

# Package: goodpractice (via r-universe)

June 3, 2026

**Title** Advice on R Package Building

**Version** 1.0.5.9001

**Description** Give advice about good practices when building R packages.  
Advice includes functions and syntax to avoid, package structure, code complexity, code formatting, etc.

**License** MIT + file LICENSE

**URL** <https://docs.ropensci.org/goodpractice/>,  
<https://github.com/ropensci-review-tools/goodpractice>

**BugReports** <https://github.com/ropensci-review-tools/goodpractice/issues>

**Depends** R (>= 4.0.0)

**Imports** cli, covr, curl, cyclocomp (>= 1.1.0), desc, jsonlite, lintr (>= 3.0.0), praise, rcmdcheck, roxygen2, rstudioapi, spelling, tools, treesitter, treesitter.r, urlchecker, utils, whoami, withr

**Suggests** future, future.apply, kableExtra, knitr, rmarkdown, testthat (>= 3.0.0)

**VignetteBuilder** knitr

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**Config/testthat/edition** 3

**Config/ropensci/maintainer** staff

**Collate** 'api.R' 'customization.R' 'lists.R' 'chk\_avoided\_packages.R' 'treesitter.R' 'chk\_code\_structure.R' 'chk\_covr.R' 'chk\_cyclocomp.R' 'chk\_description.R' 'chk\_generic.R' 'chk\_lintr.R' 'chk\_namespace.R' 'chk\_rcmdcheck.R' 'chk\_rd.R' 'chk\_revdep.R' 'chk\_roxygen2.R' 'chk\_spelling.R' 'chk\_tidyverse.R' 'chk\_urlchecker.R' 'chk\_vignette.R' 'gp.R' 'package.R' 'prep\_utils.R' 'prep\_covr.R' 'prep\_cyclocomp.R' 'prep\_description.R' 'prep\_lintr.R' 'prep\_namespace.R' 'prep\_rcmdcheck.R' 'prep\_rd.R' 'prep\_revdep.R' 'prep\_roxygen2.R' 'prep\_source.R' 'prep\_spelling.R'

'prep\_tidyverse.R' 'prep\_urlchecker.R' 'prep\_vignette.R'  
 'print.R' 'rstudio\_markers.R' 'utils.R'

**Config/roxygen2/version** 8.0.0

**Config/pak/sysreqs** cmake git make libuv1-dev libxml2-dev libssl-dev

**Repository** <https://mpadge.r-universe.dev>

**Date/Publication** 2026-06-03 15:43:21 UTC

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

**RemoteRef** HEAD

**RemoteSha** d0f4243d1c9c79aa7ea49fa4038aca69028c2f15

## Contents

goodpractice-package . . . . .	2
all_check_groups . . . . .	3
all_checks . . . . .	4
checks . . . . .	4
checks_by_group . . . . .	5
customization . . . . .	6
default_checks . . . . .	7
describe_check . . . . .	7
describe_check_groups . . . . .	8
export_json . . . . .	9
failed_checks . . . . .	9
failed_positions . . . . .	10
gp . . . . .	11
print.goodPractice . . . . .	12
results . . . . .	13
tidyverse_checks . . . . .	14
<b>Index</b>	<b>15</b>

---

goodpractice-package    *goodpractice: Advice on R Package Building*

---

## Description

Give advice about good practices when building R packages. Advice includes functions and syntax to avoid, package structure, code complexity, code formatting, etc.

**Author(s)**

**Maintainer:** Mark Padgham <mark@ropensci.org> ([ORCID](#))

Authors:

- Mark Padgham <mark@ropensci.org> ([ORCID](#))
- Karina Marks <karina.marks@ascent.io> (GitHub: KarinaMarks)
- Daniel de Bortoli (GitHub: ddbortoli)
- Gabor Csardi <csardi.gabor@gmail.com>
- Hannah Frick <hannah.frick@gmail.com>
- Owen Jones <owenjonesuob@gmail.com> (GitHub: owenjonesuob)
- Hannah Alexander <halexander@mango-solutions.com>
- Athanasia Mo Mowinckel <a.m.mowinckel@psykologi.uio.no> ([ORCID](#)) (GitHub: dr-mowinckels)

Other contributors:

- Ascent Digital Services UK Limited (GitHub: MangoTheCat) [copyright holder]
- Ana Simmons <ana.simmons@ascent.io> (GitHub: anasimmons) [contributor]
- Fabian Scheipl (GitHub: fabian-s) [contributor]

**See Also**

Useful links:

- <https://docs.ropensci.org/goodpractice/>
- <https://github.com/ropensci-review-tools/goodpractice>
- Report bugs at <https://github.com/ropensci-review-tools/goodpractice/issues>

---

all\_check\_groups      *List available check group names*

---

**Description**

Returns the names of all registered check groups. Use these names with `checks_by_group()` to select checks by group, or with `options(goodpractice.exclude_check_groups = ...)` to skip groups. Full descriptions of each check group are return by `describe_check_groups()`.

**Usage**

```
all_check_groups()
```

**Value**

A character vector of check group names.

**Examples**

```
# Names of all check groups:
all_check_groups()

# List individual checks by group:
chks <- lapply(all_check_groups(), checks_by_group)
names(chks) <- all_check_groups()
chks
```

---

all_checks	<i>List the names of all checks</i>
------------	-------------------------------------

---

**Description**

List the names of all checks

**Usage**

```
all_checks()
```

**Value**

Character vector of checks

**Examples**

```
all_checks()
```

---

checks	<i>List all checks performed</i>
--------	----------------------------------

---

**Description**

List all checks performed

**Usage**

```
checks(gp)
```

**Arguments**

gp [gp](#) output.

**Value**

Character vector of check names.

**See Also**

Other API: [failed\\_checks\(\)](#), [results\(\)](#)

**Examples**

```
path <- system.file("bad1", package = "goodpractice")
# run a subset of all checks available
g <- gp(path, checks = all_checks()[3:16])
checks(g)
```

---

checks_by_group	<i>Select checks by check group</i>
-----------------	-------------------------------------

---

**Description**

Returns the names of all checks that belong to the given group(s). This makes it easy to run or inspect a specific category of checks without knowing individual check names.

**Usage**

```
checks_by_group(...)
```

**Arguments**

... Group names as character strings. Use [all\\_check\\_groups\(\)](#) to see available names.

**Value**

Character vector of check names

**Examples**

```
# run only DESCRIPTION and namespace checks
checks_by_group("description", "namespace")

# see what the lintr group covers
checks_by_group("lintr")
# See all checks by group:
lapply(all_check_groups(), checks_by_group)
# use directly in gp()
## Not run:
gp(".", checks = checks_by_group("description", "lintr"))

## End(Not run)
```

---

 customization

*Defining custom preparations and checks*


---

## Description

Defining custom preparations and checks

## Usage

```
make_prep(name, func)
```

```
make_check(description, check, gp, ...)
```

## Arguments

name	Name of the preparation function.
func	A function that takes two arguments: The path to the root directory of the package, and a logical argument: <code>quiet</code> . If <code>quiet</code> is true, the preparation function may print out diagnostic messages. The output of this function will be saved as the "name" entry of state, i.e. of the input for the check-functions (see example).
description	A description of the check.
check	A function that takes the state as an argument.
gp	A short description of what is good practice.
...	Further arguments. Most important: A preps argument that contains the names of all the preparation functions required for the check.

## Value

For `make_prep`: a preparation function. For `make_check`: a check object of class "check".

## Functions

- `make_prep()`: Create a preparation function
- `make_check()`: Create a check function

## Examples

```
# make a preparation function
url_prep <- make_prep(
  name = "desc",
  func = function(path, quiet) desc::description$new(path)
)
# and the corresponding check function
url_chk <- make_check(
  description = "URL field in DESCRIPTION",
```

```

tags = character(),
preps = "desc",
gp = "have a URL field in DESCRIPTION",
check = function(state) state$desc$has_fields("URL")
)
# use together in gp():
# (note that you have to list the name of your custom check in
# the checks-argument as well...)
bad1 <- system.file("bad1", package = "goodpractice")
res <- gp(bad1, checks = c("url", "no_description_depends"),
         extra_preps = list("desc" = url_prep),
         extra_checks = list("url" = url_chk))

```

---

default_checks	<i>List the names of default checks (excludes optional check sets)</i>
----------------	--

---

### Description

List the names of default checks (excludes optional check sets)

### Usage

```
default_checks()
```

### Value

Character vector of default check names

### Examples

```
default_checks()
```

---

describe_check	<i>Describe one or more checks</i>
----------------	------------------------------------

---

### Description

Describe one or more checks

### Usage

```
describe_check(check_name = NULL)
```

### Arguments

check\_name      Names of checks to be described.

