An Introduction to Stata Graphics

2022

Module taught by Kathy Baisley, Nicholas Magill & Manuela Quaresma
Module materials largely developed by Tim Collier

AN INTRODUCTION TO STATA GRAPHICS 1

1. GETTING HELP WITH STATA GRAPHS 2

2. CREATING GRAPHS WITH THE GRAPHICAL USER INTERFACE (GUI) 4

GUI Example 1: Creating a box plot 4

GUI Example 2: Overlaying plots from the twoway family of graphs 13

3. THE STATA GRAPH EDITOR 20

4. BUILDING A GRAPH IN A STATA DO-FILE 25

4.1 Dealing with long commands in do-files 25

4.2 Working efficiently with macros 26

5. GRAPH OPTIONS 27

5.1 Titles, Labels and Scale 27

5.2 Markers and Lines 31

5.3 Legends 33

5.4 Adding text to Graphs 35

5.5 Plotting on more than one axis 36

5.6 Adding special symbols, subscripts and superscripts 37

6. GRAPH SCHEMES 38

7. SAVING AND EXPORTING GRAPHS 40

7.1 Displaying Graphs in Multiple Windows 40

7.2 Copy-and-Paste 41

7.3 Saving a Graph to Disk 42

7.4 Exporting graphs 43

8. COMBINING GRAPHS 44

9. THE ANATOMY OF A STATA GRAPH 45

10. A FLAVOUR OF STATA'S GRAPHICS POTENTIAL 47

An Introduction to Stata Graphics

Stata offers the potential for producing publication-quality graphs. There are a great many different plot types available including distributional plots (e.g. box-plots, histograms), survival plots (e.g. Kaplan-Meier, Nelson-Aalen), two-way scatter plots, line plots, area plots, range plots, time-series graphs, contour plots and much more.

Graphs can be created using either the *Graphical User Interface* (GUI) accessed via drop-down menus in a point-and-click fashion, or through the command syntax. Most graphs have a series of options, sub-options and even sub-sub-options through which it is possible to control almost every aspect of the final figure. Stata comes with a built-in Graph Editor through which it is possible to add, remove, move or modify elements of a graph.

Given the broad range of plot types and the huge array of options available (the Stata Graphics Reference manual consists of 700+ pages) it is not possible to teach or demonstrate all of Stata's graphics facilities in one or two short sessions. So, firstly, you will be introduced to Stata's comprehensive help facility and pointed towards other sources of help. This should enable you to build upon what is taught here and to continue learning to create effective graphs beyond the course.

You will be shown how to create simple graphs using the Graphical User Interface and then how to create and develop graphs using the command syntax from within Stata's do-file editor. We'll cover adding titles and legends, modifying axis labels, changing colour, size and style of markers and lines and much more.

We'll spend some time looking at how to edit graphs using Stata's built in graph editor.

Along the way we'll cover copying and pasting graphs into other software e.g. Powerpoint, saving and exporting graphs and changing graph schemes.

Finally we will demonstrate a series of different graph types with options giving you a flavour of what it is possible to do using Stata's graphics facilities.

The features described herein are compatible with Stata version 17.

1. Getting Help with Stata graphs

Stata comes with a comprehensive set of easily accessible interactive help files for all of its graph commands and facilities. Stata also comes with a built-in Stata Graphics Reference manual, a 700+ page printable and searchable PDF documentation which links into the existing interactive help file system. On top of this there is also online help available through the Stata website and books such as Michael Mitchell's A Visual Guide to Stata Graphics. More recently Stata have launched the Stata YouTube Channel, which include video instruction on a number of different graph types.

The best starting point from within Stata is to type `help graph intro` in the command window. This will bring up a help viewer giving a general introduction to Stata graphics. There is a suggested reading order for the Stata help files, beginning with "A quick tour" and then `help graph`, `help twoway` and `help scatter`. If you take time to work your way through these in order you will find them extremely helpful.

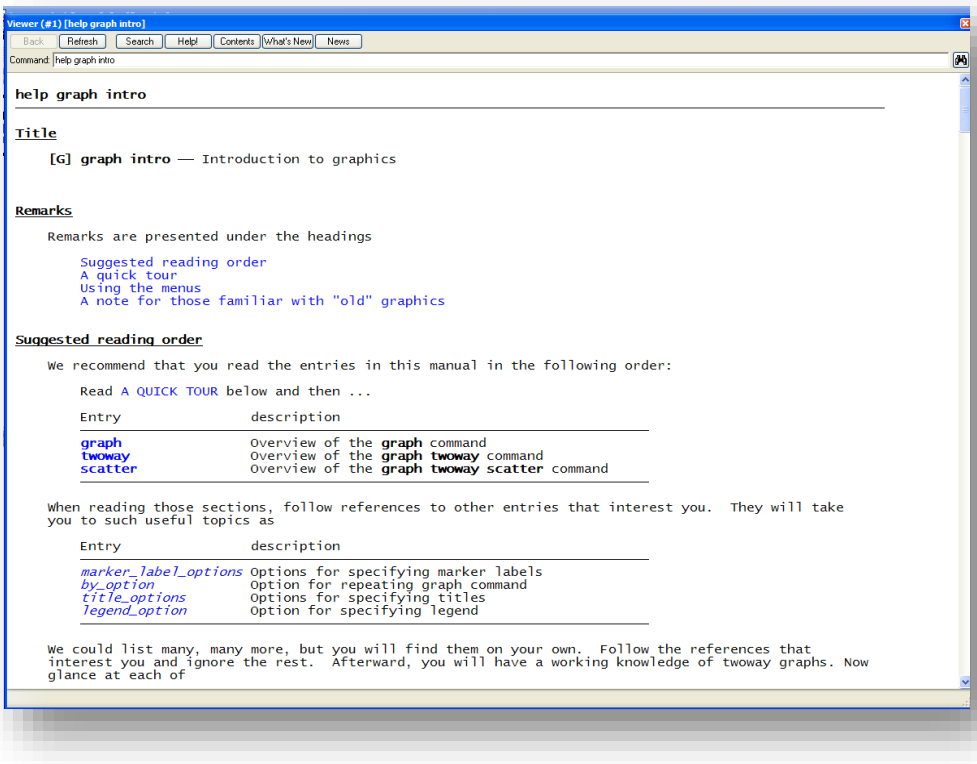


Figure 1.1: The help graph intro viewer

"A quick tour" will demonstrate (using click-to-run examples) how to build up a graph command, starting with the basic graph syntax and gradually adding options (e.g. titles, axis labels etc) to produce a publication-quality finished figure. It demonstrates twoway scatter plots, line graphs, regression fit graphs, histograms, matrix and box plots.

Another source of help with Stata graphs is the Stata web-site (www.Stata.com).

There you will be able to find interactive online tutorials on creating basic graphs in Stata (www.stata.com/support/faqs/graphics/gph/stata-graphs/) as well as information on Stata's graphical capabilities and links to video tutorials.

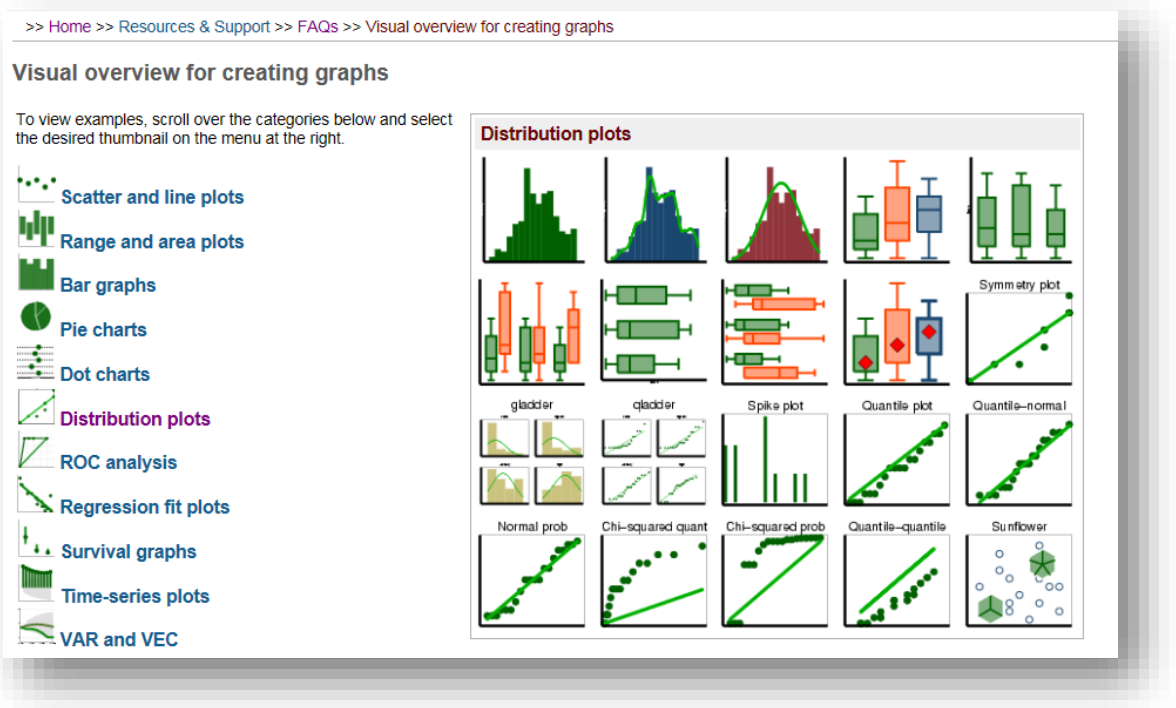


Figure 1.2: Stata website - online tutorial for creating basic graphs

2. Creating Graphs with the Graphical User Interface (GUI)

It is possible to access almost all of Stata's graphics commands using the Graphics drop-down menu (Figure 2.1).

This drop-down menu can also be used for combining graphs, managing graphs and changing graph schemes.

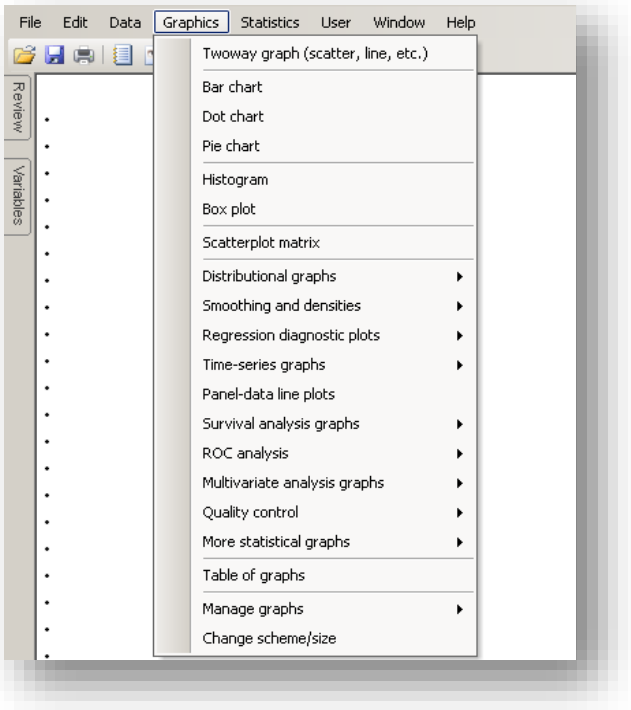


Figure 2.1: The Graphics drop-down menu

We will demonstrate how to use the GUI to produce

(i) a box plot and
(ii) a twoway overlaid plot

GUI Example 1: Creating a box plot

A box plot, sometimes called a box-and-whisker plot, is a plot that can be used to graphically display the distribution of a continuous (or metric) variable. Here we will create a box plot of systolic blood pressure (*sbp*) over categories of age-group (*agegroup*) and obesity (*obese*).

Before creating the plot we will change the working directory and load the data.

- Use the *File > Change Directory* menu to change the working directory to the folder where the graphics data have been saved.

- Use the *File > Open* menu to load the dataset *whs.dta.*

Step 1: From the *Graphics menu* select the option *Box plot*. This launches the *graph box – Box plots* dialog box (see Figure 2.2 below).

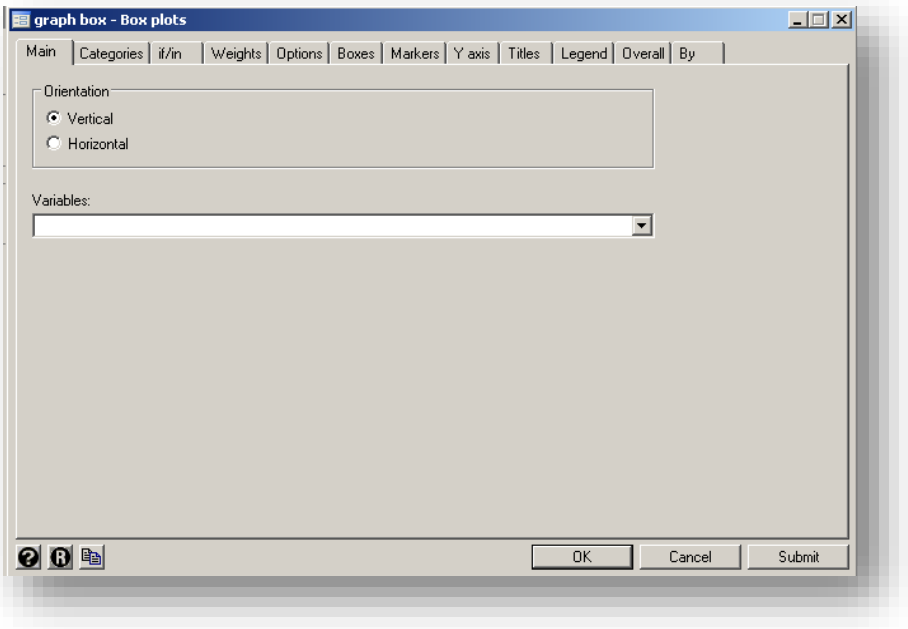


Figure 2.2: The Box plots dialog box

The dialog box has 12 tabs: Main, Categories, if/in, Weights, Options, Boxes, Markers, Y axis, Titles, Legend, Overall and By. We will investigate some of these shortly.

At the bottom of the dialog box are six buttons (see Figure 2.3).

Help Reset Copy

Submit command close dialog box

Close dialog box

Submit command keep dialog box open

Figure 2.3: Dialogue box options

From left to right these are:

Help – open help file related to this command;

Reset – set all options back to their default;

Copy – copy the command syntax for dialog box as currently filled out;

OK – executes command and closes the dialog box;

Cancel – cancels and closes the dialog box without executing the command;

Submit – executes the command leaving the dialog box open.

Step 2: From the *Main* tab we select the variable or variables to be plotted using the variables menu and the orientation e.g. whether the box plots are vertical or horizontal. Here we will select *sbp*. We will leave the orientation as vertical – the default.

Click the *Submit* button to execute the command.

Stata draws the graph in the *Graph viewer window* (Figure 2.4) which has its own set of drop-down menus and short cut buttons which can, among other things, be used to save, print or edit the graph (more on that later).

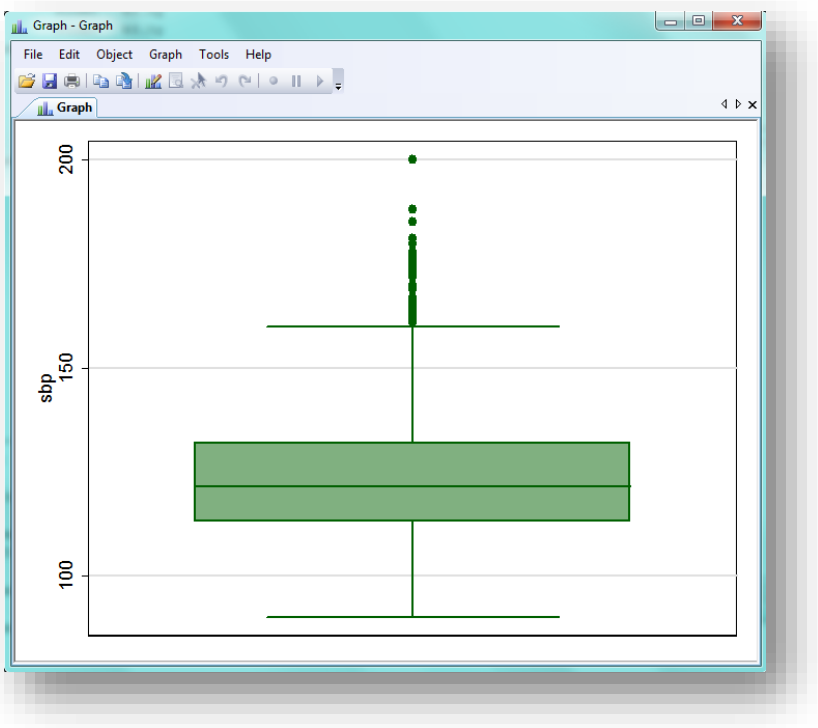


Figure 2.4: Graph window with Box plot of systolic blood pressure

You will see that Stata reproduces the command syntax for the graph in the Review Window and Results Window. Note also that Stata has chosen an appropriate range and number of labels for the y-axis and has used the variable name (*sbp*) as the y-axis title. If the variable had been labelled then the variable label would have been used.

Note that by default Stata has named the graph as '*Graph*' - see the tab at the top of the graph. If we draw another graph Stata, without using the `name(graphname)` option, then this first graph will be overdrawn. The options for naming a graph via the GUI can be found in the *Overall* tab. Once a graph name has been specified any existing graphs open in the graph window will not be overdrawn. Additionally Stata will only redraw a graph with the same name if the option `replace` is specified along with the `name()` option.

Multiple graphs can be drawn either in separate Graph window or as multiple tabs in a single window. These options can be selected either via the drop down menu on the main Stata interface (*Edit > Preferences > Graph preferences...*) or the Graph window (*Edit > Preferences*). Either of these opens the *Graph Preferences* dialog box (see Figure 2.4a) which

includes a tick box for creating multiple graphs as tabs in a single window. The graph preferences window can also be used to select the overall scheme, font and more.

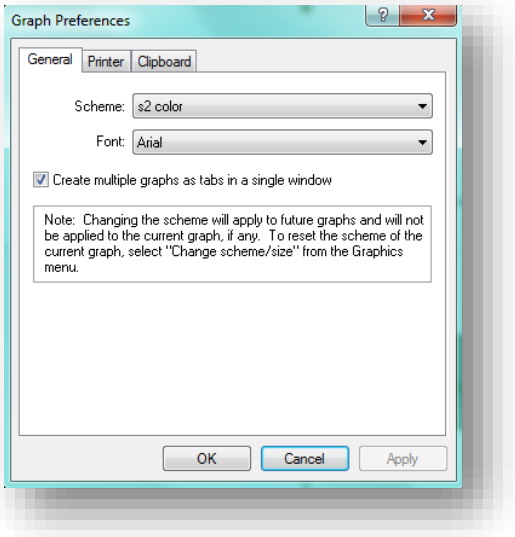


Figure 2.4a: Graph Preferences window

Step 3: So far we have created an overall Box plot of systolic blood pressure for everyone in the dataset. Our goal is to draw a Box plot of systolic blood pressure over categories of age-group and obesity. Returning to the *Box plots* dialog box select the *Categories* tab (see Figure 2.5). You will see that up to three grouping variables can be selected. Tick the *Group 1* option and select the variable *agegroup* using the *Grouping variable* menu.

