Title

stata.com

axis_label_options — Options for specifying axis labels

Description

axis_label_options control the placement and the look of ticks and labels on an axis.

Quick start

Use about 5 automatically chosen ticks and labels on the y axis

graph_command ..., ... `ylabel(#5)`

Use about 10 automatically chosen ticks and labels on the x axis

graph_command ..., ... `xlabel(#10)`

Place x axis ticks and labels at 10, 20, 30, 40, and 50

graph_command ..., ... `xlabel(10 20 30 40 50)`

Same as above

graph_command ..., ... `xlabel(10(10)50)`

Place ticks and labels for the y axis only at the minimum and maximum values of the y variables

graph_command ..., ... `ylabel(minmax)`

Place x axis ticks at 10, 20, and 30 and label those ticks "ten", "twenty", and "thirty"

graph_command ..., ... `xlabel(10 "ten" 20 "twenty" 30 "thirty")`

Place ticks and date labels at each month from 1999m1 to 2000m6 on the x axis

graph_command ..., ... `tlabel(1999m1(1)2000m6)`

Add a tick and the label "Special value" at 12.4 on the x axis

graph_command ..., ... `xlabel(12.4 "Special value", add)`

As above, but print "Special value" in red

```
graph_command ..., ... xlabel(12.4 "Special value", ///
    add custom labcolor(red))
```

Place ticks and date labels every 7 days from 01jan2010 to 11feb2010

graph_command ..., ... `tlabel(01jan2010(7)11feb2010)`

As above, using a custom date format

```
graph_command ..., ... tlabel(01jan2010(7)11feb2010, ///
    format("%tdMon_DD")
```

Add 9 unlabeled minor ticks between each major tick on the y axis

graph_command ..., ... `ymtick(#9)`

Make y labels horizontal

graph_command ..., ... `ylabel(, angle(0))`

Render x labels at a 45-degree angle

graph_command ..., ... `xlabel(, angle(45))`

Alternate the placement of labels to increase space between adjacent labels

graph_command ..., ... `xlabel(, alternate)`

Use large font to render x labels

graph_command ..., ... `xlabel(, labsize(large))`

Add grid lines from the major ticks on the x axis

graph_command ..., ... `xlabel(, grid)`

Label x axis ticks at 1, 2, 3, and 4 with the value labels of `x`

graph_command ..., ... `xlabel(1 2 3 4, valuelabel)`

Syntax

axis_label_options are a subset of *axis_options*; see [G-3] ***axis_options***. *axis_label_options* control the placement and the look of ticks and labels on an axis.

axis_label_options	Description
{ y | x | t | z } `label(`*rule_or_values*`)`	major ticks plus labels
{ y | x | t | z } `tick(`*rule_or_values*`)`	major ticks only
{ y | x | t | z } `mlabel(`*rule_or_values*`)`	minor ticks plus labels
{ y | x | t | z } `mtick(`*rule_or_values*`)`	minor ticks only

The above options are *merged-explicit*; see [G-4] **concept: repeated options**.

where *rule_or_values* is defined as

[*rule*] [*numlist* [`"`*label*`"` [*numlist* [`"`*label*`"` [...]]]]] [`,` *suboptions*]

Either *rule* or *numlist* must be specified, and both may be specified.

rule	Example	Description
`##`	`#6`	approximately 6 nice values
`###`	`##10`	$10 - 1 = 9$ values between major ticks; allowed with `mlabel()` and `mtick()` only
`#(#)#`	`-4(.5)3`	specified range: -4 to 3 in steps of .5
`minmax`	`minmax`	minimum and maximum values
`none`	`none`	label no values
`.`	`.`	skip the rule

where *numlist* is as described in [U] **11.1.8 numlist**.

`tlabel()`, `ttick()`, `tmlabel()`, and `tmtick()` also accept a *datelist* and an extra type of *rule*

rule	Example	Description
date`(#)`*date*	`1999m1(1)1999m12`	specified date range: each month assuming the axis has the `%tm` format

where *date* and *datelist* may contain dates, provided that the t (time) axis has a date format; see [U] **11.1.9 datelist**.

