The Stata Journal publishes reviewed papers together with shorter notes or comments, regular columns, book reviews, and other material of interest to Stata users. Examples of the types of papers include 1) expository papers that link the use of Stata commands or programs to associated principles, such as those that will serve as tutorials for users first encountering a new field of statistics or a major new technique; 2) papers that go “beyond the Stata manual” in explaining key features or uses of Stata that are of interest to intermediate or advanced users of Stata; 3) papers that discuss new commands or Stata programs of interest either to a wide spectrum of users (e.g., in data management or graphics) or to some large segment of Stata users (e.g., in survey statistics, survival analysis, panel analysis, or limited dependent variable modeling); 4) papers analyzing the statistical properties of new or existing estimators and tests in Stata; 5) papers that could be of interest or usefulness to researchers, especially in fields that are of practical importance but are not often included in texts or other journals, such as the use of Stata in managing datasets, especially large datasets, with advice from hard-won experience; and 6) papers of interest to those teaching, including Stata with topics such as extended examples of techniques and interpretation of results, simulations of statistical concepts, and overviews of subject areas.

For more information on the Stata Journal, including information for authors, see the web page

http://www.stata-journal.com

Subscriptions are available from StataCorp, 4905 Lakeway Drive, College Station, Texas 77845, telephone 979-696-4600 or 800-STATA-PC, fax 979-696-4601, or online at

http://www.stata.com/bookstore/sj.html

Subscription rates:

Subscriptions mailed to US and Canadian addresses:

- 3-year subscription (includes printed and electronic copy) $153
- 2-year subscription (includes printed and electronic copy) $110
- 1-year subscription (includes printed and electronic copy) $ 59
- 1-year student subscription (includes printed and electronic copy) $ 35

Subscriptions mailed to other countries:

- 3-year subscription (includes printed and electronic copy) $225
- 2-year subscription (includes printed and electronic copy) $158
- 1-year subscription (includes printed and electronic copy) $ 83
- 1-year student subscription (includes printed and electronic copy) $ 59
- 3-year subscription (electronic only) $153

Back issues of the Stata Journal may be ordered online at

http://www.stata.com/bookstore/sj.html

The Stata Journal is published quarterly by the Stata Press, College Station, Texas, USA.

Address changes should be sent to the Stata Journal, StataCorp, 4905 Lakeway Drive, College Station TX 77845, USA, or email sj@stata.com.
Value label utilities: labeldup and labelrename

Jeroen Weesie
Utrecht University

Abstract. I describe two utilities dealing with value labels. labeldup reports and optionally removes duplicate value labels. labelrename renames a value label. Both utilities, of course, preserve the links between variables and value labels and support multilingual datasets.

Keywords: dm0012, labeldup, labelrename, value labels, data integrity, multilingual datasets

1 Introduction

This brief insert describes two utilities for value labels. labeldup describes duplicate value labels, e.g., two or more value labels that consist of two value-to-string mappings, namely, 0 to “no” and 1 to “yes”. labeldup optionally removes the duplicate labels, using one of the original value labels while making sure that variables are still correctly value labeled. labelrename renames a value label and updates all associations between variables and this value label.

These two utilities were developed in parallel to mlanguage, which is described in Weesie (2005). mlanguage imposes a naming convention on the value labels. The ability to rename value labels in a dataset makes it easier to match those conventions. The ability to eliminate duplicate value labels facilitates adding a set of labels in another language—obviously, you would want to translate the value label (0 → “no”, 1 → “yes”) into, say, Spanish, only once, not 100 times. Not only would you be wasting time, but it is easy to make mistakes: different translations in different copies of the value label, typos, translating the wrong labels, etc. Duplication of value labels is a form of redundancy (non-normalization) that you should avoid. A trivial reason is that datasets are larger than necessary. In most cases, you would hardly bother about this. Much more importantly, redundancy is a threat to data integrity, increases the maintenance costs of datasets, and makes it more difficult to create correct multilingual datasets, i.e., datasets with more than one defined set of labels.