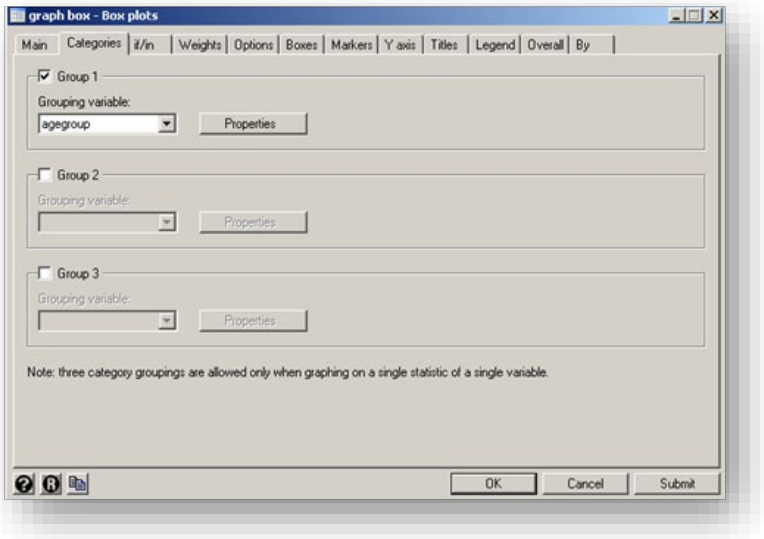


Figure 2.5: Box plot dialog box – Categories tab

Also go to the *Overall* tab and name the graph *sbp_agegroup*. Clicking the *Submit* button produces the graph shown in Figure 2.6.

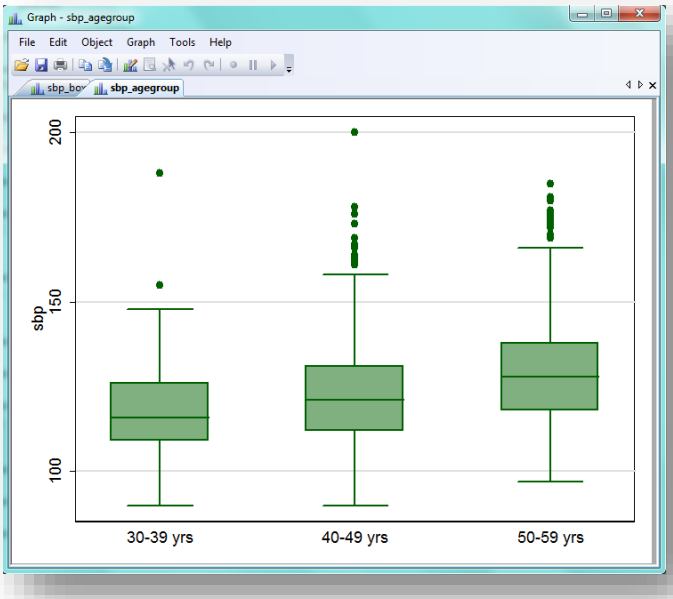


Figure 2.6: Box plot of SBP over categories of age-group.

Look at the command syntax for this graph in the Results Window. Note that by default Stata has used the value labels (*30-39 yrs etc*) that are already attached to the categories of the variable *agegroup*. Next to each of the *Grouping variable* selectors on the *Categories* tab there is a *Properties* option button. Clicking this will open the *Categorical axis and label properties* dialog box via which you can amend the labels etc.

Step 4: Recall that we want to draw the Box plot over the variables *agegroup* and *obese*. We could add *obese* as another category under the *Categories* tab or use the *By* tab; we will do both to demonstrate the difference between these two options. Under the *By* tab (Figure 2.7) tick the *Draw subgraphs for unique value of variables* option and select the variable *obese* using the *variables* menu.

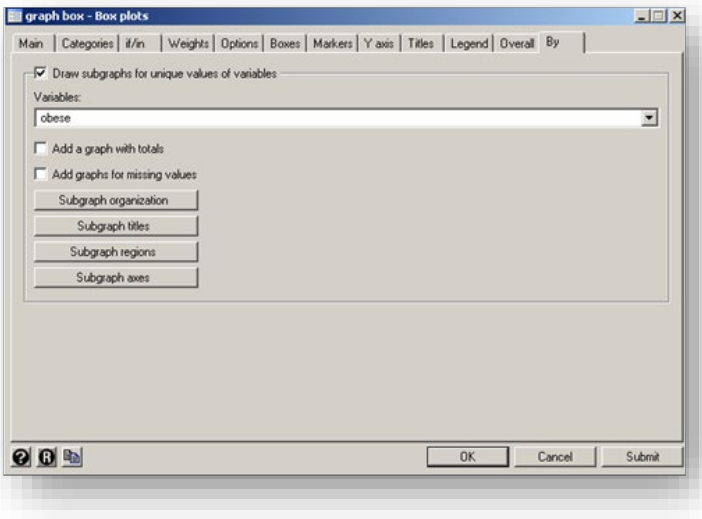


Figure 2.7: Box plots dialog box – By tab

Execute the command by clicking the *Submit* button. We see in Figure 2.8a that by using the By option Stata produces two subgraphs each with its own title and x-axis labels.

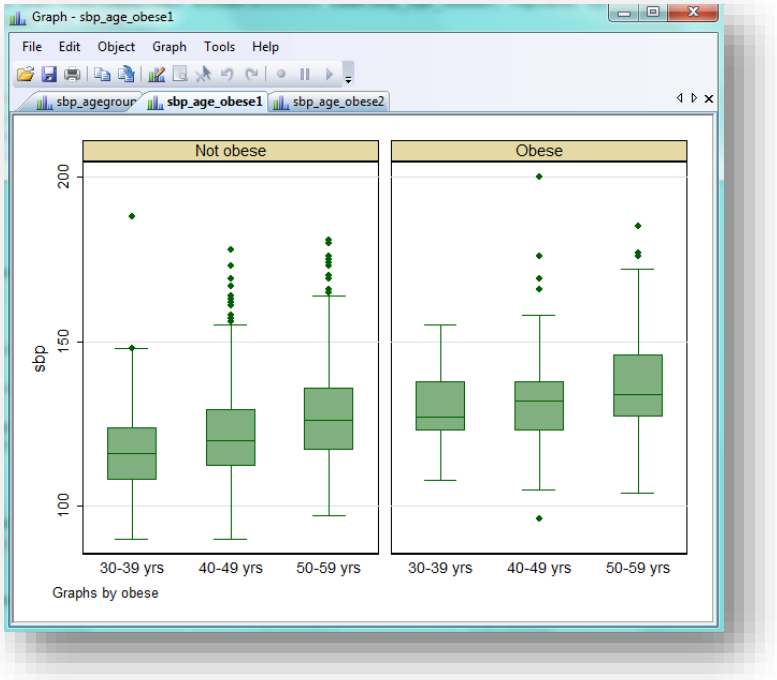


Figure 2.8a: Box plot of Systolic Blood Pressure over age-group and *by* obesity

Now under the *By* tab deselect the *draw subgraphs* tick box. Under the *Categories* tab add obese as a second grouping variable.

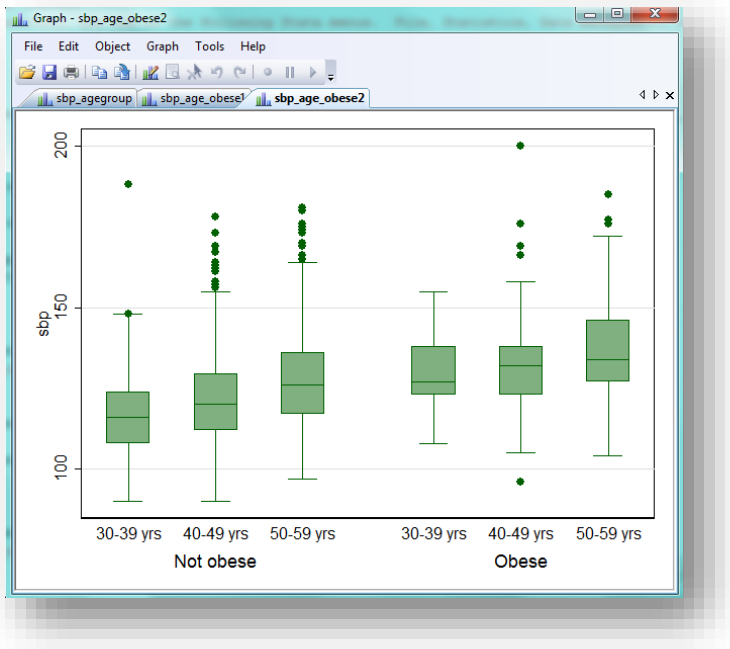


Figure 2.8b: Box plot of Systolic Blood Pressure over age-group and *over* obesity

Compare the two graphs (Figures 2.8a/2.8b) and the two commands that produce them. See what happens if you change the order of the grouping variables.

Step 5: Finally we'll make a few stylistic changes using the *Y axis* and *Markers* tabs.

(i) Add a y-axis title
(ii) Change the angle of the y-axis labels from vertical to horizontal
(iii) Specify which values are labelled on the y-axis
(iv) Change the marker symbol for the outlying values

Firstly, under the Y axis tab (Figure 2.9a), we type "Systolic Blood Pressure (mmHg)" in the *Title* box. We can change the appearance of the y-axis title using the *Properties* button to the right of the Title box.

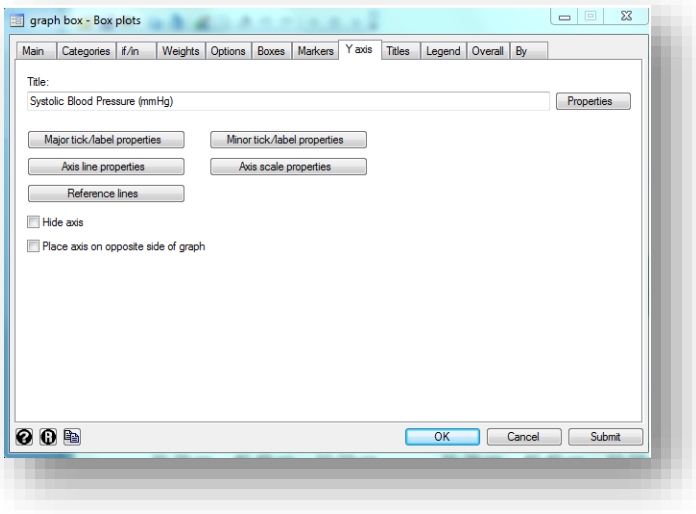


Figure 2.9a: Y Axis tab

Also under the *Y axis* tab select the *Major tick/label properties* button. This opens a new dialog box with four tabs: *Rule*, *Labels*, *Ticks* and *Grid* (Figure 2.9b).

From the *Rule* tab tick the option *Range/Delta.* This allows us to specify the minimum and maximum values for the labels and the difference (delta) between each label. Here we specify 100 (minimum), 200 (maximum) and 20 (delta). This means label values from 100 to 200 in steps of 20.

Under the *Labels* tab select *Horizontal* from the *Angle* menu. Click the *Accept* button. This closes down this dialog box without submitting the command. Note that there is now an asterisk next to the *Major tick/label properties* button indicating that some changes have been made to the default settings.

Now under the *Markers* tab, select *Marker for variable 1* (we only have one y-variable) and click the *Edit* button. This opens another dialog box with two tabs (Figure 2.10). From the *Main* tab select x from the *Symbol* menu and then click the *Accept* button. Note now that there is now an asterisk by *Marker for variable 1*. Again this indicates that you have made some changes from the default setting. Click *Submit* to draw the graph.

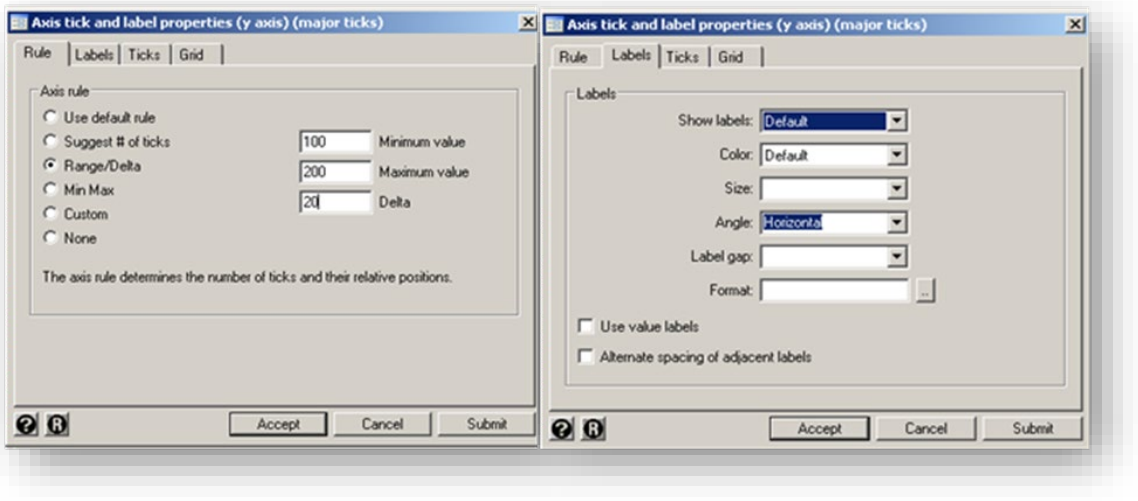


Figure 2.9b: Major Y Axis tick and label properties dialog box

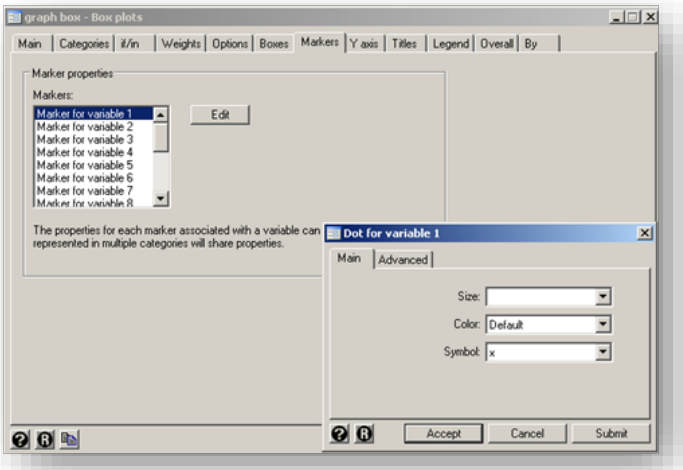


Figure 2.10 Box plots dialog box – Markers tab

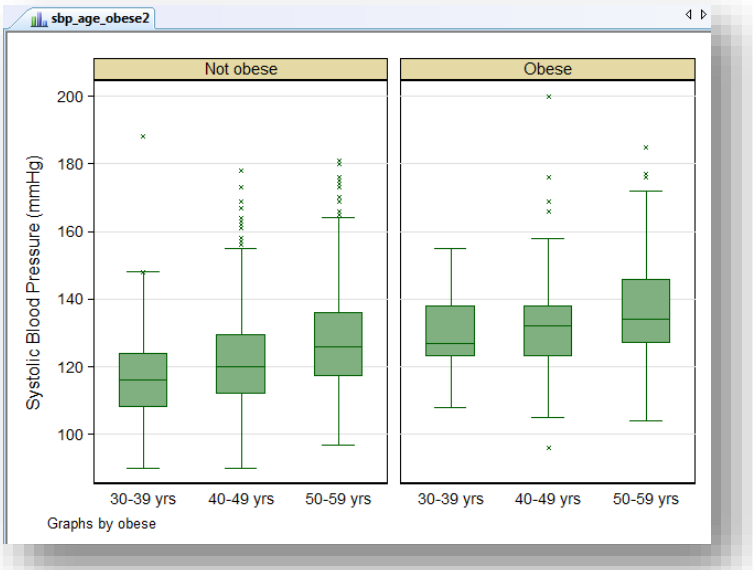


Figure 2.11: Box plot of SBP over age-group and by obesity

It is worthwhile having a look at the command syntax for this graph.

graph box	The command for producing a Box plot
sbp	The variable being plotted
,	Everything after the comma is an option
`over(agegroup)`	Specifies that the plots should be drawn over categories of agegroup
`by(obese)`	Draw sub-graphs for all unique values of variable obese
`marker(1, msymbol(smx))`	Marker properties for the first variable plotted Use a small x marker symbol
`ytitle( "Systolic Blood Pressure (mmHg)" )`	Y-title options
`ylabel( 100(20)200, angle(horizontal))`	Y-axis label properties Label 100 to 200 in steps of 20 Label angle horizontal
`name(sbp_age_obese2, replace)`	Name the graph "..."

GUI Example 2: Overlaying plots from the twoway family of graphs

Here we will create a graph plotting mean haemoglobin (g/L) over time in two treatment groups with a 95% CI for the mean at each time-point. This will involve overlaying two different plots from the twoway family.

Firstly, we will use an *rspike* plot to plot a vertical line showing the range from the lower limit to the upper limit of the 95% CI for the mean haemoglobin.

Secondly, we will use the *scatter* plot to overlay the mean haemoglobin. This will be done separately for each treatment group using an *if* expression.

The data for this are contained in *hb.dta*. This dataset has eight observations, one for each of four visits for both treatment groups.

```
. use hb, clear
. list, noobs clean
```

visit	treat	hb_gl	se	low	high	vis2
BL	A	143.5661	0.1667	143.2393	143.8928	-0.1
BL	B	143.5750	0.1624	143.2567	143.8933	0.1
Month 3	A	143.5060	0.1583	143.1957	143.8163	2.9
Month 3	B	141.6810	0.1565	141.3742	141.9877	3.1
Month 6	A	142.7615	0.1596	142.4487	143.0744	5.9
Month 6	B	141.4161	0.1652	141.0923	141.7400	6.1
Month 12	A	143.3765	0.2376	142.9109	143.8421	11.9
Month 12	B	142.3031	0.2370	141.8387	142.7676	12.1

Here *hb_gl* is the mean haemoglobin at each visit by treatment group, and *low* and *high* are the lower and upper limits of the 95% CI respectively. The variable *treat* takes values 1 and 2 and has value labels (1=A and 2=B) attached.

The variable *visit* takes values 0, 3, 6 and 12 with value labels (BL, Month 3, Month 6, Month 12 attached. A second variable *vis2* has been created to enable the means and CIs for each visit to be placed side by side rather than being overlapping. This was created as follows:

```
. gen vis2 = visit - 0.1 if treat==1
. replace vis2 = visit + 0.1 if treat==2
```

Step 1:
First we select *Graphics/Twoway graph (scatter, line, etc)* from the drop-down menu which will open the *twoway – Twoway graphs* dialogue box (Figure 2.12). This dialogue box has 8 tabs, including *Plots*, *if/in*, *Y axis*, *X axis* and *Titles*.

To create a plot we select the *Create…* button on the *Plots* tab. This opens up the *Plot 1* dialogue box as seen in Figure 2.12.

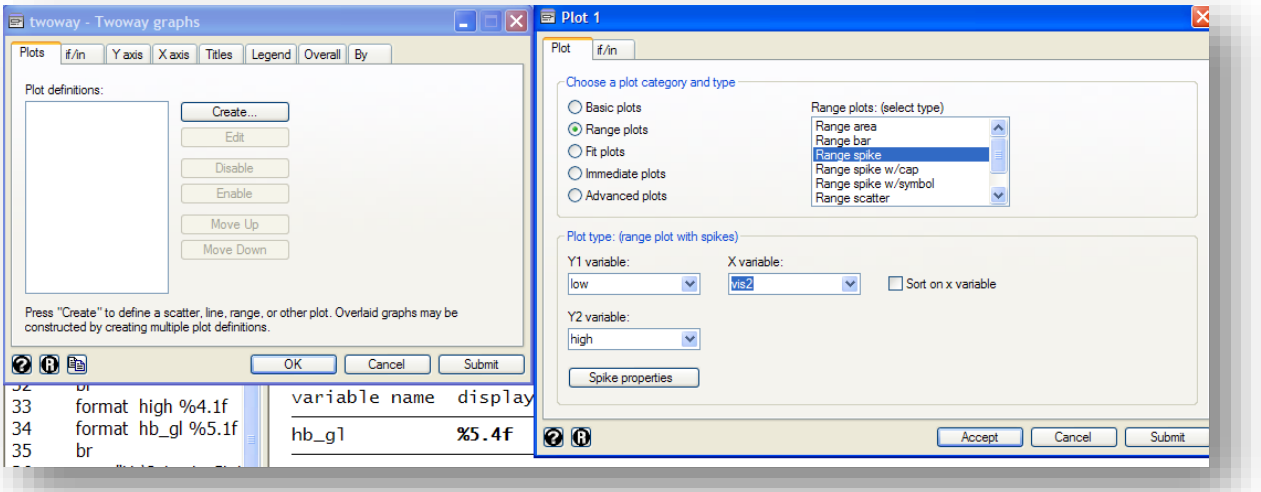


Figure 2.12: the twoway graph and plot 1 dialogue boxes

Stata overlays subsequent plots over any previously created plots, so the order of the plots is important. Once a series of plots has been created it is possible to move them up and down from within the *Plots tab* of the twoway graph dialogue box.

