1 Introduction

This chapter begins by briefly telling you about the organization of this book and giving you tips to help you use it most effectively. The next section gives a short overview of the different kinds of Stata graphs that will be examined in this book, and that section is followed by an overview of the different kinds of schemes that will be used for showing the graphs in this book. The fourth section illustrates the structure of options in Stata graph commands. In a sense, the second, third, and fourth sections of this chapter are a thumbnail preview of the entire book, showing the types of graphs covered, how you can control their overall look, and the general structure of options used within those graphs. The final section is about the process of creating graphs.

1.1 Using this book

I hope that you are eager to start reading this book but will take just a couple of minutes to read this section to get some suggestions that will make the book more useful to you. First, there are many ways you might read this book, but perhaps I can suggest some tips:

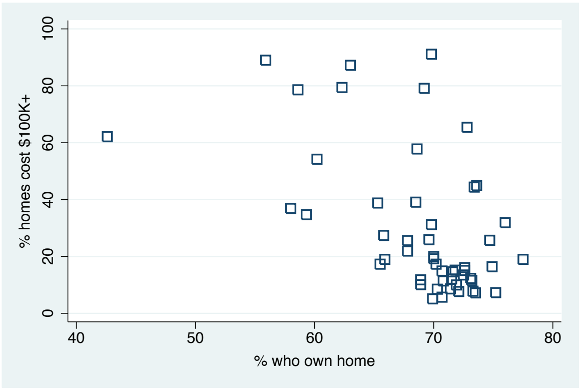
- Read this chapter before reading the other chapters, as it provides key information that will make the rest of the book more understandable.
- Although you might read a traditional book cover to cover, this book has been written so that the chapters stand on their own. You should feel free to dive into any chapter or section of any chapter.
- Sometimes you might find it useful to visually scan the graphs rather than to read. I think this is a good way to familiarize yourself with the kinds of features available in Stata graphs. If a certain feature catches your eye, you can stop and see the command that made the graph and even read the text explaining the command.
- Likewise, you might scan a chapter just by looking at the graphs and the part of the command in red, which is the part of the command highlighted in that graph. For example, scanning the chapter on bar charts in this way would quickly familiarize you with the kinds of features available for bar graphs and would show you how to obtain those features.

The right margin contains what I call the *Visual Table of Contents*. It is a useful tool for quickly finding the information you seek. I frequently use the *Visual Table of Contents* to cross-reference information within the book. By design, Stata graphs share many common features. For example, you use the same kinds of options to control legends across different types of graphs. It would be repetitive to go into detail about legends for bar charts, box plots, and so on. Within each kind of graph, legends are briefly described and illustrated, but the details are described in the *Options* chapter in the section titled *Legend*. This is cross-referenced in the book by saying something like "for more details, see Options : Legend (353)", indicating that you should look to the *Visual Table of Contents* and thumb to the *Options* chapter and then to the *Legend* section, which begins on page 353.

Sometimes it may take an extra cross-reference to get the information you need. Say that you want to make the *y*-axis title large for a bar chart by using the `ytitle()` option, so you first consult Bar : Y-axis (205). This gives you some information about using `ytitle()`, but then that section refers you to Options : Axis titles (319), where more details about axis titles are described. This section then refers you to Options : Textboxes (371) for more complete details about options to control the display of text. That section shows more details but then refers to Styles : Textsize (414), where all the possible text sizes are described. I know this sounds like a lot of jumping around, but I hope that it feels more like drilling down for more detail, that you feel you are in control of the level of detail that you want, and that the *Visual Table of Contents* eases the process of getting the additional details.

Most pages of this book have three graphs per page, with each graph being composed of the graph itself, the command that produced it, and some descriptive text. An example is shown below, followed by some points to note.

```
graph twoway scatter propval100 ownhome, msymbol(Sh)
```



Here we use the `msymbol()` (marker symbol) option to make the symbols large hollow squares; see Options : Markers (299) for more details. The `graph twoway` portion of the command is optional.

Double-click on any of the markers and change the **Symbol** to `Hollow square`.

Uses allstates.dta & scheme vg_s2c

- The command itself is displayed in a `typewriter font`, and the salient part of the command (i.e., `msymbol(Sh)`) is in `this color`—both in the command and when referenced in the descriptive text.
- When commands or parts of commands are given in the descriptive text (e.g., `graph twoway`), they are displayed in the `typewriter font`.
- Many of the descriptions contain cross-references, for example, Options : Markers (299), which means to flip to the *Options* chapter and then to the section *Markers*. Equivalently, go to page 299.
- The names of some options are shorthand for two or more words that are sometimes explained; for instance, "we use the `msymbol()` (marker symbol) option to make . . . ".

- Many examples include more instructions describing how the Stata Graph Editor can be used to accomplish the same customization as illustrated in the command (in this example, how the Graph Editor can be used to obtain the equivalent of `msymbol(Sh)`). The icon indicates that the instructions that follow apply to the use of the Graph Editor. The instructions assume that you have run the command *omitting* the highlighted portion of the command (e.g., omitting `msymbol(Sh)`) and that you have started the Graph Editor. The Graph Editor can be started in one of three ways: 1) by selecting **File** and then **Start Graph Editor** from the Graph window menu, 2) by clicking the Start Graph Editor icon in the Graph window toolbar, or 3) by right-clicking on the graph and selecting **Start Graph Editor**. Once the Graph Editor is started, you can follow the instructions given (e.g., you can double-click on any of the markers, and in the dialog box that appears, you can then change the setting for the *Symbol* option to *Hollow square*). See Editor (33) for more details about using the Graph Editor.
- The descriptive text always concludes by telling you the name of the data file and scheme used for making the graph. Here the data file was *allstates.dta*, and the scheme was *vg_s2c.scheme*. You can read the data file over the Internet by using the `vguse` command, which is added to Stata when you install the online supplements; see Appendix : Online supplements (456). If you are connected to the Internet and your Stata is fully up to date, you can simply type `vguse allstates` to use that file over the Internet, and you can run the graph command shown to create the graph.
- Sometimes there is not enough space to describe the command as well as describe how to use the Graph Editor to accomplish the customization illustrated. In such cases, the description will conclude with " See the next graph". The descriptive text for the next example will begin with the icon and will be dedicated to illustrating how to use the Graph Editor for that particular customization.

If you want your graphs to look like the ones in the book, you can display them using the same schemes. See Appendix : Online supplements (456) for information about how to download the schemes used in this book. Once you have downloaded the schemes, you can then type the following commands in the Stata Command window:

```
. set scheme vg_s2c
. vguse allstates
. graph twoway scatter propval100 ownhome, msymbol(Sh)
```

After you issue the `set scheme vg_s2c` command, subsequent graph commands will show graphs with the `vg_s2c` scheme. You could also add the `scheme(vg_sc2)` option to the graph command to specify that the scheme be used just for that graph; for example,

```
. graph twoway scatter propval100 ownhome, msymbol(Sh) scheme(vg_s2c)
```

Generally, all commands and options are provided in their complete form. Commands and options are usually not abbreviated. However, for purposes of typing, you may want to use abbreviations. The previous example could have been abbreviated to

```
. gr tw sc propval100 ownhome, m(Sh)
```

The `gr` could have been omitted, leaving

```
. tw sc propval100 ownhome, m(Sh)
```

The `tw` also could have been omitted, leaving

```
. sc propval100 ownhome, m(Sh)
```

For guidance on appropriate abbreviations, consult [G] **graph**.

This book has been written based on the features available in Stata version 10.0. In the future, Stata may evolve to make the behavior of some of these commands change. If this happens, you can use the `version` command to make Stata run the graph commands as though they were run under version 10.0. For example, if you were running Stata version 11.0 but wanted a graph command to run as though you were running Stata 10.0, you could type

```
. version 10.0: graph twoway scatter propval100 ownhome
```

and the command would be executed as if you were running version 10.0. Or, perhaps you want a command to run as it did under Stata 9.2, you would then type

```
. version 9.2: graph twoway scatter propval100 ownhome
```

This book has a number of associated online resources to complement the book. Appendix : Online supplements (456) has more information about these online resources and how to access them. I strongly suggest that you install the online supplements, which make it easier to run the examples from the book. To install the supplemental programs, schemes, and help files, type from within Stata

```
. net from http://www.stata-press.com/data/vgsg
. net install vgsg
```

For an overview of what you have installed, type `help vgsg` within Stata. Then, with the `vguse` command, you can use any dataset from the book. Likewise, all the custom schemes used in the book will be installed into your copy of Stata, and you can use them to display the graphs, as described earlier in this section.

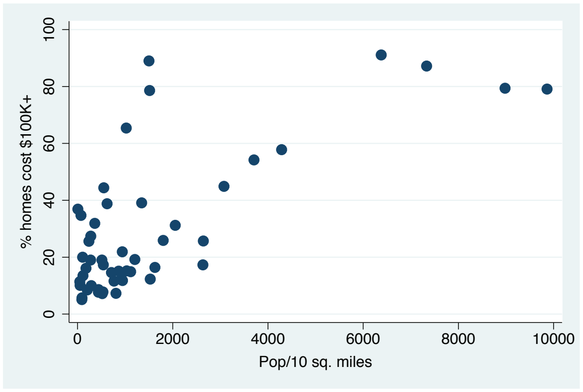
Finally, I would like to emphasize that the goal of this book is to help you learn and use the Stata graph commands and the Graph Editor for the purposes of creating graphs in Stata. I assume that you know the kind of graph you want to create and that you are turning to this book for advice on how to make that graph. I don't provide guidance on how to select the right kind of graph for visualizing your data or the merits of one graphical method over another. For such guidance, I would refer readers to books such as *The Visual Display of Quantitative Information, Second Edition* by Edward R. Tufte and *Visualizing Data* by William S. Cleveland, as well as your favorite statistical book.

1.2 Types of Stata graphs

Stata has a wide variety of graph types. This section introduces the types of graphs Stata produces, and it covers twoway plots (including scatterplots, line plots, fit plots, fit plots with confidence intervals, area plots, bar plots, range plots, and distribution plots), scatterplot matrices, bar charts, box plots, dot plots, and pie charts. Let's begin by exploring the variety of twoway plots that can be created with `graph twoway`. For this introduction, they are combined into six families of related plots: scatterplots and fit plots, line plots, area plots, bar plots, range plots, and distribution plots. Now let's turn to scatterplots and fit plots.

