# Package ‘import’

**September 24, 2020**

**Type**  Package  
**Title**  An Import Mechanism for R  
**Version**  1.2.0  
**Description**  Alternative mechanism for importing objects from packages and R modules. The syntax allows for importing multiple objects 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.  
**License**  MIT + file LICENSE  
**ByteCompile**  TRUE  
**URL**  [https://github.com/rticulate/import](https://github.com/rticulate/import)  
**BugReports**  [https://github.com/rticulate/import/issues](https://github.com/rticulate/import/issues)  
**Suggests**  knitr, rmarkdown, magrittr, testthat  
**VignetteBuilder**  knitr  
**RoxygenNote**  7.1.1  
**NeedsCompilation**  no  
**Author**  Stefan Milton Bache [aut], Magnus Thor Tørfason [aut, cre]  
**Maintainer**  Magnus Thor Tørfason <m@zulutime.net>  
**Repository**  CRAN  
**Date/Publication**  2020-09-24 12:30:11 UTC

## R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>from</td>
<td>2</td>
</tr>
<tr>
<td>import</td>
<td>4</td>
</tr>
</tbody>
</table>

Index 6
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 imports into the current environment.

Usage

from(
  .from,
  ...,
  .into = "imports",
  .library = .libPaths()[1L],
  .directory = ".",
  .all = (length(.except) > 0),
  .except = character(),
  .chdir = TRUE,
  .character_only = FALSE
)

together(
  .to,
  ...,
  .from,
  .library = .libPaths()[1L],
  .directory = ".",
  .all = (length(.except) > 0),
  .except = character(),
  .chdir = TRUE,
  .character_only = FALSE
)

here(
  .from,
  ...,
  .library = .libPaths()[1L],
  .directory = ".",
  .all = (length(.except) > 0),
  .except = character(),
  .chdir = TRUE,
  .character_only = FALSE
)

into(
  .into,
  ...,
  .from,
  .library = .libPaths()[1L],
  .directory = ".",
  .all = (length(.except) > 0),
  .except = character(),
from 3

.chdir = TRUE,
.character_only = FALSE
)

Arguments

.from The package from which to import.
... 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. Enclosing the value in curly brackets causes the parameter to be treated as an actual environment value, rather than the name of an environment. Using .into={environment()} causes imports to be made into the current environment; .into="" is an equivalent shorthand value.
.library character specifying the library to use when importing from packages. Defaults to the latest specified library.
.directory character specifying the directory to use when importing from modules. Defaults to the current working directory. If .from is a module specified using an absolute path (i.e. starting with /), this parameter is ignored.
.all logical specifying whether all available objects in a package or module should be imported. It defaults to FALSE unless .exclude is being used to omit particular functions.
.except character vector specifying any objects that should not be imported. Any values specified here override both values provided in ... and objects included because of the .all parameter
.chdir logical specifying whether to change directories before sourcing a module (this parameter is ignored for libraries)
.character_only A logical indicating whether .from and ... can be assumed to be character strings. (Note that this parameter does not apply to how the .into parameter is handled).

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 loading before use, i.e. it is designed to be used explicitly with the :: syntax, e.g. import::from(pkg,x,y).
import

Value

a reference to the environment containing the imported objects.

Packages vs. modules

import can either be used to import objects either from R packages or from R source files. If the .from parameter ends with `.R’ or `.r’, import will look for a source file to import from. A source file in this context is referred to as a module in the documentation.

Package Versions

With import you can specify package version requirements. To do this add a requirement in parentheses to the package name (which then needs to be quoted), e.g import::from("parallel (>= 3.2.0)", ...). You can use the operators <, >, <=, >=, ==, !=. Whitespace in the specification is irrelevant.

See Also

Helpful links:

- https://import.rticulate.org
- https://github.com/rticulate/import
- https://github.com/rticulate/import/issues

Examples

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

Stefan Milton Bache

See Also

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

Helpful links:

- https://import.rticulate.org
- https://github.com/rticulate/import
- https://github.com/rticulate/import/issues
Index

from, 2, 5
here, 5
here (from), 2
import, 4
into, 5
into (from), 2