

Graphing in SAS Software

Prepared by



International SAS® Training and Consulting

Destiny Corporation – 100 Great Meadow Rd Suite 601 -
Wethersfield, CT 06109-2379

Phone: (860) 721-1684 - 1-800-7TRAINING

Fax: (860) 721-9784

Email: info@destinycorp.com

Web: www.destinycorp.com

Copyright © 2001 Destiny Corporation

Proc Gplot

Proc GPLOT is the standard Sas procedure that allows us to create graphs in plot form.

The GPLOT procedure allows you to plot one variable against another, each pair derived from the same observation in the input data set.

We can produce relatively simple plots using a few statements and then enhance the result.

With GPLOT we can do the following:

- Draw reference lines on the plot
- Overlay plots
- Use any symbol to represent the points
- Reverse the order on the vertical scale
- Plot character variables (<= length 16)
- Select colors, symbols, interpolation methods, line styles
- Produce 'bubble' charts
- Plot a second vertical axis
- Produce logarithmic plots

Simple Plots

Proc GPLOT uses the Share Price Data Set and produces a scatter plot for two variables.

The Plot statement first specifies the Y-axis.

A Gplot step may contain any number of Plot statements.

A Quit statement is required at the end of the program code because the procedure resides in memory.

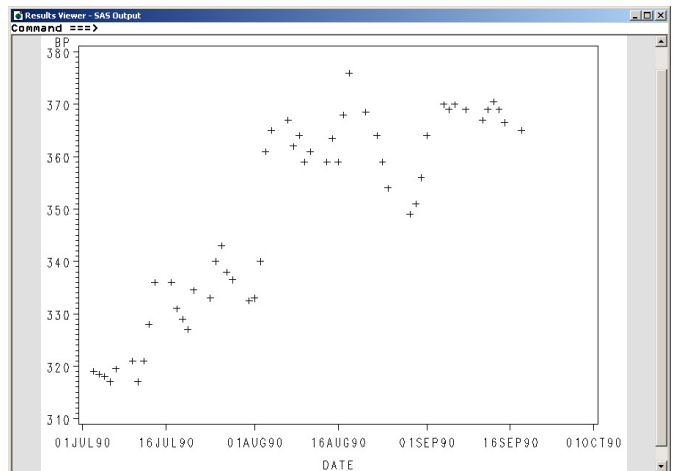
```
Program Editor - gr013
Command ==>
00001 *** Simple Plots Example;
00002
00003 proc print data=saved.shares;
00004   var date bp;
00005   format date date7.;
00006 run;
00007
00008 options reset=all reset=global;
00009
00010 proc gplot data=saved.shares;
00011   plot bp*date;
00012 run;
00013 quit;
```

Note: Confirm that the Create Listing option in the Results tab in the Preferences window is checked. This is critical for SAS to be able to store the output created (in list format) and create a graph from this output.

SAS creates the following list output and graph output:

The image shows a screenshot of the SAS Output window titled "Output - (Untitled)". It displays a list of data points with columns for "Obs", "DATE", and "BP". The data points range from observation 1 to 41, with dates from 03JUL90 to 28AUG90 and BP values ranging from 318.0 to 368.5.

Obs	DATE	BP
1	03JUL90	319.0
2	04JUL90	318.5
3	05JUL90	318.0
4	06JUL90	317.0
5	07JUL90	319.5
6	10JUL90	321.0
7	11JUL90	317.0
8	12JUL90	321.0
9	13JUL90	328.0
10	14JUL90	336.0
11	17JUL90	336.0
12	18JUL90	331.0
13	19JUL90	329.0
14	20JUL90	327.0
15	21JUL90	334.5
16	24JUL90	333.0
17	25JUL90	340.0
18	26JUL90	343.0
19	27JUL90	338.0
20	28JUL90	336.5
21	31JUL90	332.5
22	01AUG90	333.0
23	02AUG90	340.0
24	03AUG90	361.0
25	04AUG90	365.0
26	07AUG90	367.0
27	08AUG90	362.0
28	09AUG90	364.0
29	10AUG90	359.0
30	11AUG90	361.0
31	14AUG90	359.0
32	15AUG90	363.5
33	16AUG90	359.0
34	17AUG90	368.0
35	18AUG90	376.0
36	21AUG90	368.5
37	22AUG90	359.0
38	23AUG90	364.0
39	24AUG90	359.0
40	25AUG90	354.0
41	28AUG90	.



Connecting the Dots

By default, the values on the graph are left unconnected. This is probably the most accurate depiction of the data, because once lines are drawn between data points, we begin making assumptions about the data.

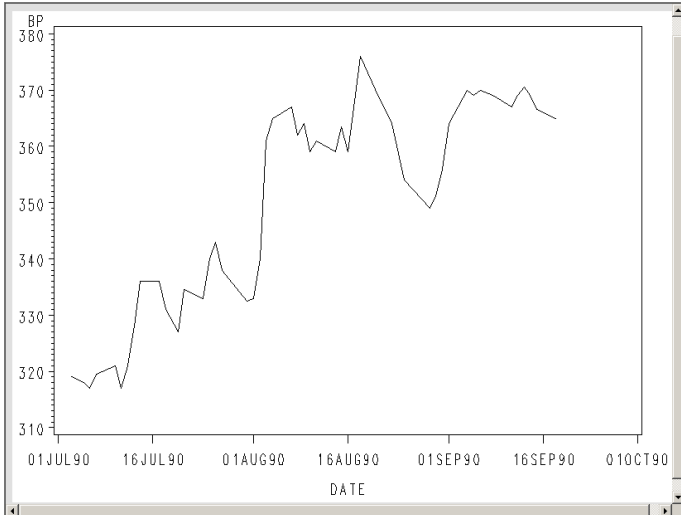
The Symbol statement is used to specify the line drawn between the data points. Join and Spline are two methods used by the symbol statement to connect the dots.

An example of the Symbol statement with the Join option is displayed below:

```

Program Editor - gr014
Command ==>
00001 *** Joining The Dots Example;
00002
00003 goptions reset=all reset=global;
00004
00005 symbol1 i=join;
00006 proc gplot data=saved.shares;
00007   plot bp*date;
00008 run;
00009 quit;

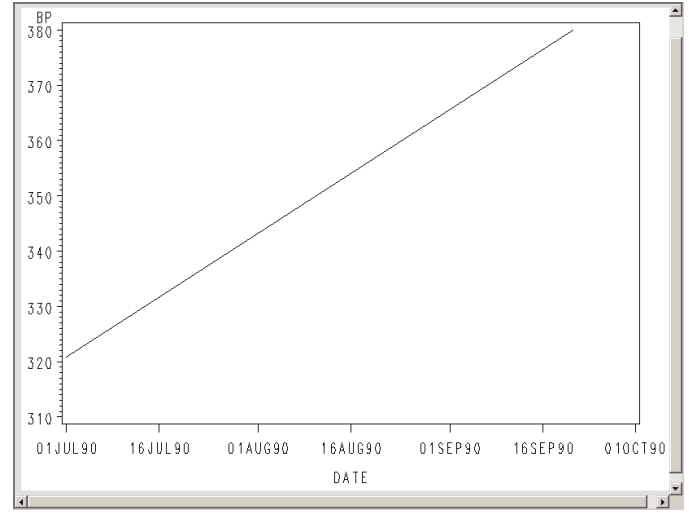
```



```

Program Editor - gr014b
Command ==>
00001 *** Joining The Dots Example;
00002
00003 goptions reset=all reset=global;
00004
00005 symbol1 i=rcli95;
00006 proc gplot data=saved.shares;
00007   plot bp*date;
00008 run;
00009 quit;

```



Once specified, the symbol statement remains in effect until it is canceled by another specification or by specifying the following:

Goptions Reset=Global;

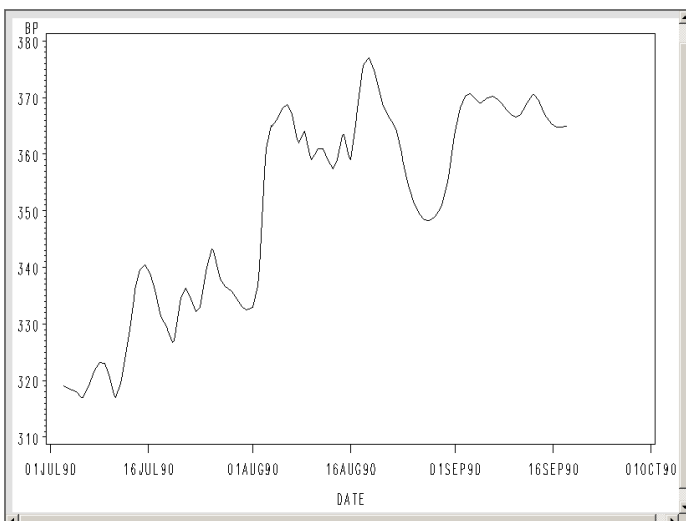
This is only one way of joining the points on a graph.

The symbol statement controls the type of line drawn, its color, width, line style and the symbol used to mark the data points. Now let's resubmit the code using the Spline option, which applies a smoothing effect to the points being connected.

```

Program Editor - gr014a
Command ==>
00001 *** Joining The Dots Example;
00002
00003 goptions reset=all reset=global;
00004
00005 symbol1 i=spline;
00006 proc gplot data=saved.shares;
00007   plot bp*date;
00008 run;
00009 quit;

```



Incorporating the rcli95 option in the Symbol Statement:

Labeling the Axes

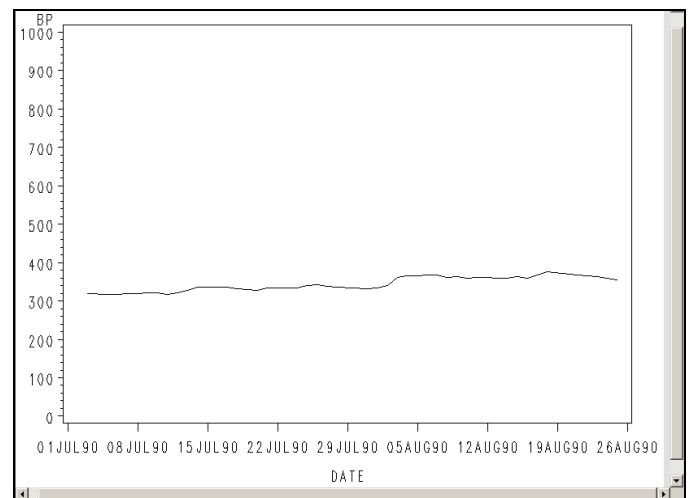
In the previous example, the range of vertical axis of the graph (representing bp) is mapped by default. It ranges from 310 to 380 in scale.

We can change the default settings for both axes by specifying the ranges for the axes, as displayed in the code below:

```

Program Editor - gr015
Command ==>
00001 *** Labeling The Axes Example;
00002
00003 goptions reset=all reset=global;
00004
00005 symbol1 i=join;
00006 proc gplot data=saved.shares;
00007   plot bp*date /
00008     vaxis=0 to 1000 by 100
00009     haxis='01jul90'd to '01sep90'd by 7;
00010 run;
00011 quit;

```



We can change the presentation of the graph, in this manner.

The symbol statement has been repeated here for clarity. It is not necessary, since it was submitted previously.

Adding Titles, Footnotes and Legends

```

Program Editor - gr016
Command ==>
00001 *** Adding Titles, Footnotes Example;
00002
00003 options reset=all reset=global;
00004
00005 symbol1 i=join;
00006 title1 h=2 c=blue f=zapfi
00007 'BP Share Price July to September 1990';
00008 footnote1 h=1 c=green f=zapfi 'Source: Financial Times';
00009
00010 proc gplot data=saved.shares;
00011 plot bp*date /
00012 vaxis=300 to 400 by 10
00013 haxis='01jul90'd to '01sep90'd by 7;
00014 run;
00015 quit;

```



There is no legend on the graph even though the legend was requested.

We will see how to display legends using a different form of the plot statement, in a subsequent section.

Legends can be constructed with footnote statements on simple plots.

Once specified, the Title, Footnote and Symbol statements remain in effect until canceled.

