

# Package: pkgcheck (via r-universe)

August 27, 2024

**Title** rOpenSci Package Checks

**Version** 0.1.2.058

**Description** Check whether a package is ready for submission to rOpenSci's peer review system.

**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, ghql, glue, goodpractice, httr, httr2, jsonlite, magrittr, methods, pkgstats, rappdirs, rmarkdown, rprojroot, rvest, srr, withr

**Suggests** callr, knitr, pkgbuild, roxygen2, testthat (>= 3.0.0), visNetwork,

**VignetteBuilder** knitr

**Remotes** mangothecat/goodpractice, ropensci-review-tools/pkgstats, ropensci-review-tools/srr

**Config/testthat/edition** 3

**Encoding** UTF-8

**Language** en-GB

**LazyData** true

**NeedsCompilation** yes

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.2

**Repository** <https://ropensci-review-tools.r-universe.dev>

**RemoteUrl** <https://github.com/ropensci-review-tools/pkgcheck>

**RemoteRef** HEAD

**RemoteSha** 559455f1f50ce317e9b8db77df5cfbda02fea609

## Contents

checks_to_markdown . . . . .	2
get_default_github_branch . . . . .	3
get_gh_token . . . . .	4
get_latest_commit . . . . .	4
list_pkgchecks . . . . .	5
logfile_names . . . . .	6
pkgcheck . . . . .	7
pkgcheck_bg . . . . .	8
print_pkgcheck . . . . .	9
read_pkg_guide . . . . .	10
render_md2html . . . . .	10
use_github_action_pkgcheck . . . . .	11

<b>Index</b>	<b>13</b>
--------------	-----------

---

checks_to_markdown	<i>Convert checks to markdown-formatted report</i>
--------------------	--

---

### Description

Convert checks to markdown-formatted report

### Usage

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

### Arguments

checks	Result of main <a href="#">pkgcheck</a> function
render	If TRUE, render output as html document and open in browser.

### Value

Markdown-formatted version of check report

### See Also

Other extra: [list\\_pkgchecks\(\)](#), [logfile\\_names\(\)](#), [render\\_md2html\(\)](#)

### Examples

```
## Not run:
checks <- pkgcheck ("/path/to/my/package")
md <- checks_to_markdown (checks) # markdown-formatted character vector
md <- checks_to_markdown (checks, render = TRUE) # HTML version

## End(Not run)
```

---

```
get_default_github_branch  
  get_default_github_branch
```

---

## Description

`get_default_github_branch`

## Usage

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

## Arguments

<code>org</code>	Github organization
<code>repo</code>	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\\_gh\\_token\(\)](#), [get\\_latest\\_commit\(\)](#), [use\\_github\\_action\\_pkgcheck\(\)](#)

## Examples

```
## Not run:  
org <- "ropensci-review-tools"  
repo <- "pkgcheck"  
branch <- get_default_github_branch (org, repo)  
  
## End(Not run)
```

---

<code>get_gh_token</code>	<i>Get GitHub token</i>
---------------------------	-------------------------

---

**Description**

Get GitHub token

**Usage**

```
get_gh_token(token_name = "")
```

**Arguments**

`token_name`      Optional name of token to use

**Value**

The value of the GitHub access token extracted from environment variables.

**See Also**

Other github: [get\\_default\\_github\\_branch\(\)](#), [get\\_latest\\_commit\(\)](#), [use\\_github\\_action\\_pkgcheck\(\)](#)

**Examples**

```
## Not run:
token <- get_gh_token ()

## End(Not run)
```

---

<code>get_latest_commit</code>	<i>get_latest_commit</i>
--------------------------------	--------------------------

---

**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\(\)](#), [get\\_gh\\_token\(\)](#), [use\\_github\\_action\\_pkgcheck\(\)](#)

**Examples**

```
## Not run:
org <- "ropensci-review-tools"
repo <- "pkgcheck"
commit <- get_latest_commit (org, repo)

## End(Not run)
```

