

# Package ‘packager’

September 28, 2020

**Title** Create, Build and Maintain Packages

**Version** 1.5.0

**Description** Helper functions for package creation, building and maintenance. Designed to work with a build system such as 'GNU make' or package 'fakemake' to help you to conditionally work through the stages of package development (such as spell checking, linting, testing, before building and checking a package).

**License** BSD\_2\_clause + file LICENSE

**URL** <https://gitlab.com/fvafrCU/packager>

**Depends** R (>= 3.3.0)

**Imports** callr, checkmate, codetools, crayon, cyclocomp, desc, fakemake, git2r, httr, methods, pkgbuild, pkgload, rcmdcheck, remotes, rprojroot, tools, usethis, utils, whisker, whoami, withr

**Suggests** cleanr, covr, devtools, digest, knitr, lintr, rmarkdown, roxygen2, RUnit, spelling, testthat

**VignetteBuilder** knitr

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.1.1

**NeedsCompilation** no

**Author** Andreas Dominik Cullmann [aut, cre]

**Maintainer** Andreas Dominik Cullmann <fvafrCU@mailbox.org>

**Repository** CRAN

**Date/Publication** 2020-09-28 10:10:02 UTC

## R topics documented:

packager-package . . . . .	2
check_archive . . . . .	2

check_codetags . . . . .	3
check_cyclomatic_complexity . . . . .	4
check_news . . . . .	5
check_usage . . . . .	6
create . . . . .	6
get_options . . . . .	7
get_package_makelist . . . . .	8
get_pkg_archive_path . . . . .	8
git_add_commit . . . . .	9
git_tag . . . . .	10
infect . . . . .	10
provide_cran_comments . . . . .	11
set_options . . . . .	12
submit . . . . .	13
use_build_ignore . . . . .	14
use_dev_version . . . . .	14
use_directory . . . . .	15
<b>Index</b>	<b>16</b>

---

packager-package	<i>Helps Me Create, Build and Maintain Packages</i>
------------------	---

---

**Description**

Helper functions for package creation, building and maintenance, heavily borrowing from **devtools** 1.13.3.

**Details**

You will find the details in  
vignette("An\_Introduction\_to\_packager", package = "packager").

---

check_archive	<i>Check a Package Archive</i>
---------------	--------------------------------

---

**Description**

This is a wrapper to `callr::rcmd_safe("check")`, similar to, but leaner than `rcmdcheck::rcmdcheck`. While the latter parses the output of `rcmd_safe` and uses **clisymbols** in the callback, we here just return bare output and use `writeLines` as callback. This should result in a screen display that is identical to the output of R CMD check.

**Usage**

```
check_archive(path, cmdargs = NULL)

check_archive_as_cran(path)
```

**Arguments**

path                    Path to the package archive.  
 cmdargs                Command line arguments (see `callr::rcmd`).

**Value**

A list with standard output, standard error and exit status of the check. (see `callr::rcmd`).

**Note**

`check_archive_as_cran` is a convenience Wrapper to `check_archive`.

**See Also**

Other maintenance functions: `check_codetags()`, `check_cyclomatic_complexity()`, `check_news()`, `check_usage()`, `get_check_status()`

Other maintenance functions: `check_codetags()`, `check_cyclomatic_complexity()`, `check_news()`, `check_usage()`, `get_check_status()`

**Examples**

```
## Not run:
package_path <- file.path(tempdir(), "fakepack")
usethis::create_package(path = package_path)
file.copy(system.file("templates", "throw.R", package = "fakemake"),
           file.path(package_path, "R"))
roxygen2::roxygenize(package_path)
print(tarball <- get_pkg_archive_path(package_path))
pkgbuild::build(pkg = package_path, path = package_path)
print(check_archive(tarball))

## End(Not run)
```

---

check\_codetags

*Check for Code Tags*

---

**Description**

You do use code tags (see [PEP 350](#) for example)? This function searches for files under a directory containing such tags.