Here we want to plot the 95% CIs first and then overlay the means. So from the *Plot* tab of the *Plot 1* dialogue box we select Range plots. Stata offers a number of types of range plot (see Figure 2.12) – we will select the range spike plot.

This requires us to specify two Y axis variables (Y1 and Y2 the upper and lower values – order is not important) and an X variable. Here we have specified *low*, *high* and *vis2* respectively.

We want to create separate plots for each treatment group (though on the same graph) so we need to restrict this command using the *if/in* tab on the Plot 1 dialogue box (*Important to note not the overall Twoway dialogue box*). We add the expression *trt==1* (note the double equals sign) to the *If: (expression)* box.

We can specify the properties (colour, width etc.) of the spike using the *Spike properties* button. Here we will specify the colour as gray 0 (equivalent to black). Click *Accept*.

Step 2:
We will now add a second *rspike* plot for treatment group 2. Within the *Plot* tab click the *Create…* button again.

We will repeat what we did above except to restrict the command to *trt==2* and specifying the spike colour to be *Gray 10*. Select *Accept* and then *Submit*. The result is seen in Figure 2.13.

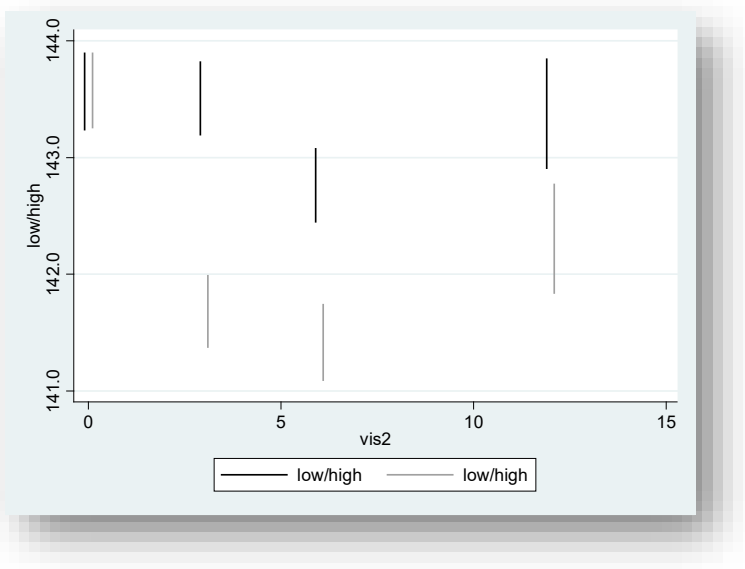


Figure 2.13: Twoway graph - range (spike) plot

The syntax for this graph command is:

```
. twoway (rspike low high vis2 if trt==1, lcolor(gs0))
      (rspike low high vis2 if trt==2, lcolor(gs10))
```

Step 3:
We will now add a third plot to show the mean for treatment group 1 at each visit. We can do this using a twoway scatter plot.

From within the *Plot* tab we again select the *Create…* button to add a third plot. Select *Basic plots* and from within the *Basic plots: (select type)* menu chose the type *Scatter*. This plot type requires one Y and one X variable. Here we select *hb_gl* as the Y variable and *vis2* as the X variable.

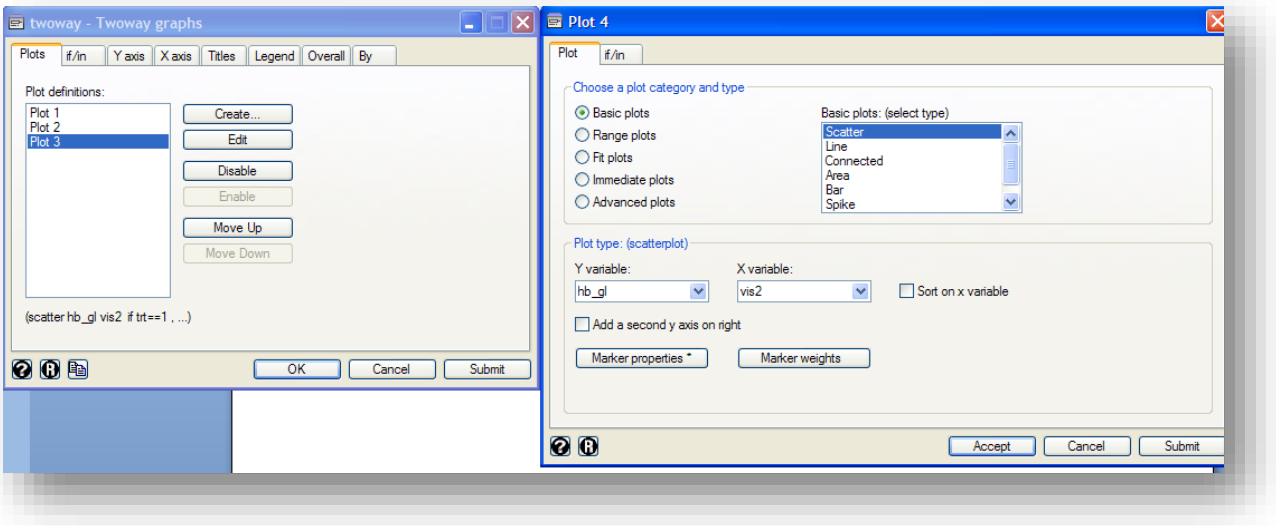


Figure 2.14: the twoway graph and plot 4 dialogue box

As with the rspike plots above, we are creating separate plots for each treatment group so we need to add *treat==1* as the *if (expression)* in the *Plot 3* dialogue box.

The marker type, colour and size can be selected using the *Marker properties* button (see Figure 2.14). We will specify the markers to be squares, medium in size for both groups. For treatment group 1 we will specify the marker colour to be *Gray 0*.

Click *Accept* on the *Plot 3* dialogue box.

Step 4:
We now add the fourth plot, a scatter plot of mean haemoglobin against visit for treatment group 2. To do this we repeat step 3 but this time add treat==2 to the if (expression) and select Gray 10 as the marker colour.

Click *Accept* on the *Plot 4* dialogue box and then click *Submit*.

This should produce a graph made up of 4 separate plots overlaid on top of one another. See Figure 2.15.

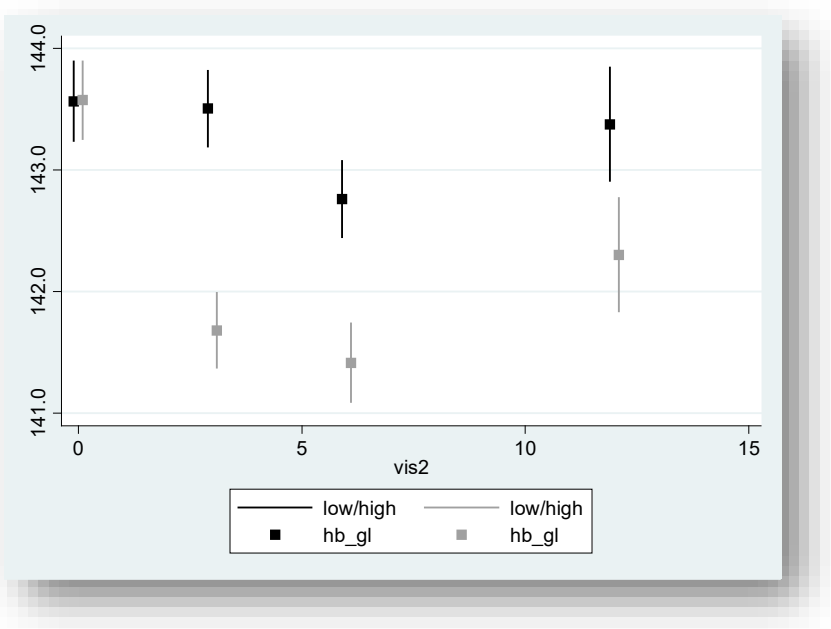


Figure 2.15: Twoway scatter plot overlaid on range (spike) plot

The Stata code for this command is:

```
. twoway (rspike low high vis2 if trt==1, lcolor(gs0))
       (rspike low high vis2 if trt==2, lcolor(gs10))
       (scatter hb_gl vis2 if trt==1, mcolor(gs0) msize(medium) msymbol(square))
       (scatter hb_gl vis2 if trt==2, mcolor(gs10) msize(medium) msymbol(square))
```

You can clearly see the 4 separate plot types and their options. Try to work out what each of the options is doing.

Step 5:
We can now add titles to the Y and X axis. This is done using the *Title* box in the *Y axis* and *X axis* tabs found on the *Twoway graphs* dialogue box (Figure 2.16).

We can also change the tick/label properties for each axis. In Figure 2.16 we have selected the *Major tick/label properties* button and specified a custom rule (0 3 6 12) equivalent to the values of the variable visit.

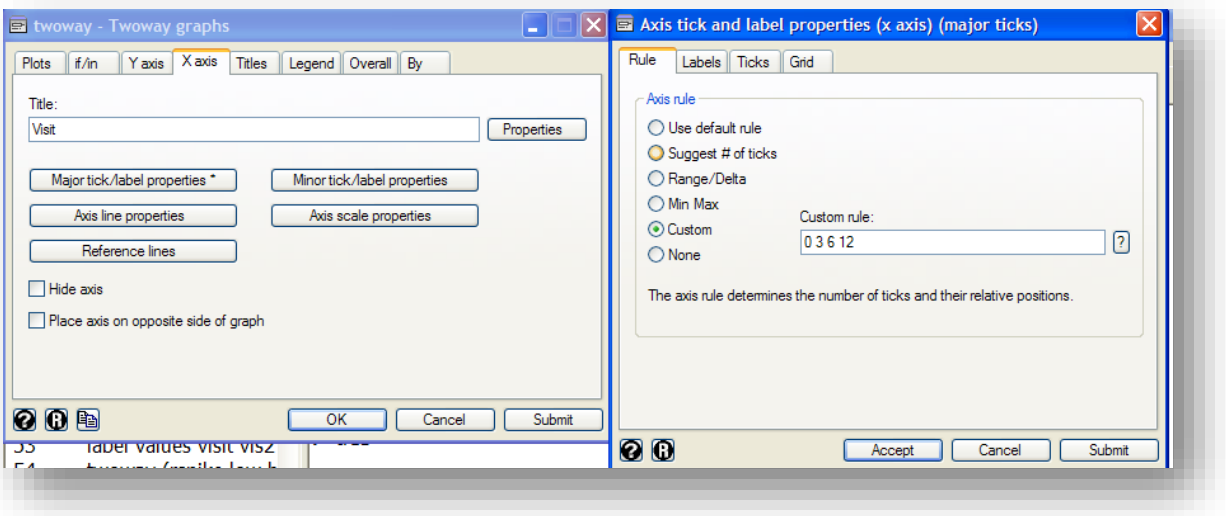


Figure 2.16: Twoway graph and Major tick dialogue boxes – Rule tab

Under the *Labels* tab (Figure 2.17) of this dialogue box we can also choose value labels to be displayed rather than the actual values by selecting *Use value labels*. The size of the labels can also be specified. We will select small for the labels on both axis. We can also specify the angle of the labels – for the Y axis I prefer the labels to be horizontal rather than Stata's default of vertical. We can also format the display of the labels – for the Y axis labels %2.0f has been chosen i.e. forcing the display to 0 decimal places.

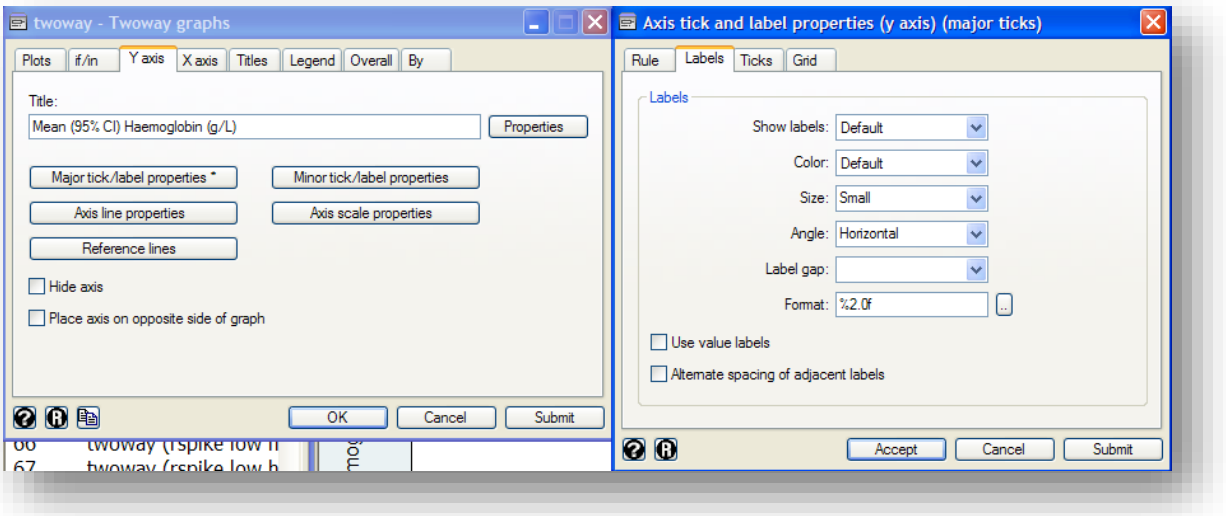


Figure 2.17: Twoway graph and Major tick dialogue boxes – Labels tab

The range of each axis can be changed (increased) using the Axis scale properties button. For this graph we will specify the range of the X axis to be -0.5 and 12.5. We will also hide the Legend box – see the Legend tab.

The resulting graph is shown in Figure 2.18.

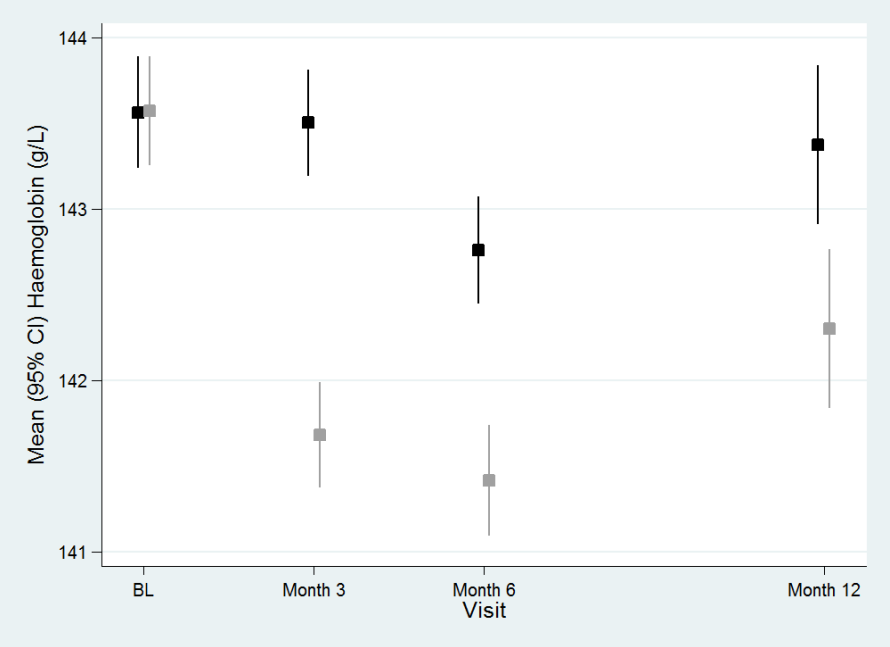


Figure 2.18: Twoway plot: scatter plot overlaid on a range plot

The Stata code or syntax for this graph is:

```
twoway (rspike low high vis2 if trt==1, lcolor(gs0))
      (rspike low high vis2 if trt==2, lcolor(gs10))
      (scatter hb_gl vis2 if trt==1, mcolor(gs0) msize(medium) msymbol(square))
      (scatter hb_gl vis2 if trt==2, mcolor(gs10) msize(medium) msymbol(square)),
      ytitle(Mean (95% CI) Haemoglobin (g/L))
      ylabel(, labsize(small) angle(horizontal) format(%2.0f))
      xtitle(Visit) xscale(range(-0.5 12.5))
      xlabel(0 3 6 12, labsize(small) valuelabel)
      legend(off)
```

As above the four subplot types with their specific options are seen within brackets on the first four lines. The overall graph options appear following the comma.

See below for a more detailed explanation of the syntax.

twoway	twoway family of plots – can be overlaid Here we overlay four plots – two range spike plots and two scatter plots
(rspike low high vis2 if trt==1, lcolor(gs0)) (rspike low high vis2 if trt==2, lcolor(gs10)) (scatter hb_gl vis2 if trt==1, mcolor(gs0) msize(medium) msymbol(square)) (scatter hb_gl vis2 if trt==2, mcolor(gs10) msize(medium) msymbol(square))	rspike = range spike plot lcolor = line colour of spike gs# = gray-scale (0=black 14=white) scatter = twoway scatter plot msymbol = marker symbol square = use a square marker msize = marker size mcolor = marker colour note that each if (expression) applies only to the plot within the brackets
,	all options following this "," apply to the overall graph
ytitle("Mean (95% CI) Haemoglobin (g/L)") xtitle("Visit")	Titles for the y-axis and x-axis Can specify further options within the brackets
ylabel(, labsize(small) angle(horizontal) format(%2.0f)) xlabel(0 3 6 12, labsize(small) valuelabel)	y-axis and x-axis labelling labsize = label size angle = angle of labels format = numeric format 0 3 6 12 = custom rule for labelling x-axis %2.0f = fixed format to 0 decimal places valuelabel = use value label rather than numbers
xscale(range(-0.5 12.5))	scaling of x-axis range = specify range of x-axis (-0.15 12.5) = min and max values of x-axis
legend(off)	do not show the legend

3. The Stata Graph Editor

Stata comes with a built in Graph Editor through which it is possible to add, remove, move and modify titles, legends, axes, lines, arrows, markers, text and more. Here we will just introduce the Editor. You really have to play with it to discover what it can do (plus see **Error! Hyperlink reference not valid.**).

Graph Editor - Example 1

We will now use the Graph Editor to make some changes to the overlaid twoway graph produced above (Figure 2.18). We will record the changes being made and then play them back on the same graph. The changes we will make are:

(i) change the plot type from a scatter to a connected line
(ii) increase the marker size
(iii) change the connecting lines from solid to dashed
(iv) increase the range of X axis
(v) add labels to indicate treatment group

Starting the Graph Editor

Firstly, we redraw the graph using syntax saved in the do-file and start the Graph Editor. The Graph Editor can be started either by clicking on the graph editor button (highlighted in Figure 3.1) or through the *File > Start Graph Editor* drop-down menu.