Overlaying

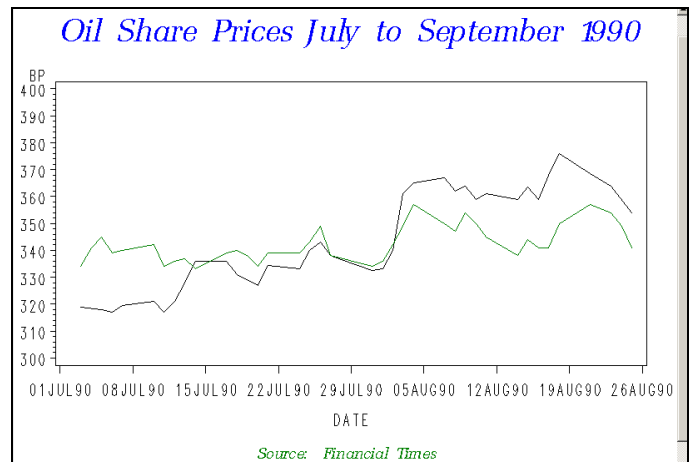
The overlay option is used to place multiple graphs on a page.

This is illustrated in the code below:

```

Program Editor - gr017
Command ==>
00001 *** Overlaying Example;
00002
00003 options reset=all reset=global;
00004
00005 symbol1 i=join;
00006 title1 h=2 c=blue f=zapfi
00007 'Oil Share Prices July to September 1990';
00008 footnote1 h=1 c=green f=zapfi 'Source: Financial Times';
00009 footnote3 h=1 c=blue f=zapfi 'NYSE';
00010 proc gplot data=saved.shares;
00011 plot bp*date burmah*date ultramar*date /
00012 overlay
00013 vaxis=300 to 400 by 10
00014 haxis='01jul90'd to '01sep90'd by 7;
00015 run;
00016 quit;

```



Notice the colors of the plot lines.

One symbol statement has been used in this graph, all with the option `i=join`.

The second plot requested was not produced. The SAS log informs us that the values to be plotted lie outside the axis range specified: either the vaxis or the haxis specification is not sufficient to cover all the data.

Since the values of date are the same for all share prices, we can assume that the vaxis needs to be extended.

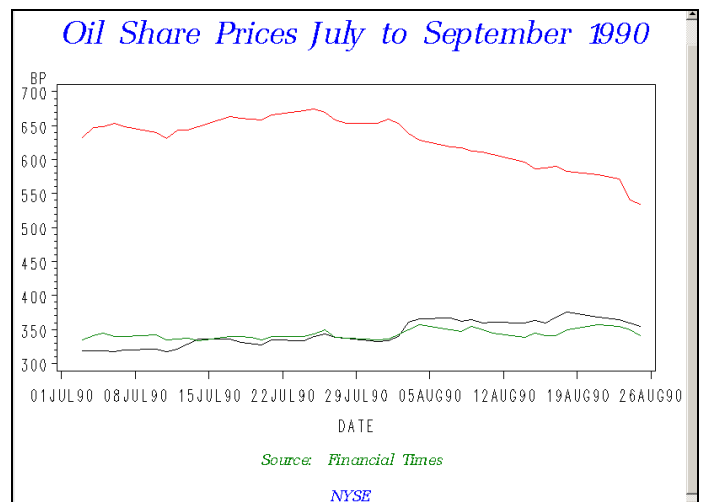
Let's re-submit the code after making modifications to the vaxis, as displayed:

```

Program Editor - gr017a
Command ==>
00001 *** Overlaying Example;
00002
00003 options reset=all reset=global;
00004
00005 symbol1 i=join;
00006 title1 h=2 c=blue f=zapfi
00007 'Oil Share Prices July to September 1990';
00008 footnote1 h=1 c=green f=zapfi 'Source: Financial Times';
00009 footnote3 h=1 c=blue f=zapfi 'NYSE';
00010 proc gplot data=saved.shares;
00011 plot bp*date burmah*date ultramar*date /
00012 overlay
00013 vaxis=300 to 700 by 50
00014 haxis='01jul90'd to '01sep90'd by 7;
00015 run;
00016 quit;

```

The graph now displays all the plots as requested in the code.



Global Statements

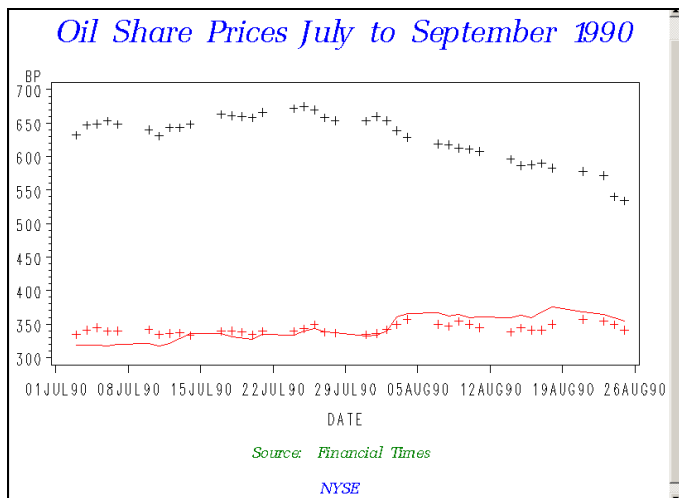
Symbol statements cycle round the list of colors if not color is specified

Symbol statements are Global.

The program below first resets the global statements to the default and then sets the Symbol, Title and Footnotes again.

This is clearly inefficient, and has been shown solely for illustration purposes.

```
Program Editor - gr018
Command ==>
00001 *** Global Statements Example;
00002
00003 goptions reset=all reset=global;
00004
00005 symbol1 i=join c=red;
00006 title1 h=2 c=blue f=zapfi
00007 'Oil Share Prices July to September 1990';
00008 footnote1 h=1 c=green f=zapfi 'Source: Financial Times';
00009 footnote3 h=1 c=blue f=zapfi 'NYSE';
00010 proc gplot data=saved.shares;
00011 plot bp*date burmah*date ultramar*date /
00012 overlay
00013 vaxis=300 to 700 by 50
00014 haxis='01jul90'd to '01sep90'd by 7;
00015 run;
00016 quit;
```



Symbol statements are additive, in addition to being global in nature.

The result of Symbol1 displayed in the previous code is:

```
Symbol1 i=join;
+      Symbol1 c=red;
-----
=      Symbol1 i=join c=red;
```

The graph shows one symbol statement being used, a joined red line.

The other plots do not have symbol statements and a join is not displayed. They have begun to cycle round the list of colors for the device.

Cycling Colors

If the Symbol statement is not given a color, then it cycles round the list of colors for the device, generating a symbol statement each time.

Given a device with the following colors:

```
goptions colors=(black,red,green,blue,orange,brown);
```

Consider the effect of the following Symbol statements:

```
Symbol1 c=green i=join;
```

```
Symbol2 i=join;
Symbol3 c=blue i=join;
```

Results in the following symbol statements generated:

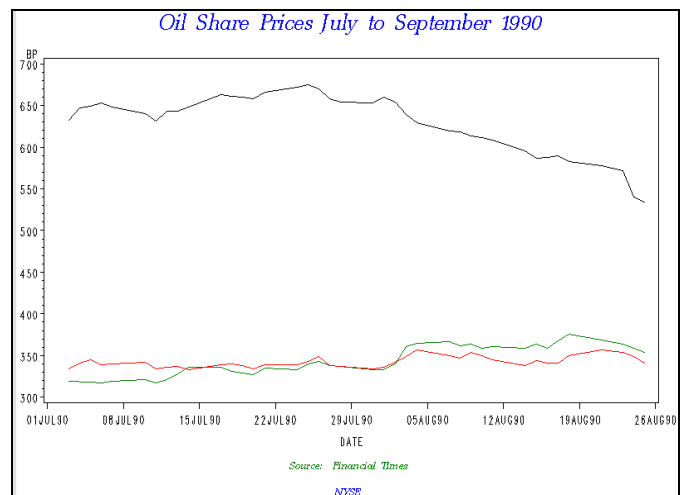
```
symbol c=green i=join;* 1st statement generated;
symbol c=black i=join;* 2nd generated;
symbol c=red i=join;* 3rd generated, etc...;
symbol c=green i=join;
symbol c=blue i=join;
symbol c=orange i=join;
symbol c=brown i=join;
symbol c=blue i=join;
```

Pointing to a Symbol Statement

The Plot statement can point to a Symbol Statement. It points to the nth generated symbol statement, as displayed below:

```
Program Editor - gr019
Command ==>
00001 *** Pointing To A Symbol Statement Example;
00002 *** The symbol2 statement without a color causes a number of symbol
00003 statements to be generated until all the default colors are
00004 exhausted. Then, the symbol3 color of blue kicks in.
00005 NOTE: ultramar*date=3 is really the second symbol statement with
00006 the second default color in the COLORS=(,...) goption.;
00007
00008 goptions reset=all reset=global;
00009
00010 symbol1 c=green i=join;
00011 symbol2 i=join;
00012 symbol3 c=blue i=none;
00013
00014 title1 h=2 c=blue f=zapfi
00015 'Oil Share Prices July to September 1990';
00016 footnote1 h=1 c=green f=zapfi 'Source: Financial Times';
00017 footnote3 h=1 c=blue f=zapfi 'NYSE';
00018 proc gplot data=saved.shares;
00019 plot bp*date=1 burmah*date=2 ultramar*date=3 /
00020 overlay
00021 vaxis=300 to 700 by 50
00022 haxis='01jul90'd to '01sep90'd by 7;
00023 run;
00024 quit;
```

The values inform SAS how to assign the Symbol Statements.



Line Control

Proc Gplot controls lines using the Symbol Statement. The symbol statement can accept many options Symbols.

SYMBOL STATEMENT LINE = OPTIONS

SYMBOL STATEMENT VALUE= OPTIONS

SYMBOLS CODE(SYMBOLS)

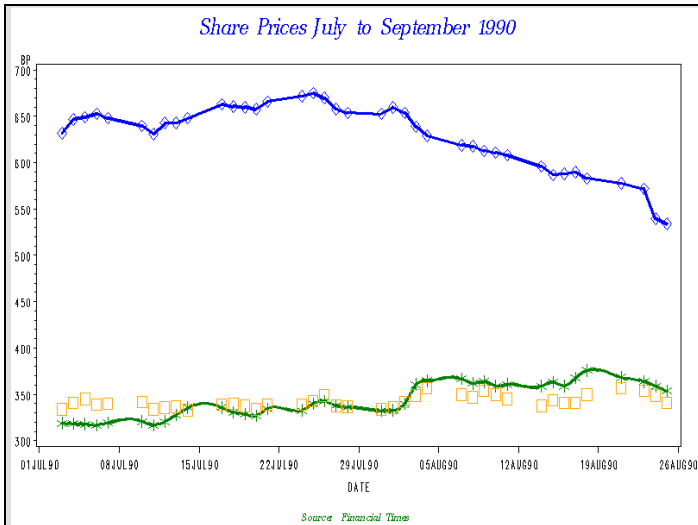
Variations

Examples of Symbol Statements:

```

Program Editor - gr020
Command ==>
00001 *** Variations of Symbol Statements Example;
00002
00003 goptions reset=all reset=global;
00004
00005 symbol1 i=spline c=green v=star h=2 w=3;
00006 symbol2 i=join c=blue v=diamond h=2 w=3;
00007 symbol3 i=none c=orange v=square h=2 w=2;
00008
00009 title1 h=2 c=blue f=zapfi 'Share Prices July to September 1990';
00010 footnote1 h=1 c=green f=zapfi 'Source: Financial Times';
00011 proc gplot data=saved.shares;
00012 plot bpxdate burmah*date ultramar*date /
00013 overlay
00014 vaxis=300 to 700 by 50
00015 haxis='01jul90'd to '01sep90'd by 7;
00016 run;
00017 quit;

```



Now let's modify the code by changing the interpolation method for symbol1 to needle.

