Package 'attachment'

September 19, 2022

Title Deal with Dependencies

Version 0.3.0

Description Manage dependencies during package
development. This can retrieve all dependencies that are used in ``.R''
files in the ``R/'' directory, in ``.Rmd'' files in ``vignettes/'' directory and
in 'roxygen2' documentation of functions. There is a function to
update the ``DESCRIPTION'' file of your package with 'CRAN' packages or any other remote package.
All functions to retrieve dependencies of ``.R'' scripts and ``.Rmd'' or ``.qmd'' files can
be used independently of a package development.

License GPL-3

URL https://thinkr-open.github.io/attachment/,
https://github.com/ThinkR-open/attachment

BugReports https://github.com/ThinkR-open/attachment/issues

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.2.1

Config/testthat/edition 3

Depends R (>= 3.4)

Imports cli, desc (>= 1.2.0), glue (>= 1.3.0), knitr (>= 1.20),
magrittr (>= 1.5), rmarkdown (>= 1.10), roxygen2, stats,
stringr (>= 1.3.1), utils, withr

Suggests lifecycle, renv (>= 0.8.4), rstudioapi, testthat (>= 3.0.0)

NeedsCompilation no

Author Sébastien Rochette [cre, aut] (<https://orcid.org/0000-0002-1565-9313>),
Vincent Guyader [aut] (<https://orcid.org/0000-0003-0671-9270>,
previous maintainer),
ThinkR [cph, fnd]

Maintainer Sébastien Rochette <sebastien@thinkr.fr>

Repository CRAN

Date/Publication 2022-09-19 12:26:10 UTC

R topics documented:

`att_amend_desc` *Amend DESCRIPTION with dependencies read from package code parsing*

Description

Amend package DESCRIPTION file with the list of dependencies extracted from R, tests, vignettes files. att_to_desc_from_pkg() is an alias of att_amend_desc(), for the correspondence with `att_to_desc_from_is()`.

Usage

```
att_amend_desc(
  path = ".",
  path.n = "NAMESPACE",
  path.d = "DESCRIPTION",
  dir.r = "R",
  dir.v = "vignettes",
  dir.t = "tests",
  extra.suggests = NULL,
  pkg_ignore = NULL,
  document = TRUE,
  normalize = TRUE,
  inside_rmd = FALSE,
  must.exist = TRUE
)

att_to_desc_from_pkg(
  path = ".",
```

```
  path.n = "NAMESPACE",
  path.d = "DESCRIPTION",
  dir.r = "R",
  dir.v = "vignettes",
  dir.t = "tests",
  extra.suggests = NULL,
  pkg_ignore = NULL,
  document = TRUE,
  normalize = TRUE,
  inside_rmd = FALSE,
  must.exist = TRUE
)
```

Arguments

path	path to the root of the package directory. Default to current directory.
path.n	path to namespace file.
path.d	path to description file.
dir.r	path to directory with R scripts.
dir.v	path to vignettes directory. Set to empty (dir.v = "") to ignore.
dir.t	path to tests directory. Set to empty (dir.t = "") to ignore.
extra.suggests	vector of other packages that should be added in Suggests (pkgdown, covr for instance)
pkg_ignore	vector of packages names to ignore.
document	Run function roxygenise of roxygen2 package
normalize	Logical. Whether to normalize the DESCRIPTION file. See desc::desc_normalize()
inside_rmd	Logical. Whether function is run inside a Rmd, in case this must be executed in an external R session
must.exist	Logical. If TRUE then an error is given if packages do not exist within installed packages. If NA, a warning.

Value

Update DESCRIPTION file.

