Title

stata.com

creturn — Return c-class values

Description Menu Syntax Remarks and examples Also see

Description

Stata's c-class, `c()`, contains the values of system parameters and settings, along with certain constants such as the value of pi. `c()` values may be referred to but may not be assigned.

Menu

Data > Other utilities > List constants and system parameters

Syntax

```
creturn list
```

Remarks and examples

stata.com

The c-class values are presented under the following headings:

There may be other c-class values that have been added since the printing of this manual. Type `help creturn` for up-to-date information.

System values

`c(current_date)` returns the current date as a string in the format "*dd Mon yyyy*", where *dd* is the day of the month (if day is less than 10, a space and one digit are used); *Mon* is one of `Jan`, `Feb`, `Mar`, `Apr`, `May`, `Jun`, `Jul`, `Aug`, `Sep`, `Oct`, `Nov`, or `Dec`; and *yyyy* is the four-digit year.

Examples:

```
 1 Jan 2003
26 Mar 2007
28 Jan 2013
```

`c(current_time)` returns the current time as a string in the format "*hh*:*mm*:*ss*", where *hh* is the hour 00–23, *mm* is the minute 00–59, and *ss* is the second 00–59.

Examples:
`09:42:55`
`13:02:01`
`21:15:59`

`c(rmsg_time)` returns a numeric scalar equal to the elapsed time last reported as a result of `set rmsg on`; see [P] **rmsg**.

`c(stata_version)` returns a numeric scalar equal to the version of Stata that you are running. In Stata 15, this number is `15`; in Stata 15.1, `15.1`; and in Stata 16, `16`. This is the version of Stata that you are running, not the version being mimicked by the `version` command.

`c(version)` returns a numeric scalar equal to the version currently set by the `version` command; see [P] **version**.

`c(userversion)` returns a numeric scalar equal to the user version currently set by the `version` command; see [P] **version**.

`c(dyndoc_version)` returns a numeric scalar equal to the current version of dynamic documents Stata understands how to convert. Stata can convert any dynamic document with a version less than or equal to `c(dyndoc_version)`. The dynamic document version is set by the `<<dd_version>>` tag within the document.

`c(born_date)` returns a string in the same format as `c(current_date)` containing the date of the Stata executable that you are running; see [R] **update**.

`c(flavor)` returns a string containing `"IC"`, according to the version of Stata that you are running. `c(flavor) == "IC"` for Stata/MP and Stata/SE, as well as for Stata/IC. Think of `c(flavor) == "IC"` as meaning "IC or better", so Stata/IC and all higher flavors of Stata are considered to be "IC".

`c(bit)` returns a numeric scalar equal to `64` if you are using a 64-bit version of Stata and `32` if you are using a 32-bit version of Stata.

`c(SE)` returns a numeric scalar equal to `1` if you are running Stata/SE or Stata/MP and returns `0` otherwise. Think of `c(SE) == 1` as meaning "SE or better", so Stata/SE and Stata/MP both return `1`.

`c(MP)` returns a numeric scalar equal to `1` if you are running Stata/MP and `0` otherwise.

`c(processors)` returns a numeric scalar equal to the number of processors/cores that Stata/MP is currently set to use. It returns `1` if you are not running Stata/MP.

`c(processors_lic)` returns a numeric scalar equal to the number of processors/cores that your Stata/MP license allows. It returns `1` if you are not running Stata/MP.

`c(processors_mach)` returns a numeric scalar equal to the number of processors/cores that your computer has if you are running Stata/MP. It returns missing value (`.`) if you are not running Stata/MP.

`c(processors_max)` returns a numeric scalar equal to the maximum number of processors/cores that Stata/MP could use, which is equal to the minimum of `c(processors_lic)` and `c(processors_mach)`. It returns `1` if you are not running Stata/MP.

`c(mode)` returns a string containing `""` or `"batch"`, depending on whether Stata was invoked in interactive mode (the usual case) or batch mode (using, perhaps, the `-b` option of Stata for Unix).

