

# Package ‘rprojroot’

May 9, 2026

**Title** Finding Files in Project Subdirectories

**Version** 2.1.1

**Description** Robust, reliable and flexible paths to files below a project root. The 'root' of a project is defined as a directory that matches a certain criterion, e.g., it contains a certain regular file.

**License** MIT + file LICENSE

**URL** <https://rprojroot.r-lib.org/>, <https://github.com/r-lib/rprojroot>

**BugReports** <https://github.com/r-lib/rprojroot/issues>

**Depends** R (>= 3.0.0)

**Suggests** covr, knitr, lifecycle, rlang, rmarkdown, testthat (>= 3.2.0), withr

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Encoding** UTF-8

**RoxygenNote** 7.3.2.9000

**Config/autostyle/scope** line\_breaks

**Config/autostyle/strict** true

**Config/Needs/website** tidyverse/tidytemplate

**NeedsCompilation** no

**Author** Kirill Müller [aut, cre] (ORCID:  
<<https://orcid.org/0000-0002-1416-3412>>)

**Maintainer** Kirill Müller <[kirill@cynkra.com](mailto:kirill@cynkra.com)>

**Repository** CRAN

**Date/Publication** 2025-08-26 16:20:02 UTC

## Contents

rprojroot-package . . . . .	2
criteria . . . . .	3
find_root . . . . .	4
find_root_file . . . . .	5
root_criterion . . . . .	6

<b>Index</b>	<b>10</b>
--------------	-----------

---

rprojroot-package      *rprojroot: Finding Files in Project Subdirectories*

---

## Description

Robust, reliable and flexible paths to files below a project root. The 'root' of a project is defined as a directory that matches a certain criterion, e.g., it contains a certain regular file.

## Details

See the "Value" section in [root\\_criterion\(\)](#) for documentation of root criterion objects, and [criteria](#) for useful predefined root criteria.

## Author(s)

**Maintainer:** Kirill Müller <kirill@cynkra.com> ([ORCID](#))

## See Also

Useful links:

- <https://rprojroot.r-lib.org/>
- <https://github.com/r-lib/rprojroot>
- Report bugs at <https://github.com/r-lib/rprojroot/issues>

## Examples

```
criteria
## Not run:
is_r_package$find_file("NAMESPACE")
root_fun <- is_r_package$make_fix_file()
root_fun("NAMESPACE")

## End(Not run)
```

---

criteria	<i>Prespecified criteria</i>
----------	------------------------------

---

**Description**

This is a collection of commonly used root criteria.

**Usage**

criteria

is\_rstudio\_project

is\_vscode\_project

is\_r\_package

is\_remake\_project

is\_drake\_project

is\_targets\_project

is\_pkgdown\_project

is\_renv\_project

is\_projectile\_project

is\_quarto\_project

is\_git\_root

is\_svn\_root

is\_vcs\_root

is\_testthat

from\_wd

**Details**

is\_rstudio\_project looks for a file with extension .Rproj.

is\_vscode\_project looks for a .vscode/settings.json file.

is\_r\_package looks for a DESCRIPTION file.

is\_remake\_project looks for a remake.yml file.

is\_drake\_project looks for a .drake directory.

is\_targets\_project looks for a \_targets.R file.

is\_pkgdown\_project looks for a \_pkgdown.yml, \_pkgdown.yaml, pkgdown/\_pkgdown.yml and/or inst/\_pkgdown.yml file.

is\_renv\_project looks for an renv.lock file.

is\_projectile\_project looks for a .projectile file.

is\_quarto\_project looks for a \_quarto.yml file.

is\_git\_root looks for a .git directory.

is\_svn\_root looks for a .svn directory.

is\_vcs\_root looks for the root of a version control system, currently only Git and SVN are supported.

is\_testthat looks for the testthat directory, works when developing, testing, and checking a package.

from\_wd uses the current working directory.

