Title

stata.com

labelbook — Label utilities

Description

`labelbook` displays information for the value labels specified or, if no labels are specified, all the labels in the data.

For multilingual datasets (see [D] **label language**), `labelbook` lists the variables to which value labels are attached in all defined languages.

`numlabel` prefixes numeric values to value labels. For example, a value mapping of `2 -> "catholic"` will be changed to `2 -> "2. catholic"`. See option `mask()` for the different formats. Stata commands that display the value labels also show the associated numeric values. Prefixes are removed with the `remove` option.

`uselabel` is a programmer's command that reads the value-label information from the currently loaded dataset or from an optionally specified filename.

`uselabel` creates a dataset in memory that contains only that value-label information. The new dataset has four variables named `label`, `lname`, `value`, and `trunc`; is sorted by `lname value`; and has 1 observation per mapping. Value labels can be longer than the maximum string length in Stata; see [R] **limits**. The new variable `trunc` contains `1` if the value label is truncated to fit in a string variable in the dataset created by `uselabel`.

`uselabel` complements `label, save`, which produces a text file of the value labels in a format that allows easy editing of the value-label texts.

Specifying no list or `_all` is equivalent to specifying all value labels. Value-label names may not be abbreviated or specified with wildcards.

Quick start

Codebook of all currently defined value labels

```
labelbook
```

As above, but only include labels `mylabel1`, `mylabel2`, and `mylabel3`

```
labelbook mylabel1 mylabel2 mylabel3
```

As above, and check that value labels are unique to the first 8 characters

```
labelbook mylabel1 mylabel2 mylabel3, length(8)
```

Prefix numeric values to `mylabel1` with the number separated from the text by a hyphen

```
numlabel mylabel1, add mask("# - ")
```

Remove a prefixed numeric value from a value label when the "# -" mask was used

```
numlabel mylabel1, remove mask("# - ")
```

Menu

labelbook

Data > Data utilities > Label utilities > Produce codebook of value labels

numlabel

Data > Data utilities > Label utilities > Prepend values to value labels

uselabel

Data > Data utilities > Label utilities > Create dataset from value labels

Syntax

Produce a codebook describing value labels

```
labelbook [lblname-list] [, labelbook_options]
```

Prefix numeric values to value labels

```
numlabel [lblname-list] , {add|remove} [numlabel_options]
```

Make dataset containing value-label information

```
uselabel [lblname-list] [using filename] [, clear var]
```

labelbook_options	Description
alpha	alphabetize label mappings
length(#)	check if value labels are unique to length #; default is length(12)
list(#)	list maximum of # mappings; default is list(32000)
problems	describe potential problems in a summary report
detail	do not suppress detailed report on variables or value labels

numlabel_options	Description
* add	prefix numeric values to value labels
* remove	remove numeric values from value labels
mask(*str*)	mask for formatting numeric labels; default mask is "#. "
force	force adding or removing of numeric labels
detail	provide details about value labels, where some labels are prefixed with numbers and others are not

* Either add or remove must be specified.

Options

Options are presented under the following headings:

Options for labelbook

`alpha` specifies that the list of value-label mappings be sorted alphabetically on label. The default is to sort the list on value.

`length(#)` specifies the minimum length that `labelbook` checks to determine whether shortened value labels are still unique. It defaults to 12, the width used by most Stata commands. `labelbook` also reports whether value labels are unique at their full length.

`list(#)` specifies the maximum number of value-label mappings to be listed. If a value label defines more mappings, a random subset of # mappings is displayed. By default, `labelbook` displays all mappings. `list(0)` suppresses the listing of the value-label definitions.

`problems` specifies that a summary report be produced describing potential problems that were diagnosed:

1. Value label has gaps in mapped values (for example, values 0 and 2 are labeled, while 1 is not)
2. Value label strings contain leading or trailing blanks
3. Value label contains duplicate labels, that is, there are different values that map into the same string
4. Value label contains duplicate labels at length 12
5. Value label contains numeric → numeric mappings
6. Value label contains numeric → null string mappings
7. Value label is not used by variables

`detail` may be specified only with `problems`. It specifies that the detailed report on the variables or value labels not be suppressed.

Options for numlabel

`add` specifies that numeric values be prefixed to value labels. Value labels that are already `numlabel`ed (using the same mask) are not modified.

`remove` specifies that numeric values be removed from the value labels. If you added numeric values by using a nondefault mask, you must specify the same mask to remove them. Value labels that are not `numlabel`ed or are `numlabel`ed using a different mask are not modified.

`mask(`*str*`)` specifies a mask for formatting the numeric labels. In the mask, # is replaced by the numeric label. The default mask is `"#. "` so that numeric value `3` is shown as `"3. "`. Spaces are relevant. For the mask `"[#]"`, numeric value 3 would be shown as `"[3]"`.

