**Syntax**

**String variable to numeric variable**

```
encode varname [if] [in], generate(newvar) [label(name) noextend]
```

**Numeric variable to string variable**

```
decode varname [if] [in], generate(newvar) [maxlength(#)]
```

**Menu**

- **encode**
  Data > Create or change data > Other variable-transformation commands > Encode value labels from string variable

- **decode**
  Data > Create or change data > Other variable-transformation commands > Decode strings from labeled numeric variable

**Description**

`encode` creates a new variable named `newvar` based on the string variable `varname`, creating, adding to, or just using (as necessary) the value label `newvar` or, if specified, `name`. Do not use `encode` if `varname` contains numbers that merely happen to be stored as strings; instead, use `generate newvar = real(varname)` or `destring`; see [U] 23.2 Categorical string variables, String functions in [D] functions, and [D] destring.

`decode` creates a new string variable named `newvar` based on the “encoded” numeric variable `varname` and its value label.

**Options for encode**

- `generate(newvar)` is required and specifies the name of the variable to be created.
- `label(name)` specifies the name of the value label to be created or used and added to if the named value label already exists. If `label()` is not specified, `encode` uses the same name for the label as it does for the new variable.
- `noextend` specifies that `varname` not be encoded if there are values contained in `varname` that are not present in `label(name)`. By default, any values not present in `label(name)` will be added to that label.
**Options for decode**

`generate(newvar)` is required and specifies the name of the variable to be created.

`maxlength(#)` specifies how many characters of the value label to retain; `#` must be between 1 and 2045. The default is `maxlength(2045)`.

---

**Remarks and examples**

Remarks are presented under the following headings:

- `encode`
- `decode`

---

### `encode`

`encode` is most useful in making string variables accessible to Stata’s statistical routines, most of which can work only with numeric variables. `encode` is also useful in reducing the size of a dataset. If you are not familiar with value labels, read [U] 12.6.3 Value labels.

The maximum number of associations within each value label is 65,536 (1,000 for Small Stata). Each association in a value label maps a string of up to 2045 characters to a number. If your string has entries longer than that, only the first 2045 characters are retained and are significant.

---

#### Example 1

We have a dataset on high blood pressure, and among the variables is `sex`, a string variable containing either “male” or “female”. We wish to run a regression of high blood pressure on race, sex, and age group. We type `regress hbp race sex age_grp` and get the message “no observations”.

```
use http://www.stata-press.com/data/r13/hbp2
regress hbp sex race age_grp
no observations
r(2000);
```

Stata’s statistical procedures cannot directly deal with string variables; as far as they are concerned, all observations on `sex` are missing. `encode` provides the solution:

```
encode sex, gen(gender)
regress hbp gender race age_grp
```

<table>
<thead>
<tr>
<th>Source</th>
<th>SS</th>
<th>df</th>
<th>MS</th>
<th>Number of obs = 1121</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>2.01013476</td>
<td>3</td>
<td>0.67004492</td>
<td>F( 3, 1117) = 15.15</td>
</tr>
<tr>
<td>Residual</td>
<td>49.3886164</td>
<td>1117</td>
<td>0.044215413</td>
<td>Prob &gt; F = 0.0000</td>
</tr>
<tr>
<td>Total</td>
<td>51.3987511</td>
<td>1120</td>
<td>0.045891742</td>
<td>R-squared = 0.0391</td>
</tr>
</tbody>
</table>

| hbp     | Coef.  | Std. Err. | t     | P>|t|   | [95% Conf. Interval] |
|---------|--------|-----------|-------|-------|---------------------|
| gender  | 0.0394747 | 0.0130022 | 3.04  | 0.002 | 0.0139633 — 0.0649861 |
| race    | -0.0409453 | 0.0113721 | -3.60 | 0.000 | -0.0632584 — -0.0186322 |
| age_grp | 0.0241484 | 0.00624 | 3.87  | 0.000 | 0.0119049 — 0.0363919 |
| _cons   | -0.016815 | 0.0389167 | -0.43 | 0.666 | -0.093173 — 0.059543 |
**Example 2**

It is difficult to distinguish the result of `encode` from the original string variable. For instance, in our last two examples, we typed `encode sex, gen(gender)`. Let’s compare the two variables:

```stata
. list sex gender in 1/4

<table>
<thead>
<tr>
<th>sex</th>
<th>gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>female</td>
<td>female</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>male</td>
<td>male</td>
</tr>
<tr>
<td>male</td>
<td>male</td>
</tr>
</tbody>
</table>
```

They look almost identical, although you should notice the missing value for gender in the second observation.

The difference does show, however, if we tell `list` to ignore the value labels and show how the data really appear:

```stata
. list sex gender in 1/4, nolabel

<table>
<thead>
<tr>
<th>sex</th>
<th>gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>female</td>
<td>1</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>male</td>
<td>2</td>
</tr>
<tr>
<td>male</td>
<td>2</td>
</tr>
</tbody>
</table>
```

We could also ask to see the underlying value label:

```stata
. label list gender

gender: 1 female 2 male
```

`gender` really is a numeric variable, but because all Stata commands understand value labels, the variable displays as “male” and “female”, just as the underlying string variable `sex` would.

**Example 3**

We can drastically reduce the size of our dataset by encoding strings and then discarding the underlying string variable. We have a string variable, `sex`, that records each person’s sex as “male” and “female”. Because female has six characters, the variable is stored as a `str6`.