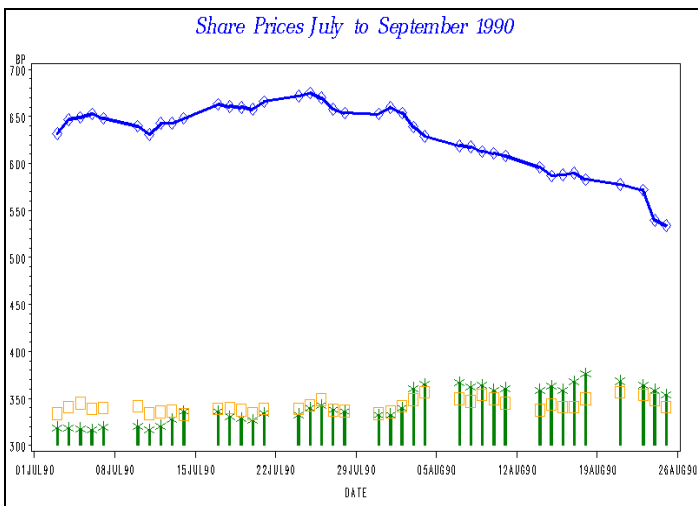
Resubmit the code:

```

Program Editor - gr020a
Command ==>
00001 *** Variations of Symbol Statements Example;
00002
00003 goptions reset=all reset=global;
00004
00005 symbol1 i=needle c=green v=star h=2 w=3;
00006 symbol2 i=join c=blue v=diamond h=2 w=3;
00007 symbol3 i=none c=orange v=square h=2 w=2;
00008
00009 title1 h=2 c=blue f=zapfi 'Share Prices July to September 1990';
00010 footnote1 h=1 c=green f=zapfi 'Source: Financial Times';
00011 proc gplot data=saved.shares;
00012 plot bpxdate burmah*date ultramar*date /
00013 overlay
00014 vaxis=280 to 400 by 20
00015 haxis='01jul90'd to '01sep90'd by 7;
00016 run;
00017 quit;

```

The output is displayed below.

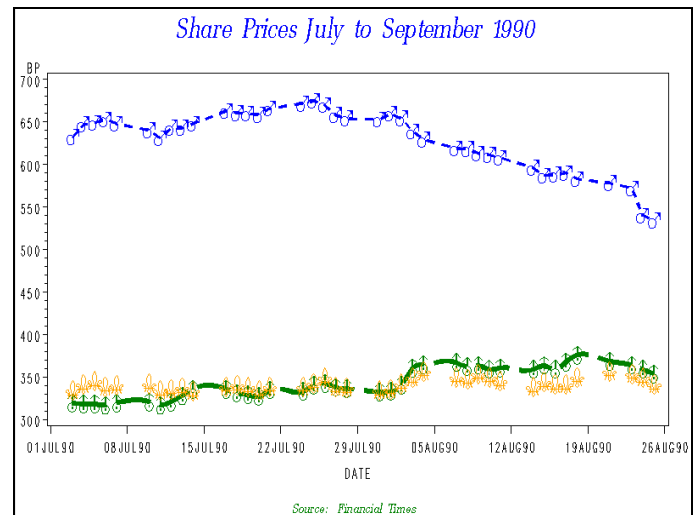


Symbol Statement Examples

```

Program Editor - gr023
Command ==>
00001 *** Variations of Symbol Statements Example 2;
00002
00003 goptions reset=all reset=global;
00004
00005 symbol1 i=spline c=green v=',' h=2 w=6 l=5;
00006 symbol2 i=join c=blue v='>' h=2 w=3 l=2;
00007 symbol3 i=none c=orange v='"' h=2 w=2 l=10;
00008
00009 title1 h=2 c=blue f=zapfi 'Share Prices July to September 1990';
00010 footnote1 h=1 c=green f=zapfi 'Source: Financial Times';
00011 proc gplot data=saved.shares;
00012 plot bpxdate burmah*date ultramar*date /
00013 overlay
00014 vaxis=300 to 700 by 50
00015 haxis='01jul90'd to '01sep90'd by 7;
00016 run;
00017 quit;

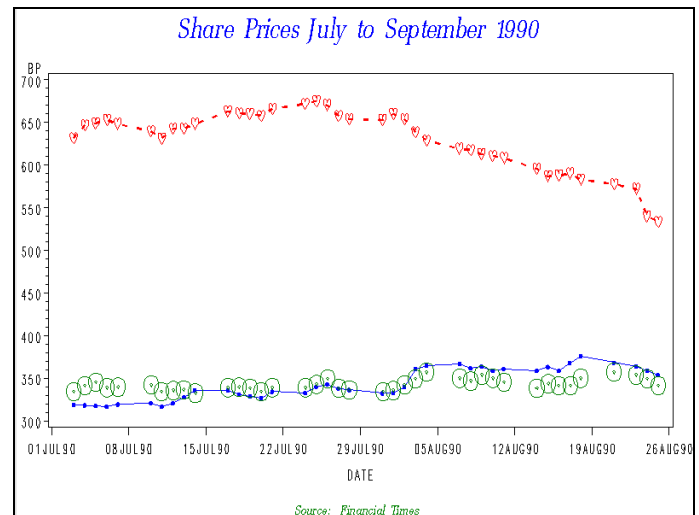
```



```

Program Editor - gr024
Command ==>
00001 *** Variations of Symbol Statements Example 3;
00002
00003 goptions reset=all reset=global;
00004
00005 symbol1 i=join l=1 c=blue v=dot h=0.5 w=1;
00006 symbol2 i=L l=34 c=red v='*' h=1.2 w=2;
00007 symbol3 i=none l=45 c=green v='-' h=2.3 w=3;
00008
00009 title1 h=2 c=blue f=zapfi 'Share Prices July to September 1990';
00010 footnote1 h=1 c=green f=zapfi 'Source: Financial Times';
00011 proc gplot data=saved.shares;
00012 plot bpxdate burmah*date ultramar*date /
00013 overlay
00014 vaxis=300 to 700 by 50
00015 haxis='01jul90'd to '01sep90'd by 7;
00016 run;
00017 quit;

```

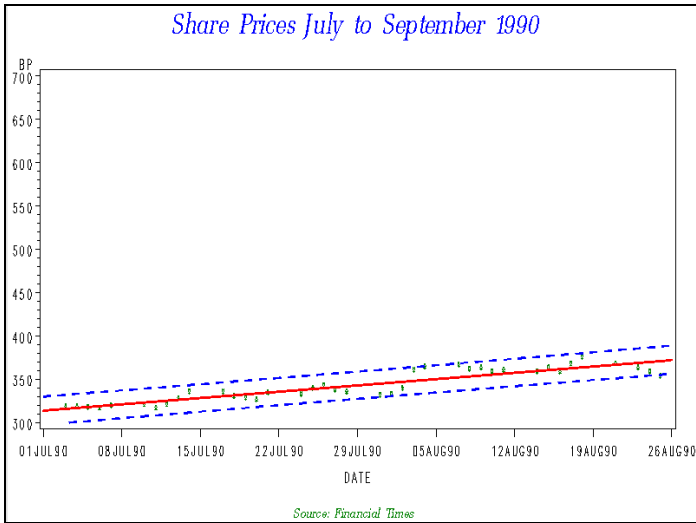


Next, we use the option designed to present information graphing confidence limits.

```

Program Editor - gr026
Command ==>
00001 *** Symbol Statement Examples 5;
00002
00003 goptions reset=global reset=all;
00004 symbol i=rqli195 ci=red co=blue cv=green v='e' w=2 h=0.5;
00005
00006 title1 h=2 c=blue f=zapfl 'Share Prices July to September 1990';
00007 footnote1 h=1 c=green f=zapfl 'Source: Financial Times';
00008 proc gplot data=saved.shares;
00009 plot bp*date /
00010 vaxis=300 to 700 by 50
00011 haxis='01jul90'd to '01sep90'd by 7;
00012 run;
00013 quit;

```



X*Y=Z and Axis Control

A normal plot statement of the form:

Plot A*C B*C / Overlay;

The above code displays two plots on one graph, but no legend showing which line corresponded to A, and which to B.

X*Y=Z

Using the X*Y=Z form of the Plot statement generates automatic Legends which display what each line represents.

Plot X*Y=Z;

To do this, the data must be in a certain structure.

Review the Flu data set:

```

Program Editor - gr027
Command ==>
00001 *** Axis Control FLU Data Set Printout;
00002
00003 title;
00004 footnote;
00005 proc print data=saved.flu;
00006 run;

```

Output in list form:

Output - (Untitled)

Command ==>

14:57 Wednesday, April 4, 2001

Obs	A	B	C	D	E	WEEK_NO	DATE
1	10	8	12	15	17	1	10838
2	21	15	12	11	26	2	10845
3	25	16	13	18	22	3	10852
4	32	16	13	11	41	4	10859
5	36	24	23	45	56	5	10866
6	45	32	35	43	65	6	10873
7	56	34	38	56	75	7	10880
8	86	45	48	68	97	8	10887
9	120	75	88	94	110	9	10894
10	140	89	97	106	120	10	10901
11	168	102	110	132	154	11	10908
12	190	110	128	154	178	12	10915
13	224	100	121	131	191	13	10922
14	180	94	112	125	160	14	10929
15	128	82	96	102	112	15	10936
16	114	67	78	88	95	16	10943
17	85	45	56	59	76	17	10957
18	63	34	44	49	54	18	10964
19	35	23	36	38	17	19	10971
20	23	11	18	19	19	20	10978
21	12	4	8	9	10	21	10985

Output in HTML form:

Obs	A	B	C	D	E	WEEK_NO	DATE
1	10	8	12	15	17	1	10838
2	21	15	12	11	26	2	10845
3	25	16	13	18	22	3	10852
4	32	16	13	11	41	4	10859
5	36	24	23	45	56	5	10866
6	45	32	35	43	65	6	10873
7	56	34	38	56	75	7	10880
8	86	45	48	68	97	8	10887
9	120	75	88	94	110	9	10894
10	140	89	97	106	120	10	10901
11	168	102	110	132	154	11	10908
12	190	110	128	154	178	12	10915
13	224	100	121	131	191	13	10922
14	180	94	112	125	160	14	10929
15	128	82	96	102	112	15	10936
16	114	67	78	88	95	16	10943
17	85	45	56	59	76	17	10957
18	63	34	44	49	54	18	10964
19	35	23	36	38	17	19	10971
20	23	11	18	19	19	20	10978
21	12	4	8	9	10	21	10985

We could Plot A*Date B*Date C*Date D*Date E*Date /Overlay; but this would not generate a legend.

We need to rearrange the data for the Y*X=Z form.

Rearranging Data

```

Program Editor - gr028
Command ==>
00001 *** Rearranging The Data;
00002
00003 data epidemic;
00004 set saved.flu;
00005 keep cases category week_no date;
00006 category = 'A'; cases=a; output;
00007 category = 'B'; cases=b; output;
00008 category = 'C'; cases=c; output;
00009 category = 'D'; cases=d; output;
00010 category = 'E'; cases=e; output;
00011 run;
00012 title;
00013 footnote;
00014 proc print data=epidemic;
00015 format date date7.;
00016 run;

```

Obs	WEEK_NO	DATE	category	cases
1	1	03SEP89	A	10
2	1	03SEP89	B	8
3	1	03SEP89	C	12
4	1	03SEP89	D	15
5	1	03SEP89	E	17
6	2	10SEP89	A	21
7	2	10SEP89	B	15
8	2	10SEP89	C	12
9	2	10SEP89	D	11
10	2	10SEP89	E	26
11	3	17SEP89	A	25
12	3	17SEP89	B	16
13	3	17SEP89	C	13
14	3	17SEP89	D	18
15	3	17SEP89	E	22

We can now Plot Cases*Date=Category;

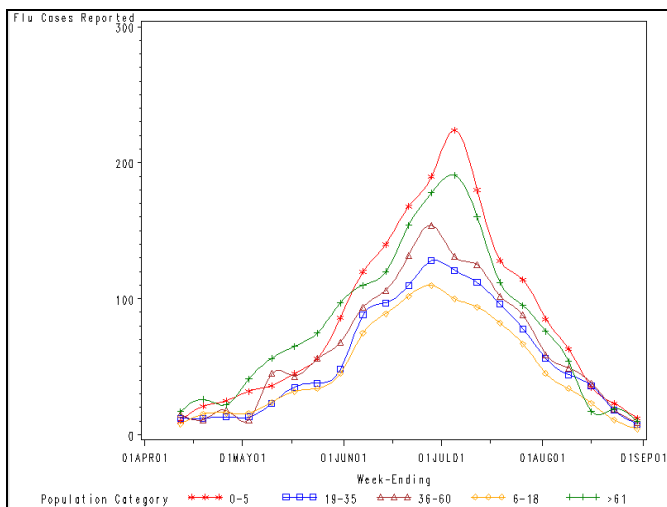
For each Category, A, B, C, D, E a separate line will be drawn on the graph.

```

Program Editor - gr029
Command ==>
00001 *** Automated Legend Example;
00002
00003 options reset=global reset=all;
00004 proc format;
00005     value $age 'A'='0-5'
00006               'B'='6-18'
00007               'C'='19-35'
00008               'D'='36-60'
00009               'E'='>61';
00010 run;
00011 symbol1 v=star c=red i=spline;
00012 symbol2 v=square c=blue i=spline;
00013 symbol3 v=triangle c=brown i=spline;
00014 symbol4 v=diamond c=orange i=spline;
00015 symbol5 v=plus c=green i=spline;
00016 proc plot data=saved.epidemic;
00017     plot cases*date=category;
00018     label date='Week-Ending'
00019           category='Population Category';
00020     cases='Flu Cases Reported';
00021     format date date7. category $age.;
00022 run;
00023 quit;

```

The output is displayed below.