---

 find\_root

*Find the root of a directory hierarchy*


---

## Description

A *root* is defined as a directory that contains a regular file whose name matches a given pattern and which optionally contains a given text. The search for a root starts at a given directory (the working directory by default), and proceeds up the directory hierarchy.

get\_root\_desc() returns the description of the criterion for a root path. This is especially useful for composite root criteria created with `|.root_criterion()`.

## Usage

```
find_root(criterion, path = ".")
```

```
get_root_desc(criterion, path)
```

## Arguments

criterion	[root_criterion] A criterion, one of the predefined <a href="#">criteria</a> or created by <code>root_criterion()</code> . Will be coerced using <code>as_root_criterion()</code> .
path	[character(1)] The start directory.

## Details

Starting from the working directory, the `find_root()` function searches for the root. If a root is found, the `...` arguments are used to construct a path; thus, if no extra arguments are given, the root is returned. If no root is found, an error is thrown.

## Value

The normalized path of the root as specified by the search criterion. Throws an error if no root is found

## See Also

`utils::glob2rx()` `file.path()`

## Examples

```
## Not run:
find_root(has_file_pattern(
  pattern = glob2rx("DESCRIPTION"),
  contents = "^Package: "
))

## End(Not run)
```

---

find\_root\_file

*File paths relative to the root of a directory hierarchy*

---

## Description

`find_root_file()` is a wrapper around `find_root()` that appends an arbitrary number of path components to the root using `base::file.path()`.

## Usage

```
find_root_file(..., criterion, path = ".")
```

```
find_rstudio_root_file(..., path = ".")
```

```
find_package_root_file(..., path = ".")
```

```
find_remake_root_file(..., path = ".")
```

```
find_testthat_root_file(..., path = ".")
```

**Arguments**

...	[character] Further path components passed to <code>file.path()</code> . All arguments must be the same length or length one.
criterion	[root_criterion] A criterion, one of the predefined <code>criteria</code> or created by <code>root_criterion()</code> . Will be coerced using <code>as_root_criterion()</code> .
path	[character(1)] The start directory.

**Details**

This function operates on the notion of relative paths. The ... argument is expected to contain a path relative to the root. If the first path component passed to ... is already an absolute path, the criterion and path arguments are ignored, and ... is forwarded to `file.path()`.

**Value**

The normalized path of the root as specified by the search criteria, with the additional path components appended. Throws an error if no root is found.

**See Also**

`find_root()` `utils::glob2rx()` `base::file.path()`

**Examples**

```
## Not run:
find_package_root_file("tests", "testthat.R")
has_file("DESCRIPTION", "^Package: ")$find_file
has_file("DESCRIPTION", "^Package: ")$make_fix_file(".")

## End(Not run)
```

---

root_criterion	<i>Is a directory the project root?</i>
----------------	---

---

**Description**

Objects of the `root_criterion` class decide if a given directory is a project root.

**Usage**

```

root_criterion(testfun, desc, subdir = NULL)

is_root_criterion(x)

as_root_criterion(x)

## S3 method for class 'character'
as_root_criterion(x)

## S3 method for class 'root_criterion'
as_root_criterion(x)

## S3 method for class 'root_criterion'
x | y

has_file(filepath, contents = NULL, n = -1L, fixed = FALSE)

has_dir(filepath)

has_file_pattern(pattern, contents = NULL, n = -1L, fixed = FALSE)

has_basename(basename, subdir = NULL)

```

**Arguments**

testfun	[function list(function)] A function with one parameter that returns TRUE if the directory specified by this parameter is the project root, and FALSE otherwise. Can also be a list of such functions.
desc	[character] A textual description of the test criterion, of the same length as testfun.
subdir	[character] If given, the criterion will also be tested in the subdirectories defined by this argument, in the order given. The first existing directory will be used as a starting point. This is used for the <a href="#">is_testthat</a> criterion that needs to <i>descend</i> into tests/testthat if starting at the package root, but stay inside tests/testthat if called from a testthat test.
x	[object] An object.
y	[object] An object.
filepath	[character(1)] File path (can contain directories).
contents, fixed	[character(1)] If contents is NULL (the default), file contents are not checked. Otherwise, contents is a regular expression (if fixed is FALSE) or a search string (if fixed is TRUE), and file contents are checked matching lines.

n	[integerish(1)] Maximum number of lines to read to check file contents.
pattern	[character(1)] Regular expression to match the file name against.
basename	[character(1)] The required name of the root directory.