**Usage**

```
check_codetags(
  path = ".",
  exclude_pattern = "\\Rcheck/",
  include_pattern = "\\.[Rr]$|\\.[Rr]md$",
  pattern = "XXX:|FIXME:|TODO:"
)
```

**Arguments**

path	to a directory, typically a package root.
exclude_pattern	A pattern for exclusions based on the file names. Stronger than include_pattern.
include_pattern	A pattern for inclusions based on the file names.
pattern	The pattern to search for.

**Value**

A character vector of hits.

**See Also**

Other maintenance functions: [check\\_archive\(\)](#), [check\\_cyclomatic\\_complexity\(\)](#), [check\\_news\(\)](#), [check\\_usage\(\)](#), [get\\_check\\_status\(\)](#)

**Examples**

```
dir <- system.file("templates", package = "packager")
check_codetags(dir)
```

---

```
check_cyclomatic_complexity
      Check Cyclomatic Complexity
```

---

**Description**

Run [cyclocomp\\_package\\_dir](#) on the package throwing an error when the maximum complexity is exceeded.

**Usage**

```
check_cyclomatic_complexity(path = ".", max_complexity = 10)
```

**Arguments**

path	Path to the package directory (see <a href="#">devtools::as.package</a> ).
max_complexity	The maximum cyclomatic complexity (which must not be exceeded).

**Value**

[Invisibly TRUE](#) if maximum cyclomatic complexity is not exceeded, throws an error otherwise.

**See Also**

Other maintenance functions: [check\\_archive\(\)](#), [check\\_codetags\(\)](#), [check\\_news\(\)](#), [check\\_usage\(\)](#), [get\\_check\\_status\(\)](#)

## Examples

```
## Not run:
# download and untar sources of some archived package
package <- "excerptr"
root <- paste0("http://cran.r-project.org/src/contrib/Archive/", package)
version <- "1.0.0"
tarball <- paste0(paste(package, version, sep = "_"), ".tar.gz")
remote_tarball <- paste(root, tarball, sep = "/")
local_tarball <- file.path(tempdir(), tarball)
utils::download.file(remote_tarball, local_tarball)
utils::untar(local_tarball, exdir = tempdir())
res <- tryCatch(check_cyclomatic_complexity(path = file.path(tempdir(),
                                                             package)),
                error = identity)

print(res)

## End(Not run)
```

---

check\_news

*Check for 'NEWS.md' Being Up to Date*

---

## Description

Compare your 'NEWS.md' file to the 'Version' entry in DESCRIPTION.

## Usage

```
check_news(path = ".")
```

## Arguments

path                      Path to the package directory (see [devtools::as.package](#)).

## Value

Invisibly **TRUE** if 'NEWS.md' matches DESCRIPTION, throws an error otherwise.

## See Also

Other maintenance functions: [check\\_archive\(\)](#), [check\\_codetags\(\)](#), [check\\_cyclomatic\\_complexity\(\)](#), [check\\_usage\(\)](#), [get\\_check\\_status\(\)](#)

---

check_usage	Check Usage with <b>codetools</b> ' <a href="#">checkUsagePackage</a>
-------------	---

---

### Description

This is just a convenience wrapper to [checkUsagePackage](#) (which needs loading of the [development version of the] package).

### Usage

```
check_usage(path = ".")
```

### Arguments

path	Path to the package directory (see <a href="#">devtools::as.package</a> ).
------	--

### Value

A character vector of issues found by [checkUsagePackage](#).

### See Also

Other maintenance functions: [check\\_archive\(\)](#), [check\\_codetags\(\)](#), [check\\_cyclomatic\\_complexity\(\)](#), [check\\_news\(\)](#), [get\\_check\\_status\(\)](#)

---

create	Create a Package Template
--------	---------------------------

---

### Description

This is just a wrapper to create a package using [usethis::create\\_package](#) and infect it using [infect](#).

### Usage

```
create(path, force = TRUE, ...)
```

### Arguments

path	See <a href="#">usethis::create_package</a> .
force	Recursively <a href="#">unlink</a> path before calling <a href="#">usethis::create_package(path)</a> ?
...	Arguments to be passed to <a href="#">infect</a> .

### Value

[Invisibly NULL](#).

