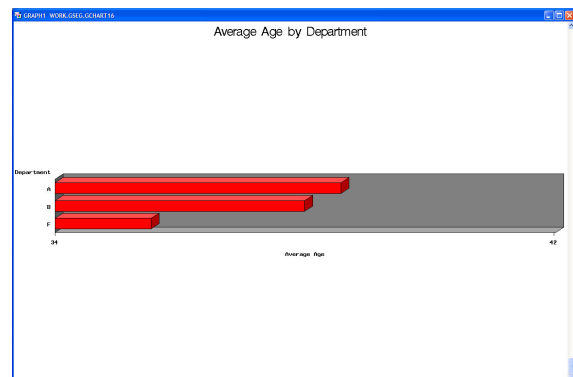
Note that the font, height and color options must be specified before the axis label, otherwise they are ignored.



The axis points can be controlled using the ORDER= option. This can take a number of forms:

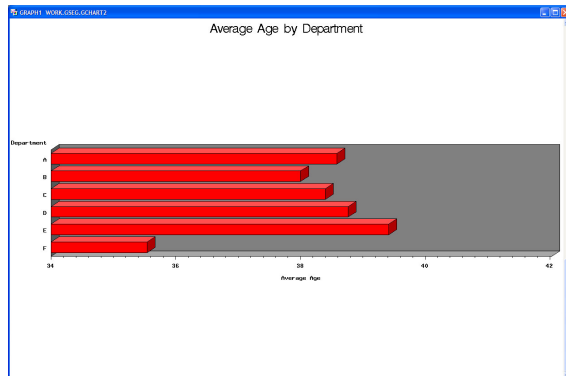
List of actual values in required order:

```
axis1 label=('Department')
order=('A' 'B' 'F');
axis2 label=('Average Age')
order=34,42;
```



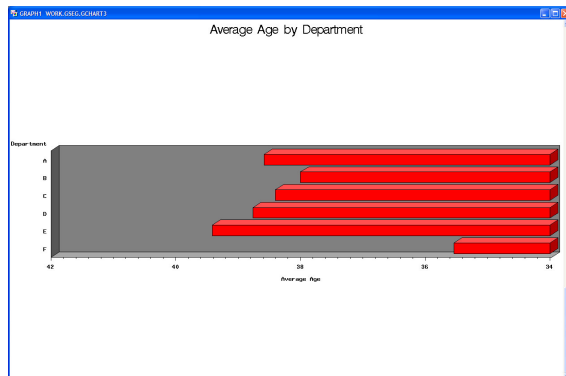
Range of values :

```
axis2 label= ('Average Age')
order=34 to 42 by 2;
```



Range of values in reverse order:

```
axis2 label= ('Average Age')
order=42 to 34 by -2;
```

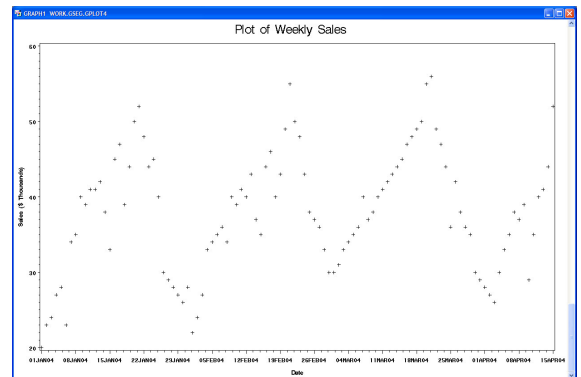


The ORDER= option can also be used to specify an axis scale based on date and time intervals. The following example uses the WEEK interval:

```
axis1 label=(font=SwissB 'Date')
order=('01JAN04'd to '15APR04'd by WEEK);

axis2 label=(angle=90 font=SwissB 'Sales ($ Thousands)');

title 'Plot of Weekly Sales';
proc gplot data=views.dates;
  plot sale*date / haxis=axis1 vaxis=axis2;
  format date date7.;
run;
```



A similar plot by month:

```
axis1 label=(font=SwissB 'Month')
order=('01JAN04'd to '15APR04'd by MONTH);
```



However, note the message in the Log window:

```
WARNING: The intervals on the axis labeled
date are not evenly spaced.
WARNING: No minor tick marks will be drawn
because major tick increments have been
specified in
          uneven or unordered intervals.
```

The above message results because months have a varying number of days and therefore the axis points (tick marks) are not evenly spaced.

