list — List values of variables

Description

`list` displays the values of variables. If no *varlist* is specified, the values of all the variables are displayed. Also see `browse` in [D] **edit**.

Quick start

List the data in memory

```
list
```

List only data in variables `v1`, `v2`, and `v3`

```
list v1 v2 v3
```

Same as above, but include only the first 10 observations and suppress numbering

```
list v1 v2 v3 in f/10, noobs
```

Same as above, but list the last 10 observations

```
list v1 v2 v3 in -10/l, noobs
```

Draw separator line every 10 observations, and repeat header row every 20 observations

```
list v1 v2 v3, separator(10) header(20)
```

Same as above, but also show a footer row of variable names

```
list v1 v2 v3, separator(10) header(20) footer
```

Same as above, but draw separator line between values of `v1` and do not show the header and footer

```
list v1 v2 v3, sepby(v1) noheader
```

Same as above, but draw separator line before and after observation 4, with a header

```
list v1 v2 v3, sepbyexp(_n==4)
```

Add the mean and sum of the observations at the end of the table, and suppress separator and divider lines

```
list v1 v2 v3, mean sum clean
```

Menu

Data > Describe data > List data

Syntax

list [*varlist*] [*if*] [*in*] [, *options*]

flist is equivalent to list with the fast option.

options	Description
Main	
compress	compress width of columns in both table and display formats
nocompress	use display format of each variable
fast	synonym for nocompress; no delay in output of large datasets
abbreviate(#)	abbreviate variable names to # display columns; default is ab(8)
string(#)	truncate string variables to # display columns
noobs	do not list observation numbers
fvall	display all levels of factor variables
Options	
table	force table format
display	force display format
header	display variable header once; default is table mode
noheader	suppress variable header
header(#)	display variable header every # lines
footer	display variable names as a footer
clean	force table format with no divider or separator lines
divider	draw divider lines between columns
separator(#)	draw a separator line every # lines; default is separator(5)
sepby(*varlist*$_2$)	draw a separator line whenever *varlist*$_2$ values change
sepbyexp(*exp*)	draw a separator line whenever value of *exp* changes
ds	use double-spaced lines
nolabel	display numeric codes rather than label values
Summary	
mean[(*varlist*$_2$)]	add line reporting the mean for the (specified) variables
sum[(*varlist*$_2$)]	add line reporting the sum for the (specified) variables
N[(*varlist*$_2$)]	add line reporting the number of nonmissing values for the (specified) variables
labvar(*varname*)	substitute Mean, Sum, or N for value of *varname* in added line reporting mean, sum, or N
Advanced	
constant[(*varlist*$_2$)]	separate and list variables that are constant only once
notrim	suppress string trimming
absolute	display overall observation numbers when using by *varlist*:
relative	display relative observation numbers for a subset of observations specified by qualifiers if and in
nodotz	display numerical values equal to .z as field of blanks
subvarname	substitute characteristic for variable name in header
linesize(#)	columns per line; default is linesize(79)

varlist may contain factor variables; see [U] **11.4.3 Factor variables**.
varlist may contain time-series operators; see [U] **11.4.4 Time-series varlists**.
by is allowed with list; see [D] **by**.

Options

Main

compress and nocompress change the width of the columns in both table and display formats. By default, list examines the data and allocates the needed width to each variable. For instance, a variable might be a string with a %18s format, and yet the longest string will be only 12 characters long. Or a numeric variable might have a %9.0g format, and yet, given the values actually present, the widest number needs only four columns.

nocompress prevents list from examining the data. Widths will be set according to the display format of each variable. Output generally looks better when nocompress is not specified, but for very large datasets (say, 1,000,000 observations or more), nocompress can speed up the execution of list.

compress allows list to engage in a little more compression than it otherwise would by telling list to abbreviate variable names to fewer than eight characters.

fast is a synonym for nocompress. fast may be of interest to those with very large datasets who wish to see output appear without delay.

abbreviate(#) is an alternative to compress that allows you to specify the minimum abbreviation of variable names to be considered. For example, you could specify abbreviate(16) if you never wanted variables abbreviated to less than 16 display columns. For most users, the number of display columns is equal to the number of characters. However, some languages, such as Chinese, Japanese, and Korean (CJK), require two display columns per character.

string(#) specifies that when string variables are listed, they be truncated to # display columns in the output. Any value that is truncated will be appended with "..." to indicate the truncation. string() is useful for displaying just a part of long strings.

noobs suppresses the listing of the observation numbers.

fvall specifies that the entire dataset be used to determine how many levels are in any factor variables specified in *varlist*. The default is to determine the number of levels by using only the observations in the if and in qualifiers.

