---

graph pie — Pie charts

Syntax   Menu   Description   Options
Remarks and examples   References   Also see

Syntax

Slices as totals or percentages of each variable

> <u>gr</u>aph pie *varlist* [*if*] [*in*] [*weight*] [, *options*]

Slices as totals or percentages within over() *categories*

> <u>gr</u>aph pie *varname* [*if*] [*in*] [*weight*], over(*varname*) [*options*]

Slices as frequencies within over() *categories*

> <u>gr</u>aph pie [*if*] [*in*] [*weight*], over(*varname*) [*options*]

options	Description
* over(*varname*)	slices are distinct values of *varname*
<u>missing</u>	do not ignore missing values of *varname*
<u>all</u>categories	include all categories in the dataset
cw	casewise treatment of missing values
<u>noc</u>lockwise	counterclockwise pie chart
angle0(*#*)	angle of first slice; default is angle(90)
sort	put slices in size order
sort(*varname*)	put slices in *varname* order
<u>des</u>cending	reverse default or specified order
<u>p</u>ie(...)	look of slice, including explosion
<u>p</u>label(...)	labels to appear on the slice
<u>p</u>text(...)	text to appear on the pie
<u>in</u>tensity([*]#)	color intensity of slices
<u>li</u>ne(*line_options*)	outline of slices
pcycle(*#*)	slice styles before *pstyles* recycle
legend(...)	legend explaining slices
std_options	titles, saving to disk
by(*varlist*, ...)	repeat for subgroups

* over(*varname*) is required in syntaxes 2 and 3.

See [G-3] *line_options*, [G-3] *legend_options*, [G-3] *std_options*, and [G-3] *by_option*.

The syntax of the pie() option is

> <u>pie</u>({*numlist* | _all} [, *pie_subopts*])

pie_subopts	Description
explode	explode slice by *relativesize* = 3.8
explode(*relativesize*)	explode slice by *relativesize*
<u>c</u>olor(*colorstyle*)	color of slice

See [G-4] *relativesize* and [G-4] *colorstyle*.

The syntax of the plabel() option is

> <u>plabel</u>({#| _all} {sum | <u>percent</u> | name | "*text*"} [, *plabel_subopts*])

plabel_subopts	Description
<u>for</u>mat(%*fmt*)	display format for sum or percent
gap(*relativesize*)	additional radial distance
textbox_options	look of label

See [G-4] *relativesize* and [G-3] *textbox_options*.

The syntax for the ptext() option is

> <u>ptext</u>($\#_a\#_r$ "*text*" ["*text*" ...] [$\#_a \ \#_r$ ...] [, *ptext_subopts*])

ptext_subopts	Description
textbox_options	look of added text

See [G-3] *textbox_options*.

aweights, fweights, and pweights are allowed; see [U] **11.1.6 weight**.

Menu

Graphics > Pie chart

Description

graph pie draws pie charts.

graph pie has three modes of operation. The first corresponds to the specification of two or more variables:

    . graph pie div1_revenue div2_revenue div3_revenue

Three pie slices are drawn, the first corresponding to the sum of variable div1_revenue, the second to the sum of div2_revenue, and the third to the sum of div3_revenue.

The second mode of operation corresponds to the specification of one variable and the over() option:

    . graph pie revenue, over(division)

Pie slices are drawn for each value of variable `division`; the first slice corresponds to the sum of revenue for the first division, the second to the sum of revenue for the second division, and so on.

The third mode of operation corresponds to the specification of `over()` with no variables:

```
. graph pie, over(popgroup)
```

Pie slices are drawn for each value of variable `popgroup`; the slices correspond to the number of observations in each group.

Options

over(*varname*) specifies a categorical variable to correspond to the pie slices. *varname* may be string or numeric.

missing is for use with over(); it specifies that missing values of *varname* not be ignored. Instead, separate slices are to be formed for *varname*==., *varname*==.a, ..., or *varname*=="".

