Introduction

The busgrap pre-processor is intended to let you embed business type graphics within the flow of your groff document. All the graphics are produced using groff's native drawing primitives, so the results are pure vector based drawings, no bit mapped graphics. Each graph quires you to embed a call to macro ".BGS" followed by a set of parameters, described below, which are terminated by a call to ".BGE". Between these two macro calls only busgrap parameters are expected, any normal groff language will cause an error. The format of the ".BGS" parameter is:-

```
.BGS name
```

Where 'name' can be "pi" or "graph". The difference determines whether the graph will be a pie chart or a line/bar chart.

In order to place a graph in the document you need to specify a "frame" which will contain the actual graph when it is generated.

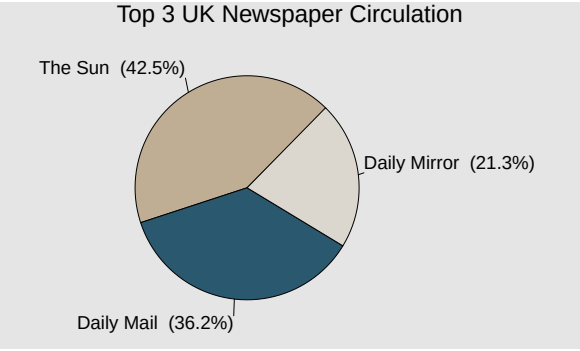
The Frame

In order to place a graph in the document you need to specify a "frame" which will contain the actual graph when it is generated. Busgrap will try to fit your graph within the given frame, but it is not very intelligent so it may not be successful in keeping within the frame, if this happens, increase the size of the frame until it is successful, or reduce the size of labels, etc..

The series labels, (the name associated with the series of data you are plotting), can either be shown on the graph or in a separate "key box" which is a separate box holding the legend of labels. If you define a key box it must be included in the area defined as the frame.

These are the parameters which can affect a frame:- (note throughout that " ^ " represent a tab character)

```
Frame:width ^ height      # default: 6c ^ 5c
Origin:relx ^ rely
Just:{left|centre|right}  # default: left
Flow:{yes|no}             # default: no
Border:size               # default: zero (no border)
BRDcolour:                # default: black border
BGcol:colour              # default: no frame background
```

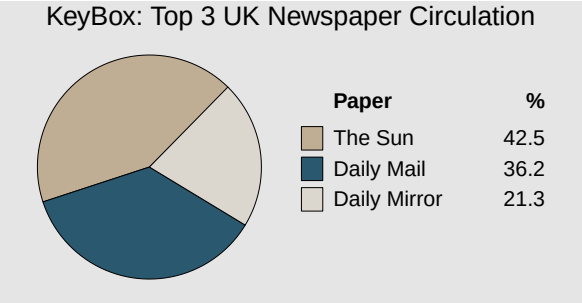


When dealing with pie charts the Origin parameter moves the centre of the pie, the x and y offsets are relative to the top left corner of the frame. The default value is to centre the pie within the frame. For line or bar charts it has a slightly different format which is described later.

The "Flow" parameter causes the text following the graph to be flowed down the side of the graph. It is only active if the "Just" parameter is left or right, you cannot flow test if the graph is centred.

This example has Flow turned on and the text is flowing down the right side of the graph. Along with these parameters there are a set which affect any key box required:-

```
KeyBox:{yes:no}           # default: no (series names shown within the graph)
BoxFrame:xoffset ^ yoffset { ^ width ^ height}
BoxHeads:Col1 ^ Col2 ^ Col3# default: Name ^ Value ^ %
BoxLabels:template        # default: $text ^ $value ^ $percent
BoxTabs:groff tab string  # default: 0cL 2.8cR 4cR
```



We will now show the same graph but with a KeyBox instead of labelling the graph on the pie itself. We will need to move the origin to the left to leave room for the KeyBox on the right. The contents of the box will be set so that the BoxHeads and BoxLabels only show the text and percent.

Using a KeyBox with a legend for the pie can often produce a clearer graph for the reader, and you can use a slightly larger font size.

Graphing Data

The data for graphing is entered as series of data. The format is slightly different, depending on whether you wish to produce a pie chart or a line/bar chart. For pie chart each series is a wedge of pie and just consists of the name and a value, so the format is simply:-

```
Series:name ^ value
Label:template            # default: $text ($percent%)
```

The Series parameter is the only graph parameter which may appear multiple times in the graph definition, once for each wedge. The software will add up the value in each series and calculate the percentage for each pie.

