



The World's Largest Open Access Agricultural & Applied Economics Digital Library

This document is discoverable and free to researchers across the globe due to the work of AgEcon Search.

Help ensure our sustainability.

Give to AgEcon Search

AgEcon Search

<http://ageconsearch.umn.edu>

aesearch@umn.edu

*Papers downloaded from **AgEcon Search** may be used for non-commercial purposes and personal study only. No other use, including posting to another Internet site, is permitted without permission from the copyright owner (not AgEcon Search), or as allowed under the provisions of Fair Use, U.S. Copyright Act, Title 17 U.S.C.*

No endorsement of AgEcon Search or its fundraising activities by the author(s) of the following work or their employer(s) is intended or implied.

The Stata Journal (2017)
17, Number 3, pp. 760–773

Speaking Stata: Tables as lists: The `groups` command

Nicholas J. Cox
Department of Geography
Durham University
Durham, UK
n.j.cox@durham.ac.uk

Abstract. Tables can often be conveniently considered or produced as lists. The `list` command is therefore a vehicle for obtaining such tables. The `groups` command for tabulation is built around a call to `list`. It has no particular limits on the number of identifiers (row, column, or other variables defining cells). Among other features, it offers support for various kinds of frequencies, percents, and cumulations thereof; for various subsetting and ordering by frequencies, percents, and so on; for reordering of columns from the default display; and for saving tabulated data to new datasets.

Keywords: st0496, groups, tables, lists

1 Introduction

`groups` is a basic command for listing group frequencies and percents. An early version was described briefly in Cox (2003c). In this article, I bring the story up to date.

The main idea is that many tables are easily and helpfully presented as lists. Specifically, the `list` (see [D] `list`) command is a convenient engine for producing tables in list form. That command underlies `groups`. The idea also deserves a good push for other applications you may have.

Additionally, the opening sentence understates what `groups` can do. You can also suppress any display of frequencies and percents. That being so, `groups` can be used for almost any table; you may need to calculate beforehand what you want to tabulate, but that is generally true. All two-way and higher tabulations are reduced to one-way listings. The argument is that this constraint on layout is not so much of a loss as it may first appear.

Stata is already well endowed with a variety of tabulation commands. Offering another does need some kind of justification. For a good start, we have the official Stata commands `tabulate`, `table` (see [R] `table`), and `tabstat` (see [R] `tabstat`), not to mention special-purpose tabulation commands nor various community-contributed commands. But my experience over several years, in my own work and in answering Statalist questions, is that `groups` solves many problems not easily tackled otherwise.

Sections 2 and 3 explain `groups` informally and then more formally.

2 Examples

Some simple examples will give the basic idea. Following hallowed tradition, we start by reading in the auto data.

```
. sysuse auto, clear
(1978 Automobile Data)
```

First, we look at one-way tables. The resemblance between the results of `groups foreign` and `tabulate foreign` will be clear:

```
. groups foreign
```

foreign	Freq.	Percent	%<=
Domestic	52	70.27	70.27
Foreign	22	29.73	100.00

```
. tabulate foreign
```

Car type	Freq.	Percent	Cum.
Domestic	52	70.27	70.27
Foreign	22	29.73	100.00
Total	74	100.00	

When we turn to two-way tables, some key differences appear:

```
. groups foreign rep78
```

foreign	rep78	Freq.	Percent
Domestic	1	2	2.90
Domestic	2	8	11.59
Domestic	3	27	39.13
Domestic	4	9	13.04
Domestic	5	2	2.90
Foreign	3	3	4.35
Foreign	4	9	13.04
Foreign	5	9	13.04

```
. tabulate foreign rep78
```

Car type	Repair Record 1978				Total
	1	2	3	4	
Domestic	2	8	27	9	48
Foreign	0	0	3	9	21
Total	2	8	30	18	69

Car type	Repair Record 1978	Total
	5	
Domestic	2	48
Foreign	9	21
Total	11	69

The leading idea for two-way tables is simple. Instead of a layout of rows \times columns defining cells inside the table, **groups** gives you a listing in which each item is

row identifier, column identifier, cell variable

The example just given was chosen to make a point. Even with a simple table of two rows and five columns, **tabulate** is wrapping output awkwardly. Trivially in this case, you could just swap variables and have five rows and two columns. But it is easy to get tables that are awkward both ways. The major problem is that there are too many columns for comfort. (Too many rows for comfort is awkward with any command.)