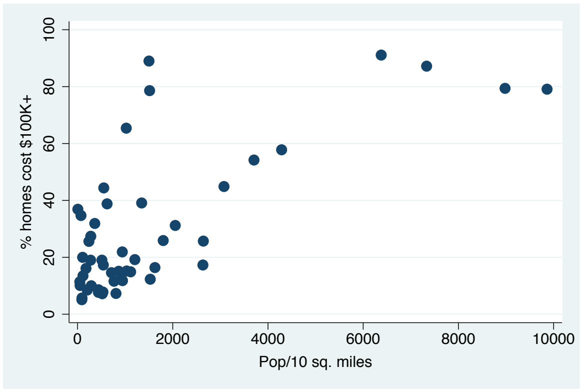
```
graph twoway scatter propval100 popden
```

Here is a basic scatterplot. The variable `propval100` is placed on the y axis, and `popden` is placed on the x axis. See Twoway : Scatter (87) for more details about these kinds of plots.
Uses allstates.dta & scheme vg_s2c

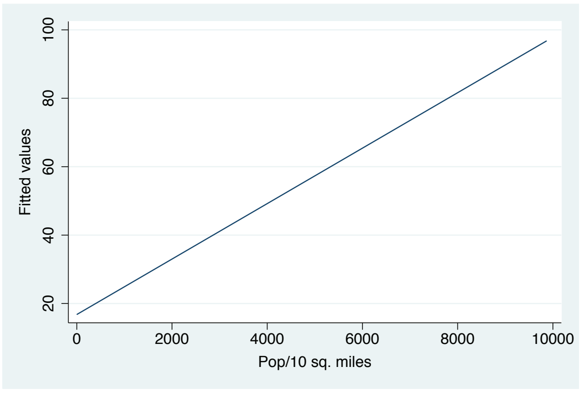


```
twoway scatter propval100 popden
```

We can start the previous command with just `twoway`, and Stata understands that this is shorthand for `graph twoway`.
Uses allstates.dta & scheme vg_s2c

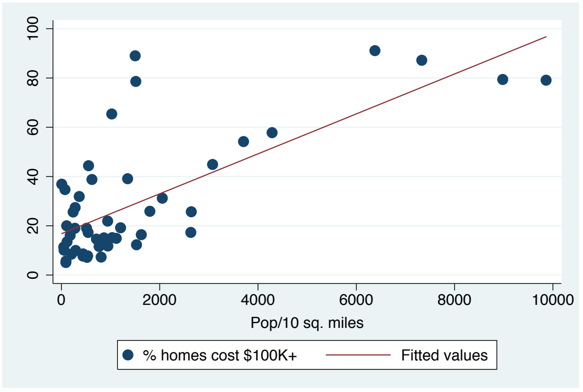


```
twoway lfit propval100 popden
```



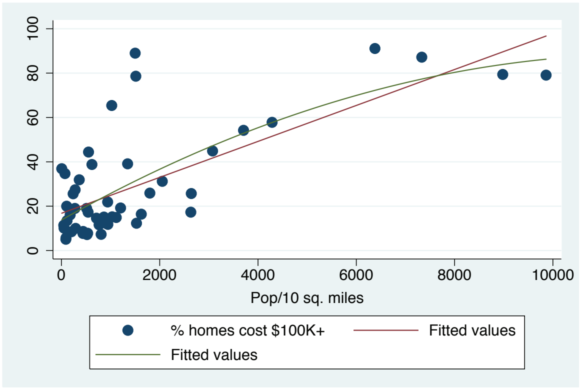
We now make a linear fit (`lfit`) line predicting `propval100` from `popden`. See Twoway : Fit (104) for more information about these kinds of plots.
Uses allstates.dta & scheme vg_s2c

```
twoway (scatter propval100 popden) (lfit propval100 popden)
```



Stata allows us to overlay `twoway` graphs. In this example, we make a classic plot showing a scatterplot overlaid with a fit line by using the `scatter` and `lfit` commands. For more details about overlaying graphs, see Twoway : Overlaying (143).
Uses allstates.dta & scheme vg_s2c

```
twoway (scatter propval100 popden) (lfit propval100 popden)
   (qfit propval100 popden)
```

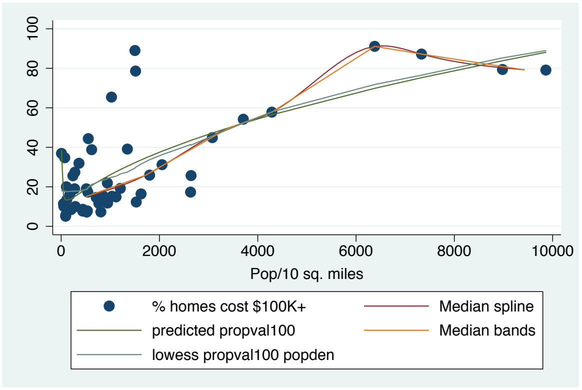


The ability to combine `twoway` plots is not limited to overlaying just two plots; we can overlay multiple plots. Here we overlay a scatterplot (`scatter`) with a linear fit (`lfit`) line and a quadratic fit (`qfit`) line.
Uses allstates.dta & scheme vg_s2c

```
twoway (scatter propval100 popden) (mspline propval100 popden)
   (fpfit propval100 popden) (mband propval100 popden)
   (lowess propval100 popden)
```

Stata has other kinds of fit methods in addition to linear and quadratic fits. This example includes a median spline (`mspline`), fractional polynomial fit (`fpfit`), median band (`mband`), and lowess (`lowess`). For more details, see Twoway : Fit (104).

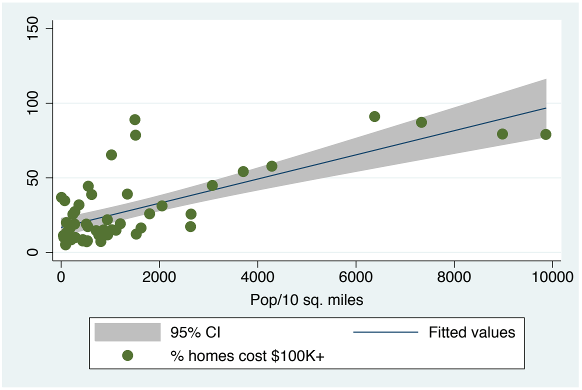
Uses allstates.dta & scheme vg_s2c



```
twoway (lfitci propval100 popden) (scatter propval100 popden)
```

In addition to being able to plot a fit line, we can plot a linear fit line with a confidence interval by using the `lfitci` command. We also overlay the linear fit and confidence interval with a scatterplot. See Twoway : CI fit (106) for more information about fit lines with confidence intervals.

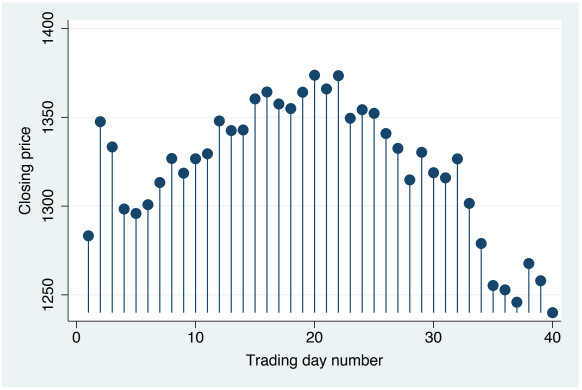
Uses allstates.dta & scheme vg_s2c



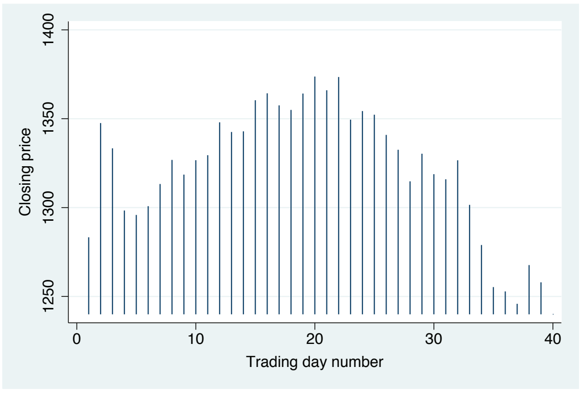
```
twoway dropline close tradeday
```

This `dropline` graph shows the closing prices of the S&P 500 by trading day for the first 40 days of 2001. A `dropline` graph is like a `scatterplot` because each data point is shown with a marker, but a dropline for each marker is shown as well. For more details, see Twoway : Scatter (87).

Uses spjanfeb2001.dta & scheme vg_s2c

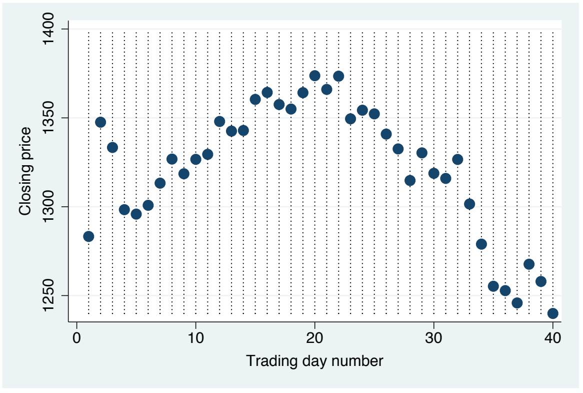


```
twoway spike close tradeday
```



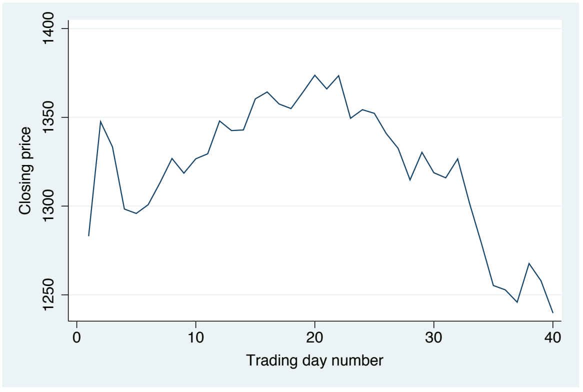
Here we use a `spike` plot to show the same graph as the previous one. It is like the `dropline` plot, but no markers are put on the top. For more details, see Twoway : Scatter (87).
Uses spjanfeb2001.dta & scheme vg_s2c

```
twoway dot close tradeday
```



The `dot` plot, like the `scatter`plot, shows markers for each data point but also adds a dotted line for each of the x values. For more details, see Twoway : Scatter (87).
Uses spjanfeb2001.dta & scheme vg_s2c

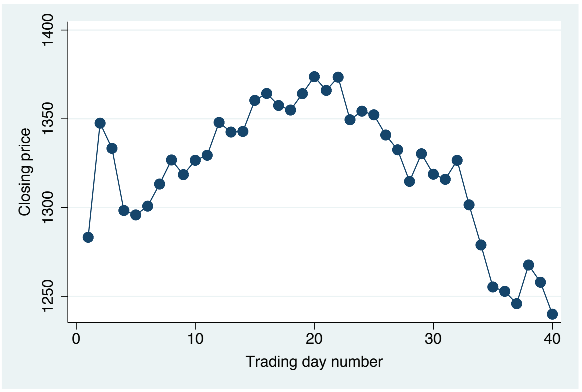
```
twoway line close tradeday, sort
```



We use the `line` command in this example to make a simple line graph. See Twoway : Line (110) for more details about line graphs.
Uses spjanfeb2001.dta & scheme vg_s2c

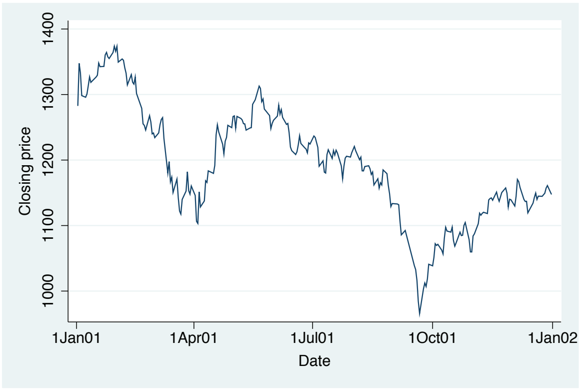
```
twoway connected close tradeday, sort
```

This `twoway` `connected` graph is similar to the `twoway line` graph, except that a symbol is shown for each data point. For more information, see Twoway : Line (110).
Uses spjanfeb2001.dta & scheme vg_s2c