The five Symbols statements generated will match with the five lines drawn, one for each value of category.

Always review the SAS log for notes and messages. They inform you about plots that lie outside the available range.

Introduction to Charts

SAS/GRAPH can create different chart styles.

Let's illustrate with a few examples.

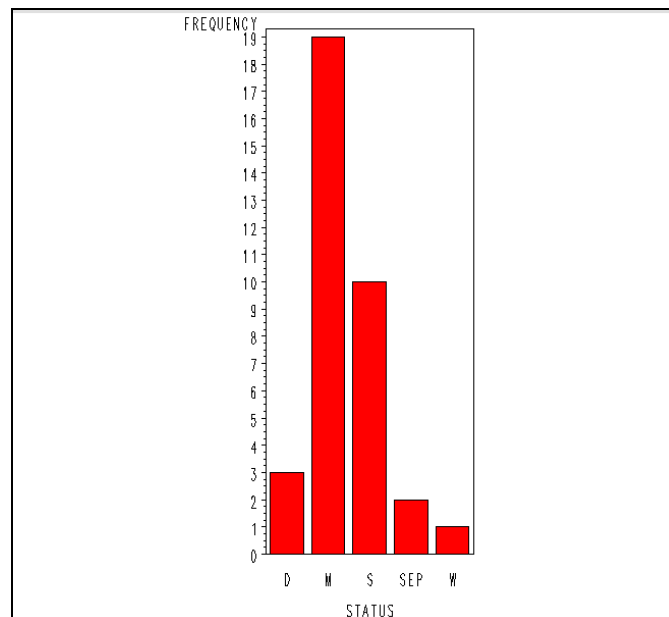
Vertical Bar Chart Example

Submit the following code:

```

Program Editor - ch4_1
Command ==>
00001 options reset=global reset=all;
00002
00003 proc gchart data=saved.demograf;
00004     vbar status;
00005 run;
00006 quit;

```

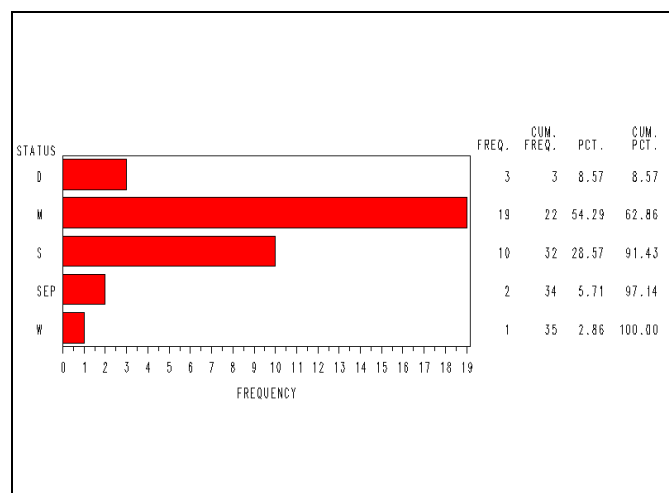


Horizontal Bar Chart Example

```

Program Editor - ch4_2
Command ==>
00001 options reset=global reset=all;
00002
00003 proc gchart data=saved.demograf;
00004     hbar status;
00005 run;
00006 quit;

```



Notes:

Unless otherwise specified, SAS displays the frequency of the selected variable for display, by default.

Each bar or block represents a value of a variable, either character or numeric.

Bars can be grouped, sub-grouped and various patterns and colors can be used to enhance the presentation.

By default, Horizontal Bar Charts display graph statistics on the graph area next to the graph.

Pie charts can be of two types, Pie Charts and Star Charts.

Terminology

Physical Forms

The physical form of a chart is determined by the type specified in SAS code:

```
Proc Gchart Data=SAS_data_set;
  Vbar Variable / Options ;
```

The Vbar statement requests a vertical bar chart.

The variable after Vbar determines the number of bars the graph will contain.

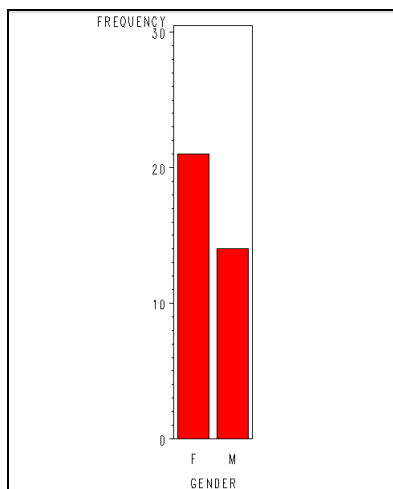
Options on the statement control other aspects of the graph.

For example, the Demograf data set has a variable named gender, which has two values: F and M.

If we specify the following:

```
Program Editor - gr038
Command ==>
00001 *** Ubar Example;
00002
00003 options reset=global reset=all;
00004
00005 proc gchart data=saved.demograf;
00006   vbar gender;
00007 run;
00008 quit;
```

The following chart is displayed.



By default, the vertical axis displays the frequency or the number of observations in the data set.

In this case, there are 21 females and 14 males.

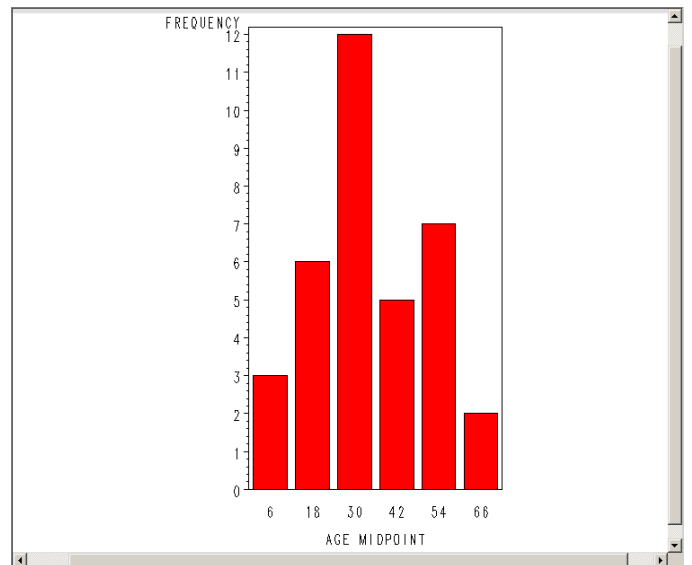
Midpoints

For numeric variables, SAS/GRAPH will chart the midpoints of a data range.

Sometimes this produces unexpected results.

Let's consider the same vertical bar chart as before, but use a numeric variable such as Age that takes on values from 15 to 65.

```
Program Editor - gr039
Command ==>
00001 *** Midpoints Example;
00002
00003 options reset=global reset=all;
00004
00005 proc gchart data=saved.demograf;
00006   vbar age;
00007 run;
00008 quit;
```



As seen from the graph, it does not produce one bar for each value in the data.

The data range (2 to 65) is divided into ranges, and the midpoint of each range is charted:

Range	Midpoint
2-12	6
12-23	18
24-35	30
36-47	42
48-59	54
60-71	66

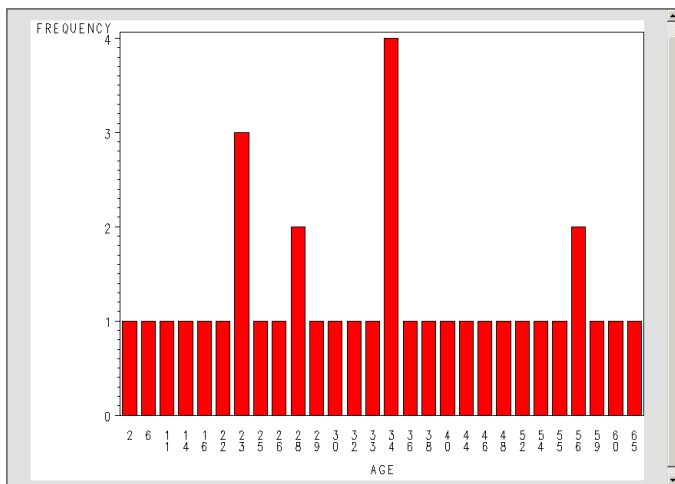
DISCRETE

The discrete option suppresses the calculation of these ranges and forces the procedure to produce one bar for each value in the data.

Submit the code below for illustration purposes.

```
Program Editor - gr040
Command ==>
00001 *** Discrete Example;
00002
00003 options reset=global reset=all;
00004
00005 proc gchart data=saved.demograf;
00006   vbar age / discrete;
00007 run;
00008 quit;
```

The graph will be displayed, as follows:



The Discrete option is valid for all types of charts.

Valid Options for GCHART

Some options are common to the Vbar, Hbar, Block, Pie and Star Statements, while others are specific to the type of chart being drawn.

The table below lists options the types of chart(s) to which they can be applied.

Option	Vbar	Hbar	Block	Pie	Star
ANNOTATE	*	*	*	*	*
CAXIS	*	*	*	*	*
COUTLINE	*	*	*	*	*
CTEXT	*	*	*	*	*
BLOCKMAX	*	*	*	*	*
PATTERNID=BY	*	*	*	*	*
GROUP	*	*	*	*	*
MIDPOINT	*	*	*	*	*
SUBGROUP	*	*	*	*	*
LEGEND	*	*	*	*	*
NOLEGEND	*	*	*	*	*
NOHEADING	*	*	*	*	*
FREQ	*	*	*	*	*
G100	*	*	*	*	*
SUMVAR	*	*	*	*	*
TYPE=CFREQ	*	*	*	*	*
CPERCENT	*	*	*	*	*
FREQ	*	*	*	*	*
MEAN	*	*	*	*	*
PERCENT	*	*	*	*	*
SUM	*	*	*	*	*
DISCRETE	*	*	*	*	*
LEVELS	*	*	*	*	*
MIDPOINTS	*	*	*	*	*
MISSING	*	*	*	*	*
GROUP	*	*	*	*	*
SUBGROUP	*	*	*	*	*
DESCRIPTION	*	*	*	*	*
NAME	*	*	*	*	*
GSPACE	*	*	*	*	*
SPACE	*	*	*	*	*
WIDTH	*	*	*	*	*
CFRAME	*	*	*	*	*

Option	Vbar	Hbar	Block	Pie	Star
FRAME	*	*			
CFREQ	*	*			
CPERCENT	*	*			
FREQ	*	*			
MEAN	*	*			
NOSTATS	*	*			
PERCENT	*	*			
SUM	*	*			
GAXIS	*	*			
MAXIS	*	*			
NOAXIS	*	*			
RAXIS	*	*			
MINOR	*	*			
ASCENDING	*	*			
DESCENDING	*	*			
NOZERO	*	*			
AUTOREF	*	*			
CLIPREF	*	*			
NOBASEREF	*	*			
REF	*	*			
CFILL				*	*
MATCHCOLOR				*	*
ANGLE				*	*
EXPLODE				*	*
FILL				*	*
INVISIBLE				*	*
PERCENT				*	*
SLICE				*	*
VALUE				*	*
NOGROUPHEADING				*	*
OTHER				*	*
ACROSS				*	*
DOWN				*	*
NOCONNECT				*	*

Bar Charts

Vertical Bar Charts

The Vbar Statement produces Vertical Bar Charts, also known as Histograms.

