Title

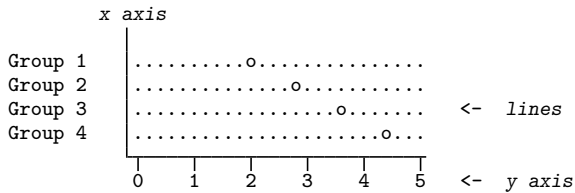> **graph dot** — Dot charts (summary statistics)

Description

graph dot draws horizontal dot charts. In a dot chart, the categorical axis is presented vertically, and the numerical axis is presented horizontally. Even so, the numerical axis is called the y axis, and the categorical axis is still called the x axis:
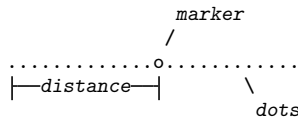
```
. graph dot (mean) numeric_var, over(cat_var)
```

```
                   x
    first group   |......o.............
                  |
   second group   |.........o.........
          .       |
          .       |
                  |_____
                   0   2   4   6   8
```
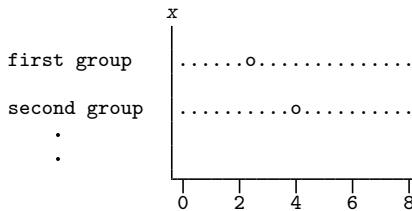
The syntax for dot charts is identical to that for bar charts; see [G-2] **graph bar**.

We use the following words to describe a dot chart:

```
            x axis
   Group 1  |.........o..............
   Group 2  |.............o..........
   Group 3  |.................o.......    <-  lines
   Group 4  |....................o...
            |_____
             0   1   2   3   4   5    <-  y axis
```

The above dot chart contains four *lines*. The words used to describe a line are

```
                              marker
                             /
          ............o...........
          |—distance—|          \
                                 dots
```

Quick start

Dot graph showing the mean of v1

```
graph dot v1
```

As above, with dots for the means of v1 and v2 on a single line

```
graph dot v1 v2
```

As above, but with dots for the means of v1 and v2 on separate lines

```
graph dot v1 v2, ascategory
```

1

As above, with dots showing the means of v1 and v2 for each level of categorical variable `catvar1`
```
graph dot v1 v2, over(catvar1)
```

Include missing values of `catvar1` as their own category
```
graph dot v1 v2, over(catvar1) missing
```

Dot graph with dots for each combination of the levels of `catvar1` and `catvar2` for levels of `catvar1` grouped by levels of `catvar2`
```
graph dot v1 v2, over(catvar1) over(catvar2)
```

As above, but with levels of `catvar2` grouped by levels of `catvar1`
```
graph dot v1, over(catvar2) over(catvar1)
```

Dots for the medians of v1 and v2 for each level of `catvar1`
```
graph dot (median) v1 v2, over(catvar1)
```

A separate graph area for each dot graph of the mean of v1 in groups defined by levels of `catvar2`
```
graph dot v1, by(catvar2)
```

As above, but with dots for each level of `catvar1` within each graph area
```
graph dot v1, over(catvar1) by(catvar2)
```

Dot graph of the sums of v1 and v2 for each level of `catvar1`
```
graph dot (sum) v1 v2, over(catvar1)
```

As above, but show the mean and median of v1
```
graph dot (mean) v1 (median) v1, over(catvar1)
```

Change the label of v1 and v2 to "Variable 1" and "Variable 2" in the legend
```
graph dot v1 v2, over(catvar1) legend(label(1 "Variable 1")   ///
    label(2 "Variable 2"))
```

Menu

Graphics > Dot chart

Syntax

>   <u>gr</u>aph dot *yvars* $\begin{bmatrix} if \end{bmatrix}$ $\begin{bmatrix} in \end{bmatrix}$ $\begin{bmatrix} weight \end{bmatrix}$ $\begin{bmatrix} , options \end{bmatrix}$

where *yvars* is

>>   (asis) *varlist*

>   or is

>>   (percent) $\begin{bmatrix} varlist \end{bmatrix}$ | (count) $\begin{bmatrix} varlist \end{bmatrix}$