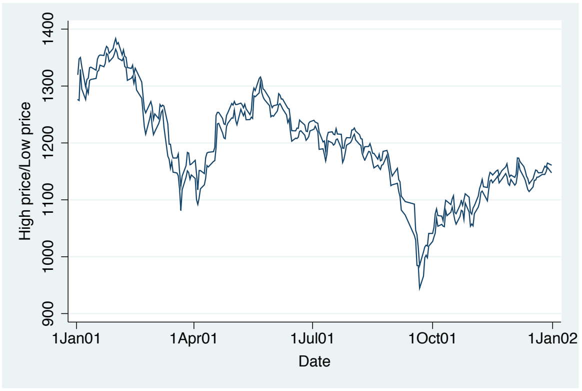
```
twoway tsline close, sort
```

The `tsline` (time-series line) command makes a line graph where the x variable is a date variable that has been previously declared by using `tsset`; see [TS] **tsset**. This example shows the closing price of the S&P 500 by trading date. For more information, see Twoway : Line (110).
Uses sp2001ts.dta & scheme vg_s2c

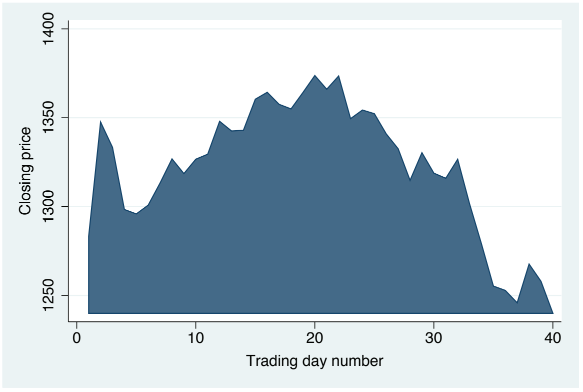


```
twoway tsrline high low, sort
```

The `tsrline` (time-series range line) command makes a line graph showing the high and low prices of the S&P 500 by trading date. For more information, see Twoway : Line (110).
Uses sp2001ts.dta & scheme vg_s2c

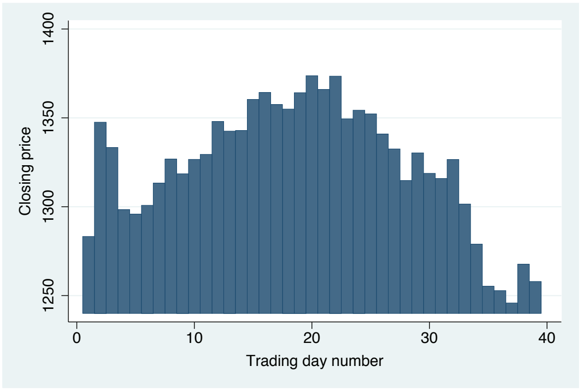


```
twoway area close tradeday, sort
```



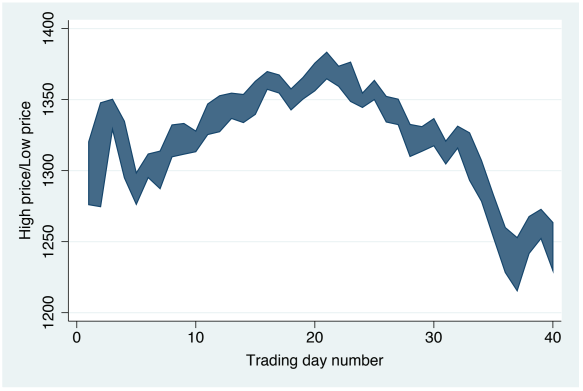
An area plot is similar to a line plot, but the area under the line is shaded. See Twoway : Area (117) for more information about area plots.
Uses spjanfeb2001.dta & scheme vg_s2c

```
twoway bar close tradeday
```



Here is an example of a twoway bar plot. For each x value, a bar is shown corresponding to the height of the y variable. This command shows a continuous x variable as compared with the graph bar command, which would be useful when you have a categorical x variable. See Twoway : Bar (119) for more details about bar plots.
Uses spjanfeb2001.dta & scheme vg_s2c

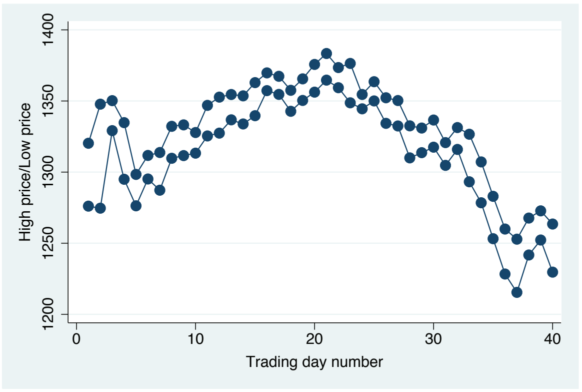
```
twoway rarea high low tradeday, sort
```



This example illustrates the use of rarea (range area) to graph the high and low prices with the area filled. If you used rline (range line), the area would not be filled. See Twoway : Range (121) for more details.
Uses spjanfeb2001.dta & scheme vg_s2c

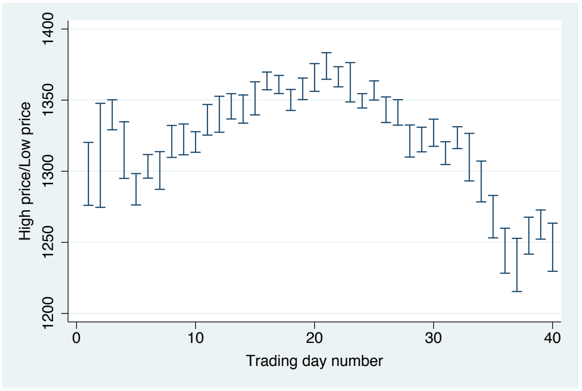
```
twoway rconnected high low tradeday, sort
```

The rconnected (range connected) command makes a graph similar to the previous one, except that a marker is shown at each value of the x variable and the area between is not filled. If you instead used rscatter (range scatter), the points would not be connected. See Twoway : Range (121) for more details.
Uses spjanfeb2001.dta & scheme vg_s2c



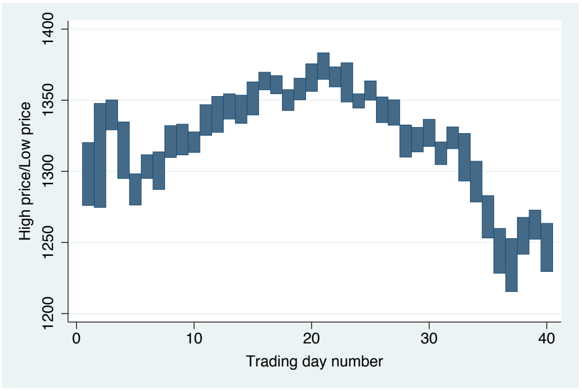
```
twoway rcap high low tradeday, sort
```

Here we use rcap (range cap) to graph the high and low prices with a spike and a cap at each value of the x variable. If you used rspike instead, spikes would be displayed but not caps. If you used rcapsym, the caps would be symbols that could be modified. See Twoway : Range (121) for more details.
Uses spjanfeb2001.dta & scheme vg_s2c

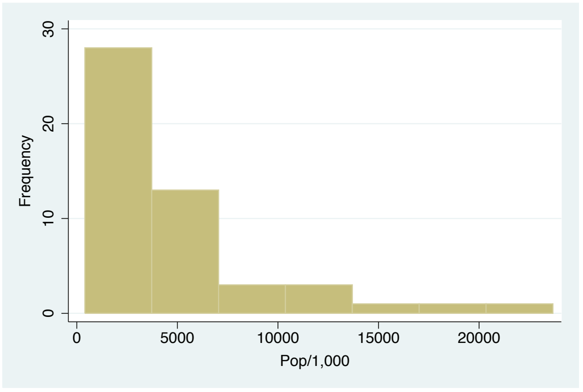


```
twoway rbar high low tradeday, sort
```

The rbar command graphs the high and low prices with bars at each value of the x variable. See Twoway : Range (121) for more details.
Uses spjanfeb2001.dta & scheme vg_s2c

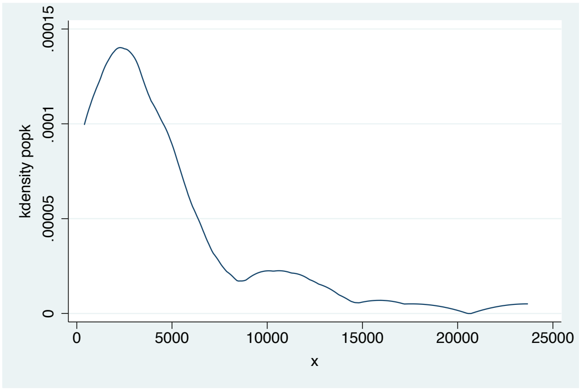


```
twoway histogram popk, freq
```



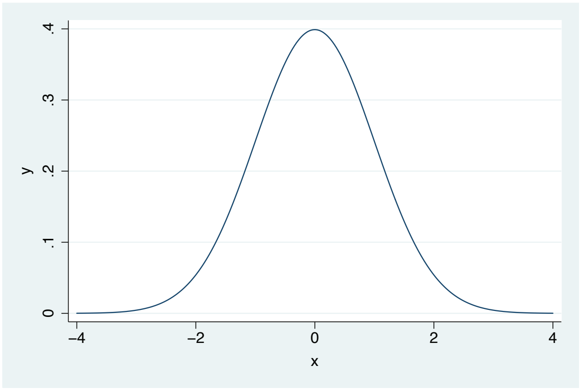
The twoway histogram command shows the distribution of one variable. It is often useful when overlaid with other twoway plots; otherwise, the histogram command would be preferable. See Twoway : Distribution (131) for more details.
Uses allstates.dta & scheme vg_s2c

```
twoway kdensity popk
```



The twoway kdensity command shows a kernel-density plot and is useful for examining the distribution of one variable. It can be overlaid with other twoway plots; otherwise, the kdensity command would be preferable. See Twoway : Distribution (131) for more details.
Uses allstates.dta & scheme vg_s2c

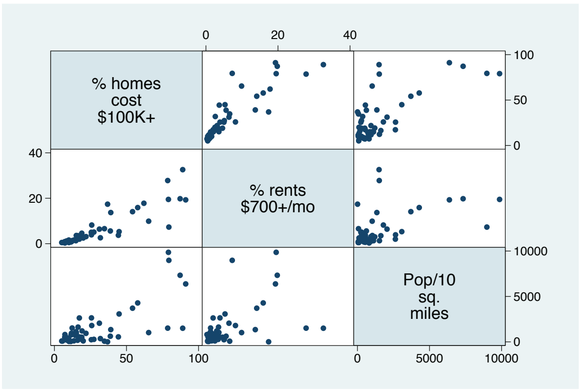
```
twoway function y=normden(x), range(-4 4)
```



The twoway function command allows an arbitrary function to be drawn over a range of specified values. See Twoway : Distribution (131) for more details.
Uses allstates.dta & scheme vg_s2c

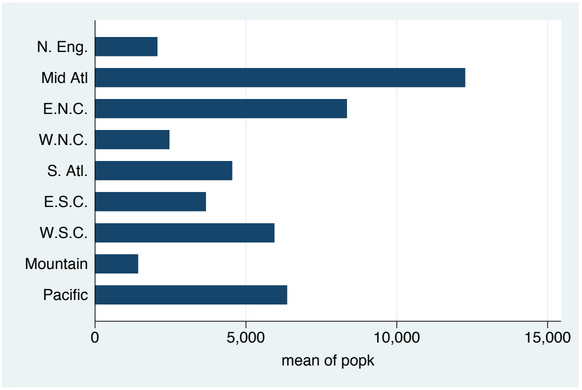
```
graph matrix propval100 rent700 popden
```