Options

table and display determine the style of output. By default, list determines whether to use table or display on the basis of the width of your screen and the linesize() option, if you specify it.

table forces table format. Forcing table format when list would have chosen otherwise generally produces impossible-to-read output because of the linewraps. However, if you are logging output in SMCL format and plan to print the output on wide paper later, specifying table can be a reasonable thing to do.

display forces display format.

`header`, `noheader`, `header(#)`, and `footer` specify how the variable header or footer is to be displayed.

`header` is the default in table mode and displays the variable header once, at the top of the table.

`noheader` suppresses the header altogether.

`header(#)` redisplays the variable header every # observations. For example, `header(10)` would display a new header every 10 observations.

`footer` displays variable names as a footer. With `footer`, a variable header is also displayed; `footer` cannot be combined with `noheader`.

The default in display mode is to display the variable names interweaved with the data:

```
1.
+-------------------------------------------------------------------------+
| make          price   mpg   rep78   headroom   trunk   weight   length |
| AMC Concord   4,099    22       3        2.5      11    2,930      186 |
|-------------------------------------------------------------------------|
|    turn        displa~t        gear_r~o          foreign                |
|      40             121            3.58        Domestic                |
+-------------------------------------------------------------------------+
```

However, if you specify `header`, the header is displayed once, at the top of the table:

```
+-------------------------------------------------------------------------+
| make          price   mpg   rep78   headroom   trunk   weight   length |
|-------------------------------------------------------------------------|
|    turn        displa~t        gear_r~o          foreign                |
+-------------------------------------------------------------------------+

1.
+-------------------------------------------------------------------------+
| AMC Concord   4,099    22       3        2.5      11    2,930      186 |
|-------------------------------------------------------------------------|
|      40             121            3.58        Domestic                |
+-------------------------------------------------------------------------+
```

`clean` is a better alternative to `table` when you want to force table format and your goal is to produce more readable output on the screen. `clean` implies `table`, and it removes dividing and separating lines, which is what makes wrapped table output nearly impossible to read. Blank separator lines may be included by specifying the `ds` option.

`divider`, `separator(#)`, `sepby(`*varlist*$_2$`)`, `sepbyexp(`*exp*`)`, and `ds` specify how dividers and separator lines should be displayed. These five options affect only table format.

`divider` specifies that divider lines be drawn between columns. The default is to not display a divider.

`separator(#)` and `sepby(`*varlist*$_2$`)` indicate when separator lines should be drawn between rows. To make these separator lines blank, specify the `ds` option.

`separator(#)` specifies how often separator lines should be drawn between rows. The default is `separator(5)`, meaning every 5 observations. You may specify `separator(0)` to suppress separators altogether.

`sepby(`*varlist*$_2$`)` specifies that a separator line be drawn whenever any of the variables in `sepby(`*varlist*$_2$`)` change their values; up to 10 variables may be specified. You need not make sure the data were sorted on `sepby(`*varlist*$_2$`)` before issuing the `list` command. The variables in `sepby(`*varlist*$_2$`)` also need not be among the variables being listed.

`sepbyexp(`*exp*`)` specifies that a separator line be drawn whenever the value of *exp* changes. *exp* can be any expression and does not necessarily have to refer to the variables being listed.

ds specifies that the lines be double spaced, meaning that a blank separator line be inserted after every observation. To control when blank separator lines are inserted, specify ds with separator(#), sepby(*varlist*$_2$), or sepbyexp(*exp*).

By default, separator lines are suppressed when specifying the clean option unless ds is specified, in which case blank separator lines will be used.

nolabel specifies that numeric codes be displayed rather than the label values.

Summary

mean, sum, N, mean(*varlist*$_2$), sum(*varlist*$_2$), and N(*varlist*$_2$) all specify that lines be added to the output reporting the mean, sum, or number of nonmissing values for the (specified) variables. If you do not specify the variables, all numeric variables in the *varlist* following list are used.

labvar(*varname*) is for use with mean[()], sum[()], and N[()]. list displays Mean, Sum, or N where the observation number would usually appear to indicate the end of the table—where a row represents the calculated mean, sum, or number of observations.

labvar(*varname*) changes that. Instead, Mean, Sum, or N is displayed where the value for *varname* would be displayed. For instance, you might type

```
. list group costs profits, sum(costs profits) labvar(group)
```

	group	costs	profits
1.	1	47	5
2.	2	123	10
3.	3	22	2
	Sum	192	17

and then also specify the noobs option to suppress the observation numbers.

