**destring — Convert string variables to numeric variables and vice versa**

### Description

destring converts variables in `varlist` from string to numeric. If `varlist` is not specified, `destring` will attempt to convert all variables in the dataset from string to numeric. Characters listed in `ignore()` are removed. Variables in `varlist` that are already numeric will not be changed. `destring` treats both empty strings “” and “.” as indicating sysmiss (.) and interprets the strings “.a”, “.b”, ..., “.z” as the extended missing values .a, .b, ..., .z; see [U] 12.2.1 Missing values. `destring` also ignores any leading or trailing spaces so that, for example, “ ” is equivalent to “ ” and “ .” is equivalent to “.”.

tostring converts variables in `varlist` from numeric to string. The most compact string format possible is used. Variables in `varlist` that are already string will not be converted.

### Quick start

Convert `strg1` from string to numeric, and place result in `num1`

```
destring strg1, generate(num1)
```

As above, but ignore the % character in `strg1`

```
destring strg1, generate(num1) ignore(%)
```

As above, but return . for observations with nonnumeric characters

```
destring strg1, generate(num1) force
```

Convert `num2` from numeric to string, and place result in `strg2`

```
tostring num2, generate(strg2)
```

As above, but format with a leading zero and 3 digits after the decimal

```
tostring num2, generate(strg2) format(%09.3f)
```

### Menu

**destring**

Data > Create or change data > Other variable-transformation commands > Convert variables from string to numeric

**tostring**

Data > Create or change data > Other variable-transformation commands > Convert variables from numeric to string
### Syntax

**Convert string variables to numeric variables**

```
destring [ varlist ] , { generate(newvarlist) | replace } [ destring_options ]
```

**Convert numeric variables to string variables**

```
tostring varlist , { generate(newvarlist) | replace } [ tostring_options ]
```

<table>
<thead>
<tr>
<th><strong>destring_options</strong></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>generate(newvarlist)</em></td>
<td>generate <code>newvar_1, ..., newvar_k</code> for each variable in <code>varlist</code></td>
</tr>
<tr>
<td><em>replace</em></td>
<td>replace string variables in <code>varlist</code> with numeric variables</td>
</tr>
<tr>
<td><em>ignore(&quot;chars&quot; [, ignoreopts])</em></td>
<td>remove specified nonnumeric characters, as characters or as bytes, and illegal Unicode characters</td>
</tr>
<tr>
<td>force</td>
<td>convert nonnumeric strings to missing values</td>
</tr>
<tr>
<td>float</td>
<td>generate numeric variables as type <code>float</code></td>
</tr>
<tr>
<td>percent</td>
<td>convert percent variables to fractional form</td>
</tr>
<tr>
<td>dpcomma</td>
<td>convert variables with commas as decimals to period-decimal format</td>
</tr>
</tbody>
</table>

* Either `generate(newvarlist)` or `replace` is required.

<table>
<thead>
<tr>
<th><strong>tostring_options</strong></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>generate(newvarlist)</em></td>
<td>generate <code>newvar_1, ..., newvar_k</code> for each variable in <code>varlist</code></td>
</tr>
<tr>
<td><em>replace</em></td>
<td>replace numeric variables in <code>varlist</code> with string variables</td>
</tr>
<tr>
<td>force</td>
<td>force conversion ignoring information loss</td>
</tr>
<tr>
<td>format(format)</td>
<td>convert using specified format</td>
</tr>
<tr>
<td>usedisplayformat</td>
<td>convert using display format</td>
</tr>
</tbody>
</table>

* Either `generate(newvarlist)` or `replace` is required.

### Options for destring

Either `generate()` or `replace` must be specified. With either option, if any string variable contains nonnumeric characters not specified with `ignore()`, then no corresponding variable will be generated, nor will that variable be replaced (unless `force` is specified).

`generate(newvarlist)` specifies that a new variable be created for each variable in `varlist`. `newvarlist` must contain the same number of new variable names as there are variables in `varlist`. If `varlist` is not specified, `destring` attempts to generate a numeric variable for each variable in the dataset; `newvarlist` must then contain the same number of new variable names as there are variables in the dataset. Any variable labels or characteristics will be copied to the new variables created.