The problem of space is usually compounded with three-way and higher tables. Even with enough space, the sparsity (many 0s) of some tables often makes other kinds of tabulation attractive. Broadly, it is with such tables with three or more variables that **groups** does especially well. A list structure is clearly general enough to extend to

one or more identifiers, one or more cell variables

In the jargon of the **reshape** (see [D] **reshape**) command, listing is thinking long, as long as is needed, not wide.

By default, **groups** will not list cross-combinations of two or more identifiers that are not present in the data. Note how in the last **groups** output, there are no rows for foreign cars with repair record 1 or 2: as is explicit in the **tabulate** output, there are no such cars. A **fillin** option is available for existentialists who like to contemplate nothingness ([Bakewell 2016](#)):

```
. groups foreign rep78, fillin
```

foreign	rep78	Freq.	Percent
Domestic	1	2	2.90
Domestic	2	8	11.59
Domestic	3	27	39.13
Domestic	4	9	13.04
Domestic	5	2	2.90
Foreign	1	0	0.00
Foreign	2	0	0.00
Foreign	3	3	4.35
Foreign	4	9	13.04
Foreign	5	9	13.04

The default for **groups** is that percents are calculated with reference to all observations analyzed. Thus, the 2.90% of observations that are domestic with repair record 1 is 2/69 (or 2.90%) of the observations that have nonmissing values on **foreign** and **rep78**. So the percents have as the base or denominator the total frequency of all observations reported in the last table.

You can override the default, and there are two ways to do that.

First, you can issue **groups** under the aegis of **by:**, which formulates output as a series of separate tables and also requests that percents be calculated separately for each table.

```
. bysort foreign: groups rep78
```

```
-> foreign = Domestic
```

rep78	Freq.	Percent	%<=
1	2	4.17	4.17
2	8	16.67	20.83
3	27	56.25	77.08
4	9	18.75	95.83
5	2	4.17	100.00

```
-> foreign = Foreign
```

rep78	Freq.	Percent	%<=
3	3	14.29	14.29
4	9	42.86	57.14
5	9	42.86	100.00

Using **by:** gives a clear separation of subtables. The downside for some purposes is that it takes up more space. That leads to the next way to get separate calculation of percents.

Second, you can use the `percentvar()` option.

```
. groups foreign rep78, percent(foreign)
```

foreign	rep78	Freq.	Percent
Domestic	1	2	4.17
Domestic	2	8	16.67
Domestic	3	27	56.25
Domestic	4	9	18.75
Domestic	5	2	4.17
Foreign	3	3	14.29
Foreign	4	9	42.86
Foreign	5	9	42.86

The `percentvar()` option lets you specify the variable or variables that determine groups for calculation of percents.

Why did cumulative percents not appear in the last output when they appeared in the one before? It is time to explain the defaults. The frequencies and percents shown by default are

1. frequencies and percents for one or more variables in *varlist*
2. cumulative percents for one variable in *varlist*

We can surmise that cumulatives are more arbitrary with two or more variables, being necessarily dependent on the order of variables. That is not the law, however, and a `show()` option allows you to have none or one or two or three of those.

```
. groups foreign rep78, percent(foreign) show(f p P)
```

foreign	rep78	Freq.	Percent	%<=
Domestic	1	2	4.17	4.17
Domestic	2	8	16.67	20.83
Domestic	3	27	56.25	77.08
Domestic	4	9	18.75	95.83
Domestic	5	2	4.17	100.00
Foreign	3	3	14.29	14.29
Foreign	4	9	42.86	57.14
Foreign	5	9	42.86	100.00

Indeed, cumulative frequencies are also available on request.

```
. groups mpg, show(f F)
```

mpg	Freq.	#<=
12	2	2
14	6	8
15	2	10
16	4	14
17	4	18
18	9	27
19	8	35
20	3	38
21	5	43
22	5	48
23	3	51
24	4	55
25	5	60
26	3	63
28	3	66
29	1	67
30	2	69
31	1	70
34	1	71
35	2	73
41	1	74

Here **f** stands for frequency and **F** stands for cumulative frequency. In addition, reverse cumulatives (**#** or **% > value** rather than **#** or **% ≤ value**) are also available. This is not a complete list, but I should mention **show(none)**, which is one of the most useful choices; more on that in a moment.

