|  |  |
| --- | --- |
| library {base} | R Documentation |

**Loading/Attaching and Listing of Packages**

**Description**

library and require load and attach add-on packages.

**Usage**

library(package, help, pos = 2, lib.loc = NULL,

character.only = FALSE, logical.return = FALSE,

warn.conflicts = TRUE, quietly = FALSE,

verbose = getOption("verbose"))

require(package, lib.loc = NULL, quietly = FALSE,

warn.conflicts = TRUE,

character.only = FALSE)

**Arguments**

|  |  |
| --- | --- |
| package, help | the name of a package, given as a [name](http://127.0.0.1:14695/library/base/help/name) or literal character string, or a character string, depending on whether character.only is FALSE (default) or TRUE). |
| pos | the position on the search list at which to attach the loaded namespace. Can also be the name of a position on the current search list as given by [search](http://127.0.0.1:14695/library/base/help/search)(). |
| lib.loc | a character vector describing the location of **R** library trees to search through, or NULL. The default value of NULL corresponds to all libraries currently known to [.libPaths](http://127.0.0.1:14695/library/base/help/.libPaths)(). Non-existent library trees are silently ignored. |
| character.only | a logical indicating whether package or help can be assumed to be character strings. |
| logical.return | logical. If it is TRUE, FALSE or TRUE is returned to indicate success. |
| warn.conflicts | logical. If TRUE, warnings are printed about [conflicts](http://127.0.0.1:14695/library/base/help/conflicts) from attaching the new package. A conflict is a function masking a function, or a non-function masking a non-function. |
| verbose | a logical. If TRUE, additional diagnostics are printed. |
| quietly | a logical. If TRUE, no message confirming package attaching is printed, and most often, no errors/warnings are printed if package attaching fails. |

**Details**

library(package) and require(package) both load the namespace of the package with name package and attach it on the search list. require is designed for use inside other functions; it returns FALSE and gives a warning (rather than an error as library() does by default) if the package does not exist. Both functions check and update the list of currently attached packages and do not reload a namespace which is already loaded. (If you want to reload such a package, call [detach](http://127.0.0.1:14695/library/base/help/detach)(unload = TRUE) or [unloadNamespace](http://127.0.0.1:14695/library/base/help/unloadNamespace) first.) If you want to load a package without attaching it on the search list, see [requireNamespace](http://127.0.0.1:14695/library/base/help/requireNamespace).

To suppress messages during the loading of packages use [suppressPackageStartupMessages](http://127.0.0.1:14695/library/base/help/suppressPackageStartupMessages): this will suppress all messages from **R** itself but not necessarily all those from package authors.

If library is called with no package or help argument, it lists all available packages in the libraries specified by lib.loc, and returns the corresponding information in an object of class "libraryIQR". (The structure of this class may change in future versions.) Use .packages(all = TRUE) to obtain just the names of all available packages, and [installed.packages](http://127.0.0.1:14695/library/base/help/installed.packages)() for even more information.

library(help = somename) computes basic information about the package **somename**, and returns this in an object of class "packageInfo". (The structure of this class may change in future versions.) When used with the default value (NULL) for lib.loc, the attached packages are searched before the libraries.

**Value**

Normally library returns (invisibly) the list of attached packages, but TRUE or FALSE if logical.return is TRUE. When called as library() it returns an object of class "libraryIQR", and for library(help=), one of class "packageInfo".

require returns (invisibly) a logical indicating whether the required package is available.

**Licenses**

Some packages have restrictive licenses, and there is a mechanism to allow users to be aware of such licenses. If [getOption](http://127.0.0.1:14695/library/base/help/getOption)("checkPackageLicense") == TRUE, then at first use of a package with a not-known-to-be-FOSS (see below) license the user is asked to view and accept the license: a list of accepted licenses is stored in file ‘~/.R/licensed’. In a non-interactive session it is an error to use such a package whose license has not already been accepted.

Free or Open Source Software (FOSS, e.g. <https://en.wikipedia.org/wiki/FOSS>) packages are determined by the same filters used by [available.packages](http://127.0.0.1:14695/library/base/help/available.packages) but applied to just the current package, not its dependencies.

There can also be a site-wide file ‘R\_HOME/etc/licensed.site’ of packages (one per line).

**Formal methods**

library takes some further actions when package **methods** is attached (as it is by default). Packages may define formal generic functions as well as re-defining functions in other packages (notably **base**) to be generic, and this information is cached whenever such a namespace is loaded after **methods** and re-defined functions ([implicit generic](http://127.0.0.1:14695/library/base/help/implicit%20generic)s) are excluded from the list of conflicts. The caching and check for conflicts require looking for a pattern of objects; the search may be avoided by defining an object .noGenerics (with any value) in the namespace. Naturally, if the package *does* have any such methods, this will prevent them from being used.

**Note**

library and require can only load/attach an *installed* package, and this is detected by having a ‘DESCRIPTION’ file containing a Built: field.

Under Unix-alikes, the code checks that the package was installed under a similar operating system as given by R.version$platform (the canonical name of the platform under which R was compiled), provided it contains compiled code. Packages which do not contain compiled code can be shared between Unix-alikes, but not to other OSes because of potential problems with line endings and OS-specific help files. If sub-architectures are used, the OS similarity is not checked since the OS used to build may differ (e.g. i386-pc-linux-gnu code can be built on an x86\_64-unknown-linux-gnu OS).

The package name given to library and require must match the name given in the package's ‘DESCRIPTION’ file exactly, even on case-insensitive file systems such as are common on Windows and macOS.

**References**

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

**See Also**

[.libPaths](http://127.0.0.1:14695/library/base/help/.libPaths), [.packages](http://127.0.0.1:14695/library/base/help/.packages).

[attach](http://127.0.0.1:14695/library/base/help/attach), [detach](http://127.0.0.1:14695/library/base/help/detach), [search](http://127.0.0.1:14695/library/base/help/search), [objects](http://127.0.0.1:14695/library/base/help/objects), [autoload](http://127.0.0.1:14695/library/base/help/autoload), [requireNamespace](http://127.0.0.1:14695/library/base/help/requireNamespace), [library.dynam](http://127.0.0.1:14695/library/base/help/library.dynam), [data](http://127.0.0.1:14695/library/base/help/data), [install.packages](http://127.0.0.1:14695/library/base/help/install.packages) and [installed.packages](http://127.0.0.1:14695/library/base/help/installed.packages); [INSTALL](http://127.0.0.1:14695/library/base/help/INSTALL), [REMOVE](http://127.0.0.1:14695/library/base/help/REMOVE).

The initial set of packages attached is set by [options](http://127.0.0.1:14695/library/base/help/options)(defaultPackages=): see also [Startup](http://127.0.0.1:14695/library/base/help/Startup).

**Examples**

library() # list all available packages

library(lib.loc = .Library) # list all packages in the default library

library(help = splines) # documentation on package 'splines'

library(splines) # attach package 'splines'

require(splines) # the same

search() # "splines", too

detach("package:splines")

# if the package name is in a character vector, use

pkg <- "splines"

library(pkg, character.only = TRUE)

detach(pos = match(paste("package", pkg, sep = ":"), search()))

require(pkg, character.only = TRUE)

detach(pos = match(paste("package", pkg, sep = ":"), search()))

require(nonexistent) # FALSE

## Not run:

## if you want to mask as little as possible, use

library(mypkg, pos = "package:base")

## End(Not run)