`force` specifies that adding or removing numeric labels be performed, even if some value labels are `numlabel`ed using the mask and others are not. Here only labels that are not `numlabel`ed will be modified.

`detail` specifies that details be provided about the value labels that are sometimes, but not always, `numlabel`ed using the mask.

Options for uselabel

`clear` permits the dataset to be created, even if the dataset already in memory has changed since it was last saved.

`var` specifies that the varlists using value label *vl* be returned in `r(`*vl*`)`.

Remarks and examples

stata.com

Remarks are presented under the following headings:

labelbook

`labelbook` produces a detailed report of the value labels in your data. You can restrict the report to a list of labels, meaning that no abbreviations or wildcards will be allowed. `labelbook` is a companion command to [D] **codebook**, which describes the data, focusing on the variables.

For multilingual datasets (see [D] **label language**), `labelbook` lists the variables to which value labels are attached in any of the languages.

▷ Example 1

We request a `labelbook` report for value labels in a large dataset on the internal organization of households. We restrict output to three value labels: `agree5` (used for five-point Likert-style items), `divlabor` (division of labor between husband and wife), and `noyes` for simple no-or-yes questions.

```
. use http://www.stata-press.com/data/r15/labelbook1
. labelbook agree5 divlabor noyes
```

```
-------------------------------------------------------------------------------
value label agree5
-------------------------------------------------------------------------------

      values                                    labels
       range:  [1,5]                    string length:  [8,11]
           N:  5                  unique at full length:  yes
        gaps:  no                   unique at length 12:  yes
 missing .*:  0                             null string:  no
                              leading/trailing blanks:  no
                                   numeric -> numeric:  no

 definition
           1   -- disagree
           2   - disagree
           3   indifferent
           4   + agree
           5   ++ agree

  variables:  rs056 rs057 rs058 rs059 rs060 rs061 rs062 rs063 rs064 rs065 rs066
              rs067 rs068 rs069 rs070 rs071 rs072 rs073 rs074 rs075 rs076 rs077
              rs078 rs079 rs080 rs081
```

```
value label divlabor
```

values		labels	
range:	[1,7]	string length:	[7,16]
N:	7	unique at full length:	yes
gaps:	no	unique at length 12:	yes
missing .*:	0	null string:	no
		leading/trailing blanks:	yes
		numeric -> numeric:	no

definition

1	wife only
2	wife >> husband
3	wife > husband
4	equally
5	husband > wife
6	husband >> wife
7	husband only

variables: hm01_a hm01_b hm01_c hm01_d hm01_e hn19 hn21 hn25_a hn25_b hn25_c hn25_d hn25_e hn27_a hn27_b hn27_c hn27_d hn27_e hn31 hn36 hn38 hn42 hn46_a hn46_b hn46_c hn46_d hn46_e ho01_a ho01_b ho01_c ho01_d ho01_e

```
value label noyes
```

values		labels	
range:	[1,2]	string length:	[2,16]
N:	4	unique at full length:	yes
gaps:	no	unique at length 12:	yes
missing .*:	2	null string:	no
		leading/trailing blanks:	no
		numeric -> numeric:	no

definition

1	no
2	yes
.a	not applicable
.b	ambiguous answer

variables: hb12 hd01_a hd01_b hd03 hd04_a hd04_b he03_a he03_b hlat hn09_b hn24_a hn34 hn49 hu05_a hu06_1c hu06_2c hx07_a hx08 hlat2 hfinish rh02 rj10_01 rk16_a rk16_b rl01 rl03 rl08_a rl08_b rl09_a rs047 rs048 rs049 rs050 rs051 rs052 rs053 rs054 rs093 rs095 rs096 rs098

The report is largely self-explanatory. Extended missing values are denoted by "`.*`". In the definition of the mappings, the leading 12 characters of longer value labels are underlined to make it easier to check that the value labels still make sense after truncation. The following example emphasizes this feature. The option `alpha` specifies that the value-label mappings be sorted in alphabetical order by the label strings rather than by the mapped values.

```
. use http://www.stata-press.com/data/r15/labelbook2
. labelbook sports, alpha
```

```
value label sports
```

```
      values                                    labels
       range:  [1,5]                  string length:  [16,23]
           N:  4                 unique at full length:  yes
        gaps:  yes                 unique at length 12:  no
  missing .*:  0                           null string:  no
                               leading/trailing blanks:  no
                                    numeric -> numeric:  no
  definition
           5   college baseball
           4   college basketball
           2   professional baseball
           1   professional basketball
   variables:  active passive
```

The report includes information about potential problems in the data. These are discussed in greater detail in the next section.