The label parameter controls how the data is labelled, when you choose to have inline labelling rather than a separate KeyBox. It performs a similar purpose to the BoxLabel parameter shown above, but produces a string, rather than a tab separated table. In the template there are three inbuilt variables which can be used in the template string: $text is replaced by the name of the series, $value is replaced with the value passed with a particular name, and $percent is replaced with the value as a percentage of the total of all series values added together. Also, note that the value for each Series can be a "formatted"" number, i.e. may have currency symbol or thousand separators, it will be this string which is passed back as $value, rather than a pure number.

If you are creating a line or bar chart, the following parameters control the data:-

```
Series:type ^ name ^ value1 ^ value2 ^ ...
Steps:number
Floor:{C}value
```

The series parameter now has an extra field called "type" and each series can contain multiple values. There are two types of series you must supply: the "X" type (which labels the X-axis of the graph) and at least 1 numerical type, i.e. the type is 1-n (each numbered series contains the values for one line in the graph). Sometimes the X axis will be time based. If the dates are given as mmm-yy{yy} or mm/yy{yy} then the value is recognised as a valid date and the X-labels will be formatted on two lines for the year and month. The X-type series should hold a label for each data point you intend to graph, so if your data is monthly and you have 12 months worth of data there should be 12 values in the X-type (and 12 in each of the numbered series parameters).

Another Series type is "A" (anchor) this should hold one value per numbered series and is the starting value for that series, so lines are drawn from the Y-axis to the first data point. If, for example, you have two a data set which holds the value of a portfolio, and one for a financial benchmark. Both data sets start in January 2014. You rebase both data sets so that they start at 100 and the following figures will show gain/loss for each following month. In this case you would remove the first data point from both series, and the X-series and add two values of 100 to the A-series parameter. The effect of this would be for the lines for both series starting at the 100 point on the Y-axis to the first data point which would be February 2014.

The final series type is "B". This can be used instead of Series 1 (you must not have a series 1 if you have a series "B". The only difference is that the data in a Series B is plotted as an area graph, i.e. the area under the line is filled with colour. Since this Base series is a replacement for Series 1 it plotted first so the lines of any subsequent series are plotted on top of the area.

Using all the values in the numbered series, busgrap will calculate the appropriate Y-scale labels to use, it

tries hard to settle on a scale which is "natural". The parameter Steps influences the algorithm, by telling it how many steps you would prefer labelled in the Y-axis, but it is quite stubborn in using what it thinks is best! The Floor of the graph is normally the bottom of the graph but the Floor parameter can be used to change this, the value given will be used as the new floor. If the floor value is prefixed by "C" the floor will be centred in the graph. This is probably only useful when doing bar chart since bars are drawn upwards or downwards from the floor, so if you are graphing both positive and negative data as a bar chart it can help clarity if you use Floor:0. The other occasion to use Floor, for all types of line graph is when the Y-axis does not start from zero but you would like it to start from zero.

X-labels can be hierarchical, up to 4 levels. Dates are a typical example, where the date can be split into day month and year. If busgrap recognises a date format it will split the date into a hierarchical form showing the date on different lines under the X axis. The formats it recognises are:-

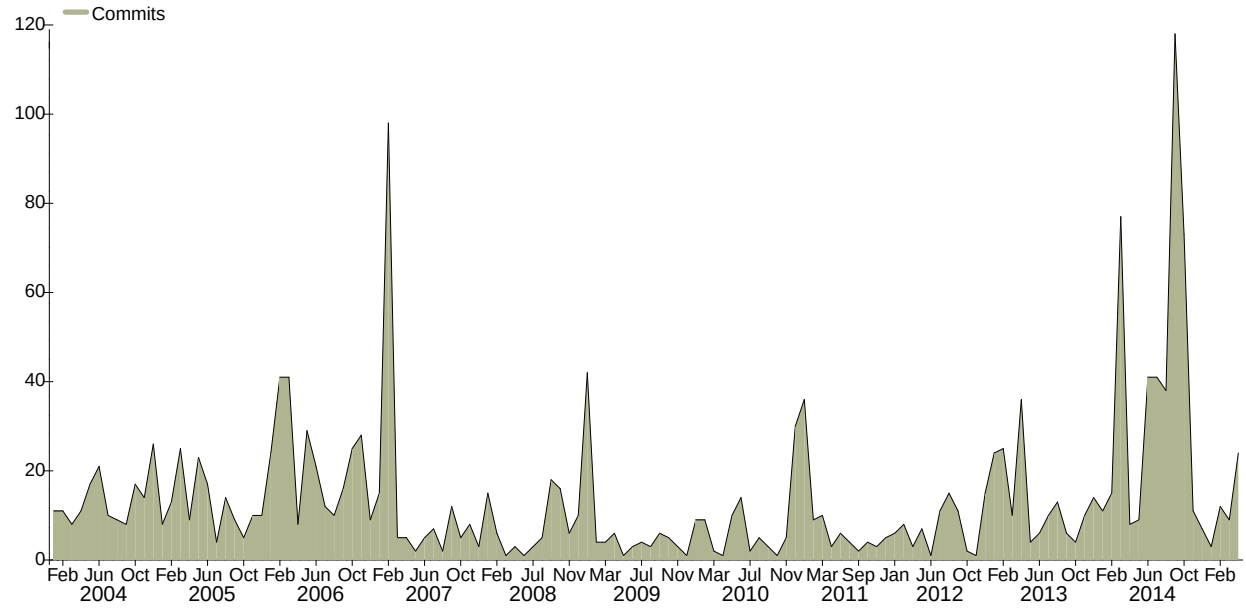
MMM-##(##)	Convert to month/year.
#Q##(##)	Convert to quarter/year (Jan-Mar, Apr-Jun, Aug-Sep, Oct-Dec)
#H##(##)	Convert to period/year (Jan-Jun, Aug-Dec)