>   or is

>> $\begin{bmatrix} (stat) \end{bmatrix}$ *varname* $\begin{bmatrix} \begin{bmatrix} (stat) \end{bmatrix} \dots \end{bmatrix}$

>> $\begin{bmatrix} (stat) \end{bmatrix}$ *varlist* $\begin{bmatrix} \begin{bmatrix} (stat) \end{bmatrix} \dots \end{bmatrix}$

>> $\begin{bmatrix} (stat) \end{bmatrix}$ $\begin{bmatrix} name= \end{bmatrix}$*varname*$\begin{bmatrix} \dots \end{bmatrix}$ $\begin{bmatrix} \begin{bmatrix} (stat) \end{bmatrix} \dots \end{bmatrix}$

where *stat* may be any of

>>   mean median p1 p2 ... p99 sum count percent min max

>   or

>>   any of the other *stats* defined in [D] **collapse**

yvars is optional if the option over(*varname*) is specified. percent is the default statistic, and percentages are calculated over *varname*.

mean is the default when *varname* or *varlist* is specified and *stat* is not specified. p1 means the first percentile, p2 means the second percentile, and so on; p50 means the same as median. count means the number of nonmissing values of the specified variable.

options	Description
group_options	groups over which lines of dots are drawn
yvar_options	variables that are the dots
linelook_options	how the lines of dots look
legending_options	how *yvars* are labeled
axis_options	how numerical y axis is labeled
title_and_other_options	titles, added text, aspect ratio, etc.

Each is defined below.

group_options	Description
<u>over</u>(*varname*[, *over_subopts*])	categories; option may be repeated
nofill	omit empty categories
missing	keep missing value as category
allcategories	include all categories in the dataset

yvar_options	Description
ascategory	treat *yvars* as first over() group
asyvars	treat first over() group as *yvars*
<u>percent</u>ages	show percentages within *yvars*
cw	calculate *yvar* statistics omitting missing values of any *yvar*

linelook_options	Description
outergap([*]#)	gap between top and first line and between last line and bottom
linegap(#)	gap between *yvar* lines; default is 0
marker(#, *marker_options*)	marker used for #th *yvar* line
pcycle(#)	marker styles before [pstyles](#) recycle
<u>linet</u>ype(dot | line | rectangle)	type of line
ndots(#)	# of dots if linetype(dot); default is 100
dots(*marker_options*)	look if linetype(dot)
lines(*line_options*)	look if linetype(line)
<u>recta</u>ngles(*area_options*)	look if linetype(rectangle)
rwidth(*size*)	rectangle width if linetype(rectangle)
[no]<u>extend</u>line	whether line extends through plot region margins; extendline is usual default
lowextension(*size*)	extend line through axis (advanced)
highextension(*size*)	extend line through axis (advanced)

legending_options	Description
legend_options	control of *yvar* legend
nolabel	use *yvar* names, not labels, in legend
yvaroptions(*over_subopts*)	*over_subopts* for *yvars*; seldom specified
showyvars	label *yvars* on x axis; seldom specified

axis_options	Description
<u>y</u>alternate	put numerical y axis on right (top)
<u>x</u>alternate	put categorical x axis on top (right)
exclude0	do not force y axis to include 0
<u>yr</u>everse	reverse y axis
axis_scale_options	y-axis scaling and look
axis_label_options	y-axis labeling
<u>yti</u>tle(...)	y-axis titling

title_and_other_options	Description
text(...)	add text on graph; x range $\left[\, 0,\, 100 \,\right]$
<u>yli</u>ne(...)	add y lines to graph
aspect_option	constrain aspect ratio of plot region
std_options	titles, graph size, saving to disk
by(*varlist*, ...)	repeat for subgroups