suboptions	Description
`axis(#)`	which axis, $1 \leq \# \leq 9$
`add`	combine options
`[no]ticks`	suppress ticks
`[no]labels`	suppress labels
`valuelabel`	label values using first variable's value label
`format(`*%fmt*`)`	format values per *%fmt*
`angle(`*anglestyle*`)`	angle the labels
`alternate`	offset adjacent labels
`norescale`	do not rescale the axis
`tstyle(`*tickstyle*`)`	labels and ticks: overall style
`labgap(`*relativesize*`)`	labels: margin between tick and label
`labstyle(`*textstyle*`)`	labels: overall style
`labsize(`*textsizestyle*`)`	labels: size of text
`labcolor(`*colorstyle*`)`	labels: color and opacity of text
`tlength(`*relativesize*`)`	ticks: length
`tposition(outside|crossing|inside)`	ticks: position/direction
`tlstyle(`*linestyle*`)`	ticks: linestyle of
`tlwidth(`*linewidthstyle*`)`	ticks: thickness of line
`tlcolor(`*colorstyle*`)`	ticks: color and opacity of line
`custom`	tick- and label-rendition options apply only to these labels
`[no]grid`	grid: include
`[no]gmin`	grid: grid line at minimum
`[no]gmax`	grid: grid line at maximum
`gstyle(`*gridstyle*`)`	grid: overall style
`[no]gextend`	grid: extend into plot region margin
`glstyle(`*linestyle*`)`	grid: linestyle of
`glwidth(`*linewidthstyle*`)`	grid: thickness of line
`glcolor(`*colorstyle*`)`	grid: color and opacity of line
`glpattern(`*linepatternstyle*`)`	grid: line pattern of line

Options

`ylabel(`*rule_or_values*`)`, `xlabel(`*rule_or_values*`)`, `tlabel(`*rule_or_values*`)`, and `zlabel(`*rule_or_values*`)` specify the major values to be labeled and ticked along the axis. For instance, to label the values 0, 5, 10, ..., 25 along the x axis, specify `xlabel(0(5)25)`. If the t axis has the `%tm` format, `tlabel(1999m1(1)1999m12)` will label all the months in 1999.

`ytick(`*rule_or_values*`)`, `xtick(`*rule_or_values*`)`, `ttick(`*rule_or_values*`)`, and `ztick(`*rule_or_values*`)` specify the major values to be ticked but not labeled along the axis. For instance, to tick the values 0, 5, 10, ..., 25 along the x axis, specify `xtick(0(5)25)`. Specify `ttick(1999m1(1)1999m12)` to place ticks for each month in the year 1999.

`ymlabel(`*rule_or_values*`)`, `xmlabel(`*rule_or_values*`)`, `tmlabel(`*rule_or_values*`)`, and `zmlabel(`*rule_or_values*`)` specify minor values to be labeled and ticked along the axis.

`ymtick(`*rule_or_values*`)`, `xmtick(`*rule_or_values*`)`, `tmtick(`*rule_or_values*`)`, and `zmtick(`*rule_or_values*`)` specify minor values to be ticked along the axis.

`zlabel(`*rule_or_values*`)`, `ztick(`*rule_or_values*`)`, `zmlabel(`*rule_or_values*`)`, and `zmtick(`*rule_or_values*`)`; see *Contour axes—zlabel(), etc.* below.

Suboptions

`axis(#)` specifies to which scale this axis belongs and is specified when dealing with multiple x (t) or y axes; see [G-3] ***axis_choice_options***.

`add` specifies what is to be added to any `xlabel()`, `ylabel()`, `xtick()`, ..., or `ymtick()` option previously specified. Labels or ticks are added to any default labels or ticks or to any labels or ticks specified in previous `xlabel()`, `ylabel()`, `xtick()`, ..., or `ymtick()` options. Only value specifications are added; rule specifications always replace any existing rule. See *Interpretation of repeated options* below.

`noticks` and `ticks` suppress/force the drawing of ticks. `ticks` is the usual default, so `noticks` makes { `y` | `x` }`label()` and { `y` | `x` }`mlabel()` display the labels only.

`nolabels` and `labels` suppress/force the display of the labels. `labels` is the usual default, so `nolabels` turns { `y` | `x` }`label()` into { `y` | `x` }`tick()` and { `y` | `x` }`mlabel()` into { `y` | `x` }`mtick()`. Why anyone would want to do this is difficult to imagine.

`valuelabel` specifies that values should be mapped through the first y variable's value label (`y*()` options) or the x variable's value label (`x*()` options). Consider the command `scatter yvar xvar` and assume that `xvar` has been previously given a value label:

```
. label define cat 1 "Low" 2 "Med" 3 "Hi"
. label values xvar cat
```

Then

```
. scatter yvar xvar, xlabel(1 2 3, valuelabel)
```

would, rather than putting the numbers 1, 2, and 3, put the words Low, Med, and Hi on the x axis. It would have the same effect as

```
. scatter yvar xvar, xlabel(1 "Low" 2 "Med" 3 "Hi")
```

`format(%`*fmt*`)` specifies how numeric values on the axes should be formatted. The default `format()` is obtained from the variables specified with the `graph` command, which for `ylabel()`, `ytick()`, `ymlabel()`, and `ymtick()` usually means the first y variable, and for `xlabel()`, ..., `xmtick()`, means the x variable. For instance, in

```
. scatter y1var y2var xvar
```

the default format for the y axis would be `y1var`'s format, and the default for the x axis would be `xvar`'s format.