Also note that, because the data only goes as far as 15th April and therefore does not represent a complete month, none of the sales for April have been plotted. The only way around this is to increase the date range on the ORDER= option, so that it goes as far as the 1st May:

```
axis1 label=(font=SwissB 'Date')
order=('01JAN04'd to '01MAY04'd by MONTH);
```



The above examples show the most commonly used axis options, but there are others that are worth investigating.

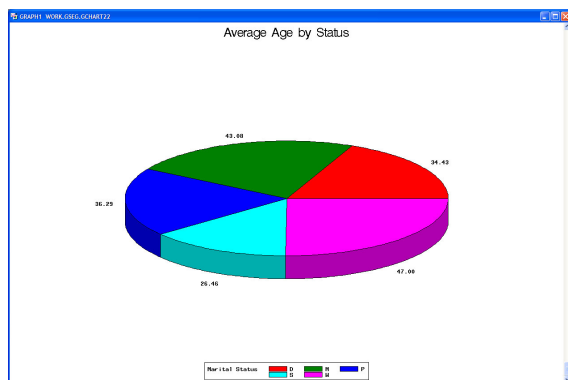
LEGENDS

When generating graphs that involve groups or sub groups of data, the legend is paramount in making sense of the chart. Although SAS will generate a default legend when required, it will not be labeled, its position may not be ideal and its layout may intrude on the graph itself. All of these settings and others can be controlled through the use of a Legend statement. Like axis definitions, legends are stored in memory and once defined can be used by any graph procedure.

The following example of a legend statement assigns a legend label, specifies the layout in terms of 2 rows and 3 columns, encloses the legend in a box and positions it at the bottom of the output:

```
legend1 label=('Marital Status') down=2
across=3 frame position=(bottom center);

title1 "Average Age by Department";
proc gchart data=views.demog;
  pie3d status / sumvar=age type=mean
  value=outside legend=legend1 noheading;
run;
```



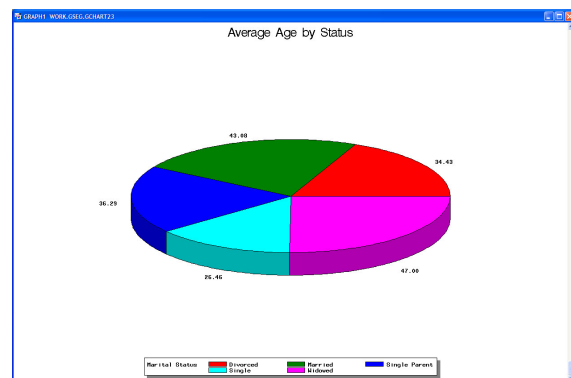
If the specified legend layout does not define enough positions, then it is ignored and the default layout is used.

To include a background effect on a legend you can use the CSHADOW option:

```
legend1 label=('Marital Status') down=2
across=3 frame cshadow=gray;
```

To specify descriptions for each individual legend value, the VALUE= option is included in the legend definition:

```
legend1 label=('Marital Status') down=2
across=3 frame cshadow=gray
value=(t=1 'Divorced' t=2 'Married' t=3
'Single Parent' t=4 'Single' t=5 'Widowed');
```



Note that the order of the values must match the order in which they are displayed on the chart. With pie charts and star charts, segments are ordered anticlockwise starting at the 3 o'clock position, so legend value labels need to be specified in that order.

The font and colour of the legend values can be specified using the FONT= and COLOR= options:

```
legend1 label=('Status') down=2 across=3
frame cshadow=gray
value=(font=CentB color=blue t=1 'Divorced'
t=2 'Married'
t=3 'Single Parent' t=4 'Single' t=5
'Widowed');
```

Note that the to control the font and color for all legend values, they must be specified before the list of legend descriptions, otherwise they are ignored.

