Title

stata.com

levelsof — Distinct levels of a variable

Description Syntax Options Remarks and examples
Stored results Acknowledgments References Also see

Description

`levelsof` displays a sorted list of the distinct values of *varname*.

Syntax

`levelsof` *varname* [*if*] [*in*] [`,` *options*]

options	Description
`clean`	display string values without compound double quotes
`local(`*macname*`)`	insert the list of values in the local macro *macname*
`missing`	include missing values of *varname* in calculation
`separate(`*separator*`)`	separator to serve as punctuation for the values of returned list; default is a space
`matcell(`*matname*`)`	save frequencies of distinct values in *matname*
`matrow(`*matname*`)`	save distinct values of *varname* in *matname*
`hexadecimal`	use hexadecimal format for numerical values

`collect` is allowed; see [U] **11.1.10 Prefix commands**.

Options

`clean` displays string values without compound double quotes. By default, each distinct string value is displayed within compound double quotes, as these are the most general delimiters. If you know that the string values in *varname* do not include embedded spaces or embedded quotes, this is an appropriate option. `clean` does not affect the display of values from numeric variables.

`local(`*macname*`)` inserts the list of values in local macro *macname* within the calling program's space. Hence, that macro will be accessible after `levelsof` has finished. This is helpful for subsequent use, especially with `foreach`; see [P] **foreach**.

`missing` specifies that missing values of *varname* be included in the tabulation. The default is to exclude them.

`separate(`*separator*`)` specifies a separator to serve as punctuation for the values of the returned list. The default is a space. A useful alternative is a comma.

`matcell(`*matname*`)` saves the frequencies of the distinct values in *matname*.

`matrow(`*matname*`)` saves the distinct values of *varname* in *matname*. `matrow()` may not be specified if *varname* is a string.

`hexadecimal` specifies that hexadecimal format `%21x` be used when *varname* is numeric. See [D] **format**. This option guarantees that the values in the macro that `levelsof` creates are exactly numerically equal to their values in *varname*. For integer data, except for extremely large integers (absolute value $\geq 10^{19}$), `levelsof` always produces values that give equality without this option. For noninteger data or extremely large integers, exact numerical equality may not be true in all cases by default. Specifying `hexadecimal` guarantees equality in all cases.

Remarks and examples

stata.com

`levelsof` serves two different functions. First, it provides a compact list of the distinct values of *varname*. More commonly, it is useful when you desire to cycle through the distinct values of *varname* with (say) `foreach`; see [P] **foreach**. `levelsof` leaves behind a list in `r(levels)` that may be used in a subsequent command. When wanting to get the levels of noninteger data, one may use `matrow(`*matname*`)` to obtain the levels in full precision.

`levelsof` may hit the limits imposed by your Stata. However, it is typically used when the number of distinct values of *varname* is not extremely large.

The terminology of levels of a factor has long been standard in experimental design. See Cochran and Cox (1957, 148), Fisher (1942), or Yates (1937, 5).

▷ Example 1

```
. use https://www.stata-press.com/data/r17/auto
(1978 automobile data)
. levelsof rep78
1 2 3 4 5
. display "`r(levels)'"
1 2 3 4 5
. levelsof rep78, miss local(mylevs)
1 2 3 4 5 .
. display "`mylevs'"
1 2 3 4 5 .
. levelsof rep78, sep(,)
1,2,3,4,5
. display "`r(levels)'"
1,2,3,4,5
```

Showing value labels when defined:

```
. levelsof factor, local(levels)
. foreach l of local levels {
.        di "-> factor = `: label (factor) `l''"
.        whatever if factor == `l'
. }
```

◁

▷ Example 2

By default, `levelsof` gives exact numerical equality for all integers except extremely large ones (absolute value $\geq 10^{19}$). If you are working with integers that are not of this extreme size, `levelsof` will do what you want in all cases. You can test the equality of the macro values with the variable values using `==`. When working with nonintegers or extremely large integers, however, using `==` may not always be true when it "should" be true.

levelsof fills a macro with numbers formatted in base 10, but the values of a variable are stored in base 2. The conversion of values from base 2 to base 10 may yield an approximation where an expression using == does not evaluate to true.

Here's a simple example:

```
. clear
. set obs 5
Number of observations (_N) was 0, now 5.
. generate double x = ln(_n + 1)
. levelsof x
.6931471805599453 1.09861228866811 1.386294361119891 1.6094379124341
> 1.791759469228055
. foreach level in `r(levels)' {
  2.         count if x == `level'
  3. }
  1
  0
  0
  0
  1
```

In 3 cases out of 5, equality was not true. Using the option hexadecimal solves this problem. The macro values are formatted in base 16 using Stata's hexadecimal format %21x. See [D] **format**.

```
. levelsof x, hexadecimal
+1.62e42fefa39efX-001 +1.193ea7aad030bX+000 +1.62e42fefa39efX+000
> +1.9c041f7ed8d33X+000 +1.cab0bfa2a2002X+000
. foreach level in `r(levels)' {
  2.         display "x =" %10.0g `level'
  3.         count if x == `level'
  4. }
x = .69314718
  1
x = 1.0986123
  1
x = 1.3862944
  1
x = 1.6094379
  1
x = 1.7917595
  1
```

The only downside of using hexadecimal is that the values of the levels may be hard to read if we do not reformat them. Or we can just learn to read base 16.

◁

Stored results

levelsof stores the following in r():

Scalars

r(N)	number of observations
r(r)	number of distinct values

Macros

r(levels)	list of distinct values

Acknowledgments

The original version of `levelsof` was written by Nicholas J. Cox of the Department of Geography at Durham University, UK, who is coeditor of the *Stata Journal* and author of *Speaking Stata Graphics*. He in turn thanks Christopher F. Baum of the Department of Economics at Boston College and author of the Stata Press books *An Introduction to Modern Econometrics Using Stata* and *An Introduction to Stata Programming* and Nicholas Winter of the Department of Politics at the University of Virginia, for their input.

References

Cochran, W. G., and G. M. Cox. 1957. *Experimental Designs*. 2nd ed. New York: Wiley.

Cox, N. J. 2001. dm90: Listing distinct values of a variable. *Stata Technical Bulletin* 60: 8–11. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 46–49. College Station, TX: Stata Press.

Fisher, R. A. 1942. The theory of confounding in factorial experiments in relation to the theory of groups. *Annals of Eugenics* 11: 341–353. https://doi.org/10.1111/j.1469-1809.1941.tb02298.x.

Yates, F. 1937. *The Design and Analysis of Factorial Experiments*. Harpenden, England: Technical Communication 35, Imperial Bureau of Soil Science.

Also see

[P] **foreach** — Loop over items

[D] **codebook** — Describe data contents

[D] **format** — Set variables' output format

[D] **inspect** — Display simple summary of data's attributes

[R] **tabulate oneway** — One-way table of frequencies