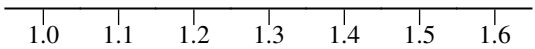
You may specify the `format()` suboption (or any suboption) without specifying values if you want the default labeling presented differently. For instance,

```
. scatter y1var y2var xvar, ylabel(,format(%9.2fc))
```

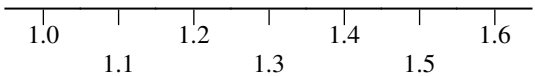
would present default labeling of the y axis, but the numbers would be formatted with the `%9.2fc` format. Note carefully the comma in front of `format`. Inside the `ylabel()` option, we are specifying suboptions only.

`angle(`*anglestyle*`)` causes the labels to be presented at an angle. See [G-4] ***anglestyle***.

`alternate` causes adjacent labels to be offset from one another and is useful when many values are being labeled. For instance, rather than obtaining



with `alternate`, you would obtain



`norescale` specifies that the ticks or labels in the option be placed directly on the graph without rescaling the axis or associated plot region for the new values. By default, label options automatically rescale the axis and plot region to include the range of values in the new labels or ticks. `norescale` allows you to plot ticks or labels outside the normal bounds of an axis.

`tstyle(`*tickstyle*`)` specifies the overall look of ticks and labels; see [G-4] ***tickstyle***. The options documented below will allow you to change each attribute of a tick and its label, but the *tickstyle* specifies the starting point.

You need not specify `tstyle()` just because there is something you want to change about the look of ticks. You specify `tstyle()` when another style exists that is exactly what you desire or when another style would allow you to specify fewer changes to obtain what you want.

`labgap(`*relativesize*`)`, `labstyle(`*textstyle*`)`, `labsize(`*textsizestyle*`)`, and `labcolor(`*colorstyle*`)` specify details about how the labels are presented. See [G-4] ***relativesize***, [G-4] ***textstyle***, [G-4] ***textsizestyle***, and [G-4] ***colorstyle***.

`tlength(`*relativesize*`)` specifies the overall length of the ticks; see [G-4] ***relativesize***.

`tposition(outside`|`crossing`|`inside)` specifies whether the ticks are to extend `outside` (from the axis out, the usual default), `crossing` (crossing the axis line, extending in and out), or `inside` (from the axis into the plot region).

`tlstyle(`*linestyle*`)`, `tlwidth(`*linewidthstyle*`)`, and `tlcolor(`*colorstyle*`)` specify other details about the look of the ticks. See [G-4] ***linestyle***, [G-4] ***linewidthstyle***, and [G-4] ***colorstyle***. Ticks are just lines. See [G-4] **concept: lines** for more information.

`custom` specifies that the label-rendition suboptions, the tick-rendition options, and the `angle()` option apply only to the labels added on the current { `y`|`x`|`t` }[`m`]`label()` option, rather than being applied to all major or minor labels on the axis. Customizable suboptions are `tstyle()`, `labgap()`, `labstyle()`, `labsize()`, `labcolor()`, `tlength()`, `tposition()`, `tlstyle()`, `tlwidth()`, and `tlcolor()`.

`custom` is usually combined with suboption `add` to emphasize points on the axis by extending the length of the tick, changing the color or size of the label, or otherwise changing the look of the custom labels or ticks.

`grid` and `nogrid` specify whether grid lines are to be drawn across the plot region in addition to whatever else is specified in the {`y`|`x`}[`m`]`label()` or {`y`|`x`}[`m`]`tick()` option in which `grid` or `nogrid` appears. Typically, `nogrid` is the default, and `grid` is the option for all except `ylabel()`, where things are reversed and `grid` is the default and `nogrid` is the option. (Which is the default and which is the option is controlled by the scheme; see [G-4] **schemes intro**.)

For instance, specifying option

```
ylabel(, nogrid)
```

would suppress the grid lines in the y direction and specifying

```
xlabel(, grid)
```

would add them in the x. Specifying

```
xlabel(0(1)10, grid)
```

would place major labels, major ticks, and grid lines at $x = 0, 1, 2, \ldots, 10$.

[`no`]`gmin` and [`no`]`gmax` are relevant only if `grid` is in effect (because `grid` is the default and `nogrid` was not specified or because `grid` was specified). [`no`]`gmin` and [`no`]`gmax` specify whether grid lines are to be drawn at the minimum and maximum values. Consider

```
. scatter yvar xvar, xlabel(0(1)10, grid)
```

Clearly the values 0, 1, ..., 10 are to be ticked and labeled, and clearly, grid lines should be drawn at 1, 2, ..., 9; but should grid lines be drawn at 0 and 10? If 0 and 10 are at the edge of the plot region, you probably do not want grid lines there. They will be too close to the axis and border of the graph.

What you want will differ from graph to graph, so the `graph` command tries to be smart, meaning that neither `gmin` nor `nogmin` (and neither `gmax` nor `nogmax`) is the default: The default is for `graph` to decide which looks best; the options force the decision one way or the other.

