Extensions to the label commands

Daniel Klein
daniel.klein@uni-kassel.de

University of Kassel

13th German Stata Users Group meeting
IAB Nürnberg
June 26, 2015

1. **Motivation**

2. **Illustrating examples**
   - Enhancing existing label commands
   - Providing additional tools

3. **To do**

Labels

One aspect of data managment

Data management tasks

- import datasets into Stata
- combine and/or reorganize datasets
- manage variables
- manage value and variable labels
- ...

Labels in data management

- seem like a minor aspect, however ...
- `label` commands part of STATA [sic] version 1.0

Data management

Manipulating variables

Stata excels at manipulating variables

- consistent and intuitive syntax
- abbreviations and wildcard characters
- time-series operators and factor-variable notation
- change contents using expressions and functions
  - `replace`
- change contents using simple transformation rules
  - `recode`, `mvdecode`
- change names systematically
  - `rename`

Data management tasks

Manipulating labels

Manipulating labels is not as convenient

- `label` commands do have a consistent and intuitive syntax, but . . .
- neither name abbreviations nor wildcard characters
  - need to know and spell out names
- expressions or functions not applicable
  - need to change integer values and text one at a time
- no equivalent to transformation rules
- changing value label names not straight forward

Existing additions

Potentials and Problems

Quite a few user-written additions

- `labutil` (Cox 2000, SSC), `labeldup` and `labelrename` (Weesie 2005, SJ), `labelsof` (Jann 2007, SSC), `lablist` (Newson 2007, SSC), `labutil2` (Klein 2011, SSC) ...

Fully functional, however ...

- no shared naming conventions
  - sometimes not easy to locate appropriate tool for given problem
- often implement numerous (new) options
  - despite intuitive naming within given command/package great variation between (different authors') commands

The elab package

Two overall objectives

- enhance functionality of existing `label` commands
- provide additional tools for managing labels

Further objectives

- consistent syntax within the package
- maximize similarity to official Stata's syntax

Building on the work of others

- drawing heavily on ideas and concepts from
  - Nick Cox, Jeroen Weesie, Ben Jann, ...

Type `elab` instead of `label`

```
. sysuse auto
(1978 Automobile Data)

. elab list origin
origin:
           0 Domestic
           1 Foreign
```

Abbreviate value label names . . .

```
. elab list o
origin:
           0 Domestic
           1 Foreign

. elab define original ///
>         42 "foo" .a "don´t know" .b "NA"

. elab list o
o ambiguous abbreviation
r(111);
```

. . . and use wildcard characters

```
. elab list ori*n
origin:
           0 Domestic
           1 Foreign

. elab list ori~l
original:
          42 foo
          .a don´t know
          .b NA
```

Additional returned results

```
. return list

scalars:
                  r(min) =  42
                  r(max) =  42
               r(nemiss) =  2
                    r(k) =  3

macros:
                 r(name) : "original"
               r(labels) : ""foo" "don´t know" "NA""
               r(values) : "42 .a .b"
```

Specifying value label names indirectly

The `vl.` operator

```
. elab list vl.foreign
origin:
           0 Domestic
           1 Foreign
```

Subsets of integer to text mappings

Qualifiers

```
. elab copy original emiss if missing(#)

. elab list emiss
missing:
           .a don´t know
           .b NA
```

- The # character represents integer values

Changing value labels

using arithmetic expressions

```
. elab copy origin domestic

. elab replace domestic = 1 - #

. elab list origin domestic
origin:
           0 Domestic
           1 Foreign
domestic:
           0 Foreign
           1 Domestic
```

What about variable labels?

```
. elab list (foreign)
     foreign "Car type"
origin:
           0 Domestic
           1 Foreign
```

- Enclose variable names in parentheses

Changing variable labels

using string functions

```
. elab replace (foreign) = strupper(@)

. elab list (foreign)
     foreign "CAR TYPE"
origin:
           0 Domestic
           1 Foreign
```

- The `@` character represents text

Changing value labels

using transformation rules

```
. elab recode origin (0/1 = 1/0) , define(domestic2)

. return list

macros:
                r(rules) : "(0 = 1) (1 = 0)"

. elab list domestic2
domestic2:
             0 Foreign
             1 Domestic
```

Changing value label names

```
. sysuse nlsw88
(NLSW, 1988 extract)

. elab dir
occlbl
indlbl
racelbl
marlbl
gradlbl
smsalbl
unionlbl
```

Changing value label names

```
. elab rename (*lbl) (vl_*)

. elab dir
vl_union
vl_smsa
vl_grad
vl_mar
vl_race
vl_ind
vl_occ
```

To do list

much (a)do

The plan

- rewrite programming tools using more subroutines and Mata
  - current version works but neither good style nor easy to enhance and debug
- fix known bugs in some routines (e.g. `elab recode`)
- revise documentation
- release first version, hopefully until end of 2015
- and add some more bells and whistles later on
  - explicit subscripting for # and `@`
  - `ds`, `relabel` ...

To do list

much (a)do

The plan

- rewrite programming tools using more subroutines and Mata
  - current version works but neither good style nor easy to enhance and debug
- fix known bugs in some routines (e.g. `elab recode`)
- revise documentation
- release first version, hopefully until end of 2015
- and add some more bells and whistles later on
  - explicit subscripting for # and `@`
  - `ds`, `relabel` ...

For now

- Thank audience for attention
- Answer questions and remember comments and suggestions