Another option, **select()**, lets you select which groups are to be listed, for example, by a condition on the frequencies. **select(f == 1)** selects those groups that occur precisely once, in which case there is no need to see a frequency column of 1s, and the percents and cumulative percents are possibly of no use or interest:

```
. groups mpg, select(f==1) show(none)
```

mpg
29
31
34
41

The `select()` option can be used in another way. `select(5)`, for example, says to list just the first five of the groups which would otherwise have been listed. By default, with just one variable specified, that is just the five lowest groups of values of the variable. Each group, naturally, could occur more than once.

```
. groups mpg, select(5)
```

mpg	Freq.	Percent	%<=
12	2	2.70	2.70
14	6	8.11	10.81
15	2	2.70	13.51
16	4	5.41	18.92
17	4	5.41	24.32

You can guess that `select(-5)` starts at the other end:

```
. groups mpg, select(-5)
```

mpg	Freq.	Percent	%<=
30	2	2.70	93.24
31	1	1.35	94.59
34	1	1.35	95.95
35	2	2.70	98.65
41	1	1.35	100.00

So these commands give you pictures of the tails of a distribution. (For single variables, `extremes` [Cox 2003a] in the Statistical Software Components archive offers another way to do something similar.)

You can specify `order(high)` or `order(low)`. In either case, that option specifies to list in order of the frequencies, not the values of the variables in each group. In the first case, `select(5)` gives you the five most frequent groups. That gives you a stab at showing especially common values, otherwise often called modes. (For another approach, see the `modes` command [Cox 1999, 2003b, 2009].)

```
. groups mpg, select(5) order(h)
```

mpg	Freq.	Percent	%<=
18	9	12.16	12.16
19	8	10.81	22.97
14	6	8.11	31.08
21	5	6.76	37.84
22	5	6.76	44.59

If you specify `fillin` with two or more variables, cross-combinations with frequencies of 0 are shown explicitly. These are in cells that would be shown as 0s by `tabulate` or as blanks by `table`. `select()`ing 0s gives you a listing of the cells *not* present in your dataset. This is not often wanted. But when it is, it can be tricky to automate, unless

you know about `fillin` (see [D] `fillin`), the command after which the option is named (see also Cox [2005]). Because such values do not exist in the dataset, we do not need to be told that their frequencies are all 0, so `show(none)` can be specified.

```
. groups foreign rep78, fillin select(f==0) show(none)
```

foreign	rep78
Foreign	1
Foreign	2

`groups` is just sitting on the shoulders of the giant `list`, so there are several ways to tweak appearances. Here is one:

```
. groups foreign rep78, sepby(foreign)
```

foreign	rep78	Freq.	Percent
Domestic	1	2	2.90
Domestic	2	8	11.59
Domestic	3	27	39.13
Domestic	4	9	13.04
Domestic	5	2	2.90
Foreign	3	3	4.35
Foreign	4	9	13.04
Foreign	5	9	13.04

We did get the same appearance earlier, but that was just fortuitous. The default of `list` separating every five lines happened to give a sensible answer.

`groups` has further unusual options for table commands. `colorder()` reorders columns in the table from what it would have been. Typically, this is an option for the second or later pass at tabulation. Suppose we want to highlight which combinations of categories are most common.

To impart a little variety, we will look at a different dataset.

```
. webuse nlswork
(National Longitudinal Survey. Young Women 14-26 years of age in 1968)
. groups collgrad not_smsa c_city south, order(high) sep(0)
```

collgrad	not_smsa	c_city	south	Freq.	Percent
0	0	0	0	5742	20.13
0	0	1	0	4941	17.32
0	1	0	1	3982	13.96
0	0	1	1	3455	12.11
0	1	0	0	3086	10.82
0	0	0	1	2527	8.86
1	0	0	0	1412	4.95
1	0	1	0	1096	3.84
1	0	1	1	698	2.45
1	0	0	1	598	2.10
1	1	0	0	566	1.98
1	1	0	1	423	1.48

We might want the frequencies and percents to be more prominent. Say that existing columns 5 and 6 should be moved to the left. It is sufficient to say `colorder(5 6)`. Specifying `5 6` is equivalent to specifying `5 6 1 2 3 4`.

```
. groups collgrad not_smsa c_city south, order(high) sep(0) colorder(5 6)
```

Freq.	Percent	collgrad	not_smsa	c_city	south
5742	20.13	0	0	0	0
4941	17.32	0	0	1	0
3982	13.96	0	1	0	1
3455	12.11	0	0	1	1
3086	10.82	0	1	0	0
2527	8.86	0	0	0	1
1412	4.95	1	0	0	0
1096	3.84	1	0	1	0
698	2.45	1	0	1	1
598	2.10	1	0	0	1
566	1.98	1	1	0	0
423	1.48	1	1	0	1

You may have noticed the `list` option `separator(0)` being used to turn off separator lines.

Another unusual option is `saving()`, which saves the data being tabulated to a new dataset. In essence (and minor details aside), they are data, at least temporarily, because otherwise `list` would not be able to show them. That should help in tabulation of forms not supported by `groups`, for graphics or other analyses, or in export to other software.