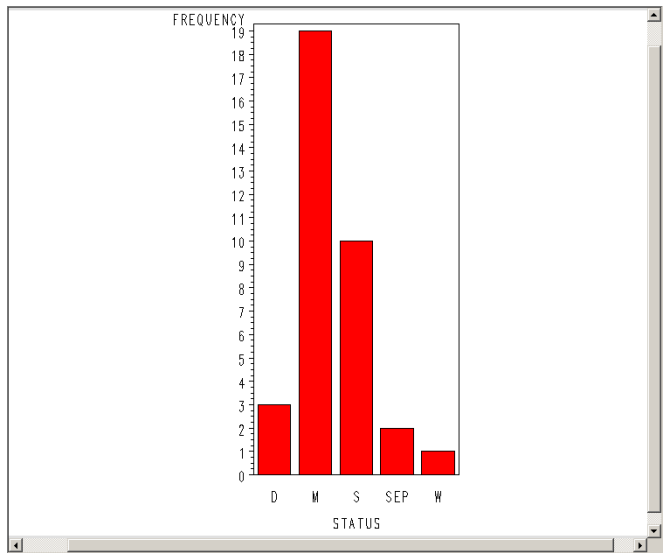
Character Variables

The discrete option does not add value when used with a character variable since each value in the data is given a bar.

```

Program Editor - gr041
Command ==> |
00001 *** Character Variables Example;
00002
00003 options reset=global reset=all;
00004
00005 proc gchart data=saved.demograp;
00006   vbar status;
00007 run;
00008 quit;

```

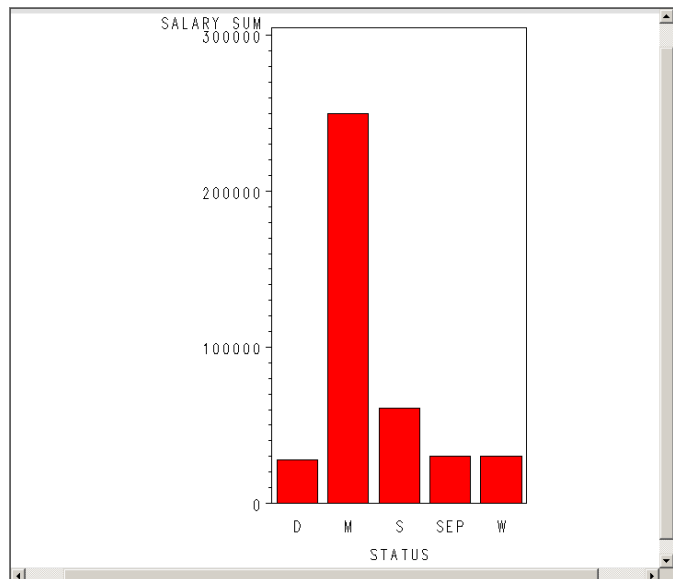


The Response Axis

The response axis is controlled with the SUMVAR (summing variable) option.

```
Program Editor - gr042
Command ==>
00001 *** Response Axis Example;
00002
00003 goptions reset=global reset=all;
00004
00005 proc gchart data=saved.demograp;
00006   vbar status / sumvar=salary;
00007 run;
00008 quit;
```

By default, the sumvar option displays the sum for the specified variable.



Statistics for SUMVAR

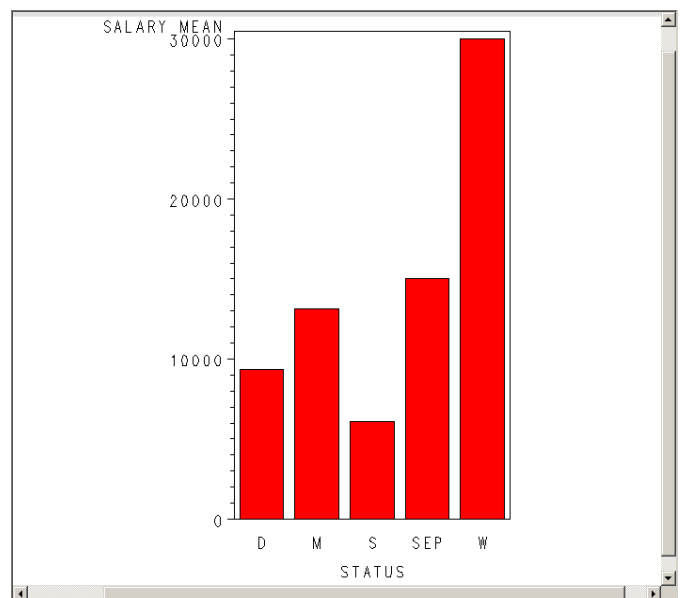
By default, the statistic displayed for Salary is SUM.

The response axis can display several different statistics, such as Mean, Sum (total), Freq, Cumulative Percent, Cumulative Frequency.

The option controlling statistics is TYPE.

Type and Sumvar work together to control which variable and statistic is displayed on the response axis.

```
Program Editor - gr043
Command ==>
00001 *** Statistics For Sumvar Example;
00002
00003 goptions reset=global reset=all;
00004
00005 proc gchart data=saved.demograp;
00006   vbar status / sumvar=salary type=mean;
00007 run;
00008 quit;
```

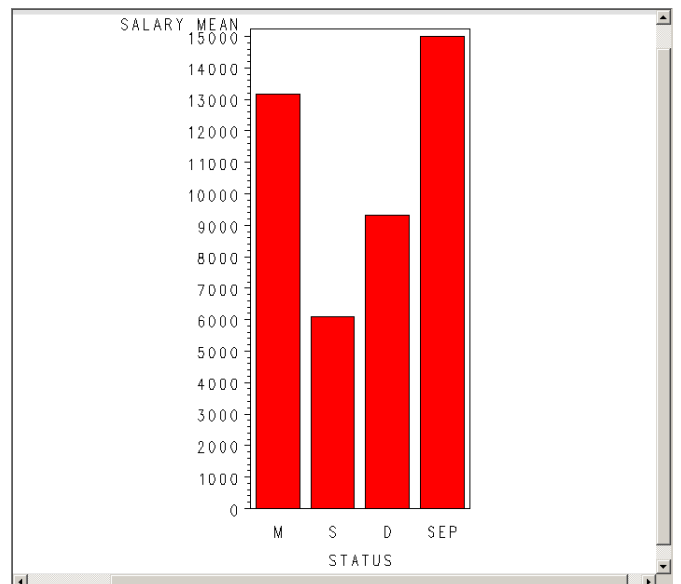


Controlling the Bars

The Midpoints option allows specification of which bars to chart. Use quotation marks around character values.

The usual SAS shortcuts can be coded when using Midpoints, e.g. 10 to 100 by 10.

```
Program Editor - gr044
Command ==>
00001 *** Controlling The Bars Example;
00002
00003 goptions reset=global reset=all;
00004
00005 proc gchart data=saved.demograp;
00006   vbar status / sumvar=salary type=mean
00007   midpoints='M' 'S' 'D' 'SEP';
00008 run;
00009 quit;
```

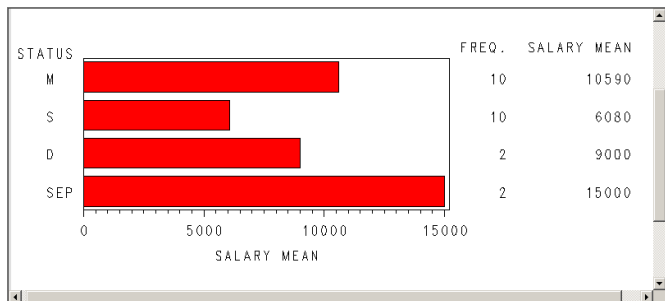


Selecting Observations

The Where clause can be used to subset certain observations you wish to chart.

```

Program Editor - gr045
Command ==>
00001 *** Selecting Observations Example;
00002
00003 options reset=global reset=all;
00004
00005 proc gchart data=saved.demograp;
00006   hbar status / sumvar=salary type=mean
00007   midpoints='M' 'S' 'D' 'SEP';
00008   where children le 2;
00009 run;
00010 quit;
  
```

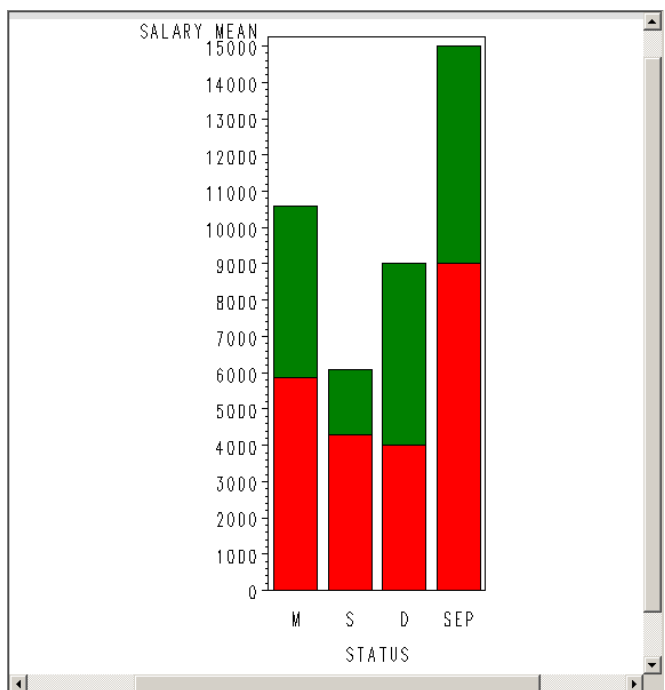


Sub-Dividing the Bars

The SUBGROUP option allows us to use another variable to divide the bars.

```

Program Editor - gr046
Command ==>
00001 *** Sub-Dividing The Bars Example;
00002
00003 options reset=global reset=all;
00004
00005 proc gchart data=saved.demograp;
00006   vbar status / sumvar=salary type=mean
00007   midpoints='M' 'S' 'D' 'SEP'
00008   subgroup=gender;
00009   where children le 2;
00010 run;
00011 quit;
  
```



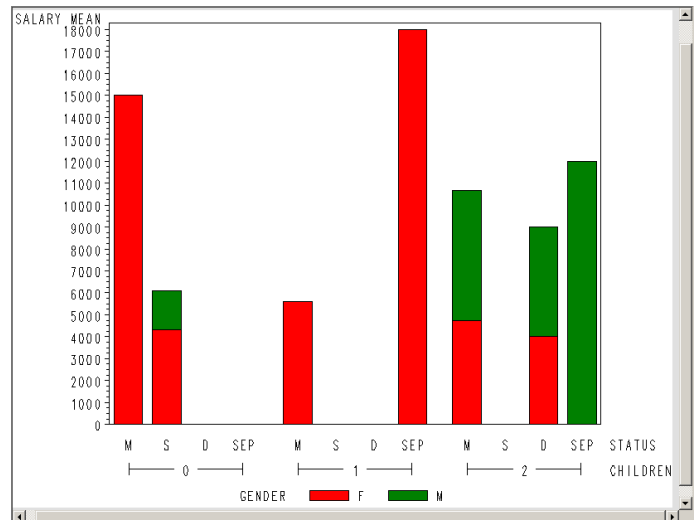
Note that an automatic legend is produced when subgroup is specified.

Grouping Bars

In addition to sub-dividing bars, they can also be grouped together using the GROUP option. This is displayed below.

```

Program Editor - gr047
Command ==>
00001 *** Grouping Bars Example;
00002
00003 options reset=global reset=all;
00004
00005 proc gchart data=saved.demograp;
00006   vbar status / sumvar=salary type=mean
00007   midpoints='M' 'S' 'D' 'SEP'
00008   subgroup=gender
00009   group=children;
00010   where children le 2;
00011 run;
00012 quit;
  
```



Additional Options

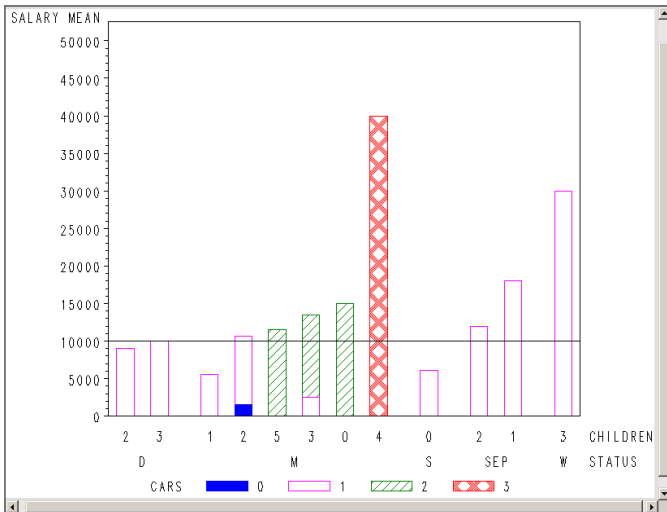
Additional options are available that can control the appearance of the report.