Where '#' is a numeric, and MMM is an abbreviated month name. A more generic method of creating these hierarchical labels is:-

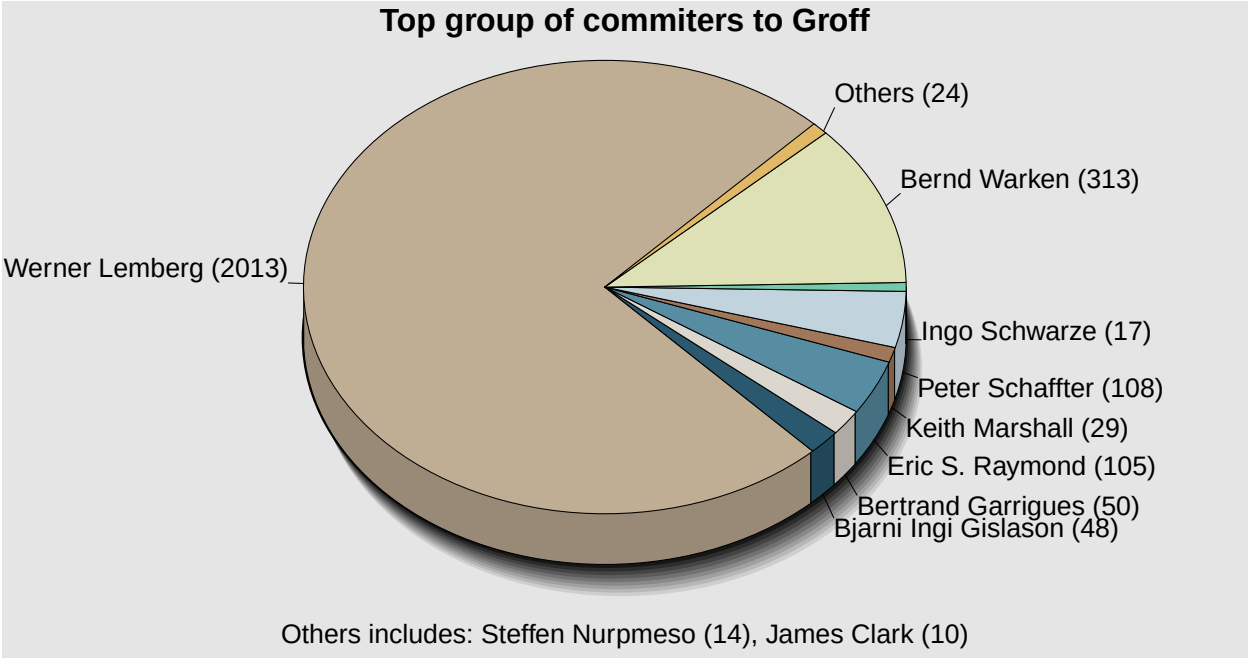
level1~level2~level3~level4

If the label cannot be split in the above forms it will be used as the complete label. In all cases busgrap will try to avoid overwriting labels by skipping them if there is not enough room for them. In the next example the date has been split into two lines of X-labels. But, in the first level (months), there is insufficient room to include all the month labels, so some have been left out.

In this example, which shows the number of commits to groff by month since January 2004, only one series has been graphed, but it has been specified as a Series:B, so it produced an area graph.



