Package 'pkgcheck'

September 28, 2026

Title Package Checks for 'rOpenSci'

Version 0.3.1

Description Check whether a package is ready for submission to the 'rOpenSci' peer review system ('rOpenSci' authors (2026) <doi:10.5281/zenodo.2553043> ``'rOpenSci' Packages: Development, Maintenance, and Peer Review"). Incorporates the 'goodpractice' package and many additional checks, including aspects related to maintenance of online public code repositories.

License GPL-3

URL https://docs.ropensci.org/pkgcheck/, https://github.com/ropensci-review-tools/pkgcheck

BugReports https://github.com/ropensci-review-tools/pkgcheck/issues

Depends R (>= 3.5.0)

Imports cli, covr, curl, fs, gert, gh, goodpractice, httr2, methods, pkgstats, praise, rappdirs, rmarkdown, rprojroot, rvest, srr, withr

Suggests callr, httptest2, jsonlite, knitr, memoise, pkgbuild, roxygen2, testthat (>= 3.0.0), xfun

VignetteBuilder knitr

Config/testthat/edition 3

Config/ropensci/maintainer staff

Encoding UTF-8

Language en-GB

NeedsCompilation yes

Config/roxygen2/version 8.1.0

Author Mark Padgham [aut, cre] (ORCID: <https://orcid.org/0000-0003-2172-5265>), Maëlle Salmon [aut], Jacob Wujciak-Jens [aut] (ORCID: <https://orcid.org/0000-0002-7281-3989>),

Kelli F. Johnson [ctb] (ORCID: <https://orcid.org/0000-0002-5149-451X>),
Eunseop Kim [aut] (ORCID: <https://orcid.org/0009-0000-2138-788X>),
Katrina Brock [ctb] (ORCID: <https://orcid.org/0009-0003-4678-9545>),
Andy Teucher [aut] (ORCID: <https://orcid.org/0000-0002-7840-692X>),
Eric R. Scott [aut] (ORCID: <https://orcid.org/0000-0002-7430-7879>)

Maintainer Mark Padgham <mark.padgham@email.com>

Repository CRAN

Date/Publication 2026-09-28 12:20:02 UTC

Contents

`checks_to_markdown` *Convert checks to markdown-formatted report*

Description

Convert checks to markdown-formatted report

Usage

```
checks_to_markdown(checks, render = FALSE)
```

Arguments

`checks`	Result of main pkgcheck function
`render`	If `TRUE`, render output as `html` document and open in browser.

Value

Markdown-formatted version of check report

See Also

Other extra: fn_names_on_cran(), list_pkgchecks(), logfile_names(), render_md2html()

Examples

```
f <- system.file ("extdata", "pkgstats_9.9.tar.gz", package = "pkgstats")
path <- pkgstats::extract_tarball (f)

## Not run:
checks <- pkgcheck (path, goodpractice = FALSE)
md <- checks_to_markdown (checks) # markdown-formatted character vector
md <- checks_to_markdown (checks, render = TRUE) # HTML version

## End(Not run)
fs::dir_delete (path)
```

---

fn_names_on_cran *Check whether a function name exists in any CRAN packages*

Description

Check whether a function name exists in any CRAN packages

Usage

```
fn_names_on_cran(fn_name, force_update = FALSE)
```

Arguments

fn_name	Character vector of one or more function names to check.
force_update	If 'TRUE', locally-cached data of all function names from all CRAN packages will be updated to latest version.

Value

A data.frame of three columns, "package", "version", and "fn_name", identifying any other packages matching specified function name(s). If no matches are found, the data.frame will have no rows.

See Also

Other extra: checks_to_markdown(), list_pkgchecks(), logfile_names(), render_md2html()

Examples

```
fn_names_on_cran (c ("min", "max"))
```

get_default_github_branch *get_default_github_branch*

Description

get_default_github_branch

Usage

```
get_default_github_branch(org, repo)
```

Arguments

org	Github organization
repo	Github repository

Value

Name of default branch on GitHub

Note

This function is not intended to be called directly, and is only exported to enable it to be used within the **plumber** API.

See Also

Other github: `get_latest_commit()`, `use_github_action_pkgcheck()`

Examples

```
org <- "ropensci-review-tools"
repo <- "pkgcheck"

branch <- get_default_github_branch (org, repo)
```

get_latest_commit *get_latest_commit*

Description

get_latest_commit

Usage

```
get_latest_commit(org, repo, branch = NULL)
```

Arguments

org	Github organization
repo	Github repository
branch	Branch from which to get latest commit

Value

Details of latest commit including OID hash

Note

This returns the latest commit from the default branch as specified on GitHub, which will not necessarily be the same as information returned from `gert::git_info` if the `HEAD` of a local repository does not point to the same default branch.