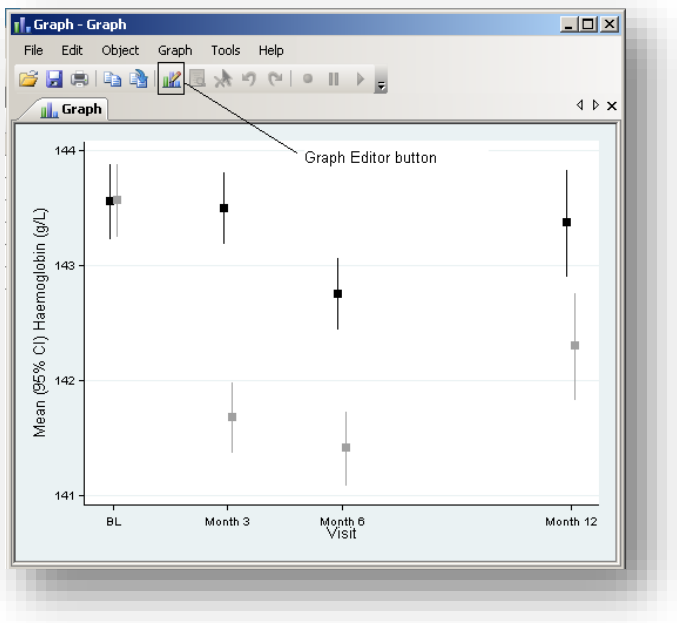


Figure 3.1: The Graph Window and Editor Button

Recording your edits

It is important to note that Stata does not save the command syntax for any changes you make using the Editor. However there is a record facility which allows you to record any edits made and then play them back on the same or different graphs.

Changes are only recorded once you have started recording. To start recording click the red *Start/Stop Recording* button on the toolbar (see Figure 3.2) – this can also be accessed through the Tools drop-down menu. Once you have completed editing the figure clicking the red button again will stop the recording and prompt you to save the recording. These recordings are saved in files with a *.grec* extension.

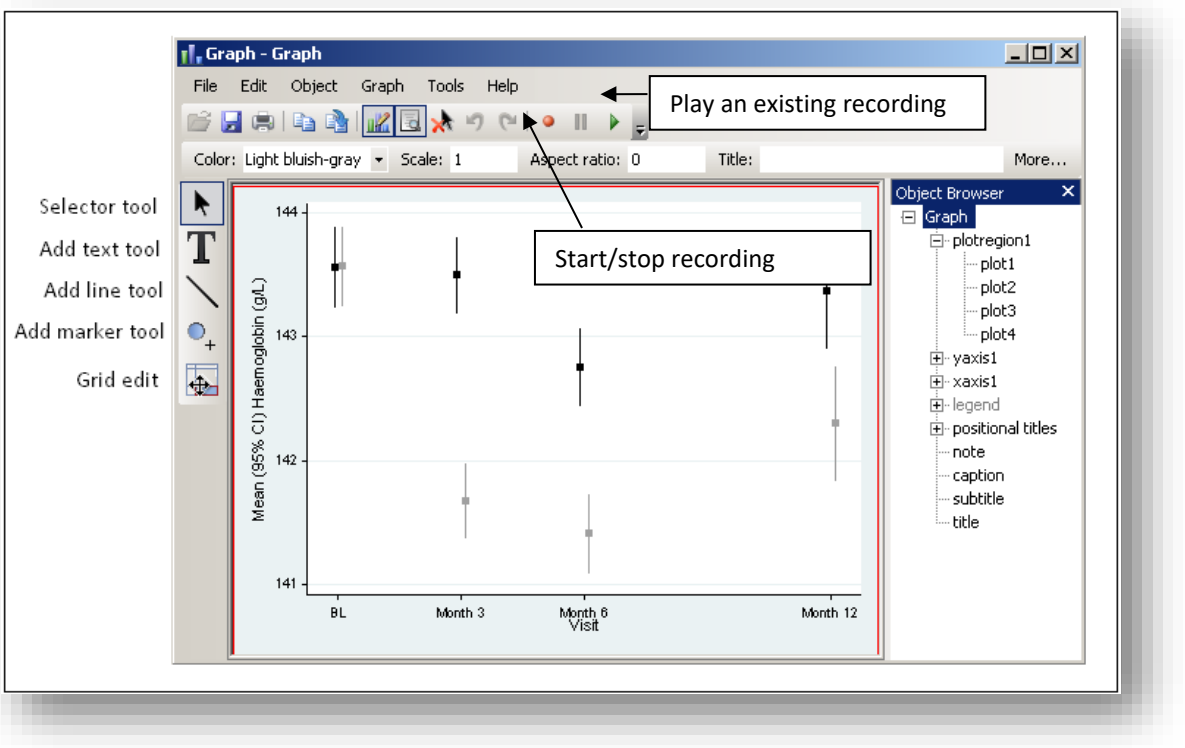


Figure 3.2: Graph Editor – Tool bars and Recording buttons

(i) Change the plot type from a scatter to a connected line plot.

Within the Graph Editor window use the Selector Tool (or Object Browser) to select plot 4. The contextual toolbar will now, among other things, show the Plot-type (Figure 3.3). We will change this from *Scatter* to *Connected*. Note that a connected plot is a scatter plot with connecting lines. This differs from a line plot which connects points with a line but does not display a marker at those points. For consistency we should also specify the colour of the connecting lines to be the same as the markers and spikes. We repeat this for plot 3.

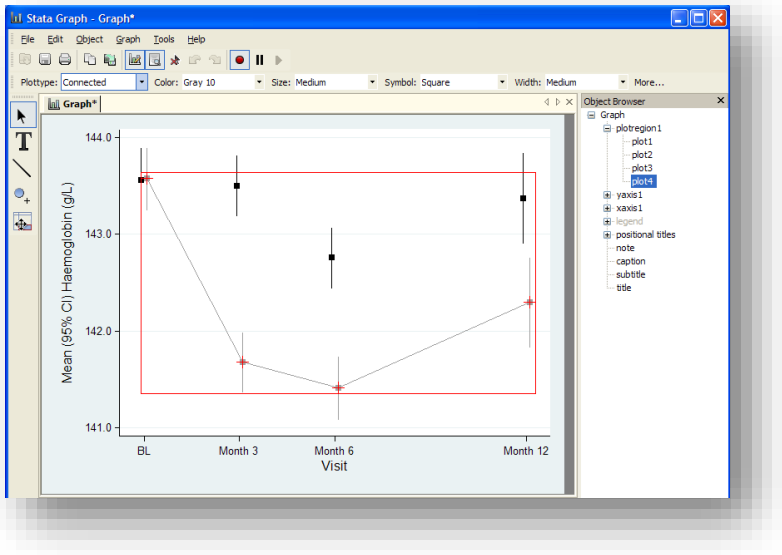


Figure 3.3: Editing the plot-type

(ii) Increase the marker size

While we were editing the plot-type we could also have changed the marker size. Using the same procedure as in (i) select plots 3 and 4 and from the contextual toolbar change the Size to Medium-large.

(iii) Change the pattern of the connecting lines from solid to dashed

By default Stata draws solid lines for the connected plot. To change these to dashed lines (or some other pattern) we again select plots 3 and 4 in turn and use the contextual toolbar. However, this time there is no obvious drop-down menu for changing the pattern of the lines. We have to select *More…* . This opens up the Connected Properties dialogue box within which we can specify the connecting line or marker properties including pattern, width, colour etc. of the line (Figure 3.4). We will select the pattern dash for each of plots 3 and 4.

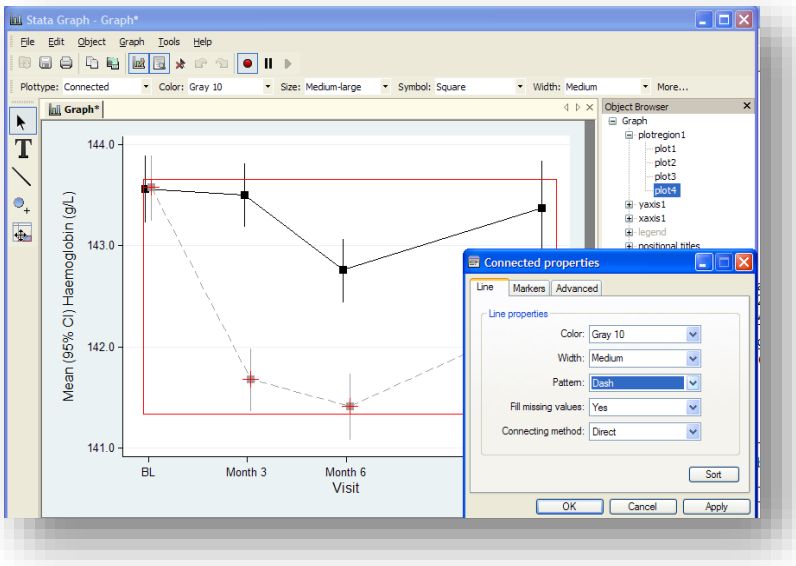


Figure 3.4: Connected properties dialogue box

(iv) Increase the range of X axis

We want to add some labels to indicate treatment group. The natural position for these labels in this graph would appear to be to the right of the month 12 markers. However, there is not enough space for the labels as the graph currently appears so we may wish to extend the range of the X axis from a maximum of around 12.5 to around 13.5.

This time we will use the Object Browser to select the X axis (xaxis1). Using the mouse-right-click, select Axis Properties from the menu that appears. This opens up the Axis properties dialogue box (Figure 3.5). Under Global properties select the Scale button – this opens the Axis scale properties box (Figure 3.5). Click *Extend range of axis scale* and change the upper limit to 13.5. Click OK and then Apply.

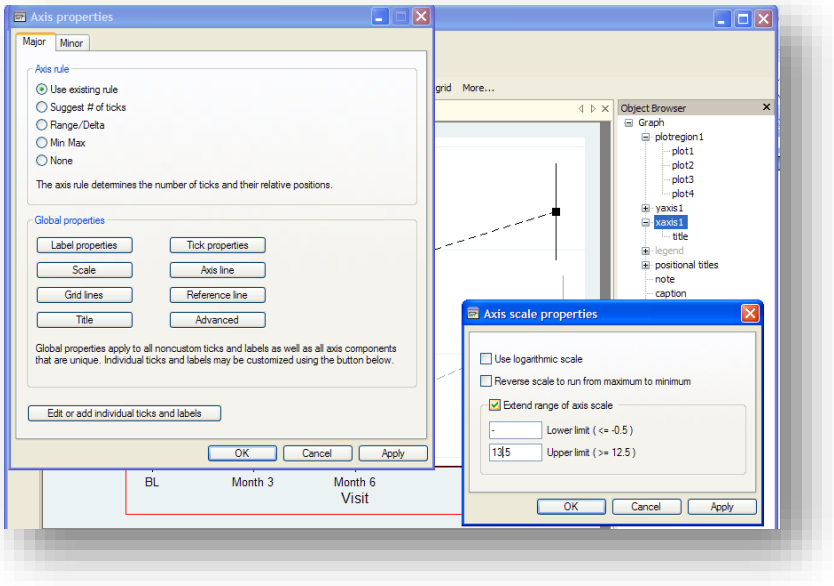


Figure 3.5: Axis properties and Axis scale properties

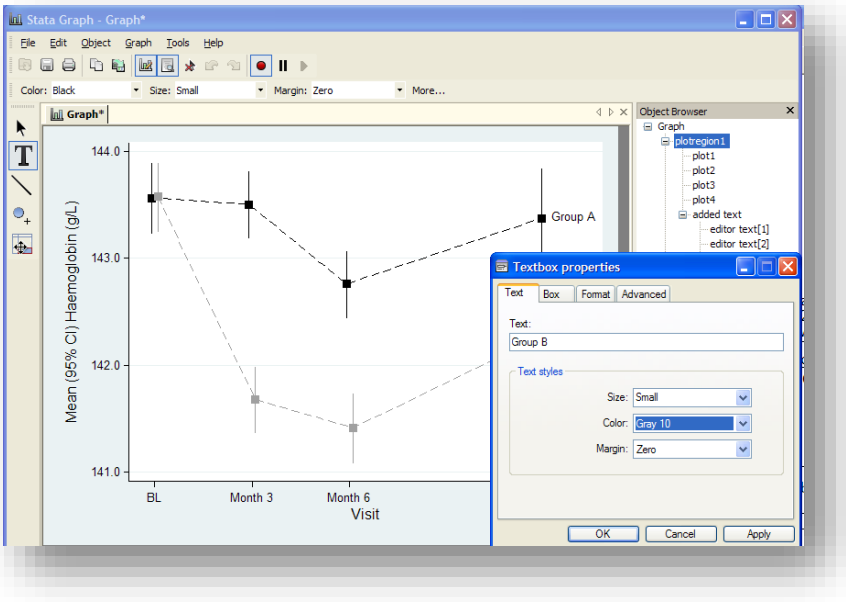


Figure 3.6: Adding text using the textbox properties dialogue box

(v) Add labels to indicate treatment group

Finally we will add some labels to indicate treatment group using the add Text tool. The size, colour etc. of the labels or text can be specified within the Textbox properties dialogue box (Figure 3.6). Once the text has been added it can be moved around the plot region using the object selected.

We will now stop the recording, save the recording file as *demo1.grec*, exit the Graph Editor and close down the graph. Now rerun our original graph command, open the Graph Editor and select play recording from the toolbar. Select demo1.grec and click open. This automatically replays the changes that we made above.

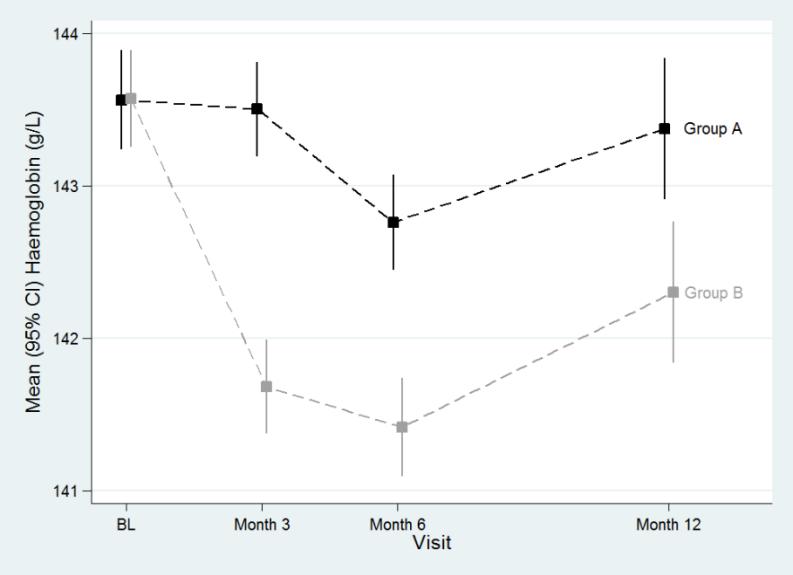


Figure 3.7: Twoway overlaid connected and range plot

Close the graph editor, redraw the graph using the command in the Review window. Open the Graph Editor and replay the graph editor recording.

4. Building a Graph in a Stata Do-file

For the rest of this session we will leave the GUI and concentrate on learning the command syntax and building up a graph command in a do-file.

4.1 Dealing with long commands in do-files

As we have already seen, once we start adding options graph commands can quickly become very long. It is then not convenient to have the whole command on a single line. However, by default Stata views the carriage return character as marking the end of a command. One way of allowing a command to extend over more than one line is to add a triple slash (///) at the end of each line (but not the final line). For example,

```
graph box sbp, over(agegroup) marker(1, msymbol(x)) ///
      ylabel(100(20)200, angle(horizontal)) ///
      by(obese)
```

This is fine for a command that extends over a few lines. If a command extends over more than 5 or so lines it is better to use the `#delimit` command to change the end of line delimiter to the semi colon character. Stata will view everything that follows this as a single command, only submitting the command when it finds the semi-colon. To revert to the carriage return character use `#delimit cr`.

Copy the code for the twoway plot created above into a new do-file. Add some appropriate comments at the head of the do-file and a command to load the data. Change the end of line delimiter to a semi-colon and space out the graph command over several lines. It can be helpful to use tabs. Place a semi-colon at the end of the command (perhaps best to do this on a separate line) and then change the end of line delimiter back to the carriage return key (cr). Save the do-file. Try executing the command from within the do-file.

```
* Stata Graph Commands
* Date created:     1 July 2022
* Author:           K Baisley

cd "H:/Intro to Stata/graphics data/"
use hb, clear

#delimit ;
twoway (rspike low high vis2 if trt==1, lcolor(gs0))
      (rspike low high vis2 if trt==2, lcolor(gs10))
      (scatter hb_gl vis2 if trt==1, mcolor(gs0) msize(medium) msymbol(square))
      (scatter hb_gl vis2 if trt==2, mcolor(gs10) msize(medium) msymbol(square)),
      ytitle(Mean (95% CI) Haemoglobin (g/L))
      ylabel( , labsize(small) angle(horizontal) format(%2.0f))
      xtitle(Visit)
      xscale(range(-0.5 12.5))
      xlabel(0 3 6 12, labsize(small) valuelabel)
      legend(off)
;
```

Repeat this for the box plot and Kaplan-Meier plots created above.

4.2 Working efficiently with macros

Quite often you may find yourself producing a series of very similar graphs with many repeated options. Changing your mind about the look of the graph or some of the options may then involve making changes to all the graph commands, which can be time consuming. Using macros can be a very efficient way of dealing with such a situation.

Stata has *local* and *global* macros – here we will focus on global macros.

Global macros are defined using the `global` command and are de-referenced using a dollar sign. A macro consists of the *macro name* and the *macro contents* both of which are specified when the macro is defined. For example,

```
. global X hello
```

would create a global macro called X containing the string "hello". We put a dollar sign in front of the global macro to de-reference or use this macro. For example,

```
. display "$X"
 hello
```

So in the example above we could create global macros at the top of our do-file and then de-reference them within the graph command.

```
. global mopts msize(medium) mysmbol(square)
. global yopts labsize(small) angle(horizontal) format(%2.0f)

#delimit ;
twoway (rspike low high vis2 if trt==1, lcolor(gs0))
      (rspike low high vis2 if trt==2, lcolor(gs10))
      (scatter hb_gl vis2 if trt==1, mcolor(gs0) $mopts)
      (scatter hb_gl vis2 if trt==2, mcolor(gs10) $mopts),
      ytitle(Mean (95% CI) Haemoglobin (g/L))
      ylabel(, $yopts)
      xtitle(Visit)
      xscale(range(-0.5 12.5))
      xlabel(0 3 6 12, labsize(small) valuelabel)
      legend(off)
name(twoway1, replace)
;
```

Macros can also be useful for specifying directories for various folders.

5. Graph Options

5.1 Titles, Labels and Scale

If we do not specify any axis title, label or scale options then Stata will use its default rules. For example, when graphing a variable with a variable label attached Stata will use the variable label as the axis title; if there is no variable label then the variable name is used. Stata will also decide on the optimal number of axis labels and the scale of the x- and y-axis. However, we can manipulate all these aspects of the graph. We first describe some of the main options and then demonstrate with some example graphs.

Overall Titles

An overall graph title and subtitle can be added using the `title()` and `subtitle()` options.

```
. title("Histogram of SBP by BMI categories")
. subtitle("WHS study 2001-2004")
```

There are also options for adding captions and notes, i.e. `caption()` and `note()` which adds information below the plot area. E.g.

```
. caption("WHS study 2001-04")
```

Each of these title options have sub-options for controlling the size, colour, justification, region *etc* of the titles. E.g.

```
. title("Histogram of SBP by BMI categories", size(medium) color(red))
```

See *help title_options* for more details.