---

list_pkgchecks	<i>List all checks currently implemented</i>
----------------	--

---

**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\(\)](#), [logfile\\_names\(\)](#), [render\\_md2html\(\)](#)

**Examples**

```
list_pkgchecks ()
```

---

logfile_names	<i>Set up stdout &amp; stderr cache files for r_bg process</i>
---------------	--

---

## 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\(\)](#), [list\\_pkgchecks\(\)](#), [render\\_md2html\(\)](#)

## Examples

```
## Not run:  
logfiles <- logfile_names ("/path/to/my/package")  
print (logfiles)  
  
## End(Not run)
```

---

pkgcheck	<i>Generate report on package compliance with rOpenSci Statistical Software requirements</i>
----------	--

---

## 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 goodpractice 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 for 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

```
## Not run:  
checks <- pkgcheck ("/path/to/my/package") # default full check  
summary (checks)  
# Or to run only checks implemented in 'pkgcheck' and not the  
# additional \pkg{goodpractice} checks:  
checks <- pkgcheck ("/path/to/my/package", goodpractice = FALSE)  
summary (checks)  
  
## End(Not run)
```

---

pkgcheck_bg	<i>Generate report on package compliance with rOpenSci Statistical Software requirements as background process</i>
-------------	--

---

### 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

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

# Or run process in background, do other things in the meantime,
# and obtain checks once they have finished:
ps <- pkgcheck_bg ("/path/to/my/package")
ps # print status to screen, same as 'ps$print()'
# To examine process state while running:
f <- ps$get_output_file ()
readLines (f) # or directly open file with local file viewer
```



```
# ... ultimately wait until 'running' changes to 'finished', then:
checks <- ps$get_result ()

## End(Not run)
```

---

print.pkgcheck	<i>Generic print method for 'pkgcheck' objects.</i>
----------------	---

---

## 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

```
## Not run:
checks <- pkgcheck ("/path/to/my/package")
print (checks) # print full checks, starting with summary
summary (checks) # print summary only

## End(Not run)
```

---

read_pkg_guide	<i>Browse packaging guidelines</i>
----------------	------------------------------------

---

**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**

```
## Not run:  
read_pkg_guide ()  
  
## End(Not run)
```

---

render_md2html	<i>render markdown-formatted input into 'html'</i>
----------------	--

---

**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 hmtl-rendered version in web browser.

**Value**

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

**See Also**

Other extra: [checks\\_to\\_markdown\(\)](#), [list\\_pkgchecks\(\)](#), [logfile\\_names\(\)](#)

**Examples**

```
## Not run:
checks <- pkgcheck ("/path/to/my/package")
# 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)
```

---

```
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\\_gh\\_token\(\)](#), [get\\_latest\\_commit\(\)](#)

**Examples**

```
## Not run:
use_github_action_pkgcheck (inputs = list (`post-to-issue` = "false"))
use_github_action_pkgcheck (branch = "main")

## End(Not run)
```

# Index

## \* **extra**

- checks\_to\_markdown, 2
- list\_pkgchecks, 5
- logfile\_names, 6
- render\_md2html, 10

## \* **github**

- get\_default\_github\_branch, 3
- get\_gh\_token, 4
- get\_latest\_commit, 4
- use\_github\_action\_pkgcheck, 11

## \* **pkgcheck\_fns**

- pkgcheck, 7
- pkgcheck\_bg, 8
- print.pkgcheck, 9

checks\_to\_markdown, 2, 5, 6, 10, 11

get\_default\_github\_branch, 3, 4, 5, 12

get\_gh\_token, 3, 4, 5, 12

get\_latest\_commit, 3, 4, 4, 12

list\_pkgchecks, 2, 5, 6, 11

logfile\_names, 2, 5, 6, 11

pkgcheck, 2, 6, 7, 8, 9

pkgcheck(), 11

pkgcheck\_bg, 7, 8, 9

print.pkgcheck, 7, 8, 9

read\_pkg\_guide, 10

render\_md2html, 2, 5, 6, 10

use\_github\_action\_pkgcheck, 3–5, 11