Title

stata.com

data types — Quick reference for data types

Description Remarks and examples Also see

Description

This entry provides a quick reference for data types allowed by Stata. See **[U] 12 Data** for details.

Remarks and examples

stata.com

Storage type	Minimum	Maximum	Closest to 0 without being 0	Bytes
`byte`	-127	100	± 1	1
`int`	$-32{,}767$	32,740	± 1	2
`long`	$-2{,}147{,}483{,}647$	2,147,483,620	± 1	4
`float`	$-1.70141173319 \times 10^{38}$	$1.70141173319 \times 10^{38}$	$\pm 10^{-38}$	4
`double`	$-8.9884656743 \times 10^{307}$	$8.9884656743 \times 10^{307}$	$\pm 10^{-323}$	8

Precision for `float` is 3.795×10^{-8}.

Precision for `double` is 1.414×10^{-16}.

String storage type	Maximum length	Bytes
`str1`	1	1
`str2`	2	2
...	.	.
...	.	.
...	.	.
`str2045`	2045	2045
`strL`	2000000000	2000000000

Each element of data is said to be either type numeric or type string. The word "real" is sometimes used in place of numeric. Associated with each data type is a storage type.

Numbers are stored as `byte`, `int`, `long`, `float`, or `double`, with the default being `float`. `byte`, `int`, and `long` are said to be of integer type in that they can hold only integers.

Strings are stored as `str#`, for instance, `str1`, `str2`, `str3`, ..., `str2045`, or as `strL`. The number after the `str` indicates the maximum length of the string. A `str5` could hold the word "male", but not the word "female" because "female" has six characters. A `strL` can hold strings of arbitrary lengths, up to 2000000000 characters, and can even hold binary data containing embedded `\0` characters.

Stata keeps data in memory, and you should record your data as parsimoniously as possible. If you have a string variable that has maximum length 6, it would waste memory to store it as a `str20`. Similarly, if you have an integer variable, it would be a waste to store it as a `double`.

Precision of numeric storage types

`floats` have about 7 digits of accuracy; the magnitude of the number does not matter. Thus, 1234567 can be stored perfectly as a `float`, as can 1234567e+20. The number 123456789, however, would be rounded to 123456792. In general, this rounding does not matter.

If you are storing identification numbers, the rounding could matter. If the identification numbers are integers and take 9 digits or less, store them as `longs`; otherwise, store them as `doubles`. `doubles` have 16 digits of accuracy.

Stata stores numbers in binary, and this has a second effect on numbers less than 1. 1/10 has no perfect binary representation just as 1/11 has no perfect decimal representation. In `float`, .1 is stored as .10000000149011612. Note that there are 7 digits of accuracy, just as with numbers larger than 1. Stata, however, performs all calculations in double precision. If you were to store 0.1 in a `float` called `x` and then ask, say, `list if x==.1`, there would be nothing in the list. The .1 that you just typed was converted to double, with 16 digits of accuracy (.10000000000000014...), and that number is never equal to 0.1 stored with `float` accuracy.

One solution is to type `list if x==float(.1)`. The `float()` function rounds its argument to float accuracy; see [FN] **Programming functions**. The other alternative would be store your data as `double`, but this is probably a waste of memory. Few people have data that is accurate to 1 part in 10 to the 7th. Among the exceptions are banks, who keep records accurate to the penny on amounts of billions of dollars. If you are dealing with such financial data, store your dollar amounts as `doubles`.

Also see

[D] **compress** — Compress data in memory

[D] **destring** — Convert string variables to numeric variables and vice versa

[D] **encode** — Encode string into numeric and vice versa

[D] **format** — Set variables' output format

[D] **recast** — Change storage type of variable

[U] 12.2.2 Numeric storage types

[U] 12.4 Strings

[U] 12.5 Formats: Controlling how data are displayed

[U] 13.12 Precision and problems therein