Axis Titles

Titles can be added to the y-axis and x-axis using `ytitle()` and `xtitle()`. E.g.

```
. ytitle("Percentage (%)")
. xtitle("Time of Visit (months)")
```

The axis title options have a number of sub-options for specifying the size, position, colour etc. of the titles. E.g.

```
. ytitle("Percentage (%)", col(blue) size(medsmall))
. xtitle("Time of Visit (months)" margin(t+1))
```

The `margin` option is used to expand the margins of the (currently invisible) text box surrounding the title. Here we expand the top of the margin by 1 point (t+1) which in effect moves the position of the title slightly down from the x-axis labels.

You can also specify titles at the top, left, right and bottom of the plot are using `t1title()`, `t2title()`, `l1title()`, `l2title()`, `r1title()`, `r2title()`, `b1title()` and

`b2(title)` respectively. These options can be useful when no x-axis exists, e.g. with the vertical box plots created in Chapter 2 there is no x-axis title option and then the `b1` and `b2` options are helpful.

See *help axis_title_options* for more details.

Axis Labels

The number, location and appearance of the axis labels and ticks are specified using `ylabel()` and `xlabel()`. E.g.

```
. ylab(0(0.2)1)           // meaning label the y-axis from 0 to 1 in steps of 0.2
. xlab(1 3 6 12)          // meaning label the x-axis at 1, 3, 6 and 12
. xlab(0 "0%" 20 "20%")   // meaning label the x-axis at 0 as 0%, 20 as 20%
```

These axis options have sub-options which can be used to control the size, colour, location, orientation etc of the labels and ticks. E.g.

```
. ylab(0(0.2)1, format(%2.1f))  // display numbers to 1 dp e.g 0.0, 0.2 et
. ylab(0(2)10, angle(hori))     // display labels horizontally
. xlab (1 3 6 12, value)        // meaning use value labels rather than numbers
```

Other useful axis label options include grid, nogrid, gridmin, gridmax and similar which can be used to modify the grid lines.

See *help axis_label_options* for more details.

Axis Scale

The range, look and scale (e.g. log, reversed) of the y-axis and x-axis can be specified using `yscale()` and `xscale()`. E.g.

```
. yscale(log)             //  draw y-axis on the natural logarithm scale
. xscale(range(-10 40))   // extend the scale of x-axis from -10 to 40.
. xscale(alt)             // move y-axis to right or x-axis to top
. yscale(reverse)         //  reverse scale to run from max to min
```

See *help axis_scale_options* for more details

Example graph for options

The Stata code for the graph below (Figure 5.1) can be found in *graph_options.do*.

We will focus firstly on that part of the Stata code that relates to the title and axis options.

Axis Titles and Labels: see graph_option.do and Figure 5.1.

,	all options following this "," apply to whole graph
xlab(1 2 3 , valuelabel labsize(*1.1) labcol(gs4)) ylab(0 "0%" 20 "20%" 40 "40%" 60 "60%" , angle(hori))	specifying the x-label xlab() and y-label ylab() options 1 2 3 = labels at 1 2 and 3 on the x-axis valuelabel = use the value label not numbers labsize = increase default label size by 10% labcol(gs4) = label colour is gray-scale 4 0 "0%" etc = label 0 as "0%" angle(hori) = y-axis labels horizontal
xscale(range(0.5 3.5)) yscale(range(0 60))	xscale( ) = modify scale of x-axis using options range(#1 #2) = min and max values for x and y axis (note will be ignored if data lie beyond these values)
xtitle("Risk Group" , margin(t+2) size(*1.2)) ytitle("2-year Incidence of Primary Endpoint" , margin(r+1))	x and y-titles margin(t+2) = axis title is placed in a box (not visible); to move x-axis title down expand top margin (t) by 2 points; to move y-axis title to left expand right margin (r) by 1 point. size(*1.2) = increase size title by 20% (times 1.2)
title("Validation of EMPHASIS-HF Risk Score", size(*0.8))	overall title for graph size(*0.8) = reduce default size of title by 20%

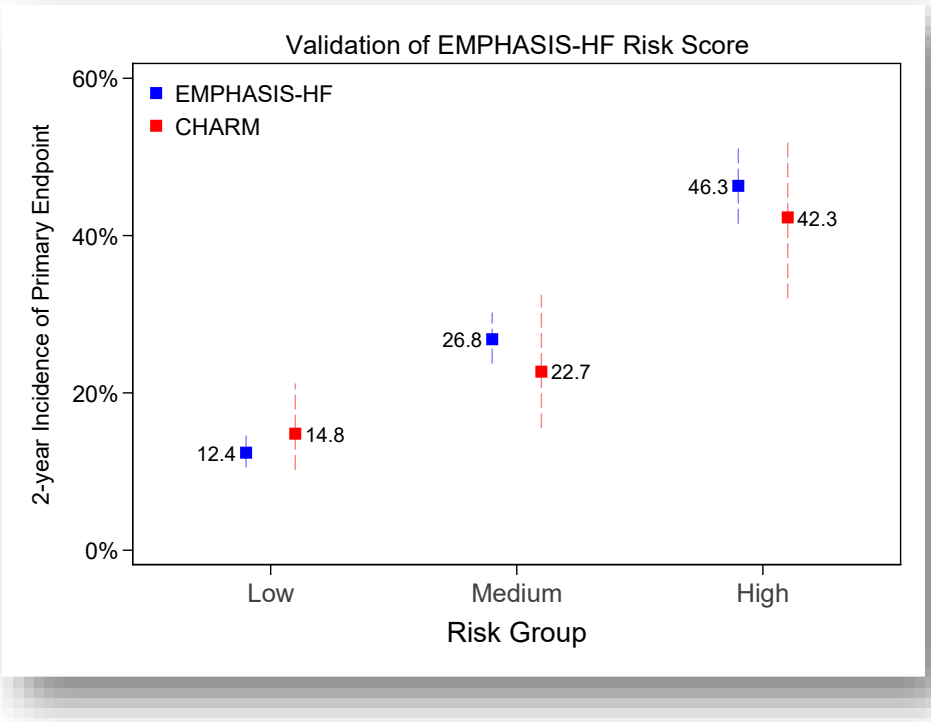


Figure 5.1: Twoway range spike and scatter plot with titles and labels

It is also possible to add a second y-axis, reverse the scale of the axes, plot on a log scale and much more. Some of these features are demonstrated in the last section in the example graphs. See *help axis_options* for more information.

5.2 Markers and Lines

It is possible to specify the type, size, colour, pattern etc of markers and lines. We can also add labels to markers and manipulate their size, positioning and more. The key options are described briefly below and are then demonstrated using an example graph.

Aspect	Example	Description
Marker Symbol	msym(S)	a large solid square
	msym(d)	a small solid diamond
	msym(oh)	small hollow circle
	see help symbolstyle for more options	
Marker Size	msize(large)	large symbol
	msize(small)	small symbol
	msize(*1.1)	relative size = 10% bigger than default
	see help markersizestyle for more options	
Marker Colour (overall)	mcol(blue)	blue marker colour
	mcol(gs1)	gray-scale 1 = black
	mcol(red*0.8)	default red with 20% reduction in intensity
	mcol(red%50)	default red with 50% reduction in opacity
	mcol("20 200 20")	use RGB (can also use CMYK)
	see help colorstyle for more options	
Marker Label	mlab(*varname*)	use *varname* as a marker label
	mlabpos(3)	clock position of label relative to marker see *help clockposstyle*
	mlabsize(small)	small label size, can also use relative size
	see help marker_label_options for other options	
Line	lcol(green)	line colour green
	lpat(dash)	line pattern dashed, *see help linepatternstyle*
	lwidth(vthin)	line width thin, *see help linewidthstyle*

We can also separately specify the colour of the marker fill and outline and the width of the marker outline. See *help marker_options* for more details.

Here we explain the parts of the graph command relating to markers and lines in Figure 5.1.

Markers and Lines: see graph_options.do and Figure 5.1

twoway	twoway family of plots
(rspike lcl ucl riskcat if study==1, lcol(blue*0.5) lwidth(thin) lpat(dash))	range spike plot requires 2 y variables (upper and lower limit) and x variable if expression: restrict this plot to study = 1 lcol = line colour of spikes blue with 50% reduction in intensity lwidth = width of spike thin lpat = use dash line pattern for spike NB: these options only apply to this specific rspike plot
(scatter hr riskcat if study==1, msym(S) mcol(blue) mlab(hr) mlabpos(9) mlabsize(*1.1))	scatter plot requires y variable and x variable if expression: restrict this plot to study = 1 msym(S) = use large (upper case) solid square marker symbol mcol(blue) = marker colour blue mlab(hr) = use variable hr as a marker label mlabpos(9) = position of label relative to marker at 9 o'clock mlabsize(*1.1) = increase default label size by 10%

5.3 Legends

The default Stata legend appears below the x-axis title. Stata decides on an appropriate configuration depending on the number of labels. We often may wish to hide the legend, or change its location and configuration e.g. titles, number of rows or columns, length of the symbol etc.

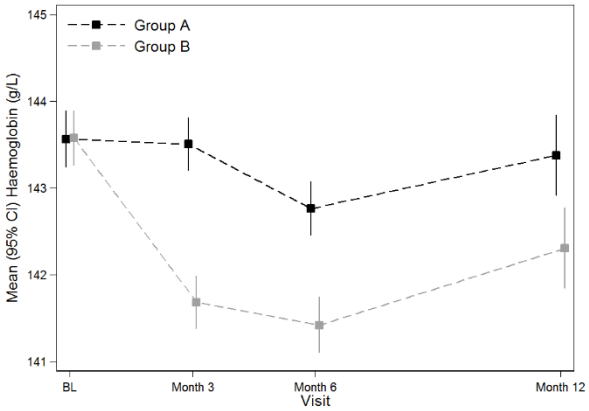
There are many options and sub-options relating to legends. Here we will illustrate a few of the most useful options with our example graph (Figure 5.1).

To simply hide a legend we use the option legend(off).

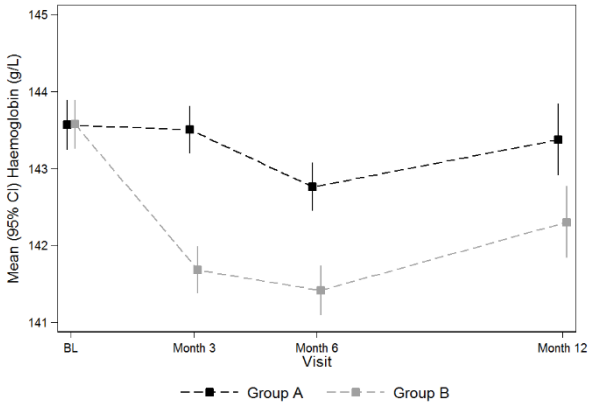
Legends: see graph_options.do and Figure 5.1

legend(	options relating to the graph legend
ring(0) pos(10) col(1) order(3 "EMPHASIS-HF" 4 "CHARM")	ring(0) = position of legend in graph. 0 means inner plot region; 1 would be for outer graph region pos(10) = position of graph within plot/graph region. 10 means at 10 o'clock. col(1) = place symbols in 1 column; can also specify row(#) order(# "title" ...) = where # relates to the plot number; here plot 3 is the scatter plot where study=1
region(lcol(none) fcol(none))	region() = the legend appears in a box with its own region properties; by default this has a colour outline and fill. lcol(none) = draw the outline of the legend box with no colour i.e. invisible fcol(none) = no fill colour for the legend box
)	bracket ending legend options

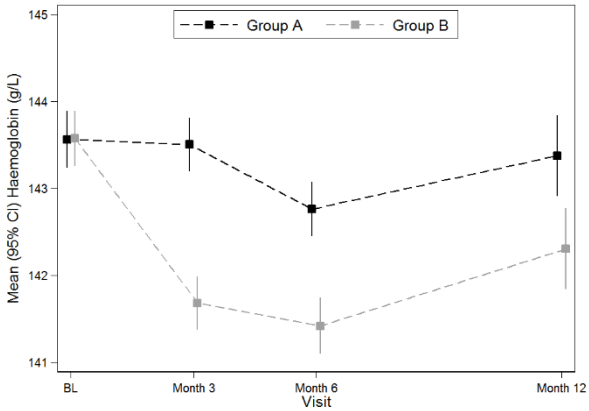
More examples of how to change the legend position and layout are shown on the next page.



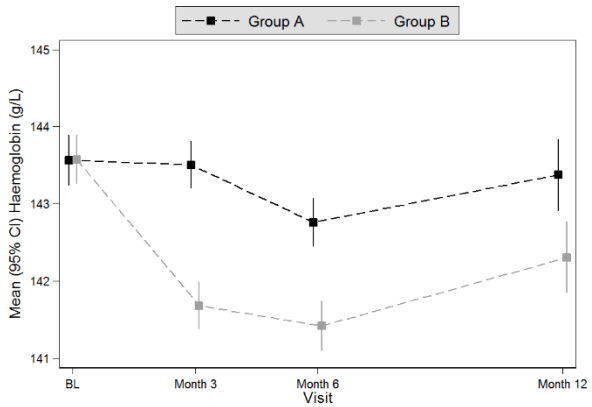

```
legend(
ring(0)                 inside plot area
pos(10)                 position 10 o'clock
col(1)                  layout at single column
order(3 "Group A"       3rd and 4th plots
4 "Group B" )
region(                 legend region options
lcol(none)              no border line
fcol(none))             no fill colour
)
```




```
legend(
ring(1)                 outside plot area
pos(6)                  position 6 o'clock
row(1)                  layout at single row
order(3 "Group A"       3rd and 4th plots
4 "Group B" )
region(                 legend region options
lcol(none)              no line (border)
fcol(none))             no fill colour
)
```




```
legend(
ring(0)                 inside plot area
pos(12)                 position 12 o'clock
col(2)                  layout at two columns
order(3 "Group A"       3rd and 4th plots
4 "Group B" )
region(                 legend region options
lcol(gs2)               border line colour
fcol(white))            fill colour
)
```




```
legend(
ring(1)                 outside plot area
pos(12)                 position 12 o'clock
row(1)                  layout at single row
order(3 "Group A"       3rd and 4th plots
4 "Group B" )
region(                 legend region options
lcol(gs1)               border line colour
fcol(none))             fill colour
)
```

Which layout to use depends on the graph itself (i.e. where is there space) and on the number of symbols to be plotted. See `help graph legend` for more details.

5.4 Adding text to Graphs

We may want to add text to a graph in preference to having a legend or to add, for example, an estimate and confidence interval or p-value from a statistical test.

Text can be added in the Graph Editor but can also be done with Stata code using the `text(#1 #2 "text")` option where #1 and #2 are the y and x axis coordinates respectively.

We'll demonstrate how to do this amending the `twoway` plot of haemoglobin created above. We want to add the text "Group A" and "Group B" to the right of the 12 month markers. Mean haemoglobin at month 12 is 143.4 g/L in group A and 142.3 g/L in group B. We will need to extend the right hand end of the x-axis slightly. We will also specify the colour, size and placement/alignment of the text.

```
#delimit ;
twoway (rspike low high vis2 if trt==1, lcolor(gs0))
      (rspike low high vis2 if trt==2, lcolor(gs10))
      (connected hb_gl vis2 if trt==1, color(gs0) msize(medium) msymbol(square)
             lpat(dash))
      (connected hb_gl vis2 if trt==2, color(gs10) msize(medium) msymbol(square)
             lpat(dash)),
      ytitle(Mean (95% CI) Haemoglobin (g/L))
      ylabel(, labsize(small) angle(horizontal) format(%2.0f))
      xtitle(Visit) xscale(range(-0.5 14))
      xlabel(0 3 6 12, labsize(small) valuelabel)
      legend(off)
      text(143.38 12.3 "Group A", size(small) col(gs0) place(e))
      text(142.3 12.3 "Group B", size(small) col(gs10) place(e))
;
```

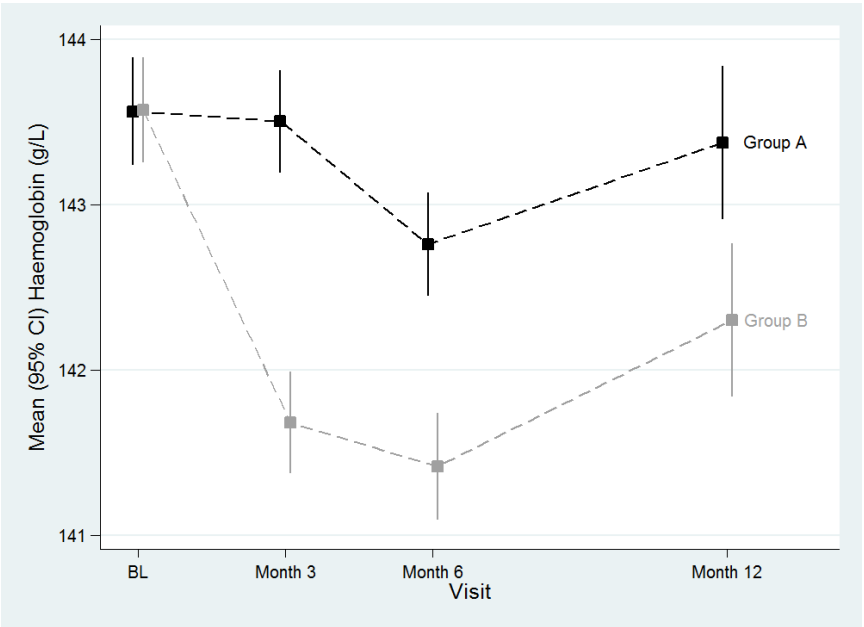


Figure 5.2: Adding text to the plot region

5.5 Plotting on more than one axis

We may wish to plot on a single graph two outcomes/variables that have quite different ranges of values. In that case plotting on one axis will probably not work. A nice way around this problem is to plot each outcome on separate y-axes i.e. one of the left-hand and the other on the right-hand, each of which can use different scales. We demonstrate this here by plotting a histogram showing the distribution of a risk score in a group of patients (on the standard left-hand y-axis) and then a line graph showing the predicted risk associated with the risk score (on the right-hand y-axis). The code for this can be found in graph_twoaxes.do.

```
twoway (hist rscore, yaxis(1) percent discrete bfcol(gs14) blcol(gs5))
      (line predfail2 rscore, yaxis(2) c(l) sort lcol(gs2)),
      legend(off)
      xtitle("Risk Score", m(t+2))
      ytitle("Percentage of patients", axis(1))
      ytitle("Predicted 2-year risk of CV death or HF hosp", axis(2))
      xlab(0(1)12)
      ytick(0(5)25)
      ylab(0"0%"5"5%"10"10%"15"15%"20"20%"25"25%", gmax angle(hori) axis(1))
      ylab(0(0.2)1, angle(hori) format(%2.1f) axis(2))
      name(riskscore, replace)
;
```

