Package 'envir'

May 8, 2026

Title Manage R Environments Better

Version 0.3.0

Description Provides a small set of functions for managing R environments, with defaults designed to encourage usage patterns that scale well to larger code bases. It provides: import_from(), a flexible way to assign bindings that defaults to the current environment; include(), a vectorized alternative to base::source() that also default to the current environment; and attach_eval() and attach_source(), a way to evaluate expressions in attached environments. Together, these (and other) functions pair to provide a robust alternative to base::library() and base::source().

License GPL-3

URL https://t-kalinowski.github.io/envir/

BugReports https://github.com/t-kalinowski/envir/issues

Encoding UTF-8

RoxygenNote 7.3.1

NeedsCompilation no

Author Tomasz Kalinowski [aut, cre]

Maintainer Tomasz Kalinowski <kalinowskit@gmail.com>

Repository CRAN

Date/Publication 2024-04-19 13:22:42 UTC

Contents

attach_eval *Evaluate R expressions in an attached environment.*

Description

Evaluate R expressions in an attached environment.

Usage

```
attach_eval(
  unquoted_expr,
  name = "local:utils",
  pos = 2L,
  warn.conflicts = TRUE,
  ...,
  expr = substitute(unquoted_expr),
  mask.ok = NULL
)
```

Arguments

unquoted_expr	The expression to be evaluated, This is automatically quoted.
name	The environment name. If an environment of that name already exists, it is reused, otherwise, a new environment is attached.
pos	The position where to attach the environment, if creating a new one. If an environment of name already exists, pos is ignored.
warn.conflicts	logical. If TRUE (the default), print warnings about objects in the attached environment that that are masking or masked by other objects of the same name.
...	Ignored.
expr	An R language object. This is an escape hatch from the automatic quoting of unquoted_expr.
mask.ok	character vector of names of objects that can mask objects on the search path without signaling a warning if warn.conflicts is TRUE.

Value

The result after evaluating expr, invisibly.

Examples

```
attach_eval({
  my_helper_funct <- function(x, y) x + y
})

search() # environment "local:utils" is now attached
my_helper_funct(1, 1) # the local utility is now available
```

```
detach(local:utils) # cleanup
```

`attach_source` *Source R files into an attached environment*

Description

Source R files into an attached environment

Usage

```
attach_source(
  ...,
  name = as_tidy_env_name(c(...), prefix = "source:"),
  recursive = FALSE,
  pos = 2L,
  chdir = FALSE,
  warn.conflicts = TRUE,
  mask.ok = NULL,
  parent = .GlobalEnv
)
```

Arguments

`...`	filepaths to R files, or paths to directories containing R files.
`name`	A string, the name for the attached environment. By default, the name is constructed from paths supplied to `...`. If the requested name is not on the search path, a new environment of this name is attached.
`recursive`	If directories are passed to `...`, whether to search them recursively.
`pos`	The position where to attach the environment, if creating a new one. If an environment of `name` already exists, `pos` is ignored.
`chdir`	logical. if TRUE, the R working directory is temporarily changed to the directory containing the file(s) being sourced.
`warn.conflicts`	logical. If TRUE (the default), print warnings about objects in the attached environment that that are masking or masked by other objects of the same name. If the environment specified by `name` is was attached previously, then only newly defined objects are warned about. N.B., Even though the name is `warn.conflicts`, the messages about conflicts are not `warning()`s but `packageStartupMessage()`s, and can be suppressed with `suppressPackageStartupMessages()`
`mask.ok`	character vector of names of objects that can mask objects on the search path without signaling a warning if `warn.conflicts` is `TRUE`. The sourced R script can also define `.mask.OK` in the R environment, which has the same effect as passing it as an argument.

parent — R environment. If an environment specified by name is not already attached, then the supplied R scripts are first sourced into a new environment with the supplied parent. The default of globalenv() enables calling library() in the scripts and having the subsequent code in the scripts "see" the attached packages.

Value

The attached environment, invisibly.

