

label — Manipulate labels

Description
Options
Also see

Quick start
Remarks and examples

Menu
Stored results

Syntax
References

Description

`label data` attaches a label (up to 80 characters) to the dataset in memory. Dataset labels are displayed when you use the dataset and when you `describe` it. If no label is specified, any existing label is removed.

`label variable` attaches a label (up to 80 characters) to a variable. If no label is specified, any existing variable label is removed.

`label define` defines a list of up to 65,536 associations of integers and text called value labels. Value labels are attached to variables by `label values`.

`label values` attaches a value label to *varlist*. If `.` is specified instead of *lblname*, any existing value label is detached from that *varlist*. The value label, however, is not deleted. The syntax `label values varname` (that is, nothing following the *varname*) acts the same as specifying the `..`. Value labels may be up to 32,000 characters long.

`label dir` lists the names of value labels stored in memory.

`label list` lists the names and contents of value labels stored in memory.

`label copy` makes a copy of an existing value label.

`label drop` eliminates value labels.

`label save` saves value labels in a do-file. This is particularly useful for value labels that are not attached to a variable because these labels are not saved with the data.

See [\[D\] label language](#) for information on the `label language` command.

Quick start

Label the dataset “My data”

```
label data "My data"
```

Label `v1` “First variable”

```
label variable v1 "First variable"
```

Define value label named `mylabel1`

```
label define mylabel1 1 "value 1" 2 "value 2"
```

Add value labels for 0 and 3 to `mylabel1`

```
label define mylabel1 0 "value 0" 3 "value 3", add
```

Copy `mylabel1` to `mylabel2`

```
label copy mylabel1 mylabel2
```

Redefine value 0 in mylabel2 to mean “Null”

```
label define mylabel2 0 "Null", modify
```

Apply value label mylabel1 to v1

```
label values v1 mylabel1
```

Save all currently defined value labels to mylabels.do for use with other datasets

```
label save using mylabels.do
```

List names and contents of all value labels

```
label list
```

Drop all value labels

```
label drop _all
```

Menu

label data

Data > Data utilities > Label utilities > Label dataset

label variable

Data > Variables Manager

label define

Data > Variables Manager

label values

Data > Variables Manager

label list

Data > Data utilities > Label utilities > List value labels

label copy

Data > Data utilities > Label utilities > Copy value labels

label drop

Data > Variables Manager

label save

Data > Data utilities > Label utilities > Save value labels as do-file

Syntax

Label dataset

```
label data ["label"]
```

Label variable

```
label variable varname ["label"]
```

Define value label

```
label define lblname # "label" [# "label" ...] [, add modify replace nofix]
```

Assign value label to variables

```
label values varlist [lblname | .] [, nofix]
```

List names of value labels

```
label dir
```

List names and contents of value labels

```
label list [lblname [lblname ...]]
```

Copy value labels

```
label copy lblname lblname [, replace]
```

Drop value labels

```
label drop { lblname [lblname ...] | _all }
```

Save value labels in do-file

```
label save [lblname [lblname ...]] using filename [, replace]
```

Labels for variables and values in multiple languages

```
label language ... (see \[D\] label language)
```

where # is an integer or an extended missing value (.a, .b, ..., .z).

Options

add allows you to add # ↔ *label* correspondences to *lblname*. If add is not specified, you may create only new *lblnames*. If add is specified, you may create new *lblnames* or add new entries to existing *lblnames*.

`modify` allows you to modify or delete existing #↔*label* correspondences and add new correspondences. Specifying `modify` implies `add`, even if you do not type the `add` option.

`replace`, with `label define`, allows an existing value label to be redefined. `replace`, with `label copy`, allows an existing value label to be copied over. `replace`, with `label save`, allows *filename* to be replaced.

`nofix` prevents display formats from being widened according to the maximum length of the value label. Consider `label` values `myvar mylab`, and say that `myvar` has a `%9.0g` display format right now. Say that the maximum length of the strings in `mylab` is 12 characters. `label` values would change the format of `myvar` from `%9.0g` to `%12.0g`. `nofix` prevents this.

`nofix` is also allowed with `label define`, but it is relevant only when you are modifying an existing value label. Without the `nofix` option, `label define` finds all the variables that use this value label and considers widening their display formats. `nofix` prevents this.

Remarks and examples

[stata.com](http://www.stata.com)

See [U] [12.6 Dataset, variable, and value labels](#) for a complete description of labels. This entry deals only with details not covered there.

Remarks are presented under the following headings:

[Overview](#)

[Video examples](#)

Overview

`label dir` lists the names of all defined value labels. `label list` displays the contents of a value label.

