

Package: pkgmd (via r-universe)

October 11, 2026

Title Generate Markdown Reference Documentation for R Packages

Version 0.1

Description Generates plain Markdown reference documentation for any R package, installed or a local development source tree, optimized for rendering in the 'GitHub'/'Gitea' browser UI and for use as LLM context (e.g. 'Claude Code'). Reads Rd documentation via `tools::Rd_db()` (installed packages) or directly from `man/*.Rd` (development packages, no installation required) and renders one file per topic plus a navigable index.

License MIT + file LICENSE

URL <https://github.com/peterczerner/pkgmd>

BugReports <https://github.com/peterczerner/pkgmd/issues>

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

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), dplyr, withr, roxygen2 (>= 7.0.0)

Config/testthat/edition 3

Config/pak/sysreqs cmake make libuv1-dev

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

Date/Publication 2026-07-29 19:39:47 UTC

RemoteUrl <https://github.com/peterczerner/pkgmd>

RemoteRef HEAD

RemoteSha 61e52b12087bc9a62f65215fd4641777e41ad3e0

Contents

build_reference	2
get_documentation	3
Index	5

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:
pkgmd::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():
pkgmd::build_reference(".", output_dir = "md-docs")

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

## 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 `build_reference()`.
`include_private` If TRUE, documented topics that aren't part of the package's public API are included too - see `build_reference()`. Defaults to FALSE here (unlike `build_reference()`'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 <- pkgmd::get_documentation("dplyr")  
  
# Development package, not installed:  
docs <- pkgmd::get_documentation(".")  
  
## End(Not run)
```

Index

`build_reference`, [2](#)
`build_reference()`, [3](#), [4](#)
`get_documentation`, [3](#)
`tools::Rd_db()`, [3](#)