Examples

```
tmpdir <- tempdir()
file.copy(system.file("dummypackage",package = "attachment"), tmpdir,
 recursive = TRUE)
dummypackage <- file.path(tmpdir, "dummypackage")
# browseURL(dummypackage)
att_amend_desc(path = dummypackage)
```

att_from_description *Return all package dependencies from current package*

Description

Return all package dependencies from current package

Usage

```
att_from_description(
  path = "DESCRIPTION",
  dput = FALSE,
  field = c("Depends", "Imports", "Suggests")
)
```

Arguments

path	path to the DESCRIPTION file
dput	if FALSE return a vector instead of dput output
field	DESCRIPTION field to parse, Import, Suggests and Depends by default

Value

A character vector with packages names

Examples

```
dummypackage <- system.file("dummypackage", package = "attachment")
# browseURL(dummypackage)
att_from_description(path = file.path(dummypackage, "DESCRIPTION"))
```

att_from_namespace *return package dependencies from NAMESPACE file*

Description

return package dependencies from NAMESPACE file

Usage

```
att_from_namespace(path = "NAMESPACE", document = TRUE, clean = TRUE)
```

Arguments

path	path to NAMESPACE file
document	Run function roxygenise of roxygen2 package
clean	Logical. Whether to remove the original NAMESPACE before updating

Value

a vector

Examples

```
tmpdir <- tempdir()
file.copy(system.file("dummypackage",package = "attachment"), tmpdir,
 recursive = TRUE)
dummypackage <- file.path(tmpdir, "dummypackage")
# browseURL(dummypackage)
att_from_namespace(path = file.path(dummypackage, "NAMESPACE"))
```

---

att_from_rmd *Get all dependencies from a Rmd file*

Description

Get all dependencies from a Rmd file

Usage

```
att_from_rmd(
  path,
  temp_dir = tempdir(),
  warn = -1,
  encoding = getOption("encoding"),
  inside_rmd = FALSE,
  inline = TRUE
)

att_from_qmd(
  path,
  temp_dir = tempdir(),
  warn = -1,
  encoding = getOption("encoding"),
  inside_rmd = FALSE,
  inline = TRUE
)
```

Arguments

path	Path to a Rmd file
temp_dir	Path to temporary script from purl vignette
warn	-1 for quiet warnings with purl, 0 to see warnings
encoding	Encoding of the input file; always assumed to be UTF-8 (i.e., this argument is effectively ignored).
inside_rmd	Logical. Whether function is run inside a Rmd, in case this must be executed in an external R session
inline	Logical. Default TRUE. Whether to explore inline code for dependencies.

Value

vector of character of packages names found in the Rmd

Examples

```
dummypackage <- system.file("dummypackage",package = "attachment")
# browseURL(dummypackage)
att_from_rmd(path = file.path(dummypackage,"vignettes/demo.Rmd"))
```

---

att_from_rmds *Get all packages called in vignettes folder*

Description

Get all packages called in vignettes folder

Usage

```
att_from_rmds(
  path = "vignettes",
  pattern = "*.[.](Rmd|rmd|qmd)$",
  recursive = TRUE,
  warn = -1,
  inside_rmd = FALSE,
  inline = TRUE
)

att_from_qmds(
  path = "vignettes",
  pattern = "*.[.](Rmd|rmd|qmd)$",
  recursive = TRUE,
  warn = -1,
  inside_rmd = FALSE,
  inline = TRUE
)
```

Arguments

path	path to directory with Rmds or vector of Rmd files
pattern	pattern to detect Rmd files
recursive	logical. Should the listing recurse into directories?
warn	-1 for quiet warnings with purl, 0 to see warnings
inside_rmd	Logical. Whether function is run inside a Rmd, in case this must be executed in an external R session
inline	Logical. Default TRUE. Whether to explore inline code for dependencies.

Value

Character vector of packages called with library or require. knitr and rmarkdown are added by default to allow building the vignettes if the directory contains "vignettes" in the path

Examples

```
dummypackage <- system.file("dummypackage",package = "attachment")
# browseURL(dummypackage)
att_from_rmds(path = file.path(dummypackage,"vignettes"))
```

---

att_from_rscript *Look for functions called with :: and library/requires in one script*

Description

Look for functions called with :: and library/requires in one script

Usage

```
att_from_rscript(path)
```

Arguments

path	path to R script file

Details

Calls from pkg::fun in roxygen skeleton and comments are ignored

Value

a vector

Examples

```
dummypackage <- system.file("dummypackage",package = "attachment")
# browseURL(dummypackage)

att_from_rscript(path = file.path(dummypackage,"R","my_mean.R"))
```

att_from_rscripts *Look for functions called with :: and library/requires in folder of scripts*

Description

Look for functions called with :: and library/requires in folder of scripts

Usage

```
att_from_rscripts(path = "R", pattern = "*.[.](r|R)$", recursive = TRUE)
```

Arguments

path	directory with R scripts inside or vector of R scripts
pattern	pattern to detect R script files
recursive	logical. Should the listing recurse into directories?

