Title

stata.com

unicode collator — Language-specific Unicode collators

Description Syntax Remarks and examples Also see

Description

`unicode collator list` lists the subset of locales that have language-specific collators for the Unicode string comparison functions: `ustrcompare()`, `ustrcompareex()`, `ustrsortkey()`, and `ustrsortkeyex()`.

Syntax

`unicode collator list` [*pattern*]

pattern is one of `_all`, `*`, `*`*name*`*`, `*`*name*, or *name*`*`. If you specify nothing, `_all`, or `*`, then all results will be listed. `*`*name*`*` lists all results containing *name*; `*`*name* lists all results ending with *name*; and *name*`*` lists all results starting with *name*.

Remarks and examples

stata.com

Remarks are presented under the following headings:

Overview of collation

Collation is the process of comparing and sorting Unicode character strings as a human might logically order them. We call this ordering strings in a language-sensitive manner. To do this, Stata uses a Unicode tool known as the Unicode collation algorithm, or UCA.

To perform language-sensitive string sorts, you must combine `ustrsortkey()` or `ustrsortkeyex()` with `sort`. It is a complicated process and there are several issues about which you need to be aware. For details, see **[U] 12.4.2.5 Sorting strings containing Unicode characters**. To perform language-sensitive string comparisons, you can use `ustrcompare()` or `ustrcompareex()`.

For details about the UCA, see http://www.unicode.org/reports/tr10/.

The role of locales in collation

During collation, Stata can use the default collator or it can perform language-sensitive string comparisons or sorts that require knowledge of a locale.

A locale identifies a community with a certain set of preferences for how their language should be written; see **[U] 12.4.2.4 Locales in Unicode**. For example, in English, the uppercase letter of the Latin small letter "i" is the Latin capital letter "I". However, in Turkish, the uppercase letter is "I" with a dot above it (Unicode `\u0130`); hence, the case mapping is locale-sensitive.

Collation in Stata involves the locale-sensitive functions `ustrcompare()`, `ustrcompareex()`, `ustrsortkey()`, and `ustrsortkeyex()`. If you specify a locale with one of these functions or if you have set the locale globally (see [P] **set locale_functions**), then collation may be performed using a language-specific collator.

Because a locale is simply an identifier to locate the resources for specific services, there is no validation of the locale. For example, specifying "klingon" is as valid as specifying "en" when calling `ustrcompare()` or the other functions discussed here. If the collation data for the "klingon" locale is found, then the locale is populated; otherwise, fallback rules are followed. For more information, see *Default locale and locale fallback* in [D] **unicode locale**.

Stata supports hundreds of locales, but only about 100 have a language-specific collator. `unicode collator list` lets you determine whether your locale (or language) has its own collator. For example, Stata supports two locales for the Zulu language: `zu` is a general locale and `zu_ZA` is Zulu specific to South Africa. Only `zu` has a language-specific collator.

Further controlling collation

`ustrcompare()` and `ustrsort()` use the default collation algorithm for the locale. However, you can exercise finer control over the collation algorithm if you use `ustrcompareex()` or `ustrsortkeyex()`.

An International Components for Unicode (ICU) locale may contain up to five subtags in the following order: language, script, country, variant, and keywords. Stata usually uses only the language and country tags. However, collation keywords may be used in the `ustrcompareex()` and `ustrsortkeyex()` functions.

The collation keyword specifies the string sort order of the locale. For example, "pinyin" and "stroke" for Chinese language produce different string sort orders. In most cases, it is not necessary to specify a collation keyword; the default collator (either for Stata or for the language) provides sufficient control. However, some programmers may wish to specify a specific value. If you do not know the value of the collation keyword, you can obtain a list of valid collation values and their meanings in XML format at http://unicode.org/repos/cldr/trunk/common/bcp47/collation.xml.

If you are comparing or sorting Unicode strings that have come from different data sources, then you may need to normalize the strings before ordering them. See `ustrnormalize()` for details on normalization, and note the *norm* parameter in `ustrcompareex()` and `ustrsortkeyex()`.

Also see

[D] **unicode** — Unicode utilities

[D] **unicode locale** — Unicode locale utilities

[U] **12.4.2 Handling Unicode strings**

[U] **12.4.2.5 Sorting strings containing Unicode characters**