**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 (.). It 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 ]
```

**destring_options**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>generate(newvarlist)</em> generate newvar₁, . . . , newvarₖ for each variable in varlist</td>
</tr>
<tr>
<td><em>replace</em> replace string variables in varlist with numeric variables</td>
</tr>
<tr>
<td><em>ignore(&quot;chars&quot; [, ignoreopts])</em> remove specified nonnumeric characters, as characters or as bytes, and illegal Unicode characters</td>
</tr>
<tr>
<td><em>force</em> convert nonnumeric strings to missing values</td>
</tr>
<tr>
<td><em>float</em> generate numeric variables as type float</td>
</tr>
<tr>
<td><em>percent</em> convert percent variables to fractional form</td>
</tr>
<tr>
<td><em>dpcomma</em> convert variables with commas as decimals to period-decimal format</td>
</tr>
</tbody>
</table>

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

**tostring_options**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>generate(newvarlist)</em> generate newvar₁, . . . , newvarₖ for each variable in varlist</td>
</tr>
<tr>
<td><em>replace</em> replace numeric variables in varlist with string variables</td>
</tr>
<tr>
<td><em>force</em> force conversion ignoring information loss</td>
</tr>
<tr>
<td><em>format(format)</em> convert using specified format</td>
</tr>
<tr>
<td><em>usedisplayformat</em> 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.

debug specifies that any new numeric variables be created initially as type debug. 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  Sorted by:
   vars:         5
                  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
```

We then use the `destring` command to convert the variables:

```
. 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
```
. describe
Contains data from https://www.stata-press.com/data/r16/destring1.dta
obs: 10  
vars: 5  
3 Mar 2018 10:15

<table>
<thead>
<tr>
<th>variable name</th>
<th>storage</th>
<th>type</th>
<th>format</th>
<th>label</th>
<th>variable label</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>int</td>
<td>%10.0g</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>num</td>
<td>int</td>
<td>%10.0g</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>code</td>
<td>int</td>
<td>%10.0g</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>total</td>
<td>long</td>
<td>%10.0g</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>income</td>
<td>long</td>
<td>%10.0g</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sorted by:
Note: Dataset has changed since last saved.

. list

<table>
<thead>
<tr>
<th>id</th>
<th>num</th>
<th>code</th>
<th>total</th>
<th>income</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>111</td>
<td>243</td>
<td>1234</td>
<td>543</td>
</tr>
<tr>
<td>2</td>
<td>111</td>
<td>123</td>
<td>2345</td>
<td>67854</td>
</tr>
<tr>
<td>3</td>
<td>111</td>
<td>234</td>
<td>3456</td>
<td>345</td>
</tr>
<tr>
<td>4</td>
<td>222</td>
<td>345</td>
<td>4567</td>
<td>57</td>
</tr>
<tr>
<td>5</td>
<td>333</td>
<td>456</td>
<td>5678</td>
<td>23</td>
</tr>
<tr>
<td>6</td>
<td>333</td>
<td>567</td>
<td>6789</td>
<td>23465</td>
</tr>
<tr>
<td>7</td>
<td>333</td>
<td>678</td>
<td>7890</td>
<td>65</td>
</tr>
<tr>
<td>8</td>
<td>444</td>
<td>789</td>
<td>8976</td>
<td>23</td>
</tr>
<tr>
<td>9</td>
<td>444</td>
<td>901</td>
<td>7654</td>
<td>23</td>
</tr>
<tr>
<td>10</td>
<td>555</td>
<td>890</td>
<td>6543</td>
<td>423</td>
</tr>
</tbody>
</table>

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.

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

<table>
<thead>
<tr>
<th>variable name</th>
<th>storage</th>
<th>type</th>
<th>format</th>
<th>label</th>
<th>variable label</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>str14</td>
<td>%10s</td>
<td></td>
<td></td>
<td></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  3 Mar 2018 22:50
    vars: 3
storage  display  value
variable name  type  format  label  variable label
    date   str14   %10s          
    price  str11   %11s          
    percent  str3   %9s          
Sorted by:
. list
     +---------------------------------+
     | date | price  | percent  |
     +---------------------------------+
     | 1999 | 12   | 10     | $2,343.68 | 34% |
     | 2000 | 07   | 08     | 7,233.44  | 86% |
     | 1997 | 03   | 02     | 12,442.89 | 12% |
     | 1999 | 09   | 00     | 233,325.31| 6%  |
     | 1998 | 10   | 04     | 1,549.23  | 76% |
     | 2000 | 03   | 28     | 23,517.03 | 35% |
     | 2000 | 08   | 08     | 2.43      | 69% |
     | 1997 | 10   | 20     | 9,382.47  | 32% |
     | 1998 | 01   | 16     | 289,209.32| 45% |
     | 1999 | 11   | 12     | 8,282.49  | 1%  |
     +---------------------------------+
. destring date price percent, generate(date2 price2 percent2) ignore("$ ,%")
> 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  3 Mar 2018 22:50
    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.
## tostring

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.

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

+------------+----------+----+------------+----------+----------+----------+----------+----------+----------+
| id        | month    | day |           |          | percent  | percent2 |
|------------+----------+----+------------+----------+----------+----------+
| 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     |
+------------+----------+----+------------+----------+----------+----------+
```

### Conversion

```stata
. 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>1</td>
<td>123456789</td>
<td>jan</td>
<td>10</td>
<td>2001</td>
<td>jan/10/2001</td>
</tr>
<tr>
<td>2</td>
<td>123456710</td>
<td>mar</td>
<td>20</td>
<td>2001</td>
<td>mar/20/2001</td>
</tr>
<tr>
<td>3</td>
<td>123456711</td>
<td>may</td>
<td>30</td>
<td>2001</td>
<td>may/30/2001</td>
</tr>
<tr>
<td>4</td>
<td>123456712</td>
<td>jun</td>
<td>9</td>
<td>2001</td>
<td>jun/9/2001</td>
</tr>
<tr>
<td>5</td>
<td>123456713</td>
<td>oct</td>
<td>17</td>
<td>2001</td>
<td>oct/17/2001</td>
</tr>
<tr>
<td>6</td>
<td>123456714</td>
<td>nov</td>
<td>15</td>
<td>2001</td>
<td>nov/15/2001</td>
</tr>
<tr>
<td>8</td>
<td>123456716</td>
<td>apr</td>
<td>29</td>
<td>2001</td>
<td>apr/29/2001</td>
</tr>
<tr>
<td>10</td>
<td>123456718</td>
<td>jul</td>
<td>3</td>
<td>2001</td>
<td>jul/3/2001</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*