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, and zero-sized slices 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().

cw specifies casewise deletion and is for use when over() is not specified. cw specifies that, in calculating the sums, observations be ignored for which any of the variables in *varlist* contain missing values. The default is to calculate sums for each variable by using all nonmissing observations.

noclockwise and angle0(#) specify how the slices are oriented on the pie. The default is to start at 12 o'clock (known as angle(90)) and to proceed clockwise.

noclockwise causes slices to be placed counterclockwise.

angle0(#) specifies the angle at which the first slice is to appear. Angles are recorded in degrees and measured in the usual mathematical way: counterclockwise from the horizontal.

sort, sort(*varname*), and descending specify how the slices are to be ordered. The default is to put the slices in the order specified; see *How slices are ordered* under *Remarks and examples* below.

sort specifies that the smallest slice be put first, followed by the next largest, etc. See *Ordering slices by size* under *Remarks and examples* below.

sort(*varname*) specifies that the slices be put in (ascending) order of *varname*. See *Reordering the slices* under *Remarks and examples* below.

descending, which may be specified whether or not sort or sort(*varname*) is specified, reverses the order.

pie({*numlist* | _all}, *pie_subopts*) specifies the look of a slice or of a set of slices. This option allows you to "explode" (offset) one or more slices of the pie and to control the color of the slices. Examples include

```
. graph pie ..., ... pie(2, explode)
. graph pie ..., ... pie(2, explode color(red))
. graph pie ..., ... pie(2, explode color(red)) pie(5, explode)
```
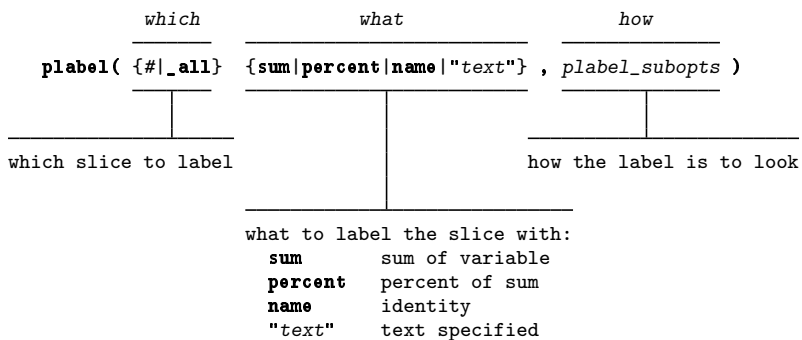
numlist specifies the slices; see [U] **11.1.8 numlist**. The slices (after any sorting) are referred to as slice 1, slice 2, etc. `pie(1 ...)` would change the look of the first slice. `pie(2 ...)` would change the look of the second slice. `pie(1 2 3 ...)` would change the look of the first through third slices, as would `pie(1/3 ...)`. The `pie()` option may be specified more than once to specify a different look for different slices. You may also specify `pie(_all ...)` to specify a common characteristic for all slices.

The *pie_subopts* are `explode`, `explode(`*relativesize*`)`, and `color(`*colorstyle*`)`.

`explode` and `explode(`*relativesize*`)` specify that the slice be offset. Specifying `explode` is equivalent to specifying `explode(3.8)`. `explode(`*relativesize*`)` specifies by how much (measured radially) the slice is to be offset; see [G-4] *relativesize*.

`color(`*colorstyle*`)` sets the color of the slice. See [G-4] *colorstyle* for a list of color choices.

`plabel(`{`#` | `_all`} {`sum` | `percent` | `name` | `"`*text*`"`}, *plabel_subopts*`)` specifies labels to appear on the slice. Slices may be labeled with their sum, their percentage of the overall sum, their identity, or with text you specify. The default is that no labels appear. Think of the syntax of `plabel()` as

```
            which              what                 how
         ─────────     ──────────────────     ─────────────
plabel(  {#|_all}     {sum|percent|name|"text"} , plabel_subopts )
         ───┬───      ──────────┬──────────     ──────┬──────
      ──────┴──────              │              ──────┴──────────
     which slice to label        │             how the label is to look
                                  │
                        ──────────┴──────────
                        what to label the slice with:
                          sum       sum of variable
                          percent   percent of sum
                          name      identity
                          "text"    text specified
```

Thus you might type

    . graph pie ..., ... plabel(_all sum)

    . graph pie ..., ... plabel(_all percent)

    . graph pie ..., ... plabel(1 "New appropriation")

The `plabel()` option may appear more than once, so you might also type

    . graph pie ..., ... plabel(1 "New appropriation") plabel(2 "old")

If you choose to label the slices with their identities, you will probably also want to suppress the legend:

    . graph pie ..., ... plabel(_all name) legend(off)

The *plabel_subopts* are `format(%fmt)`, `gap(`*relativesize*`)`, and *textbox_options*.