The following example illustrates Frame, Gspace, Space, Ref, Patternid, Nozeros, Ascending, Sum and Raxis.

```

Program Editor - gr049
Command ==>
00001 *** Other Options Example;
00002
00003 options reset=global reset=all;
00004
00005 pattern1 v=S c=b;
00006 pattern2 v=E c=m;
00007 pattern3 v=R1 c=g;
00008 pattern4 v=X5 c=r;
00009
00010 proc gchart data=saved.demograp;
00011   vbar children / frame gspace=2 space=2 ref=10000
00012   sumvar=salary discrete type=mean
00013   patternid=subgroup
00014   subgroup=cars group=status
00015   nozeros ascending sum
00016   raxis=0 to 50000 by 5000;
00017 run;
00018 quit;
  
```

The above code has four pattern statements (lines 5-8) and the pattern id (line 13) changes patterns by sub-group.



The graph used four different patterns.

The space between the bars and the space occupied by the bars is set to two.

A reference line is drawn at 10,000, and the bars are drawn in ascending order.

No space is left for non-existent bars, and patterns change across the sub-groups.

The response axis is ordered with the Raxis option.

Block Charts

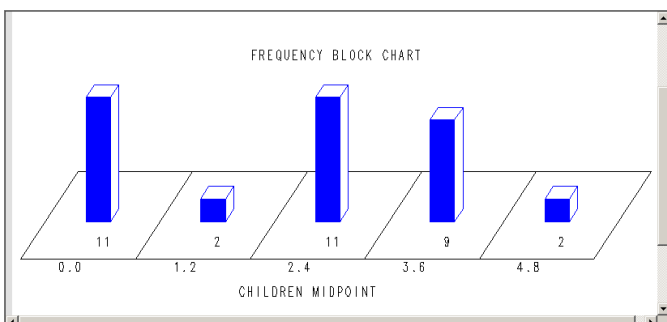
Block Charts are specified with the Block statement.

They use more space on the graphics area than a Vbar or Hbar and you may need to increase Hpos and Vpos. This is displayed on the next page.

```

Program Editor - gr050
Command ==>
00001 *** Block Charts Example;
00002
00003 goptions reset=global reset=all;
00004
00005 pattern1 v=s c=blue;
00006
00007 proc gchart data=saved.demograf;
00008   block children;
00009 run;
00010 quit;

```



Manhattan Charts

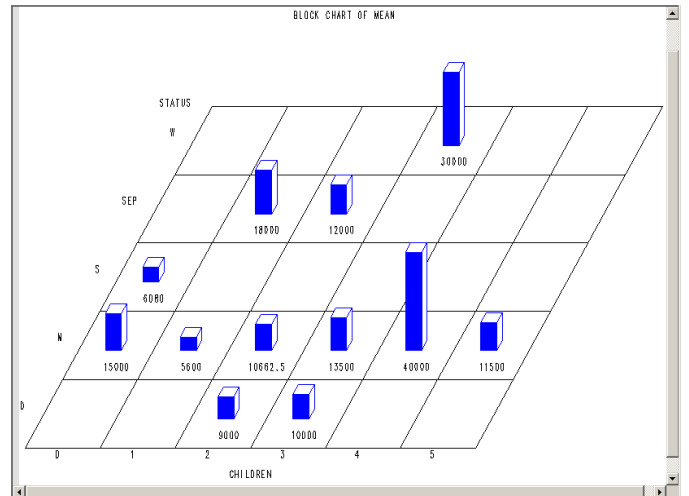
The GROUP option produces Manhattan Charts.

Note how the use of the discrete option changes the graph.

```

Program Editor - gr051
Command ==>
00001 *** Manhattan Charts Example;
00002
00003 goptions reset=global reset=all hpos=121 vpos=50;
00004
00005 pattern1 v=s c=blue;
00006
00007 proc gchart data=saved.demograf;
00008   block children/sumvar=salary discrete type=mean group=status;
00009 run;
00010 quit;

```



Also called Manhattan Charts, this Grouped Block Chart displays the data across the two axes.

Note the values of Hpos and Vpos.

The original values of Hpos=80, Vpos=62 have been increased by 50% each.

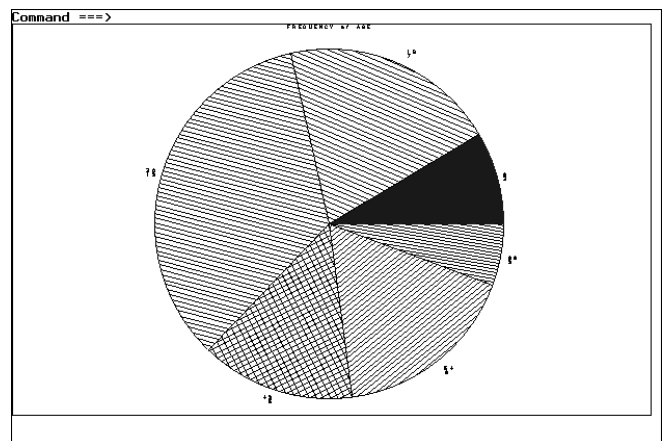
The aspect ratio then remains the same.

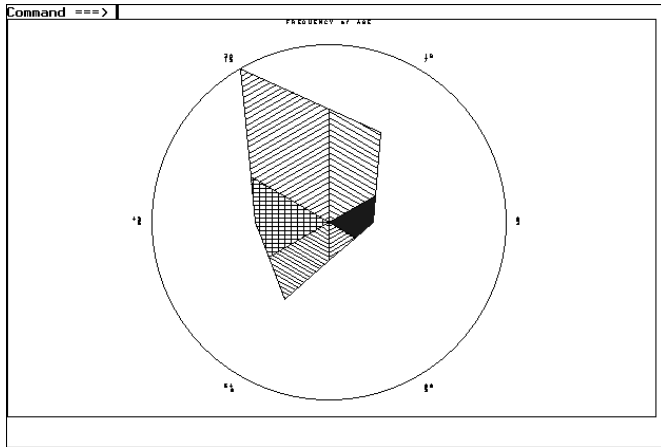
Calculations: $80+40 = 120$ $62+31=93$

Pie Charts

Pie Charts can be displayed using pies and stars.

Both types are capable of displaying data in a circular pie form.





Pie and Star charts are specified in a fashion similar to Vbar, Hbar and Block Charts:

Star Variable
Pie Variable

Default Option: OTHER

The option OTHER specifies a value of 4%, by default.

If you do not change this value, then any slice of the pie, which contributes less than 4%, is grouped together with all the OTHER small values.

This will be illustrated in the following examples.

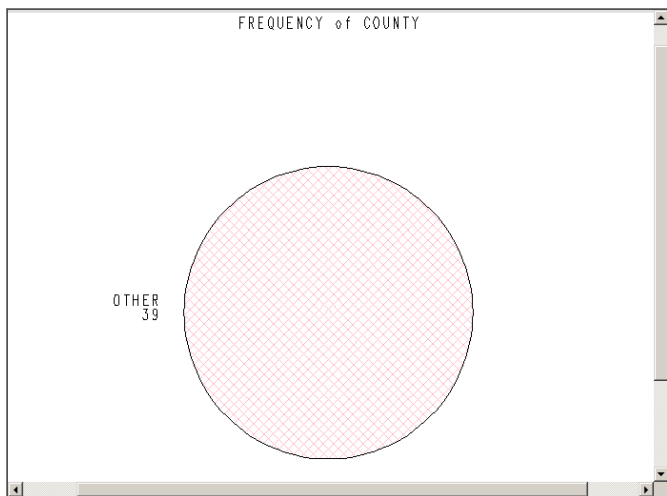
Pie Chart of County

Let's create a Pie Chart displaying the counties of the UK.

```

Program Editor - gr052
Command ==>
00001 *** Pie Chart of County Example;
00002
00003 goptions reset=all reset=global;
00004
00005 proc gchart data=saved.ukpop;
00006 pie county;
00007 run;
00008 quit;

```



When Pie charts are drawn for numeric variables, the Discrete option needs to be applied.

Since all the values in the data were under 4% of the total, they have been grouped together into the OTHER category.
Review the log.

```

Log - (Untitled)
Command ==>
percent of the pie/donut.
NOTE: The Bedfordshire midpoint was included into the OTHER slice with a value of 1 representing
2.6 percent of the pie/donut.
NOTE: The Berkshire midpoint was included into the OTHER slice with a value of 1 representing
2.6 percent of the pie/donut.
NOTE: The Buckinghamshire midpoint was included into the OTHER slice with a value of 1 representing
2.6 percent of the pie/donut.
NOTE: The Cambridgeshire midpoint was included into the OTHER slice with a value of 1 representing
2.6 percent of the pie/donut.
NOTE: The Cheshire midpoint was included into the OTHER slice with a value of 1 representing
2.6 percent of the pie/donut.
NOTE: The Cleveland midpoint was included into the OTHER slice with a value of 1 representing
2.6 percent of the pie/donut.
NOTE: The Cornwall midpoint was included into the OTHER slice with a value of 1 representing
2.6 percent of the pie/donut.
NOTE: The Cumbria midpoint was included into the OTHER slice with a value of 1 representing
2.6 percent of the pie/donut.

```

The notes state that all the counties were included in the OTHER slice because none of them represented more than 2.6% of the total pie.

In such a situation, we incorporate the other option where we specify the value of the other slice.

We need to adjust the other slice in relation to the values that we would like to display as slices of the pie.

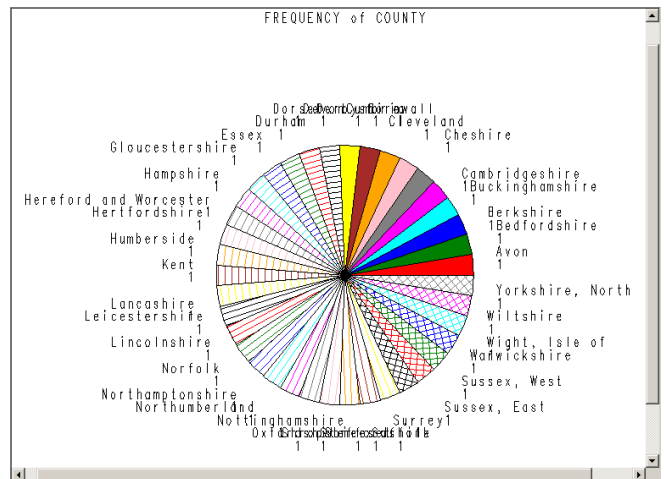
Let's set the value for Other at 0.1. This is demonstrated below.

```

Program Editor - gr053
Command ==>
00001 *** Pie Chart of County with Other Option Example;
00002
00003 goptions reset=all reset=global;
00004
00005 proc gchart data=saved.ukpop;
00006 pie county / other=0.1;
00007 run;
00008 quit;

```

The output is displayed below:



All the slices of the pie are displayed. The display is not clearly legible.

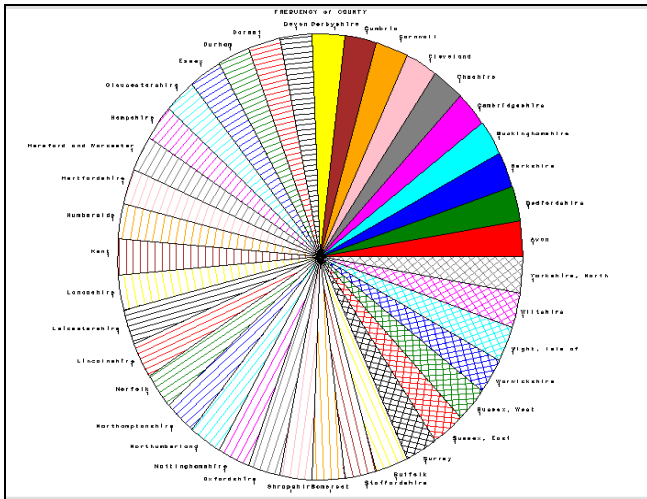
Let's increase the hpos to 150 and vpos to 120 and re-submit the code.

```

Program Editor - gr053_a
Command ==>
00001 *** Pie Chart of County with Other Option Example;
00002
00003 goptions reset=all reset=global
00004 hpos=150 vpos=120;
00005
00006 proc gchart data=saved.ukpop;
00007 pie county / other=0.1;
00008 run;
00009 quit;

```

This addresses the spacing issue and makes the graph legible.