**See Also**[infect](#)**Examples**

```

path <- file.path(tempdir(), "myFirstPackage")
packager::create(path = path, fakemake = "roxygen2")
list.files(path, recursive = TRUE)
## Not run:
if (require("roxygen2")) {
  ml <- packager::get_package_makelist(is_cran = TRUE)
  d <- file.path(tempdir(), "somePackage")
  dir.create(d)
  packager::create(d, fakemake = FALSE, fakemake = FALSE)
  withr::with_dir(d, fakemake::make("check", ml))
  check_log <- file.path(d, "log", "check.Rout")
  status <- packager::get_check_status(check_log)
  RUnit::checkEqualsNumeric(status[["status"]][["errors"]], 0)
  list.files(d, recursive = TRUE)
  unlink(d, recursive = TRUE)
}

## End(Not run)

```

get\_options

*Get Options For Packages***Description**

A convenience function for [getOption](#).

**Usage**

```
get_options(package_name, ..., remove_names = FALSE, flatten_list = TRUE)
```

**Arguments**

package_name	The package's name.
...	See <a href="#">getOption</a> .
remove_names	[boolean(1)] Remove the names?
flatten_list	[boolean(1)] Return a vector?

**Value**

A (possibly named) list or a vector.

**See Also**

Other option functions: [set\\_options\(\)](#)

---

get\_package\_makelist    *Provide a makelist Suitable for Packages with **packager***

---

**Description**

Provide a makelist Suitable for Packages with **packager**

**Usage**

```
get_package_makelist(is_cran = FALSE, gitlab_token = NULL)
```

**Arguments**

`is_cran`                Streamline makelist for usage on CRAN?  
`gitlab_token`        A private gitlab token. Used to query logs on <https://about.gitlab.com>.

**Value**

A list for [fakemake::make](#).

**Examples**

```
ml <- packager::get_package_makelist()
cbind(lapply(ml, function(x) x[["target"]]),
      lapply(ml, function(x) x[["alias"]]))

cl <- packager::get_package_makelist(is_cran = TRUE)
setdiff(sapply(ml, function(x) x[["target"]]),
        sapply(cl, function(x) x[["target"]]))
```

---

get\_pkg\_archive\_path    *Create a Package's Archive Path From the Package's 'DESCRIPTION'*

---

**Description**

The archive file does not have to exist. Use `file.exists(get_pkg_archive_path())` to test existence.

**Usage**

```
get_pkg_archive_path(path = ".", absolute = TRUE)
```



**Arguments**

path	Path to the package directory (see <a href="#">devtools::as.package</a> ).
absolute	Return the absolute path?

**Value**

Path to the package's archive file.

**Examples**

```
package_path <- file.path(tempdir(), "anRpackage")
usethis::create_package(path = package_path)
print(tarball <- get_pkg_archive_path(package_path))
file.exists(tarball)
```

---

git_add_commit	<i>Git Add All Changes and Commit</i>
----------------	---------------------------------------

---

**Description**

Much like `git commit -am"M"`, where M is the message.

**Usage**

```
git_add_commit(
  path,
  message = "Uncommented Changes: Backing Up",
  untracked = FALSE,
  ...
)
```

**Arguments**

path	The path to the repository.
message	The commit message to use.
untracked	Add files not tracked yet before committing?
...	Arguments passed to <a href="#">git2r::status</a> .

**Value**

The return value of [git2r::commit](#).

**See Also**

Other git wrappers: [git\\_tag\(\)](#)

---

<code>git_tag</code>	<i>Create a Git Tag Based on the Current Version Number</i>
----------------------	---

---

### Description

This is basically the same as `git tag -a T -m M` where T is the version read from the package's DESCRIPTION file and M is given by message (see below).

### Usage

```
git_tag(path = ".", tag_uncommitted = FALSE, message = "CRAN release")
```

### Arguments

<code>path</code>	Path to the package.
<code>tag_uncommitted</code>	Tag if there are uncommitted changes?
<code>message</code>	The tag message to be used.

### Value

`FALSE` or the value of `git2r::tag`.

### See Also

Other git wrappers: `git_add_commit()`

---

<code>infect</code>	<i>Adjust a Package</i>
---------------------	-------------------------

---

### Description

Add a variety of extensions to a package (skeleton) and run `fakemake::make` on it.