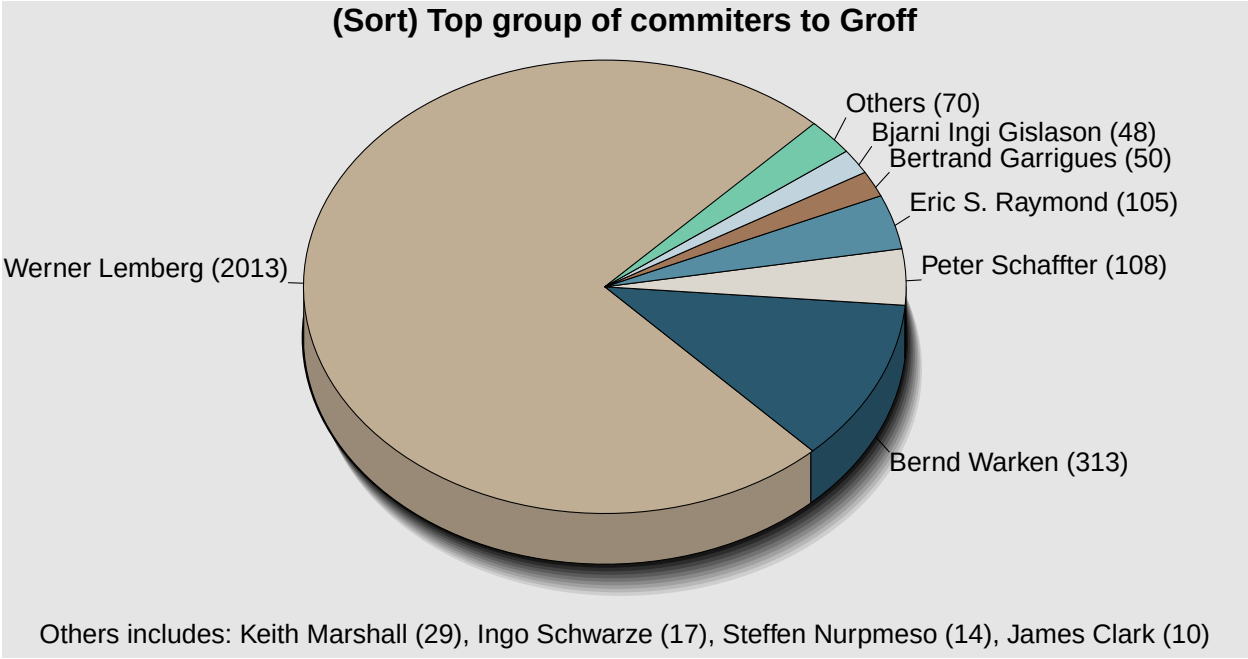
The data comes from the git log and the next graph comes from the same source.



In this pie chart you can see that two of the series have been automatically combined into one segment, called "Others", this is because the segments would be too small to have individual inline labels.

```
Others:percentage          # default: 2.05
Sort:yes|no                # default: yes
```

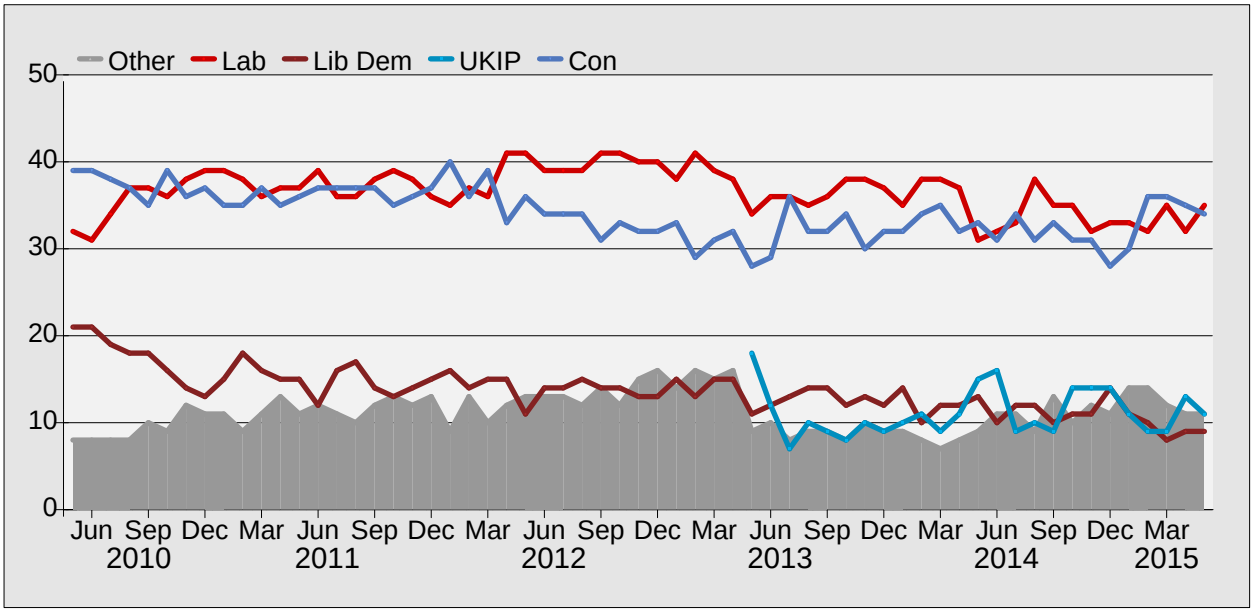
The Others parameter specifies the minimum percentage, if a particular segment is below this size then it is a candidate for combination. Whether it will be combined depends on the size of its neighbours, since if both neighbours are large enough there will be room for the small segment to be labelled. This is why the Sort parameter has an effect, since when Sort is turned on the segments are sorted decreasing from largest to smallest, so the smallest segments will clump together, so more will be put into Others. With Sort turned off, and careful ordering of the series, you can arrange so the smallest are interspersed with the largest. This is the same graph with Sort turned on and Others increased slightly to avoid overwriting:-