`c(console)` returns a string containing `""` or `"console"`, depending on whether you are running a windowed version of Stata or Stata(console).

c(os) returns a string containing "MacOSX", "Unix", or "Windows", depending on the operating system that you are using. The list of alternatives, although complete as of the date of this writing, may not be complete.

c(osdtl) returns an additional string, depending on the operating system, providing the release number or other details about the operating system. c(osdtl) is often "".

c(hostname) returns a string containing the name of the host machine.

c(machine_type) returns a string that describes the hardware platform, such as "PC", "PC (64-bit x86-64)", or "Macintosh (Intel 64-bit)".

c(byteorder) returns a string containing "lohi" or "hilo", depending on the byte order of the hardware. Consider a two-byte integer. On some computers, the most significant byte is written first, so x'0001' (meaning the byte 00 followed by 01) would mean the number 1. Such computers are designated "hilo". Other computers write the least-significant byte first, so x'0001' would be 256, and 1 would be x'0100'. Such computers are designated "lohi".

c(username) returns the user ID (provided by the operating system) of the user currently using Stata.

Directories and paths

Note: The directory paths returned below usually end in a directory separator, so if you wish to construct the full path name of file abc.def in directory c(...), you code

```
...`c(...)'abc.def...
```

and not

```
...`c(...)'/abc.def...
```

If c(...) returns a directory name that does not end in a directory separator, a special note of the fact is made.

c(sysdir_stata) returns a string containing the name of the directory in which Stata is installed. More technically, c(sysdir_stata) returns the STATA directory as defined by sysdir; see [P] **sysdir**.

Example: C:\Program Files\Stata15/

The above example contains no typographical errors. Under Windows, the directory name will end in forward slash. That is so you can code things such as `c(sysdir_stata)'`filename'. If c(sysdir_stata) ended in backslash, Stata's macro expander would interpret the backslash as an escape character and so not expand `filename'.

c(sysdir_base) returns a string containing the name of the directory in which the original official ado-files that were shipped with Stata were installed.

Example: C:\Program Files\Stata15\ado\base/

c(sysdir_site) returns a string containing the name of the directory in which additions may be installed for sitewide use. More technically, c(sysdir_site) returns the SITE directory as defined by sysdir; see [P] **sysdir**.

Example: C:\Program Files\Stata15\ado\site/

c(sysdir_plus) returns a string containing the name of the directory in which additions written by others may be installed for personal use. More technically, c(sysdir_plus) returns the PLUS directory, as defined by sysdir; see [P] **sysdir**.

Example: C:\ado\plus/

`c(sysdir_personal)` returns a string containing the name of the directory in which additions written by you may be installed. More technically, `c(sysdir_personal)` returns the `PERSONAL` directory, as defined by `sysdir`; see [P] **sysdir**.

Example: `C:\ado\personal/`

`c(sysdir_oldplace)` identifies another directory in which community-contributed ado-files might be installed. `c(sysdir_oldplace)` maintains compatibility with very ancient versions of Stata.

`c(tmpdir)` returns a string containing the name of the directory used by Stata for temporary files.

Example: `/tmp`

`c(adopath)` returns a string containing the directories that are to be searched when Stata is attempting to locate an ado-file. The path elements are separated by a semicolon (`;`), and the elements themselves may be directory names, `"."` to indicate the current directory, or `sysdir` references.

Example: `BASE;SITE;.;PERSONAL;PLUS;OLDPLACE`

`c(pwd)` returns a string containing the current (working) directory.

Example: `C:\data`

Notice that `c(pwd)` does not end in a directory separator, so in a program, to save the name of the file `abc.def` prefixed by the current directory (for example, because you were about to change directories and still wanted to refer to the file), you would code

```
local file "`c(pwd)'/abc.def"
```

or

```
local file "`c(pwd)'`c(dirsep)'abc.def"
```

The second form is preferred if you want to construct "pretty" filenames, but the first form is acceptable because Stata understands a forward slash (`/`) as a directory separator.

`c(dirsep)` returns a string containing `"/"`.

Example: `/`

For Windows operating systems, a forward slash (`/`) is returned rather than a backslash (`\`). Stata for Windows understands both, but in programs, use of the forward slash is recommended because the backslash can interfere with Stata's interpretation of macro expansion characters. Do not be concerned if the result of your code is a mix of backslash and forward slash characters, such as `\a\b/myfile.dta`; Stata will understand it just as it would understand `/a/b/myfile.dta` or `\a\b\myfile.dta`.

System limits

`c(max_N_theory)` returns a numeric scalar reporting the maximum number of observations allowed.

`c(max_N_theory)` reports the maximum number of observations that Stata can process if it has enough memory. This is usually 2,147,483,647 for Stata/SE and Stata/IC and is 1,099,511,627,775 for Stata/MP.

`c(max_k_theory)` returns a numeric scalar reporting the maximum number of variables allowed. If you have Stata/MP or Stata/SE, you can change this number with `set maxvar`; see [D] **memory**.

