

# Package: attachment (via r-universe)

July 2, 2024

**Title** Deal with Dependencies

**Version** 0.4.2

**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

**Config/fusen/version** 0.5.0.9006

**Config/Needs/website** ThinkR-open/thinkrtemplate

**Config/testthat/edition** 3

**Encoding** UTF-8

**RoxygenNote** 7.3.0

**Language** en-US

**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, yaml

**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), Murielle Delmotte [aut]  
 (<<https://orcid.org/0000-0002-1339-2424>>), Swann Floc'hlay  
 [aut] (<<https://orcid.org/0000-0003-1477-830X>>), ThinkR [cph,  
 fnd]

**Maintainer** Sébastien Rochette <sebastien@thinkr.fr>

**Repository** CRAN

**Date/Publication** 2024-07-01 18:30:02 UTC

## Contents

att_amend_desc . . . . .	2
att_from_description . . . . .	5
att_from_namespace . . . . .	5
att_from_rmd . . . . .	6
att_from_rmds . . . . .	7
att_from_rscript . . . . .	8
att_from_rscripts . . . . .	9
att_to_desc_from_is . . . . .	9
create_dependencies_file . . . . .	10
create_renv_for_dev . . . . .	12
find_remotes . . . . .	13
install_from_description . . . . .	14
install_if_missing . . . . .	15
set_remotes_to_desc . . . . .	16
<b>Index</b>	<b>17</b>

---

att_amend_desc	<i>Amend DESCRIPTION with dependencies read from package code parsing</i>
----------------	---

---

## 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,
    check_if_suggests_is_installed = TRUE,
    update.config = FALSE,
    use.config = TRUE,
    path.c = "dev/config_attachment.yaml"
  )

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,
  check_if_suggests_is_installed = TRUE,
  update.config = FALSE,
  use.config = TRUE,
  path.c = "dev/config_attachment.yaml"
)

```

## Arguments

<code>path</code>	path to the root of the package directory. Default to current directory.
<code>path.n</code>	path to namespace file.
<code>path.d</code>	path to description file.
<code>dir.r</code>	path to directory with R scripts.
<code>dir.v</code>	path to vignettes directory. Set to empty ( <code>dir.v = ""</code> ) to ignore.
<code>dir.t</code>	path to tests directory. Set to empty ( <code>dir.t = ""</code> ) to ignore.
<code>extra.suggests</code>	vector of other packages that should be added in Suggests (pkgdown, covr for instance)
<code>pkg_ignore</code>	vector of packages names to ignore.
<code>document</code>	Run function roxygenise of roxygen2 package
<code>normalize</code>	Logical. Whether to normalize the DESCRIPTION file. See <a href="#">desc::desc_normalize()</a>
<code>inside_rmd</code>	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.
check_if_suggests_is_installed	Logical. Whether to require that packages in the Suggests section are installed.
update.config	logical. Should the parameters used in this call be saved in the config file of the package
use.config	logical. Should the command use the parameters from the config file to run
path.c	character Path to the yaml config file where parameters are saved

### Details

Your daily use is to run `att_amend_desc()`, as is. You will want to run this function sometimes with some extra information like `att_amend_desc(pkg_ignore = "x", update.config = TRUE)` if you have to update the configuration file. Next time `att_amend_desc()` will use these parameters from the configuration file directly.

### Value

Update DESCRIPTION file.

### Examples

```
# Run on an external "dummyspackage" as an example
# For your local use, you do not have to specify the `path` as below
# By default, `att_amend_desc()` will run on the current working directory

# Create a fake package for the example
tmpdir <- tempfile(pattern = "description")
dir.create(tmpdir)
file.copy(system.file("dummyspackage", package = "attachment"), tmpdir,
  recursive = TRUE)
dummyspackage <- file.path(tmpdir, "dummyspackage")

# Update documentation and dependencies
att_amend_desc(path = dummyspackage)

# You can look at the content of this external package
#' # browseURL(dummyspackage)

# Update the config file with extra parameters
# We recommend that you store this code in a file in your "dev/" directory
# to run it when needed
att_amend_desc(path = dummyspackage, extra.suggests = "testthat", update.config = TRUE)

# Next time, in your daily development
att_amend_desc(path = dummyspackage)

# Clean after examples
unlink(tmpdir, recursive = TRUE)
```

---

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**

```
dummyspackage <- system.file("dummyspackage", package = "attachment")  
# browseURL(dummyspackage)  
att_from_description(path = file.path(dummyspackage, "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 <- tempfile(pattern = "namespace")
dir.create(tmpdir)
file.copy(system.file("dummyspackage", package = "attachment"), tmpdir,
  recursive = TRUE)
dummyspackage <- file.path(tmpdir, "dummyspackage")
# browseURL(dummyspackage)
att_from_namespace(path = file.path(dummyspackage, "NAMESPACE"))

# Clean temp files after this example
unlink(tmpdir, recursive = TRUE)
```

---

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**

```
dummyspackage <- system.file("dummyspackage", package = "attachment")
# browseURL(dummyspackage)
att_from_rmd(path = file.path(dummyspackage, "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**

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

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

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



---

att_from_rscripts	<i>Look for functions called with :: and library/requires in folder of scripts</i>
-------------------	--

---

**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**

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

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

---

att_to_desc_from_is	<i>Amend DESCRIPTION with dependencies from imports and suggests package list</i>
---------------------	---