**Value**

List of character descriptions for each check\_name

**Examples**

```
describe_check("rcmdcheck_non_portable_makevars")
check_name <- c("no_description_depends",
               "lintr_assignment_linter",
               "no_import_package_as_a_whole",
               "rcmdcheck_missing_docs")
describe_check(check_name)
# Or to see all checks:
## Not run:
  describe_check(all_checks())

## End(Not run)
```

---

describe\_check\_groups *Describe available check groups*

---

**Description**

Returns full descriptions of all registered check groups.

**Usage**

```
describe_check_groups()
```

**Value**

A named list of each check group defined in `all_check_groups()`, with text descriptions of each group.

**Examples**

```
# Names of all check groups:
all_check_groups()

# And corresponding descriptions:
describe_check_groups()
```

---

export_json	<i>Export failed checks to JSON</i>
-------------	-------------------------------------

---

**Description**

Export failed checks to JSON

**Usage**

```
export_json(gp, file, pretty = FALSE)
```

**Arguments**

gp	gp output.
file	Output connection or file.
pretty	Whether to pretty-print the JSON.

**Value**

Invisibly returns the path to the output file.

**Examples**

```
path <- system.file("bad1", package = "goodpractice")
g <- gp(path, checks = "description_url")
tmp <- tempfile(fileext = ".json")
export_json(g, tmp)
unlink(tmp)
```

---

failed_checks	<i>Names of the failed checks</i>
---------------	-----------------------------------

---

**Description**

Names of the failed checks

**Usage**

```
failed_checks(gp)
```

**Arguments**

gp	gp output.
----	------------

**Value**

Names of the failed checks.

**See Also**

Other API: [checks\(\)](#), [results\(\)](#)

**Examples**

```
path <- system.file("bad1", package = "goodpractice")
# run a subset of all checks available
g <- gp(path, checks = all_checks()[3:16])
failed_checks(g)
```

---

failed_positions	<i>Positions of check failures in the source code</i>
------------------	---

---

**Description**

Note that not all checks refer to the source code. For these the result will be NULL.

**Usage**

```
failed_positions(gp)
```

**Arguments**

gp [gp](#) output.

**Details**

For the ones that do, the results is a list, one for each failure. Since the same check can fail multiple times. A single failure is a list with entries: filename, line\_number, column\_number, ranges. ranges is a list of pairs of start and end positions for each line involved in the check.

**Value**

A list of lists of positions. See details below.

**Examples**

```
path <- system.file("bad1", package = "goodpractice")
g <- gp(path, checks = "description_url")
failed_positions(g)
```

---

gp *Run good practice checks*

---

**Description**

To see the results, just print it to the screen.

**Usage**

```
gp(
  path = ".",
  checks = default_checks(),
  extra_preps = NULL,
  extra_checks = NULL,
  quiet = TRUE
)
```

**Arguments**

path	Path to a package root.
checks	Character vector, the checks to run. Defaults to <code>default_checks</code> . Use <code>all_checks</code> to list all checks, or add optional sets like <code>tidyverse_checks</code> . When NULL, all registered checks are run, subject to any exclusions from <code>goodpractice.exclude_check_groups</code> or <code>GP_EXCLUDE_CHECK_GROUPS</code> .
extra_preps	Custom preparation functions. See <code>make_prep</code> on creating preparation functions.
extra_checks	Custom checks. See <code>make_check</code> on creating checks.
quiet	Whether to suppress output from the preparation functions. Note that not all preparation functions produce output, even if this option is set to FALSE.

**Value**

A `goodpractice` object that you can query with a simple API. See `results` to start.