See Also

import_from, set_library_default_pos

---

import_from *import objects*

Description

This is inspired by the python idiom from module import object as new_name.

Usage

```
import_from(
  x,
  ...,
  .into = parent.frame(),
  .parent = .GlobalEnv,
  .overwrite = interactive(),
  .chdir = FALSE,
  .recursive = FALSE,
  .pos = 2L
)
```

Arguments

x — a bare symbol name of a package, a character vector of filepaths, an environment (which could be a python module), or any object with names and [[methods defined.

... — objects to import from x into .into. if named, the name will be the the new name after import. Alternatively, you can also supply the wildcard string "*" or "**", along with some additional overrides. See examples for details.

.into — An R environment, or something coercible to one by as.environment, or a character string that is the name of a (potentially new) attached environment. The default is the current frame.

.parent, .chdir, .recursive — Only applicable if x is a character vector of filepaths to R scripts, in which case these are passed on to include (chdir, recursive) or new.env(parent)

.overwrite	One of "warn", "error" or "ignore". Can also be a boolean TRUE (same as "ignore") or FALSE (same as "error"). What should be done if the requested import operation would overwrite an existing object. Character arguments can be abbreviated as partial matching is performed.
.pos	Only applicable if .into is a string that is the name of a new environment that will be attached, in which case this will be the position on new environment on the search path.

Value

The R environment or object that x resolved to, invisibly.

Note

If x is a package name, then no check is performed to ensure the object being imported is an exported function. As such, import_from() can be used to access package internal objects, though doing so is usually bad practice.

Examples

```
show_whats_imported <- function(...) {
  import_from(...)
  setdiff(names(environment()), "...")
}

## Importing from an R package
# import one object
show_whats_imported(envir, include)

# rename an object on import
show_whats_imported(envir, sys_source = include)

# import all NAMESPACE exports
show_whats_imported(envir, "*")
show_whats_imported(envir) # missing `...` is interpreted as "*"

# import all NAMESPACE exports, except for `include`
show_whats_imported(envir, "*", -include)

# import all NAMESPACE exports, except rename `include` to `sys_source`
show_whats_imported(envir, "*", sys_source = include)

# exclude more than one
show_whats_imported(envir, "*", -include, -attach_eval)
show_whats_imported(envir, "*", -c(include, attach_eval))

# import all NAMESPACE exports, also one internal function names `find_r_files`
show_whats_imported(envir, "*", find_r_files)

# import ALL package functions, including all internal functions
show_whats_imported(envir, "**")
```

```
# import ALL objects in the package NAMESPACE, including R's NAMESPACE machinery
show_whats_imported(envir, "***")


## Importing from R files
# setup
dir.create(tmpdir <- tempfile())
owd <- setwd(tmpdir)
writeLines(c("useful_function <- function() 'I am useful'",
             ".less_useful_fn <- function() 'less useful'"),
           "my_helpers.R")

# import one function by name
show_whats_imported("my_helpers.R", useful_function)

# import all objects whose names don't start with a "." or "_"
show_whats_imported("my_helpers.R", "*")

# import all objects
show_whats_imported("my_helpers.R", "**")

# if the filepath to your scripts is stored in a variable, supply it in a call
x <- "my_helpers.R"
try(show_whats_imported(x)) # errors out, because no package 'x'
# to force the value to be used, just supply it as a call rather than a bare symbol.
# the simplest call can be just wrapping in () or {}
show_whats_imported({x})
show_whats_imported((x))
show_whats_imported(c(x))
show_whats_imported({{x}}) # tidyverse style unquoting

## Importing R objects

# if you have an actual R object that you want to import from, you will
# have to supply it in a call
x <- list(obj1 = "one", obj2 = "two")
show_whats_imported({x})

## Not run:
  # don't run this so we don't take a reticulate dependency
  import_from(reticulate, py_module = import) # rename object on import

  # import one object
  show_whats_imported(py_module("numpy"), random)

  # to prevent automatic conversion
  show_whats_imported(py_module("numpy", convert = FALSE), random)

  # import all objects that don't begin with a `_`
  # by default, other modules found in the module are also not imported
  show_whats_imported(py_module("glob"), "*")
```

```
  # to import EVERYTHING pass "**"
  # now includes modules that your modules imported, like `os`
  show_whats_imported(py_module("glob"), "**")

  rm(py_module) # clean up

## End(Not run)

# cleanup
setwd(owd)
unlink(tmpdir, recursive = TRUE)
rm(show_whats_imported, tmpdir, owd)
```