These two utilities provide some support for multilingual datasets generated with the commands label language (included in the Stata update 8.1) and mlanguage (see Weesie 2005). It is possible to rename value labels that belong to a dormant (inactive) language. Duplicates among the value labels are found, irrespective of the language sets to which they belong. This is only a first step in multilingual support. Currently, no simple way is provided to rename matching value labels in multiple languages; you have to rename value labels in each of the languages “manually”. Also, no support is provided to select among duplicates only if equivalent duplications exist in each of the languages.

© 2005 StataCorp LP
2 Example

Many Stata users will sometimes face the challenge of importing a dataset stored in another format. Stat/Transfer, an independent software program from Circle Systems, can transfer a dataset between, say, SPSS's .sav format and Stata's .dta format. Many other statistical database systems support value labels, treating them as properties of variables rather than as separate objects that may be attached to variables, as Stata does. Stat/Transfer generates a Stata dataset with separate value labels for each value-labeled variable. Thus imported databases will likely have considerable value label redundancy. We will illustrate how the commands labeldup and labelrename can be used to "polish" the value labels of such an imported dataset.

The starting point of this example is an artificial Stata dataset that I constructed for this purpose from the standard automobile data.

```
. use auto_labutil, clear
   (1978 Automobile Data)
. describe
    Contains data from auto_labutil.dta
    obs: 74 1978 Automobile Data
    vars: 7 18 Apr 2005 13:00
    size: 2,072 (99.9% of memory free) (_dta has notes)
          variable name  type format     value label     variable label
    make       str17  %-17s Make and Model
    price      int    %8.0gc Price
    rep78      byte   %13.0g repair Repair Record 1978
    rep79      byte   %9.0g repair Repair Record 1979
    rep80      byte   %9.0g repair Repair Record 1980
    foreign    byte   %8.0g origin Car type
    engine     byte   %8.0g origin Engine type

 Sorted by:  foreign
. label list
 origin:
    0 Domestic
    1 Foreign
repair:
    1 very bad
    2 bad
    3 mediocre
    4 good
    5 very good
```

This is a standard Stata dataset with five value-labeled variables and two value labels. The value label repair is attached to three variables (rep78, rep79, and rep80); the value label origin is attached to two variables (engine and foreign).

Now suppose that the data are not yet in Stata format, but, say, are in a SPSS system file auto_labutil.sav. I want to convert this SPSS system file into Stata format. On my computer, I use the command-line interface of Stat/Transfer to convert the dataset
from SPSS into Stata format with the following command (the flag /y indicates the output files may overwrite existing files):

```
. shell c:\ProgramFiles\StatTransfer7\st auto_labutil.sav
> auto_labutil_from_spss.dta /y
```

I can now load and describe the data:

```
. use auto_labutil_from_spss, clear
. des
```

Contains data from auto_labutil_from_spss.dta

<table>
<thead>
<tr>
<th>obs:</th>
<th>74</th>
</tr>
</thead>
<tbody>
<tr>
<td>vars:</td>
<td>7</td>
</tr>
<tr>
<td>size:</td>
<td>2,072 (99.9% of memory free)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>variable name</th>
<th>storage type</th>
<th>display format</th>
<th>value label</th>
</tr>
</thead>
<tbody>
<tr>
<td>make</td>
<td>str17</td>
<td>%17s</td>
<td>make and model</td>
</tr>
<tr>
<td>price</td>
<td>int</td>
<td>%8.0g</td>
<td>price</td>
</tr>
<tr>
<td>rep78</td>
<td>byte</td>
<td>%8.0g</td>
<td>rep78</td>
</tr>
<tr>
<td>rep79</td>
<td>byte</td>
<td>%8.0g</td>
<td>rep79</td>
</tr>
<tr>
<td>rep80</td>
<td>byte</td>
<td>%8.0g</td>
<td>rep80</td>
</tr>
<tr>
<td>foreign</td>
<td>byte</td>
<td>%8.0g</td>
<td>foreign</td>
</tr>
<tr>
<td>engine</td>
<td>byte</td>
<td>%8.0g</td>
<td>engine</td>
</tr>
</tbody>
</table>

Sorted by:

```
. label list
```

rep78:

1 very bad
2 bad
3 mediocre
4 good
5 very good

rep79:

1 very bad
2 bad
3 mediocre
4 good
5 very good

rep80:

1 very bad
2 bad
3 mediocre
4 good
5 very good

foreign:

0 domestic
1 foreign

eengine:

0 domestic
1 foreign

I focus on the value labels. As stated before, Stat/Transfer has created five value labels named after the variables to which they are attached. Clearly the value labels `engine` and `foreign`, and similarly `rep78`, `rep79`, and `rep80`, are identically defined. In this pet example, these replications are easy to spot, and it would be easy to clean up
the data. For a big dataset with thousands of variables and thousands of value labels, such replications are much harder to track, and eliminating the duplicates requires a lot of irritating and error-prone work. Here `labeldup` automates the process.

```
. labeldup
2 sets of duplicate value labels found:
Dupset 1: engine foreign
Dupset 2: rep78 rep79 rep80
Specify option select to compress value labels using underlined labels
Specify option names() to select other value names to be retained
```

`labeldup` has indeed correctly identified the two sets of duplicate value labels. It also informs us what will happen if instructed to select unique value labels among the duplicate sets: From the first set, it will use the underlined label `engine`; from the second set, it will use the underlined `rep78`. These choices may be overruled with the option `names()`. In this example, I accept the defaults and reinvoke `labeldup` with the option `select`.

```
. labeldup, select
2 sets of duplicate value labels found:
Dupset 1: engine foreign
Dupset 2: rep78 rep79 rep80
```

```
Contains data from auto_labutil_from_spss.dta
obs: 74 vars: 7 18 Apr 2005 13:00
size: 2,072 (99.9% of memory free)
```

```
<table>
<thead>
<tr>
<th>variable name</th>
<th>storage type</th>
<th>display format</th>
<th>value label</th>
</tr>
</thead>
<tbody>
<tr>
<td>make</td>
<td>str17 %17s</td>
<td>make and model</td>
<td>make and model</td>
</tr>
<tr>
<td>price</td>
<td>int %8.0g</td>
<td>price</td>
<td>price</td>
</tr>
<tr>
<td>rep78</td>
<td>byte %8.0g</td>
<td>rep78</td>
<td>repair record 1978</td>
</tr>
<tr>
<td>rep79</td>
<td>byte %9.0g</td>
<td>rep78</td>
<td>repair record 1979</td>
</tr>
<tr>
<td>rep80</td>
<td>byte %9.0g</td>
<td>rep78</td>
<td>repair record 1980</td>
</tr>
<tr>
<td>foreign</td>
<td>byte %8.0g</td>
<td>engine</td>
<td>car type</td>
</tr>
<tr>
<td>engine</td>
<td>byte %8.0g</td>
<td>engine</td>
<td>engine type</td>
</tr>
</tbody>
</table>

Sorted by:

The variables are now correctly labeled using the two value labels named `rep78` and `engine`, but I am not satisfied. I don’t like mixing up names of variables and names of value labels. The last modification that I want to make is to rename the value labels from `rep78` to `repair` and from `engine` to `origin`, just as in the dataset that I showed you in the beginning of this insert.

```
. labelrename rep78 repair
Value label rep78 renamed to repair
value label rep78 was attached to variables rep78 rep79 rep80
```
Value label utilities

. labelrename engine origin
Value label engine renamed to origin
value label engine was attached to variables foreign engine

. des
Contains data from auto_labutil_from_spss.dta
obs: 74
vars: 7
size: 2,072 (99.9% of memory free)

<table>
<thead>
<tr>
<th>variable name</th>
<th>storage</th>
<th>display</th>
<th>value</th>
<th>variable label</th>
</tr>
</thead>
<tbody>
<tr>
<td>make</td>
<td>str17</td>
<td>%17s</td>
<td></td>
<td>make and model</td>
</tr>
<tr>
<td>price</td>
<td>int</td>
<td>%8.0g</td>
<td></td>
<td>price</td>
</tr>
<tr>
<td>rep78</td>
<td>byte</td>
<td>%9.0g</td>
<td>repair</td>
<td>repair record 1978</td>
</tr>
<tr>
<td>rep79</td>
<td>byte</td>
<td>%9.0g</td>
<td>repair</td>
<td>repair record 1979</td>
</tr>
<tr>
<td>rep80</td>
<td>byte</td>
<td>%9.0g</td>
<td>repair</td>
<td>repair record 1980</td>
</tr>
<tr>
<td>foreign</td>
<td>byte</td>
<td>%8.0g</td>
<td>origin</td>
<td>car type</td>
</tr>
<tr>
<td>engine</td>
<td>byte</td>
<td>%8.0g</td>
<td>origin</td>
<td>engine type</td>
</tr>
</tbody>
</table>

Sorted by:
. label list
origin:
  0 domestic
  1 foreign
repair:
  1 very bad
  2 bad
  3 mediocre
  4 good
  5 very good

Technical Note

The careful reader will have noticed that the capitalization of the variable and value labels is not the same as in the original data. Converting data to SPSS and back to Stata loses capitalization since SPSS does not distinguish between uppercase and lowercase letters. For the variable labels, this can be fixed relatively easily using the case-conversion function `proper()`. This function puts the first characters of words in uppercase and the other characters in lowercase. Thus we can loop over all variables, extract the variable label, convert it into the new form, and assign it as the variable label:

```
. foreach v of varlist _all {
    .   local oldlabel : variable label 'v'
    .   local newlabel = proper('"'oldlabel"')
    .   label var 'v' '="newlabel"'
    .}
```

Stata 8 introduced new inline macro expansion functions `:` and `=`, which make it possible to code this more compactly as
foreach v of varlist _all {
    local newlabel = proper("":variable label "v")
    label var "v" ""newlabel"
}.}

or even more compactly, but almost incomprehensibly, as

foreach v of varlist _all {
    label var "v" "=proper("":variable label "v")"
}.}

Now the variable labels have the appropriate capitalization. There is no comparable method to convert the case of value labels. We have to accept the lowercase labels, unless we are willing to do some intricate programming via uselabel.

3 The commands

3.1 The command labeldup

Syntax

labeldup [labellist1] [, select names(labellist2) nodrop]

Description

labeldup reports, and optionally removes, duplicate value labels among the value labels in labellist1 or in all value labels if no labellist1 is specified. Duplicate value labels consists of identical value-to-text mappings, e.g., two value labels A and B that both map 0 to “no” and 1 to “yes” (and nothing else). labeldup reports such duplicate value labels. It can also compress the dataset, using one value label rather than multiple labels. Links between variables and value labels will, of course, be preserved, even in languages that are inactive (see the description of mlanguage in Weesie [2005]).

Options

select specifies that duplicate value labels be removed, using the value label names that come first alphabetically. For instance, if value labels B, C, and D are duplicates, the name B is selected. Among the duplicate value labels V101, V102, and V103, the label V101 is selected. See option names() to overrule this behavior.

names(labellist2) specifies a list labellist2 of value labels that you prefer to retain as value label names; in each list of duplicate value labels, at most one of the preferred names may occur. If no preferred name is found among the duplicates, labeldup takes the first name alphabetically.
nodrop suppresses dropping value labels that are no longer used, i.e., that are not attached to a variable. In the case of multilingual datasets, a value label is not used if it is not attached to a variable either in active language or in one of the dormant languages.

3.2 The command labelrename

Syntax

labelrename oldname newname [, force]

Description

labelrename renames a value label from oldname to newname, making sure that all variables to which oldname was attached are now attached to newname.

If a value label newname already exists, Stata verifies that oldname and newname define the same set of value-to-text mappings.

labelrename supports multilingual datasets (see the description of mlanguage in Weesie [2005]). If oldname is attached to variables in other languages, they are redirected to newname as well.

Option

force attach name newname to all variables that currently use oldname, even if value label oldname has not yet been defined.

3.3 Also see

We also recommend looking at the following commands that are related to labeldup and labelrename:

<table>
<thead>
<tr>
<th>Commands</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>uselabel and labelbook</td>
<td>[D] labelbook</td>
</tr>
<tr>
<td>label</td>
<td>[D] label</td>
</tr>
<tr>
<td>label language</td>
<td>[D] label language</td>
</tr>
<tr>
<td>mlanguage</td>
<td>Weesie (2005)</td>
</tr>
</tbody>
</table>
4 References


About the Author

Jeroen Weesie is associate professor of Mathematical Sociology at the Department of Sociology at Utrecht University.