`replace` specifies that the variables in `varlist` be converted to numeric variables. If `varlist` is not specified, `destring` attempts to convert all variables from string to numeric. Any variable labels or characteristics will be retained.
ignore("chars" [, ignoreopts]) specifies nonnumeric characters be removed. ignoreopts may be aschars, asbytes, or illegal. The default behavior is to remove characters as characters, which is the same as specifying aschars. asbytes specifies removal of all bytes included in all characters in the ignore string, regardless of whether these bytes form complete Unicode characters. illegal specifies removal of all illegal Unicode characters, which is useful for removing high-ASCII characters. illegal may not be specified with asbytes. If any string variable still contains any nonnumeric or illegal Unicode characters after the ignore string has been removed, no action will take place for that variable unless force is also specified. Note that to Stata the comma is a nonnumeric character; see also the dpcomma option below.

force specifies that any string values containing nonnumeric characters, in addition to any specified with ignore(), be treated as indicating missing numeric values.

float specifies that any new numeric variables be created initially as type float. The default is type double; see [D] Data types. destring attempts automatically to compress each new numeric variable after creation.

percent removes any percent signs found in the values of a variable, and all values of that variable are divided by 100 to convert the values to fractional form. percent by itself implies that the percent sign, “%”, is an argument to ignore(), but the converse is not true.

dpcomma specifies that variables with commas as decimal values should be converted to have periods as decimal values.

Options for tostring

Either generate() or replace must be specified. If converting any numeric variable to string would result in loss of information, no variable will be produced unless force is specified. For more details, see force below.

generate(newvarlist) specifies that a new variable be created for each variable in varlist. newvarlist must contain the same number of new variable names as there are variables in varlist. Any variable labels or characteristics will be copied to the new variables created.

replace specifies that the variables in varlist be converted to string variables. Any variable labels or characteristics will be retained.

force specifies that conversions be forced even if they entail loss of information. Loss of information means one of two circumstances: 1) The result of real(string(varname, "format")) is not equal to varname; that is, the conversion is not reversible without loss of information; 2) replace was specified, but a variable has associated value labels. In circumstance 1, it is usually best to specify usedisplayformat or format(). In circumstance 2, value labels will be ignored in a forced conversion. decode (see [D] encode) is the standard way to generate a string variable based on value labels.

format(format) specifies that a numeric format be used as an argument to the string() function, which controls the conversion of the numeric variable to string. For example, a format of %7.2f specifies that numbers are to be rounded to two decimal places before conversion to string. See Remarks and examples below and [FN] String functions and [D] format. format() cannot be specified with usedisplayformat.

usedisplayformat specifies that the current display format be used for each variable. For example, this option could be useful when using U.S. Social Security numbers or daily or other dates with some %d or %t format assigned. usedisplayformat cannot be specified with format().
Remarks and examples

Remarks are presented under the following headings:

destring
tostring
Saved characteristics
Video example

destring

Example 1

We read in a dataset, but somehow all the variables were created as strings. The variables contain no nonnumeric characters, and we want to convert them all from string to numeric data types.

```
. use https://www.stata-press.com/data/r16/destring1
. describe
Contains data from https://www.stata-press.com/data/r16/destring1.dta
obs: 10
vars: 5 3 Mar 2018 10:15
storage  display value
variable name  type format label  variable label
id          str3  %9s
num         str3  %9s
code        str4  %9s
total       str5  %9s
income      str5  %9s
```

Sorted by:

```
. list
                   id  num  code  total  income
1.          111  243  1234   543  23423
2.          111  123  2345  67854  12654
3.          111  234  3456   345  43658
4.          222  345  4567     57  23546
5.          333  456  5678     23  21432
6.          333  567  6789  23465  12987
7.          333  678  7890   65  9823
8.          444  789  8976   23  32980
9.          444  901  7654   23  18565
10.         555  890  6543   423  19234
```

. destring, replace
id: all characters numeric; replaced as int
num: all characters numeric; replaced as int
code: all characters numeric; replaced as int
total: all characters numeric; replaced as long
income: all characters numeric; replaced as long
Example 2

Our dataset contains the variable `date`, which was accidentally recorded as a string because of spaces after the year and month. We want to remove the spaces. `destring` will convert it to numeric and remove the spaces.

```stata
. use https://www.stata-press.com/data/r16/destring2, clear
. describe date
```

```plaintext
storage  display  value
variable name  type  format  label  variable label
date          str14  %10s
```

```stata
. destring date, replace
done
```

```
Example 2

Our dataset contains the variable `date`, which was accidentally recorded as a string because of spaces after the year and month. We want to remove the spaces. `destring` will convert it to numeric and remove the spaces.