### Usage

```
infect(path, fakemake = "check", git_add_and_commit = TRUE, ...)
```

### Arguments

<code>path</code>	Path to the package directory (see <code>devtools::as.package</code> ).
<code>fakemake</code>	The name for a <code>makelist</code> for <code>fakemake</code> . Set to <code>NULL</code> or <code>FALSE</code> to disable running <code>fakemake::make</code> .
<code>git_add_and_commit</code>	Add and commit changes in git?
<code>...</code>	Arguments to be passed to <code>set_package_info</code> .

**Value**

Invisibly NULL.

**See Also**

[create](#)

**Examples**

```
## Not run:
if (require("roxygen2")) {
  path <- file.path(tempdir(), "mySecondPackage")
  usethis::create_package(path = path, open = FALSE)
  l1 <- list.files(path, recursive = TRUE)
  packager::infect(path = path, fakemake = "roxygen2", fakemake = FALSE)
  l2 <- list.files(path, recursive = TRUE)
  print(l1); print(l2)
  unlink(path, recursive = TRUE)
}

## End(Not run)
```

---

provide\_cran\_comments *Provide a Template for Your Comments To CRAN*

---

**Description**

[submit](#) reads a file ‘cran-comments.md’. This function provides a template based on your R version, your R CMD check output and the package’s ‘NEWS.md’.

**Usage**

```
provide_cran_comments(
  check_log = NULL,
  path = ".",
  initial = FALSE,
  write_to_file = TRUE,
  private_token = NULL,
  name = NA,
  proxy = NULL
)
```

**Arguments**

check_log	Path to the check log relative to path. Typically file.path("log", "check.Rout").
path	Path to the package directory (see <a href="#">devtools::as.package</a> ).
initial	Is this an initial submission?

write_to_file	Do write the comment to 'cran-comment.md'?
private_token	Provide a private token to access <a href="https://about.gitlab.com">https://about.gitlab.com</a> .
name	The name to sign with, if NA, the given name of the package maintainer as stated in file DESCRIPTION is used.
proxy	A proxy to use.

**Value**

Character vector containing the CRAN comments, which are written to 'cran-comments.md' (see Note).

**Note**

By default this function writes to disk as side effect.

**Examples**

```
## Not run:

if (Sys.info()[["nodename"]] == "fvafrdebianCU") {
  gitlab_token <- readLines(file.path("~", ".gitlab_private_token.txt"))
  proxy <- httr::use_proxy("10.127.255.17", 8080)
  comments <- provide_cran_comments(path = ".",
                                    write_to_file = TRUE,
                                    private_token = gitlab_token,
                                    proxy = proxy)
} else {
  gitlab_token <- readLines(file.path("~", ".gitlab_private_token.txt"))
  comments <- provide_cran_comments(path = ".",
                                    write_to_file = TRUE,
                                    private_token = gitlab_token)
}
cat(comments, sep = "")

## End(Not run)
```

---

set\_options

*Set Options For Packages*


---

**Description**

A convenience function for [options](#).

**Usage**

```
set_options(package_name, ..., overwrite = TRUE)
```

**Arguments**

package_name	The package's name.
...	See <a href="#">options</a> .
overwrite	[boolean(1)] Overwrite options already set?

**Value**

invisible(TRUE)

**See Also**

Other option functions: [get\\_options\(\)](#)

**Examples**

```
options("cleanr" = NULL)
defaults <- list(max_file_width = 80, max_file_length = 300,
                 max_lines = 65, max_lines_of_code = 50,
                 max_num_arguments = 5, max_nesting_depth = 3,
                 max_line_width = 80, check_return = TRUE)

set_options("cleanr", defaults)
getOption("cleanr")
set_options("cleanr", list(max_line_width = 3,
                           max_lines = "This is nonsense!"))
set_options("cleanr", check_return = NULL, max_lines = 4000)
get_options("cleanr")
```

---

submit

*Release a Package to CRAN*


---

**Description**

This is a stripped version of **devtools'** [release function](#), omitting most of the interactive checks.