See Also

Other github: `get_default_github_branch()`, `use_github_action_pkgcheck()`

Examples

```
org <- "ropensci-review-tools"
repo <- "pkgcheck"

commit <- get_latest_commit (org, repo)
```

list_pkgchecks *List all checks currently implemented*

Description

List all checks currently implemented

Usage

```
list_pkgchecks(quiet = FALSE)
```

Arguments

quiet	If TRUE, print all checks to screen. Function invisibly returns list of checks regardless.

Value

Character vector of names of all checks (invisibly)

See Also

Other extra: checks_to_markdown(), fn_names_on_cran(), logfile_names(), render_md2html()

Examples

```
list_pkgchecks ()
```

logfile_names *Set up stdout & stderr cache files for* r_bg *process*

Description

Set up stdout & stderr cache files for r_bg process

Usage

```
logfile_names(path)
```

Arguments

path	Path to local repository

Value

Vector of two strings holding respective local paths to `stdout` and `stderr` files for `r_bg` process controlling the main pkgcheck function when executed in background mode.

Note

These files are needed for the **callr** `r_bg` process which controls the main pkgcheck. The `stdout` and `stderr` pipes from the process are stored in the cache directory so they can be inspected via their own distinct endpoint calls.

See Also

Other extra: `checks_to_markdown()`, `fn_names_on_cran()`, `list_pkgchecks()`, `render_md2html()`

Examples

```
f <- system.file ("extdata", "pkgstats_9.9.tar.gz", package = "pkgstats")
path <- pkgstats::extract_tarball (f)
on.exit (fs::dir_delete (path))

logfiles <- logfile_names (path)
print (logfiles)
```

---

pkgcheck *Generate report on package compliance with rOpenSci Statistical Software requirements*

Description

Generate report on package compliance with rOpenSci Statistical Software requirements

Usage

```
pkgcheck(
  path = ".",
  goodpractice = TRUE,
  use_cache = TRUE,
  extra_env = .GlobalEnv
)
```

Arguments

`path`	Path to local repository
`goodpractice`	If `FALSE`, skip most goodpractice checks except **lintr** and 'DESCRIPTION' checks. May be useful in development stages to more quickly check other aspects.

use_cache	Checks are cached for rapid retrieval, and only re-run if the git hash of the local repository changes. Setting use_cache to FALSE will force checks to be re-run even if the git hash has not changed.
extra_env	Additional environments from which to collate checks. Other package names may be appended using c, as in c(.GlobalEnv, "mypkg").

Value

A pkgcheck object detailing all package assessments automatically applied to packages submitted for peer review.

See Also

Other pkgcheck_fns: pkgcheck_bg(), print.pkgcheck()

Examples

```
f <- system.file ("extdata", "pkgstats_9.9.tar.gz", package = "pkgstats")
path <- pkgstats::extract_tarball (f)

## Not run:
checks <- pkgcheck (path)
summary (checks)

## End(Not run)
fs::dir_delete (path)
```

---

pkgcheck_bg *Generate report on package compliance with rOpenSci Statistical Software requirements as background process*

Description

Generate report on package compliance with rOpenSci Statistical Software requirements as background process

Usage

```
pkgcheck_bg(path)
```

Arguments

path	Path to local repository

Value

A **processx** object connecting to the background process generating the main pkgcheck results (see Note).

Note

The return object will by default display whether it is still running, or whether it has finished. Once it has finished, the results can be obtained by calling `$get_result()`, or the main pkgcheck function can be called to quickly retrieve the main results from local cache.

This function does not accept the `extra_env` parameter of the main pkgcheck function, and can not be used to run extra, locally-defined checks.

See Also

Other pkgcheck_fns: `pkgcheck()`, `print.pkgcheck()`

Examples

```
f <- system.file ("extdata", "pkgstats_9.9.tar.gz", package = "pkgstats")
path <- pkgstats::extract_tarball (f)

## Not run:
# Foreground checks as "blocking" process which will return
# only after all checks have finished:
checks <- pkgcheck (path)

# Or run process in background, do other things in the meantime,
# and obtain checks once they have finished:
ps <- pkgcheck_bg (path)
ps # print status to screen, same as 'ps$print()'
# Once finished, 'pkgcheck' results can be extracted with:
checks <- ps$get_result ()

## End(Not run)
fs::dir_delete (path)
```

`print.pkgcheck` *Generic print method for 'pkgcheck' objects.*

Description

Generic print method for 'pkgcheck' objects.

