Package 'rcompendium'

March 24, 2021

Type Package

Title Create a Package or Research Compendium Structure

Version 0.5.1

Description Makes easier the creation of R package or research compendium (i.e. a predefined files/folders structure) so that users can focus on the code/analysis instead of wasting time organizing files. A full ready-to-work structure is set up with some additional features: version control, remote repository creation, CI/CD configuration (check package integrity under several OS, test code with 'testthat', and build and deploy website using 'pkgdown'). This package heavily relies on the R packages 'devtools' and 'usethis' and follows recommendations made by Wickham H. (2015) <ISBN:9781491910597> and Marwick B. et al. (2018) <doi:10.7287/peerj.preprints.3192v2>.

URL https://github.com/FRBCesab/rcompendium

BugReports https://github.com/FRBCesab/rcompendium/issues

License GPL (>= 2)

Encoding UTF-8

LazyData true

Depends R (>= 2.10)

Imports clisymbols, crayon, devtools, gert, gh, gtools, rmarkdown, rstudioapi, stringr, usethis, utils, xfun

RoxygenNote 7.1.1

VignetteBuilder knitr

Suggests fs, knitr, remotes, testthat (>= 3.0.0), withr

Config/testthat/edition 3

NeedsCompilation no

Author Nicolas Casajus [aut, cre, cph] (<https://orcid.org/0000-0002-5537-5294>)

Maintainer Nicolas Casajus <nicolas.casajus@fondationbiodiversite.fr>

Repository CRAN

Date/Publication 2021-03-24 10:20:02 UTC

R topics documented:

`add_citation` *Create a CITATION file*

Description

This function creates a `CITATION` file in the folder inst/. This file contains a BiBTeX entry to cite the package as a manual. User will need to edit by hand some information (title, version, etc.).

Usage

```
add_citation(
  given = NULL,
  family = NULL,
  organisation = NULL,
  open = TRUE,
  overwrite = FALSE,
  quiet = FALSE
)
```

Arguments

given	a character of length 1 The given name of the package maintainer.
family	a character of length 1 The family name of the package maintainer.
organisation	a character of length 1 The name of the GitHub organisation to host the package. If NULL (default) the GitHub account will be used. This argument is used to set the URL of the package (hosted on GitHub).
open	a logical value If TRUE (default) the file is opened in the editor.
overwrite	a logical value If this file is already present and overwrite = TRUE, it will be erased and replaced. Default is FALSE.
quiet	a logical value If TRUE messages are deleted. Default is FALSE.

Value

None

See Also

Other create files: add_compendium(), add_description(), add_license(), add_makefile(), add_package_doc(), add_readme_rmd(), add_testthat(), add_vignette()

Examples

```
## Not run:
add_citation()
readCitationFile("inst/CITATION")
citation("pkg")    # If you have installed your package <pkg>

## End(Not run)
```

add_codecov_badge *Add a Codecov badge*

Description

This function adds a Codecov badge to the `README.Rmd`, i.e. the percentage of code cover by units tests. This percentage is computed by codecov.io service.

Usage

```
add_codecov_badge(organisation = NULL, quiet = FALSE)
```

Arguments

`organisation`	a character of length 1 The name of the GitHub organisation to host the package. If `NULL` (default) the GitHub account will be used.
`quiet`	a logical value If `TRUE` messages are deleted. Default is `FALSE`.

Value

A Markdown badge expression

See Also

Other adding badges: `add_cran_badge()`, `add_dependencies_badge()`, `add_github_actions_check_badge()`, `add_github_actions_codecov_badge()`, `add_github_actions_pkgdown_badge()`, `add_license_badge()`, `add_lifecycle_badge()`, `add_repostatus_badge()`

Examples

```
## Not run:
add_codecov_badge()

## End(Not run)
```

add_compendium *Create compendium folders*

Description

This function creates the following additional folders data/, rscripts/, outputs/, figures/, and paper/. Each folder has a README to provide instructions. The argument compendium allows user to choose if these folders are created at the root of the project (default) or nested inside a parent directory. All theses folders area added to `.Rbuildignore`.

Usage

```
add_compendium(compendium = ".")
```

Arguments

`compendium`	a character of length 1 By default, compendium folders are created at the root of the project. User can change their location with this argument. For instance, if `compendium = 'analysis'`, compendium folders will be created inside the directory analysis/.

Value

None

See Also

Other create files: `add_citation()`, `add_description()`, `add_license()`, `add_makefile()`, `add_package_doc()`, `add_readme_rmd()`, `add_testthat()`, `add_vignette()`

Examples

```
## Not run:
add_compendium()

## End(Not run)
```

add_cran_badge *Add a CRAN status badge*

Description

This function adds a CRAN status badge to the `README.Rmd`. If the package is not hosted on the CRAN the badge will indicate *not published on the CRAN*.

Usage

```
add_cran_badge(quiet = FALSE)
```

Arguments

quiet	a logical value If TRUE messages are deleted. Default is FALSE.

Value

A Markdown badge expression

See Also

Other adding badges: add_codecov_badge(), add_dependencies_badge(), add_github_actions_check_badge(), add_github_actions_codecov_badge(), add_github_actions_pkgdown_badge(), add_license_badge(), add_lifecycle_badge(), add_repostatus_badge()

Examples

```
## Not run:
add_cran_badge()

## End(Not run)
```

---

add_dependencies *Add dependencies in DESCRIPTION*

Description

This function detects external dependencies used in R/, NAMESPACE, and @examples sections of roxygen2 headers and automatically adds these dependencies in the Imports section of the DESCRIPTION file.

In the NAMESPACE this function detects dependencies mentioned as import(pkg) and importFrom(pkg,fun).

In the R/ folder it detects functions called as pkg::fun() in the code of each R files. In @examples sections it also detects packages attached by library() or require().

The vignettes/ folder is also inspected and detected dependencies (pkg::fun(), library() or require()) are added to the Suggests field of the DESCRIPTION file (in addition to the packages knitr and rmarkdown).

If the project is a research compendium user can also inspect additional folder(s) with the argument compendium to add dependencies to the Imports section of the DESCRIPTION file. The detection process is the same as the one used for vignettes/.

The tests/ folder is also inspected and detected dependencies (pkg::fun(), library() or require()) are added to the Suggests field of the DESCRIPTION file (in addition to the package testthat).