`format(%fmt)` specifies the display format to be used to format the number when `sum` or `percent` is chosen; see [D] **format**.

`gap(`*relativesize*`)` specifies a radial distance from the origin by which the usual location of the label is to be adjusted. `gap(0)` is the default. `gap(#)`, `#<` 0, moves the text inward. `gap(#)`, `#>` 0, moves the text outward. See [G-4] *relativesize*.

textbox_options specify the size, color, etc., of the text; see [G-3] *textbox_options*.

ptext(#$_a$ #$_r$ "text" ["text" ...] [#$_a$ #$_r$ ...], *ptext_subopts*) specifies additional text to appear on the pie. The position of the text is specified by the polar coordinates #$_a$ and #$_r$. #$_a$ specifies the angle in degrees, and #$_r$ specifies the distance from the origin in relative-size units; see [G-4] *relativesize*.

intensity(#) and intensity(*#) specify the intensity of the color used to fill the slices. intensity(#) specifies the intensity, and intensity(*#) specifies the intensity relative to the default.

Specify intensity(*#), # < 1, to attenuate the interior color and specify intensity(*#), # > 1, to amplify it.

Specify intensity(0) if you do not want the slice filled at all.

line(*line_options*) specifies the look of the line used to outline the slices. See [G-3] *line_options*, but ignore option lpattern(), which is not allowed for pie charts.

pcycle(#) specifies how many slices are to be plotted before the *pstyle* of the slices for the next slice begins again at the *pstyle* of the first slice—p1pie (with the slices following that using p2pie, p3pie, and so on). Put another way: # specifies how quickly the look of slices is recycled when more than # slices are specified. The default for most schemes is pcycle(15).

legend() allows you to control the legend. See [G-3] *legend_options*.

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

by(*varlist*, ...) draws separate pies within one graph; see [G-3] *by_option* and see *Use with by( )* under *Remarks and examples* below.

Remarks and examples   stata.com

Remarks are presented under the following headings:

> *Typical use*
> *Data are summed*
> *Data may be long rather than wide*
> *How slices are ordered*
> *Ordering slices by size*
> *Reordering the slices*
> *Use with by( )*
> *Video example*
> *History*

Typical use

We have been told that the expenditures for XYZ Corp. are $12 million in sales, $14 million in marketing, $2 million in research, and $8 million in development:

```
. input sales marketing research development
     sales  marketing   research  develop~t
  1. 12 14 2 8
  2. end
. label var sales "Sales"
. label var market "Marketing"
. label var research "Research"
. label var develop  "Development"
```

```
. graph pie sales marketing research development,
      plabel(_all name, size(*1.5) color(white))        (Note 1)
      legend(off)                                        (Note 2)
      plotregion(lstyle(none))                           (Note 3)
      title("Expenditures, XYZ Corp.")
      subtitle("2002")
      note("Source:  2002 Financial Report (fictional data)")
```
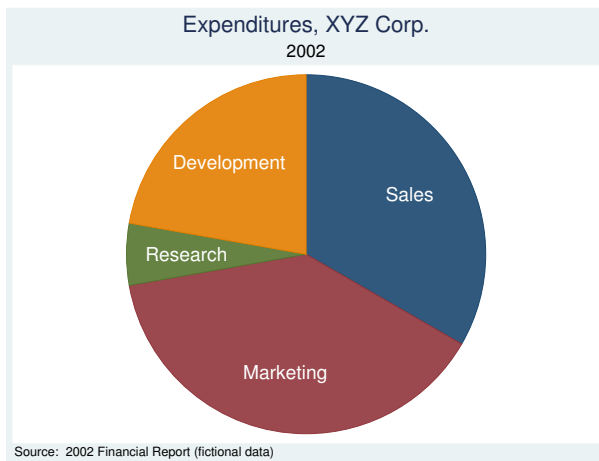

Notes:

1. We specified `plabel(_all name)` to put the division names on the slices. We specified `plabel()`'s textbox-option `size(*1.5)` to make the text 50% larger than usual. We specified `plabel()`'s textbox-option `color(white)` to make the text white. See [G-3] *textbox_options*.