The graph matrix command shows a scatterplot matrix. See Matrix (153) for more details.
Uses allstates.dta & scheme vg_s2c



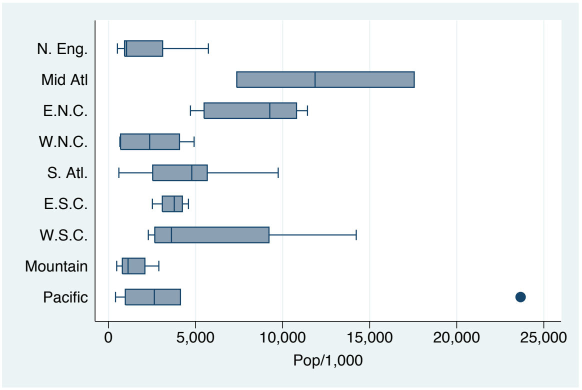
```
graph hbar popk, over(division)
```

This example shows how the graph hbar (horizontal bar) command is often used to show the values of a continuous variable broken down by one or more categorical variables. graph hbar is merely a rotated version of graph bar. See Bar (165) for more details.
Uses allstates.dta & scheme vg_s2c

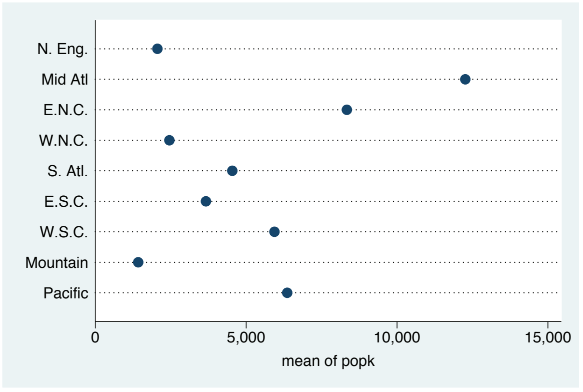


```
graph hbox popk, over(division)
```

Here is the previous graph as a box plot by using the graph hbox (horizontal box) command, which is commonly used for showing the distribution of one or more continuous variables, broken down by one or more categorical variables. graph hbox is merely a rotated version of graph box. See Box (219) for more details.
Uses allstates.dta & scheme vg_s2c

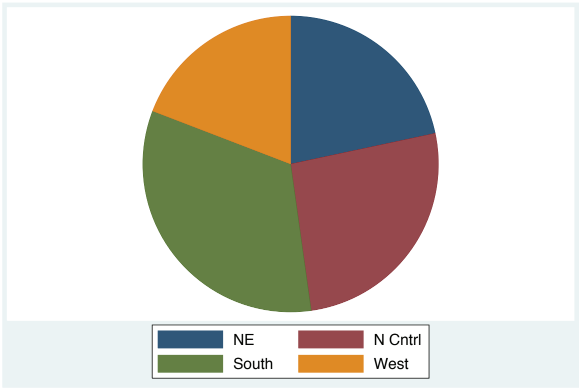


```
graph dot popk, over(division)
```



Here the previous plot is shown as a dot plot by using `graph dot`. Dot plots are often used to show one or more summary statistics for one or more continuous variables, broken down by one or more categorical variables. See Dot (255) for more details.
Uses allstates.dta & scheme vg_s2c

```
graph pie popk, over(region)
```



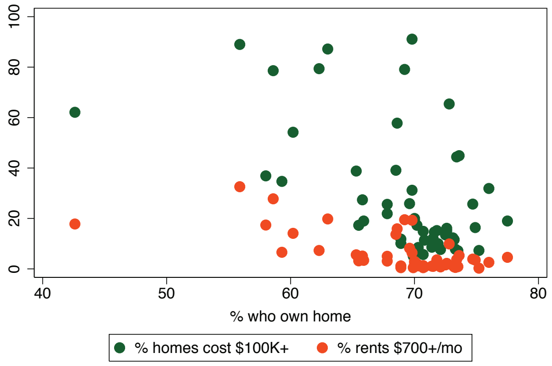
The `graph pie` command creates a pie chart. See Pie (281) for more details.
Uses allstates.dta & scheme vg_s2c

1.3 Schemes

Whereas the previous section was about the different types of graphs Stata can make, this section is about the different kinds of looks that you can have for Stata graphs. The basic starting point for the look of a graph is a scheme, which controls just about every aspect of the look of the graph. A scheme sets the stage for the graph, but you can use options to override the settings in a scheme. As you might surmise, if you choose (or develop) a scheme that produces graphs similar to the final graph you want to make, you can reduce the need to customize your graphs using options. This section gives you a basic idea of what schemes can do and introduces you to the schemes used throughout the book. See Intro : Using this book (1) for more details about how to select and use schemes and Appendix : Online supplements (456) for more information about how to download them.

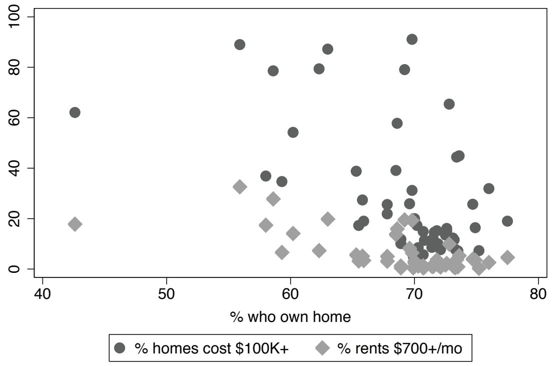
```
twoway scatter propval100 rent700 ownhome, scheme(vg_s1c)
```

This scatterplot illustrates the vg_s1c scheme. It is based on the s1color scheme but increases the sizes of elements in the graph to make them more readable. This scheme is in color and has a white background, both inside the plot region and in the surrounding area.
Uses allstates.dta & scheme vg_s1c



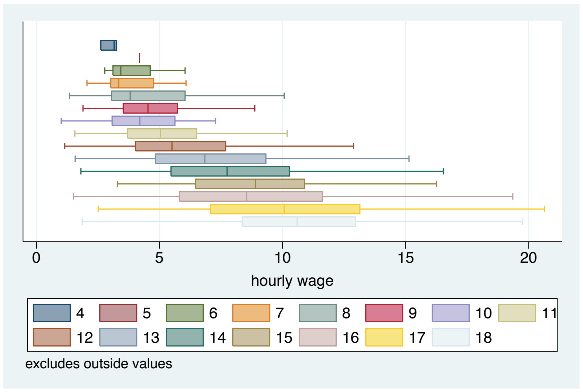
```
twoway scatter propval100 rent700 ownhome, scheme(vg_s1m)
```

This scatterplot is similar to the last one but uses the vg_s1m scheme, the monochrome equivalent of the vg_s1c scheme. It is based on the s1mono scheme but increases the sizes of elements in the graph to make them more readable. This scheme is in black and white and has a white background, both inside the plot region and in the surrounding area.
Uses allstates.dta & scheme vg_s1m

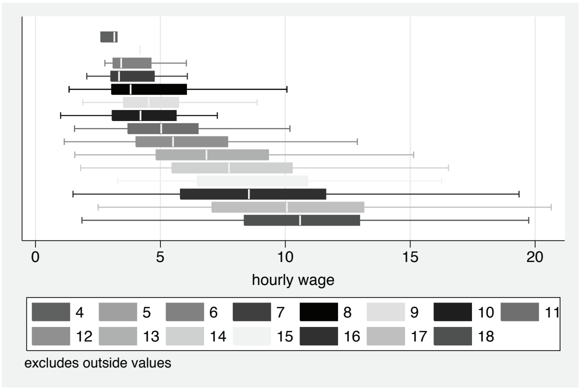


```
graph hbox wage, over(grade) asyvar nooutsides legend(rows(2))
    scheme(vg_s2c)
```

This box plot shows an example of the vg_s2c scheme. It is based on the s2color scheme but increases the sizes of elements in the graph to make them more readable. In this scheme, the plot region has a white background, but the surrounding area (the graph region) is light blue.
Uses nlsw.dta & scheme vg_s2c



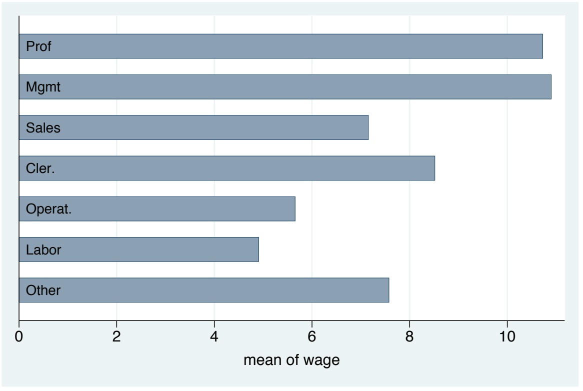
```
graph hbox wage, over(grade) asyvar nooutsides legend(rows(2))
    scheme(vg_s2m)
```



This box plot is similar to the previous one but uses the vg_s2m scheme, the monochrome equivalent of the vg_s2c scheme. This scheme is based on the s2mono scheme but increases the sizes of elements in the graph to make them more readable. This scheme is in black and white, and it has a white background in the plot region but is light gray in the surrounding graph region.

Uses nlsw.dta & scheme vg_s2m

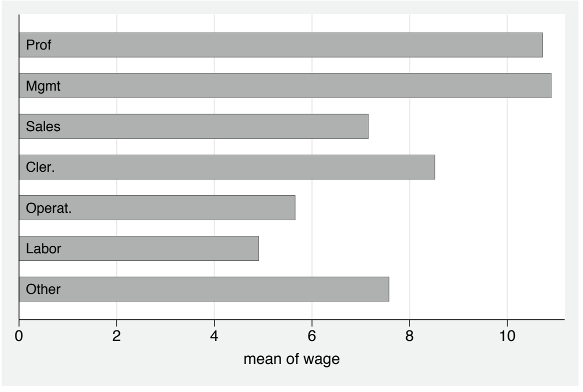
```
graph hbar wage, over(occ7, label(nolabels)) blabel(group, position(base))
    scheme(vg_palec)
```



This horizontal bar chart shows an example of the vg_palec scheme. It is based on the s2color scheme but makes the colors of the bars/boxes/markers paler by decreasing the intensity of the colors. As shown in this example, one use of this scheme is to make the colors of the bars pale enough to include text labels inside bars.

Uses nlsw.dta & scheme vg_palec

```
graph hbar wage, over(occ7, label(nolabels)) blabel(group, position(base))
    scheme(vg_palem)
```

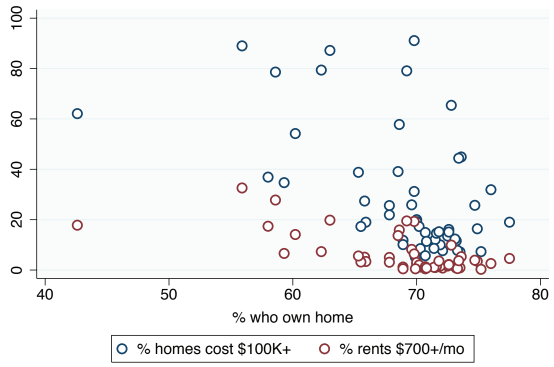


This example is the same as the last one but uses the vg_palem scheme, the monochrome equivalent of the vg_palec scheme. This scheme is based on the s2mono scheme but makes the colors of the bars/boxes/markers paler by decreasing the intensity of the colors.