If `graph` decided to suppress the grids at the extremes and you wanted them, you could type

```
. scatter yvar xvar, xlabel(0(1)10, grid gmin gmax)
```

`gstyle(`*gridstyle*`)` specifies the overall style of the grid lines, including whether the lines extend beyond the plot region and into the plot region's margins, along with the style, color, width, and pattern of the lines themselves. The options that follow allow you to change each attribute, but the *gridstyle* provides the starting point. See [G-4] ***gridstyle***.

You need not specify `gstyle()` just because there is something you want to change. You specify `gstyle()` when another style exists that is exactly what you desire or when another style would allow you to specify fewer changes to obtain what you want.

`gextend` and `nogextend` specify whether the grid lines should extend beyond the plot region and pass through the plot region's margins; see [G-3] ***region_options***. The default is determined by the `gstyle()` and scheme, but usually, `nogextend` is the default and `gextend` is the option.

`glstyle(`*linestyle*`)`, `glwidth(`*linewidthstyle*`)`, `glcolor(`*colorstyle*`)`, and `glpattern(`*linepatternstyle*`)` specify other details about the look of the grid. See [G-4] ***linestyle***, [G-4] ***linewidthstyle***, [G-4] ***colorstyle***, and [G-4] ***linepatternstyle***. Grids are just lines. See [G-4] **concept: lines** for more information. Of these options, `glpattern()` is of particular interest because, with it, you can make the grid lines dashed.

Remarks and examples

stata.com

axis_label_options are a subset of *axis_options*; see [G-3] ***axis_options*** for an overview. The other appearance options are

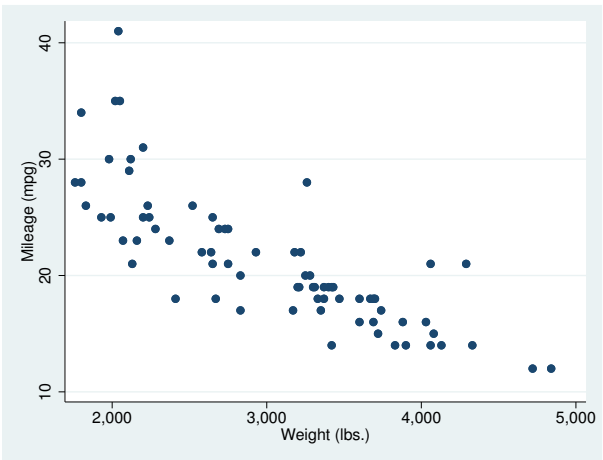
axis_scale_options	(see [G-3] ***axis_scale_options***)
axis_title_options	(see [G-3] ***axis_title_options***)

Remarks are presented under the following headings:

Default labeling and ticking

By default, approximately five values are labeled and ticked on each axis. For example, in

```
. use http://www.stata-press.com/data/r15/auto
(1978 Automobile Data)
. scatter mpg weight
```



four values are labeled on each axis because choosing five would have required widening the scale too much.

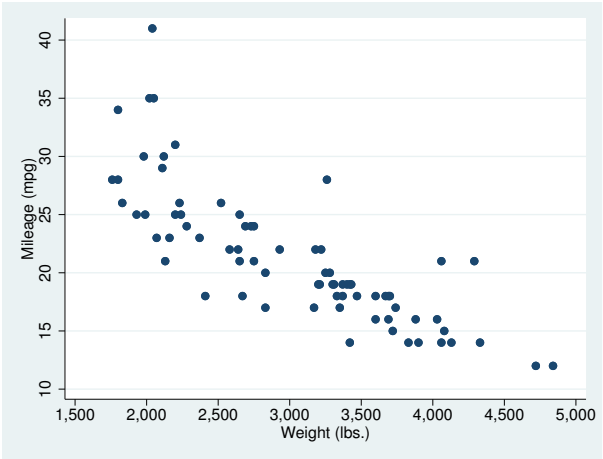
Controlling the labeling and ticking

We would obtain the same results as we did in the above example if we typed

```
. scatter mpg weight, ylabel(#5) xlabel(#5)
```

Options `ylabel()` and `xlabel()` specify the values to be labeled and ticked, and `#5` specifies that Stata choose approximately five values for us. If we wanted many values labeled, we might type

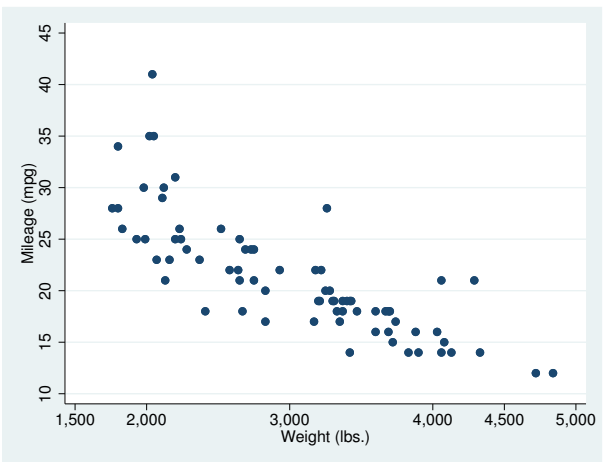
```
. scatter mpg weight, ylabel(#10) xlabel(#10)
```



As with `#5`, `#10` was not taken too seriously; we obtained seven labels on the y axis and eight on the x axis.