Advanced

constant and constant(*varlist*$_2$) specify that variables that do not vary observation by observation be separated out and listed only once.

constant specifies that list determine for itself which variables are constant.

constant(*varlist*$_2$) allows you to specify which of the constant variables you want listed separately. list verifies that the variables you specify really are constant and issues an error message if they are not.

constant and constant() respect if *exp* and in *range*. If you type

```
. list if group==3
```

variable x might be constant in the selected observations, even though the variable varies in the entire dataset.

notrim affects how string variables are listed. The default is to trim strings at the width implied by the widest possible column given your screen width (or linesize(), if you specified that). notrim specifies that strings not be trimmed. notrim implies clean (see above) and, in fact, is equivalent to the clean option, so specifying either makes no difference.

absolute affects output only when list is prefixed with by *varlist*:. Observation numbers are displayed, but the overall observation numbers are used rather than the observation numbers within each by-group. For example, if the first group had 4 observations and the second had 2, by default the observations would be numbered 1, 2, 3, 4 and 1, 2. If absolute is specified, the observations will be numbered 1, 2, 3, 4 and 5, 6.

relative affects output only when a subset of observations is listed by using qualifiers if and in. Observation numbers are displayed, but the observations are numbered 1, 2, 3, etc. When list is prefixed with by *varlist*: and relative is specified, relative observation numbers will be used for each subgroup formed by *varlist*.

nodotz is a programmer's option that specifies that numerical values equal to .z be listed as a field of blanks rather than as .z.

subvarname is a programmer's option. If a variable has the characteristic *var*[varname] set, then the contents of that characteristic will be used in place of the variable's name in the headers.

linesize(#) specifies the width of the page to be used for determining whether table or display format should be used and for formatting the resulting table. Specifying a value of linesize() that is wider than your screen width can produce truly ugly output on the screen, but that output can nevertheless be useful if you are logging output and plan to print the log later on a wide printer.

Remarks and examples

list, typed by itself, lists all the observations and variables in the dataset. If you specify *varlist*, only those variables are listed. Specifying one or both of in *range* and if *exp* limits the observations listed.

list respects line size. That is, if you resize the Results window (in windowed versions of Stata) before running list, it will take advantage of the available horizontal space. Stata for Unix(console) users can instead use the set linesize command to take advantage of this feature; see [R] **log**.

list may not display all the large strings. You have two choices: 1) you can specify the clean option, which makes a different, less attractive listing, or 2) you can increase line size, as discussed above.

▷ Example 1

`list` has two output formats, known as table and display. The table format is suitable for listing a few variables, whereas the display format is suitable for listing an unlimited number of variables. Stata chooses automatically between those two formats:

```
. use https://www.stata-press.com/data/r19/auto
(1978 automobile data)
. list in 1/2
```

```
  1. +-------------------------------------------------------------------------------+
     | make        price   mpg   rep78   headroom   trunk   weight   length       |
     | AMC Concord 4,099    22    3         2.5      11      2,930    186          |
     |-------------------------------------------------------------------------------|
     | turn        displa~t          gear_r~o          foreign                     |
     | 40          121               3.58              Domestic                    |
     +-------------------------------------------------------------------------------+

  2. +-------------------------------------------------------------------------------+
     | make        price   mpg   rep78   headroom   trunk   weight   length       |
     | AMC Pacer   4,749   17    3         3.0      11      3,350    173          |
     |-------------------------------------------------------------------------------|
     | turn        displa~t          gear_r~o          foreign                     |
     | 40          258               2.53              Domestic                    |
     +-------------------------------------------------------------------------------+
```

```
. list make mpg weight displ rep78 in 1/5
```

```
     +-------------------------------------------------+
     | make            mpg   weight   displa~t   rep78 |
     |-------------------------------------------------|
  1. | AMC Concord      22    2,930        121       3 |
  2. | AMC Pacer        17    3,350        258       3 |
  3. | AMC Spirit       22    2,640        121       . |
  4. | Buick Century    20    3,250        196       3 |
  5. | Buick Electra    15    4,080        350       4 |
     +-------------------------------------------------+
```

The first case is an example of display format; the second is an example of table format. The table format is more readable and takes less space, but it is effective only if the variables can fit on one line across the screen. Stata chose to list all 12 variables in display format, but when the *varlist* was restricted to five variables, Stata chose table format.

If you are dissatisfied with Stata's choice, you can decide for yourself. You can specify the `display` option to force display format and the `nodisplay` option to force table format.