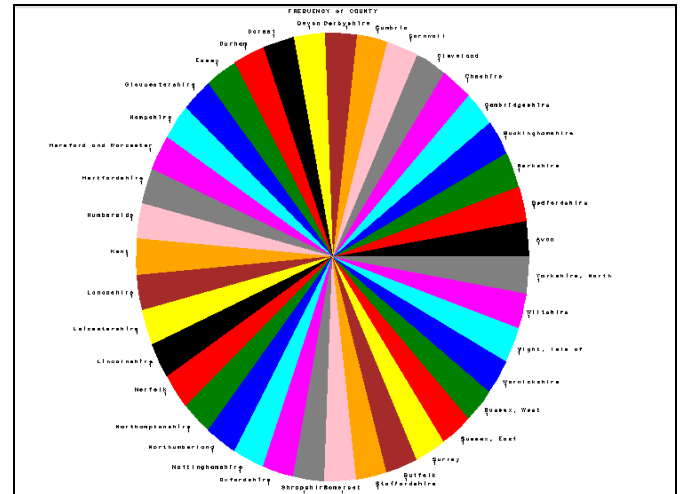


```

Program Editor - gr053_c
Command ==>
00001 *** Pie Chart of County with Other Option Example;
00002
00003 goptions reset=all reset=global
00004         hpos=150 vpos=120;
00005         pattern1 v=s;
00006         pattern2 v=s;
00007         pattern3 v=s;
00008
00009 proc gchart data=saved.ukpop;
00010     pie county / other=0.1;
00011 run;
00012 quit;

```

Finally, we see that all the slices have been assigned solid colors.



Review the log and you will see messages stating that not all the slices in the pie were labeled.

```

Log - (Untitled)
Command ==>
WARNING: Text boundaries for the pie/donut slices overlap.
NOTE: Two or more pie/donut slice labels have been overwritten. This can occur
several adjacent small slices. To avoid overwriting labels you can use on
following techniques:
1. Increase GOPTIONS HPOS= and VPOS= to make the text smaller.
2. Reduce the size of the text with GOPTIONS HTEXT=.
3. Use a different labeling method and perhaps a legend.
4. Use the OTHER= option on the PIE/PIE3D/DONUT statement to group small
single category.
5. Use the MIDPOINTS= or ANGLE= options on the PIE/PIE3D/ DONUT statement
order of slices or starting angle for the first slice so that two small
fall next to each other.
6. Use a VBAR or HBAR chart instead of a PIE/DONUT chart.
7. Use the graphics editor to reposition or resize the labels.
Consult the proper documentation for details on any of these techniques.

```

This is addressed in the following section.

Labeling just the Midpoint

Let's take the previously submitted code and add the slice, value and percent options.

```

Program Editor - gr054
Command ==>
00001 *** Labeling Options Example;
00002
00003 goptions reset=all reset=global hpos=85 vpos=85;
00004         pattern1 v=s;
00005         pattern2 v=s;
00006         pattern3 v=s;
00007
00008 proc gchart data=saved.ukpop;
00009     pie county / other=0.1 slice=arrow value=none percent=inside;
00010 run;
00011 quit;

```

Submit the code.

Now let's fill all the slices in the pie with solid colors.

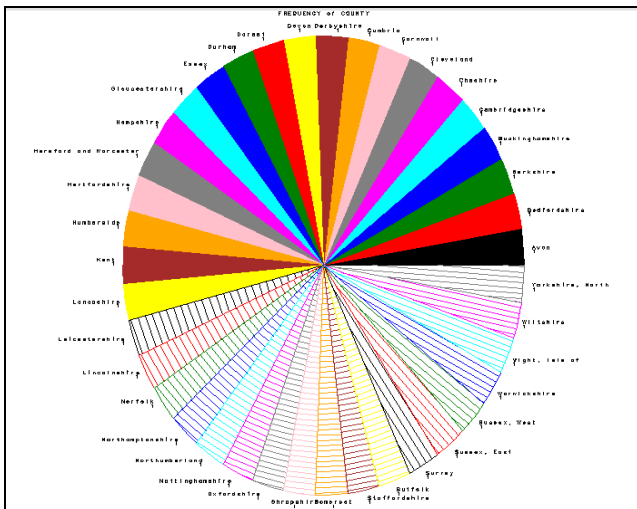
Let's add a pattern option, as follows:

```

Program Editor - gr053_b
Command ==>
00001 *** Pie Chart of County with Other Option Example;
00002
00003 goptions reset=all reset=global
00004         hpos=150 vpos=120;
00005         pattern1 v=s;
00006
00007 proc gchart data=saved.ukpop;
00008     pie county / other=0.1;
00009 run;
00010 quit;

```

We see that about half the total number of slices in the pie are solid. This is because the large number of slices requires more than one pattern specification.

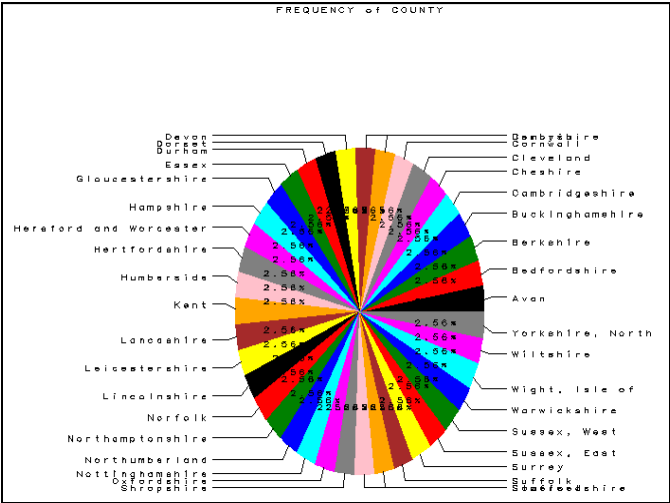


We need to keep adding pattern statements (pattern1, pattern2, pattern3) until all the slices are solid.

In this example, we need to add three pattern statements to make all the slices in the pie solid colors.

There are so many slices in this pie that they have exhausted three patterns.

We see that the output is displayed as follows.



Regional Percentages

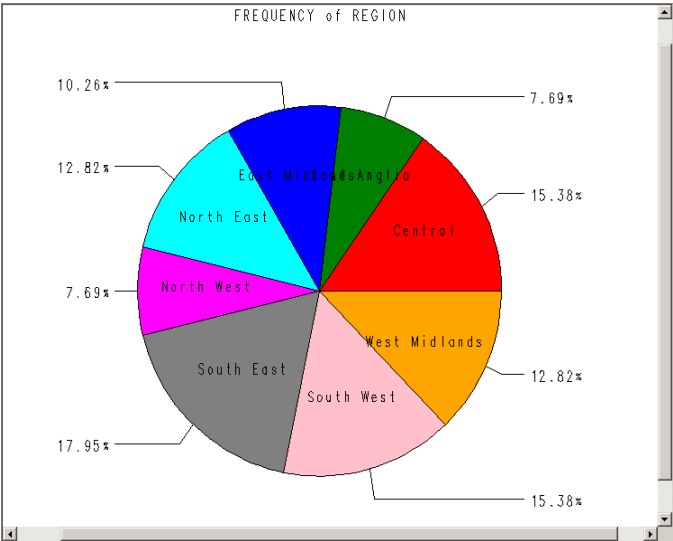
In the previous examples, by default the frequency was displayed on the charts.

When the SUMVAR option is absent, the frequency, or the number of observations in the data set, is charted.

Let's label each slice of the pie, as displayed in the following code.

```
Program Editor - gr055
Command ==>
00001 *** Regional Percentages Example;
00002
00003 goptions reset=all reset=global;
00004
00005 proc gchart data=saved.ukpop;
00006   pie region / other=0.1 slice=inside value=none percent=arrow;
00007 run;
00008 quit;
```

We see that there are no warning messages in the Log.



Contour Plots

Contour plots present output in three-dimensional forms.

There are two procedures for plotting in three dimensions:

1. GCONTOUR

2. G3D

In GCONTOUR the response to two independent variables is displayed as different contour lines

G3D is a 3-dimensional perspective representation, either as a 'sheet' of joined points or a scatter plot.

In this section, we will demonstrate how to produce the various shapes of a plot.

It is beyond the scope of this course to explain all the options and features available.

The next sections explain how to create basic shapes and enhance plots using features like patterns, axis and legend specifications and definitions of different plotting symbols.

The data set used for our examples is typical of monitoring drug dosages.

The two independent variables used are the dosage, which is recorded in milligrams and the frequency of dosage, which is shown in hours.

The response to the dosage and frequency is measured in terms of the patients' pulse rate.

Here is a sample of the data:

```
Program Editor - gr072
Command ==>
00001 *** Contour Data Example 1;
00002
00003 proc print data=saved.contour;
00004 run;
```

Obs	DOSE	REGULARY	PULSE
1	40	4	115
2	40	5	130
3	40	6	130
4	40	7	134
5	40	8	140
6	50	4	115
7	50	5	120
8	50	6	122
9	50	7	128
10	50	8	140
11	60	4	95
12	60	5	99
13	60	6	115
14	60	7	120
15	60	8	133

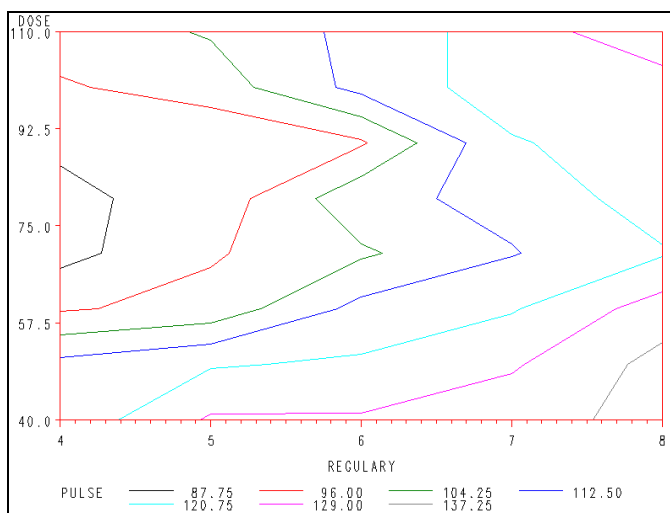
Dose is measured in milligrams and regularly is measured in number of hours.

The goal here is to lower the patients' pulse rate as much as possible based on the combination of dosage and regularity.

The first example shows the default contour plot produced with the Y*X=N form of plot statement.

The vertical axis is the Y, the horizontal is the X and the contour, or the response axis, is represented by values of the N.

```
Program Editor - gr073
Command ==>
00001 *** Contour Example 2;
00002
00003 goptions reset=all reset=global;
00004
00005 proc gcontour data=saved.contour;
00006   plot dose*regulary=pulse;
00007 run;
```



The SAS System has chosen the line types, axis gradations and contour steps.

We can infer a pattern from the data that the patient response is getting better with increased dose and regularity.

We can also see that dosages greater than around 75 milligrams at 4-hour intervals (regularly) do not decrease the pulse rate - in fact, it increases again.

The goal is to find the point at which the pulse rate is minimized. This trend will be shown in many ways in the following graphs.

This example shows that for a given regularity of dosage, increasing dosage results in a decrease in pulse rate. After reaching a certain dosage level pulse rates start to rise again.

The most effective dosage seems to be 75 mg at four-hour intervals.

Increasing the dosage to 110 mg has a less positive response.

LEVELS= Option

The first way we can enhance the plot is to specify the contour levels.

This is done by the LEVELS= option on the plot statement.