The *over_subopts*—used in over(*varname*, *over_subopts*) and, on rare occasion, in yvaroptions(*over_subopts*)—are

over_subopts	Description
<u>rel</u>abel(*#* "*text*" ...)	change axis labels
<u>lab</u>el(*cat_axis_label_options*)	rendition of labels
<u>ax</u>is(*cat_axis_line_options*)	rendition of axis line
gap($\left[\,*\,\right]$*#*)	gap between lines within over() category
sort(*varname*)	put lines in prespecified order
sort(*#*)	put lines in height order
sort((*stat*) *varname*)	put lines in derived order
<u>desc</u>ending	reverse default or specified line order

aweights, fweights, and pweights are allowed; see [U] **11.1.6 weight** and see note concerning weights in [D] **collapse**.

Options

Options are presented under the following headings:

> *group_options*
> *yvar_options*
> *linelook_options*
> *legending_options*
> *axis_options*
> *title_and_other_options*
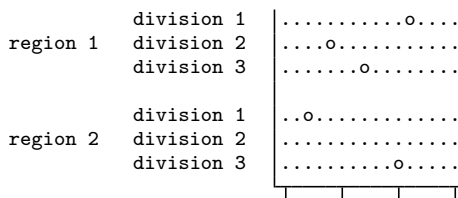> *Suboptions for use with over( ) and yvaroptions( )*

group_options

over(*varname* [, *over_subopts*]) specifies a categorical variable over which the *yvars* are to be repeated. *varname* may be string or numeric. Up to two over() options may be specified when multiple *yvars* are specified, and up to three over()s may be specified when one *yvar* is specified; options may be specified; see *Appendix: Examples of syntax* below.
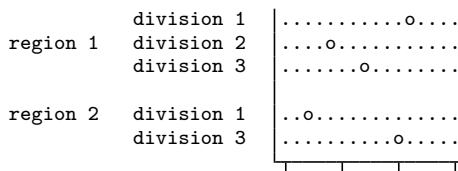
nofill specifies that missing subcategories be omitted. For instance, consider

```
. graph dot (mean) y, over(division) over(region)
```

Say that one of the divisions has no data for one of the regions, either because there are no such observations or because y==. for such observations. In the resulting chart, the marker will be missing:

```
                   division 1  |...........o....
        region 1   division 2  |....o...........
                   division 3  |.......o........

                   division 1  |..o.............
        region 2   division 2  |................
                   division 3  |..........o.....
                               └──┬───┬───┬───┬──
```

If you specify nofill, the missing category will be removed from the chart:

```
                   division 1  |...........o....
        region 1   division 2  |....o...........
                   division 3  |.......o........

                   division 1  |..o.............
        region 2   division 3  |..........o.....
                               └──┬───┬───┬───┬──
```

missing specifies that missing values of the over() variables be kept as their own categories, one for ., another for .a, etc. The default is to ignore such observations. An over() variable is considered to be missing if it is numeric and contains a missing value or if it is string and contains "".

allcategories specifies that all categories in the entire dataset be retained for the over() variables. When if or in is specified without allcategories, the graph is drawn, completely excluding any categories for the over() variables that do not occur in the specified subsample. With the allcategories option, categories that do not occur in the subsample still appear in the legend, but no markers are drawn where these categories would appear. Such behavior can be convenient when comparing graphs of subsamples that do not include completely common categories for all over() variables. This option has an effect only when if or in is specified or if there are missing values in the variables. allcategories may not be combined with by().

yvar_options

ascategory specifies that the *yvars* be treated as the first over() group.

When you specify ascategory, results are the same as if you specified one *yvar* and introduced a new first over() variable. Anyplace you read in the documentation that something is done over the first over() category, or using the first over() category, it will be done over or using *yvars*.

Suppose that you specified

```
. graph dot y1 y2 y3, ascategory whatever_other_options
```

The results will be the same as if you typed

    . graph dot *y*, over(*newcategoryvariable*) *whatever_other_options*

with a long rather than wide dataset in memory.

asyvars specifies that the first over() group be treated as *yvars*.