We can `encode` the `sex` variable and use `compress` to store the variable as a `byte`, which takes only 1 byte. Because our dataset contains 1,130 people, the string variable takes 6,780 bytes, but the encoded variable will take only 1,130 bytes.
. use http://www.stata-press.com/data/r13/hbp2, clear

. describe
Contains data from http://www.stata-press.com/data/r13/hbp2.dta
    obs:  1,130  3 Mar 2013 06:47
    vars:  7
    size: 24,860

storage  display  value
variable name   type format   label  variable label

id         str10   %10s  Record identification number
city       byte    %8.0g
year       int     %8.0g
age_grp    byte    %8.0g  agefmt
race       byte    %8.0g  racefmt
hbp        byte    %8.0g  yn  high blood pressure
sex        str6    %9s

Sorted by:
. encode sex, generate(gender)

. list sex gender in 1/5

<table>
<thead>
<tr>
<th>sex</th>
<th>gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>female</td>
</tr>
<tr>
<td>2.</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>male</td>
</tr>
<tr>
<td>4.</td>
<td>male</td>
</tr>
<tr>
<td>5.</td>
<td>female</td>
</tr>
</tbody>
</table>

. drop sex

. rename gender sex

. compress
    sex was long now byte
    (3,390 bytes saved)

. describe
Contains data from http://www.stata-press.com/data/r13/hbp2.dta
    obs:  1,130  3 Mar 2013 06:47
    vars:  7
    size: 19,210

storage  display  value
variable name   type format   label  variable label

id         str10   %10s  Record identification number
city       byte    %8.0g
year       int     %8.0g
age_grp    byte    %8.0g  agefmt
race       byte    %8.0g  racefmt
hbp        byte    %8.0g  yn  high blood pressure
sex        byte    %8.0g  gender

Sorted by:
    Note: dataset has changed since last saved

The size of our dataset has fallen from 24,860 bytes to 19,210 bytes.
Technical note

In the examples above, the value label did not exist before `encode` created it, because that is not required. If the value label does exist, `encode` uses your encoding as far as it can and adds new mappings for anything not found in your value label. For instance, if you wanted “female” to be encoded as 0 rather than 1 (possibly for use in linear regression), you could type

```
. label define gender 0 "female"
. encode sex, gen(gender)
```

You can also specify the name of the value label. If you do not, the value label is assumed to have the same name as the newly created variable. For instance,

```
. label define sexlbl 0 "female"
. encode sex, gen(gender) label(sexlbl)
```

decode

decode is used to convert numeric variables with associated value labels into true string variables.

Example 4

We have a numeric variable named `female` that records the values 0 and 1. `female` is associated with a value label named `sexlbl` that says that 0 means male and 1 means female:

```
. use http://www.stata-press.com/data/r13/hbp3, clear
. describe female
```

```
storage  display value
variable name   type   format   label      variable label
female         byte   %8.0g    sexlbl
```

We see that `female` is stored as a byte. It is a numeric variable. Nevertheless, it has an associated value label describing what the numeric codes mean, so if we `tabulate` the variable, for instance, it appears to contain the strings “male” and “female”:

```
. tabulate female
```

```
<table>
<thead>
<tr>
<th></th>
<th>Freq.</th>
<th>Percent</th>
<th>Cum.</th>
</tr>
</thead>
<tbody>
<tr>
<td>male</td>
<td>695</td>
<td>61.61%</td>
<td>61.61%</td>
</tr>
<tr>
<td>female</td>
<td>433</td>
<td>38.39%</td>
<td>100.00%</td>
</tr>
<tr>
<td>Total</td>
<td>1,128</td>
<td>100.00%</td>
<td></td>
</tr>
</tbody>
</table>
```

We can create a real string variable from this numerically encoded variable by using `decode`:

```
. decode female, gen(sex)
. describe sex
```

```
storage  display value
variable name   type   format   label      variable label
sex      str6    %9s
```

We have a new variable called `sex`. It is a string, and Stata automatically created the shortest possible string. The word “female” has six characters, so our new variable is a str6. `female` and `sex` appear indistinguishable:
encode — Encode string into numeric and vice versa

. list female sex in 1/4

<table>
<thead>
<tr>
<th>female</th>
<th>sex</th>
</tr>
</thead>
<tbody>
<tr>
<td>female</td>
<td>female</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>male</td>
<td>male</td>
</tr>
<tr>
<td>male</td>
<td>male</td>
</tr>
</tbody>
</table>

But when we add nolabel, the difference is apparent:

. list female sex in 1/4, nolabel

<table>
<thead>
<tr>
<th>female</th>
<th>sex</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>female</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>0</td>
<td>male</td>
</tr>
<tr>
<td>0</td>
<td>male</td>
</tr>
</tbody>
</table>

Example 5

decode is most useful in instances when we wish to match-merge two datasets on a variable that has been encoded inconsistently.

For instance, we have two datasets on individual states in which one of the variables (state) takes on values such as “CA” and “NY”. The state variable was originally a string, but along the way the variable was encoded into an integer with a corresponding value label in one or both datasets.

We wish to merge these two datasets, but either 1) one of the datasets has a string variable for state and the other an encoded variable or 2) although both are numeric, we are not certain that the codings are consistent. Perhaps “CA” has been coded 5 in one dataset and 6 in another.

Because decode will take an encoded variable and turn it back into a string, decode provides the solution:

use first (load the first dataset)
decode state, gen(st) (make a string state variable)
drop state (discard the encoded variable)
sort st (sort on string)
save first, replace (save the dataset)
use second (load the second dataset)
decode state, gen(st) (make a string variable)
drop state (discard the encoded variable)
sort st (sort on string)
merge 1:1 st using first (merge the data)

Reference

Also see

[D] `compress` — Compress data in memory

[D] `destring` — Convert string variables to numeric variables and vice versa

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

[U] 12.6.3 Value labels

[U] 23.2 Categorical string variables