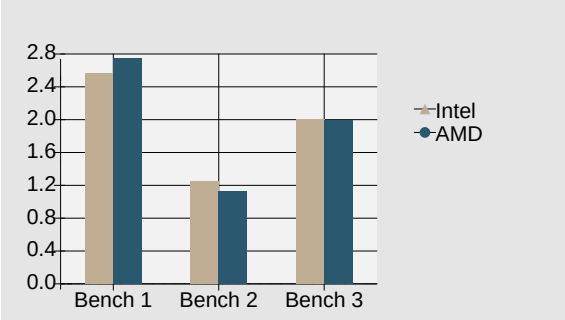
Other methods of avoiding labels overwriting each other are to adjust the point size of the labels and/or increase the size of the pie, see affecting pie style below.

In any statistical work it is useful to have a "missing value". Busgrap uses a single "." (dot) to signify a value is missing, so rather than have a blank value, which equates to zero, use the dot instead.

In the following graph which shows the polling figures for UK political parties from May 2010, the UKIP party was not established until May 2013, so the missing values have been replaced with dots in the Series parameter for the UKIP party.



When the input data includes decimal places the algorithm which decides on the Y-labels may or may not use decimals. The decision is based on how close the range of numbers is between the lowest and the highest. When the range is very close together the algorithm will choose an appropriate number of decimals. The choice can be overridden by setting parameter PDecimals to the number of places required. In this example the algorithm has chosen 1 decimal place, which is probably the correct choice in this case.



To graph a particular series as bars, instead of lines you just put "BAR:" in front of the series name, so two series in this graph were called "BAR:Intel" and "BAR:AMD". It is perfectly permissible to mix bars and lines in the same graph. Any bars will always be drawn first then any lines appear over the top.

The actual data to produce this bar chart was:-

```
    Series:X ^  ^ Bench 1 ^ Bench 2 ^ Bench 3
    Series:1 ^ BAR:Intel ^ 2.566 ^ 1.253 ^ 2.000
    Series:2 ^ BAR:AMD ^ 2.744 ^ 1.122 ^ 1.990
```

So, even though the data was to 3 decimals, only 1 place of decimals was relevent to the graph labels.

It is possible to have a sideways bar chart if you use the parameter:-

```
Horizontal:yes
```

When you select a horizontal bar chart, the format of the Series parameter changes:-

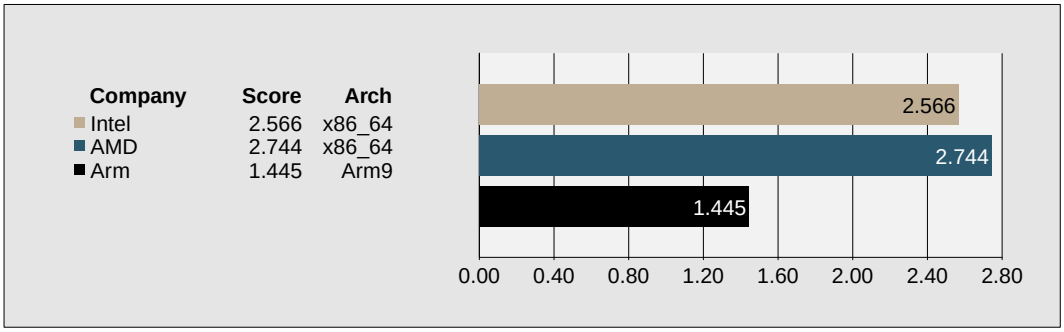
```
Series:type ^ name ^ value {^ col1 ...}
```