When you specify asyvars, results are the same as if you removed the first over() group and introduced multiple *yvars*. We said in most ways, not all ways, but let's ignore that for a moment. If you previously had *k yvars* and, in your first over() category, *G* groups, results will be the same as if you specified *k*G* yvars and removed the over(). Anyplace you read in the documentation that something is done over the *yvars* or using the *yvars*, it will be done over or using the first over() group.

Suppose that you specified

    . graph dot y, over(group) asyvars *whatever_other_options*

Results will be the same as if you typed

    . graph dot *y1 y2 y3* ..., *whatever_other_options*

with a wide rather than long dataset in memory. Variables *y1*, *y2*, ..., are sometimes called the virtual *yvars*.

percentages specifies that marker positions be based on percentages that *yvar_i* represents of all the *yvars*. That is,

    . graph dot (mean) inc_male inc_female

would produce a chart with the markers reflecting average income.

    . graph dot (mean) inc_male inc_female, percentage

would produce a chart with the markers being located at $100 \times$ inc_male$/($inc_male $+$ inc_female$)$ and $100 \times$ inc_female$/($inc_male $+$ inc_female$)$.

If you have one *yvar* and want percentages calculated over the first over() group, specify the asyvars option. For instance,

    . graph dot (mean) wage, over(*i*) over(*j*)

would produce a chart where marker positions reflect mean wages.

    . graph dot (mean) wage, over(*i*) over(*j*) asyvars percentages

would produce a chart where marker positions are $100 \times (\text{mean}_{ij}/(\text{Sum}_i\ \text{mean}_{ij}))$

cw specifies casewise deletion. If cw is specified, observations for which any of the *yvars* are missing are ignored. The default is to calculate each statistic by using all the data possible.

linelook_options

outergap(**#*) and outergap(*#*) specify the gap between the top of the graph to the beginning of the first line and the last line to the bottom of the graph.

outergap(**#*) specifies that the default be modified. Specifying outergap(*1.2) increases the gap by 20%, and specifying outergap(*.8) reduces the gap by 20%.

outergap(*#*) specifies the gap as a percentage-of-bar-width units. graph dot is related to graph bar. Just remember that outergap(50) specifies a sizable but not excessive gap.

linegap(#) specifies the gap to be left between *yvar* lines. The default is linegap(0), meaning that multiple *yvars* appear on the same line. For instance, typing

   . graph dot y1 y2, over(group)

results in

```
group 1 |..x....o........
group 2 |........x..o....
group 3 |.......x.....o..
        ⊤    ⊤    ⊤    ⊤
```

In the above, o represents the symbol for y1 and x the symbol for y2. If you want to have separate lines for the separate *yvars*, specify linegap(20):

   . graph dot y1 y2, over(group) linegap(20)

```
group 1 |.......o........
        |..x.............

group 2 |...........o....
        |........x.......

group 3 |.............o..
        |.......x........
        ⊤    ⊤    ⊤    ⊤
```

Specify a number smaller or larger than 20 to reduce or increase the distance between the y1 and y2 lines.

Alternatively, and generally preferred, is specifying option ascategory, which will result in

   . graph dot y1 y2, over(group) ascategory

```
group 1  y1 |.......o........
         y2 |..o............

group 2  y1 |...........o....
         y2 |........o.......

group 3  y1 |.............o..
         y2 |.......o........
            ⊤    ⊤    ⊤    ⊤
```

linegap() affects only the *yvar* lines. If you want to change the gap for the first, second, or third over() groups, specify the *over_subopt* gap() inside the over() itself.

