

string — String manipulation functions

[Contents](#)[Description](#)[Remarks and examples](#)[Also see](#)

Contents

[M-5] Manual entry	Function	Purpose
Parsing		
tokens()	<code>tokens()</code>	obtain tokens (words) from string
invtokens()	<code>invtokens()</code>	concatenate string vector into string scalar
ustrword()	<code>ustrword()</code> <code>ustrwordcount()</code>	return <i>n</i> th Unicode word return the number of Unicode words
strmatch()	<code>strmatch()</code>	pattern matching
tokenget()	...	advanced parsing
Length & position		
strlen()	<code>strlen()</code>	length of string in bytes
ustrlen()	<code>ustrlen()</code>	length of string in Unicode characters
udstrlen()	<code>udstrlen()</code>	length of string in display columns
fmtwidth()	<code>fmtwidth()</code>	width of <i>%fmt</i>
strpos()	<code>strpos()</code> <code>strrpos()</code>	find substring within string from left find substring within string from right
ustrpos()	<code>ustrpos()</code> <code>ustrrpos()</code>	find Unicode substring within string, first occurrence find Unicode substring within string, last occurrence
indexnot()	<code>indexnot()</code>	find character not in list
Editing		
substr()	<code>substr()</code>	extract substring
usubstr()	<code>usubstr()</code>	extract Unicode substring
udsubstr()	<code>udsubstr()</code>	extract Unicode substring based on display columns

Editing, <i>continued</i>

strupper()	<code>strupper()</code> <code>strlower()</code> <code>strproper()</code>	convert to uppercase convert to lowercase convert to proper case
ustrupper()	<code>ustrupper()</code> <code>ustrlower()</code> <code>ustrtitle()</code>	convert Unicode characters to uppercase convert Unicode characters to lowercase convert Unicode characters to titlecase
strtrim()	<code>stritrim()</code> <code>strltrim()</code> <code>strrrtrim()</code> <code>strtrim()</code>	replace multiple, consecutive internal blanks with one blank remove leading blanks remove trailing blanks remove leading and trailing blanks
ustrtrim()	<code>ustrtrim()</code> <code>ustrltrim()</code> <code>ustrrrtrim()</code>	remove leading and trailing Unicode whitespace characters and blanks remove leading Unicode whitespace characters and blanks remove trailing Unicode whitespace characters and blanks
subinstr()	<code>subinstr()</code> <code>subinword()</code>	substitute text substitute word
usubinstr()	<code>usubinstr()</code>	replace Unicode substring
_substr()	<code>_substr()</code>	substitute into string
_usubstr()	<code>_usubstr()</code>	substitute into Unicode string
strdup()	<code>*</code>	duplicate string
strreverse()	<code>strreverse()</code>	reverse string in bytes
ustrreverse()	<code>ustrreverse()</code>	reverse string in Unicode characters
soundex()	<code>soundex()</code> <code>soundex_nara()</code>	convert to soundex code convert to U.S. Census soundex code

Stata

abbrev()	<code>abbrev()</code>	abbreviate Unicode strings to display columns
strtoname()	<code>strtoname()</code>	translate strings to Stata 13 compatible names
ustrtoname()	<code>ustrtoname()</code>	translate Unicode strings to Stata names

Text translation

strotoreal()	strotoreal()	convert real to string
strtooreal()	strtooreal()	convert string to real
ustrto()	ustrto()	convert a Unicode string to a string in another encoding
	ustrfrom()	convert a string in one encoding to a Unicode string
ustrunescape()	ustrunescape()	convert the escaped hex sequences to Unicode
	ustrtohex()	convert a Unicode sequence to hex sequences
ascii()	ascii()	obtain ASCII or byte codes of string
	char()	make string from ASCII or byte codes
uchar()	uchar()	make Unicode character from Unicode code-point value

Unicode utilities

ustrcompare()	ustrcompare()	compare or sort Unicode strings
	ustrsortkey()	obtain sort key of Unicode string
ustrfix()	ustrfix()	replace invalid sequences in Unicode string
ustrnormalize()	ustrnormalize()	normalize Unicode string

Description

The above functions are for manipulating strings. Strings in Mata are strings of Unicode characters in [UTF-8 encoding](#), usually the printable characters, but Mata enforces no such restriction. In particular, strings may contain binary 0.

Remarks and examples

In addition to the above functions, two operators are especially useful for dealing with strings.

The first is `+`. Addition is how you concatenate strings:

```
: "abc" + "def"
abcdef
: "Café " + "de Flore"
Café de Flore
: command = "list"
: args = "mpg weight"
: result = command + " " + args
: result
list mpg weight
```

The second is `*`. Multiplication is how you duplicate strings:

```
: 5*"a"
aaaaa
: "Allô"*2
AllôAllô
: indent = 20
: title = indent*" " + "My Title"
: title
                My Title
```

Also see

[\[M-4\] intro](#) — Index and guide to functions