Value

vector of character of packages names found in the R script

Examples

```
dummypackage <- system.file("dummypackage",package = "attachment")
# browseURL(dummypackage)

att_from_rscripts(path = file.path(dummypackage, "R"))
att_from_rscripts(path = list.files(file.path(dummypackage, "R"), full.names = TRUE))
```

att_to_desc_from_is *Amend DESCRIPTION with dependencies from imports and suggests package list*

Description

Amend DESCRIPTION with dependencies from imports and suggests package list

Usage

```
att_to_desc_from_is(
  path.d = "DESCRIPTION",
  imports = NULL,
  suggests = NULL,
  normalize = TRUE,
  must.exist = TRUE
)
```

Arguments

path.d	path to description file.
imports	character vector of package names to add in Imports section
suggests	character vector of package names to add in Suggests section
normalize	Logical. Whether to normalize the DESCRIPTION file. See desc::desc_normalize()
must.exist	Logical. If TRUE then an error is given if packages do not exist within installed packages. If NA, a warning.

Details

must.exist is better set to TRUE during package development. This stops the process when a package does not exists on your system. This avoids check errors with typos in package names in DESCRIPTION. When used in CI to discover dependencies, for a bookdown for instance, you may want to set to FALSE (no message at all) or NA (warning for not installed).

Value

Fill in Description file

Examples

```
tmpdir <- tempdir()
file.copy(system.file("dummypackage",package = "attachment"), tmpdir,
 recursive = TRUE)
dummypackage <- file.path(tmpdir, "dummypackage")
# browseURL(dummypackage)
att_to_desc_from_is(path.d = file.path(dummypackage, "DESCRIPTION"),
imports = c("magrittr", "attachment"), suggests = c("knitr"))
# In combination with other functions
att_to_desc_from_is(path.d = file.path(dummypackage, "DESCRIPTION"),
imports = att_from_rscripts(file.path(dummypackage, "R")),
suggests = att_from_rmds(file.path(dummypackage, "vignettes")))
```

---

create_dependencies_file

Create a dependencies.R in the inst *folder*

Description

Create a dependencies.R in the inst folder

Usage

```
create_dependencies_file(
  path = "DESCRIPTION",
  field = c("Depends", "Imports"),
  to = "inst/dependencies.R",
  open_file = TRUE,
  ignore_base = TRUE
)
```

Arguments

path	path to the DESCRIPTION file
field	DESCRIPTION field to parse, "Import" and "Depends" by default. Can add "Suggests"
to	path to dependencies.R. "inst/dependencies.R" by default
open_file	Logical. Open the file created in an editor
ignore_base	Logical. Whether to ignore package coming with base, as they cannot be installed

Value

Used for side effect. Shows a message with installation instructions and creates a R file containing these instructions.

Examples

```
tmpdir <- tempdir()
file.copy(system.file("dummypackage",package = "attachment"), tmpdir,
 recursive = TRUE)
dummypackage <- file.path(tmpdir, "dummypackage")
# browseURL(dummypackage)

create_dependencies_file(path = file.path(dummypackage,"DESCRIPTION"),
to = file.path(dummypackage, "inst/dependencies.R"),
open_file = FALSE)
```

---

create_renv_for_dev *Create reproducible environments for your R projects with renv*

Description

[Experimental]

Tool to create and maintain renv.lock files. The idea is to have 2 distinct files, one for development and the other for deployment. Indeed, although packages like attachment or pkgload must be installed to develop, they are not necessary in your project, package or Shiny application.