To specify the font and colour of individual legend values, use the following approach:

```
legend1 label=('Marital Status') down=2
across=3 frame cshadow=gray
value=(
t=1 font=CentB color=red 'Divorced'
t=2 font=Swiss color=green 'Married'
t=3 font=SwissB color=blue 'Single Parent'
t=4 font=Arial color=cyan 'Single'
t=5 font=CentB color=magenta 'Widowed');
```

```
value=outside percent=inside;
run;
```

ENHANCING SPECIFIC TYPES OF GRAPH

There are many options available within the individual graph types for controlling the structure and appearance of a chart and it's worthwhile scanning the reference material to get a feel for what is available. In this section we'll look at a few options relating to specific types of chart, as well as two other types of statement, that can be used with certain graph procedures to make the output a lot more presentable.

BAR CHARTS

The following example enhances a bar chart by specifying the following options:

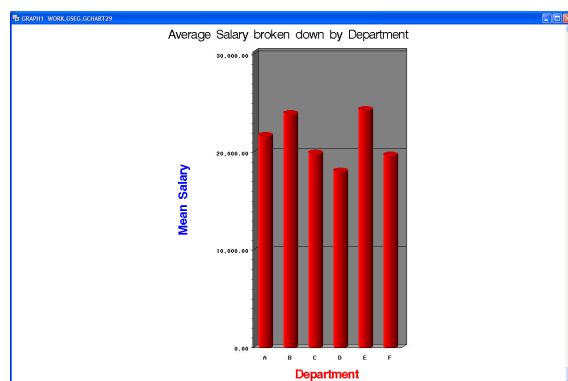
- Bar Shape (3D only)
- Bar Spacing
- Reference Lines

```
proc gchart data=views.demog;
  vbar3d dept / sumvar=salary type=mean
    maxis=axis1 raxis=axis2
    shape=cylinder space=3
    autoref;
run;
```

The SHAPE= option requests a bar shape other than the default block shape. Other options include: Prism, Star and Hexagon.

The SPACE= option specifies the space between the bars in terms of the number of character cells. If the value is too high to accommodate all of the bars then the option is ignored.

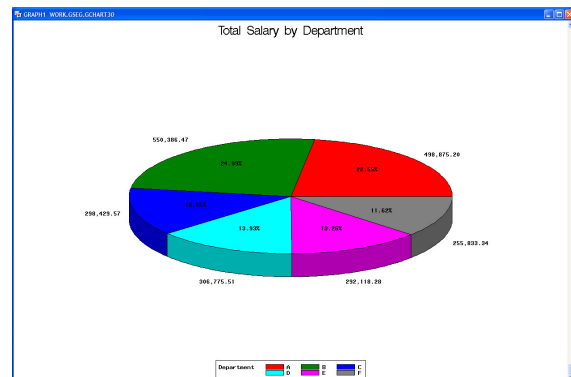
The AUTOREF option requests that reference lines are drawn at the major points on the vertical axis. To request reference lines at specific points, use the REF= option, where you specify a list of values.



PIE CHARTS

When creating a pie chart that needs to display more than one statistic, the position of the specified statistic and the percentage statistic can be controlled using the VALUE= and PERCENT= options as shown in the following example:

```
proc gchart data=views.demog;
  pie3d dept / sumvar=salary type=sum
  legend=legend1 noheading
```



USE OF SYMBOL STATEMENTS WITH SCATTER PLOTS AND CONTOUR PLOTS

The shape, style and colour of data points and plot lines generated by the GPLOT and GCONTOUR procedures can be controlled through the use of Symbol statements.

Symbol statements are stored in memory and once defined can be used by any GPLOT or GCONTOUR procedure.

With GPLOT, symbol statements can be used to set the appearance of data points and lines as well as the interpolation method. With GCONTOUR, symbol statements are restricted to setting the appearance of contour labels and lines.