`c(max_width_theory)` returns the theoretical maximum width allowed. The width of a dataset is defined as the sum of the byte lengths of its individual variables. If you had a dataset with two `int` variables, three `float`s, one `double`, and a `str20` variable, the width of the dataset would be $2 * 2 + 3 * 4 + 8 + 20 = 44$ bytes.

c(max_matsize) and c(min_matsize) each return a numeric scalar reporting the maximum and minimum values to which matsize may be set. If the version of Stata you are running does not allow the setting of matsize, the two values will be equal. c(matsize), documented under *Memory settings* below, returns the current value of matsize.

c(max_macrolen) and c(macrolen) each return a numeric scalar reporting the maximum length of macros. c(max_macrolen) and c(macrolen) may not be equal under Stata/MP or Stata/SE and will be equal otherwise. For Stata/MP or Stata/SE, macrolen is set according to maxvar: the length is long enough to hold a macro referring to every variable in the dataset.

c(charlen) returns a numeric scalar reporting the maximum length of a characteristic.

c(max_cmdlen) and c(cmdlen) each return a numeric scalar reporting the maximum length of a Stata command. c(max_cmdlen) and c(cmdlen) may not be equal under Stata/MP or Stata/SE and will be equal otherwise. For Stata/MP or Stata/SE, cmdlen is set according to maxvar: the length is long enough to hold a command referring to every variable in the dataset.

c(namelenbyte) returns a numeric scalar equal to 128, which is the current maximum length in bytes of names in Stata.

c(namelenchar) returns a numeric scalar equal to 32, which is the current maximum length in Unicode characters of names in Stata.

c(eqlen) returns the maximum length that Stata allows for equation names.

Numerical and string limits

c(mindouble), c(maxdouble), and c(epsdouble) each return a numeric scalar. c(mindouble) is the largest negative number that can be stored in the 8-byte double storage type. c(maxdouble) is the largest positive number that can be stored in a double. c(epsdouble) is the smallest nonzero, positive number (epsilon) that, when added to 1 and stored as a double, does not equal 1.

c(smallestdouble) returns a numeric scalar containing the smallest full-precision double that is bigger than zero. There are smaller positive values that can be stored; these are denormalized numbers. Denormalized numbers do not have full precision.

c(minfloat), c(maxfloat), and c(epsfloat) each return a numeric scalar that reports for the 4-byte float storage type what c(mindouble), c(maxdouble), and c(epsdouble) report for double.

c(minlong) and c(maxlong) return scalars reporting the largest negative number and the largest positive number that can be stored in the 4-byte, integer long storage type. There is no c(epslong), but if there were, it would return 1.

c(minint) and c(maxint) return scalars reporting the largest negative number and the largest positive number that can be stored in the 2-byte, integer int storage type.

c(minbyte) and c(maxbyte) return scalars reporting the largest negative number and the largest positive number that can be stored in the 1-byte, integer byte storage type.

c(maxstrvarlen) returns the longest str# string storage type allowed, which is 2045. Do not confuse c(maxstrvarlen) with c(macrolen). c(maxstrvarlen) corresponds to string variables stored in the data.

c(maxstrlvarlen) returns the length of the longest string that can be stored in a strL, which is 2,000,000,000.

c(maxvlabellen) returns the maximum length for one value label string, which is 32,000.

Current dataset

`c(N)` returns a numeric scalar equal to _N, the number of observations in the dataset in memory. In an expression, it makes no difference whether you refer to _N or `c(N)`. However, when used in expressions with the `by` prefix, `c(N)` does not change with the by-group like _N.

The advantage of `c(N)` is in nonexpression contexts. Say that you are calling a subroutine, `mysub`, which takes as an argument the number of observations in the dataset. Then you could code

```
local nobs = _N
mysub `nobs'
```

or

```
mysub `c(N)'
```

The second requires less typing.

`c(k)` returns a numeric scalar equal to the number of variables in the dataset in memory. `c(k)` is equal to `r(k)`, which is returned by `describe`.

`c(width)` returns a numeric scalar equal to the width, in bytes, of the dataset in memory. If you had a dataset with two `int` variables, three `float`s, one `double`, and a `str20` variable, the width of the dataset would be $2 * 2 + 3 * 4 + 8 + 20 = 44$ bytes. `c(width)` is equal to `r(width)`, which is returned by `describe`.