Usage

```
add_dependencies(compendium = NULL)
```

Arguments

compendium	a character of length 1 The name of the folder to recursively detect dependencies to be added to the Imports field of DESCRIPTION file. It can be 'analysis/' (if additional folders, i.e. data/, outputs/, figures/, etc. have been created in this folder), '.' (if folders data/, outputs/, figures/, etc. have been created at the root of the project), etc. See new_compendium() for further information. Default is compendium = NULL (i.e. no additional folder are inspected but R/, NAMESPACE, vignettes/, and tests/ are still inspected).

Value

None

See Also

Other development functions: add_github_actions_check(), add_github_actions_codecov(), add_github_actions_pkgdown(), add_r_depend(), add_to_buildignore(), add_to_gitignore()

Examples

```
## Not run:
add_dependencies()

## End(Not run)
```

---

add_dependencies_badge

Add a Dependencies badge

Description

This function adds a dependencies badge to the README.Rmd. The first number corresponds to the direct dependencies and the second to the recursive dependencies.

Note/ this function can work with packages not published on the CRAN and is based on the function gtools::getDependencies(). See also the function get_all_dependencies().

Usage

```
add_dependencies_badge(quiet = FALSE)
```

Arguments

quiet	a logical value If TRUE messages are deleted. Default is FALSE.

Value

A Markdown badge expression

See Also

Other adding badges: add_codecov_badge(), add_cran_badge(), add_github_actions_check_badge(), add_github_actions_codecov_badge(), add_github_actions_pkgdown_badge(), add_license_badge(), add_lifecycle_badge(), add_repostatus_badge()

Examples

```
## Not run:
add_dependencies_badge()

## End(Not run)
```

---

add_description *Create a DESCRIPTION file*

Description

This function creates a DESCRIPTION file at the root of the project based on a template (adapted from usethis). User credentials can be passed as arguments but it is recommended to store them in the .Rprofile file with set_credentials().

Usage

```
add_description(
  given = NULL,
  family = NULL,
  email = NULL,
  orcid = NULL,
  organisation = NULL,
  open = TRUE,
  overwrite = FALSE,
  quiet = FALSE
)
```

Arguments

given	a character of length 1 The given name of the package maintainer.
family	a character of length 1 The family name of the package maintainer.
email	a character of length 1 The email address of the package maintainer.
orcid	a character of length 1 The ORCID of the package maintainer.
organisation	a character of length 1 The name of the GitHub organisation to host the package. If NULL (default) the GitHub account will be used.
open	a logical value If TRUE (default) the file is opened in the editor.
overwrite	a logical value If a DESCRIPTION is already present and overwrite = TRUE, this file will be erased and replaced. Default is FALSE.
quiet	a logical value If TRUE messages are deleted. Default is FALSE.

Value

None

See Also

Other create files: add_citation(), add_compendium(), add_license(), add_makefile(), add_package_doc(), add_readme_rmd(), add_testthat(), add_vignette()

Examples

```
## Not run:
add_description(organisation = "MySociety")

## End(Not run)
```

add_github_actions_check *Setup GitHub Actions to check package*

Description

This function creates a configuration file (`.yaml`) to setup GitHub Actions to check the package. This file will be written (from a template adapted from `usethis::use_github_action()`) in the folder `.github/workflows/R-CMD-check.yaml`.

Usage

```
add_github_actions_check(open = FALSE, overwrite = FALSE, quiet = FALSE)
```

Arguments

`open`	a logical value If `TRUE` (default) the file is opened in the editor.
`overwrite`	a logical value If this file is already present and `overwrite = TRUE`, it will be erased and replaced. Default is `FALSE`.
`quiet`	a logical value If `TRUE` messages are deleted. Default is `FALSE`.

Details

This workflow runs R CMD check on the three major operating systems (Ubuntu, macOS, and Windows) on the latest release of R. If user want to change this setting he can replace this `.yaml` file be another (e.g. generated by running `usethis::use_github_action()`).

Value

None

See Also

Other development functions: `add_dependencies()`, `add_github_actions_codecov()`, `add_github_actions_pkgdown(`
`add_r_depend()`, `add_to_buildignore()`, `add_to_gitignore()`

Examples

```
## Not run:
add_github_actions_check()

## End(Not run)
```

add_github_actions_check_badge *Add a R CMD Check badge*

Description

This function adds a R CMD Check badge to the `README.Rmd`. This function must be run after `add_github_actions_check()` which will setup GitHub Actions to check and test the package.

Usage

```
add_github_actions_check_badge(organisation = NULL, quiet = FALSE)
```

Arguments

`organisation`	a character of length 1 The name of the GitHub organisation to host the package. If `NULL` (default) the GitHub account will be used.
`quiet`	a logical value If `TRUE` messages are deleted. Default is `FALSE`.

Value

A Markdown badge expression

See Also

Other adding badges: `add_codecov_badge()`, `add_cran_badge()`, `add_dependencies_badge()`, `add_github_actions_codecov_badge()`, `add_github_actions_pkgdown_badge()`, `add_license_badge()`, `add_lifecycle_badge()`, `add_repostatus_badge()`

Examples

```
## Not run:
add_github_actions_check_badge()

## End(Not run)
```

add_github_actions_codecov *Setup GitHub Actions to report code coverage*

Description

This function creates a configuration file (`.yaml`) to setup GitHub Actions to report code coverage when testing the package. This file will be added (from a template adapted from `usethis::use_github_action()`) in the folder `.github/workflows/test-coverage.yaml`.

Usage

```
add_github_actions_codecov(open = FALSE, overwrite = FALSE, quiet = FALSE)
```

Arguments

`open`	a logical value If `TRUE` (default) the file is opened in the editor.
`overwrite`	a logical value If this file is already present and `overwrite = TRUE`, it will be erased and replaced. Default is `FALSE`.
`quiet`	a logical value If `TRUE` messages are deleted. Default is `FALSE`.

Value

None

See Also

Other development functions: `add_dependencies()`, `add_github_actions_check()`, `add_github_actions_pkgdown()`, `add_r_depend()`, `add_to_buildignore()`, `add_to_gitignore()`

Examples

```
## Not run:
add_github_actions_codecov()

## End(Not run)
```

add_github_actions_codecov_badge *Add a Test coverage badge*

Description

This function adds a Test coverage badge to the `README.Rmd`. This function must be run after `add_github_actions_codecov()` which will setup GitHub Actions to report the percentage of code cover by units tests.