marker(#, *marker_options*) specifies the shape, size, color, etc., of the marker to be used to mark the value of the #th *yvar* variable. marker(1, ...) refers to the marker associated with the first *yvar*, marker(2, ...) refers to the marker associated with the second, and so on. A particularly useful *marker_option* is mcolor(*colorstyle*), which sets the color and opacity of the marker. For instance, you might specify marker(1, mcolor(green)) to make the marker associated with the first *yvar* green. See [G-4] *colorstyle* for a list of color choices, and see [G-3] *marker_options* for information on the other *marker_options*.

pcycle(#) specifies how many variables are to be plotted before the pstyle (see [G-4] *pstyle*) of the markers for the next variable begins again at the pstyle of the first variable—p1dot (with the markers for the variable following that using p2dot and so on). Put another way, # specifies how quickly the look of markers is recycled when more than # variables are specified. The default for most schemes is pcycle(15).

linetype(dot), linetype(line), and linetype(rectangle) specify the style of the line.

> linetype(dot) is the usual default. In this style, dots are used to fill the line around the marker:

> > . . . . .o. . . . .

> linetype(line) specifies that a solid line be used to fill the line around the marker:

> > ————o————

> linetype(rectangle) specifies that a long "rectangle" (which looks more like two parallel lines) be used to fill the area around the marker:

> > ========o=======

ndots(#) and dots(*marker_options*) are relevant only in the linetype(dots) case.

> ndots(#) specifies the number of dots to be used to fill the line. The default is ndots(100).

> dots(*marker_options*) specifies the marker symbol, color, and size to be used as the dot symbol. The default is to use dots(msymbol(p)). See [G-3] *marker_options*.

lines(*line_options*) is relevant only if linetype(line) is specified. It specifies the look of the line to be used; see [G-3] *line_options*.

rectangles(*area_options*) and rwidth(*size*) are relevant only if linetype(rectangle) is specified.

> rectangles(*area_options*) specifies the look of the parallel lines (rectangle); see [G-3] *area_options*.

> rwidth(*size*) specifies the width (height) of the rectangle (the distance between the parallel lines). The default is usually rwidth(.45); see [G-4] *size*.

noextendline and extendline are relevant in all cases. They specify whether the line (dots, a line, or a rectangle) is to extend through the plot region margin and touch the axes. The usual default is extendline, so noextendline is the option. See [G-3] *region_options* for a definition of the plot region.

lowextension(*size*) and highextension(*size*) are advanced options that specify the amount by which the line (dots, line or a rectangle) is extended through the axes. The usual defaults are lowextension(0) and highextension(0). See [G-4] *size*.

legending_options

legend_options allows you to control the legend. If more than one *yvar* is specified, a legend is produced. Otherwise, no legend is needed because the over() groups are labeled on the categorical x axis. See [G-3] *legend_options*.

nolabel specifies that, in automatically constructing the legend, the variable names of the *yvars* be used in preference to "mean of *varname*" or "sum of *varname*", etc.

yvaroptions(*over_subopts*) allows you to specify *over_subopts* for the *yvars*. This is seldom specified.

showyvars specifies that, in addition to building a legend, the identities of the *yvars* be shown on the categorical x axis. If showyvars is specified, it is typical to also specify legend(off).

axis_options

yalternate and xalternate switch the side on which the axes appear. yalternate moves the numerical y axis from the bottom to the top; xalternate moves the categorical x axis from the left to the right. If your scheme by default puts the axes on the opposite sides, yalternate and xalternate reverse their actions.

exclude0 specifies that the numerical y axis need not be scaled to include 0.

yreverse specifies that the numerical y axis have its scale reversed so that it runs from maximum to minimum.

axis_scale_options specify how the numerical y axis is scaled and how it looks; see [G-3] *axis_scale_options*. There you will also see option xscale() in addition to yscale(). Ignore xscale(), which is irrelevant for dot plots.

axis_label_options specify how the numerical y axis is to be labeled. The *axis_label_options* also allow you to add and suppress grid lines; see [G-3] *axis_label_options*. There you will see that, in addition to options ylabel(), ytick(), ymlabel(), and ymtick(), options xlabel(), ..., xmtick() are allowed. Ignore the x*() options, which are irrelevant for dot charts.

