

list — List values of variables

[Description](#)
[Options](#)

[Quick start](#)
[Remarks and examples](#)

[Menu](#)
[References](#)

[Syntax](#)
[Also see](#)

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
```

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

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

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)
```

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

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

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.

<i>options</i>	Description
Main	
<u>compress</u>	compress width of columns in both table and display formats
<u>nocompress</u>	use display format of each variable
<u>fast</u>	synonym for <u>nocompress</u> ; no delay in output of large datasets
<u>abbreviate</u> (#)	abbreviate variable names to # display columns ; default is <u>ab</u> (8)
<u>string</u> (#)	truncate string variables to # display columns
<u>noobs</u>	do not list observation numbers
<u>fvall</u>	display all levels of factor variables
Options	
<u>table</u>	force table format
<u>display</u>	force display format
<u>header</u>	display variable header once; default is table mode
<u>noheader</u>	suppress variable header
<u>header</u> (#)	display variable header every # lines
<u>clean</u>	force table format with no divider or separator lines
<u>divider</u>	draw divider lines between columns
<u>separator</u> (#)	draw a separator line every # lines; default is <u>separator</u> (5)
<u>sepy</u> (<i>varlist</i> ₂)	draw a separator line whenever <i>varlist</i> ₂ values change
<u>no</u> label	display numeric codes rather than label values
Summary	
<u>mean</u> [(<i>varlist</i> ₂)]	add line reporting the mean for the (specified) variables
<u>sum</u> [(<i>varlist</i> ₂)]	add line reporting the sum for the (specified) variables
<u>N</u> [(<i>varlist</i> ₂)]	add line reporting the number of nonmissing values for the (specified) variables
<u>labvar</u> (<i>varname</i>)	substitute Mean, Sum, or N for value of <i>varname</i> in last row of table
Advanced	
<u>constant</u> [(<i>varlist</i> ₂)]	separate and list variables that are constant only once
<u>notrim</u>	suppress string trimming
<u>absolute</u>	display overall observation numbers when using <u>by</u> <i>varlist</i> :
<u>nodotz</u>	display numerical values equal to .z as field of blanks
<u>subvarname</u>	substitute characteristic for variable name in header
<u>linesize</u> (#)	columns per line; default is <u>linesize</u> (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`, and `header(#)` specify how the variable header 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(#)` redisplay the variable header every # observations. For example, `header(10)` would display a new header every 10 observations.

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

1.

make AMC Concord	price 4,099	mpg 22	rep78 3	headroom 2.5	trunk 11	weight 2,930	length 186
turn 40	displa~t 121		gear_r~o 3.58		foreign 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 all dividing and separating lines, which is what makes wrapped table output nearly impossible to read.

`divider`, `separator(#)`, and `sepy(varlist2)` specify how dividers and separator lines should be displayed. These three options affect only table format.

`divider` specifies that divider lines be drawn between columns. The default is `nodivider`.

`separator(#)` and `sepy(varlist2)` indicate when separator lines should be drawn between rows.

`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.

`sepy(varlist2)` specifies that a separator line be drawn whenever any of the variables in `sepy(varlist2)` change their values; up to 10 variables may be specified. You need not make sure the data were sorted on `sepy(varlist2)` before issuing the `list` command. The variables in `sepy(varlist2)` also need not be among the variables being listed.

`no label` specifies that numeric codes be displayed rather than the label values.

Summary

`mean`, `sum`, `N`, `mean(varlist2)`, `sum(varlist2)`, and `N(varlist2)` 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(varlist2)` 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(varlist2)` 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.

`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

stata.com

`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 http://www.stata-press.com/data/r15/auto
(1978 Automobile Data)
. list in 1/2
```

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

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

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

	make	mpg	weight	displ~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 name	storage type	display format	value 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 name	storage type	display format	value label	variable label
foreign	byte	%8.0g	origin	Car type

`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	displ	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 `sepyby(foreign)`.

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

make	mpg	weight	displ	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	displ	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 `no label` 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



References

- Harrison, D. A. 2006. [Stata tip 34: Tabulation by listing](#). *Stata Journal* 6: 425–427.
- Lauritsen, J. M. 2001. [dm84: labjl: Adding numerical codes to value labels](#). *Stata Technical Bulletin* 59: 6–7. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 35–37. College Station, TX: Stata Press.
- Riley, A. R. 1993. [dm15: Interactively list values of variables](#). *Stata Technical Bulletin* 16: 2–6. Reprinted in *Stata Technical Bulletin Reprints*, vol. 3, pp. 37–41. College Station, TX: Stata Press.
- Royston, P., and P. D. Sasieni. 1994. [dm16: Compact listing of a single variable](#). *Stata Technical Bulletin* 17: 7–8. Reprinted in *Stata Technical Bulletin Reprints*, vol. 3, pp. 41–43. College Station, TX: Stata Press.
- Weesie, J. 1999. [dm68: Display of variables in blocks](#). *Stata Technical Bulletin* 50: 3–4. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 27–29. College Station, TX: Stata Press.

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](#) — Flexible table of summary statistics