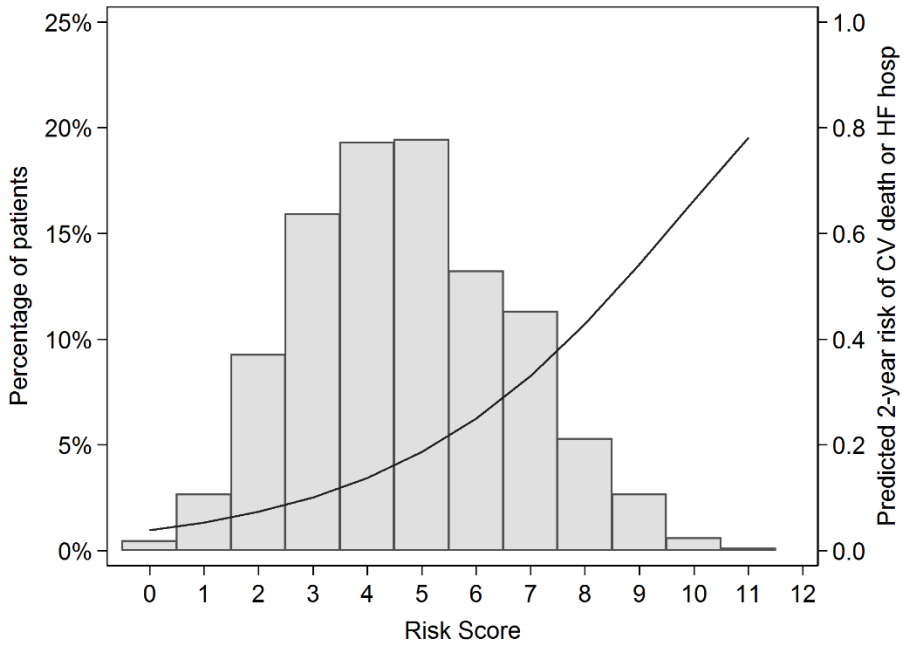


Figure 5.3: Plotting on two axes

The key things to note here are the use of `yaxis(1)` and `yaxis(2)` within the `hist` and `line` graph specific options. Then within the overall options there are two ytitle() and two ylab() specified. Which relates to which axis is specified within each of those options by use of axis(1) or axis(2). E.g. `ytitle("Percentage of patients", axis(1))`

5.6 Adding special symbols, subscripts and superscripts

It may be desired to make some titles bold, add a symbol or a superscript to a title or other added text. The Stata code and graph below demonstrate some of the options available. The code can be found in *graph_symbols.do*.

```
scatter sbp bmi in 1/100, ms(oh) msize(*0.5)
title(`"You can add {it:italics} and use {fontface "Garamond":different fonts}"')
      ytitle("{bf:Systolic Blood Pressure} (mmHg)", m(r+1))
      xtitle("{bf:Body Mass Index} (kg/m{sup:2})" , m(t+2 b+2) )
      text(165 16 "You can add Greek symbols like {&alpha}, {&Pi} or {&Sigma}",
             place(e))
      text(150 16 "or Math symbols such as {&plusmn} , {&ge} or {&infin}", place(e))
      text(135 30 "or functions like {&function}(x)=2e{sup:-2x}", place(e))
```

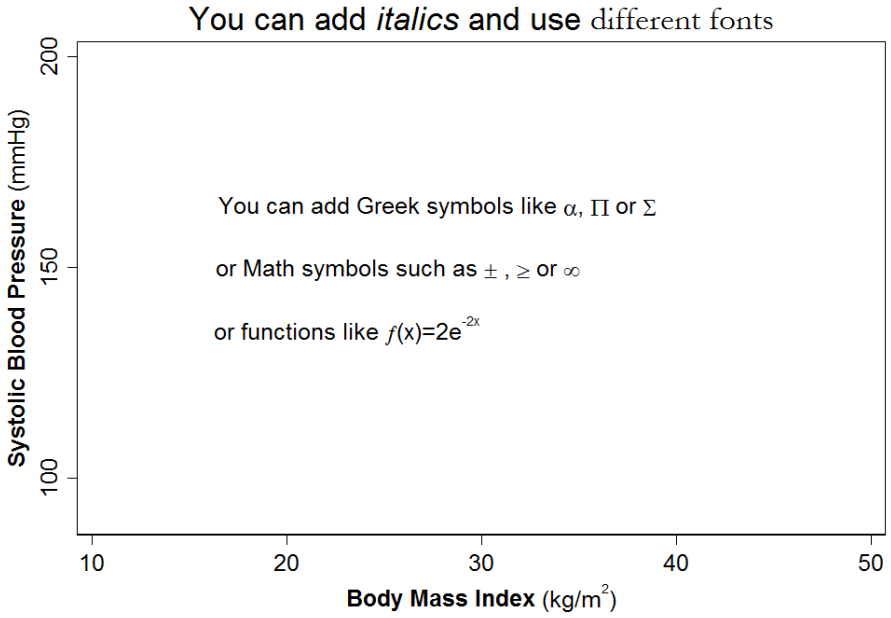


Figure 5.4: Adding special characters and formatting text

{bf:Body Mass Index}	requests bold font for the text “Body Mass Index”
(kg/m{sup:2})	requests a superscript 2 following the “m”
{it:Hello}	requests to italicize “Hello”
{&alpha}	special characters such as alpha, beta, pi

6. Graph Schemes

A graph scheme specifies the overall look of the graph e.g. region colours, default axis positions, marker symbols and colours and everything else about the graph.

There are a number of built-in schemes. They can be viewed and selected through the *Edit > Preferences > Graph Preferences* menu (see Figure 6.1).

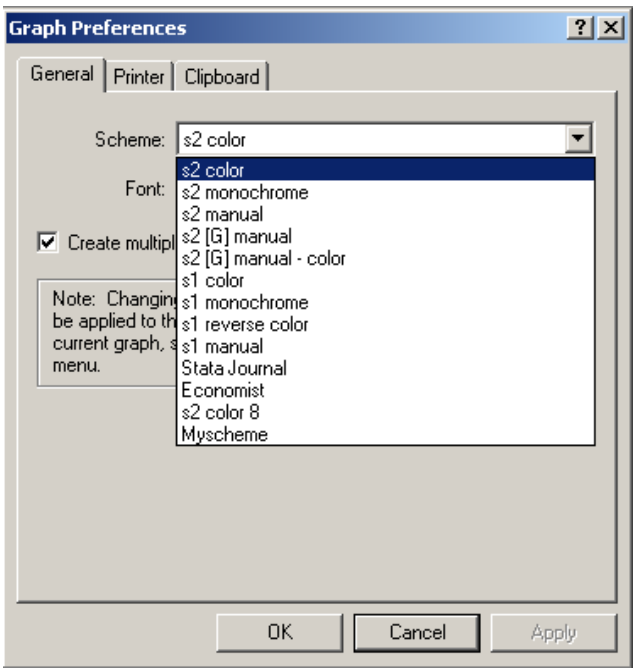


Figure 6.1: Graph Schemes

Available graph schemes can also be viewed by typing `graph query, schemes`

```
. graph query, scheme

Available schemes are

    s2color         see help scheme_s2color
    s2mono          see help scheme_s2mono
    s2manual        see help scheme_s2manual
    s2gmanual
    s2gcolor
    s1color         see help scheme_s1color
    s1mono          see help scheme_s1mono
    s1rcolor        see help scheme_s1rcolor
    s1manual        see help scheme_s1manual
    sj              see help scheme_sj
    economist       see help scheme_economist
    s2color8
    myscheme
```

A scheme can either be selected using the Edit drop down menu (Figure 6.1) or can be specified along with the graph code using the `scheme()` option. For example:

```
twoway histogram sbp, name(sbphist) scheme(s2color) title(Scheme S2 color)
twoway histogram dbp, name(dbphist) scheme(s1color) title(Scheme S1 color)
twoway histogram bmi, name(bmihist) scheme(s1mono) title(Scheme S1 mononchrome)
twoway histogram wc, name(wchist) scheme(sj) title(Scheme Stata Journal)

graph combine sbphist dbphist bmihist wchist, scheme(sj)
```

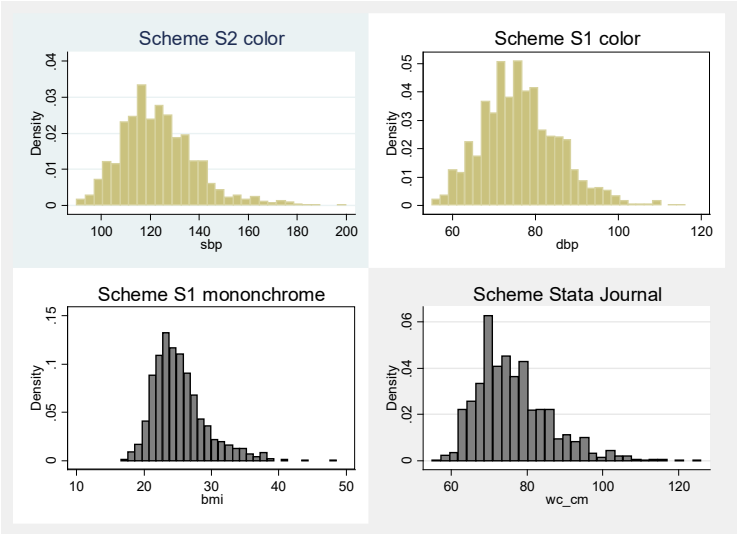


Figure 6.2: Different Stata graph schemes

The text font used in each graph can also be selected using the Graph Preferences dialog box via the *Edit > Preferences > Graph Preferences* menu as shown in Figure 6.3.

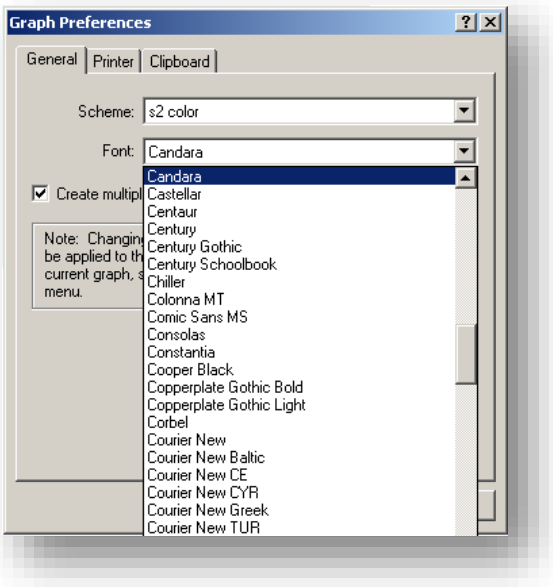


Figure 6.3: Selecting the graph font

7. Saving and Exporting Graphs

7.1 Displaying Graphs in Multiple Windows

Each graph command we have submitted so far has produced a new graph in the Graph Window. The new graph replaces any previously remembered graph. It is possible to get Stata to draw multiple graphs in separate tabs in a single Graph Window by naming graphs.

First we need to tick the "Create multiple graphs as tabs in a single window" option under *Edit > Preferences > Graph Preferences* (Figure 7.1).

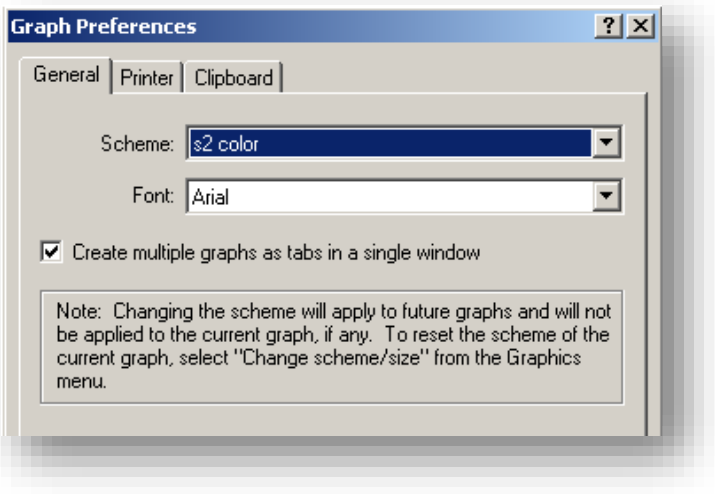


Figure 7.1: Graph Preferences Dialog Box

We then need to name each graph we produce using the `name("graphname")` so that they each appear in a separate tab. These graphs are also saved in memory, and can be retrieved at any time during a session. This is not the same as saving a graph to disk, which we will cover shortly. Once a Stata session is ended any graphs saved in memory are lost.

Once a graph has been saved to memory using the name option it can only be overwritten by using the replace option, e.g. `name(graphname, replace)`. It's probably a good idea to always add the option replace as you'll often want to redraw a graph more than once.

We use the whs.dta file and three box-and-whisker plots to demonstrate this.

```
. use whs, clear
. graph box sbp, over(bmi4) name(sbp_box, replace)
. graph box dbp, over(bmi4) name(dbp_box, replace)
. graph box crp, over(bmi4) yscale(log) name(crp_box, replace)
```

The Graph Window with multiple named tabs can be seen in Figure 7.2 below.

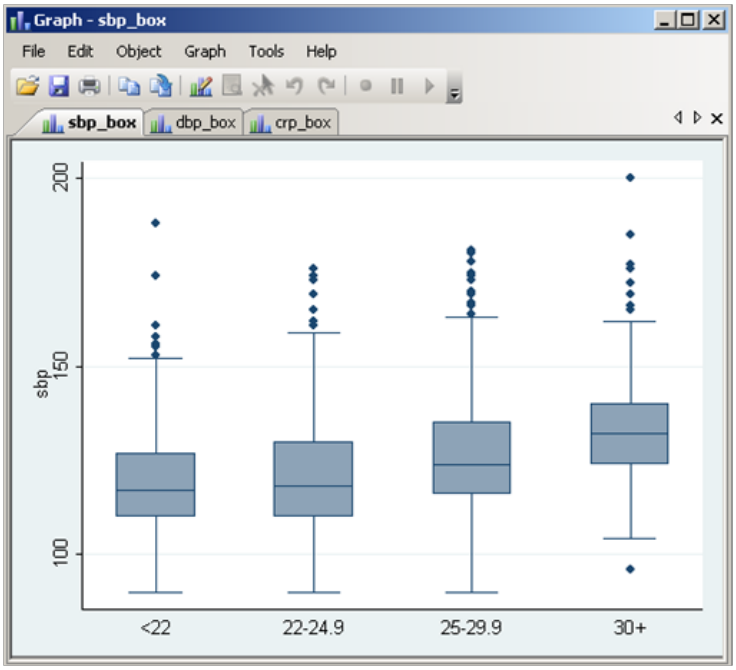


Figure 7.2: Graph Window with multiple named tabs

A named graph can be redrawn from memory using the `graph display` command. E.g.

```
. graph display crp_box
```

The `graph dir` command will show a list of any graphs saved in memory.

```
. graph dir
    Graph    crp_box  dbp_box  sbp_box
```

The details of any graph saved in memory can be seen using graph describe.

```
. graph describe crp_box

crp_box stored in memory

        name:  crp_box
      format:  live
     created:  20 Sep 2019 16:40:31
      scheme:  s2color
        size:  4 x 5.5
    dta file:  whs.dta dated 11 Sep 2019 18:00
     command:  box crp, over(bmi4) yscale(log) name(crp_box, replace)
```

7.2 Copy-and-Paste

It is possible to copy-and-paste a Stata graph. For example, to copy-and-paste into a Word document, simply execute the graph command, right-click the mouse within the graph window and select copy. Then within a Word document use either paste or paste-special to paste the graph.

7.3 Saving a Graph to Disk

Graphs can be saved to disk as Stata graphs with the .gph extension. These can then be opened in Stata independently of the data i.e. they can be redrawn without any need for the data originally used to draw it.

Graphs can be saved to disk in Stata format in several ways.

Firstly, having drawn a graph, it is possible to save as a Stata graph interactively by right clicking within the Graph Window and selecting the Save as... option (Figure 7.3). The default is to save as a Stata graph (.gph) but there are other options which we will come back to later.

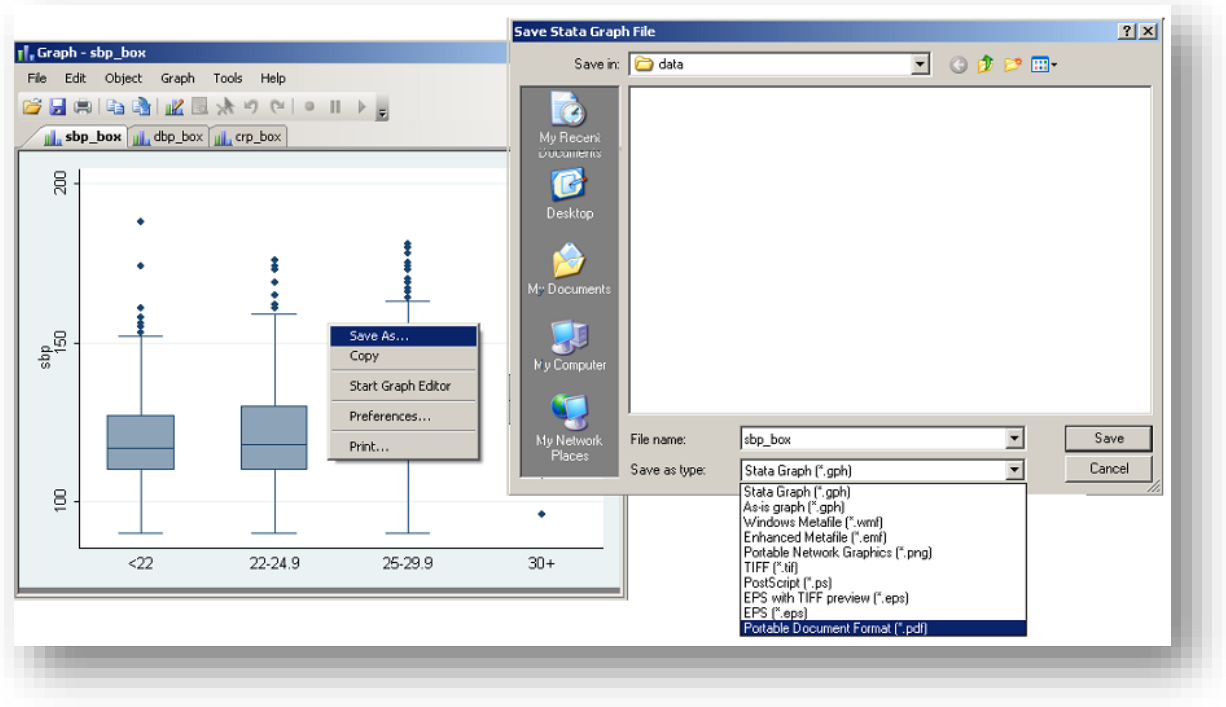


Figure 7.3: Saving a graph interactively

Secondly, as part of the graph syntax, we can save a graph using `saving(`*`filename`* `[,replace])`. For example,

```
. graph twoway scatter sbp bmi, ms(oh) saving(graph1, replace)
```

This will save the graph as *graph1.gph* to the current working directory. We specified the option replace which allows *graph1.gph* to be overwritten should it already exist in the working directory.

Thirdly, a graph can be saved by using the `graph save filename [, replace]` command which saves the graph currently displayed in the graph window, e.g.

```
. graph twoway scatter sbp age
. graph save graph1, replace
```

To redraw a Stata graph saved on disk we can use the graph use command. E.g.

```
. graph use "E:\Stata Graphics\Data\sbp_box.gph"
```

Or if the graph is in the current working directory just;

```
. graph use sbp_box
```

7.4 Exporting graphs

The `graph export` command allows you to export the graph in a number of different formats. The export format is specified either by adding the appropriate file extension to the filename (e.g. graph1.wmf) or by specifying the `as(format)` option. The options include Windows Metafile (wmf), Windows Enhanced Metafile (emf), PDF (pdf), PostScript (ps), Encapsulated PostScript (eps), Scalable Vector Graphics (svg), Portable Network Graphics (png), or TIFF (tif).

For example, suppose we have a Stata graph saved on disk with the filename graph1.gph which we wish to export to disk as a Windows Enhanced Metafile called graph1.emf we first draw the graph and then export as follows;

```
. graph use graph1
. graph export graph1.emf
```

or

```
. graph export graph1, as(emf)
```

See `help graph export` for more details.

We could also do this interactively using the Save as... option (see Figure 7.3).