---

**Description**

Amend DESCRIPTION with dependencies from imports and suggests package list

**Usage**

```
att_to_desc_from_is(
  path.d = "DESCRIPTION",
  imports = NULL,
  suggests = NULL,
  check_if_suggests_is_installed = TRUE,
  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
check_if_suggests_is_installed	Logical. Whether to require that packages in the Suggests section are installed.
normalize	Logical. Whether to normalize the DESCRIPTION file. See <a href="#">desc::desc_normalize()</a>
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 exist 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 <- tempfile(pattern = "descfromis")
dir.create(tmpdir)
file.copy(system.file("dummyspackage", package = "attachment"), tmpdir,
  recursive = TRUE)
dummyspackage <- file.path(tmpdir, "dummyspackage")
# browseURL(dummyspackage)
att_to_desc_from_is(path.d = file.path(dummyspackage, "DESCRIPTION"),
  imports = c("magrittr", "attachment"), suggests = c("knitr"))

# In combination with other functions
att_to_desc_from_is(path.d = file.path(dummyspackage, "DESCRIPTION"),
  imports = att_from_rscripts(file.path(dummyspackage, "R")),
  suggests = att_from_rmds(file.path(dummyspackage, "vignettes"))))

# Clean temp files after this example
unlink(tmpdir, recursive = TRUE)
```

---

create\_dependencies\_file

*Create the list of instructions to install dependencies from a DESCRIPTION file*

---

**Description**

Outputs the list of instructions and a "dependencies.R" file with instructions in the "inst/" directory

**Usage**

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

**Arguments**

path	path to the DESCRIPTION file
field	DESCRIPTION field to parse, "Import" and "Depends" by default. Can add "Suggests"
to	path where to save the dependencies file. Set to "inst/dependencies.R" by default. Set to NULL if you do not want the file, but only the instructions as a list of character.
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 (default TRUE)
install_only_if_missing	Logical Modify the installation instructions to check, beforehand, if the packages are missing . (default FALSE)

**Value**

List of R instructions to install all dependencies from a DESCRIPTION file. Side effect: creates a R file containing these instructions.

**Examples**

```
# Create a fake package  
tmpdir <- tempfile(pattern = "depsfile")  
dir.create(tmpdir)  
file.copy(system.file("dummyspackage", package = "attachment"), tmpdir,  
           recursive = TRUE)  
dummyspackage <- file.path(tmpdir, "dummyspackage")  
  
# Create the dependencies commands but no file  
create_dependencies_file(  
  path = file.path(dummyspackage, "DESCRIPTION"),  
  to = NULL,  
  open_file = FALSE)
```

```
# Create the dependencies files in the package
create_dependencies_file(
  path = file.path(dummyspackage, "DESCRIPTION"),
  to = file.path(dummyspackage, "inst/dependencies.R"),
  open_file = FALSE)
list.files(file.path(dummyspackage, "inst"))
# browseURL(dummyspackage)

# Clean temp files after this example
unlink(tmpdir, recursive = TRUE)
```

---

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,
  check_if_suggests_is_installed = TRUE,
  ...
)

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

## Arguments

path                    Path to your current package source folder

dev_pkg	Vector of packages you need for development. Use <code>_default</code> (with underscore before to avoid confusing with a package name), to use the default list. Use <code>NULL</code> for no extra package. Use <code>attachment:::extra_dev_pkg</code> for the list.
folder_to_include	Folder to scan to detect development packages
output	Path and name of the file created, default is <code>./renv.lock</code>
install_if_missing	Logical. Install missing packages. TRUE by default
document	Logical. Whether to run <code>att_amend_desc()</code> 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.
check_if_suggests_is_installed	Logical. Whether to require that packages in the Suggests section are installed.
...	Other arguments to pass to <code>renv::snapshot()</code>

**Value**

a `renv.lock` file

**Examples**

```
## Not run:
# Writes a renv.lock a file in the user directory
create_renv_for_dev()
create_renv_for_dev(dev_pkg = "attachment")
create_renv_for_prod()

## End(Not run)
```

---

find_remotes	<i>Proposes values for Remotes field for DESCRIPTION file based on your installation</i>
--------------	--

---

**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 <a href="#">utils::install.packages()</a>

**Value**

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

**Examples**

```
## Not run:  
# This will install packages on your system  
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:  
# This will install packages on your system  
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 <- tempfile(pattern = "setremotes")
dir.create(tmpdir)
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()

# Clean temp files after this example
unlink(tmpdir, recursive = TRUE)

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

## End(Not run)
```



# Index

`att_amend_desc`, 2  
`att_amend_desc()`, 13  
`att_from_description`, 5  
`att_from_namespace`, 5  
`att_from_qmd` (`att_from_rmd`), 6  
`att_from_qmds` (`att_from_rmds`), 7  
`att_from_rmd`, 6  
`att_from_rmds`, 7  
`att_from_rscript`, 8  
`att_from_rscripts`, 9  
`att_to_desc_from_is`, 9  
`att_to_desc_from_is()`, 2  
`att_to_desc_from_pkg` (`att_amend_desc`), 2

`create_dependencies_file`, 10  
`create_renv_for_dev`, 12  
`create_renv_for_prod`  
    (`create_renv_for_dev`), 12

`desc::desc_normalize()`, 3, 10

`find_remotes`, 13

`install_from_description`, 14  
`install_if_missing`, 15

`renv::snapshot()`, 13

`set_remotes_to_desc`, 16

`utils::install.packages()`, 14, 15