Usage

```
add_github_actions_codecov_badge(organisation = NULL, quiet = FALSE)
```

Arguments

`organisation`	a character of length 1 The name of the GitHub organisation to host the package. If `NULL` (default) the GitHub account will be used.
`quiet`	a logical value If `TRUE` messages are deleted. Default is `FALSE`.

Value

A Markdown badge expression

See Also

Other adding badges: `add_codecov_badge()`, `add_cran_badge()`, `add_dependencies_badge()`, `add_github_actions_check_badge()`, `add_github_actions_pkgdown_badge()`, `add_license_badge()`, `add_lifecycle_badge()`, `add_repostatus_badge()`

Examples

```
## Not run:
add_github_actions_codecov_badge()

## End(Not run)
```

add_github_actions_pkgdown *Setup GitHub Actions to build and deploy package website*

Description

This function creates a configuration file (`.yaml`) to setup GitHub Actions to automatically build and deploy the website using pkgdown. This file will be written (from a template adapted from `usethis::use_github_action()`) in the folder `.github/workflows/pkgdown.yaml`. An additional empty file (_pkgdown.yaml) will also be written: it can be used to customize the website.

Usage

```
add_github_actions_pkgdown(open = FALSE, overwrite = FALSE, quiet = FALSE)
```

Arguments

`open`	a logical value If `TRUE` (default) the file is opened in the editor.
`overwrite`	a logical value If this file is already present and `overwrite = TRUE`, it will be erased and replaced. Default is `FALSE`.
`quiet`	a logical value If `TRUE` messages are deleted. Default is `FALSE`.

Value

None

See Also

Other development functions: `add_dependencies()`, `add_github_actions_check()`, `add_github_actions_codecov()`, `add_r_depend()`, `add_to_buildignore()`, `add_to_gitignore()`

Examples

```
## Not run:
add_github_actions_pkgdown()

## End(Not run)
```

add_github_actions_pkgdown_badge *Add a Website deployment badge*

Description

This function adds a Website deployment badge to the `README.Rmd`. This function must be run after `add_github_actions_pkgdown()` which will setup GitHub Actions to build and deploy package website.

Usage

```
add_github_actions_pkgdown_badge(organisation = NULL, quiet = FALSE)
```

Arguments

`organisation`	a character of length 1 The name of the GitHub organisation to host the package. If `NULL` (default) the GitHub account will be used.
`quiet`	a logical value If `TRUE` messages are deleted. Default is `FALSE`.

Value

A Markdown badge expression

See Also

Other adding badges: `add_codecov_badge()`, `add_cran_badge()`, `add_dependencies_badge()`, `add_github_actions_check_badge()`, `add_github_actions_codecov_badge()`, `add_license_badge()`, `add_lifecycle_badge()`, `add_repostatus_badge()`

Examples

```
## Not run:
add_github_actions_pkgdown_badge()

## End(Not run)
```

add_license *Add a LICENSE*

Description

This function adds a license to the project. It will add the license name in the `License` field of the `DESCRIPTION` file and write the content of the license in the `License.md` file.

Usage

```
add_license(license = NULL, given = NULL, family = NULL, quiet = FALSE)
```

Arguments

`license`	a character of length 1 The chosen license. Run `get_licenses()` to select an appropriate one.
`given`	a character of length 1 The given name of the copyright holder. Only required if `license = 'MIT'`. If is `NULL` (default) and `license = 'MIT'`, this function will try to retrieve the value of this parameter from the `.Rprofile` file (edited with `set_credentials()`).
`family`	a character of length 1 The family name of the copyright holder. Only required if `license = 'MIT'`. If is `NULL` (default) and `license = 'MIT'`, this function will try to retrieve the value of this parameter from the `.Rprofile` file (edited with `set_credentials()`).
`quiet`	a logical value If `TRUE` messages are deleted. Default is `FALSE`.

Value

None

See Also

Other create files: `add_citation()`, `add_compendium()`, `add_description()`, `add_makefile()`, `add_package_doc()`, `add_readme_rmd()`, `add_testthat()`, `add_vignette()`

Examples

```
## Not run:
add_license(license = "MIT")
add_license(license = "GPL (>= 2)")

## End(Not run)
```

add_license_badge *Add a license badge*

Description

This function adds or updates the license badge to the `README.Rmd`. This function reads the `License` field of the `DESCRIPTION` file. Be sure to previously run `add_license()` function.

Usage

```
add_license_badge(quiet = FALSE)
```

Arguments

`quiet`	a logical value If `TRUE` messages are deleted. Default is `FALSE`.

Value

A Markdown badge expression

See Also

Other adding badges: `add_codecov_badge()`, `add_cran_badge()`, `add_dependencies_badge()`, `add_github_actions_check_badge()`, `add_github_actions_codecov_badge()`, `add_github_actions_pkgdown_badge()`, `add_lifecycle_badge()`, `add_repostatus_badge()`

Examples

```
## Not run:
add_license_badge()

## End(Not run)
```

add_lifecycle_badge *Add a Life cycle badge*

Description

This function adds or updates the Life cycle badge to the `README.Rmd`. It is based on the standard defined at https://lifecycle.r-lib.org/articles/stages.html.

Usage

```
add_lifecycle_badge(lifecycle = "experimental", quiet = FALSE)
```

Arguments

`lifecycle`	a character of length 1 Accepted stages are: `'experimental'` (default), `'stable'`, `'deprecated'`, or `'superseded'`.
`quiet`	a logical value If `TRUE` messages are deleted. Default is `FALSE`.

Details

The project can have the following life cycle stage:

- **Experimental** - An experimental project is made available so user can try it out and provide feedback, but come with no promises for long term stability.
- **Stable** - A project is considered stable when the author is happy with its interface, does not see major issues, and is happy to share it with the world.
- **Superseded** - A superseded project has a known better alternative, and it is not going away. Superseded project will not receive new features, but will receive any critical bug fixes needed to keep it working.
- **Deprecated** - A deprecated project has a better alternative available and is scheduled for removal.