`c(changed)` returns a numeric scalar equal to `0` if the dataset in memory has not changed since it was last saved and `1` otherwise. `c(changed)` is equal to `r(changed)`, which is returned by `describe`.

`c(filename)` returns a string containing the filename last specified with a `use` or `save`, such as `"C:\Data\auto.dta"`. `c(filename)` is equal to `$S_FN`.

`c(filedate)` returns a string containing the date and time the file in `c(filename)` was last saved, such as `"7 Jul 2016 13:51"`. `c(filedate)` is equal to `$S_FNDATE`.

Memory settings

`c(memory)` returns a numeric scalar reporting the amount of memory, in bytes, currently allocated by Stata.

`c(maxvar)` returns a numeric scalar reporting the maximum number of variables currently allowed in a dataset, as set by `set maxvar` if you are running Stata/MP or Stata/SE. Otherwise, `c(maxvar)` is a constant.

`c(matsize)` returns a numeric scalar reporting the current value of `matsize`, as set by `set matsize`.

`c(niceness)` returns a numeric scalar recording how soon Stata gives back unused segments to the operating system.

`c(min_memory)` returns a numeric scalar recording the minimum value to which memory can be reduced when its memory is unused.

`c(max_memory)` returns a numeric scalar recording the maximum amount of memory that Stata may allocate.

`c(segmentsize)` returns a numeric scalar recording the size of the segments in which memory is allocated.

Output settings

c(more) returns a string containing "on" or "off", according to the current set more setting.

c(rmsg) returns a string containing "on" or "off", according to the current set rmsg setting.

c(dp) returns a string containing "period" or "comma", according to the current set dp setting.

c(linesize) returns a numeric scalar equal to the current set linesize setting.

c(pagesize) returns a numeric scalar equal to the current set pagesize setting.

c(logtype) returns a string containing "smcl" or "text", according to the current set logtype setting.

c(noisily) returns a numeric scalar equal to 0 if output is being suppressed and 1 if output is being displayed; see [P] **quietly**.

c(notifyuser) (Mac only) returns a string containing "on" or "off", according to the current set notifyuser setting.

c(playsnd) (Mac only) returns a string containing "on" or "off", according to the current set playsnd setting.

c(include_bitmap) (Mac only) returns a string containing "on" or "off", according to the current set include_bitmap setting.

c(level) returns a numeric scalar equal to the current set level setting.

c(clevel) returns a numeric scalar equal to the current set clevel setting.

c(showbaselevels) returns a string containing "", "on", "off", or "all", according to the current set showbaselevels setting. See [R] **set showbaselevels**.

c(showemptycells) returns a string containing "", "on", or "off", according to the current set showemptycells setting. See [R] **set showbaselevels**.

c(showomitted) returns a string containing "", "on", or "off", according to the current set showomitted setting. See [R] **set showbaselevels**.

c(fvlabel) returns a string containing "on" or "off", according to the current set fvlabel setting. See [R] **set showbaselevels**.

c(fvwrap) returns a numeric scalar equal to the current set fvwrap setting. See [R] **set showbaselevels**.

c(fvwrapon) returns a string containing "word" or "width", according to the current set fvwrapon setting. See [R] **set showbaselevels**.

c(fvtrack) returns a string containing "term" or "factor", according to the current set fvtrack setting.

c(lstretch) returns a string containing "", "on", or "off", according to the current set lstretch setting.

c(cformat) returns a string containing the current set cformat setting. See [R] **set cformat**.

c(sformat) returns a string containing the current set sformat setting. See [R] **set cformat**.

c(pformat) returns a string containing the current set pformat setting. See [R] **set cformat**.

c(coeftabresults) returns a string containing "on" or "off", according to the current set coeftabresults setting.

Interface settings

`c(dockable)` (Windows only) returns a string containing `"on"` or `"off"`, according to the current `set dockable` setting.

`c(dockingguides)` (Windows only) returns a string containing `"on"` or `"off"`, according to the current `set dockingguides` setting.

`c(locksplitters)` (Windows only) returns a string containing `"on"` or `"off"`, according to the current `set locksplitters` setting.

`c(pinnable)` (Windows only) returns a string containing `"on"` or `"off"`, according to the current `set pinnable` setting.

`c(doublebuffer)` (Windows only) returns a string containing `"on"` or `"off"`, according to the current `set doublebuffer` setting.

`c(reventries)` returns a numeric scalar containing the maximum number of commands stored by the Review window.

`c(fastscroll)` (Unix and Windows only) returns a string containing `"on"` or `"off"`, according to the current `set fastscroll` setting.