Usage

```
## S3 method for class 'pkgcheck'
print(x, deps = FALSE, ...)
```

Arguments

`x`	A 'pkgcheck' object to be printed.
`deps`	If 'TRUE', include details of dependency packages and function usage.
`...`	Further arguments pass to or from other methods (not used here).

Value

Nothing. Method called purely for side-effect of printing to screen.

See Also

Other pkgcheck_fns: pkgcheck(), pkgcheck_bg()

Examples

```
f <- system.file ("extdata", "pkgstats_9.9.tar.gz", package = "pkgstats")
path <- pkgstats::extract_tarball (f)

## Not run:
checks <- pkgcheck (path)
summary (checks) # print summary only
print (checks) # print full checks, starting with summary

## End(Not run)
fs::dir_delete (path)
```

---

read_pkg_guide *Browse packaging guidelines*

Description

A convenience function to automatically open the web page of rOpenSci's "Package Development Guide" in the default browser.

Usage

```
read_pkg_guide(which = c("release", "dev"))
```

Arguments

which	Whether to read the released or "dev" development version.

Value

Nothing. Function called purely for side-effect of opening web page with package guidelines.

Examples

```
read_pkg_guide ()
```

render_md2html *render markdown-formatted input into 'html'*

Description

render markdown-formatted input into 'html'

Usage

```
render_md2html(md, open = TRUE)
```

Arguments

md	Result of checks_to_markdown function.
open	If TRUE, open html-rendered version in web browser.

Value

(invisible) Location of .html-formatted version of input.

See Also

Other extra: checks_to_markdown(), fn_names_on_cran(), list_pkgchecks(), logfile_names()

Examples

```
f <- system.file ("extdata", "pkgstats_9.9.tar.gz", package = "pkgstats")
path <- pkgstats::extract_tarball (f)

## Not run:
checks <- pkgcheck (path, goodpractice = FALSE)
# Generate standard markdown-formatted character vector:
md <- checks_to_markdown (checks)

# Directly generate HTML output:
h <- checks_to_markdown (checks, render = TRUE) # HTML version

# Or convert markdown-formatted version to HTML:
h <- render_md2html (md)

## End(Not run)
fs::dir_delete (path)
```

use_github_action_pkgcheck *Use pkgcheck Github Action*

Description

Creates a Github workflow file in `dir` integrate `pkgcheck()` into your CI.

Usage

```
use_github_action_pkgcheck(
  dir = ".github/workflows",
  overwrite = FALSE,
  file_name = "pkgcheck.yaml",
  branch = gert::git_branch(),
  inputs = NULL
)
```

Arguments

`dir`	Directory the file is written to.
`overwrite`	Overwrite existing file?
`file_name`	Name of the workflow file.
`branch`	Name of git branch for checks to run; defaults to currently active branch.
`inputs`	Named list of inputs to the `ropensci-review-tools/pkgcheck-action`. See details below.

Details

For more information on the action and advanced usage visit the action repository.

Value

The path to the new file, invisibly.

Inputs

Inputs with description and default values. Pass all values as strings, see examples.

```
inputs:
  ref:
   description: "The ref to checkout and check. Set to empty string to skip checkout."
    default: "${{ github.ref }}"
    required: true
  post-to-issue:
    description: "Should the pkgcheck results be posted as an issue?"
```

```
    # If you use the 'pull_request' trigger and the PR is from outside the repo
    # (e.g. a fork), the job will fail due to permission issues
    # if this is set to 'true'. The default will prevent this.
    default: ${{ github.event_name != 'pull_request' }}
    required: true
  issue-title:
   description: "Name for the issue containing the pkgcheck results. Will be created or updated."
    # This will create a new issue for every branch, set it to something fixed
    # to only create one issue that is updated via edits.
    default: "pkgcheck results - ${{ github.ref_name }}"
    required: true
  summary-only:
   description: "Only post the check summary to issue. Set to false to get the full results in the issue.'
    default: true
    required: true
  append-to-issue:
   description: "Should issue results be appended to existing issue, or posted in new issues."
    default: true
    required: true
```

See Also

Other github: `get_default_github_branch()`, `get_latest_commit()`

Examples

```
# These functions will by default create new files in the default `.dir`.
## Not run:
f <- use_github_action_pkgcheck (inputs = list (`post-to-issue` = "false"))
readLines (f)

f <- use_github_action_pkgcheck (branch = "main")

## End(Not run)
```

Index