2. We specified the legend-option `legend(off)` to keep the division names from being repeated in a key at the bottom of the graph; see [G-3] *legend_options*.

3. We specified the region-option `plotregion(lstyle(none))` to prevent a border from being drawn around the plot area; see [G-3] *region_options*.

Data are summed

Rather than having the above summary data, we have

```
. list
```

	qtr	sales	marketing	research	development
1.	1	3	4.5	.3	1
2.	2	4	3	.5	2
3.	3	4	4	.6	2
4.	4	2	2.5	.6	3

The sums of these data are the same as the totals in the previous section. The same `graph pie` command

```
. graph pie sales marketing research development, ...
```

will result in the same chart.

Data may be long rather than wide

Rather than having the quarterly data in wide form, we have it in the long form:

```
. list, sepby(qtr)
```

|	qtr	division	cost
1.	1	Development	1
2.	1	Marketing	4.5
3.	1	Research	.3
4.	1	Sales	3
5.	2	Development	2
6.	2	Marketing	3
7.	2	Research	.5
8.	2	Sales	4
9.	3	Development	2
10.	3	Marketing	4
11.	3	Research	.6
12.	3	Sales	3
13.	4	Development	3
14.	4	Marketing	2.5
15.	4	Research	.6
16.	4	Sales	2

Here rather than typing

```
. graph pie sales marketing research development, ...
```

we type

```
. graph pie cost, over(division) ...
```

For example,

```
. graph pie cost, over(division),
        plabel(_all name, size(*1.5) color(white))
        legend(off)
        plotregion(lstyle(none))
        title("Expenditures, XYZ Corp.")
        subtitle("2002")
        note("Source:  2002 Financial Report (fictional data)")
```
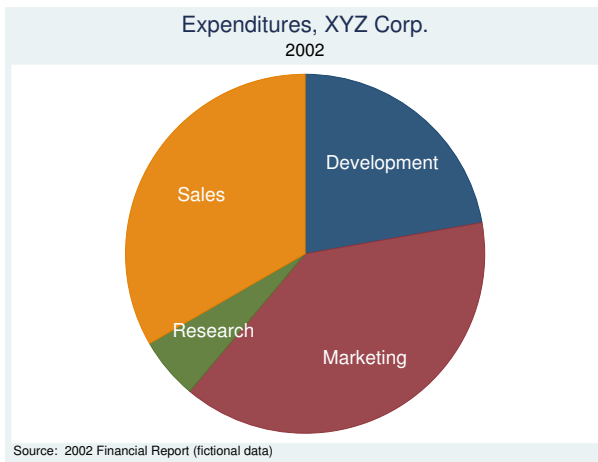

This is the same pie chart as the one drawn previously, except for the order in which the divisions are presented.

How slices are ordered

When we type

```
. graph pie sales marketing research development, ...
```

the slices are presented in the order we specify. When we type

```
. graph pie cost, over(division) ...
```

the slices are presented in the order implied by variable division. If division is numeric, slices are presented in ascending order of division. If division is string, slices are presented in alphabetical order (except that all capital letters occur before lowercase letters).

Ordering slices by size

Regardless of whether we type

```
. graph pie sales marketing research development, ...
```

or

```
. graph pie cost, over(division) ...
```

if we add the sort option, slices will be presented in the order of the size, smallest first:

```
. graph pie sales marketing research development, sort ...
. graph pie cost, over(division) sort ...
```

If we also specify the `descending` option, the largest slice will be presented first:

```
. graph pie sales marketing research development, sort descending ...
. graph pie cost, over(division) sort descending ...
```