ytitle() overrides the default title for the numerical y axis; see [G-3] *axis_title_options*. There you will also find option xtitle() documented, which is irrelevant for dot charts.

title_and_other_options

text() adds text to a specified location on the graph; see [G-3] *added_text_options*. The basic syntax of text() is

text($#_y$ $#_x$ "*text*")

text() is documented in terms of twoway graphs. When used with dot charts, the "numeric" x axis is scaled to run from 0 to 100.

yline() adds vertical lines at specified y values; see [G-3] *added_line_options*. The xline() option, also documented there, is irrelevant for dot charts. If your interest is in adding grid lines, see [G-3] *axis_label_options*.

aspect_option allows you to control the relationship between the height and width of a graph's plot region; see [G-3] *aspect_option*.

std_options allow you to add titles, control the graph size, save the graph on disk, and much more; see [G-3] *std_options*.

by(*varlist*, ...) draws separate plots within one graph; see [G-3] *by_option*.

Suboptions for use with over() and yvaroptions()

relabel(# "*text*" ...) specifies text to override the default category labeling. See the description of the relabel() option in [G-2] **graph bar** for more information about this very useful option.

label(*cat_axis_label_options*) determines other aspects of the look of the category labels on the x axis. Except for label(labcolor()) and label(labsize()), these options are seldom specified; see [G-3] *cat_axis_label_options*.

axis(*cat_axis_line_options*) specifies how the axis line is rendered. This is a seldom specified option. See [G-3] *cat_axis_line_options*.

gap(*#*) and gap(**#*) specify the gap between the lines in this over() group. gap(*#*) is specified in percentage-of-bar-width units. Just remember that gap(50) is a considerable, but not excessive width. gap(**#*) allows modifying the default gap. gap(*1.2) would increase the gap by 20%, and gap(*.8) would decrease the gap by 20%.

sort(*varname*), sort(*#*), and sort((*stat*) *varname*) control how the lines are ordered. See *How bars are ordered* and *Reordering the bars* in [G-2] **graph bar**.

> sort(*varname*) puts the lines in the order of *varname*.

> sort(*#*) puts the markers in distance order. *#* refers to the *yvar* number on which the ordering should be performed.

> sort((*stat*) *varname*) puts the lines in an order based on a calculated statistic.

descending specifies that the order of the lines—default or as specified by sort()—be reversed.

Remarks and examples

Remarks are presented under the following headings:

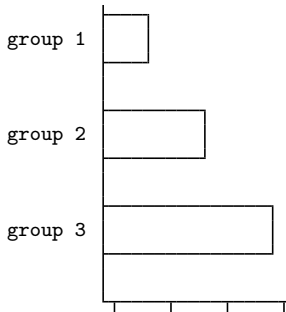> *Relationship between dot plots and horizontal bar charts*
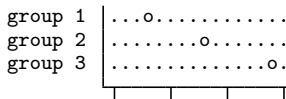> *Examples*
> *Appendix: Examples of syntax*

Relationship between dot plots and horizontal bar charts

Despite appearances, graph hbar and graph dot are in fact the same command, meaning that concepts and options are the same:
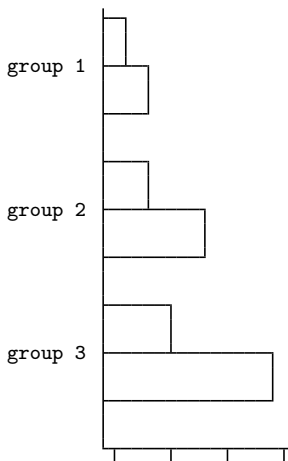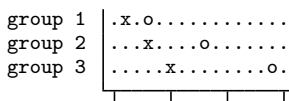
There is only one substantive difference between the two commands: Given multiple *yvars*, graph hbar draws multiple bars:

. graph hbar y1 y2, over(group)

graph dot draws multiple markers on single lines:

. graph dot y1 y2, over(group)
```
group 1 |.x.o...........
group 2 |...x....o.......
group 3 |.....x........o.
        ⊤  ⊤  ⊤  ⊤
```

The way around this problem (if it is a problem) is to specify option ascategory or to specify option linegap(#). Specifying ascategory is usually best.