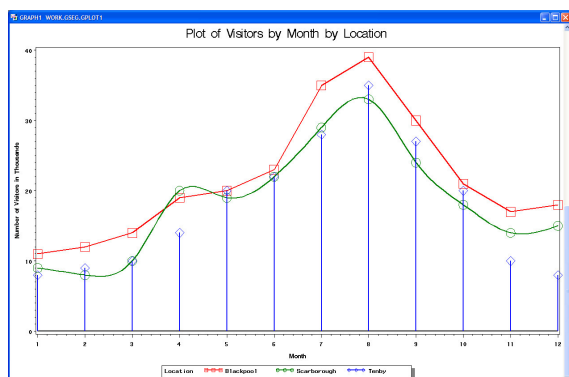
The following example shows three symbol statements that define different data point styles and colours, together with different interpolation options for use with a GPLOT:

```
symbol1 color=red interpol=join
  width=2 value=square
  height=3;
symbol2 color=green interpol=spline
  width=2 value=circle
  height=3;
symbol3 color=blue interpol=needle
  width=2 value=diamond
  height=3;

proc gplot data=tourist;
  plot visitors*month=location/
  legend=legend1
  vaxis=axis1 haxis=axis2;
run;
quit;
```

Note that the symbol definitions are not referenced within the procedure itself.

Note the use of the PATTERNID option within the procedure, to force a change in fill pattern when the value of the group variable changes.



Interpolation options:

- Join causes points to be linked by straight lines.
- Spline gives a smooth curve through the points.
- Needle connects each point to the baseline.

USE OF PATTERN STATEMENTS

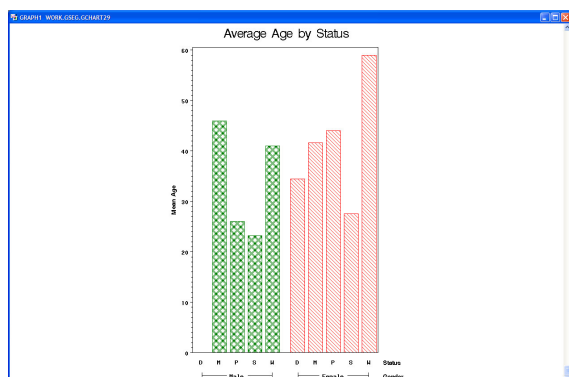
Pattern statements can be used with a range of graph types, but are most commonly used with the GCHART procedure to control the colour and fill pattern of bars, pie slices, blocks etc. Pattern statements are stored in memory and remain in effect for the session or until they are re-assigned.

The following example shows how pattern statements are used to control the appearance of the bars:

```
axis1 label=(font=SwissB height=1 'Status');
axis2 label=(angle=90 font=SwissB height=1 'Mean Age' );
axis3 label=(font=SwissB height=1 'Gender')
value=('Male' 'Female');
title 'Average Age by Status';

pattern1 color=green value=x4;
pattern2 color=red value=Li;

proc gchart data=views.demog;
vbar status / group=gender sumvar=age
type=mean patternid=group
maxis=axis1 raxis=axis2 gaxis=axis3;
run;
```



The following fill pattern options can be used:

- Li = left diagonal lines
- Ri = right diagonal lines
- Xi = hatched lines
- S = Solid
- E = Empty

Where "i" is a value of 1 to 5 that controls the strength of the line.

ODS AND DEVICE DRIVERS

Having created the required graph, the final stage is to decide on how the chart is to be output. By default, the graph is output to a SAS catalog in the form of a GSEG member, but it is likely that you will want your chart output to a particular type of external file such as RTF(Word), HTML or PDF for example. This can be accomplished using ODS (Output Delivery System) statements. In addition, you need to select a graphics device, which will control the overall format of the chart.

To set the graphics device, use the GDEVICE= option within a GOPTIONS statement. The GOPTIONS statement is used to apply global graph options. Once set, they remain in effect for the session, or until their values are changed.