---

include *Source* R *files*

Description

Source R files

Usage

```
include(
  files_andor_dirs,
  envir = parent.frame(),
  chdir = FALSE,
  recursive = FALSE
)
```

Arguments

files_andor_dirs	A character vector of filepaths to R files, or directories containing R files. Directories are searched for files that end with extension ".R" or ".r", ignoring those that start with a period (.) or an underscore (_). The found files files from each directory are sorted by their basename() before being sourced. Filepaths can be supplied explicitly to override the default sorting.
envir	An R environment. By default, the current R evaluation environment.
chdir	logical; if TRUE, the R working directory is changed to the directory containing file for evaluating.
recursive	whether to search directories recursively for R files.

Details

This is a vectorized wrapper around base::sys.source with some differences. Notably:

- envir defaults to the current frame

- `envir` is returned (invisibly)
- `keep.source` and `keep.parse.data` default to `getOption("keep.source")` and `getOption("keep.parse.data")` respectively, instead of `getOption("keep.source.pkgs")` and `getOption("keep.parse.data.pkgs")`
- `toplevel.env` is set to `getOption("topLevelEnvironment", envir)`. In other words, if the option `topLevelEnvironment` is already set, it is respected.

Value

The environment `envir`, invisibly.

`set_library_default_pos` *Modify default attach position for* `base::library()`

Description

This function is documented but not exported. Reach in with `envir:::set_library_default_pos()` to use it.

Usage

```
set_library_default_pos(..., after = NULL, before = NULL, value = NULL)
```

Arguments

`...`	Ignored. Arguments must be named
`after, before`	string; the name of the environment on the search path that library() calls should by default attach after or before.
`value`	The value (or quoted expression) the new argument should be.

Details

This is primarily a way to "pin" a particular environment on the search path. For example, say you have a "project_utils" environment where you've defined a variety of useful functions. To prevent future `library()` calls from masking any objects in your attached "project_utils" environment, you can modify the default pos argument to library.

```
attach_source("project_utils.R", name = "project_utils)
set_library_default_pos(after = "project_utils")
library(foo) # now foo will attach after the "project_utils" environment
```

Value

The original default value of pos, invisibly

`within.environment` `within` *methods for* R *environments*

Description

`within` methods for R environments

Usage

```
## S3 method for class 'environment'
within(data, expr, ..., quote = substitute(expr))

## S3 method for class 'character'
within(
  data,
  expr,
  ...,
  pos = 2L,
  warn.conflicts = TRUE,
  mask.ok = NULL,
  quote = substitute(expr)
)
```

Arguments

`data`	An R environment, or the name of a (potentially new) attached environment.
`expr`	The bare R expression to evaluate. Automatically quoted.
`...`	Ignored. Added for compatibility with the S3 generic. Throws an error if any arguments are passed to `....`
`quote`	An R language object. This is an escape hatch from the automatic quoting of `expr`.
`pos`	The position where to attach the environment, if creating a new one. If an environment of `name` already exists, `pos` is ignored.
`warn.conflicts`	logical. If TRUE (the default), print warnings about objects in the attached environment that that are masking or masked by other objects of the same name.
`mask.ok`	character vector of names of objects that can mask objects on the search path without signaling a warning if `warn.conflicts` is `TRUE`.

Details

The only difference between `attach_eval` and `within.character` is the order of the arguments and the return value; the first returns the result of evaluating the expression, the latter the environment.

Value

The R environment, invisibly.

Note

See the note in `attach_source` about a potential pitfall of evaluating code directly in an attached environment.

See Also

attach_eval attach_source eval within

Index