◁

Diagnosing problems

`labelbook` can diagnose a series of potential problems in the value-label mappings. `labelbook` produces warning messages for a series of problems:

1. Gaps in the labeled values (for example, values 0 and 2 are labeled, whereas 1 is not) may occur when value labels of the intermediate values have not been defined.
2. Leading or trailing blanks in the value labels may distort Stata output.
3. Stata allows you to define blank labels, that is, the mapping of a number to the empty string. Below we give you an example of the unexpected output that may result. Blank labels are most often the result of a mistaken value-label definition, for instance, the expansion of a nonexisting macro in the definition of a value label.
4. Stata does not require that the labels within each value label consist of *unique* strings, that is, that different values be mapped into different strings. For instance, you might accidentally define the value label `gender` as

   ```
   label define gender 1 female 2 female
   ```

   You will probably catch most of the problems, but in more complicated value labels, it is easy to miss the error. `labelbook` finds such problems and displays a warning.
5. Stata allows long value labels (32,000 characters), so labels can be long. However, some commands may need to display truncated value labels, typically at length 12. Consequently, even if the value labels are unique, the truncated value labels may not be, which can cause problems. `labelbook` warns you for value labels that are not unique at length 12.
6. Stata allows value labels that can be interpreted as numbers. This is sometimes useful, but it can cause highly misleading output. Think about tabulating a variable for which the associated value label incorrectly maps 1 into “2”, 2 into “3”, and 3 into “1”. `labelbook` looks for such problematic labels and warns you if they are found.

7. In Stata, value labels are defined as separate objects that can be associated with more than one variable:

```
label define labname # str # str ....
label value varname1 labname
label value varname2 labname
...
```

If you forget to associate a variable label with a variable, Stata considers the label unused and drops its definition. `labelbook` reports unused value labels so that you may fix the problem.

The related command `codebook` reports on two other potential problems concerning value labels:

a. A variable is value labeled, but some values of the variable are not labeled. You may have forgotten to define a mapping for some values, or you generated a variable incorrectly; for example, your `sex` variable has an unlabeled value 3, and you are not working in experimental genetics!

b. A variable has been associated with an undefined value label.

`labelbook` can also be invoked with the `problems` option, specifying that only a report on potential problems be displayed without the standard detailed description of the value labels.

❑ Technical note

The following two examples demonstrate some features of value labels that may be difficult to understand. In the first example, we `encode` a string variable with blank strings of various sizes; that is, we turn a string variable into a value-labeled numeric variable. Then we tabulate the generated variable.

```
. clear all
. set obs 5
number of observations (_N) was 0, now 5
. generate str10 horror = substr("      ", 1, _n)
. encode horror, gen(Ihorror)
. tabulate horror
```

horror	Freq.	Percent	Cum.
	1	20.00	20.00
	1	20.00	40.00
	1	20.00	60.00
	1	20.00	80.00
	1	20.00	100.00
Total	5	100.00	

It may look as if you have discovered a bug in Stata because there are no value labels in the first column of the table. This happened because we encoded a variable with only blank strings, so the associated value label maps integers into blank strings.

```
. label list Ihorror
Ihorror:
           1
           2
           3
           4
           5
```

In the first column of the table, `tabulate` displayed the value-label texts, just as it should. Because these texts are all blank, the first column is empty. As illustrated below, `labelbook` would have warned you about this odd value label.

Our second example illustrates what could go wrong with numeric values stored as string values. We want to turn this into a numeric variable, but we incorrectly `encode` the variable rather than using the appropriate command, `destring`.

```
. generate str10 horror2 = string(_n+1)
. encode horror2, gen(Ihorror2)
. tabulate Ihorror2
```

Ihorror2	Freq.	Percent	Cum.
2	1	20.00	20.00
3	1	20.00	40.00
4	1	20.00	60.00
5	1	20.00	80.00
6	1	20.00	100.00
Total	5	100.00	

```
. tabulate Ihorror2, nolabel
```

Ihorror2	Freq.	Percent	Cum.
1	1	20.00	20.00
2	1	20.00	40.00
3	1	20.00	60.00
4	1	20.00	80.00
5	1	20.00	100.00
Total	5	100.00	

```
. label list Ihorror2
Ihorror2:
           1 2
           2 3
           3 4
           4 5
           5 6
```

❑

`labelbook` skips the detailed descriptions of the value labels and reports only the potential problems in the value labels if the `problems` option is specified. This report would have alerted you to the problems with the value labels we just described.

```
. use http://www.stata-press.com/data/r15/data_in_trouble, clear
. labelbook, problem
   Potential problems in dataset   http://www.stata-press.com/data/r15/
> data_in_trouble.dta
               potential problem   value labels
-------------------------------------------------
             numeric -> numeric   Ihorror2
      leading or trailing blanks   Ihorror
            numeric -> null str   Ihorror
-------------------------------------------------
```