Uses nlsw.dta & scheme vg_palem

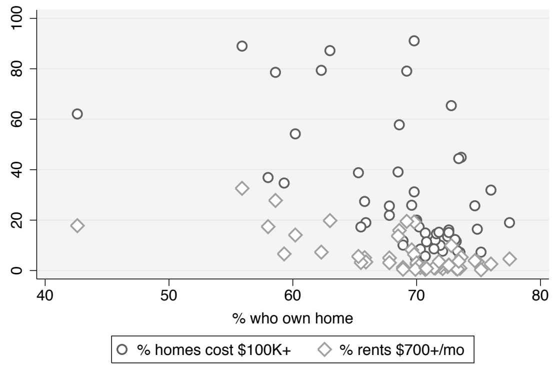
```
scatter propval100 rent700 ownhome, scheme(vg_outc)
```

This scatterplot illustrates the vg_outc scheme. It is based on the s2color scheme but makes the fill color of the bars/boxes/markers white, so they appear hollow. The plot region is a light blue to contrast with the white fill color. This scheme is useful to see the number of markers present where numerous markers are close or partially overlapping.
Uses allstates.dta & scheme vg_outc



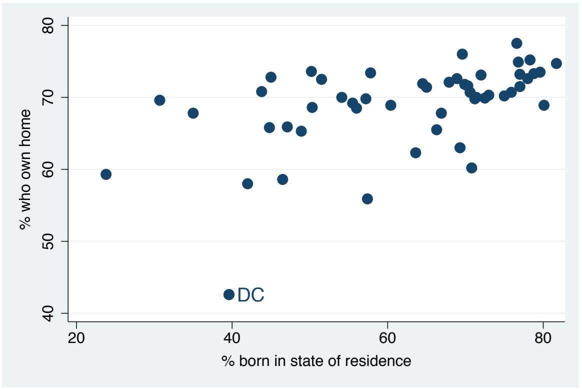
```
scatter propval100 rent700 ownhome, scheme(vg_outm)
```

This example is similar to the previous one but illustrates the vg_outm scheme, the monochrome equivalent of the vg_outc scheme. It is based on the s2mono scheme but makes the fill color of the bars/boxes/markers white, so they appear hollow.
Uses allstates.dta & scheme vg_outm

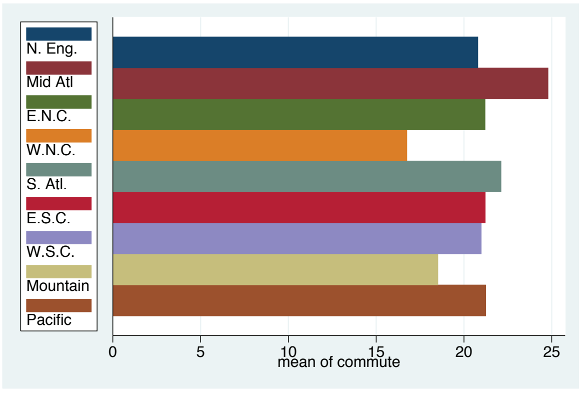


```
twoway (scatter ownhome borninstate if stateab=="DC", mlabel(stateab))
   (scatter ownhome borninstate), legend(off) scheme(vg_samec)
```

This is an example of the vg_samec scheme, which is based on the s2color scheme and makes all the markers, lines, bars, etc., the same color, shape, and pattern. Here the second scatter command labels Washington DC, which normally would be shown in a different color; with this scheme, the marker is the same. This scheme has a monochrome equivalent called vg_samem, which is not illustrated.
Uses allstates.dta & scheme vg_samec

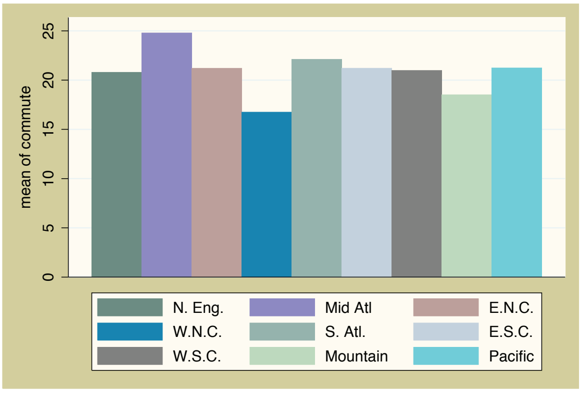


```
graph hbar commute, over(division) asyvar scheme(vg_lgndc)
```



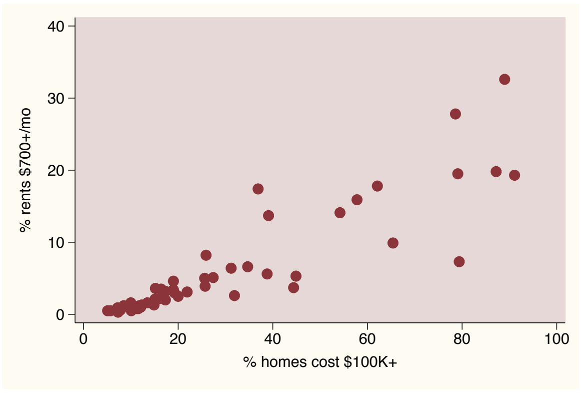
This horizontal bar chart shows an example of the `vg_lgndc` scheme. It is based on the `s2color` scheme but changes the default attributes of the legend, namely, showing the legend in one column to the left of the plot region, with the key and symbols placed atop each other. It can be efficient to place the legend to the left of the graph. This scheme has a monochrome equivalent called `vg_lgndm`, which is not illustrated here.
Uses allstates.dta & scheme vg_lgndc

```
graph bar commute, over(division) asyvar legend(rows(3)) scheme(vg_past)
```



This bar chart shows an example of the `vg_past` scheme. It is based on the `s2color` scheme but selects subdued pastel colors and provides a sand background for the surrounding graph region and an eggshell color for the inner plot region and legend area.
Uses allstates.dta & scheme vg_past

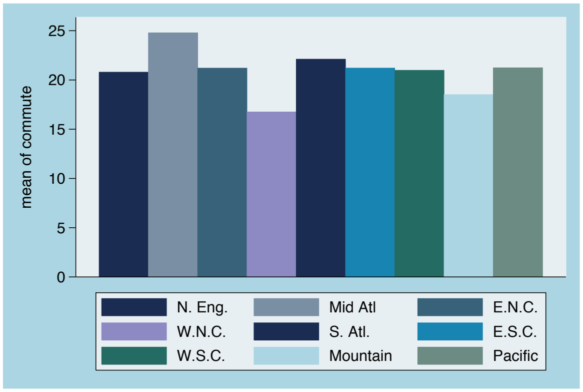
```
twoway scatter rent700 propval100, scheme(vg_rose)
```



This bar chart shows an example of the `vg_rose` scheme. It is based on the `s2color` scheme but uses a different set of colors for the background (eggshell) and for the plot area (a light rose color). By default, the grid lines are omitted and the labels for the y axis are horizontal.
Uses allstates.dta & scheme vg_rose

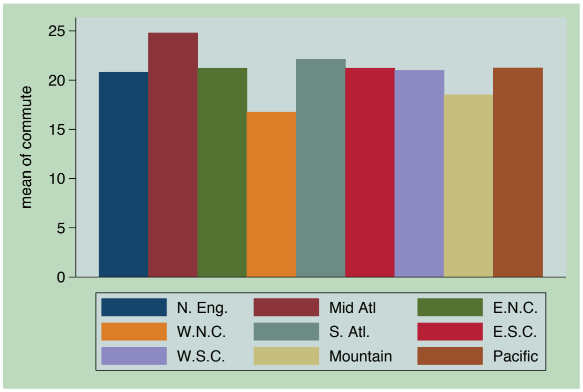
```
graph bar commute, over(division) asyvar legend(rows(3)) scheme(vg_blue)
```

This bar chart shows an example of the vg_blue scheme. It is based on the s2color scheme but uses a set of blue colors, with a light blue background and a light blue-gray color for the plot area. By default, the grid lines are omitted and the labels for the y axis are horizontal.
Uses allstates.dta & scheme vg_blue



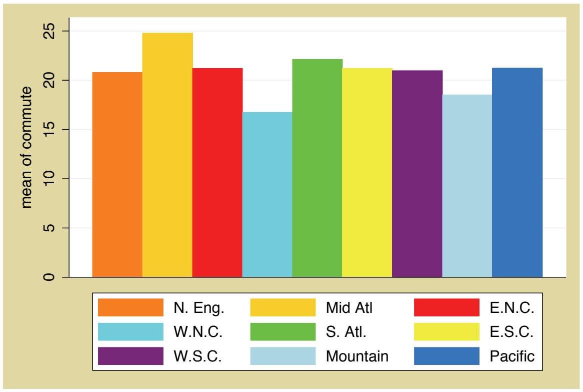
```
graph bar commute, over(division) asyvar legend(rows(3)) scheme(vg_teal)
```

This is an example of the vg_teal scheme. This scheme is also based on the s2color scheme but uses an olive-teal background. It also suppresses the display of grid lines and makes the labels for the y axis display horizontally by default.
Uses allstates.dta & scheme vg_teal



```
graph bar commute, over(division) asyvar legend(rows(3)) scheme(vg_brite)
```

This bar chart shows an example of the vg_brite scheme. It is based on the s2color scheme but selects a bright set of colors and changes the background to light khaki.
Uses allstates.dta & scheme vg_brite

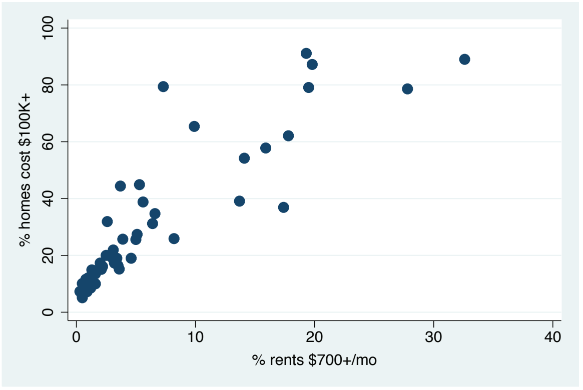


This section has just scratched the surface of all there is to know about schemes in Stata. I hope that it helps you see how schemes create a starting point for your graph and that, by choosing a scheme that is most similar to the look you want, you can save time and effort in customizing your graphs.

1.4 Options

Learning to create effective Stata graphs is ultimately about using options to customize the look of a graph until you are pleased with it. This section illustrates the general rules and syntax for Stata graph commands, starting with their basic structure and followed by illustrations showing how options work in the same way across different kinds of commands. Stata graph options work much like other options in Stata; however, there are more features that extend their power and functionality. Although these examples will use the `twoway scatter` command for illustration, most of the principles illustrated extend to all kinds of Stata graph commands. Although the emphasis of this section is on the use of Stata commands for customizing graphs, you could accomplish many of these customizations by using the Graph Editor; see Editor (33) for more information.

```
twoway scatter propval100 rent700
```

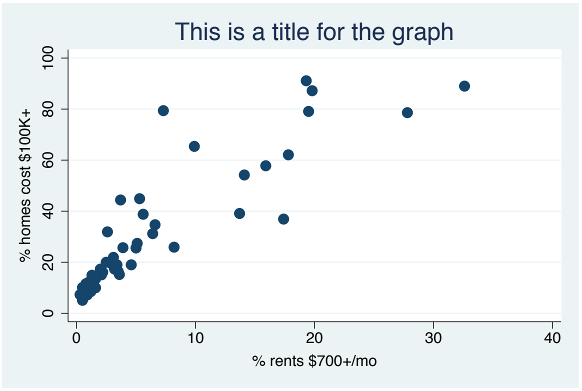


Consider this basic scatterplot. To add a title to this graph, we can use the `title()` option as illustrated in the next example.