◁

❑ Technical note

If you have long string variables in your data—say, `str75` or longer—by default, `list` displays only the first 70 or so characters of each; the exact number is determined by the width of your Results window. The first 70 or so characters will be shown followed by "…". If you need to see the entire contents of the string, you can

1. specify the `clean` option, which makes a different (and uglier) style of list, or
2. make your Results window wider [Stata for Unix(console) users: increase `set linesize`].

❑

❑ Technical note

Among the things that determine the widths of the columns, the variable names play a role. Left to itself, `list` will never abbreviate variable names to fewer than eight characters. You can use the `compress` option to abbreviate variable names to fewer characters than that.

❑

❑ Technical note

When Stata lists a string variable in table output format, the variable is displayed right-justified by default.

When Stata lists a string variable in display output format, it decides whether to display the variable right-justified or left-justified according to the display format for the string variable; see **[U] 12.5 Formats: Controlling how data are displayed**. In our previous example, `make` has a display format of `%-18s`.

```
. describe make
Variable      Storage   Display    Value
    name         type    format    label      Variable label
------------------------------------------------------------------------------
make            str18   %-18s                 Make and model
```

The negative sign in the `%-18s` instructs Stata to left-justify this variable. If the display format had been `%18s`, Stata would have right-justified the variable.

The `foreign` variable appears to be string, but if we `describe` it, we see that it is not:

```
. describe foreign
Variable      Storage   Display    Value
    name         type    format    label      Variable label
------------------------------------------------------------------------------
foreign         byte    %8.0g      origin     Car origin
```

`foreign` is stored as a `byte`, but it has an associated value label named `origin`; see **[U] 12.6.3 Value labels**. Stata decides whether to right-justify or left-justify a numeric variable with an associated value label by using the same rule used for string variables: it looks at the display format of the variable. Here the display format of `%8.0g` tells Stata to right-justify the variable. If the display format had been `%-8.0g`, Stata would have left-justified this variable.

❑

❑ Technical note

You can `list` the variables in any order. When you specify the *varlist*, `list` displays the variables in the order you specify. You may also include variables more than once in the *varlist*.

❑

Example 2

Sometimes you may wish to suppress the observation numbers. You do this by specifying the noobs option:

```
. list make mpg weight displ foreign in 46/55, noobs
```

make	mpg	weight	displa~t	foreign
Plym. Volare	18	3,330	225	Domestic
Pont. Catalina	18	3,700	231	Domestic
Pont. Firebird	18	3,470	231	Domestic
Pont. Grand Prix	19	3,210	231	Domestic
Pont. Le Mans	19	3,200	231	Domestic
Pont. Phoenix	19	3,420	231	Domestic
Pont. Sunbird	24	2,690	151	Domestic
Audi 5000	17	2,830	131	Foreign
Audi Fox	23	2,070	97	Foreign
BMW 320i	25	2,650	121	Foreign

After seeing the table, we decide that we want to separate the "Domestic" observations from the "Foreign" observations, so we specify sepby(foreign).

```
. list make mpg weight displ foreign in 46/55, noobs sepby(foreign)
```

make	mpg	weight	displa~t	foreign
Plym. Volare	18	3,330	225	Domestic
Pont. Catalina	18	3,700	231	Domestic
Pont. Firebird	18	3,470	231	Domestic
Pont. Grand Prix	19	3,210	231	Domestic
Pont. Le Mans	19	3,200	231	Domestic
Pont. Phoenix	19	3,420	231	Domestic
Pont. Sunbird	24	2,690	151	Domestic
Audi 5000	17	2,830	131	Foreign
Audi Fox	23	2,070	97	Foreign
BMW 320i	25	2,650	121	Foreign

Example 3

We want to add vertical lines in the table to separate the variables, so we specify the `divider` option. We also want to draw a horizontal line after every 2 observations, so we specify `separator(2)`.

```
. list make mpg weight displ foreign in 46/55, divider separator(2)
```

	make	mpg	weight	displa~t	foreign
46.	Plym. Volare	18	3,330	225	Domestic
47.	Pont. Catalina	18	3,700	231	Domestic
48.	Pont. Firebird	18	3,470	231	Domestic
49.	Pont. Grand Prix	19	3,210	231	Domestic
50.	Pont. Le Mans	19	3,200	231	Domestic
51.	Pont. Phoenix	19	3,420	231	Domestic
52.	Pont. Sunbird	24	2,690	151	Domestic
53.	Audi 5000	17	2,830	131	Foreign
54.	Audi Fox	23	2,070	97	Foreign
55.	BMW 320i	25	2,650	121	Foreign