`c(revkeyboard)` (Mac only) returns a string containing `"on"` or `"off"`, according to the current `set revkeyboard` setting.

`c(varkeyboard)` (Mac only) returns a string containing `"on"` or `"off"`, according to the current `set varkeyboard` setting.

`c(smoothfonts)` (Mac only) returns a string containing `"on"` or `"off"`, according to the current `set smoothfonts` setting.

`c(linegap)` returns a numeric scalar equal to the current `set linegap` setting. If `set linegap` is irrelevant under the version of Stata that you are running, `c(linegap)` returns a system missing value.

`c(scrollbufsize)` returns a numeric scalar equal to the current `set scrollbufsize` setting. If `set scrollbufsize` is irrelevant under the version of Stata that you are running, `c(scrollbufsize)` returns a system missing value.

`c(maxdb)` returns a numeric scalar containing the maximum number of dialog boxes whose contents are remembered from one invocation to the next during a session; see [R] **db**.

Graphics settings

`c(graphics)` returns a string containing `"on"` or `"off"`, according to the current `set graphics` setting.

`c(autotabgraphs)` (Windows only) returns a string containing `"on"` or `"off"`, according to the current `set autotabgraphs` setting.

`c(scheme)` returns the name of the current `set scheme` setting.

`c(printcolor)` returns `"automatic"`, `"asis"`, `"gs1"`, `"gs2"`, or `"gs3"`, according to the current `set printcolor` setting.

`c(copycolor)` (Mac and Windows only) returns `"automatic"`, `"asis"`, `"gs1"`, `"gs2"`, or `"gs3"`, according to the current `set copycolor` setting.

`c(maxbezierpath)` (Mac only) returns a numeric scalar containing the maximum number of lines that can be added to a Bézier path when rendering a Stata graph to a screen; see `set maxbezierpath`.

Efficiency settings

c(adosize) returns a numeric scalar equal to the current set adosize setting.

Network settings

c(checksum) returns a string containing "on" or "off", according to the current set checksum setting.

c(timeout1) returns a numeric scalar equal to the current set timeout1 setting.

c(timeout2) returns a numeric scalar equal to the current set timeout2 setting.

c(httpproxy) returns a string containing "on" or "off", according to the current set httpproxy setting.

c(httpproxyhost) returns a string containing the name of the proxy host or "" if no proxy host is set. c(httpproxyhost) is relevant only if c(httpproxy) = "on".

c(httpproxyport) returns a numeric scalar equal to the proxy port number. c(httpproxyport) is relevant only if c(httpproxy) = "on".

c(httpproxyauth) returns a string containing "on" or "off", according to the current set httpproxyauth setting. c(httpproxyauth) is relevant only if c(httpproxy) = "on".

c(httpproxyuser) returns a string containing the name of the proxy user, if one is set, or "" otherwise. c(httpproxyuser) is relevant only if c(httpproxy) = "on" and c(httpproxyauth) = "on".

c(httpproxypw) returns a string containing "*" if a password is set or "" otherwise. c(httpproxypw) is relevant only if c(httpproxy) = "on" and c(httpproxyauth) = "on".

Update settings

c(update_query) (Mac and Windows only) returns a string containing "on" or "off", according to the current set update_query setting.

c(update_interval) (Mac and Windows only) returns a numeric scalar containing the current set update_interval setting.

c(update_prompt) (Mac and Windows only) returns a string containing "on" or "off", according to the current set update_prompt setting.

Trace (program debugging) settings

c(trace) returns a string containing "on" or "off", according to the current set trace setting.

c(tracedepth) returns a numeric scalar reporting the current set tracedepth setting.

c(tracesep) returns a string containing "on" or "off", according to the current set tracesep setting.

c(traceindent) returns a string containing "on" or "off", according to the current set traceindent setting.

c(traceexpand) returns a string containing "on" or "off", according to the current set traceexpand setting.

`c(tracenumber)` returns a string containing `"on"` or `"off"`, according to the current `set tracenumber` setting.

`c(tracehilite)` returns a string containing "*pattern*", according to the current `set tracehilite` setting.

Mata settings

`c(matastrict)` returns a string containing `"on"` or `"off"`, according to the current `set matastrict` setting.

`c(matalnum)` returns a string containing `"on"` or `"off"`, according to the current `set matalnum` setting.

`c(mataoptimize)` returns a string containing `"on"` or `"off"`, according to the current `set mataoptimize` setting.

