Package 'import'

June 22, 2015

Type Package

Title An Import Mechanism for R

Version 1.1.0

Author Stefan Milton Bache

Maintainer Stefan Milton Bache <stefan@stefanbache.dk>

Description This is an alternative mechanism for importing objects from packages. The syntax allows for importing multiple objects from a package with a single command in an expressive way. The import package bridges some of the gap between using library (or require) and direct (single-object) imports. Furthermore the imported objects are not placed in the current environment. It is also possible to import objects from stand-alone .R files. For more information, refer to the package vignette.

License MIT + file LICENSE

ByteCompile TRUE

URL https://github.com/smbache/import

BugReports https://github.com/smbache/import/issues

Suggests knitr

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2015-06-22 01:47:36

R topics documented:

from *Import objects from a package.*

Description

The `import::from` and `import::into` functions provide an alternative way to import objects (e.g. functions) from packages. It is sometimes preferred over using `library` (or `require`) which will import all objects exported by the package. The benefit over `obj <-pkg::obj` is that the imported objects will (by default) be placed in a separate entry in the search path (which can be specified), rather in the global/current environment. Also, it is a more succinct way of importing several objects. Note that the two functions are symmetric, and usage is a matter of preference and whether specifying the `.into` argument is desired. The function `import::here` is short-hand for `import::from` with `.into = ""` which imports into the current environment.

Usage

```
from(.from, ..., .into = "imports", .library = .libPaths()[1L])

here(..., .from, .library = .libPaths()[1L])

into(.into, ..., .from, .library = .libPaths()[1L])
```

Arguments

`.from`	The package from which to import, or the path to a stand-alone .R file.
`...`	Names or name-value pairs specifying objects to import. If arguments are named, then the imported object will have this new name.
`.into`	The name of the search path entry. Use `""` to import into the current environment.
`.library`	character specifying the library to use. Defaults to the latest specified library. This is not used if `.from` is a stand-alone R file.

Details

The function arguments can be quoted or unquoted as with e.g. `library`. In any case, the character representation is used when unquoted arguments are provided (and not the value of objects with matching names). The period in the argument names `.into` and `.from` are there to avoid name clash with package objects. The double-colon syntax `import::from` allows for imports of exported objects (and lazy data) only. To import objects that are not exported, use triple-colon syntax, e.g. `import:::from`. The two ways of calling the `import` functions analogue the `::` and `:::` operators themselves.

Note that the `import` functions usually have the (intended) side-effect of altering the search path, as they (by default) import objects into the "imports" search path entry rather than the global environment.

The import package is not meant to be loaded with library (and will output a message about this if attached), but rather it is named to make the function calls expressive without the need to preload, i.e. it is designed to be used explicitly with the :: syntax, e.g. import::from(pkg, x, y).

It is also possible to import objects from an R file, in which case the relative (or absolute) path to the file is provided, rather than a package name.

Value

a reference to the environment with the imports or NULL if into = "", invisibly.

Examples

```
import::from(parallel, makeCluster, parLapply)
import::into("imports:parallel", makeCluster, parLapply, .from = parallel)
```

---

import *An Import Mechanism for R*

Description

This is an alternative mechanism for importing objects from packages. The syntax allows for importing multiple objects from a package with a single command in an expressive way. The import package bridges some of the gap between using library (or require) and direct (single-object) imports. Furthermore the imported objects are not placed in the current environment (although possible), but in a named entry in the search path.

Details

This package is not intended for use with library. It is named to make calls like import::from(pkg, fun1, fun2) expressive. Using the import functions complements the standard use of library(pkg)(when most objects are needed, and context is clear) and obj <- pkg::obj (when only a single object is needed).

Author(s)

Stefan Milton Bache

See Also

For usage instructions and examples, see from, into, or here.

make_import_call *Make an import call object.*

Description

The import call constructed by this function can be evaluated with `eval` to perform the actual import.

Usage

```
make_import_call(params, exports_only)
```

Arguments

`params`	list of parameters to substitute in the call.
`exports_only`	logical indicating whether only exported objects are allowed.

Value

A call object.

symbol_as_character *Convert a possible symbol to character.*

Description

Convert a possible symbol to character.

Usage

```
symbol_as_character(symbol)
```

Arguments

`symbol`	A symbol or character (of length one)

Value

character

`symbol_list` *Process a symbol list provided passed as ellipses*

Description

This function is used within a function that receives dot-arguments, and will create a named character vector. It ensures that all entries are named, and will use the value as name when it is missing.

Usage

```
symbol_list(...)
```

Arguments

`...`	dot arguments as passed to

Value

A named character vector.

Index