**Excluding check groups**

When using the default `checks = all_checks()`, entire groups of checks can be excluded by group name via the `goodpractice.exclude_check_groups` option or the `GP_EXCLUDE_CHECK_GROUPS` environment variable (comma-separated). The option takes precedence.

```
# Skip URL and coverage checks:
options(goodpractice.exclude_check_groups = c("urlchecker", "covr"))
```

```
# Or via environment variable:
Sys.setenv(GP_EXCLUDE_CHECK_GROUPS = "urlchecker,covr")
```

Exclusion only applies when `checks = NULL` (the default). Explicit checks arguments are never filtered.

**Excluding files**

Specific files can be excluded from checks via the `goodpractice.exclude_path` option or the `GP_EXCLUDE_PATH` environment variable (comma-separated). Paths are relative to the package root.

```
options(goodpractice.exclude_path = c("R/RcppExports.R", "R/generated.R"))
```

```
# Or via environment variable:
```

```
Sys.setenv(GP_EXCLUDE_PATH = "R/RcppExports.R,R/generated.R")
```

Excluded files are skipped by `lintr`, `tree sitter`, `expression`, and `roxygen2` checks.

**Parallel preparation**

Preparation steps run sequentially by default. To run them in parallel, install **future.apply** and set a [plan](#):

```
future::plan("multisession")
gp(".")
```

Preps run in parallel only when a non-sequential plan is active. Prep functions must be independent: in parallel mode each prep receives the initial state snapshot, so a prep cannot read another prep's output. Only new state fields are merged back; if two preps write the same field, the second is dropped with a warning.

**Examples**

```
path <- system.file("bad1", package = "goodpractice")
# run a subset of all checks available
g <- gp(path, checks = all_checks()[3:16])
g
```

---

```
print.goodPractice    Print goodpractice results
```

---

**Description**

Print goodpractice results

**Usage**

```
## S3 method for class 'goodPractice'
print(x, groups = NULL, positions_limit = 5, ...)
```

**Arguments**

<code>x</code>	Object of class <code>goodPractice</code> , as returned by <code>gp()</code> .
<code>groups</code>	Name of check groups for which to print results, as vector of one or more of <code>all_check_groups()</code> .
<code>positions_limit</code>	How many positions to print at most.
<code>...</code>	Unused, for compatibility with <code>base::print()</code> generic method.

---

<code>results</code>	<i>Return all check results in a data frame</i>
----------------------	---

---

**Description**

Return all check results in a data frame

**Usage**

```
results(gp)
```

**Arguments**

<code>gp</code>	<code>gp</code> output.
-----------------	-------------------------

**Value**

Data frame, with columns:

<code>check</code>	The name of the check.
<code>passed</code>	Logical, whether the check passed.

**See Also**

Other API: `checks()`, `failed_checks()`

**Examples**

```
path <- system.file("bad1", package = "goodpractice")
# run a subset of all checks available
g <- gp(path, checks = all_checks()[3:16])
results(g)
```

---

tidyverse_checks	<i>List the names of tidyverse style checks</i>
------------------	---

---

**Description**

These checks are optional and not included in the default set. They are powered by [lint\\_package](#) using lintr's default linter set and respect any `.lintr` configuration file in the package root (e.g. to disable specific linters or add exclusions). Add them via `checks = c(default_checks(), tidyverse_checks())`.

**Usage**

```
tidyverse_checks()
```

**Value**

Character vector of tidyverse check names

**Examples**

```
tidyverse_checks()
```

# Index

## \* API

- checks, 4
- failed\_checks, 9
- results, 13
  
- all\_check\_groups, 3
- all\_check\_groups(), 5, 8, 13
- all\_checks, 4, 11
  
- base::print(), 13
  
- checks, 4
- checks(), 10, 13
- checks\_by\_group, 5
- checks\_by\_group(), 3
- customization, 6
  
- default\_checks, 7, 11
- describe\_check, 7
- describe\_check\_groups, 8
- describe\_check\_groups(), 3
  
- export\_json, 9
  
- failed\_checks, 9
- failed\_checks(), 5, 13
- failed\_positions, 10
  
- goodpractice (gp), 11
- goodpractice-package, 2
- gp, 4, 9, 10, 11, 13
- gp(), 13
  
- lint\_package, 14
  
- make\_check, 11
- make\_check (customization), 6
- make\_prep, 11
- make\_prep (customization), 6
  
- plan, 12
  
- print.goodPractice, 12
  
- results, 11, 13
- results(), 5, 10
  
- tidyverse\_checks, 11, 14