After seeing the table, we decide that we do not want to abbreviate `displacement`, so we specify `abbreviate(12)`.

```
. list make mpg weight displ foreign in 46/55, divider sep(2) abbreviate(12)
```

	make	mpg	weight	displacement	foreign
46.	Plym. Volare	18	3,330	225	Domestic
47.	Pont. Catalina	18	3,700	231	Domestic
48.	Pont. Firebird	18	3,470	231	Domestic
49.	Pont. Grand Prix	19	3,210	231	Domestic
50.	Pont. Le Mans	19	3,200	231	Domestic
51.	Pont. Phoenix	19	3,420	231	Domestic
52.	Pont. Sunbird	24	2,690	151	Domestic
53.	Audi 5000	17	2,830	131	Foreign
54.	Audi Fox	23	2,070	97	Foreign
55.	BMW 320i	25	2,650	121	Foreign

❑ Technical note

You can suppress the use of value labels by specifying the `nolabel` option. For instance, the `foreign` variable in the examples above really contains numeric codes, with 0 meaning `Domestic` and 1 meaning `Foreign`. When we `list` the variable, however, we see the corresponding value labels rather than the underlying numeric code:

```
. list foreign in 51/55
```

	foreign
51.	Domestic
52.	Domestic
53.	Foreign
54.	Foreign
55.	Foreign

Specifying the `nolabel` option displays the underlying numeric codes:

```
. list foreign in 51/55, nolabel
```

	foreign
51.	0
52.	0
53.	1
54.	1
55.	1

❑

▷ Example 4

With the `separator(#)` option, a separator line is drawn every # observations. With the `sepby(`*varlist*`)` option, a separator line is drawn every time *varlist* values change. The `sepbyexp(`*exp*`)` option allows more flexible conditions for drawing a separator line: a line is drawn every time expression *exp* changes. For example, you may want a separator line whenever a string variable starts with a different letter of the alphabet:

```
. use https://www.stata-press.com/data/r19/auto
(1978 automobile data)
. sort make
. list make weight in 1/15, sepbyexp(substr(make,1,1))
```

	make	weight
1.	AMC Concord	2,930
2.	AMC Pacer	3,350
3.	AMC Spirit	2,640
4.	Audi 5000	2,830
5.	Audi Fox	2,070
6.	BMW 320i	2,650
7.	Buick Century	3,250
8.	Buick Electra	4,080
9.	Buick LeSabre	3,670
10.	Buick Opel	2,230
11.	Buick Regal	3,280
12.	Buick Riviera	3,880
13.	Buick Skylark	3,400
14.	Cad. Deville	4,330
15.	Cad. Eldorado	3,900

You can separate observations based on the value of a numerical variable, for example, weight in the thousands, two thousands, etc.:

```
. sort weight
. list make weight in 1/10, sepbyexp(floor(weight/1000))
```

	make	weight
1.	Honda Civic	1,760
2.	Plym. Champ	1,800
3.	Ford Fiesta	1,800
4.	Renault Le Car	1,830
5.	VW Rabbit	1,930
6.	Mazda GLC	1,980
7.	VW Scirocco	1,990
8.	Datsun 210	2,020
9.	VW Diesel	2,040
10.	Subaru	2,050

Observations can be delineated based on more elaborate expressions. For example, in daily time-series data, weekdays can be separated from weekend days as follows. (Function `dow()` returns the day of the week, where 0 = Sunday, 1 = Monday, …, 6 = Saturday.)

```
. use https://www.stata-press.com/data/r19/tsline2
(Simulated data of calories consumed for 365 days)
. sort day
. generate dow = dow(day)
. list day dow in 1/14, sepbyexp(dow==0 | dow==6)
```

	day	dow
1.	01jan2002	2
2.	02jan2002	3
3.	03jan2002	4
4.	04jan2002	5
5.	05jan2002	6
6.	06jan2002	0
7.	07jan2002	1
8.	08jan2002	2
9.	09jan2002	3
10.	10jan2002	4
11.	11jan2002	5
12.	12jan2002	6
13.	13jan2002	0
14.	14jan2002	1

References

Cox, N. J. 2017. Speaking Stata: Tables as lists: The groups command. *Stata Journal* 17: 760–773.

Harrison, D. A. 2006. Stata tip 34: Tabulation by listing. *Stata Journal* 6: 425–427.

Also see

[D] **edit** — Browse or edit data with Data Editor

[P] **display** — Display strings and values of scalar expressions

[P] **tabdisp** — Display tables

[R] **table** — Table of frequencies, summaries, and command results