Read about graph hbar in [G-2] **graph bar**.

Examples

Because graph dot and graph hbar are so related, the following examples should require little by way of explanation:

```
. use https://www.stata-press.com/data/r17/nlsw88
(NLSW, 1988 extract)

. graph dot wage, over(occ, sort(1))
        ytitle("")
        title("Average hourly wage, 1988, women aged 34-46", span)
        subtitle(" ")
        note("Source:  1988 data from NLS, U.S. Dept. of Labor,
                Bureau of Labor Statistics", span)
```
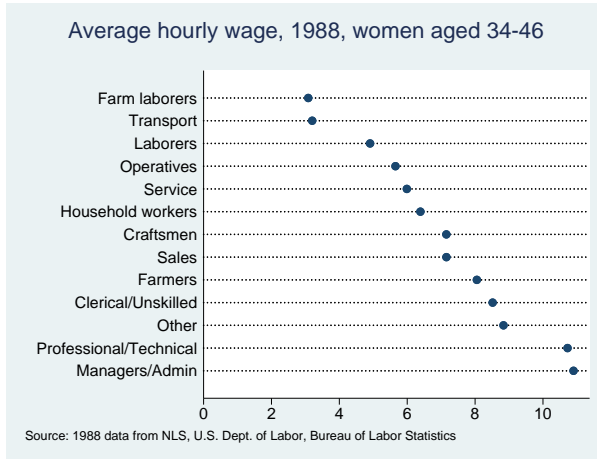


```
. graph dot (p10) wage (p90) wage,
        over(occ, sort(2))
        legend(label(1 "10th percentile") label(2 "90th percentile"))
        title("10th and 90th percentiles of hourly wage", span)
        subtitle("Women aged 34-46, 1988" " ", span)
        note("Source:  1988 data from NLS, U.S. Dept. of Labor,
                Bureau of Labor Statistics", span)
```
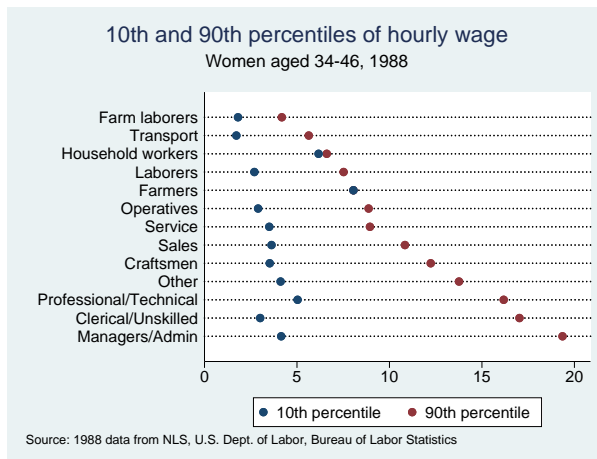
```
. graph dot (mean) wage,
        over(occ, sort(1))
        by(collgrad,
            title("Average hourly wage, 1988, women aged 34-46", span)
            subtitle(" ")
            note("Source:  1988 data from NLS, U.S. Dept. of Labor,
                Bureau of Labor Statistics", span)
        )
```
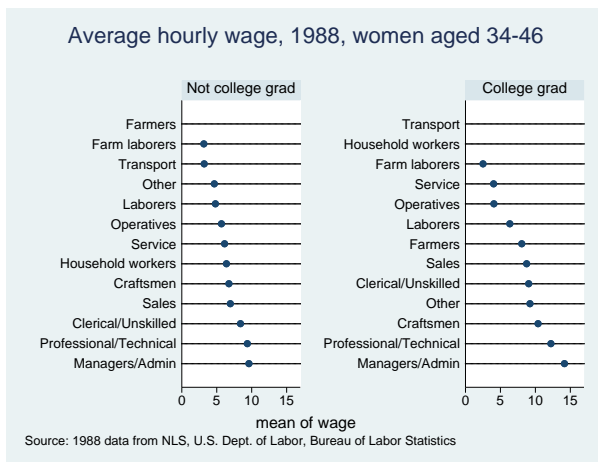