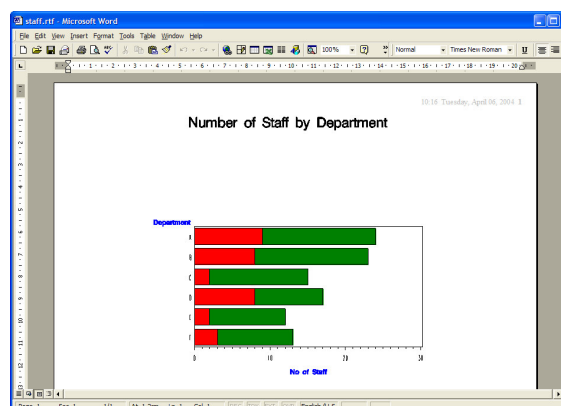
To control the output file type, use ODS statements in the same way as you would for other procedure output.

The following example specifies a graphics format of GIF and includes ODS statements which direct the output to a Rich Text Format (RTF) file:

```
goptions device=gif;
ods listing close;
ods rtf file='c:\temp\staff.rtf';
```

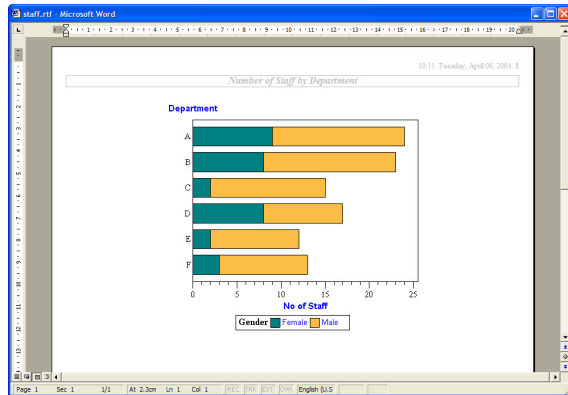
```
proc gchart data=views.demog;
hbar dept / subgroup=gender legend=legend1
maxis=axis1 raxis=axis2 noheading nostats;
run;
```

```
ods rtf close;
ods listing;
```

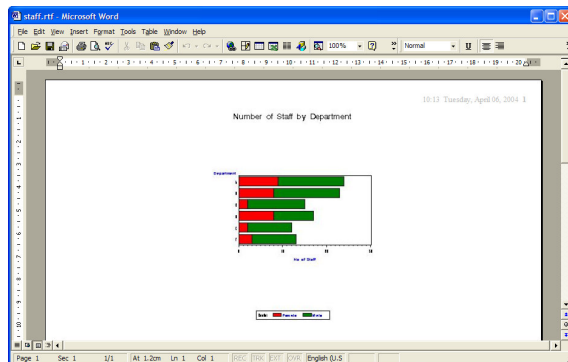


Simply changing the graphics device has a significant effect on the appearance of the chart:

```
goptions device=java;
```



```
goptions device=jpeg;
```



REFERENCES

SAS System Help
SAS System Catalogs

CONTACT INFORMATION

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

Author Name	Andrew Beggs
Company:	Amadeus Software Limited
Address:	Orchard Farm, Witney Lane, Leafield, Oxon OX29 9PG
Work Phone:	01993 878287
Fax:	01993 878042
Email:	Andrew.Beggs@amadeus.co.uk
Web:	www.amadeus.co.uk

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. © indicates USA registration.

Other brand and product names are trademarks of their respective companies.

SUMMARY

Hopefully, this paper has proved useful in showing you the tools that are required to make SAS graphs presentable. The key to a decent graph is:

- Define the axes correctly.
- Specify a legend where required.
- Take advantage of the many options that are available for the specific graph type.
- Choose an appropriate graphics format and output the chart to the required file type.

This paper has only scratched the surface in terms of the wide range of SAS/GRAPH® statements and options that are available. It is therefore worthwhile scanning through the extensive reference material that is available in this area, to get a more complete picture of what is available. The paper has not mentioned the Annotate facility, which allows you to label individual data points on a chart. This is well worth getting to grips with and is not all that difficult to use.