► Example 1

Although `describe` shows the names of the value labels, those value labels may not exist. Stata does not consider it an error to label the values of a variable with a nonexistent label. When this occurs, Stata still shows the association on `describe` but otherwise acts as if the variable's values are unlabeled. This way, you can associate a value label name with a variable before creating the corresponding label. Similarly, you can define labels that you have not yet used.

```
. use http://www.stata-press.com/data/r15/hbp4
. describe
Contains data from http://www.stata-press.com/data/r15/hbp4.dta
  obs:      1,130
  vars:      7                22 Jan 2016 11:12
  size:     19,210
```

variable name	storage type	display format	value label	variable label
<code>id</code>	<code>str10</code>	<code>%10s</code>		Record identification number
<code>city</code>	<code>byte</code>	<code>%8.0g</code>		
<code>year</code>	<code>int</code>	<code>%8.0g</code>		
<code>age_grp</code>	<code>byte</code>	<code>%8.0g</code>		
<code>race</code>	<code>byte</code>	<code>%8.0g</code>		
<code>hbp</code>	<code>byte</code>	<code>%8.0g</code>		
<code>female</code>	<code>byte</code>	<code>%8.0g</code>	<code>sex1b1</code>	

Sorted by:

The dataset is using the value label `sexlbl`. Let's define the value label `yesno`:

```
. label define yesno 0 "no" 1 "yes"
```

`label dir` shows you the labels that you have actually defined:

```
. label dir
yesno
sexlbl
```

We have two value labels stored in memory: `yesno` and `sexlbl`.

We can display the contents of a value label with the `label list` command:

```
. label list yesno
yesno:
      0 no
      1 yes
```

The value label `yesno` labels the values 0 as `no` and 1 as `yes`.

If you do not specify the name of the value label on the `label list` command, Stata lists all the value labels:

```
. label list
yesno:
      0 no
      1 yes
sexlbl:
      0 male
      1 female
```

◀

□ Technical note

Because Stata can have more value labels stored in memory than are actually used in the dataset, you may wonder what happens when you `save` the dataset. Stata stores only those value labels actually associated with variables.

When you use a dataset, Stata eliminates all the value labels stored in memory before loading the dataset.

□

You can add new codings to an existing value label by using the `add` option with the `label define` command. You can modify existing codings by using the `modify` option. You can redefine a value label by specifying the `replace` option.

▷ Example 2

The label `yesno` codes 0 as `no` and 1 as `yes`. You might wish later to add a third coding: 2 as `maybe`. Typing `label define` with no options results in an error:

```
. label define yesno 2 maybe
label yesno already defined
r(110);
```

If you do not specify the `add`, `modify`, or `replace` options, `label define` can be used only to create *new* value labels. The `add` option lets you add codings to an existing label:

```
. label define yesno 2 maybe, add
. label list yesno
yesno:
      0 no
      1 yes
      2 maybe
```

Perhaps you have accidentally mislabeled a value. For instance, 2 may not mean “maybe” but may instead mean “don’t know”. `add` does not allow you to change an existing label:

```
. label define yesno 2 "don't know", add
invalid attempt to modify label
r(180);
```

Instead, you would specify the `modify` option:

```
. label define yesno 2 "don't know", modify
. label list yesno
yesno:
      0 no
      1 yes
      2 don't know
```

In this way, Stata attempts to protect you from yourself. If you type `label define` with no options, you can only create a new value label—you cannot accidentally change an existing one. If you specify the `add` option, you can add new labels to a label, but you cannot accidentally change any existing label. If you specify the `modify` option, which you may not abbreviate, you can change any existing label.

You can even use the `modify` option to eliminate existing labels. To do this, you map the numeric code to a *null string*, that is, `""`:

```
. label define yesno 2 "", modify
. label list yesno
yesno:
      0 no
      1 yes
```



You can eliminate entire value labels by using the `label drop` command.

▶ Example 3

We currently have two value labels stored in memory—`sex1b1` and `yesno`—as shown by the `label dir` command:

```
. label dir
yesno
sex1b1
```

The dataset that we have in memory uses only one of the labels—`sex1b1`. `describe` reports that `yesno` is not being used:

```
. describe
Contains data from http://www.stata-press.com/data/r15/hbp4.dta
  obs:      1,130
  vars:      7                22 Jan 2016 11:12
  size:     19,210
```

variable name	storage type	display format	value label	variable label
id	str10	%10s		Record identification number
city	byte	%8.0g		
year	int	%8.0g		
age_grp	byte	%8.0g		
race	byte	%8.0g		
hbp	byte	%8.0g		
female	byte	%8.0g	sex1b1	

Sorted by:

We can eliminate the `yesno` label by typing

```
. label drop yesno
. label dir
sex1b1
```

We could eliminate *all* the value labels in memory by typing

```
. label drop _all
. label dir
```

The value label `sex1b1`, which no longer exists, was associated with the variable `female`. Even after dropping the value label, `sex1b1` is still associated with the variable:

```
. describe
Contains data from http://www.stata-press.com/data/r15/hbp4.dta
  obs:      1,130
  vars:      7                22 Jan 2016 11:12
  size:     19,210
```

variable name	storage type	display format	value label	variable label
id	str10	%10s		Record identification number
city	byte	%8.0g		
year	int	%8.0g		
age_grp	byte	%8.0g		
race	byte	%8.0g		
hbp	byte	%8.0g		
female	byte	%8.0g	sex1b1	

Sorted by:

Stata does not mind if a nonexistent value label is associated with a variable. When Stata uses such a variable, it simply acts as if the variable is not labeled:

```
. list in 1/4
```

	id	city	year	age_grp	race	hbp	female
1.	8008238923	1	1993	2	2	0	1
2.	8007143470	1	1992	5	.	0	.
3.	8000468015	1	1988	4	2	0	0
4.	8006167153	1	1991	4	2	0	0

The `label save` command creates a *do-file* containing `label define` commands for each label you specify. If you do not specify the *lblnames*, all value labels are stored in the file. If you do not specify the extension for *filename*, `.do` is assumed.