Appendix: Examples of syntax

Let us consider some graph dot commands and what they do:

graph dot revenue
   One line showing average revenue.

graph dot revenue profit
   One line with two markers, one showing average revenue and the other average profit.
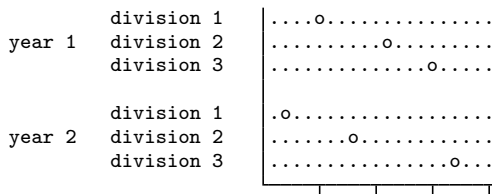
graph dot revenue, over(division)
   #_of_divisions lines, each with one marker showing average revenue for each division.

graph dot revenue profit, over(division)
   #_of_divisions lines, each with two markers, one showing average revenue and the other average profit for each division.
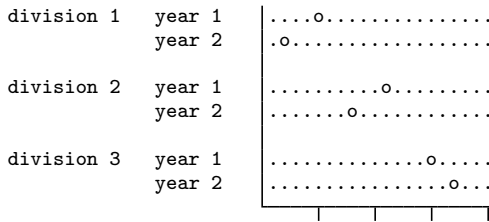
graph dot revenue, over(division) over(year)
   #_of_divisions × #_of_years lines, each with one marker showing average revenue for each division, repeated for each of the years. The grouping would look like this (assuming 3 divisions and 2 years):

```
                    division 1   |....o...............
        year 1      division 2   |.........o.........
                    division 3   |.............o.....

                    division 1   |.o.................
        year 2      division 2   |.......o...........
                    division 3   |...............o...
                                  └──┬────┬────┬────┬──
```

```
graph dot revenue, over(year) over(division)
```
Same as above, but ordered differently. In the previous example, we typed `over(division)` `over(year)`. This time, we reverse it:

```
    division 1    year 1    |....o..............
                  year 2    |.o.................

    division 2    year 1    |.........o.........
                  year 2    |.......o...........

    division 3    year 1    |..............o.....
                  year 2    |................o...
                            └────┴────┴────┴────
```

```
graph dot revenue profit, over(division) over(year)
```
#_of_divisions × *#_of_years* lines each with two markers, one showing average revenue and the other showing average profit for each division, repeated for each of the years.

```
graph dot (sum) revenue profit, over(division) over(year)
```
#_of_divisions × *#_of_years* lines each with two markers, the first showing the sum of revenue and the second showing the sum of profit for each division, repeated for each of the years.

```
graph dot (median) revenue profit, over(division) over(year)
```
#_of_divisions × *#_of_years* lines each with two markers showing the median of revenue and median of profit for each division, repeated for each of the years.

```
graph dot (median) revenue (mean) profit, over(division) over(year)
```
#_of_divisions × *#_of_years* lines each with two markers showing the median of revenue and mean of profit for each division, repeated for each of the years.

References

Cleveland, W. S. 1993. *Visualizing Data*. Summit, NJ: Hobart.

———. 1994. *The Elements of Graphing Data*. Rev. ed. Summit, NJ: Hobart.

Cox, N. J. 2008. Speaking Stata: Between tables and graphs. *Stata Journal* 8: 269–289.

Robbins, N. B. 2010. Trellis display. *Wiley Interdisciplinary Reviews: Computational Statistics* 2: 600–605. https://doi.org/10.1002/wics.121.

Also see

[G-2] **graph bar** — Bar charts

[D] **collapse** — Make dataset of summary statistics