

Package: csdocs (via r-universe)

October 12, 2026

Title Markdown Reference and Site Documentation for R Packages

Version 2026.10.12

Description Renders the Rd documentation of an R package as plain 'Markdown': one file per topic, an index, and one file with the full reference. The output reads well on 'GitHub' and works as context for a large language model. It reads an installed package with `tools::Rd_db()`, or a source package from its `man/*.Rd` files without installation.

License MIT + file LICENSE

URL <https://niphr.github.io/csdocs/>, <https://github.com/niphr/csdocs>

BugReports <https://github.com/niphr/csdocs/issues>

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.1.0)

Imports purrr (>= 1.0.0), glue (>= 1.6.0), cli (>= 3.0.0), fs (>= 1.6.0)

Suggests testthat (>= 3.0.0), withr, roxygen2 (>= 7.0.0), rmarkdown, knitr, yaml

Config/testthat/edition 3

Config/Needs/website niphr/cstemplate

Config/roxygen2/version 8.1.1

Config/pak/sysreqs cmake make libuv1-dev

Repository <https://niphr.r-universe.dev>

Date/Publication 2026-10-12 13:10:48 UTC

RemoteUrl <https://github.com/niphr/csdocs>

RemoteRef HEAD

RemoteSha 0bf1e1d4a768071f87dfc046cbd96ef0a3b9ceaf

Contents

build_reference	2
build_site	3
get_documentation	5
Index	6

build_reference	<i>Generate Markdown reference documentation for a package</i>
-----------------	--

Description

Reads the Rd documentation of a package - either installed, or a local development package that has not been installed - renders it as Markdown, and writes one .md file per function into output_dir. Also produces a README.md (index table of all functions) and a _full_reference.md (every function in one file, meant to be injected as LLM context, e.g. via @docs/reference/_full_reference.md in Claude Code).

Usage

```
build_reference(  
  package = ".",  
  output_dir = NULL,  
  overwrite = TRUE,  
  include_private = TRUE  
)
```

Arguments

package	Name of an installed package, or a path to a local package source directory. Defaults to "." - the common case of running build_reference() from within the package you're documenting.
output_dir	Target directory for the generated .md files. Created if it does not exist. Defaults to "md-docs", but only when package resolves to a local source directory - for an installed package (e.g. "dplyr") there's no similarly obvious default location, so output_dir must be given explicitly.
overwrite	If TRUE (the default), any .md files already in output_dir are removed before writing the new set - this also clears stale files left behind by topics that were since renamed or removed from the package. If FALSE, aborts instead when output_dir already contains .md files, without touching them.
include_private	If TRUE (the default), documented topics that aren't part of the package's public API (i.e. not in its NAMESPACE exports - typically internal helpers documented with @keywords internal rather than @export) are included and marked _(private)_ next to their heading, in both the topic file and the index. If FALSE, such topics are left out of the output entirely. Datasets and other

non-callable topics are never treated as private, even though they also aren't in NAMESPACE's exports - see `.compute_is_private()`. When export status can't be determined at all (a dev package that has never run `devtools::document()`, so it has no NAMESPACE file yet), every topic is treated as public.

Details

package accepts two kinds of input:

- the name of an installed package, e.g. "dplyr" - Rd data is read via `tools::Rd_db()`;
- a path to a local package source directory, e.g. "." or "../myotherpkg" (detected by the presence of a DESCRIPTION file at that path) - Rd data is read directly from `man/*.Rd`, falling back to a live `roxygen2::parse_package()` parse if `man/` is missing or empty. This mode does **not** require the package to be installed.

Value

Invisibly, the path to `output_dir`.

Examples

```
## Not run:
# Installed package:
csdocs::build_reference("dplyr", output_dir = "docs/reference")

# Development package, not installed - "." and "md-docs" are both
# defaults, so this is equivalent to just build_reference():
csdocs::build_reference(".", output_dir = "md-docs")

# Public API only, no internal helpers:
csdocs::build_reference("dplyr", output_dir = "docs/reference", include_private = FALSE)

## End(Not run)
```

build_site

Build a Markdown site for a source package

Description

Writes the reference, every R Markdown vignette and a front page as Markdown files that read as linked pages on github.com. The output has this layout:

- `README.md`: the front page. It shows the package name, the version, the source commit when known, a list of articles and a link to the reference.
- `reference/`: the output of `build_reference()`.
- `articles/`: one `<name>.md` file for each vignettes/*.Rmd file, with its figures in a folder beside it.

Usage

```
build_site(package = ".", output_dir, overwrite = TRUE)
```

Arguments

package	Path to a package source directory, which MUST contain a DESCRIPTION file. An installed package name is an error, because an installed package has no vignette sources.
output_dir	Target directory. Created if it does not exist.
overwrite	If TRUE (the default), build_site() deletes the README.md file and the reference/ and articles/ folders in output_dir before it writes the new set. If FALSE, it aborts when output_dir contains any file or folder.

Details

build_site() copies the whole vignettes/ folder to articles/, so images and other assets stay beside the articles. It renders each .Rmd file there with rmarkdown::github_document(html_preview = FALSE), in a new environment whose parent is the global environment. Then it deletes the copied .Rmd files and any .Rmd.orig files.

The front page lists each article under the title field of its YAML header, or under its file name when the header has no title. The commit comes from the GITHUB_SHA environment variable, else from git rev-parse HEAD in package. The front page shows its first 7 characters. When neither source gives a commit, the front page omits it.

Install the package before you call build_site() when a vignette calls library() on it. The vignettes run against the installed version, and the reference reads the source.

build_site() does not render Quarto (.qmd) vignettes. It copies them unrendered and gives a warning that names each one.

Value

Invisibly, the path to output_dir.

Examples

```
## Not run:
csdocs::build_site(".", output_dir = "docs-build")

## End(Not run)
```

get_documentation	<i>Get a package's full reference documentation as a single string</i>
-------------------	--

Description

Returns the same Markdown content that `build_reference()` writes to `_full_reference.md` - every topic rendered and concatenated into one string - without writing anything to disk. Meant for programmatic consumption, e.g. an LLM agent that wants a package's complete reference injected directly into its context in one call.

Usage

```
get_documentation(package = ".", include_private = FALSE)
```

Arguments

package	As in <code>build_reference()</code> .
include_private	If TRUE, documented topics that aren't part of the package's public API are included too - see <code>build_reference()</code> . Defaults to FALSE here (unlike <code>build_reference()</code> 's TRUE), since the primary use case - handing a package's reference to an LLM agent as context - usually wants the public API only.

Value

A single string: the package's full reference documentation in Markdown.

Examples

```
## Not run:  
# Installed package:  
docs <- csdocs::get_documentation("dplyr")  
  
# Development package, not installed:  
docs <- csdocs::get_documentation(".")  
  
## End(Not run)
```

Index

`build_reference`, [2](#)
`build_reference()`, [3](#), [5](#)
`build_site`, [3](#)

`get_documentation`, [5](#)

`tools::Rd_db()`, [3](#)