Value

A Markdown badge expression

See Also

Other adding badges: `add_codecov_badge()`, `add_cran_badge()`, `add_dependencies_badge()`, `add_github_actions_check_badge()`, `add_github_actions_codecov_badge()`, `add_github_actions_pkgdown_badge()`, `add_license_badge()`, `add_repostatus_badge()`

Examples

```
## Not run:
add_lifecycle_badge()
add_lifecycle_badge(lifecycle = "stable")

## End(Not run)
```

add_makefile *Create a Make-like R file*

Description

This function creates a Make-like R file (`make.R`) at the root of the project based on a template. To be used only if the project is a research compendium. The content of this file provides some guidelines. See also `new_compendium()` for further information.

Usage

```
add_makefile(
  given = NULL,
  family = NULL,
  email = NULL,
  open = TRUE,
  overwrite = FALSE,
  quiet = FALSE
)
```

Arguments

`given`	a character of length 1 The given name of the package maintainer.
`family`	a character of length 1 The family name of the package maintainer.
`email`	a character of length 1 The email address of the package maintainer.
`open`	a logical value If `TRUE` (default) the file is opened in the editor.
`overwrite`	a logical value If this file is already present and `overwrite = TRUE`, it will be erased and replaced. Default is `FALSE`.
`quiet`	a logical value If `TRUE` messages are deleted. Default is `FALSE`.

Value

None

See Also

Other create files: `add_citation()`, `add_compendium()`, `add_description()`, `add_license()`, `add_package_doc()`, `add_readme_rmd()`, `add_testthat()`, `add_vignette()`

Examples

```
## Not run:
add_makefile()

## End(Not run)
```

---

add_package_doc *Create a package-level documentation file*

Description

This function adds a package-level documentation file (pkg-package.R) in the R/ folder. This file will make help available to the user via ?pkg (where pkg is the name of the package). It a good place to put general directives like @import and @importFrom.

Usage

```
add_package_doc(open = TRUE, overwrite = FALSE, quiet = FALSE)
```

Arguments

open	a logical value If TRUE (default) the file is opened in the editor.
overwrite	a logical value If this file is already present and overwrite = TRUE, it will be erased and replaced. Default is FALSE.
quiet	a logical value If TRUE messages are deleted. Default is FALSE.

Value

None

See Also

Other create files: add_citation(), add_compendium(), add_description(), add_license(), add_makefile(), add_readme_rmd(), add_testthat(), add_vignette()

Examples

```
## Not run:
add_package_doc()

## End(Not run)
```

add_readme_rmd *Create a README file*

Description

This function creates a README.Rmd file at the root of the project based on a template (adapted from usethis). Once edited user needs to knit it into a README.Rmd (or use the function refresh()).

Usage

```
add_readme_rmd(
  type = "package",
  given = NULL,
  family = NULL,
  organisation = NULL,
  open = TRUE,
  overwrite = FALSE,
  quiet = FALSE
)
```

Arguments

type	a character of length 1 If package (default) a GitHub README.Rmd specific to an R package will be created. If compendium a GitHub README.Rmd specific to a research compendium will be created.
given	a character of length 1 The given name of the package maintainer.
family	a character of length 1 The family name of the package maintainer.
organisation	a character of length 1 The name of the GitHub organisation to host the package. If NULL (default) the GitHub account will be used. This argument is used to set the URL of the package (hosted on GitHub).
open	a logical value If TRUE (default) the file is opened in the editor.
overwrite	a logical value If this file is already present and overwrite = TRUE, it will be erased and replaced. Default is FALSE.
quiet	a logical value If TRUE messages are deleted. Default is FALSE.

Value

None

See Also

Other create files: add_citation(), add_compendium(), add_description(), add_license(), add_makefile(), add_package_doc(), add_testthat(), add_vignette()

Examples

```
## Not run:
add_readme_rmd(type = "package")

## End(Not run)
```

`add_repostatus_badge` *Add a Repository status badge*

Description

This function adds or updates the Repo Status badge of the project to the `README.Rmd`. It is based on the standard defined by the https://www.repostatus.org project.

Usage

```
add_repostatus_badge(status = "concept", quiet = FALSE)
```

Arguments

`status`	a character of length 1 Accepted status are: `'concept'` (default), `'wip'`, `'suspended'`, `'abandoned'`, `'active'`, `'inactive'`, or `'unsupported'`.
`quiet`	a logical value If `TRUE` messages are deleted. Default is `FALSE`.

Details

The project can have the following status:

- **Concept** - Minimal or no implementation has been done yet, or the repository is only intended to be a limited example, demo, or proof-of-concept.
- **WIP** - Initial development is in progress, but there has not yet been a stable, usable release suitable for the public.
- **Suspended** - Initial development has started, but there has not yet been a stable, usable release; work has been stopped for the time being but the author(s) intend on resuming work.
- **Abandoned** - Initial development has started, but there has not yet been a stable, usable release; the project has been abandoned and the author(s) do not intend on continuing development.
- **Active** - The project has reached a stable, usable state and is being actively developed.

- **Inactive** - The project has reached a stable, usable state but is no longer being actively developed; support/maintenance will be provided as time allows.
- **Unsupported** - The project has reached a stable, usable state but the author(s) have ceased all work on it. A new maintainer may be desired.

Value

A Markdown badge expression

See Also

Other adding badges: `add_codecov_badge()`, `add_cran_badge()`, `add_dependencies_badge()`, `add_github_actions_check_badge()`, `add_github_actions_codecov_badge()`, `add_github_actions_pkgdown_badge()`, `add_license_badge()`, `add_lifecycle_badge()`

Examples

```
## Not run:
add_repostatus_badge()
add_repostatus_badge(status = "active")

## End(Not run)
```

---

add_r_depend *Add minimal R version to DESCRIPTION*

Description

This function adds the minimal R version to the `Depends` field of the `DESCRIPTION` file. This version corresponds to the higher version of R among all dependencies. If no dependencies mentions minimal R version, the `DESCRIPTION` is not modified.

Usage

```
add_r_depend()
```

Value

None

See Also

Other development functions: `add_dependencies()`, `add_github_actions_check()`, `add_github_actions_codecov()`, `add_github_actions_pkgdown()`, `add_to_buildignore()`, `add_to_gitignore()`

Examples

```
## Not run:
add_r_depend()

## End(Not run)
```

`add_testthat` *Initialize units tests*

Description

This function initializes units tests settings by running `usethis::use_testthat()` and by adding an example units tests file `tests/testthat/test-demo.R`. The sample file will test a demo function created in `R/fun-demo.R`.