The type can only be a number. The name is printed on the bar, but is allowed to be null if this is not required. Only a single value is allowed per series. Multiple col items may appear, which are treated the same way as BoxLabels inside a KeyBox, i.e. they are printed as a little table underneath the BoxHeads

according to the settings in BoxTabs. There can be multiple col values which make up the table. The following graph used this:-

```
Horizontal:yes
Series:1 ^ 2.566 ^ 2.566 ^ Intel ^ 2.566 ^ x86_64
Series:2 ^ 2.744 ^ 2.744 ^ AMD ^ 2.744 ^ x86_64
Series:3 ^ 1.445 ^ 1.445 ^ Arm ^ 1.445 ^ Arm9
```

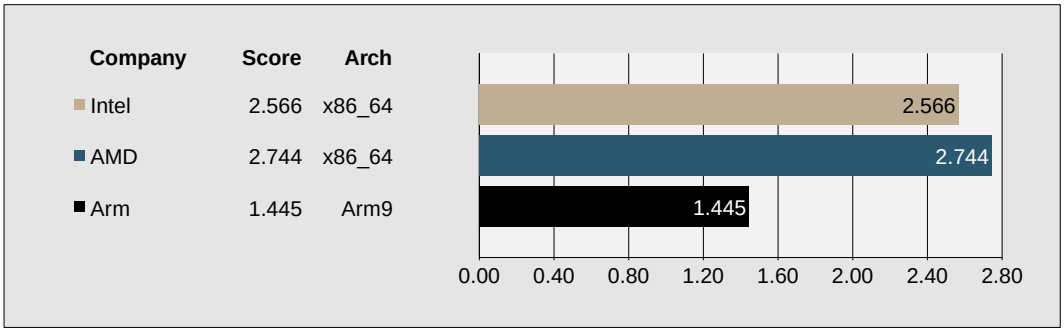
Only bar charts can be Horizontal so pre-pending "BAR:" to the name is not necessary.



If you specify:-

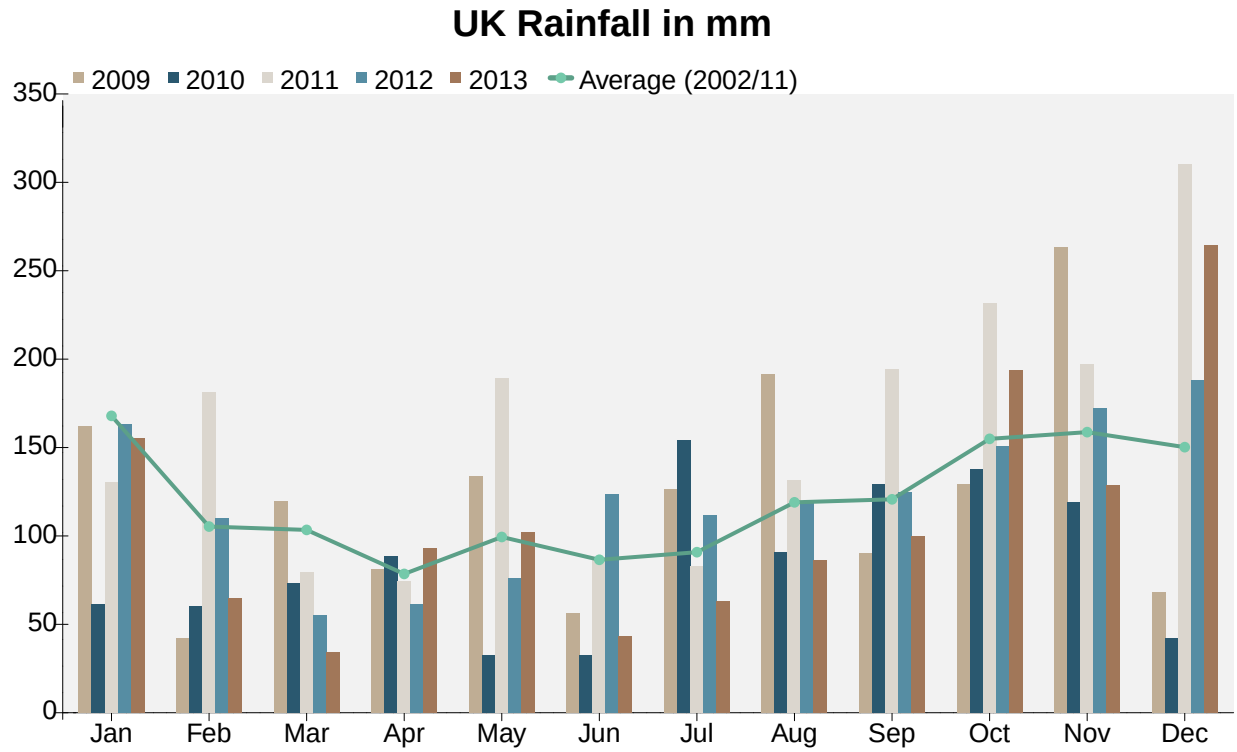
```
Sync:yes
```

The entries in the KeyBox will be aligned with their corresponding bar.