8. Combining Graphs

Stata graphs can be combined into a single Stata graph using the `graph combine` command. First we need to draw the graphs and either save them to memory using `name()` or to disk using `saving()`. For example,

```
. use whs, clear

. graph twoway histogram sbp, name(sbphist)
. graph twoway histogram dbp, name(dbphist)
. graph twoway histogram bmi, name(bmihist)
. graph twoway histogram wc, name(wchist)

. graph combine sbphist dbphist bmihist wchist
```

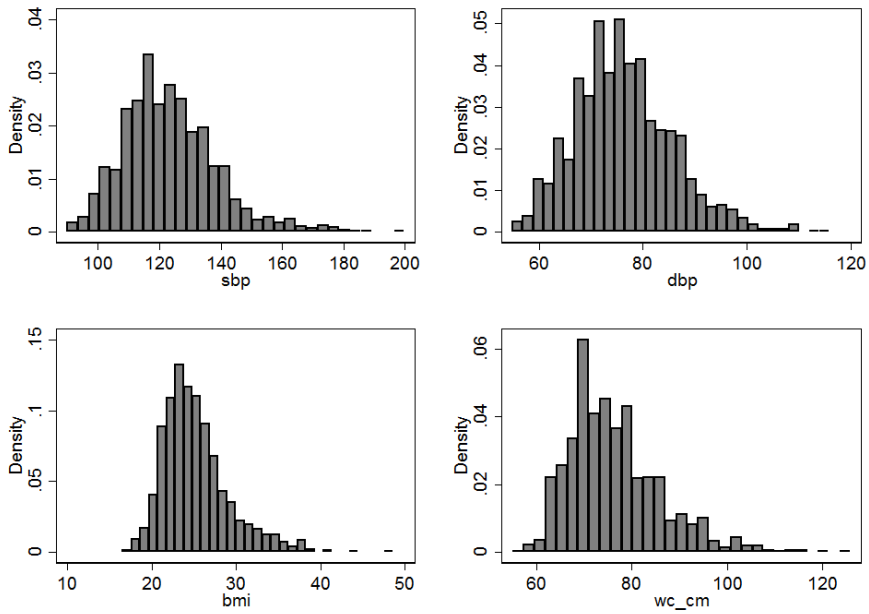


Figure 8.1: Combining Graphs using graph combine

The number of rows and columns can be specified using `rows(#)` and `cols(#)` and common x or y-axes scales can be requested using `xcommon` or `ycommon`.

Using the `holes(#)` option it is possible to specify which position in the grid to leave blank if for example you are combining only 3 graphs in a 2 by 2 grid. Scaling of text and markers in the combined graph can be carried out using `iscale(*#)` and the margins for the individual graphs can be increased or decreased using `imargin(marginstyle)` where marginstyle can be zero, medsmall, medium etc. (see `help marginstyle`).

Some of these options are demonstrated in the demo graphs below.

If combining graphs with a common legend try `net search grc1leg` and download this useful user written command.

9. The Anatomy of a Stata Graph

It can be helpful to understand the anatomy of a Stata graph and how to refer to the various parts of the graph including areas or regions, axis and titles. The command below illustrates this.

```
use whs, clear
#delimit ;
scatter bfp bmi,
msymbol(i)                  // invisible marker symbol to keep plot area clear
      yaxis(1 2)
      plotregion(color(blue*0.4) icolor(blue*0.1) margin(large))
      graphregion(color(green*0.4) icolor(green*0.1) margin(large))
      title(Main title)
      subtitle(Subtitle)
      ytitle("Ytitle (axis 1)" "subtitle")
      ytitle("Ytitle (axis 2)" "subtitle", axis(2))
      xtitle("Xtitle" "subtitle")
      caption(Caption)
      note(Note)
;
```

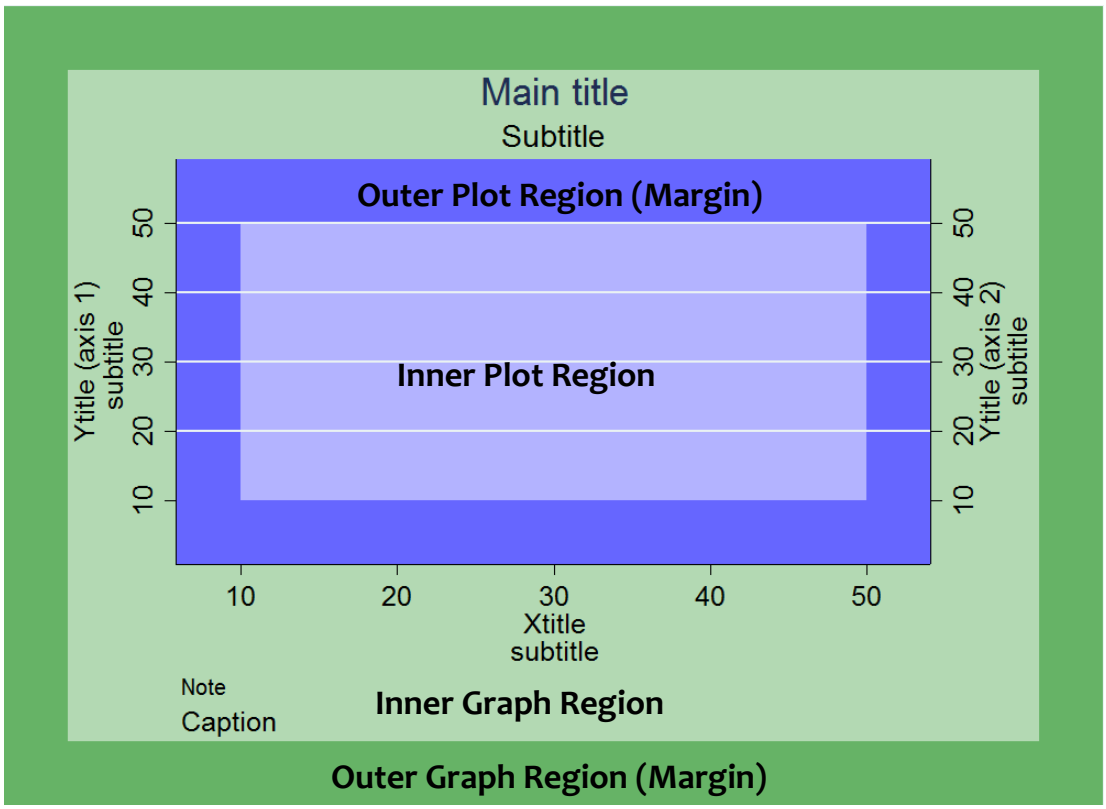


Figure 9.1: Graph Regions and Axes

The plot region is the internal area (light blue) of the graph on which the data are actually plotted. This plot region has a margin (darker blue) which we've specified to be large but could be any size, e.g. zero (for no margin), medium, etc.

The graph region is the rest of the graph (green) including the axis labels, titles and so on. The graph region also has an external margin which here we requested to be large and dark green for emphasis.

We've specified two y-axis using `yaxis(1 2)` and then refer to `axis(1)` or `axis(2)` in the `ytitle()` option.

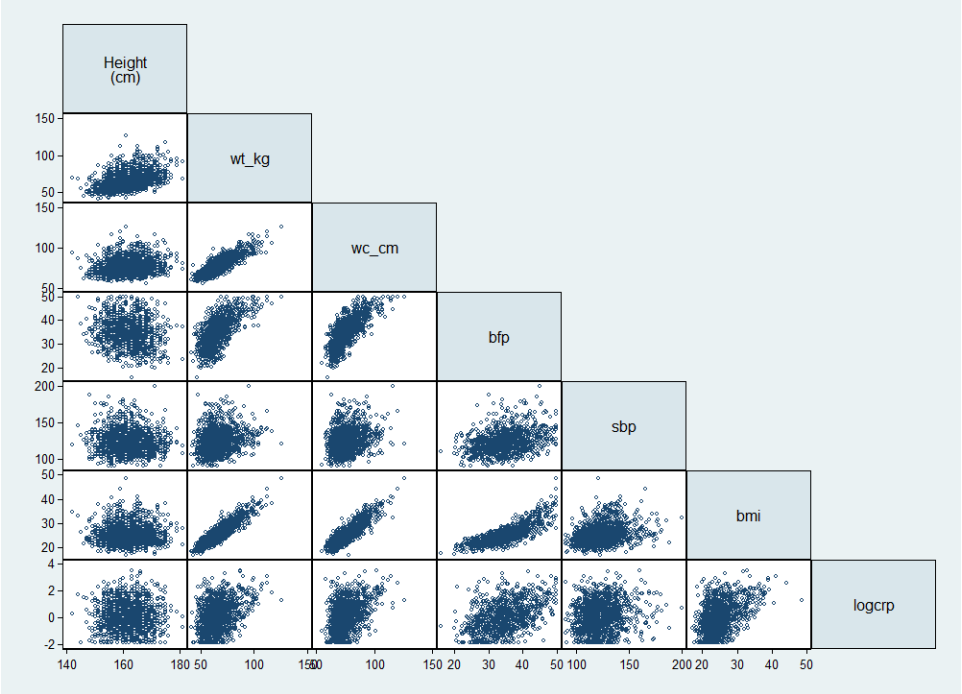
Hopefully the title, x and y-axis titles and note/caption are obvious.

10. A flavour of Stata's graphics potential

The code for the following graphs has been saved in *graph_demo.do.* Before running make sure that you change to the correct working directory.

10.1 Graph Scatter-plot Matrix

```
. use whs, clear
. graph matrix ht_cm wt_kg wc_cm bfp sbp bmi logcrp, half ms(oh)
```

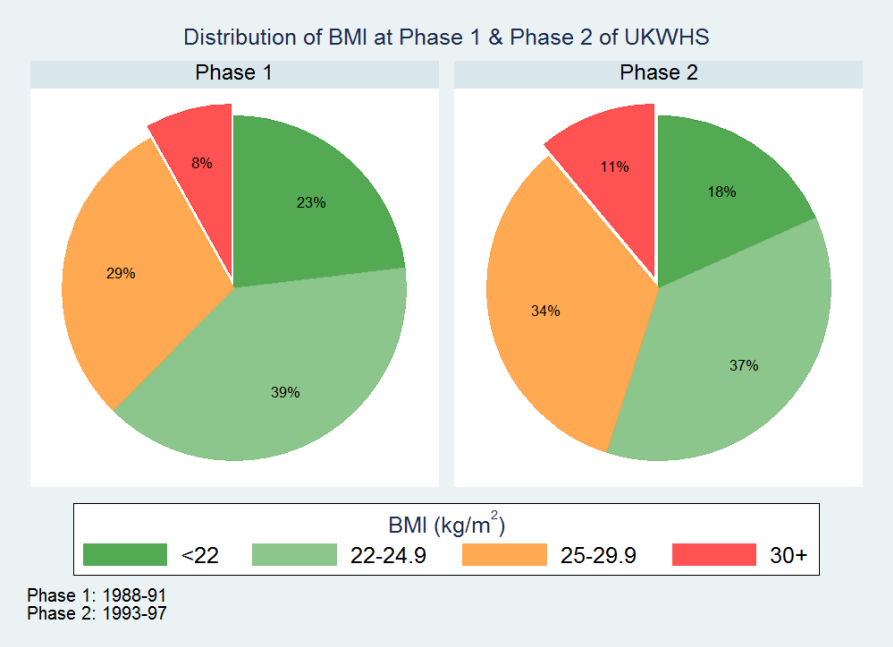


This graph type is useful for visual examinations of all pair-wise two-way associations between a number of continuous variables.

By default Stata will produce a complete (square) matrix. Here the `half` option has been specified and the `ms(oh)` option requests a small hollow circle for the marker symbol.

10.2 Pie charts

```
use graphpie, clear
#delimit ;
graph pie _freq, over(bmi)
      by(phase,
              title("Distribution of BMI at Phase 1 & Phase 2 of UKWHS",
                    size(medsmall))
              note("Phase 1: 1988-91" "Phase 2: 1993-97")) // end of by options
      plabel(_all percent, format(%2.0f))
      pie(1 , color(green*0.75))
      pie(2, color(green*0.5))
      pie(3, color(orange*0.75))
      pie(4, explode color(red*0.75))
      legend(title("BMI (kg/m-sq)", size(medsmall))
      rows(1))
;
```



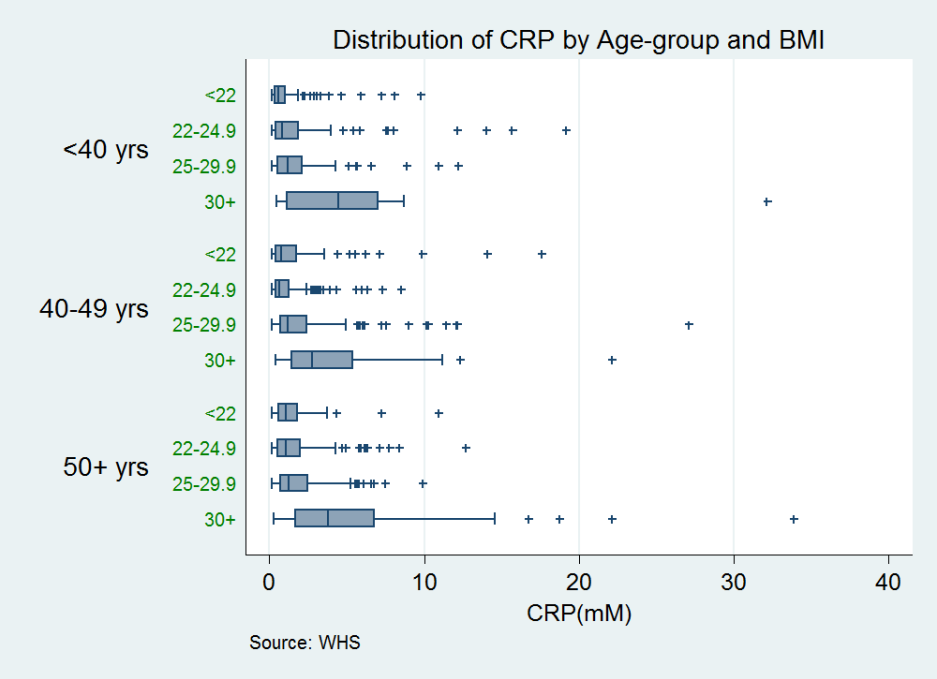
Note that the option `color(green*0.75)` means that we want the colour green toned down in intensity by 25%, i.e. times 0.75. See `help colorstyle` for more details on colour options.

We have added a note to the graph - the way this has been specified means it will be shown on two lines.

Note that the overall title and note are specified within the `by()` option.

10.3 Horizontal Box Plots

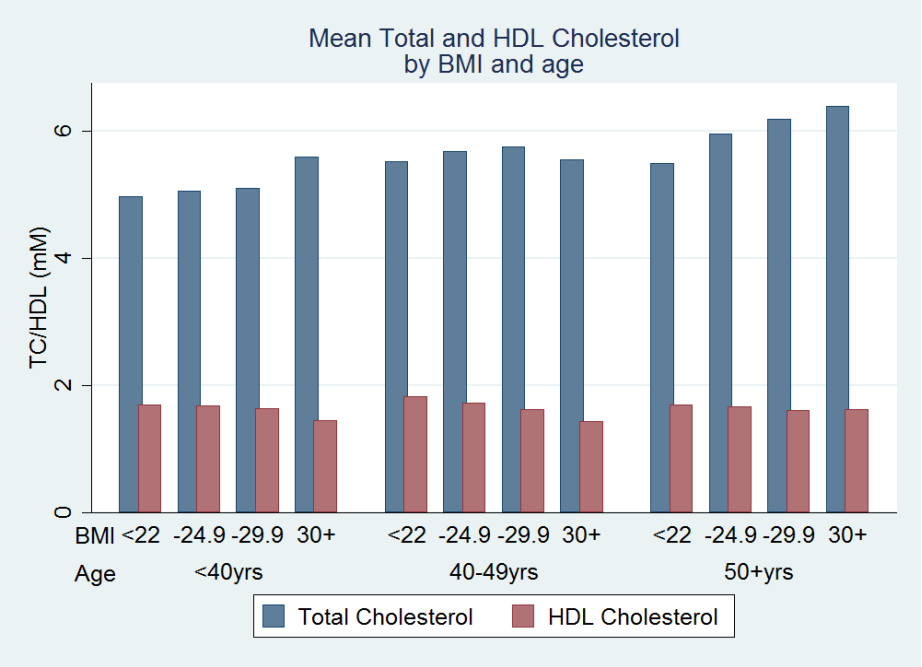
```
use whs, clear
#delimit ;
graph hbox log_crp,
      over(bmi4, label(labsize(small) labcolor(green)))
      over(agegroup, relabel(1 "<40 yrs" 2 "40-49 yrs" 3 "50+ yrs"))
      marker(1, msymbol(smplus))
      title("Distribution of log CRP by BMI and age", color(black)
            size(medium))
      ytitle(Log(CRP(mM)), size(medsmall))
      note(Source: UKWHS)
;
```



Note the option `relabel()` within the `over(agegroup)` option.