We can also specify precisely the values we want labeled by specifying `#(#)#` or by specifying a list of numbers:

```
. scatter mpg weight, ylabel(10(5)45)
                      xlabel(1500 2000 3000 4000 4500 5000)
```

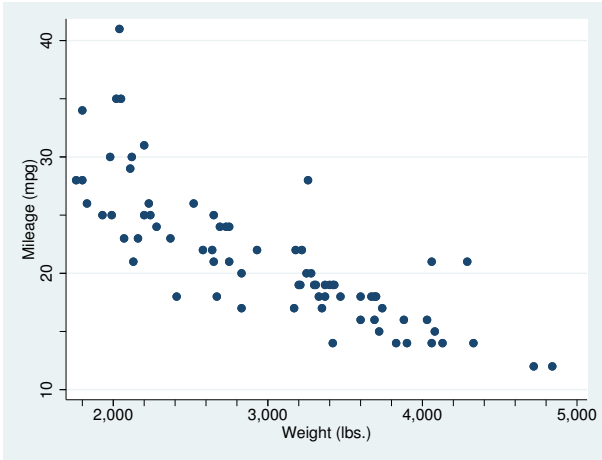


In option `ylabel()`, we specified the rule `10(5)45`, which means to label from 10 to 45 in steps of 5. In option `xlabel()`, we typed out the values to be labeled.

Adding extra ticks

Options `ylabel()` and `xlabel()` draw ticks plus labels. Options `ytick()` and `xtick()` draw ticks only, so you can do things such as

```
. scatter mpg weight, ytick(#10) xtick(#15)
```



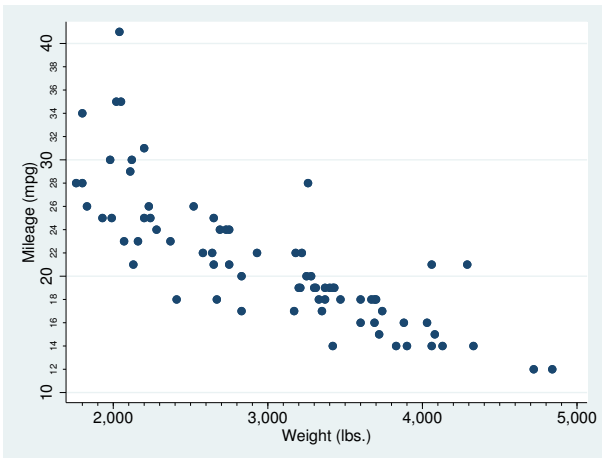
Of course, as with `ylabel()` and `xlabel()`, you can specify the exact values you want ticked.

Adding minor labels and ticks

Minor ticks and minor labels are smaller than regular ticks and regular labels. Options `ymlabel()` and `xmlabel()` allow you to place minor ticks with labels, and `ymtick()` and `xmtick()` allow you to place minor ticks without labels. When using minor ticks and labels, in addition to the usual syntax of `#5` to mean approximately 5 values, `10(5)45` to mean 10 to 45 in steps of 5, and a list of numbers, there is an additional syntax: `##5`. `##5` means that each major interval is divided into 5 minor intervals.

The graph below is intended more for demonstration than as an example of a good-looking graph:

```
. scatter mpg weight, ymlabel(##5) xmtick(##10)
```



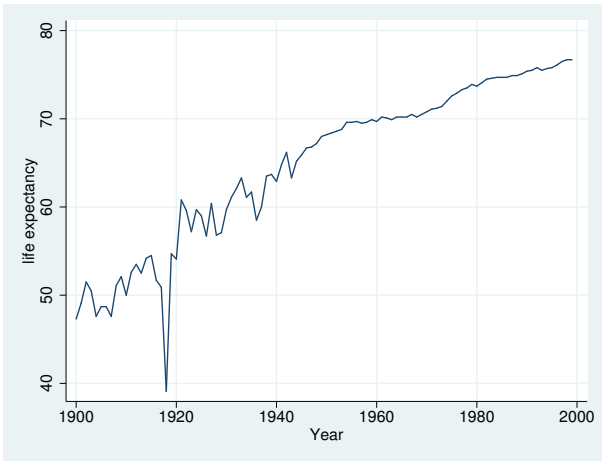
`##5` means four ticks, and `##10` means nine ticks because most people think in reciprocals they say to themselves, "I want to tick the fourths so I want 4 ticks between," or, "I want to tick the tenths so I want 10 ticks between". They think incorrectly. They should think that if they want fourths, they want $4 - 1 = 3$ ticks between, or if they want tenths, they want $10 - 1 = 9$ ticks between. Stata subtracts one so that they can think—and correctly—when they want fourths that they want `##4` ticks between and that when they want tenths they want `##10` ticks between.

