Package: rprojroot (via r-universe)

May 23, 2026

Title Finding Files in Project Subdirectories

Version 2.1.1.9006

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

Roxygen list(markdown = TRUE, load = ``source'')

RoxygenNote 7.3.3.9000

Config/autostyle/scope line_breaks

Config/autostyle/strict true

Config/Needs/website tidyverse/tidytemplate

Repository https://cynkra.r-universe.dev

Date/Publication 2026-05-23 18:04:23 UTC

RemoteUrl https://github.com/r-lib/rprojroot

RemoteRef HEAD

RemoteSha c8c23e8126f149b1fd105922b444ab9be1b0b440

Contents

`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)

Authors:

- 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` *Prespecified criteria*

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 criteria or created by `root_criterion()`. Will be coerced using `as_root_criterion()`.
`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 `file.path()`. All arguments must be the same length or length one.
`criterion`	`[root_criterion]` A criterion, one of the predefined criteria or created by `root_criterion()`. Will be coerced using `as_root_criterion()`.
`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` *Is a directory the project root?*

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 is_testthat criterion that needs to *descend* 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