Usage

```
add_testthat()
```

Value

None

See Also

Other create files: `add_citation()`, `add_compendium()`, `add_description()`, `add_license()`, `add_makefile()`, `add_package_doc()`, `add_readme_rmd()`, `add_vignette()`

Examples

```
## Not run:
add_testthat()

## End(Not run)
```

`add_to_buildignore` *Add to Rbuildignore file*

Description

This function adds files/folders to the `.Rbuildignore` file. If a `.Rbuildignore` is already present, files to be ignored while checking package are just added to this file. Otherwise a new file is created.

Usage

```
add_to_buildignore(x, open = FALSE, quiet = FALSE)
```

Arguments

x	a character vector One or several files/directories names to be added to the `.Rbuildignore`. This argument is mandatory.
open	a logical value If `TRUE` the `.Rbuildignore` file is opened in the editor. Default is `FALSE`.
quiet	a logical value If `TRUE` messages are deleted. Default is `FALSE`.

Value

None

See Also

Other development functions: `add_dependencies()`, `add_github_actions_check()`, `add_github_actions_codecov()`, `add_github_actions_pkgdown()`, `add_r_depend()`, `add_to_gitignore()`

Examples

```
## Not run:
add_to_buildignore(open = TRUE)
add_to_buildignore(".DS_Store")

## End(Not run)
```

---

add_to_gitignore *Create a gitignore file*

Description

This function creates a `.gitignore` file at the root of the project based on a template (specific to R). If a `.gitignore` is already present, files to be untracked by **git** are just added to this file.

Usage

```
add_to_gitignore(x, open = FALSE, quiet = FALSE)
```

Arguments

x	a character vector One or several files/directories names to be added to the `.gitignore`.
open	a logical value If `TRUE` the `.gitignore` file is opened in the editor. Default is `FALSE`.
quiet	a logical value If `TRUE` messages are deleted. Default is `FALSE`.

Value

None

See Also

Other development functions: add_dependencies(), add_github_actions_check(), add_github_actions_codecov(), add_github_actions_pkgdown(), add_r_depend(), add_to_buildignore()

Examples

```
## Not run:
add_to_gitignore(open = TRUE)
add_to_gitignore(".DS_Store")

## End(Not run)
```

---

add_vignette *Create a vignette document*

Description

This function adds a vignette in the folder vignettes/. It also adds dependencies knitr and rmarkdown in the field Suggests of the DESCRIPTION file (if not already present in fields Imports).

Usage

```
add_vignette(
  filename = NULL,
  title = NULL,
  open = TRUE,
  overwrite = FALSE,
  quiet = FALSE
)
```

Arguments

filename	a character of length 1 The name of the .Rmd file to be created. If NULL (default)the .Rmd file will be named pkg.Rmd where pkg is your package name.
title	a character of length 1 The title of the vignette. If NULL (default) the title will be Introduction to pkg (where pkg is your package name).
open	a logical value If TRUE (default) the file is opened in the editor.
overwrite	a logical value If this file is already present and overwrite = TRUE, it will be erased and replaced. Default is FALSE.

quiet — a logical value
If TRUE messages are deleted. Default is FALSE.

Value

None

See Also

Other create files: add_citation(), add_compendium(), add_description(), add_license(), add_makefile(), add_package_doc(), add_readme_rmd(), add_testthat()

Examples

```
## Not run:
## Default vignette ----
add_vignette()

## Custom vignette ----
add_vignette(filename = "get_started", title = "Get started")

## End(Not run)
```

---

get_all_dependencies *Get all external dependencies*

Description

This function gets all the external packages that the project needs. It is used the generate the *Dependencies* badge (add_dependencies_badge()).

Usage

```
get_all_dependencies(pkg = NULL)
```

Arguments

pkg — a character of length 1
The name of a CRAN package or NULL (default). If NULL get dependencies of the local (uninstalled) project (package or compendium).

Value

A list of three vectors:

- base_deps, a vector of base packages;
- direct_deps, a vector of direct packages;
- all_deps, a vector of all dependencies (recursively obtained).

See Also

Other utilities functions: `get_all_functions()`, `get_licenses()`, `get_minimal_r_version()`

Examples

```
## Not run:
## Update dependencies ----
add_dependencies()

## Get all dependencies ----
deps <- get_all_dependencies()
unlist(lapply(deps, length))

## Can be used for a CRAN package ----
deps <- get_all_dependencies("usethis")
unlist(lapply(deps, length))

## End(Not run)
```

---

get_all_functions *List all functions in the package*

Description

This function returns a list of all the functions (exported and internal) available with the package. As this function scans the NAMESPACE and the R/ folder, it is recommended to run `devtools::document()` before.

Usage

```
get_all_functions()
```

Value

A list of two vectors:

- `external`, a vector of exported functions name;
- `internal`, a vector of internal functions name.

See Also

Other utilities functions: `get_all_dependencies()`, `get_licenses()`, `get_minimal_r_version()`

Examples

```
## Not run:
## Update NAMESPACE ----
devtools::document()

## List all implemented functions ----
get_all_functions()

## End(Not run)
```

`get_licenses` *List all available licenses*

Description

This function returns a list of all available licenses. This is particularly useful to get the right spelling of the license to be passed to `new_package()`, `new_compendium()`, or `add_license()`.

Usage

```
get_licenses()
```

Value

A data frame with the following two variables:

- `tag`, the license name to be used with `add_license()`;
- `url`, the URL of the license description.

See Also

Other utilities functions: `get_all_dependencies()`, `get_all_functions()`, `get_minimal_r_version()`

Examples

```
get_licenses()
```

`get_minimal_r_version` *Get required minimal R version*

Description

This function detects the minimal required R version for the project based on minimal required R version of its dependencies. It can be used to update the `Depends` field of the `DESCRIPTION` file.

Usage

```
get_minimal_r_version(pkg = NULL)
```

Arguments

`pkg`	a character of length 1 The name of a CRAN package or `NULL` (default). If `NULL` get minimal required R version of the local (uninstalled) project (package or compendium).

Value

A character with the minimal required R version.