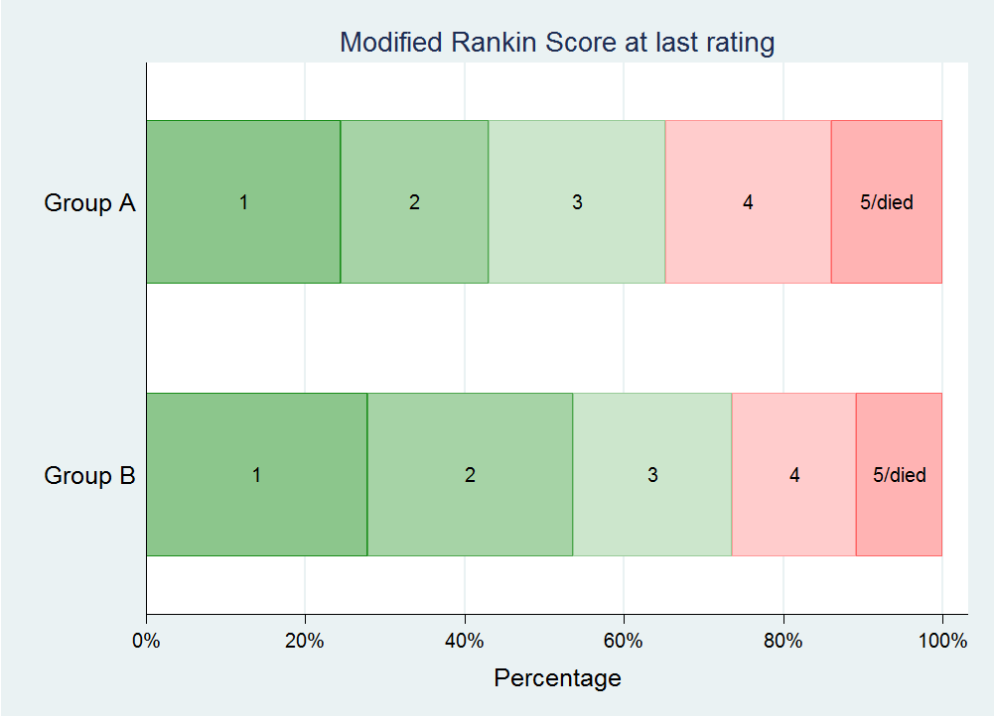
10.4 Bar Charts

```
use whs, clear
#delimit ;
graph bar tc hdl,
        over(bmi4, relabel(2 "-24.9" 3 "-29.9"))
        over(agegroup, relabel( 1 "<40yrs" 2 "40-49yrs" 3 "50+yrs")
                label(labsize(medsmall)))
        intensity(*0.7)
        bargap(-20)
        legend(label(1 "Total Cholesterol") label(2 "HDL Cholesterol")
                symx(*0.25))
        title("Mean Total and HDL Cholesterol", size(medium))
        title("by BMI and age", suffix size(medium))
        ytitle("TC/HDL (mM)")
        text(-0.35 -2.5 "BMI")
        text(-0.95 -2.5 "Age")
;
```



10.5 Stacked Bar Graph

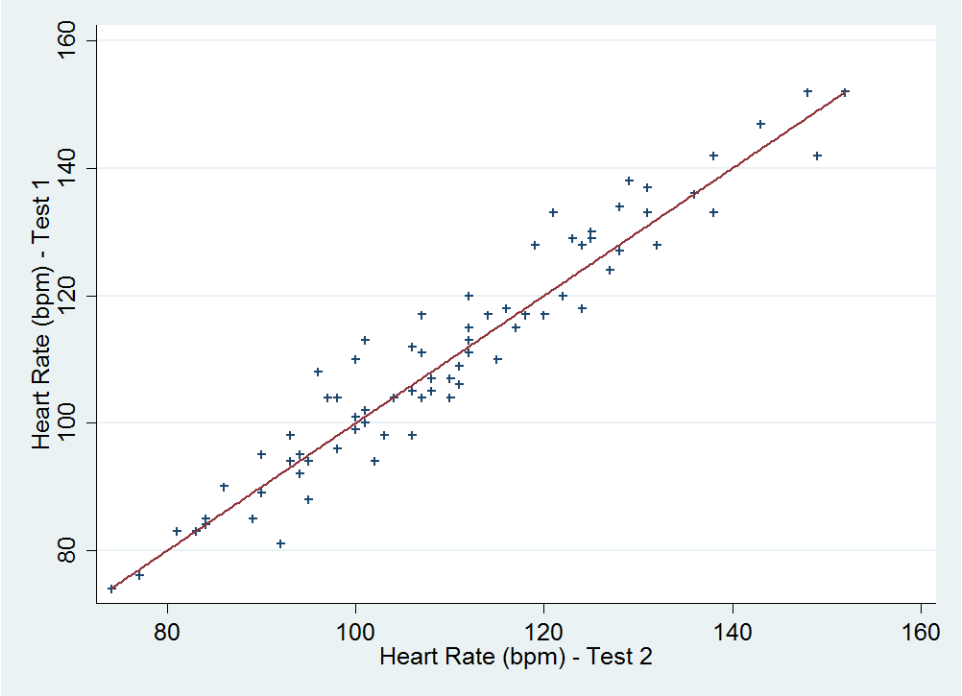
```
use mrs, clear
contract trt mrs
#delimit ;
graph hbar _freq,
          over(mrs)
          over(trt)
          asy
          percent
          stack
          legend(off)
          intensity(*0.5)
          blabel(name, pos(cent))
          ytitle("Percentage", m(t+2))
          title(Modified Rankin Score at last rating, size(medium))
          ylabel(0 "0%" 20 "20%" 40 "40%" 60 "60%" 80 "80%" 100 "100%",
                   labsize(small))
          bar(1, bcol(green*0.9))
          bar(2, bcol(green*0.7))
          bar(3, bcol(green*0.4))
          bar(4, bcol(red*0.4))
          bar(5, bcol(red*0.6))
;
```



Note the use of `bcolor(green*#)` to gradate the colours across the bars.

10.6 Using Twoway Function to Plot line of Y=X

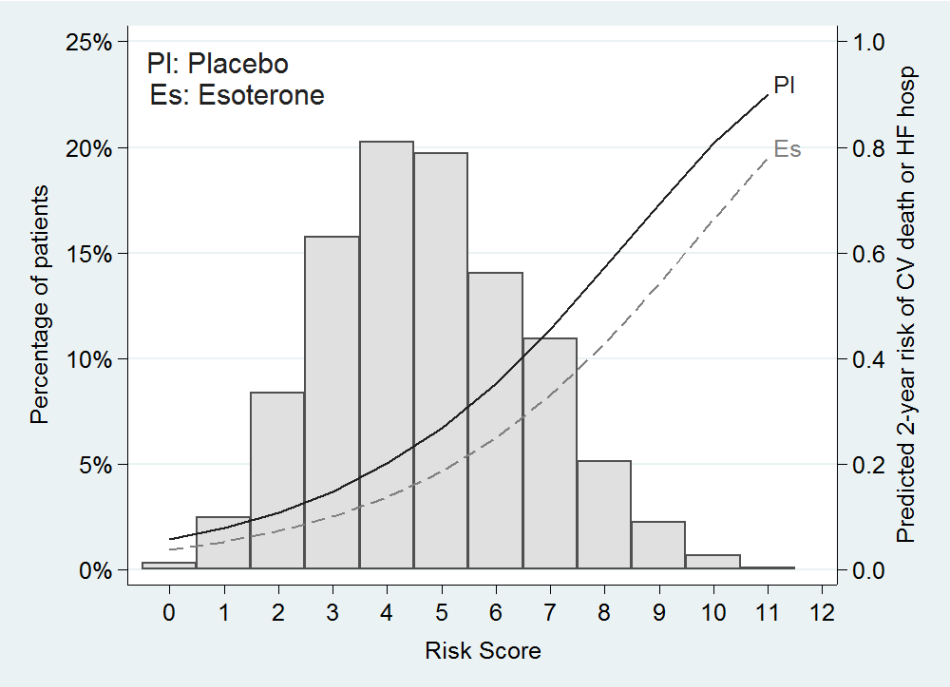
```
use chester.dta, clear
#delimit ;
twoway (scatter hr1 hr2, ms(smplus))
         (function y = x, range(hr2)),
         ytitle(Heart Rate (bpm)- Test 1)
         xtitle(Heart Rate (bpm)- Test 1)
        legend(off)
;
```



The code `function(y=x, range(hr2))` specifies that we want to plot a line of y = x drawn over the range of the variable hr2.

10.7 Overlaid histogram and line graphs: plotting on two axis

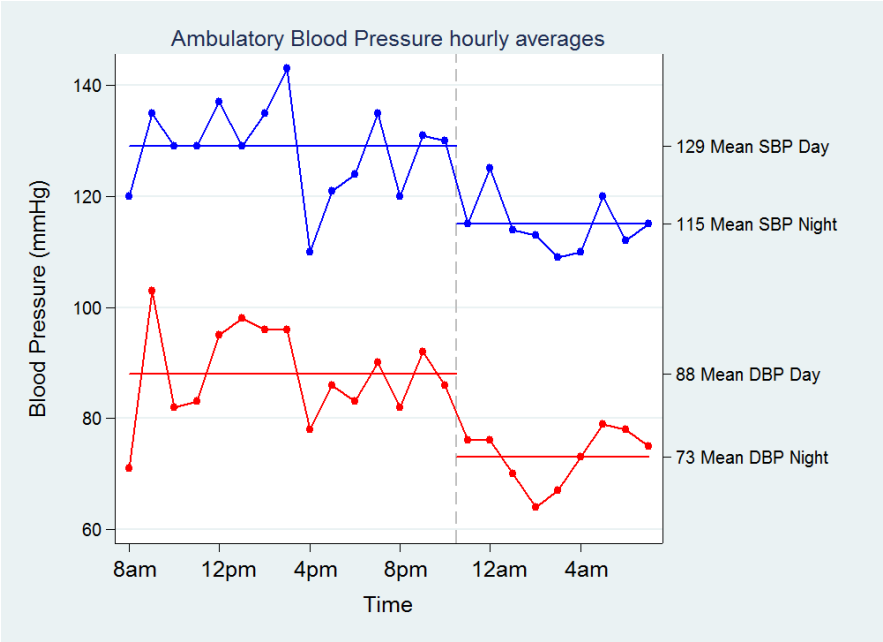
```
use rscoredist, clear
#delimit ;
twoway (hist rscore, yaxis(1) percent discrete bfcol(gs14) blcol(gs5))
      (line predfail2 rscore if trt==1, c(l) yaxis(2) sort lcol(gs8) lpat(dash))
      (line predfail2 rscore if trt==0, c(l) yaxis(2) sort lcol(gs2)),
      legend(off)
      xtitle("Risk Score", m(t+2))
      ytitle("Percentage of patients", axis(1))
      ytitle("Predicted 2-year risk of CV death or HF hosp", axis(2))
      xlab(0(1)12)
      ytick(0(5)25 )
      ylab(0 "0%" 5 "5%" 10 "10%" 15 "15%" 20 "20%" 25 "25%", gmax angle(hori) axis(1))
      ylab(0(0.2)1, angle(hori) format(%2.1f) axis(2))
      text(24 -0.5 "Pl: Placebo", place(e) size(medlarge) col(gs2))
      text(22.5 -0.5 "Es: Esoterone" , place(e) size(medlarge) col(gs2))
      text(23 11.1 "Pl", col(gs2) place(e) size(medsmall))
      text(20 11.1 "Es", col(gs8) place(e) size(medsmall))
;
```



Note that we are using two y-axes in this graph; a histogram is being plotted on y-axis 1 and two line graphs on y-axis 2.

10.8 Twoway Connected Line Plot and Function Plot

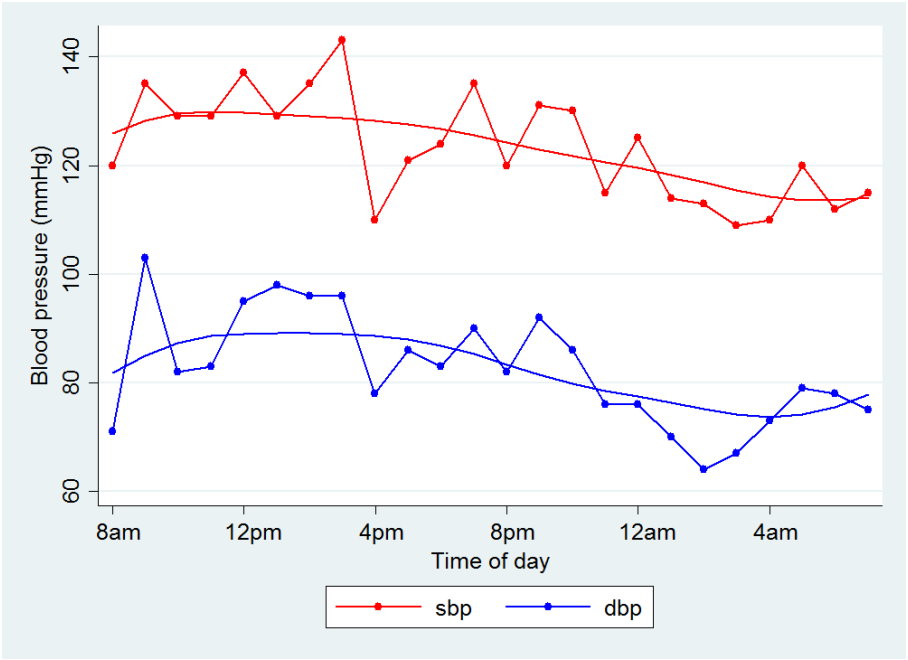
```
use amb_bp, clear
#delimit ;
twoway (connected sbp dbp obs, mcolor(blue red) clcolor(blue red)
        msize(small small) yaxis(1 2))
        (function y=129, range(1 15.5) clcolor(blue))
        (function y=88, range(1 15.5) clcolor(red))
        (function y=115, range(15.5 24) clcolor(blue))
        (function y=73,  range(15.5 24) clcolor(red)),
        title("Ambulatory Blood Pressure hourly averages", size(medsmall))
        xtitle(Time, margin(t+2) )
        ytitle("Blood Pressure (mmHg)", m(r+2) axis(1))
        xlab(1(4)23, valuelabel)
        ylab(, labsize(small)  angle(hori) axis(1))
        ylab(73 "73 Mean DBP Night" 88 "88 Mean DBP Day"
                115 "115 Mean SBP Night" 129 "129 Mean SBP Day",
                labsize(small) angle(hori) axis(2))
        xline(15.5,  lcolor(gs12) lp(dash))
        legend(off)
;
```



Note that the yaxis(1 2) specifies we will use two y-axis. Then when labelling each axis we need to specify which axis we are referring to using the sub-option axis(#).

10.9 Twoway connected plot with moving average

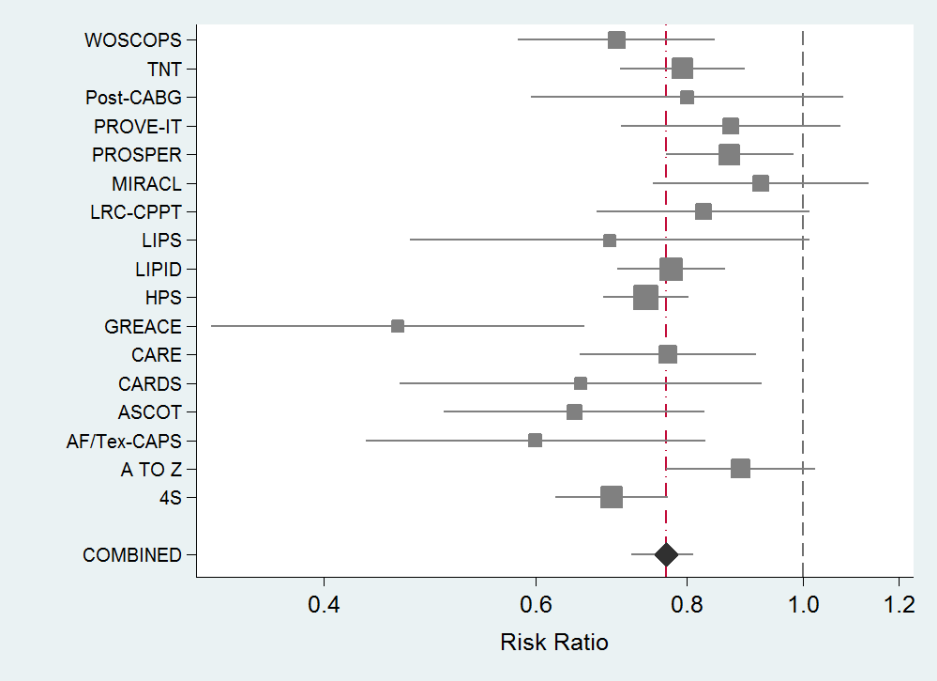
```
use amb_bp, clear

#delimit ;
graph twoway (connected sbp dbp obs, msize(small small) mc(red blue)
          clc(red blue))
          (ksm sbp obs, clc(red))
          (ksm dbp obs, clc(blue)),
          xlab(1(4)23, value)
          legend(order(1 2))
          ytitle(Blood pressure (mmHg), m(r+1))
          xtitle(Time of day, m(t+1))
          saving(abp1, replace)
;
```



`ksm()` graphs a moving average over time. Value labels in the dataset have been requested for the x-axis labels.

10.10 Use of weighting for marker symbols

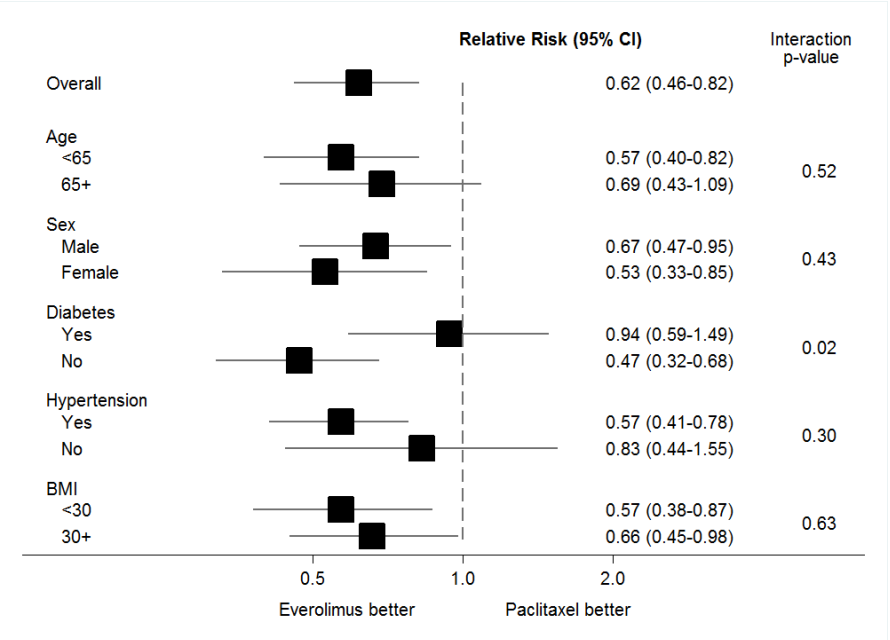
```
use lipid2, clear
#delimit ;
twoway (rspike low up trial, horizontal clc(gs8))
      (scatter trial rr if trial>0 [fw=w], ms(s) mcol(gs8))
      (scatter trial rr if trial==-1 [fw=w], ms(D) mcol(gs3)),
      ytitle("")
      xlab(, format(%2.1f))
      legend(off)
      xscale(log)
      yscale(range(-1.25 16.25))
      ylab(-1 1(1)17, valuelabel labsize(small) angle(hori) nogrid)
      xline(0.77, lp(dash_dot))
      xline(1, lp(dash) lc(gs7))
      xtitle("Risk Ratio", size(medsmall) m(t+2))
;
```



Note here the use of frequency weights in the two scatter plots `[fw=w]` where w is a variable in the dataset indicating the number of patients in each study.

10.11 Subgroup Plot

```
#delimit ;
twoway (rspike low up trial2, lcol(gs7) horizontal)
       (scatter trial2 rr, ms(S) msize(*2.5)  mcol(gs0))
       (scatter trial2 x, mlab(lab) ms(i) mlabpos(0) mlabcolor(black))
       (scatter trial y if trial==2, mlab(variable) ms(i) mlabpos(3)
              mlabcolor(black))
       (scatter trial3 y if trial>2, mlab(variable) ms(i) mlabpos(3)
              mlabcolor(black))
       (scatter trial2 z, mlab(level) ms(i) mlabpos(3) mlabcolor(black))
       (scatter trial z1, mlab(pvals) ms(i) mlabpos(3) mlabcolor(black))
       (pci 2 1 7.2 1, lcol(gs7) lpat(dash)),
       yscale(reverse range(1.5 7.2))
       xlab(0.5 1.0 2, format(%2.1f) labsize(small))
       ylab(, nogrid)
       xscale(log range(0.4 6))
       legend(off)
       xtitle("Everolimus better                Paclitaxel better",
              size(small) m(t+2))
       plotregion(style(none) icolor(white))
       graphregion(style(none) color(white))
       yscale(off)
       name(fplot, replace)
       text(1.5 1.5 "{bf:Relative Risk (95% CI)}", size(small))
       text(1.6 5 "Interaction" "p-value", size(small))
;
```



10.12 Combining Graphs

```
use whs1, clear
#delimit ;
graph twoway (qfitci bfp wc_cm, bcol(ltblue))
              (scatter bfp wc_cm, ms(+) msize(tiny)),
              ytitle(BFP (%), size(medsmall) )
              xtitle(Waist circumference (cm), size(medsmall))
              xscale(alt range(50 140))
              yscale(alt range(15 60))
              xlab(, labsize(small) grid gmax)
              ylab(20(10)60, labsize(small))
              legend(off)
              name(wc_bfp, replace)
;
graph twoway histogram bfp,
              fraction
              xscale( alt reverse)
              yscale(range(15 60))
              ylab(20(10)60, labsize(small))
              xlab(, labsize(small))
              horizontal
              name(bfp, replace)
;
graph twoway histogram wc_cm, fraction
              yscale(alt reverse)
              xscale(range(50 140))
              ylab(, nogrid labsize(small))
              xlab(60(20)140, grid gmax labsize(small))
              name(wc, replace)
;
graph combine bfp.gph wc_bfp.gph wc.gph, hole(3) imargin(0 0 0 0)
;
```