`c(matafavor)` returns a string containing `"space"` or `"speed"`, according to the current `set matafavor` setting.

`c(matacache)` returns a numeric scalar containing the maximum amount of memory, in kilobytes, that may be consumed before Mata starts looking to drop autoloaded functions that are not currently being used.

`c(matalibs)` returns a string containing the names in order of the `.mlib` libraries to be searched; see [M-1] **how**.

`c(matamofirst)` returns a string containing `"on"` or `"off"`, according to the current `set matamofirst` setting.

Unicode settings

`c(locale_ui)` returns a string containing the locale that specifies the localization package for the user interface. See [P] **set locale_ui**.

`c(locale_functions)` returns a string containing the default locale for string functions. See [P] **set locale_functions**.

`c(locale_icudflt)` returns a string containing the default ICU locale. See **[U] 12.4.2.4 Locales in Unicode**.

Other settings

`c(type)` returns a string containing `"float"` or `"double"`, according to the current `set type` setting.

`c(maxiter)` returns a numeric scalar equal to the current `set maxiter` setting.

`c(searchdefault)` returns a string containing `"local"`, `"net"`, or `"all"`, according to the current search default setting.

`c(rng)` returns a string containing the current `set rng` setting. This controls which random-number generator Stata will use. Possible values are `"mt64"`, which specifies to always use the 64-bit Mersenne Twister random-number generator; `"mt64s"`, which specifies to always use the 64-bit Mersenne Twister stream random-number generator; `"kiss32"`, which specifies to always use the 32-bit KISS (keep it simple stupid) random-number generator; or `"default"`, which specifies to

let Stata choose between these random-number generators based on version control. Stata's default random-number generator in the absence of version control and with `set rng default` is the 64-bit Mersenne Twister. See [R] **set rng**.

`c(rng_current)` returns a string containing the random-number generator currently in effect, that is, `"mt64"`, `"mt64s"`, or `"kiss32"`, depending on the current `set rng` setting. If `set rng` is currently set to `"default"`, then `c(rng_current)` depends on the current user version. See [P] **version**.

`c(rngstate)` returns a string containing the current state of the `runiform()` random-number generator. You can initialize the state of the random-number generator with `set seed`, and you can restore the state of the random-number generator to a saved state with `set rngstate`. See [R] **set seed**.

`c(rngseed_mt64s)` returns the seed last set for the stream random-number generator (`mt64s`). See [R] **set rngstream**.

`c(rngstream)` returns the current stream of the stream random-number generator (`mt64s`). See [R] **set rngstream**.

`c(varabbrev)` returns a string containing `"on"` or `"off"`, according to the current `set varabbrev` setting.

`c(emptycells)` returns a string containing `"keep"` or `"drop"`, according to the current `set emptycells` setting.

`c(haverdir)` (Windows only) returns a string containing the name of the directory that you specified to contain the Haver databases; see `set haverdir` in [D] **import haver**.

`c(odbcmgr)` (Mac and Unix only) returns a string containing `"iodbc"` or `"unixodbc"`, according to the current `set odbcmgr` setting.

`c(odbcdriver)` returns a string containing `"unicode"` or `"ansi"`, according to the current `set odbcdriver` setting.

`c(fredkey)` returns the current API key, according to the current `set fredkey` setting.

Other

`c(pi)` returns a numerical scalar equal to `_pi`, the value of the ratio of the circumference to the diameter of a circle. In an expression context, it makes no difference whether you use `c(pi)` or `_pi`. `c(pi)`, however, may be used (enclosed in single quotes) in other contexts.

`c(alpha)` returns a string containing `"a b c d e f g h i.."`.

`c(ALPHA)` returns a string containing `"A B C D E F G H I.."`.

`c(Mons)` returns a string containing `"Jan Feb Mar Apr M.."`.

`c(Months)` returns a string containing `"January February .."`.

`c(Wdays)` returns a string containing `"Sun Mon Tue Wed T.."`.

`c(Weekdays)` returns a string containing `"Sunday Monday Tue.."`.

`c(rc)` returns a numerical scalar equal to `_rc`, the value set by the `capture` command. In an expression context, it makes no difference whether you use `c(rc)` or `_rc`. `c(rc)`, however, may be used (enclosed in single quotes) in other contexts. This is less important than it sounds because you could just as easily type `‘=_rc’`.

Also see

[P] **return** — Return stored results

[R] **query** — Display system parameters

[R] **set** — Overview of system parameters