Reordering the slices

If we wish to force a particular order, then if we type

```
. graph pie sales marketing research development, ...
```

specify the variables in the desired order. If we type

```
. graph pie cost, over(division) ...
```

then create a numeric variable that has a one-to-one correspondence with the order in which we wish the divisions to appear. For instance, we might type

```
. generate order    = 1 if division=="Sales"
. replace order = 2 if division=="Marketing"
. replace order = 3 if division=="Research"
. replace order = 4 if division=="Development"
```

then type

```
. graph pie cost, over(division) sort(order) ...
```

Use with by( )
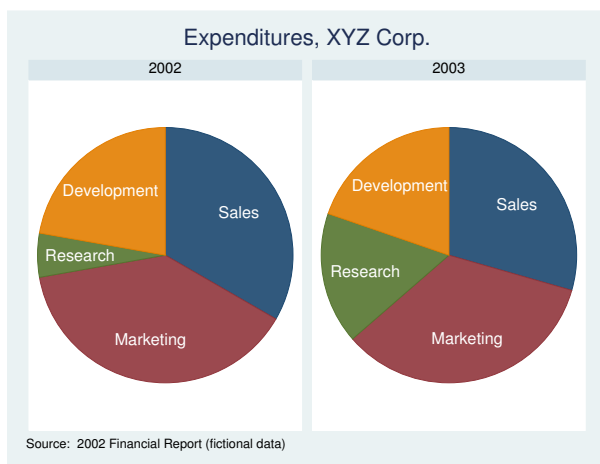
We have two years of data on XYZ Corp.:

```
. list
```

	year	sales	marketing	research	development
1.	2002	12	14	2	8
2.	2003	15	17.5	8.5	10

```
. graph pie sales marketing research development,
        plabel(_all name, size(*1.5) color(white))
        by(year,
                legend(off)
                title("Expenditures, XYZ Corp.")
                note("Source:  2002 Financial Report (fictional data)")
        )
```


Video example

Pie charts in Stata

History

The first pie chart is credited to William Playfair (1801). See Beniger and Robyn (1978), Funkhouser (1937, 283–285), or Tufte (2001, 44–45) for more historical details.

William Playfair (1759–1823) was born in Liff, Scotland. He had a varied life with many highs and lows. He participated in the storming of the Bastille, made several engineering inventions, and did path-breaking work in statistical graphics, devising bar charts and pie charts. Playfair also was involved in some shady business ventures and had to shift base from time to time. His brother John (1748–1819) was a distinguished mathematician still remembered for his discussion of Euclidean geometry and his contributions to geology.

Florence Nightingale (1820–1910) was born in Florence, Italy, to wealthy British parents who then moved to Derbyshire the following year. Perhaps best known for her pioneering work in nursing and the creation of the Nightingale School of Nurses, Nightingale also made important contributions to statistics and epidemiology. Struck by the high death toll of British soldiers in the Crimean War, she went to the medical facilities near the battlefields and determined that unsanitary conditions and widespread infections were contributing heavily to the death toll. Nightingale is known as "The Lady with the Lamp" for her habit of visiting patients in the hospitals at night. She used a form of a pie chart illustrating the causes of mortality that is now known as the polar area diagram. In one version of the diagram, each month of a year is represented by a twelfth of the circle; months with more deaths are represented by wedges with longer sides so that the area of each wedge corresponds to the number of deaths that month. After her efforts in the war, Nightingale continued to collect statistics on sanitation and mortality and to stress the important role proper hygiene plays in reducing death rates. In 1859, the compassionate statistician, as she came to be known, was inducted as the first female member of the Statistical Society.

References

Beniger, J. R., and D. L. Robyn. 1978. Quantitative graphics in statistics: A brief history. *American Statistician* 32: 1–11.

Funkhouser, H. G. 1937. Historical development of the graphical representation of statistical data. *Osiris* 3: 269–404.

Playfair, W. H. 1801. *The Statistical Breviary: Shewing, on a Principle Entirely New, the Resources of Every State and Kingdom in Europe to Which is Added, a Similar Exhibition of the Ruling Powers of Hindoostan.* London: Wallis.

——. 2005. *The Commercial and Political Atlas and Statistical Breviary.* Cambridge University Press: Cambridge.

Spence, I., and H. Wainer. 2001. William Playfair. In *Statisticians of the Centuries*, ed. C. C. Heyde and E. Seneta, 105–110. New York: Springer.

Tufte, E. R. 2001. *The Visual Display of Quantitative Information.* 2nd ed. Cheshire, CT: Graphics Press.

Also see