See Also

Other utilities functions: `get_all_dependencies()`, `get_all_functions()`, `get_licenses()`

Examples

```
## Not run:
## Update dependencies ----
add_dependencies()

## Minimal R version of a project ----
get_minimal_r_version()

## Minimal R version of a CRAN package ----
get_minimal_r_version("usethis")

## End(Not run)
```

`new_compendium` *Create an R compendium structure*

Description

This function creates a research compendium (i.e. a predefined files/folders structure) to help user organizing files/folders to run analysis.

In addition to common R packages files/folders (see `new_package()` for further information) this function will created these following folders:

- data/: a folder to store raw data. Note that these data must never be modified. If user want to modify them it is recommended to export new data in outputs/.
- rscripts/: a folder to write analyses instructions, i.e. R scripts. If user need to create R functions it is recommended to write them in the R/ folder.
- outputs/: a folder to store intermediate and final outputs generated by the R scripts.
- figures/: a folder to store figures generated by the R scripts.
- paper/: a folder to store all the material to write a paper/report (i.e. bibliographic references, templates, Rmarkdown document, etc.). It is also recommended that the user writes relative paths using the package `here` so he/she can easily access to materials stored in outputs/ and figures/.

This function also creates a Make-like R file (`make.R`). This file contains two main lines:

- `devtools::install_deps()`: downloads the external dependencies required by the project (an alternative to `install.packages()`). Ideal for sharing;
- `devtools::load_all()`: loads external dependencies and R functions (an alternative to `library()` and `source()` respectively).

As the user writes R scripts he/she can add the following line in this file: `source(here::here("rscripts","script_X.R"))`. Then he/she can source the entire `make.R` to run analysis. The function `add_dependencies()` can be used to automatically add external dependencies in the `DESCRIPTION` file.

It is recommended, for a better reproducibility, to call external dependencies as `pkg::fun()` or with @import or @importFrom in R functions instead of using `library()`.

All these files/folders are added to the `.Rbuildignore` so the rest of the project (e.g. R functions) can be used (or installed) as a R package.

Usage

```
new_compendium(
  compendium = ".",
  license = "GPL (>= 2)",
  status = NULL,
  lifecycle = NULL,
  vignette = FALSE,
  test = FALSE,
  create_repo = TRUE,
```

```
  private = FALSE,
  gh_check = TRUE,
  codecov = FALSE,
  website = TRUE,
  given = NULL,
  family = NULL,
  email = NULL,
  orcid = NULL,
  organisation = NULL,
  overwrite = FALSE,
  quiet = FALSE
)
```

Arguments

compendium	a character vector of length 1 By default, compendium folders are created at the root of the project. User can change their location with this argument. For instance, if `compendium = 'analysis'`, compendium folders will be created inside the directory analysis/.
license	a character vector of length 1 The license to be used for this project. Run `get_licenses()` to choose an appropriate one. Default is `license = 'GPL (>= 2)'` The license can be changed later by calling `add_license()` (and `add_license_badge()` or `refresh()` to update the corresponding badge in the README).
status	a character vector of length 1 The status of the project according to the standard defined by the https://www.repostatus.org project. One among `'concept'` `'wip'`, `'suspended'`, `'abandoned'`, `'active'`, `'inactive'`, or `'unsupported'`. See `add_repostatus_badge()` for further information. This argument is used to add a badge to the `README.Rmd` to help visitors to better understand your project. Default is `status = NULL`. This status can be added/changed later by using `add_repostatus_badge()`.
lifecycle	a character vector of length 1 The life cycle stage of the project according to the standard defined at https://lifecycle.r-lib.org/articles/stages.html. One among `'experimental'`, `'stable'`, `'deprecated'`, or `'superseded'`. See `add_lifecycle_badge()` for further information. This argument is used to add a badge to the `README.Rmd` to help visitors to better understand your project. Default is `lifecycle = NULL`. This stage can be added/changed later by using `add_lifecycle_badge()`.
vignette	a logical value If `TRUE` creates a vignette in vignettes/. Packages `knitr` and `rmarkdown` are also added to the `Suggests` field in the `DESCRIPTION` file. Default is `FALSE`.
test	a logical value If `TRUE` initializes units tests by running `usethis::use_testthat()`. Package `testthat` is also added to the `Suggests` field in the `DESCRIPTION` file. Default is `FALSE`.

`create_repo` a logical value

If `TRUE` (default) creates a repository (public if `private = FALSE` or private if `private = TRUE`) on GitHub. See below the section **Creating a GitHub repo**.

`private` a logical value

If `TRUE` creates a private repository on user GitHub account (or organisation). Default is `private = FALSE`.

`gh_check` a logical value

If `TRUE` (default) configures GitHub Actions to automatically check and test the package after each push. This will run R CMD check on the three major operating systems (Ubuntu, macOS, and Windows) on the latest release of R. See `add_github_actions_check()` for further information.

If `create_repo = FALSE` this argument is ignored.

`codecov` a logical value

If `TRUE` configures GitHub Actions to automatically report the code coverage of units tests after each push. See `add_github_actions_codecov()` for further information.

If `create_repo = FALSE` this argument is ignored. Default is `FALSE`.

`website` a logical value

If `TRUE` (default) configures GitHub Actions to automatically build and deploy the package website (using pkgdown) after each push. A **gh-pages** branch will be created using `usethis::use_github_pages()` and the GitHub repository will be automatically configured to deploy website.

If `create_repo = FALSE` this argument is ignored.

`given` a character vector of length 1

The given name of the maintainer of the package. If `NULL` (default) the function will try to get this value by reading the `.Rprofile` file.

For further information see `set_credentials()`.

`family` a character vector of length 1

The family name of the maintainer of the package. If `NULL` (default) the function will try to get this value by reading the `.Rprofile` file.

For further information see `set_credentials()`.

`email` a character vector of length 1

The email address of the maintainer of the package. If `NULL` (default) the function will try to get this value by reading the `.Rprofile` file.

For further information see `set_credentials()`.

`orcid` a character vector of length 1

The ORCID of the maintainer of the package. If `NULL` (default) the function will try to get this value by reading the `.Rprofile` file.

For further information see `set_credentials()`.

`organisation` a character vector of length 1

The GitHub organisation to host the repository. If defined it will overwrite the GitHub pseudo.