```stata
. use https://www.stata-press.com/data/r16/destring2, clear
. destring date, replace
done
```

```
 storage  display  value
variable name  type  format  label  variable label
date          str14  %10s
```
. list date

<table>
<thead>
<tr>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1999 12 10</td>
</tr>
<tr>
<td>2000 07 08</td>
</tr>
<tr>
<td>1997 03 02</td>
</tr>
<tr>
<td>1999 09 00</td>
</tr>
<tr>
<td>1998 10 04</td>
</tr>
<tr>
<td>2000 03 28</td>
</tr>
<tr>
<td>2000 08 08</td>
</tr>
<tr>
<td>1997 10 20</td>
</tr>
<tr>
<td>1998 01 16</td>
</tr>
<tr>
<td>1999 11 12</td>
</tr>
</tbody>
</table>

. destring date, replace ignore(" ")
date: character space removed; replaced as long

. describe date

<table>
<thead>
<tr>
<th>variable name</th>
<th>storage</th>
<th>display</th>
<th>value</th>
<th>label</th>
<th>variable label</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>long</td>
<td>%10.0g</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

. list date

<table>
<thead>
<tr>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>19991210</td>
</tr>
<tr>
<td>20000708</td>
</tr>
<tr>
<td>19970302</td>
</tr>
<tr>
<td>19990900</td>
</tr>
<tr>
<td>19981004</td>
</tr>
<tr>
<td>20000328</td>
</tr>
<tr>
<td>20000808</td>
</tr>
<tr>
<td>19971020</td>
</tr>
<tr>
<td>19980116</td>
</tr>
<tr>
<td>19991112</td>
</tr>
</tbody>
</table>

Example 3

Our dataset contains the variables `date`, `price`, and `percent`. These variables were accidentally read into Stata as string variables because they contain spaces, dollar signs, commas, and percent signs. We want to remove all of these characters and create new variables for `date`, `price`, and `percent` containing numeric values. After removing the percent sign, we want to convert the `percent` variable to decimal form.
. use https://www.stata-press.com/data/r16/destring2, clear
. describe
Contains data from https://www.stata-press.com/data/r16/destring2.dta
obs: 10
vars: 3

storage display value
variable name type format label variable label

date    str14 %10s
price   str11 %11s
percent str3 %9s

Sorted by:
. list

<table>
<thead>
<tr>
<th>date</th>
<th>price</th>
<th>percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>1999 12 10</td>
<td>$2,343.68</td>
<td>34%</td>
</tr>
<tr>
<td>2000 07 08</td>
<td>$7,233.44</td>
<td>86%</td>
</tr>
<tr>
<td>1997 03 02</td>
<td>$12,442.89</td>
<td>12%</td>
</tr>
<tr>
<td>1999 09 00</td>
<td>$233,325.31</td>
<td>6%</td>
</tr>
<tr>
<td>1998 10 04</td>
<td>$1,549.23</td>
<td>76%</td>
</tr>
<tr>
<td>2000 03 28</td>
<td>$23,517.03</td>
<td>35%</td>
</tr>
<tr>
<td>2000 08 08</td>
<td>$2.43</td>
<td>69%</td>
</tr>
<tr>
<td>1997 10 20</td>
<td>$9,382.47</td>
<td>32%</td>
</tr>
<tr>
<td>1998 01 16</td>
<td>$289,209.32</td>
<td>45%</td>
</tr>
<tr>
<td>1999 11 12</td>
<td>$8,282.49</td>
<td>1%</td>
</tr>
</tbody>
</table>

. destring date price percent, generate(date2 price2 percent2) ignore("$ ,%")
> percent
date: character space removed; date2 generated as long
price: characters $ , removed; price2 generated as double
percent: character % removed; percent2 generated as double
. describe
Contains data from https://www.stata-press.com/data/r16/destring2.dta
obs: 10
vars: 6

storage display value
variable name type format label variable label

date       str14 %10s
date2     long %10.0g
price     str11 %11s
price2   double %10.0g
percent   str3 %9s
percent2  double %10.0g

Sorted by:
Note: Dataset has changed since last saved.
8   destring — Convert string variables to numeric variables and vice versa

<table>
<thead>
<tr>
<th>date</th>
<th>date2</th>
<th>price</th>
<th>price2</th>
<th>percent</th>
<th>percent2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1999</td>
<td>12</td>
<td>10</td>
<td>19991210</td>
<td>$2,343.68</td>
<td>34%</td>
</tr>
<tr>
<td>2000</td>
<td>07</td>
<td>08</td>
<td>20000708</td>
<td>$7,233.44</td>
<td>86%</td>
</tr>
<tr>
<td>1997</td>
<td>03</td>
<td>02</td>
<td>19970302</td>
<td>$12,442.89</td>
<td>12%</td>
</tr>
<tr>
<td>1999</td>
<td>09</td>
<td>00</td>
<td>19990900</td>
<td>$233,325.31</td>
<td>6%</td>
</tr>
<tr>
<td>1998</td>
<td>10</td>
<td>04</td>
<td>19981004</td>
<td>$1,549.23</td>
<td>76%</td>
</tr>
<tr>
<td>2000</td>
<td>03</td>
<td>28</td>
<td>20000328</td>
<td>$23,517.03</td>
<td>35%</td>
</tr>
<tr>
<td>2000</td>
<td>08</td>
<td>08</td>
<td>20000808</td>
<td>$2.43</td>
<td>69%</td>
</tr>
<tr>
<td>1997</td>
<td>10</td>
<td>20</td>
<td>19971020</td>
<td>$9,382.47</td>
<td>32%</td>
</tr>
<tr>
<td>1998</td>
<td>01</td>
<td>16</td>
<td>19980116</td>
<td>$289,209.32</td>
<td>45%</td>
</tr>
<tr>
<td>1999</td>
<td>11</td>
<td>12</td>
<td>19991112</td>
<td>$8,282.49</td>
<td>1%</td>
</tr>
</tbody>
</table>

Conversion of numeric data to string equivalents can be problematic. Stata, like most software, holds numeric data to finite precision and in binary form. See the discussion in [U] 13.12 Precision and problems therein. If no `format()` is specified, `tostring` uses the format `%12.0g`. This format is, in particular, sufficient to convert integers held as bytes, ints, or longs to string equivalent without loss of precision.

However, users will often need to specify a format themselves, especially when the numeric data have fractional parts and for some reason a conversion to string is required.

Example 4

Our dataset contains a string month variable and numeric year and day variables. We want to convert the three variables to a `%td` date.