Here are some other features of **groups**:

- Support for weights, specifically frequency weights and analytical weights.
- Support for **if** and **in** as usual.
- **ge** and **lt** options allow alternative cumulations. Those with a history of Fortran programming will recognize operators, which for most computer users are now **>=** and **<**.
- **reverse** reverses output from what it would have been otherwise.
- **missing** includes observations with missing values in tabulations.
- **format()** allows specification of the display format for percents and cumulative percents. A common complaint is that table contents are spuriously precise and show too many decimal places. A less common complaint is that they show too few. This option is the answer to both problems.

3 Syntax

3.1 Syntax diagram

```
groups varlist [ if ] [ in ] [ weight ] [ , fillin ge lt missing
  select(condition| #) show(what_to_show) percentvar(varlist) format(format)
  list_options order(high|low) reverse showhead(text) colorder(integers)
  saving(filename[ , save_options]) ]
```

by: may be used with **groups**; see **help by**. This is a key to controlling how percents are calculated; that is, under **by**, percents sum to 100 within distinct categories defined by its *varlist*.

fweights and **aweights** are allowed; see **help weight**.

3.2 Description

groups lists the distinct groups of *varlist* occurring in the dataset and their frequencies or percents. **groups** is perhaps most useful with categorical variables but has other uses. Groups are presented by default in the sort order of *varlist*. There is no limit on the number of variables in *varlist*.

Frequencies are counts or other measures of abundance.

Percents are percents of each total frequency.

Cumulative frequencies and percents are cumulated in the order of groups and show frequency (percent) in each group and all earlier groups in the listing (unless the **lt** option is specified).

Reverse cumulative frequencies and percents show frequency (percent) in all later groups in the listing (unless the **ge** option is specified).

“Valid” percents are calculated relative to all pertinent nonmissing values.

3.3 Options

Specification options

fillin specifies that groups (that is, cross-combinations) of *varlist* that do not occur in the data be shown explicitly as having a frequency of 0. This has no effect with a single variable. Note that this option can backfire because the number of cross-combinations can explode combinatorially.

ge (think **g**reater than or **e**qual to) specifies that reverse frequencies and percents be calculated for the current and all later groups; that is, they are for values greater than or equal to each value.

lt (think **l**ess than) specifies that cumulative frequencies and percents be calculated only for the previous groups; that is, they are for values less than each value.

missing specifies that observations with missing values on any of the variables in *varlist* be included in the listing. By default, they are omitted. Note that “valid” percents will be the same as other percents unless the **missing** option is specified.

select(*condition* | *#*) specifies that only selected groups be listed. There are two syntaxes.

In the first syntax, selection is according to a condition imposed on the frequencies, or on the percents, or on the cumulative frequencies, or on the cumulative percents, or on the reverse cumulatives. The syntax is exemplified by

```
select(freq == 1)
select(percent > 5)
select(Percent < 50)
```

The elements **f**req, **p**ercent, **F**req, **P**ercent, **R**Freq, **R**Percent, **v**percent, **V**percent, and **r**vpercent may be abbreviated down to unambiguous abbreviations. Note that case matters in distinguishing **f**req and **F**req, **p**ercent and **P**ercent, and **v**percent and **V**percent. What follows must complete a simple true-or-false condition in Stata syntax, typically an inequality or equality.

In the second syntax, a positive or negative integer is specified. A positive integer specifies that only the first *#* groups be shown. A negative integer specifies that only the last *|#|* groups be shown.

First and last are determined with respect to the listing which would otherwise have been given.

Thus, with `order(h)`, `select(5)` shows the five groups with the five highest frequencies, while `select(-5)` shows the five groups with the five lowest frequencies, ties being broken according to the sort order of *varlist*. With `order(1)`, the opposite is true.

Without `order()`, `select(5)` shows the first five groups of *varlist*, and `select(-5)` shows the last five groups of *varlist*. The most obviously useful example is when *varlist* consists of a single variable, so the listing is of the five lowest (highest) groups of values of that variable.

`show(what_to_show)` specifies which frequencies should be shown. By default, frequencies, percents, and cumulative percents are shown with one variable, and frequencies and percents are shown with two or more variables, in that order. `show()` may be used to specify one or two or three of those, or cumulative frequencies, or reverse cumulative frequencies, or reverse cumulative percents, or equivalent percents for “valid” values, or to change the order of presentation. The elements `freq`, `percent`, `Freq`, `Percent`, `RFreq`, `RPercent`, `vpercent`, `Vpercent`, and `rvpercent` may be abbreviated down to unambiguous abbreviations. Note that case matters in distinguishing `freq` and `Freq`, `percent` and `Percent`, and `vpercent` and `Vpercent`.