Uses allstates.dta & scheme vg_s2c

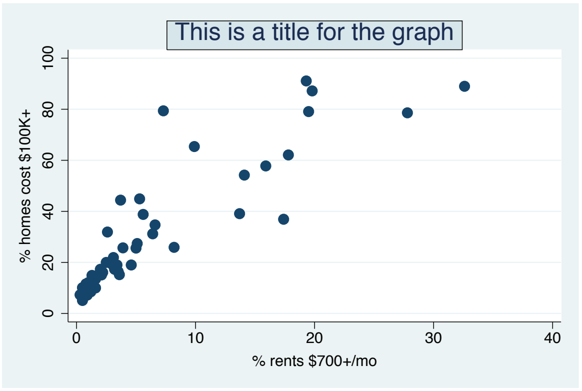
```
twoway scatter propval100 rent700,
   title("This is a title for the graph")
```

Just as with any Stata command, the `title()` option comes after a comma, and here it contains a quoted string that becomes the title of the graph.
Uses allstates.dta & scheme vg_s2c



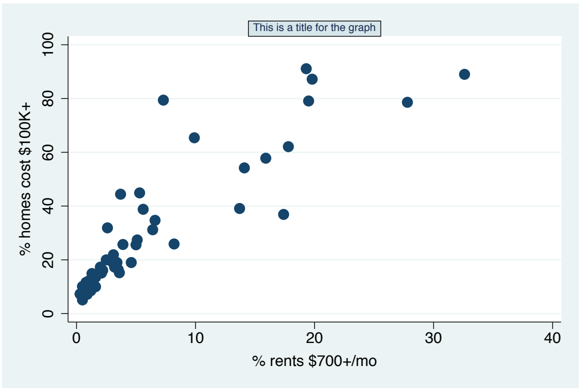
```
twoway scatter propval100 rent700,
   title("This is a title for the graph", box)
```

In this example, we add `box` as an option within `title()` to place a box around the title. If the default for the current scheme had included a box, then you could have used the `nobox` option to suppress it.
Uses allstates.dta & scheme vg_s2c

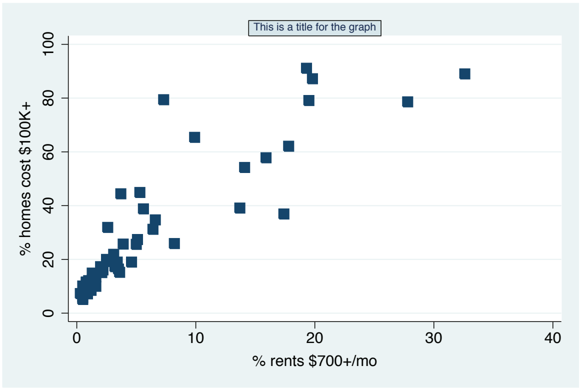


```
twoway scatter propval100 rent700,
   title("This is a title for the graph", box size(small))
```

Let's take the last graph and modify the title to make it small. We add `size(small)` to the `title()` option to change the title's size, as well as the box size, to small.
Uses allstates.dta & scheme vg_s2c

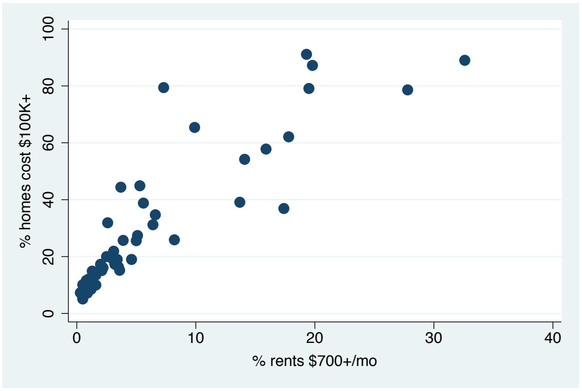


```
twoway scatter propval100 rent700,
   title("This is a title for the graph", box size(small))
   msymbol(S)
```



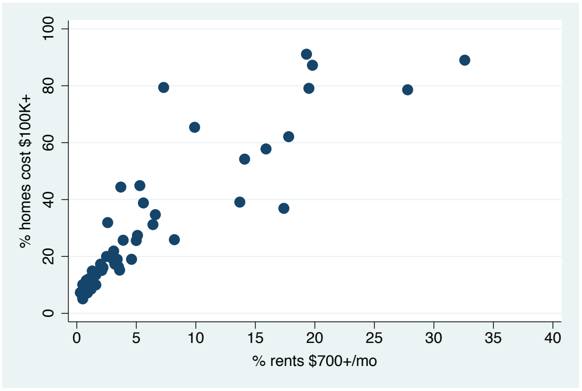
Say that we want the symbols to be displayed as squares. We add another option, `msymbol(S)`, to indicate that we want the marker symbol to be displayed as a square (`S` for square). Adding one option at a time is a common way to build a Stata graph. In the next graph, we will change gears and start building a new graph to show other aspects of options.
Uses allstates.dta & scheme vg_s2c

```
twoway scatter propval100 rent700
```



Let's return to this simple scatterplot. Say that we want the labels for the x axis to change from 0 10 20 30 40 to 0 5 10 15 20 25 30 35 40.
Uses allstates.dta & scheme vg_s2c

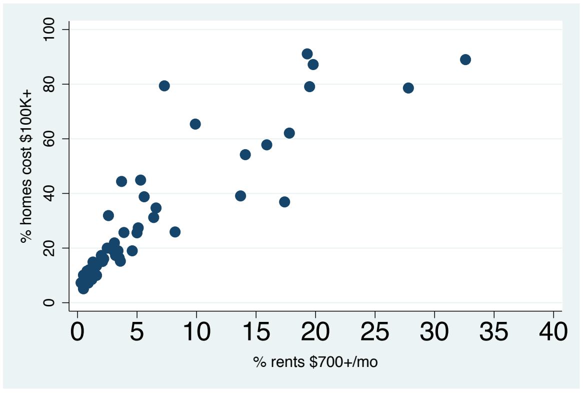
```
twoway scatter propval100 rent700, xlabel(0(5)40)
```



Here we add the `xlabel()` option to label the x axis from 0 to 40, incrementing by 5. But say that we want the labels to be displayed larger.
Uses allstates.dta & scheme vg_s2c

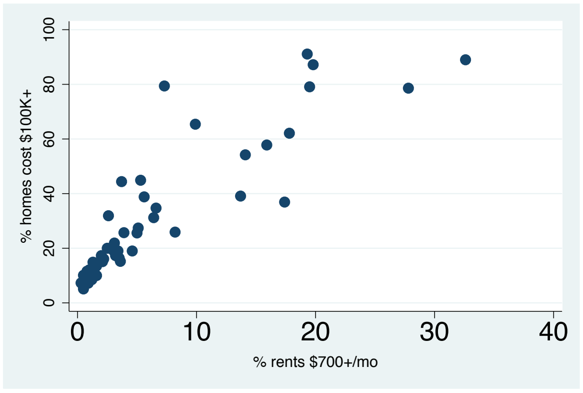
```
twoway scatter propval100 rent700, xlabel(0(5)40, labsize(huge))
```

Here we add the `labsize()` (label size) option to increase the size of the labels for the x axis. Now say that we were happy with the original numbering (0 10 20 30 40) but wanted the labels to be huge.
Uses allstates.dta & scheme vg_s2c



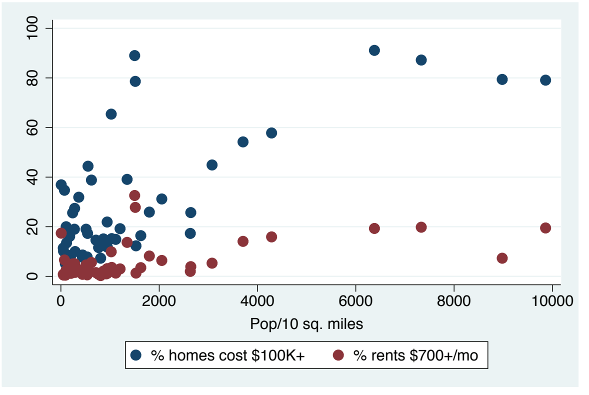
```
twoway scatter propval100 rent700, xlabel(, labsize(huge))
```

The `xlabel()` option we use here indicates that we are content with the numbers chosen for the label of the x axis because we have nothing before the comma. After the comma, we add the `labsize()` option to increase the size of the labels for the x axis.
Uses allstates.dta & scheme vg_s2c



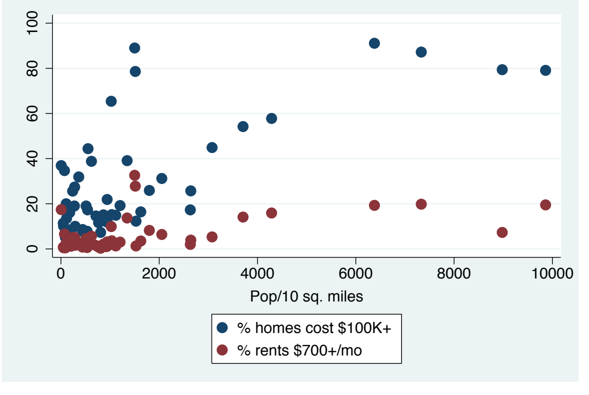
Now let's consider some examples using the `legend()` option to show that some options do not require or permit the use of commas within them. Also, the following examples show where we might properly specify an option repeatedly.

```
twoway scatter propval100 rent700 popden
```



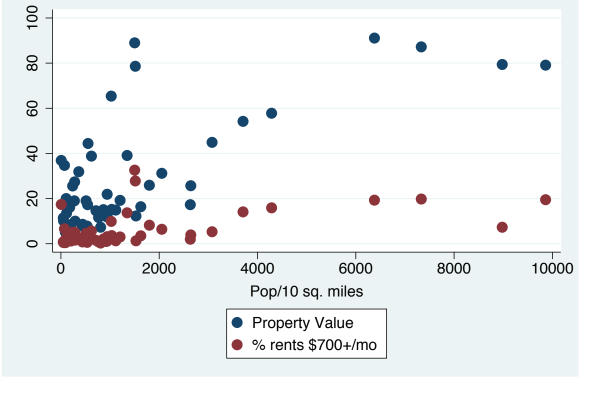
Here we show two y variables, `propval100` and `rent700`, graphed against population density, `popden`. Stata has created a legend, helping us see which symbols correspond to which variables. We can use the `legend()` option to customize the legend.
Uses allstates.dta & scheme vg_s2c

```
twoway scatter propval100 rent700 popden, legend(cols(1))
```



By using the `legend(cols(1))` option, we make the legend display in one column. We did not use a comma because, with the `legend()` option, there is no natural default argument. If we had included a comma within the `legend()` option, Stata would have reported this as an error.
Uses allstates.dta & scheme vg_s2c