The UK rainfall graph illustrates the combination of bar and line series in one graph. Using this series data:-

```
Series:X ^  ^ Jan ^ Feb ^ Mar ^ Apr   ...
Series:1 ^ BAR:2009 ^ 161.9 ^ 42.1 ^ 119.5 ^ 81.2   ...
Series:2 ^ BAR:2010 ^ 61.3 ^ 60 ^ 73.4 ^ 88.3   ...
Series:3 ^ BAR:2011 ^ 130.6 ^ 181.5 ^ 79.7 ^ 74.4   ...
Series:4 ^ BAR:2012 ^ 163.4 ^ 109.8 ^ 55 ^ 61.4   ...
Series:5 ^ BAR:2013 ^ 155.5 ^ 64.7 ^ 34.1 ^ 92.8   ...
Series:6 ^ Average (2002/11) ^ 167.9 ^ 105.3 ^ 103.4 ^ 78.5   ...
```



If there is a lot of data to load into a graph, you may prefer keep the data separate in its own file. The Sfile: parameter can name a file which will be used to provide the series data, instead of having the data embedded in the .trf file. The format of this file is identical to the Series: parameter except the "Series:" can be dropped from the start of each line. Another reason for using a separate Sfile is if your workflow dynamically creates the data and you wish to pull it into a graph by calling groff. So the series for the Rain graph could have been written to a file called UKRain.csv and included by using:-

```
SFile:UKRain.csv
```

The UKRain.csv file has one record per series using the tab character as the field separator.

Parameter List

Colours

```
BGColour:col              # Colours background of the frame (default: none)
BRDColour:col             # Colour of border round frame (default :black)
TextColour:col            # Colour of text labels (default: black)
LineColour:col            # (pie only) Affects segment lines (default: black)
WallColour:col            # (graph only) Colour drawing area (default: none)
SColours:col ^ col ^ ...  # Colour for each series
PosColour:col             # (bar chart only) Bar colour if +ve (default: none)
NegColour:col             # (bar chart only) Bar colour if -ve (default: none)
Coloured:yes|no           # Yes means segments are only filled (default: no)
Darker:number             # Between 0 and 1, factor darkens side of pie (0.8)
```

In each case "col" is any valid groff colour definition as you would specify to .defcolor, or a previously named colour defined earlier in the document (or ps.tmac).

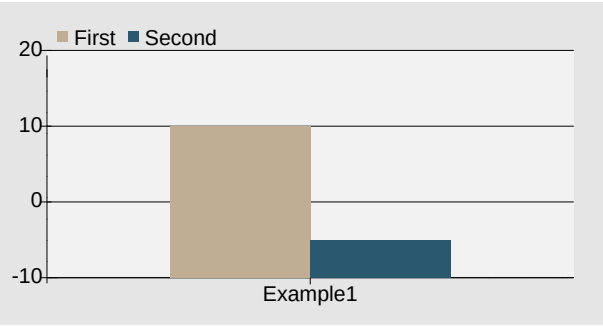
Normally, with bar charts, the colours of the bars are taken from the series colours (SColours), but there are some kinds of charts where you want to show a different colour dependent on whether the value is positive or negative. A typical example is when you want to show whether something is over or under weight compared to a benchmark. If you enter a PosColour and NegColour the SColours array of colours is ignored and the colour used is dependent on whether the value is positive or negative.

Lines

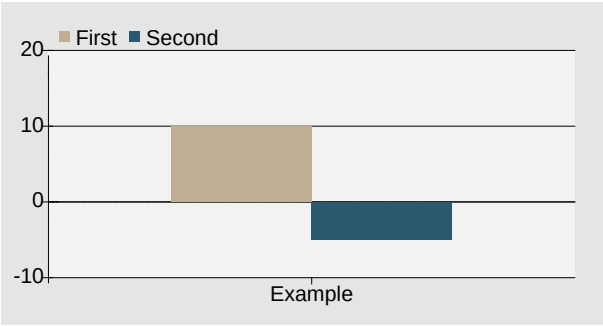
```
Thickness:size            # Thickness of lines used in graph (default: .2p)
Border:size               # Thickness of Frame border (default: 0=no border)
HGrid:yes|no              # (graph only) Plot Horizontal grid lines (default: no)
VGrid:yes|no              # (graph only) Plot Vertical grid lines (default: no)
Floor:value               # Change floor of graph (default: minimum Y on scale)
Steps:value               # (graph only) No of Y-Labels (default: auto(10))
```

Changing the Floor value has two effects. If the value is between the min and max on the line or bar graph then the bottom line (above the X-labels) is moved to the position corresponding on the Y-scale. This is particularly useful for bars which are showing a negative number. Setting Floor to zero will cause the negative bars to be plotted downwards from the zero line.