For `###` rules to work, the major ticks must be evenly spaced. This format is guaranteed only when the major ticks or labels are specified using the `#(#)#` rule. The `###` rule also works in almost all cases, the exception being daily data where the date variable is specified in the `%td` format. Here "nice" daily labels often do not have a consistent number of days between the ticks and thus the space between each major tick cannot be evenly divided. If the major ticks are not evenly spaced, the `###` rule does not produce any minor ticks.

Adding grid lines

To obtain grid lines, specify the `grid` suboption of `ylabel()`, `xlabel()`, `ymlabel()`, or `xmlabel()`. `grid` specifies that, in addition to whatever else the option would normally do, grid lines be drawn at the same values. In the example below,

```
. use http://www.stata-press.com/data/r15/uslifeexp, clear
(U.S. life expectancy, 1900-1999)
. line le year, xlabel(,grid)
```



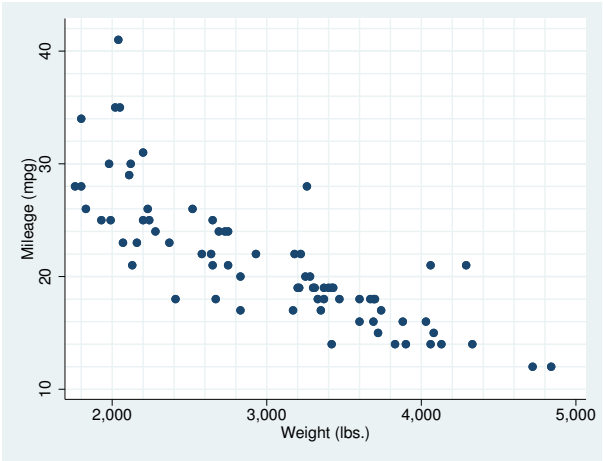
we specify `xlabel(,grid)`, omitting any mention of the specific values to use. Thus `xlabel()` did what it does ordinarily (labeled approximately five nice values), and it drew grid lines at those same values.

Of course, we could have specified the values to be labeled and gridded:

```
. line le year, xlabel(#10, grid)
. line le year, xlabel(1900(10)2000, grid)
. line le year, xlabel(1900 1918 1940(20)2000, grid)
```

The `grid` suboption is usually specified with `xlabel()` (and with `ylabel()` if, given the scheme, `grid` is not the default), but it may be specified with any of the *axis_label_options*. In the example below, we "borrow" `ymtick()` and `xmtick()`, specify `grid` to make them draw grids, and specify `style(none)` to make the ticks themselves invisible:

```
. use http://www.stata-press.com/data/r15/auto, clear
(1978 Automobile Data)
. scatter mpg weight, ymtick(#20, grid tstyle(none))
                      xmtick(#20, grid tstyle(none))
```



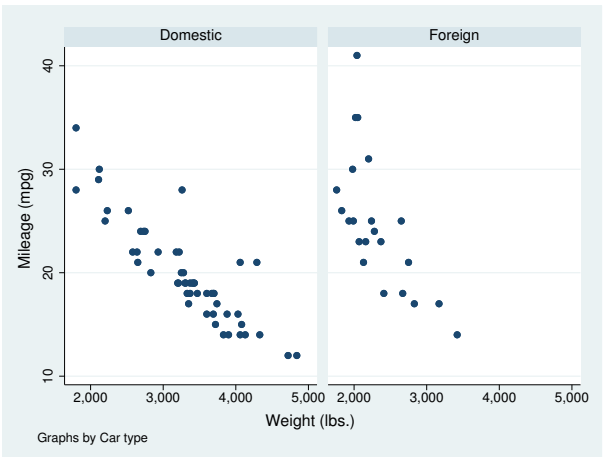
If you look carefully at the graph above, you will find that no grid line was drawn at $x = 5{,}000$. Stata suppresses grid lines when they get too close to the axes or borders of the graph. If you want to force Stata to draw them anyway, you can specify the `gmin` and `gmax` options:

```
. scatter mpg weight, ymtick(#20, grid tstyle(none))
                      xmtick(#20, grid tstyle(none) gmax)
```

Suppressing grid lines

Some commands, and option `ylabel()`, usually draw grid lines by default. For instance, in the following, results are the same as if you specified `ylabel(,grid)`:

```
. use http://www.stata-press.com/data/r15/auto, clear
(1978 Automobile Data)
. scatter mpg weight, by(foreign)
```



To suppress the grid lines, specify `ylabel(,nogrid)`:

```
. scatter mpg weight, by(foreign) ylabel(,nogrid)
```

Substituting text for labels

In addition to specifying explicitly the values to be labeled by specifying things such as `ylabel(10(10)50)` or `ylabel(10 20 30 40 50)`, you can specify text to be substituted for the label. If you type

```
. graph ... , ... ylabel(10 20 30 "mean" 40 50)
```

The values 10, 20, ..., 50 will be labeled, just as you would expect, but for the middle value, rather than the text "30" appearing, the text "mean" (without the quotes) would appear.