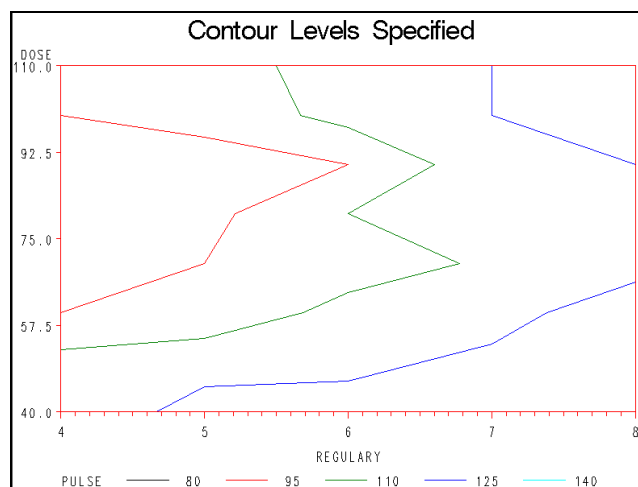
As usual, options on a plot statement follow a /.

The levels are going to start at 80 and increase in increments of 15 up to 140. The legend reflects this information.

```

Program Editor - gr074
Command ==> |
00001 *** Contour Example 3;
00002
00003 goptions reset=all reset=global;
00004
00005 title1 'Contour Levels Specified';
00006 proc gcontour data=saved.contour;
00007 plot dose*regularity=pulse / levels=80 to 140 by 15;
00008 run;

```



Note that the start value (80) is below any value in the data, and a note in the LOG states the same.

```

Log - (Untitled)
Command ==> |
NOTE: The contour level 80 is less than the lowest value of the variable PULSE, which is 85.
NOTE: Writing HTML Body file: sashta57.htm

```

Pattern Option

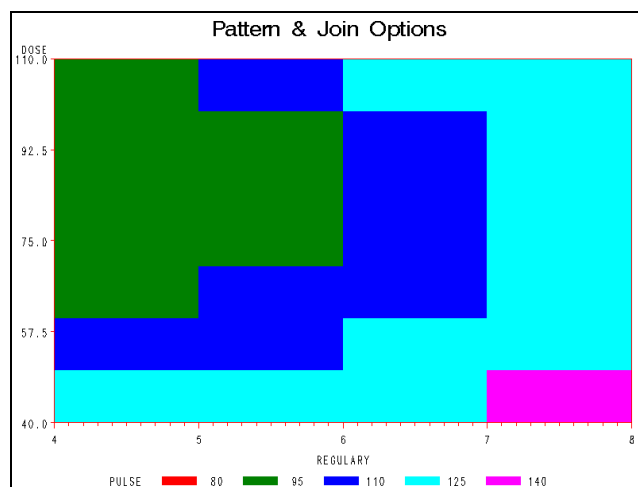
The pattern option displays the graph as 'bricks' of X and Y combinations. The contour passes through it to be patterned according to value.

The join option joins together areas of equal response.

```

Program Editor - gr075
Command ==> |
00001 *** Contour Example 4;
00002
00003 goptions reset=all reset=global;
00004
00005 title1 'Pattern & Join Options';
00006 proc gcontour data=saved.contour;
00007 plot dose*regularity=pulse / levels=80 to 140 by 15
00008 pattern join;
00009 run;

```



3-D Plots

PROC G3D can depict three-dimensional graphs.

PROC G3D generates two types of plots - 'sheets' of joined points across a surface and scatter plots.

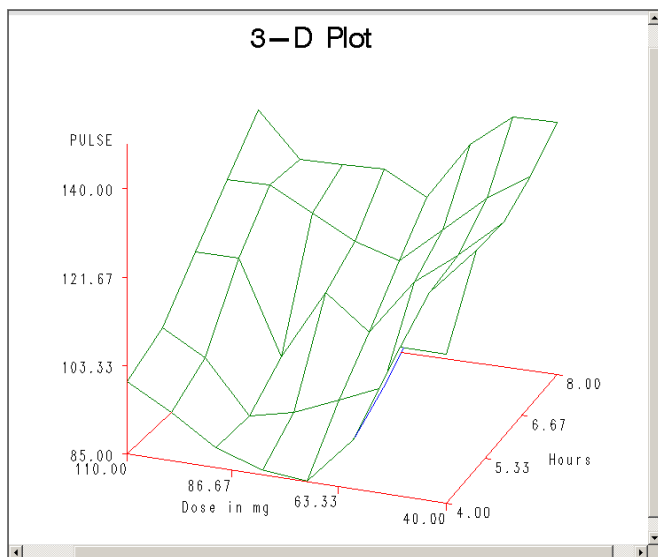
Note that if any interpolation is required between the joined points of the sheet, the data must first be processed using PROC G3GRID.

Using the same drug dosage data, we can display the relationship between the two independent variables (DOSE and REGULARY) and the dependent response variable (PULSE).

Using the code below, we get a three-dimensional presentation of the information where the exact points are joined together and graphed.

```
Program Editor - gr076
Command ==>
00001 *** G3D Example 1;
00002
00003 goptions reset=all reset=global;
00004
00005 title1 '3-D Plot';
00006 proc g3d data=saved.contour;
00007   label dose='Dose in mg' regulary='Hours';
00008   plot dose*regulary=pulse;
00009 run;
```

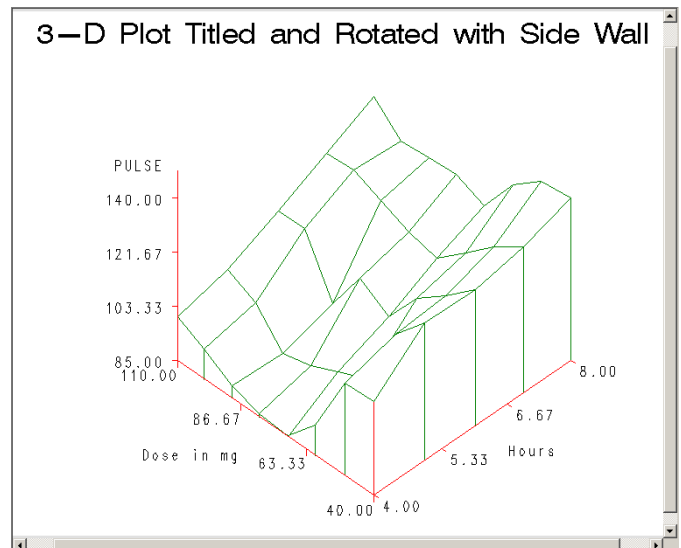
The example shows the very simple plot statement, again in the form $Y \times X = N$.



Our first enhancement is to rotate the plot around the Z-axis using the rotate option and tilt it towards the observer by tilting around the Y-axis.

Side 'walls' will also be displayed for the sides of the plot.

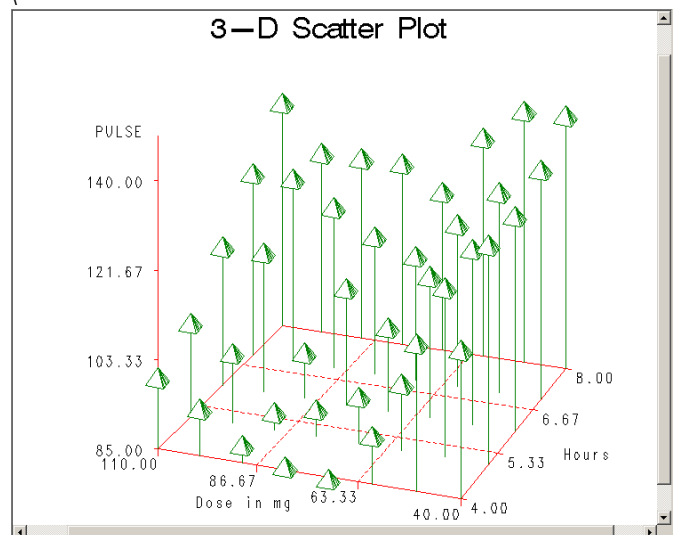
```
Program Editor - gr077
Command ==>
00001 *** G3D Example 2;
00002
00003 goptions reset=all reset=global;
00004
00005 title1 '3-D Plot Titled and Rotated with Side Wall';
00006 proc g3d data=saved.contour;
00007   label dose='Dose in mg' regulary='Hours';
00008   plot dose*regulary=pulse / rotate=45 tilt=45 side;
00009 run;
```



The next example shows a very simple example of the scatter plot.

```
Program Editor - gr078
Command ==>
00001 *** G3D Example 3;
00002
00003 goptions reset=all reset=global;
00004
00005 title1 '3-D Scatter Plot';
00006 proc g3d data=saved.contour;
00007   label dose='Dose in mg' regulary='Hours';
00008   scatter dose*regulary=pulse;
00009 run;
```

To produce this form simply substitute SCATTER for plot and specify the graph in the $Y \times X = N$ form.



The pyramid shapes are produced by default.

We can change the pyramid shape by using the shape option and specifying a variable in the input data set.

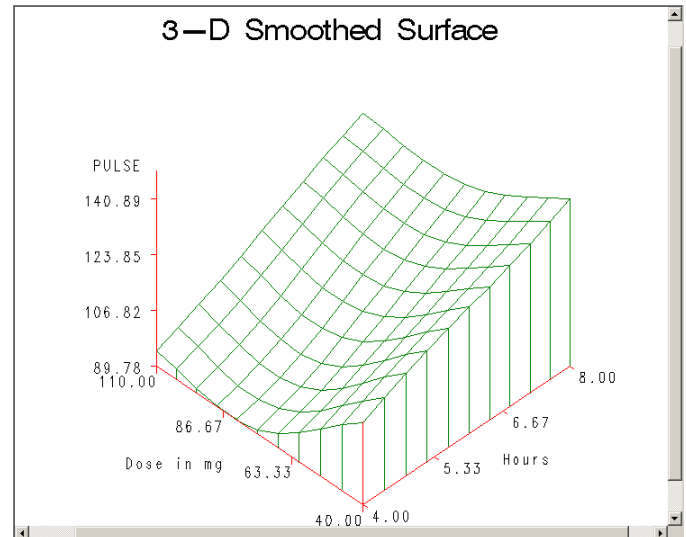
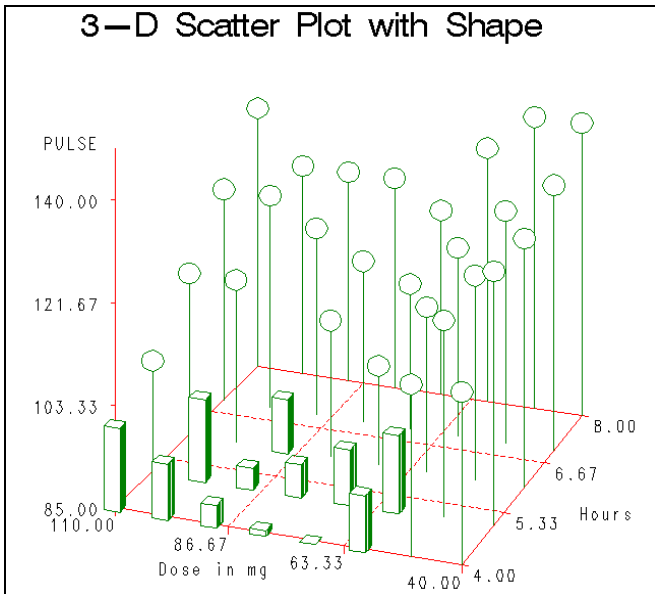
In the next example, we have specified a prism for values of PULSE of 100 and below.

Values above these values are assigned the balloon shape.

```

Program Editor - gr079
Command ==>
00001 *** G3D Example 4;
00002
00003 goptions reset=all reset=global;
00004
00005 data contour2;
00006   set saved.contour;
00007   if pulse le 100 then shape='PRISM';
00008   else shape='BALLOON';
00009 run;
00010 title1 '3-D Scatter Plot with Shape';
00011 proc g3d data=contour2;
00012   label dose='Dose in mg' regular='Hours';
00013   scatter dose*regular=pulse / shape=shape;
00014 run;

```



G3GRID Procedure

We can use the G3GRID to smooth the values in our input data set and produce an improved G3D.

See the user guide for the amazing algorithm!

```

Program Editor - gr080
Command ==>
00001 *** G3GRID Example;
00002
00003 goptions reset=all reset=global;
00004
00005 proc g3grid data=saved.contour out=smooth;
00006   grid dose*regular=pulse/smooth=1 spline;
00007 run;
00008 title1 '3-D Smoothed Surface';
00009 proc g3d data=smooth;
00010   label dose='Dose in mg' regular='Hours';
00011   plot dose*regular=pulse / tilt=45 rotate=45 side;
00012 run;

```