## Details

Construct criteria using `root_criterion` in a very general fashion by specifying a function with a path argument, and a description.

The `as_root_criterion()` function accepts objects of class `root_criterion`, and character values; the latter will be converted to criteria using `has_file`.

Root criteria can be combined with the `|` operator. The result is a composite root criterion that requires either of the original criteria to match.

The `has_file()` function constructs a criterion that checks for the existence of a specific file (which itself can be in a subdirectory of the root) with specific contents.

The `has_dir()` function constructs a criterion that checks for the existence of a specific directory.

The `has_file_pattern()` function constructs a criterion that checks for the existence of a file that matches a pattern, with specific contents.

The `has_basename()` function constructs a criterion that checks if the `base::basename()` of the root directory has a specific name, with support for case-insensitive file systems.

## Value

An S3 object of class `root_criterion` with the following members:

`testfun` The `testfun` argument

`desc` The `desc` argument

`subdir` The `subdir` argument

`find_file` A function with `...` and `path` arguments that returns a path relative to the root, as specified by this criterion. The optional `path` argument specifies the starting directory, which defaults to `."`. The function forwards to `find_root_file()`, which passes `...` directly to `file.path()` if the first argument is an absolute path.

`make_fix_file` A function with a `path` argument that returns a function that finds paths relative to the root. For a criterion `cr`, the result of `cr$make_fix_file("")(..)` is identical to `cr$find_file(..)`. The function created by `make_fix_file()` can be saved to a variable to be more independent of the current working directory.

## Examples

```
root_criterion(function(path) file.exists(file.path(path, "somefile")), "has somefile")
has_file("DESCRIPTION")
is_r_package
## Not run:
is_r_package$find_file
```

```
is_r_package$make_fix_file(".")
```

```
## End(Not run)
```

# Index

- \* **datasets**
  - criteria, 3
- as\_root\_criterion (root\_criterion), 6
- as\_root\_criterion(), 4, 6
- base::basename(), 8
- base::file.path(), 5, 6
- criteria, 2, 3, 4, 6
- file.path(), 5, 6
- find\_package\_root\_file (find\_root\_file), 5
- find\_remake\_root\_file (find\_root\_file), 5
- find\_root, 4
- find\_root(), 5, 6
- find\_root\_file, 5
- find\_root\_file(), 8
- find\_rstudio\_root\_file (find\_root\_file), 5
- find\_testthat\_root\_file (find\_root\_file), 5
- from\_wd (criteria), 3
- get\_root\_desc (find\_root), 4
- has\_basename (root\_criterion), 6
- has\_dir (root\_criterion), 6
- has\_file (root\_criterion), 6
- has\_file\_pattern (root\_criterion), 6
- is\_drake\_project (criteria), 3
- is\_git\_root (criteria), 3
- is\_pkgdown\_project (criteria), 3
- is\_projectile\_project (criteria), 3
- is\_quarto\_project (criteria), 3
- is\_r\_package (criteria), 3
- is\_remake\_project (criteria), 3
- is\_renv\_project (criteria), 3
- is\_root\_criterion (root\_criterion), 6
- is\_rstudio\_project (criteria), 3
- is\_svn\_root (criteria), 3
- is\_targets\_project (criteria), 3
- is\_testthat, 7
- is\_testthat (criteria), 3
- is\_vcs\_root (criteria), 3
- is\_vscode\_project (criteria), 3
- root\_criterion, 6
- root\_criterion(), 2, 4, 6
- rprojroot (rprojroot-package), 2
- rprojroot-package, 2
- utils::glob2rx(), 5, 6