Glossary

ASCII. ASCII stands for American Standard Code for Information Interchange. It is a way of representing text and the characters that form text in computers. It can be divided into two sections: plain, or lower, ASCII, which includes numbers, punctuation, plain letters without diacritical marks, whitespace characters such as space and tab, and some control characters such as carriage return; and extended ASCII, which includes letters with diacritical marks as well as other special characters.

Before Stata 14, datasets, do-files, ado-files, and other Stata files were encoded using ASCII.

binary 0. Binary 0, also known as the null character, is traditionally used to indicate the end of a string, such as an ASCII or UTF-8 string.

Binary 0 is obtained by using `char(0)` and is sometimes displayed as `\0`. See **[U] 12.4.10 strL variables and binary strings** for more information.

binary string. A binary string is, technically speaking, any string that does not contain text. In Stata, however, a string is only marked as binary if it contains binary 0, or if it contains the contents of a file read in using the `fileread()` function, or if it is the result of a string expression containing a string that has already been marked as binary.

In Stata, `strL` variables, string scalars, and Mata strings can store binary strings.

See **[U] 12.4.10 strL variables and binary strings** for more information.

byte. Formally, a byte is eight binary digits (bits), the units used to record computer data. Each byte can also be considered as representing a value from 0 through 255. Do not confuse this with Stata's `byte` variable storage type, which allows values from −127 to 100 to be stored. With regard to strings, all strings are composed of individual characters that are encoded using either one byte or several bytes to represent each character.

For example, in UTF-8, the encoding system used by Stata, byte value 97 encodes "a". Byte values 195 and 161 in sequence encode "á".

characteristics. Characteristics are one form of metadata about a Stata dataset and each of the variables within the dataset. They are typically used in programming situations. For example, the xt commands need to know the name of the panel variable and possibly the time variable. These variable names are stored in characteristics within the dataset. See **[U] 12.8 Characteristics** for an overview and **[P] char** for a technical description.

code pages. A code page maps extended ASCII values to a set of characters, typically for a specific language or set of languages. For example, the most commonly used code page is Windows-1252, which maps extended ASCII values to characters used in Western European languages. Code pages are essentially encodings for extended ASCII characters.

code point. A code point is the numerical value or position that represents a single character in a text system such as ASCII or Unicode. The original ASCII encoding system contains only 128 code points and thus can represent only 128 characters. Historically, the 128 additional bytes of extended ASCII have been encoded in many different and inconsistent ways to provide additional sets of 128 code points. The formal Unicode specification has 1,114,112 possible code points, of which roughly 250,000 have been assigned to actual characters. Stata uses UTF-8 encoding for Unicode. Note that the UTF-8–encoded version of a code point does not have the same numeric value as the code point itself.

display column. A display column is the space required to display one typical character in the fixed-width display used by Stata's Results window and Viewer. Some characters are too wide for one display column. Each character is displayed in one or two display columns.

All plain ASCII characters (for example, "M" and "9") and many UTF-8 characters that are not plain ASCII (for example, "é") require the same space when using a fixed-width font. That is to say, they all require a single display column.

Characters from non-Latin alphabets, such as Chinese, Cyrillic, Japanese, and Korean, may require two display columns.

See [U] **12.4.2.2 Displaying Unicode characters** for more information.

display format. The display format for a variable specifies how the variable will be displayed by Stata. For numeric variables, the display format would indicate to Stata how many digits to display, how many decimal places to display, whether to include commas, and whether to display in exponential format. Numeric variables can also be formatted as dates. For strings, the display format indicates whether the variable should be left-aligned or right-aligned in displays and how many characters to display. Display formats may be specified by the `format` command. Display formats may also be used with individual numeric or string values to control how they are displayed. Distinguish display formats from storage types.

encodings. An encoding is a way of representing a character as a byte or series of bytes. Examples of encoding systems are ASCII and UTF-8. Stata uses UTF-8 encoding.

For more information, see [U] **12.4.2.3 Encodings**.

extended ASCII. Extended ASCII, also known as higher ASCII, is the byte values 128 to 255, which were not defined as part of the original ASCII specification. Various code pages have been defined over the years to map the extended ASCII byte values to many characters not supported in the original ASCII specification, such as Latin letters with diacritical marks, such as "á" and "Á"; non-Latin alphabets, such as Chinese, Cyrillic, Japanese, and Korean; punctuation marks used in non-English languages, such as "<", complex mathematical symbols such as "±", and more.