Running `labelbook, problems` and `codebook, problems` on new data might catch a series of annoying problems.

numlabel

The numlabel command allows you to prefix numeric codes to value labels. The reason you might want to do this is best seen in an example using the automobile data. First, we create a value label for the variable rep78 (repair record in 1978),

```
. use http://www.stata-press.com/data/r15/auto
(1978 Automobile Data)
. label define repair 1 "very poor" 2 "poor" 3 "medium" 4 good 5 "very good"
. label values rep78 repair
```

and tabulate it.

```
. tabulate rep78
```

Repair Record 1978	Freq.	Percent	Cum.
very poor	2	2.90	2.90
poor	8	11.59	14.49
medium	30	43.48	57.97
good	18	26.09	84.06
very good	11	15.94	100.00
Total	69	100.00	

Suppose that we want to recode the variable by joining the categories *poor* and *very poor*. To do this, we need the numerical codes of the categories, not the value labels. However, Stata does not display both the numeric codes and the value labels. We could redisplay the table with the nolabel option. The numlabel command provides a simple alternative: it modifies the value labels so that they also contain the numeric codes.

```
. numlabel, add
. tabulate rep78
```

Repair Record 1978	Freq.	Percent	Cum.
1. very poor	2	2.90	2.90
2. poor	8	11.59	14.49
3. medium	30	43.48	57.97
4. good	18	26.09	84.06
5. very good	11	15.94	100.00
Total	69	100.00	

If you do not like the way the numeric codes are formatted, you can use numlabel to change the formatting. First, we remove the numeric codes again:

```
. numlabel repair, remove
```

In this example, we specified the name of the label. If we had not typed it, numlabel would have removed the codes from all the value labels. We can include the numeric codes while specifying a mask:

```
. numlabel, add mask("[#]  ")
. tabulate rep78
 Repair Record |
          1978 |      Freq.     Percent        Cum.
---------------+-----------------------------------
[1]  very poor |          2        2.90        2.90
     [2]  poor |          8       11.59       14.49
   [3]  medium |         30       43.48       57.97
     [4]  good |         18       26.09       84.06
[5]  very good |         11       15.94      100.00
---------------+-----------------------------------
         Total |         69      100.00
```

`numlabel` prefixes rather than postfixes the value labels with numeric codes. Because value labels can be fairly long (up to 80 characters), Stata usually displays only the first 12 characters.

uselabel

`uselabel` is of interest primarily to programmers. Here we briefly illustrate it with the `auto` dataset.

▷ Example 2

```
. use http://www.stata-press.com/data/r15/auto
(1978 Automobile Data)
. uselabel
. describe
Contains data
  obs:             2
 vars:             4
 size:            32
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
lname           str6    %9s
value           byte    %10.0g
label           str8    %9s
trunc           byte    %8.0g
-------------------------------------------------------------------------------
Sorted by: lname  value
     Note: Dataset has changed since last saved.
. list
     +-----------------------------------+
     |  lname   value      label   trunc |
     |-----------------------------------|
  1. | origin       0   Domestic       0 |
  2. | origin       1    Foreign       0 |
     +-----------------------------------+
```

`uselabel` created a dataset containing the labels and values for the value label `origin`.

The maximum length of the text associated with a value label is 32,000 characters, whereas the maximum length of a string variable in a Stata dataset is 2,045. `uselabel` uses only the first 2,045 characters of the label. The `trunc` variable will record a `1` if the text was truncated for this reason.

◁

Stored results

`labelbook` stores the following in `r()`:

Macros

`r(names)`	*lblname-list*
`r(gaps)`	gaps in mapped values
`r(blanks)`	leading or trailing blanks
`r(null)`	name of value label containing null strings
`r(nuniq)`	duplicate labels
`r(nuniq_sh)`	duplicate labels at length 12
`r(ntruniq)`	duplicate labels at maximum string length
`r(notused)`	not used by any of the variables
`r(numeric)`	name of value label containing mappings to numbers

`uselabel` stores the following in `r()`:

Macros

`r(`*lblname*`)`	list of variables that use value label *lblname* (only when `var` option is specified)

Acknowledgments

`labelbook` and `numlabel` were written by Jeroen Weesie of the Department of Sociology at Utrecht University, The Netherlands. A command similar to `numlabel` was written by J. M. Lauritsen (2001) of Odense Universiteshospital, Denmark.

References

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.

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.

Also see

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

[D] **describe** — Describe data in memory or in file

[D] **ds** — List variables matching name patterns or other characteristics

[D] **encode** — Encode string into numeric and vice versa

[D] **label** — Manipulate labels

[U] 12.6 Dataset, variable, and value labels

[U] 15 Saving and printing output—log files