In the advanced example below, we specify

```
xlabel(1 "J" 2 "F" 3 "M" 4 "A" 5 "M" 6 "J" 7 "J" 8 "A" 9 "S" 10 "O" 11 "N" 12 "D")
```

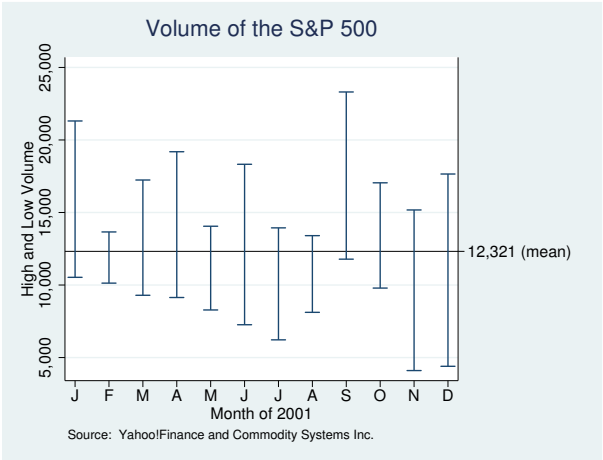
so that rather than seeing the numbers 1, 2, ..., 12 (which are month numbers), we see J, F, ..., D; and we specify

```
ylabel(12321 "12,321 (mean)", axis(2) angle(0))
```

so that we label 12321 but, rather than seeing 12321, we see "12,321 (mean)". The `axis(2)` option puts the label on the second y axis (see [G-3] ***axis_choice_options***) and `angle(0)` makes the text appear horizontally rather than vertically (see *Options* above):

```
. use http://www.stata-press.com/data/r15/sp500, clear
(S&P 500)
. generate month = month(date)
. sort month
. by month: egen lo = min(volume)
. by month: egen hi = max(volume)
. format lo hi %10.0gc
. summarize volume
    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
      volume |        248    12320.68    2585.929      4103    23308.3
. by month: keep if _n==_N
(236 observations deleted)
```

```
. twoway rcap lo hi month,
          xlabel(1 "J"  2 "F"  3 "M"  4 "A"  5 "M"  6 "J"
                 7 "J"  8 "A"  9 "S" 10 "O" 11 "N" 12 "D")
          xtitle("Month of 2001")
          ytitle("High and Low Volume")
          yaxis(1 2) ylabel(12321 "12,321 (mean)", axis(2) angle(0))
          ytitle("", axis(2))
          yline(12321, lstyle(foreground))
          msize(*2)
          title("Volume of the S&P 500", margin(b+2.5))
          note("Source:  Yahoo!Finance and Commodity Systems Inc.")
```



Contour axes—zlabel(), etc.

The `zlabel()`, `ztick()`, `zmlabel()`, and `zmtick()` options are unusual in that they apply not to axes on the plot region, but to the axis that shows the scale of a contour legend. They have effect only when the graph includes a `twoway contour` plot; see [G-2] **graph twoway contour**. In all other respects, they act like the `x`*, `y`*, and `t`* options.

For an example using `zlabel()`, see *Controlling the number of contours and their values* in [G-2] **graph twoway contour**.

The options associated with grids have no effect when specified on contour axes.

Appendix: Details of syntax

Suboptions without rules, numlists, or labels

What may appear in each of the options { `y` | `x` }{ `label` | `tick` | `mlabel` | `mtick` }`()` is a rule or numlist followed by suboptions:

[*rule*] [*numlist* ["*label*" [*numlist* ["*label*" [...]]]]] [, *suboptions*]

rule, *numlist*, and *label* are optional. If you remove those, you are left with

, *suboptions*

That is, the options { `y` | `x` }{ `label` | `tick` | `mlabel` | `mtick` }`()` may be specified with just suboptions and, in fact, they are often specified that way. If you want default labeling of the y axis and x axis, but you want grid lines in the x direction as well as the y, specify

```
. scatter yvar xvar , xlabel(,grid)
```

When you do not specify the first part—the *rule*, *numlist*, and *label*—you are saying that you do not want that part to change. You are saying that you merely wish to change how the *rule*, *numlist*, and *label* are displayed.

Of course, you may specify more than one suboption. You might type

```
. scatter yvar xvar, xlabel(,grid format(%9.2f))
```

if, in addition to grid lines, you wanted the numbers presented on the x axis to be presented in a `%9.2f` format.