Default is `organisation = NULL` (the GitHub pseudo will be used).

overwrite	a logical value If TRUE files written from templates and modified by user are erased. Default is overwrite = FALSE. **Be careful while using this argument**.
quiet	a logical value If TRUE messages are deleted. Default is FALSE.

Value

None

See Also

Other setup functions: new_package(), refresh(), set_credentials()

Examples

```
## Not run:
library(rcompendium)

## Define **ONCE FOR ALL** your credentials ----
set_credentials(given = "John", family = "Doe",
                email = "john.doe@domain.com",
                orcid = "9999-9999-9999-9999", protocol = "ssh")

## Create an R package ----
new_compendium()

## Start adding data and developing functions and scripts ----
## ...

## Update package (documentation, dependencies, README, check) ----
refresh()

## End(Not run)
```

---

new_package *Create an R package structure*

Description

This function creates a new R package structure according to the current best practices. Essential features of an R package are created (DESCRIPTION and NAMESPACE files, and R/ and man/ folders). The project is also **versioned with git** and a generic R .gitignore is added.

IMPORTANT - Before using this function user needs to create a new folder (or a new project if using RStudio) and run this function inside this folder (by using setwd() or by opening the Rproj in a new RStudio session). **The name of the package will be the same as the name of this folder**. Some rules must be respected: https://r-pkgs.org/workflows101.html#naming.

Some fields of the DESCRIPTION file (maintainer information, package name, license, URLs, and roxygen2 version) are automatically filled but others (like title and description) need to be edited manually.

Additional features are also created: a CITATION file, a README.Rmd, and tests/ and vignettes/ folders (optional). See the vignette Get started for a complete overview of the full structure.

A GitHub repository can also be created (default) following the "GitHub last" workflow (https://happygitwithr.com/existing-github-last.html). Configuration files for GitHub Actions to automatically 1) check the package, 2) test and report code coverage, and 3) deploy the website pkgdown will be added in the .github/ folder. See below the section **Creating a GitHub repo**.

Usage

```
new_package(
  license = "GPL (>= 2)",
  status = "concept",
  lifecycle = "experimental",
  vignette = TRUE,
  test = TRUE,
  create_repo = TRUE,
  private = FALSE,
  gh_check = TRUE,
  codecov = TRUE,
  website = TRUE,
  given = NULL,
  family = NULL,
  email = NULL,
  orcid = NULL,
  organisation = NULL,
  overwrite = FALSE,
  quiet = FALSE
)
```

Arguments

license	a character vector of length 1 The license to be used for this package. Run get_licenses() to choose an appropriate one. Default is license = 'GPL (>= 2)' The license can be changed later by calling add_license() (and add_license_badge() or refresh() to update the corresponding badge in the README).
status	a character vector of length 1 The status of the project according to the standard defined by the https://www.repostatus.org project. One among 'concept' (default), 'wip', 'suspended', 'abandoned', 'active', 'inactive', or 'unsupported'. See add_repostatus_badge() for further information. This argument is used to add a badge to the README.Rmd to help visitors to better understand your project. If you don't want this badge use status = NULL. This status can be added/changed later by using add_repostatus_badge().

`lifecycle` a character vector of length 1

The life cycle stage of the project according to the standard defined at https://lifecycle.r-lib.org/articles/stages.html. One among `'experimental'` (default), `'stable'`, `'deprecated'`, or `'superseded'`. See `add_lifecycle_badge()` for further information.

This argument is used to add a badge to the `README.Rmd` to help visitors to better understand your project. If you don't want this badge use `lifecycle = NULL`.

This stage can be added/changed later by using `add_lifecycle_badge()`.

`vignette` a logical value

If `TRUE` (default) creates a vignette in vignettes/. Packages `knitr` and `rmarkdown` are also added to the `Suggests` field in the `DESCRIPTION` file.

`test` a logical value

If `TRUE` (default) initializes units tests by running `usethis::use_testthat()`. Package `testthat` is also added to the `Suggests` field in the `DESCRIPTION` file.

`create_repo` a logical value

If `TRUE` (default) creates a repository (public if `private = FALSE` or private if `private = TRUE`) on GitHub. See below the section **Creating a GitHub repo**.

`private` a logical value

If `TRUE` creates a private repository on user GitHub account (or organisation). Default is `private = FALSE`.

`gh_check` a logical value

If `TRUE` (default) configures GitHub Actions to automatically check and test the package after each push. This will run R CMD check on the three major operating systems (Ubuntu, macOS, and Windows) on the latest release of R. See `add_github_actions_check()` for further information.

If `create_repo = FALSE` this argument is ignored.

`codecov` a logical value

If `TRUE` (default) configures GitHub Actions to automatically report the code coverage of units tests after each push. See `add_github_actions_codecov()` for further information.

If `create_repo = FALSE` this argument is ignored.

`website` a logical value

If `TRUE` (default) configures GitHub Actions to automatically build and deploy the package website (using `pkgdown`) after each push. A **gh-pages** branch will be created using `usethis::use_github_pages()` and the GitHub repository will be automatically configured to deploy website.

If `create_repo = FALSE` this argument is ignored.

`given` a character vector of length 1

The given name of the maintainer of the package. If `NULL` (default) the function will try to get this value by reading the `.Rprofile` file.

For further information see `set_credentials()` and below the section **Managing credentials**.

`family` a character vector of length 1

The family name of the maintainer of the package. If `NULL` (default) the function will try to get this value by reading the `.Rprofile` file.

For further information see `set_credentials()` and below the section **Managing credentials**.

`email` a character vector of length 1

The email address of the maintainer of the package. If `NULL` (default) the function will try to get this value by reading the `.Rprofile` file.

For further information see `set_credentials()` and below the section **Managing credentials**.

`orcid` a character vector of length 1

The ORCID of the maintainer of the package. If `NULL` (default) the function will try to get this value by reading the `.Rprofile` file.

For further information see `set_credentials()` and below the section **Managing credentials**.

`organisation` a character vector of length 1

The GitHub organisation to host the repository. If defined it will overwrite the GitHub pseudo.

Default is `organisation = NULL` (the GitHub pseudo will be used).

`overwrite` a logical value

If `TRUE` files written from templates and modified by user are erased. Default is `overwrite = FALSE`. **Be careful while using this argument**.

`quiet` a logical value

If `TRUE` messages are deleted. Default is `FALSE`.