**Usage**

```
submit(path = ".", stop_on_git = TRUE, force = FALSE, verbose = TRUE)
```

```
release(path = ".", stop_on_git = TRUE, force = FALSE, verbose = TRUE)
```

**Arguments**

path	The package's root directory.
stop_on_git	Stop if git has uncommitted changes or is not synced with the origin?
force	Skip user interaction?
verbose	Be verbose?

**Value**

Invisibly NULL

**Note**

release is just a link to submit as [release](#) is the original function from **devtools**.

---

use_build_ignore	<i>Add files to '.Rbuildignore'</i>
------------------	-------------------------------------

---

**Description**

This is verbatim copy of git commit a5e5805ecd630ebc46e080bd78ebcf32322efe3c of **usethis**.  
 '.Rbuildignore' has a regular expression on each line, but it's usually easier to work with specific file names. By default, will (crudely) turn filenames into regular expressions that will only match these paths. Repeated entries will be silently removed.

**Usage**

```
use_build_ignore(files, escape = TRUE, pkg = ".")
```

**Arguments**

files	Paths of files.
escape	If TRUE, the default, will escape . to \. and surround with ^ and \$.
pkg	Path to the package directory (see <a href="#">as.package</a> ).

**Value**

Nothing, called for its side effect.

---

use_dev_version	<i>Use a Development Version in DESCRIPTION and 'NEWS.md'</i>
-----------------	---

---

**Description**

This is much like [usethis::use\\_dev\\_version](#), but their conventions keep changing.

**Usage**

```
use_dev_version(path = ".", force = FALSE)
```

**Arguments**

path	Path to the package directory (see <a href="#">devtools::as.package</a> ).
force	Set to TRUE to force version bumping with uncommitted git changes.

---

use_directory	<i>Use a Directory</i>
---------------	------------------------

---

**Description**

Create a directory.  
this is verbatim copy of git commit a5e5805ecd630ebc46e080bd78ebcf32322efe3c of **usethis**.

**Usage**

```
use_directory(path, ignore = FALSE, pkg = ".")
```

**Arguments**

path	Path of the directory to create, relative to the project.
ignore	Add the directory to ‘.Rbuildignore’?
pkg	Path to the package directory (see <a href="#">as.package</a> ).

# Index

- \* **git wrappers**
  - git\_add\_commit, 9
  - git\_tag, 10
- \* **maintenance functions**
  - check\_archive, 2
  - check\_codetags, 3
  - check\_cyclomatic\_complexity, 4
  - check\_news, 5
  - check\_usage, 6
- \* **option functions**
  - get\_options, 7
  - set\_options, 12
- \* **package**
  - packager-package, 2

as.package, 14, 15

callr::rcmd, 3

callr::rcmd\_safe, 2

check\_archive, 2, 4–6

check\_archive\_as\_cran(check\_archive), 2

check\_codetags, 3, 3, 4–6

check\_cyclomatic\_complexity, 3, 4, 4, 5, 6

check\_news, 3, 4, 5, 6

check\_usage, 3–5, 6

checkUsagePackage, 6

create, 6, 11

cyclocomp\_package\_dir, 4

devtools::as.package, 4–6, 9–11, 14

fakemake::make, 8, 10

FALSE, 10

get\_check\_status, 3–6

get\_options, 7, 13

get\_package\_makelist, 8

get\_pkg\_archive\_path, 8

getOption, 7

git2r::commit, 9

git2r::status, 9

git2r::tag, 10

git\_add\_commit, 9, 10

git\_tag, 9, 10

infect, 6, 7, 10

Invisibly, 4–6, 11, 14

makelist, 10

NULL, 6, 10, 11, 14

options, 12, 13

packager-package, 2

provide\_cran\_comments, 11

rcmdcheck::rcmdcheck, 2

release, 14

release(submit), 13

release function, 13

set\_options, 8, 12

set\_package\_info, 10

submit, 11, 13

TRUE, 4, 5

unlink, 6

use\_build\_ignore, 14

use\_dev\_version, 14

use\_directory, 15

usethis::create\_package, 6

usethis::use\_dev\_version, 14

writelnLines, 2