```
. use https://www.stata-press.com/data/r16/tostring, clear
. list

   +-----------------+-----------------+-----------------+-----------------+
   | id   | month | day  | year |
   +-----------------+-----------------+-----------------+-----------------+
   1.   | 123456789 | jan   | 10   | 2001 |
   2.   | 123456710 | mar   | 20   | 2001 |
   3.   | 123456711 | may   | 30   | 2001 |
   4.   | 123456712 | jun   | 9    | 2001 |
   5.   | 123456713 | oct   | 17   | 2001 |
   6.   | 123456714 | nov   | 15   | 2001 |
   7.   | 123456715 | dec   | 28   | 2001 |
   8.   | 123456716 | apr   | 29   | 2001 |
   9.   | 123456717 | mar   | 11   | 2001 |
  10.   | 123456718 | jul   | 3    | 2001 |
   +-----------------+-----------------+-----------------+-----------------+
```

. tostring year day, replace
year was float now str4
day was float now str2
. generate date = month + "/" + day + "" + year
. generate edate = date(date, "MDY")
. format edate %td
. list

<table>
<thead>
<tr>
<th>id</th>
<th>month</th>
<th>day</th>
<th>year</th>
<th>date</th>
<th>edate</th>
</tr>
</thead>
<tbody>
<tr>
<td>123456789</td>
<td>jan</td>
<td>10</td>
<td>2001</td>
<td>jan/10/2001</td>
<td>10jan2001</td>
</tr>
<tr>
<td>123456710</td>
<td>mar</td>
<td>20</td>
<td>2001</td>
<td>mar/20/2001</td>
<td>20mar2001</td>
</tr>
<tr>
<td>123456711</td>
<td>may</td>
<td>30</td>
<td>2001</td>
<td>may/30/2001</td>
<td>30may2001</td>
</tr>
<tr>
<td>123456712</td>
<td>jun</td>
<td>9</td>
<td>2001</td>
<td>jun/9/2001</td>
<td>09jun2001</td>
</tr>
<tr>
<td>123456713</td>
<td>oct</td>
<td>17</td>
<td>2001</td>
<td>oct/17/2001</td>
<td>17oct2001</td>
</tr>
<tr>
<td>123456714</td>
<td>nov</td>
<td>15</td>
<td>2001</td>
<td>nov/15/2001</td>
<td>15nov2001</td>
</tr>
<tr>
<td>123456716</td>
<td>apr</td>
<td>29</td>
<td>2001</td>
<td>apr/29/2001</td>
<td>29apr2001</td>
</tr>
<tr>
<td>123456718</td>
<td>jul</td>
<td>3</td>
<td>2001</td>
<td>jul/3/2001</td>
<td>03jul2001</td>
</tr>
</tbody>
</table>

Saved characteristics

Each time the destring or tostring commands are issued, an entry is made in the characteristics list of each converted variable. You can type char list to view these characteristics.

After example 3, we could use char list to find out what characters were removed by the destring command.

```
. char list
date2[destring]: Character removed was: space
date2[destring_cmd]: destring date price percent, generate(date2 pri..
price2[destring]: Characters removed were: $,
price2[destring_cmd]: destring date price percent, generate(date2 pri..
percent2[destring]: Character removed was: %
percent2[destring_cmd]: destring date price percent, generate(date2 pri..
```

Video example

How to convert a string variable to a numeric variable

Acknowledgment

destring and tostring were originally written by Nicholas J. Cox of the Department of Geography at Durham University, UK, and coeditor of the Stata Journal and author of Speaking Stata Graphics.

References


**Also see**

[D] *egen* — Extensions to generate

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

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

[D] *split* — Split string variables into parts

[FN] *String functions*