Although extended ASCII characters are stored in a single byte in ASCII encoding, UTF-8 stores the same characters in two to four bytes. Because each code page maps the extended ASCII values differently, another distinguishing feature of extended ASCII characters is that their meaning can change across fonts and operating systems.

frames. Frames, also known as data frames, are in-memory areas where datasets are analyzed. Stata can hold multiple datasets in memory, and each dataset is held in a memory area called a frame. A variety of commands exist to manage frames and manipulate the data in them. See [D] **frames**.

hexadecimal. The hexadecimal number system, or simply hex, is a base-16 number system represented by digits 0 through 9 and letters A through F.

higher ASCII. See *extended ASCII*.

locale. A locale is a code that identifies a community with a certain set of rules for how their language should be written. A locale can refer to something as general as an entire language (for example, "en" for English) or something as specific as a language in a particular country (for example, "en_HK" for English in Hong Kong).

A locale specifies a set of rules that govern how the language should be written. Stata uses locales to determine how certain language-specific operations are carried out. For more information, see [U] **12.4.2.4 Locales in Unicode**.

long format and wide format. Think of a dataset as having an ID variable, i, and a variable, j, whose values denote a subobservation. For instance, a person might be the i variable, and a year might be the j variable, so you have information about a set of people across several years. If this information is organized such that the j variable is explicitly specified, then the data are in long format; otherwise, they are in wide format. For instance,

```
id    year   income
 1    1980    10000
 1    1981    12000
 1    1982    11000
 2    1980    15000
 2    1981    14000
 2    1982    17000
```

are in long format because the j variable, `year`, is explicitly specified. In the following, the data are in wide format:

```
id  income1980   income1981   income1982
 1       10000        12000        11000
 2       15000        14000        17000
```

See [D] **reshape** for how to go between long and wide format.

lower ASCII. See *plain ASCII*.

null-terminator. See *binary 0*.

numlist. A numlist is a list of numbers. That list can be one or more arbitrary numbers or can use certain shorthands to indicate ranges, such as `5/9` to indicate integers 5, 6, 7, 8, and 9. Ranges can be ascending or descending and can include an optional increment or decrement amount, such as `10.5(-2)4.5` to indicate 10.5, 8.5, 6.5, and 4.5. See [U] **11.1.8 numlist** for a list of shorthands to indicate ranges.

plain ASCII. We use plain ASCII as a nontechnical term to refer to what computer programmers call lower ASCII. These are the plain Latin letters "a" to "z" and "A" to "Z"; numbers "0" through "9"; many punctuation marks, such as "!"; simple mathematical symbols, such as "+"; and whitespace and control characters such as space (" "), tab, and carriage return.

Each plain ASCII characters is stored as a single byte with a value between 0 and 127. Another distinguishing feature is that the byte values used to encode plain ASCII characters are the same across different operating systems and are common between ASCII and UTF-8.

Also see *ASCII* and *encodings*.

prefix command. A prefix command is a command in Stata that prefixes other Stata commands. For example, `by` *varlist*`:`. The command `by region: summarize marriage_rate divorce_rate` would summarize `marriage_rate` and `divorce_rate` for each region separately. See [U] **11.1.10 Prefix commands**.

storage types. A storage type is how Stata stores a variable. The numeric storage types in Stata are `byte`, `int`, `long`, `float`, and `double`. There is also a `string` storage type. The storage type is specified before the variable name when a variable is created. See [U] **12.2.2 Numeric storage types**, [U] **12.4 Strings**, and [D] **Data types**. Distinguish storage types from display formats.

str1, **str2**, . . . , **str2045**. See *strL*.

strL. `strL` is a storage type for string variables. The full list of string storage types is `str1`, `str2`, . . . , `str2045`, and `strL`.

`str1`, `str2`, . . . , `str2045` are fixed-length storage types. If variable `mystr` is `str8`, then 8 bytes are allocated in each observation to store `mystr`'s value. If you have 2,000 observations, then 16,000 bytes in total are allocated.

Distinguish between storage length and string length. If `myvar` is `str8`, that does not mean the strings are 8 characters long in every observation. The maximum length of strings is 8 characters. Individual observations may have strings of length 0, 1, . . . , 8. Even so, every string requires 8 bytes of storage.

You need not concern yourself with the storage length because string variables are automatically promoted. If `myvar` is `str8`, and you changed the contents of `myvar` in the third observation to "Longer than 8", then `myvar` would automatically become `str13`.

If you changed the contents of `myvar` in the third observation to a string longer than 2,045 characters, `myvar` would become `strL`.

`strL` variables are not necessarily longer than 2,045 characters; they can be longer or shorter than 2,045 characters. The real difference is that `strL` variables are stored as varying length. Pretend that `myothervar` is a `strL` and its third observation contains "this". The total memory consumed by the observation would be $64+4+1 = 69$ bytes. There would be 64 bytes of tracking information, 4 bytes for the contents (there are 4 characters), and 1 more byte to terminate the string. If the fifth observation contained a 2,000,000-character string, then $64 + 2{,}000{,}000 + 1 = 2{,}000{,}069$ bytes would be used to store it.