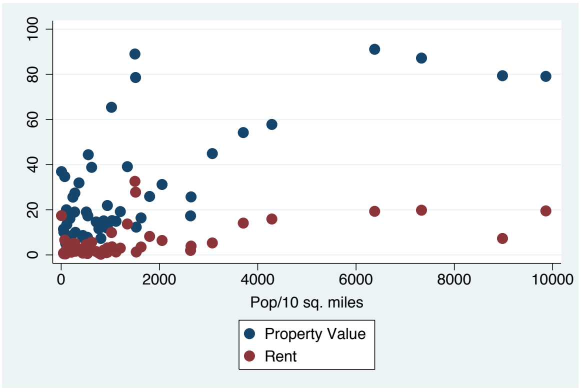
```
twoway scatter propval100 rent700 popden,
   legend(cols(1) label(1 "Property Value"))
```



This example adds another option, `label()`, within the `legend()` option to change the label for the first variable.
Uses allstates.dta & scheme vg_s2c

```
twoway scatter propval100 rent700 popden,
   legend(cols(1) label(1 "Property Value") label(2 "Rent"))
```

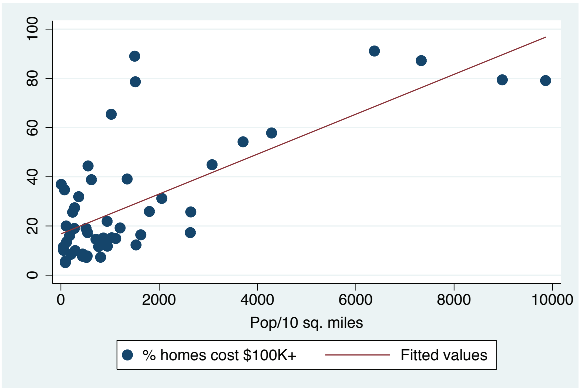
Here we add another `label()` option within the `legend()` option; this option changes the label for the second variable. We can use the `label()` option repeatedly to change the labels for the different variables.
Uses allstates.dta & scheme vg_s2c



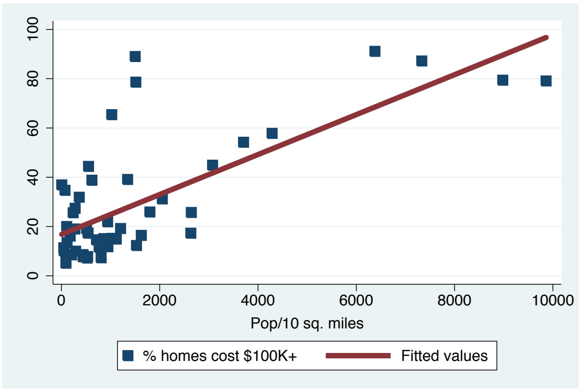
Finally, let's consider an example that shows how to use the `twoway` command to overlay two plots. The following examples show how each graph can have its own options and how options can apply to the overall graph.

```
twoway (scatter propval100 popden)
   (lfit propval100 popden)
```

This graph shows a scatterplot predicting property value from population density and shows a linear fit between these two variables. Say that we wanted to change the symbol displayed in the scatterplot and the thickness of the line for the linear fit.
Uses allstates.dta & scheme vg_s2c

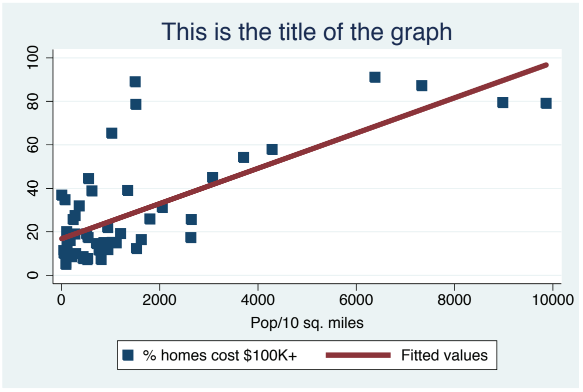


```
twoway (scatter propval100 popden, msymbol(S))
   (lfit propval100 popden, lwidth(vthick))
```



We add the `msymbol()` option to the `scatter` command to change the symbol to a square, and we add the `lwidth()` (line width) option to the `lfit` command to make the line very thick. When we overlay two plots, each plot can have its own options that operate on its respective parts of the graph. However, some parts of the graph are shared, for example, the title.
Uses allstates.dta & scheme vg_s2c

```
twoway (scatter propval100 popden, msymbol(S))
   (lfit propval100 popden, lwidth(vthick)),
   title("This is the title of the graph")
```

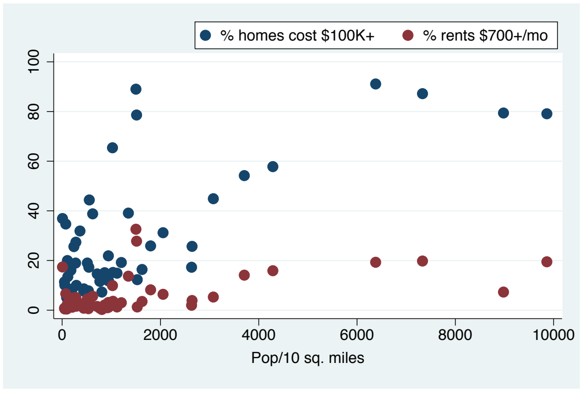


We add the `title()` option to the end of the command, placed after a comma. That final comma signals that options concerning the overall graph are to follow, here, the `title()` option.
Uses allstates.dta & scheme vg_s2c

One of the beauties of Stata graph commands is the way that different graph commands share common options. If you want to customize the display of a legend, you do it by using the same options, whether you are using a bar graph, a box plot, a scatterplot, or any other kind of Stata graph. Once you learn how to control legends with one type of graph, you have learned how to control legends for all types of graphs. This is illustrated with a couple of examples.

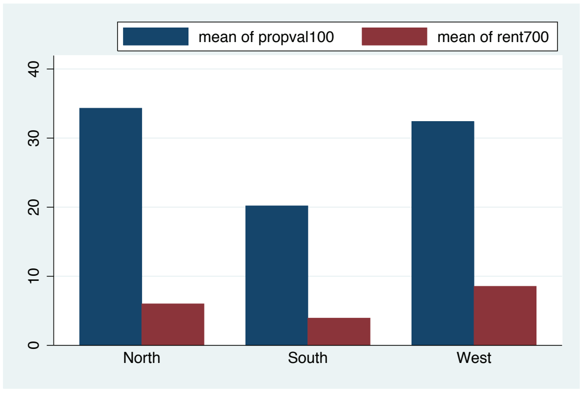
```
twoway scatter propval100 rent700 popden, legend(position(1))
```

Consider this scatterplot. We add the `legend()` option to make the legend display in the one o'clock position on the graph, putting the legend in the top right corner.
Uses allstates.dta & scheme vg_s2c



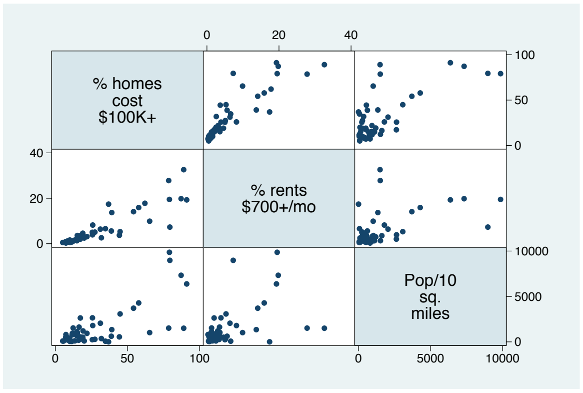
```
graph bar propval100 rent700, over(nsw) legend(position(1))
```

Here we use the **`graph bar`** command, which is a completely different command from the previous one. Even though the graphs are different, the `legend()` option we supply is the same and has the same effect. Many (but not all) options function in this way, sharing a common syntax and having common effects.
Uses allstates.dta & scheme vg_s2c



```
graph matrix propval100 rent700 popden, legend(position(1))
```

Compare this example with the previous two. The **`graph matrix`** command does not support the `legend()` option because this graph does not need or produce legends. In the Matrix (153) chapter, for example, there are no references to legends, an indication that this is not a relevant option for this kind of graph. Even though we included this irrelevant option, Stata ignored it and produced an appropriate graph anyway.
Uses allstates.dta & scheme vg_s2c



Because legends work the same way with different types of Stata graph commands, only one place discusses legends in detail: Options : Legend (353). However, it is useful to see examples of legends for each type of graph that uses them. Each chapter, therefore, includes a brief section describing legends for each type of graph discussed in that chapter. Likewise, Options (299) describes most options in detail, with a brief section in every chapter discussing how each option works for that specific type of graph. As shown for legends, some options are not appropriate for some types of graphs, so those options are not discussed with the commands that do not support them.

Although you can use an option like `legend()` with many, but not all, kinds of Stata graph commands, you can use other kinds of options with almost every kind of Stata graph. These are called *standard options*. To help you differentiate these kinds of options, they are discussed in their own chapter, Standard options (381). Since these options can be used with most types of graph commands, they are generally not discussed in the chapters about the different types of graphs, except when their usage interacts with the options illustrated. For example, `subtitle()` is a standard option, but its behavior takes on a special meaning when used with the `legend()` option, so the `subtitle()` option is discussed in the context of legends. Consistent with what was previously shown, the syntax of standard options follows the same kinds of rules that have been illustrated, and their usage and behavior are uniform across the many types of Stata graph commands.

1.5 Building graphs

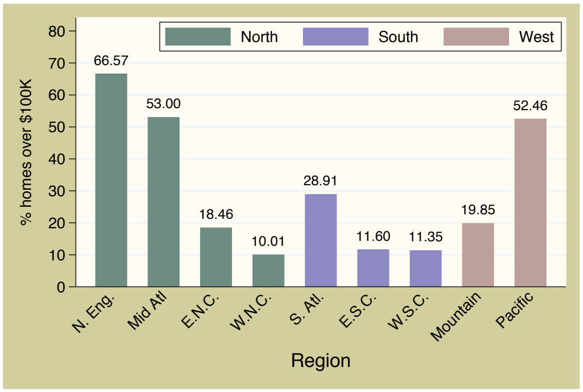
I have three agendas in writing this section. First, I wish to show the process of building complex graphs a little bit at a time. At the same time, I illustrate how to use the resources of this book to get the bits of information needed to build these graphs. Finally, I hope to show that, even though a complete Stata graph command might look complicated and overwhelming, the process of building the graph slowly is actually straightforward and logical.

Let's first build a bar chart that looks at property values broken down by region of the country. Then we will modify the legend and bar characteristics, add titles, and so forth.

Although this section emphasizes the process of building and customizing graphs with Stata commands, you could accomplish many of these customizations by using the Graph Editor; see Editor (33) for more information.

