Title

stata.com

ustrnormalize( ) — Normalize Unicode string

Description Syntax Remarks and examples Conformability
Diagnostics Also see

Description

`ustrnormalize(`*s*`,` *norm*`)` normalizes Unicode string *s* to one of the five normalization forms specified by *norm*.

When *s* is not a scalar, the function returns element-by-element results.

Syntax

string matrix `ustrnormalize(`*string matrix s*`,` *string matrix norm*`)`

Remarks and examples

stata.com

Unicode normalization removes the Unicode string differences caused by Unicode character equivalence. For example, the character "i" with two dots as in naïve can be represented either by a single Unicode code point, `\u00ef`, or by two code points, `\u0069`, which is the regular "i", and `\u0308`, which is the diaeresis character. The code point `\u00ef` and the code-point sequence `\u0069\u0308` are considered Unicode equivalent. According to the Unicode standard, they should be treated as the same single character in Unicode string operations, such as display, comparison, and selection. But Stata does not support multiple code-point characters; each code point is considered a single Unicode character. Hence, `\u0069\u0308` is displayed as two characters in the Results window. `ustrnormalize()` can be used to deal with this issue by normalizing `\u0069\u0308` to its canonical equivalent composited `\NFC` form `\u00ef`.

norm must be one of `nfc`, `nfd`, `nfkc`, `nfkd`, or `nfkcc`. The function returns an empty string for any other value of *norm*.

`nfc` specifies `Normalization Form C`, which normalizes decomposed Unicode code points to canonical composited form. `nfd` specifies `Normalization Form D`, which normalizes composited Unicode code points to canonical decomposed form. `nfc` and `nfd` produce canonical equivalent form. `nfkc` and `nfkd` are similar to `nfc` and `nfd` but produce compatibility equivalent form. `nfkcc` is similar to `nfkc` but also handles case folding. For details, see http://unicode.org/reports/tr15/.

Conformability

`ustrnormalize(`*s*`,` *norm*`)`:

s:	$r \times c$
norm:	$r \times c$ or 1×1
result:	$r \times c$

Diagnostics

None.

Also see

[M-4] **string** — String manipulation functions

[U] 12.4.2 Handling Unicode strings