Rules

What may appear in each of the axis-label options is a rule or numlist

[*rule*] [*numlist* ["*label*" [*numlist* ["*label*" [...]]]]] [, *suboptions*]

where either *rule* or *numlist* must be specified and both may be specified. Let us ignore the "*label*" part right now. Then the above simplifies to

[*rule*] [*numlist*] [, *suboptions*]

where *rule* or *numlist* must be specified, both may be specified, and most often you will simply specify the *rule*, which may be any of the following:

rule	Example	Description
`##`	`#6`	6 nice values
`###`	`##10`	$10 - 1 = 9$ values between major ticks; allowed with `mlabel()` and `mtick()` only
`#(#)#`	`-4(.5)3`	specified range: -4 to 3 in steps of .5
`minmax`	`minmax`	minimum and maximum values
`none`	`none`	label no values
`.`	`.`	skip the rule

The most commonly specified rules are `##` and `###`.

Specifying `##` says to choose # nice values. Specifying `#5` says to choose five nice values, `#6` means to choose six, and so on. If you specify `ylabel(#5)`, then five values will be labeled (on the y axis). If you also specify `ymtick(#10)`, then 10 minor ticks will also be placed on the axis. Actually, `ylabel(#5)` and `ymtick(#10)` will result in approximately five labels and 10 minor ticks because the choose-a-nice-number routine will change your choice a little if, in its opinion, that would yield a nicer overall result. You may not agree with the routine about what is nice, and then the `#(#)#` rule will let you specify exactly what you want, assuming that you want evenly spaced labels and numbers.

`###` is allowed only with the {`y`|`x`}`mlabel()` and {`y`|`x`}`mtick()` options—the options that result in minor ticks. `###` says to put #−1 minor ticks between the major ticks. `##5` would put four, and `##10` would put nine. Here # is taken seriously, at least after subtraction, and you are given exactly what you request.

`#(#)#` can be used with major or minor labels and ticks. This rule says to label the first number specified, increment by the second number, and keep labeling, as long as the result is less than or equal to the last number specified. `ylabel(1(1)10)` would label (and tick) the values 1, 2, ..., 10. `ymtick(1(.5)10)` would put minor ticks at 1, 1.5, 2, 2.5, ..., 10. It would be perfectly okay to specify both of those options. When specifying rules, minor ticks and labels will check what is specified for major ticks and labels and remove the intersection so as not to overprint. The results will be the same as if you specified `ymtick(1.5(1)9.5)`.

The rule `minmax` specifies that you want the minimum and maximum. `ylabel(minmax)` would label only the minimum and maximum.

Rule `none` means precisely that: the rule that results in no labels and no ticks.

Rule `.` makes sense only when `add` is specified, although it is allowed at other times, and then `.` means the same as `none`.

Rules and numlists

After the *rule*—or instead of it—you can specify a *numlist*. A numlist is a list of numbers, for instance, "`1 2 5 7`" (without the quotes) or "`3/9`" (without the quotes). Other shorthands are allowed (see **[U] 11.1.8 numlist**), and in fact, one of *numlist*'s syntaxes looks just like a *rule*: `#(#)#`. It has the same meaning, too.

There is, however, a subtle distinction between, for example,

`ylabel(1(1)10)` (a *rule*) and `ylabel(none 1(1)10)` (a *numlist*)

Rules are more efficient. Visually, however, there is no difference.

Use numlists when the values you wish to label or to tick are unequally spaced,

```
ylabel(none 1 2 5 7)
```

or when there is one or more extra values you want to label or to tick:

```
ylabel(1(1)10 3.5 7.5)
```

Rules and numlists and labels

Numlists serve an additional purpose—you can specify text that is to be substituted for the value to be labeled. For instance,

```
ylabel(1(1)10 3.5 "Low" 7.5 "Hi")
```

says to label 1, 2, ..., 10 (that is the *rule* part) and to label the special values 3.5 and 7.5. Rather than actually printing "3.5" and "7.5" next to the ticks at 3.5 and 7.5, however, `graph` will instead print the words "Low" and "Hi".

Interpretation of repeated options

Each of the axis-label options may be specified more than once in the same command. If you do that and you do not specify suboption `add`, the rightmost of each is honored. If you specify suboption `add`, then the option just specified and the previous options are merged. `add` specifies that any new ticks or labels are added to any existing ticks or labels on the axis. All suboptions are *rightmost*; see [G-4] **concept: repeated options**.

Reference

Cox, N. J. 2007. Stata tip 55: Better axis labeling for time points and time intervals. *Stata Journal* 7: 590–592.

Also see

[G-3] ***axis_options*** — Options for specifying numeric axes

[G-3] ***axis_scale_options*** — Options for specifying axis scale, range, and look

[G-3] ***axis_title_options*** — Options for specifying axis titles