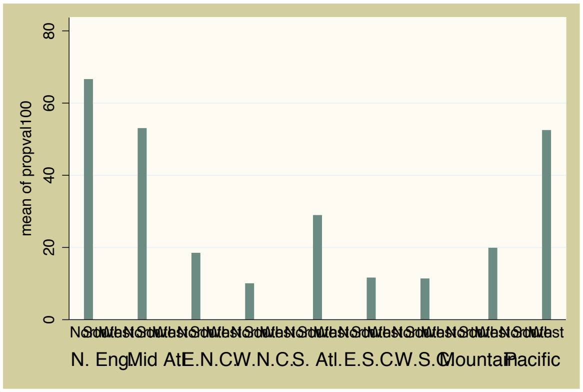
```
graph display
```

Say that we want to create this graph. For now, the syntax is concealed, just showing the `graph display` command to show the previously drawn graph. It might be overwhelming at first to determine all the options needed to make this graph. To ease our task, we will build it a bit at a time, refining the graph and fixing any problems we find.
Uses allstates.dta & scheme vg_past



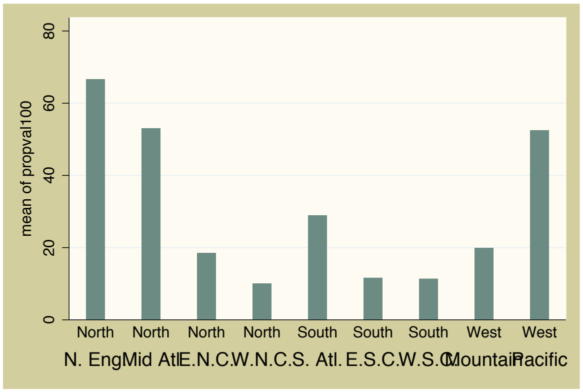
```
graph bar propval100, over(nsw) over(division)
```

We begin by seeing that this is a bar chart and look at Bar : Y-variables (165) and Bar : Over (169). We take our first step toward making this graph by making a bar chart showing `propval100` and adding `over(nsw)` and `over(division)` to break down the means by `nsw` and `division`.
Uses allstates.dta & scheme vg_past

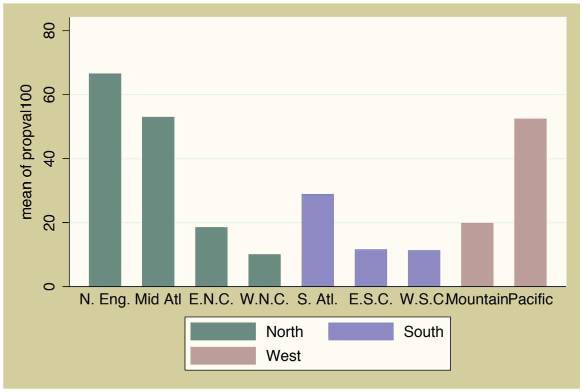


```
graph bar propval100, over(nsw) over(division) nofill
```

The previous graph is not quite what we want because we see every `division` shown with every `nsw`, but for example, the Pacific region only appears in the West. In Bar : Over (169), we see that we can add the `nofill` option to show only the combinations of `nsw` and `division` that exist in the data file. Next we will look at the colors of the bars.
Uses allstates.dta & scheme vg_past

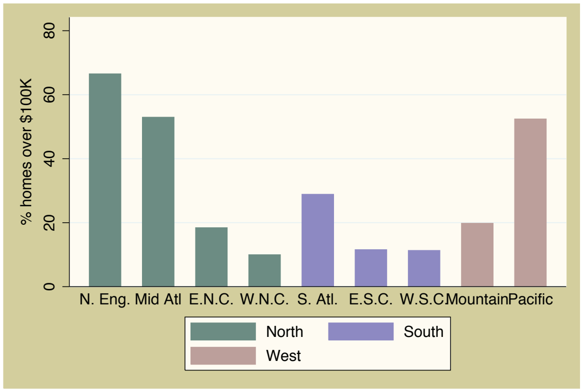


```
graph bar propval100, over(nsw) over(division) nofill asyvars
```



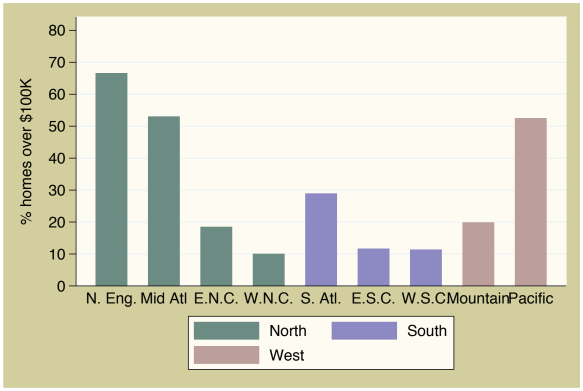
The last graph is getting closer, but we want the bars for North, South, and West displayed in different colors and labeled with a legend. In Bar : Y-variables (165), we see that the asyvars option will accomplish this. Next we will change the title for the y axis.
Uses allstates.dta & scheme vg_past

```
graph bar propval100, over(nsw) over(division) nofill asyvars
   ytitle("% homes over $100K")
```



Now we want to put a title on the y axis. In Bar : Y-axis (205), we see examples illustrating the use of ytitle() for putting a title on the y axis. Here we put a title on the y axis, but now we want to change the labels for the y axis to go from 0 to 80, incrementing by 10.
Uses allstates.dta & scheme vg_past

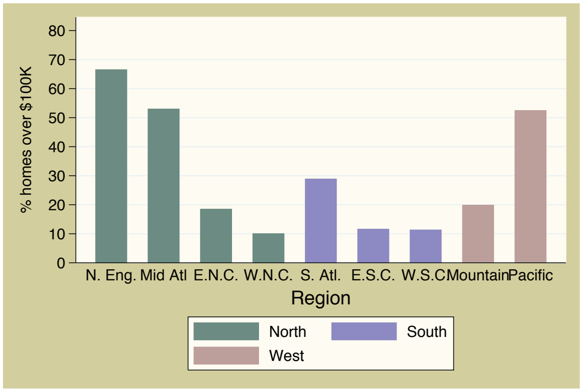
```
graph bar propval100, over(nsw) over(division) nofill asyvars
   ytitle("% homes over $100K") ylabel(0(10)80, angle(0))
```



The Bar : Y-axis (205) section also tells us about the ylabel() option. In addition to changing the labels, we want to change the angle of the labels. From the Bar : Y-axis (205) section, we see that we can use the angle() option to change the angle of the labels. Now that we have the y axis labeled as we want, let's next look at the title for the x axis.
Uses allstates.dta & scheme vg_past

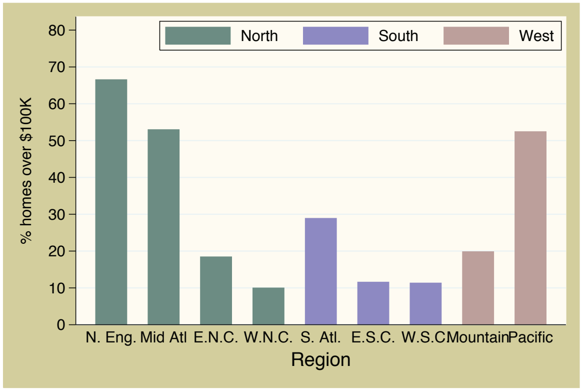
```
graph bar propval100, over(nsw) over(division) nofill asyvars
   ytitle("% homes over $100K") ylabel(0(10)80, angle(0)) b1title(Region)
```

After having used the `ytitle()` option to label the y axis, we might be tempted to use the `xtitle()` option to label the x axis, but this axis is a categorical variable. In Bar : Cat axis (185), we see that this axis is treated differently because of that. To put a title below the graph, we use the `b1title()` option. Now let's turn our attention to formatting the legend.
Uses allstates.dta & scheme vg_past



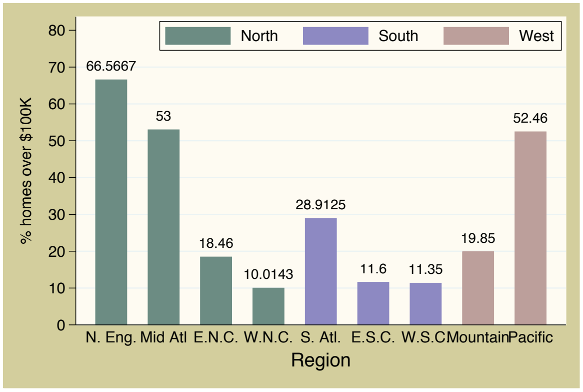
```
graph bar propval100, over(nsw) over(division) nofill asyvars
   ytitle("% homes over $100K") ylabel(0(10)80, angle(0)) b1title(Region)
   legend(rows(1) position(1) ring(0))
```

Here we want to use the `legend()` option to make the legend have one row in the top right corner within the plot area. In Bar : Legend (193), we see that the `rows(1)` option makes the legend appear in one row and that the `position(1)` option puts the legend in the one o'clock position. The `ring(0)` option puts the legend inside the plot region. Next let's label the bars.
Uses allstates.dta & scheme vg_past

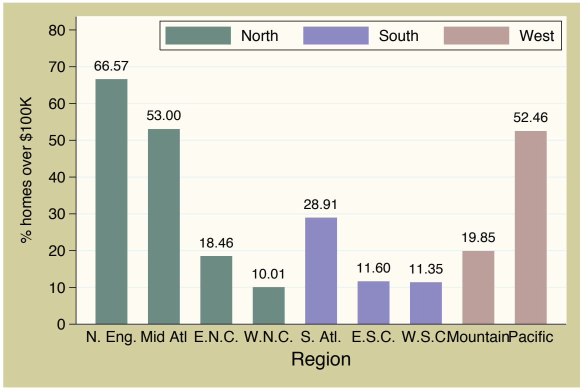


```
graph bar propval100, over(nsw) over(division) nofill asyvars
   ytitle("% homes over $100K") ylabel(0(10)80, angle(0)) b1title(Region)
   legend(rows(1) position(1) ring(0)) blabel(bar)
```

We want each bar labeled with the height of the bar. Bar : Legend (193) shows how we can do this by using the `blabel()` (bar label) option to label the bars in lieu of legends. `blabel(bar)` labels the bars with their height.
Uses allstates.dta & scheme vg_past

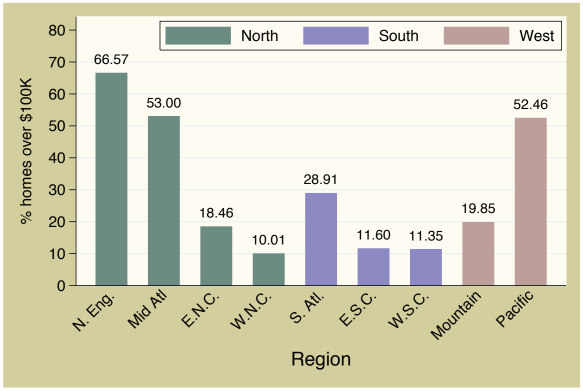


```
graph bar propval100, over(nsw) over(division) nofill asyvars
   ytitle("% homes over $100K") ylabel(0(10)80, angle(0)) b1title(Region)
   legend(rows(1) position(1) ring(0)) blabel(bar, format(%4.2f))
```



We want the label for each bar to end in two decimal places, and we see in Bar : Legend (193) that we can use the format() option to format these numbers as we want.
Uses allstates.dta & scheme vg_past

```
graph bar propval100, over(nsw) over(division, label(angle(45))) nofill
   ytitle("% homes over $100K") ylabel(0(10)80, angle(0)) b1title(Region)
   legend(rows(1) position(1) ring(0)) blabel(bar, format(%4.2f)) asyvars
```



Finally, in Bar : Cat axis (185), we see that we can add the label(angle(45)) option to the over() option to specify that labels for that variable be shown at a 45-degree angle so they do not overlap each other.
Uses allstates.dta & scheme vg_past

I hope this section has shown that it is not that difficult to create complex graphs by building them one step at a time. You can use the resources in this book to seek out each piece of information you need and then put those pieces together the way you want to create your own graphs. For more information about how to integrate options to create complex Stata graphs, see Appendix : More examples (440).