No Floor given

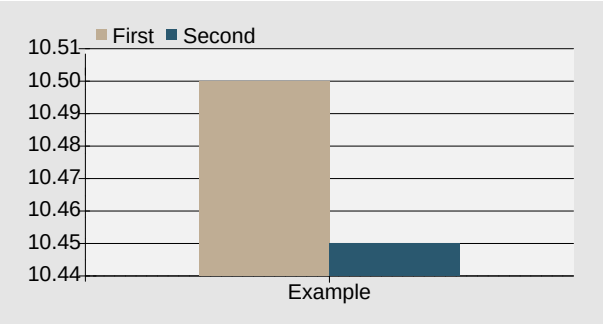


Floor:0

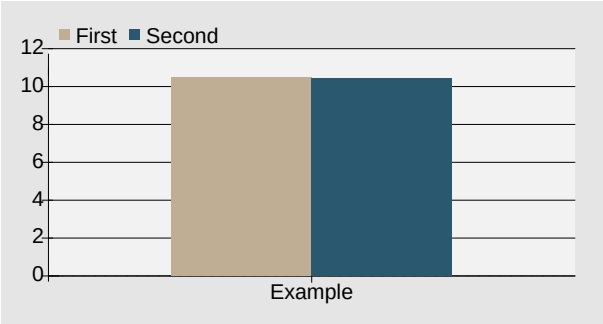


The other use is when the values for several bars are all very close to gether, the min/max difference could be very small which could lead to the step between the Y-scale to be tiny. One bar will be near the bottom of the graph and one will be near the top, but the difference between the two values could be tiny and the Y-scale labels would be to two decimals. This would give a false impression of the relative values of the two bars. If this happens set the floor to a value less than the minimum Y-label, which will cause the range to be stretched, making the two bars length much closer together. Using bar values of 10.5 and 10.45 gives:-

No Floor given



Floor:0



On bar and line charts, the actual min/max Y-labels, the step increment and the number of steps, are chosen algorithmically. Changing the value of steps affects the algorithm, but may not produce the exact number of steps you specified.

Chart Areas

```
Frame:width ^ depth       # Dimensions of chart on page (default: 6c ^ 5c)
Origin:xoffset ^ yoffset  # (pi only) Centre pie here (default: centre frame)
Origin:x ^ y ^ wid ^ dep  # (graph only) Shift plot area (default: 0 ^ 0)
KeyBox:yes|no             # Whether a KeyBox is required (default: no)
Just:left|centre|right    # Justify frame on page (default: left)
Flow:yes|no               # Flow text around Frame (default: no)
Caption:text              # (pi only) Caption above pie (default: none)
```

The Origin: parameter allows fine control of the position of the plotting area within the frame. The X and Y offsets are calculated from the top left corner of the Frame. For pie charts it moves the centre of the pie. The reasons you may want to do this includes balancing the look when the length of labels on one side is different from the labels on the other side, or you want the labels to be shown in a KeyBox and you need to shift the pie to allow room for it.

For line and bar charts the origin specifies the top left corner of the actual plot area, and it also should give the width and depth of the plot area. Again, this is to make room for a KeyBox if required.

The KeyBox parameter justs informs busgrap you require a key legend separate from the graph. There are several other parameters, described below, which control the key box.

The Flow parameter can only be used if the justification is left or right, not centred, i.e. it will not flow both left and right of a frame at the same time.

The Caption parameter has a secondary function. If you include the PDFbookmark parameter, it includes the caption text in the overview pane if the output is to a pdf. Even though Caption is not used in pie and line charts, this secondary function is still done, so graph appears in the overview.

The Key Box

```
BoxFrame:x ^ y ^ wid ^ dep# Position of KeyBox relative to top-left of frame
BoxHeads:txt { ^ txt ...} # (pie) KeyBox Heading (default: Name ^ Value ^ %)
BoxHeads:txt { ^ txt ...} # (graph) KeyBox Heading (default: none)
BoxLabels:template        # default: $text ^ $value ^ $percent
BoxTabs:tab               # String describing position of tab stops
```