Value

None

Recommended workflow

The purpose of the package `rcompendium` is to make easier the creation of R package/research compendium so that user can focus on the code/analysis instead of wasting time organizing files.

The recommended workflow is:

1. Create an empty RStudio project;
2. Store your credentials with `set_credentials()` (if not already done);
3. Run `new_package()` to create a new package structure (and the GitHub repository);
4. Edit some metadata in `DESCRIPTION`, `CITATION`, and `README.Rmd`;
5. Implement, document & test functions (the fun part);
6. Update the project (update `.Rd` files, `NAMESPACE`, external dependencies in `DESCRIPTION`, re-knit `README.Rmd`, and check package integrity) with `refresh()`;
7. Repeat steps 5 and 6 while developing the package.

Managing credentials

You can use the arguments `given`, `family`, `email`, and `orcid` directly with the function `new_package()` (and others). But if you create a lot a projects (packages and/or compendiums) it can be frustrating in the long run.

An alternative is to use **ONCE AND FOR ALL** the function `set_credentials()` to permanently store this information in the `.Rprofile` file. If these arguments are set to `NULL` (default) each function of the package `rcompendium` will search in this `.Rprofile` file. It will save your time (it's the purpose of this package).

Even if you have stored your information in the `.Rprofile` file you will always be able to modify them on-the-fly (i.e. by using arguments of the `new_package()`) or permanently by re-running `set_credentials()`.

Configuring git

First run `gh::gh_whoami()` to see if your git is correctly configured. If so you should see something like:

```
{
"name": "John Doe",
"login": "jdoe",
"html_url": "https://github.com/jdoe",
...
}
```

Otherwise you might need to run:

```
gert::git_config_global(name = "user.name", value = "John Doe")
gert::git_config_global(name = "user.email", value = "john.doe@domain.com")
gert::git_config_global(name = "github.user", value = "jdoe")
```

See `gert::git_config()` for further information.

Creating a GitHub repo

To create the GitHub repository directly from R, the package `rcompendium` uses the function `usethis::use_github()`, an client to the GitHub REST API. The interaction with this API required an authentication method: a **GITHUB PAT** (Personal Access Token).

If you don't have a **GITHUB PAT** locally stored, you must:

1. Obtain a new one from your GitHub account. **Make sure to select at least the first two scopes (private repository and workflow)**
2. Store it in the `.Renviron` file by using `usethis::edit_r_environ()` and adding the following line: `GITHUB_PAT='99z9...z9'`

Run `usethis::gh_token_help()` for more information about getting and configuring a **GITHUB PAT**.

If everything is well configured you should see something like after calling `gh::gh_whoami()`:

```
{
"name": "John Doe",
"login": "jdoe",
"html_url": "https://github.com/jdoe",
"scopes": "delete_repo, repo, workflow",
"token": "99z9...z9"
}
```

And you will be able to create a GitHub repository directly from R!

See Also

Other setup functions: `new_compendium()`, `refresh()`, `set_credentials()`

Examples

```
## Not run:
library(rcompendium)

## Define **ONCE FOR ALL** your credentials ----
set_credentials(given = "John", family = "Doe",
                email = "john.doe@domain.com",
                orcid = "9999-9999-9999-9999", protocol = "ssh")

## Create an R package ----
new_package()

## Start developing functions ----
## ...

## Update package (documentation, dependencies, README, check) ----
refresh()

## End(Not run)
```

`refresh` *Refresh a package/research compendium*

Description

This function refreshes a package/research compendium. It will:

- Update `.Rd` files and `NAMESPACE` by using `devtools::document()`;
- Update external packages (in `DESCRIPTION` file) by using `add_dependencies()`;
- Update badges in `README.Rmd` (if already present);
- Re-knitr the `README.Rmd` by using `rmarkdown::render()`;
- Check package integrity by using `devtools::check()`;
- Run analysis by sourcing `make.R` (only for compendium).

Usage

```
refresh(compendium = NULL, make = FALSE, check = TRUE, quiet = FALSE)
```

Arguments

compendium	a character of length 1 The name of the folder to recursively detect dependencies to be added to the Imports field of DESCRIPTION file. It can be 'analysis/' (if additional folders, i.e. data/, outputs/, figures/, etc. have been created in this folder), '.' (if folders data/, outputs/, figures/, etc. have been created at the root of the project), etc. See new_compendium() for further information. Default is compendium = NULL (i.e. no additional folder are inspected but R/, NAMESPACE, vignettes/, and tests/ are still inspected).
make	a logical value If TRUE the Make-like R file make.R is sourced. Only for research compendium created with new_compendium(). Default is FALSE.
check	a logical value If TRUE (default) package integrity is checked using devtools::check().
quiet	a logical value If TRUE (default) message are deleted.

Value

None

See Also

Other setup functions: new_compendium(), new_package(), set_credentials()

Examples

```
## Not run:
library(rcompendium)

## Create an R package ----
new_package()

## Start developing functions ----
## ...

## Update package (documentation, dependencies, README, check) ----
refresh()

## End(Not run)
```

`set_credentials` *Store credentials to the .RProfile*

Description

This function stores user credentials in the `.Rprofile` file. Accepted credentials are listed below. This function is useful if user creates a lot of packages and/or research compendiums.

If the `.Rprofile` file does not exist this function will create it. Users need to paste the content of the clipboard to this file.

Usage

```
set_credentials(
  given = NULL,
  family = NULL,
  email = NULL,
  orcid = NULL,
  protocol = NULL
)
```

Arguments

`given`	a character of length 1 The given name of the package maintainer.
`family`	a character of length 1 The family name of the package maintainer.
`email`	a character of length 1 The email address of the package maintainer.
`orcid`	a character of length 1 The ORCID of the package maintainer.
`protocol`	a character of length 1 The GIT protocol used to communicate with the GitHub remote. One of `'https'` or `'ssh'`. If you don't know, keep the default value (i.e. `NULL`) and the protocol will be `'https'`.

Value

None

See Also

Other setup functions: `new_compendium()`, `new_package()`, `refresh()`

Examples

```
## Not run:
library(rcompendium)


## Define **ONCE FOR ALL** your credentials ----

set_credentials("John", "Doe", "john.doe@domain.com",
                orcid = "9999-9999-9999-9999", protocol = "https")

## End(Not run)
```

Index