`show(none)` may be used to specify that none of these should be shown. For example, with `select(f == 1)`, the frequencies would all be 1 and are thus unnecessary to display, while the percents and cumulative percents may not be of interest, so `show(none)` may be desired.

`percentvar(varlist)` specifies that percents and cumulatives be calculated with respect to the combinations of the variables specified. The results shown will resemble those with `by:`, except that the variables named are displayed within each body of results. The default is that percents and cumulatives are calculated with respect to all observations selected.

For example, the same numerical results will appear for

```
. bysort foreign: groups rep78
```

and for

```
. groups foreign rep78, percent(foreign)
```

so that, in either case, percents are calculated for groups defined by distinct categories of `foreign`.

Presentation options

`format(format)` specifies a numeric format for percent and cumulative percent frequencies. The default is `format(%6.2f)`.

list_options are options of `list`. These offer several ways to change the appearance of the listing. Note that `sum` by itself produces sums only of frequencies and percents, where shown. `sepyby()` and `separator()` are often especially helpful.

`order(high|low)` specifies that groups be listed in order of their frequencies. Ordering may be **high** (highest frequencies first) or **low** (lowest frequencies first).

reverse reverses what would otherwise be displayed, putting the last values first.

`showhead(text)` specifies alternative text for the header explaining frequency variables.

There should be as many elements as the number of frequency, percent, cumulative frequency, cumulative percent, reverse cumulative frequency, reverse cumulative percent, and valid percent variables listed, and they should occur in the same order as those variables are listed. Text containing spaces should be bound in " ". Thus, with `show(f RF)`, `showhead(# "# bigger")` specifies that frequencies are indicated by "#" and reverse cumulative frequencies are indicated by "# bigger".

`colorder(integers)` specifies a reordering of what would otherwise be shown as the columns of the listing. You may specify one or more positive integers. Suppose **groups** would show four columns, but you want the third and fourth columns to be shown first (that is, as the leftmost columns) and then the first and second columns. `colorder(3 4 1 2)` or just `colorder(3 4)` would specify that. (It follows that this option will not omit columns, although it may be used to repeat columns.) Therefore, this option typically is used on a second or later pass of **groups**.

Saving results

`saving(filename[, save_options])` specifies that the results listed be saved to a named Stata `.dta` file using **save**. That does not include any sums, means, or similar summaries. Options of **save** may be specified in the usual way. This option may not be combined with `by:.`

4 Summary

list is a valuable command used interactively. That is known to most users, who typically encounter **list** early in their Stata experience. **list** can still be underestimated even by experienced users because familiarity with its several options requires repeated study and experiment.

list is also a valuable command to be used in your own programs. Here the **groups** command provides a detailed example of a command based on **list** that may be useful for your work. You may be happy just to use the command interactively, or you may even find the code of use of interest. **groups** has grown out of practical needs over an extended period and benefits from many questions and answers on Statalist.

5 Acknowledgments

During the evolution of **groups**, I received very helpful comments from Stefan Gawrich, Roger Harbord, James Keeler, William Parry, Fred Wolfe, and Eric Zbinden.

6 References

- Bakewell, S. 2016. *At the Existentialist Café: Freedom, Being, and Apricot Cocktails*. London: Chatto & Windus.
- Cox, N. J. 1999. sg113: Tabulation of modes. *Stata Technical Bulletin* 50: 26–27. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 180–181. College Station, TX: Stata Press.
- . 2003a. extremes: Stata module to list extreme values of a variable. Statistical Software Components S430801, Department of Economics, Boston College. <http://econpapers.repec.org/software/bocbocode/s430801.htm>.
- . 2003b. Software Updates: sg113_1: Tabulation of modes. *Stata Journal* 3: 211.
- . 2003c. Speaking Stata: Problems with tables, Part II. *Stata Journal* 3: 420–439.
- . 2005. Stata tip 17: Filling in the gaps. *Stata Journal* 5: 135–136.
- . 2009. Software Updates: sg113_2: Tabulation of modes. *Stata Journal* 9: 652.

About the author

Nicholas Cox is a statistically minded geographer at Durham University. He contributes talks, postings, FAQs, and programs to the Stata user community. He has also coauthored 15 commands in official Stata. He was an author of several inserts in the *Stata Technical Bulletin* and is an editor of the *Stata Journal*. His “Speaking Stata” articles on graphics from 2004 to 2013 have been collected as *Speaking Stata Graphics* (2014, College Station, TX: Stata Press).