Usage

```
create_renv_for_dev(
  path = ".",
  dev_pkg = "_default",
  folder_to_include = c("dev", "data-raw"),
  output = "renv.lock",
  install_if_missing = TRUE,
  document = TRUE,
  pkg_ignore = NULL,
  ...
)

create_renv_for_prod(
  path = ".",
  output = "renv.lock.prod",
  dev_pkg = "remotes",
  ...
)
```

Arguments

path	Path to your current package source folder
dev_pkg	Vector of packages you need for development. Use `_default` (with underscore before to avoid confusing with a package name), to use the default list. Use `NULL` for no extra package. Use `attachment:::extra_dev_pkg` for the list.
folder_to_include	Folder to scan to detect development packages
output	Path and name of the file created, default is `./renv.lock`
install_if_missing	Logical. Install missing packages. `TRUE` by default
document	Logical. Whether to run `att_amend_desc()` before detecting packages in DESCRIPTION.
pkg_ignore	Vector of packages to ignore from being discovered in your files. This does not prevent them to be in "renv.lock" if they are recursive dependencies.
...	Other arguments to pass to `renv::snapshot()`

Value

a renv.lock file

Examples

```
## Not run:
create_renv_for_dev()
create_renv_for_dev(dev_pkg = "attachment")
create_renv_for_prod()

## End(Not run)
```

find_remotes *Proposes values for Remotes field for DESCRIPTION file based on your installation*

Description

Proposes values for Remotes field for DESCRIPTION file based on your installation

Usage

```
find_remotes(pkg)
```

Arguments

pkg	Character. Packages to test for potential non-CRAN installation

Value

List of all non-CRAN packages and code to add in Remotes field in DESCRIPTION. NULL otherwise.

Examples

```
# Find from vector of packages
find_remotes(pkg = c("attachment", "desc", "glue"))
# Find from Description file
dummypackage <- system.file("dummypackage", package = "attachment")
att_from_description(
path = file.path(dummypackage, "DESCRIPTION")) %>%
find_remotes()
## Not run:
# For the current package directory
att_from_description() %>% find_remotes()

## End(Not run)


# For a specific package name
find_remotes("attachment")

# Find remotes from all installed packages
find_remotes(list.dirs(.libPaths(), full.names = FALSE, recursive = FALSE))
```

install_from_description *Install missing package from DESCRIPTION*

Description

Install missing package from DESCRIPTION

Usage

```
install_from_description(
  path = "DESCRIPTION",
  field = c("Depends", "Imports", "Suggests"),
  ...
)
```

Arguments

path	path to the DESCRIPTION file
field	DESCRIPTION fields to parse, "Depends", "Imports", "Suggests" by default
...	Arguments to be passed to utils::install.packages()

Value

Used for side effect. Installs R packages from DESCRIPTION file if missing.

Examples

```
## Not run:
dummypackage <- system.file("dummypackage", package = "attachment")
# browseURL(dummypackage)

install_from_description(path = file.path(dummypackage,"DESCRIPTION"))

## End(Not run)
```

install_if_missing *install packages if missing*

Description

install packages if missing

Usage

```
install_if_missing(to_be_installed, ...)
```

Arguments

to_be_installed	a character vector containing required packages names
...	Arguments to be passed to utils::install.packages()

Value

Used for side effect. Install missing packages from the character vector input.

Examples

```
## Not run:
install_if_missing(c("dplyr","fcuk","rusk"))

## End(Not run)
```

---

set_remotes_to_desc *Add Remotes field to DESCRIPTION based on your local installation*

Description

Add Remotes field to DESCRIPTION based on your local installation

Usage

```
set_remotes_to_desc(path.d = "DESCRIPTION", stop.local = FALSE, clean = TRUE)
```

Arguments

path.d	path to description file.
stop.local	Logical. Whether to stop if package was installed from local source. Message otherwise.
clean	Logical. Whether to clean all existing remotes before run.

Value

Used for side effect. Adds Remotes field in DESCRIPTION file.

Examples

```
tmpdir <- tempdir()
file.copy(system.file("dummypackage", package = "attachment"), tmpdir,
 recursive = TRUE)
dummypackage <- file.path(tmpdir, "dummypackage")
# Add remotes field if there are Remotes locally
att_amend_desc(dummypackage) %>%
  set_remotes_to_desc()
```

```
## Not run:
# For your current package
att_amend_desc() %>%
  set_remotes_to_desc()

## End(Not run)
```

Index