Title

stata.com

mvencode — Change missing values to numeric values and vice versa

Description

`mvencode` changes missing values in the specified *varlist* to numeric values.

`mvdecode` changes occurrences of a numlist in the specified *varlist* to a missing-value code.

Missing-value codes may be sysmiss (`.`) and the extended missing-value codes `.a`, `.b`, ..., `.z`.

String variables in *varlist* are ignored.

Quick start

Replace all missing values in `v1` with `99`

```
mvencode v1, mv(99)
```

Replace extended missing value `.a` with `888` and `.b` with `999` in `v2`

```
mvencode v2, mv(.a=888 \ .b=999)
```

Replace `.a` with `888`, `.b` with `999`, and other missing values with `99` in numeric variables

```
mvencode _all, mv(.a=888 \ .b=999 \ else=99)
```

As above, but only for observations where `catvar` equals 1

```
mvencode _all if catvar==1, mv(.a=888 \ .b=999 \ else=99)
```

Replace `888` and `999` with system missing `.` in all numeric variables

```
mvdecode _all, mv(888 999)
```

As above, but replace `888` with `.a` and `999` with `.b`

```
mvdecode _all, mv(888=.a \ 999=.b)
```

Menu

mvencode

Data > Create or change data > Other variable-transformation commands > Change missing values to numeric

mvdecode

Data > Create or change data > Other variable-transformation commands > Change numeric values to missing

Syntax

Change missing values to numeric values

```
mvencode varlist [if] [in] , mv(# | mvc=# [\ mvc=#...] [\ else=#]) [override]
```

Change numeric values to missing values

```
mvdecode varlist [if] [in] , mv(numlist | numlist=mvc [\ numlist=mvc ...])
```

where *mvc* is one of `.` | `.a` | `.b` | ... | `.z`

Options

Main

`mv(# | mvc=# [\ mvc=#...] [\ else=#])` is required and specifies the numeric values to which the missing values are to be changed.

`mv(#)` specifies that all types of missing values be changed to #.

`mv(mvc=#)` specifies that occurrences of missing-value code *mvc* be changed to #. Multiple transformation rules may be specified, separated by a backward slash (\). The list may be terminated by the special rule `else=#`, specifying that all types of missing values not yet transformed be set to #.

Examples: `mv(9)`, `mv(.=99\.a=98\.b=97)`, `mv(.=99\ else=98)`

`mv(numlist | numlist=mvc [\ numlist=mvc ...])` is required and specifies the numeric values that are to be changed to missing values.

`mv(numlist=mvc)` specifies that the values in *numlist* be changed to missing-value code *mvc*. Multiple transformation rules may be specified, separated by a backward slash (\). See [P] **numlist** for the syntax of a numlist.

Examples: `mv(9)`, `mv(99=.\98=.a\97=.b)`, `mv(99=.\ 100/999=.a)`

`override` specifies that the protection provided by `mvencode` be overridden. Without this option, `mvencode` refuses to make the requested change if any of the numeric values are already used in the data.

Remarks and examples

stata.com

You may occasionally read data in which missing (for example, a respondent failed to answer a survey question or the data were not collected) is coded with a special numeric value. Popular codings are 9, 99, −9, −99, and the like. If missing were encoded as −99, then

```
. mvdecode _all, mv(-99)
```

would translate the special code to the Stata missing value ".". Use this command cautiously because, even if −99 were not a special code, all −99s in the data would be changed to missing.

Sometimes different codes are used to represent different reasons for missing values. For instance, 98 may be used for "refused to answer" and 99 for "not applicable". Extended missing values (`.a`, `.b`, and so on) may be used to code these differences.

```
. mvdecode _all, mv(98=.a\ 99=.b)
```

Conversely, you might need to export data to software that does not understand that "." indicates a missing value, so you might code missing with a special numeric value. To change all missings to −99, you could type

```
. mvencode _all, mv(-99)
```

To change extended missing values back to numeric values, type

```
. mvencode _all, mv(.a=98\ .b=99)
```

This would leave sysmiss and all other extended missing values unchanged. To encode in addition sysmiss . to 999 and all other extended missing values to 97, you might type

```
. mvencode _all, mv(.=999\ .a=98\ .b=99\ else=97)
```

`mvencode` will automatically recast variables upward, if necessary, so even if a variable is stored as a `byte`, its missing values can be recoded to, say, 999. Also `mvencode` refuses to make the change if # (−99 here) is already used in the data, so you can be certain that your coding is unique. You can override this feature by including the `override` option.

Be aware of another potential problem with encoding and decoding missing values: value labels are not automatically adapted to the changed codings. You have to do this yourself. For example, the value label `divlabor` maps the value 99 to the string "not applicable". You used `mvdecode` to recode 99 to `.a` for all variables that are associated with this label. To fix the value label, clear the mapping for 99 and define it again for `.a`.

```
. label define divlabor 99 "", modify
. label define divlabor .a "not applicable", add
```

▷ Example 1

Our automobile dataset contains 74 observations and 12 variables. Let's first attempt to translate the missing values in the data to 1:

```
. use http://www.stata-press.com/data/r14/auto
(1978 Automobile Data)
. mvencode _all, mv(1)
        make: string variable ignored
       rep78: already 1 in     2 observations
     foreign: already 1 in    22 observations
no action taken
r(9);
```

Our attempt failed. `mvencode` first informed us that `make` is a string variable—this is not a problem but is reported merely for our information. String variables are ignored by `mvencode`. It next informed us that `rep78` was already coded 1 in 2 observations and that `foreign` was already coded 1 in 22 observations. Thus 1 would be a poor choice for encoding missing values because, after encoding, we could not tell a real 1 from a coded missing value 1.

We could force `mvencode` to encode the data with 1, anyway, by typing `mvencode _all, mv(1) override`. That would be appropriate if the 1s in our data already represented missing data. They do not, however, so we code missing as 999:

```
. mvencode _all, mv(999)
        make: string variable ignored
       rep78: 5 missing values
```

This worked, and we are informed that the only changes necessary were to 5 observations of `rep78`.

◁

▷ Example 2

Let's now pretend that we just read in the automobile data from some raw dataset in which all the missing values were coded 999. We can convert the 999s to real missings by typing

```
. mvdecode _all, mv(999)
        make: string variable ignored
       rep78: 5 missing values
```

We are informed that `make` is a string variable, so it was ignored, and that `rep78` contained 5 observations with 999. Those observations have now been changed to contain missing.

◁

Acknowledgment

These versions of `mvencode` and `mvdecode` were written by Jeroen Weesie of the Department of Sociology at Utrecht University, The Netherlands.

Also see

[D] **generate** — Create or change contents of variable

[D] **recode** — Recode categorical variables