▷ Example 4

`label copy` is useful when you want to create a new value label that is similar to an existing value label. For example, assume that we currently have the value label `yesno` in memory:

```
. label list yesno
yesno:
    1 yes
    2 no
```

Assume that we have some variables in our dataset coded with 1 and 2 for “yes” and “no” and that we have some other variables coded with 1 for “yes”, 2 for “no”, and 3 for “maybe”.

We could make a copy of label `yesno` and then add the new coding to that copy:

```
. label copy yesno yesnomaybe
. label define yesnomaybe 3 "maybe", add
. label list
yesnomaybe:
    1 yes
    2 no
    3 maybe
yesno:
    1 yes
    2 no
```

▷ Example 5

Labels are automatically stored with your dataset when you `save` it. Conversely, the `use` command drops all labels before loading the new dataset. You may occasionally wish to move a value label from one dataset to another. The `label save` command allows you to do this.

For example, assume that we currently have the value label `yesnomaybe` in memory:

```
. label list yesnomaybe
yesnomaybe:
    1 yes
    2 no
    3 maybe
```


We have a dataset stored on disk called `survey.dta` to which we wish to add this value label. We might use `survey` and then retype the `label define yesnomaybe` command. Retyping the label would not be too tedious here but if the value label in memory mapped, say, the 50 states of the United States, retyping it would be irksome. `label save` provides an alternative:

```
. label save yesnomaybe using ynfile
file ynfile.do saved
```

Typing `label save yesnomaybe using ynfile` caused Stata to create a do-file called `ynfile.do` containing the definition of the `yesnomaybe` label.

To see the contents of the file, we can use the `type` command:

```
. type ynfile.do
label define yesnomaybe 1 "yes", modify
label define yesnomaybe 2 "no", modify
label define yesnomaybe 3 "maybe", modify
```

We can now use our new dataset, `survey.dta`:

```
. use survey
(Household survey data)
. label dir
```

Using the new dataset causes Stata to eliminate all value labels stored in memory. The label `yesnomaybe` is now gone. Because we saved it in the file `ynfile.do`, however, we can get it back by typing either `do ynfile` or `run ynfile`. If we type `do`, we will see the commands in the file execute. If we type `run`, the file will execute silently:

```
. run ynfile
. label dir
yesnomaybe
```

The label is now restored just as if we had typed it from the keyboard.



□ Technical note

You can also use the `label save` command to more easily edit value labels. You can save a label in a file, leave Stata and use your word processor or editor to edit the label, and then return to Stata. Using `do` or `run`, you can load the edited values.



Video examples

[How to label variables](#)

[How to label the values of categorical variables](#)

Stored results

`label list` stores the following in `r()`:

Scalars

<code>r(k)</code>	number of mapped values, including missings
<code>r(min)</code>	minimum nonmissing value label
<code>r(max)</code>	maximum nonmissing value label
<code>r(hasemiss)</code>	1 if extended missing values labeled, 0 otherwise

label dir stores the following in `r()`:

Macros

`r(names)` names of value labels

References

- Gleason, J. R. 1998a. [dm56: A labels editor for Windows and Macintosh](#). *Stata Technical Bulletin* 43: 3–6. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 5–10. College Station, TX: Stata Press.
- . 1998b. [dm56.1: Update to labedit](#). *Stata Technical Bulletin* 51: 2. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, p. 15. College Station, TX: Stata Press.
- Long, J. S. 2009. *The Workflow of Data Analysis Using Stata*. College Station, TX: Stata Press.
- Weesie, J. 1997. [dm47: Verifying value label mappings](#). *Stata Technical Bulletin* 37: 7–8. Reprinted in *Stata Technical Bulletin Reprints*, vol. 7, pp. 39–40. College Station, TX: Stata Press.
- . 2005a. [Value label utilities: labeldup and labelrename](#). *Stata Journal* 5: 154–161.
- . 2005b. [Multilingual datasets](#). *Stata Journal* 5: 162–187.

Also see

- [D] [label language](#) — Labels for variables and values in multiple languages
- [D] [labelbook](#) — Label utilities
- [D] [encode](#) — Encode string into numeric and vice versa
- [D] [varmanage](#) — Manage variable labels, formats, and other properties
- [U] [12.6 Dataset, variable, and value labels](#)