Another difference between `str1`, `str2`, ..., `str2045`, and `strL`s is that the `str#` storage types can store only ASCII strings. `strL` can store ASCII or binary strings. Thus a `strL` variable could contain, for instance, the contents of a Word document or a JPEG image or anything else.

`strL` is pronounce *sturl*.

titlecase, **title-cased string**, and **Unicode title-cased string**. In grammar, titlecase refers to the capitalization of the key words in a phrase. In Stata, titlecase refers to (a) the capitalization of the first letter of each word in a string and (b) the capitalization of each letter after a nonletter character. There is no judgment of the word's importance in the string or whether the letter after a nonletter character is part of the same word. For example, "it's" in titlecase is "It'S".

A title-cased string is any string to which the above rules have been applied. For example, if we used the `strproper()` function with the book title *Zen and the Art of Motorcycle Maintenance*, Stata would return the title-cased string `Zen And The Art Of Motorcycle Maintenance`.

A Unicode title-cased string is a string that has had Unicode title-casing rules applied to Unicode words. This is almost, but not exactly, like capitalizing the first letter of each Unicode word. Like capitalization, title-casing letters is locale-dependent, which means that the same letter might have different titlecase forms in different locales. For example, in some locales, capital letters at the beginning of words are not supposed to have accents on them, even if that capital letter by itself would have an accent.

If you do not have characters beyond plain ASCII and your locale is English, there is no distinction in results. For example, `ustrtitle()` with an English locale locale also would return the title-cased string `Zen And The Art Of Motorcycle Maintenance`.

Use the `ustrtitle()` function to apply the appropriate capitalization rules for your language (locale).

Unicode. Unicode is a standard for encoding and dealing with text written in almost any conceivable living or dead language. Unicode specifies a set of encoding systems that are designed to hold (and, unlike extended ASCII, to keep separate) characters used in different languages. The Unicode standard defines not only the characters and encodings for them, but also rules on how to perform various operations on words in a given language (locale), such as capitalization and ordering. The most common Unicode encodings are mUTF-8, UTF-16, and UTF-32. Stata uses UTF-8.

Unicode character. Technically, a Unicode character is any character with a Unicode encoding. Colloquially, we use the term to refer to any character other than the plain ASCII characters.

Unicode normalization. Unicode normalization allows us to use a common representation and therefore compare Unicode strings that appear the same when displayed but could have more than one way of being encoded. This rarely arises in practice, but because it is possible in theory, Stata provides the `ustrnormalize()` function for converting between different normalized forms of the same string.

For example, suppose we wish to search for "ñ" (the lowercase n with a tilde over it from the Spanish alphabet). This letter may have been encoded with the single code point U+00F1. However, the sequence U+006E (the Latin lowercase "n") followed by U+0303 (the tilde) is defined by Unicode to be equivalent to U+00F1. This type of visual identicalness is called canonical equivalence. The one-code-point form is known as the canonical composited form, and the multiple-code-point form is known as the canonical decomposed form. Normalization modifies one or the other string to the opposite canonical equivalent form so that the underlying byte sequences match. If we had strings in a mixture of forms, we would want to use this normalization when sorting or when searching for strings or substrings.

Another form of Unicode normalization allows characters that appear somewhat different to be given the same meaning or interpretation. For example, when sorting or indexing, we may want the code point U+FB00 (the typographic ligature "ff") to match the sequence of two Latin "f" letters encoded as U+0066 U+0066. This is called compatible equivalence.

Unicode title-cased string. See *titlecase, title-cased string, and Unicode title-cased string*.

UTF-8. UTF-8 stands for Universal character set + Transformation Format—8-bit. It is a type of Unicode encoding system that was designed for backward compatibility with ASCII and is used by Stata 14.

value label. A value label defines a mapping between numeric data and the words used to describe what those numeric values represent. So, the variable disease might have a value label status associated with it that maps 1 to positive and 0 to negative. See **[U] 12.6.3 Value labels**.

varlist. A varlist is a list of variables that observe certain conventions: variable names may be abbreviated; the asterisk notation can be used as a shortcut to refer to groups of variables, such as `income*` or `*1995` to refer to all variable names beginning with `income` or all variable names ending in `1995`, respectively; and a dash may be used to indicate all variables stored between the two listed variables, for example, `mpg-weight`. See **[U] 11.4 varname and varlists**.

wide format. See *long and wide format*.