Package: rdtools (via r-universe)

July 17, 2026

Title Efficient Manipulation of 'Rd' Files and Help Topics

Version 0.1.0.9000

Description Provides fast, cached lookup of help topics and aliases across installed, source, and in-development packages, plus efficient retrieval of parsed 'Rd' ('R' documentation) objects. Per-package indexes are built once and cached, making repeated retrieval cheap enough to call in a tight loop.

License MIT + file LICENSE

URL https://rdtools.r-lib.org, https://github.com/r-lib/rdtools

BugReports https://github.com/r-lib/rdtools/issues

Depends R (>= 4.0)

Imports stats, tools

Suggests testthat (>= 3.0.0), withr

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 8.0.0

Config/Needs/website tidyverse/tidytemplate

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

Date/Publication 2026-07-16 18:23:21 UTC

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

RemoteRef HEAD

RemoteSha 2ba6baa6643e91c87fdceeb545a33466101a5f9e

Contents

`pkg_cache_reset` *Reset cached package indexes*

Description

Clears the cached topic index, parsed Rd objects, and dependency search set for `package`, or for every package when `package` is `NULL`. This will generally be called automatically by roxygen2 for source packages. Installed package indexes are automatically reset when their namespace is unloaded.

Usage

```
pkg_cache_reset(package = NULL)
```

Arguments

`package`	A package name, or a path to the source directory of a package. Use `NULL`, the default, to reset every cached index.

Value

`NULL`, invisibly.

Examples

```
head(pkg_topics("stats"))
pkg_cache_reset("stats")

# Reset every cached index
pkg_cache_reset()
```

pkg_macros *Load a package's Rd macros*

Description

Loads the macros declared by the package's `RdMacros` field and `man/macros/` directory, layered over R's system macros. This reproduces the macro environment used when installing a package.

Usage

```
pkg_macros(package)
```

Arguments

`package`	A package name, or a path to the source directory of a package.

Value

An environment containing the Rd macro definitions.

Examples

```
macros <- pkg_macros("stats")
head(ls(macros))
```

pkg_search_attached *Packages searched for help by default*

Description

`pkg_search_attached()` returns the packages a bare `?topic` can see: every attached package, in search path order, followed by the base packages that are always available. This is the default search set for `topic_find()`.

`pkg_search_base()` returns the base packages that are always searched.

Usage

```
pkg_search_attached()

pkg_search_base()
```

Value

A character vector of package names.

Examples

```
pkg_search_attached()
pkg_search_base()
```

`pkg_search_deps` *Packages to search for topics in a package's dependencies*

Description

`pkg_search_deps()` returns the packages whose topics `package`'s documentation can link to: the packages declared in its `Depends`, `Imports`, and `Suggests` fields, followed by the base packages, which are always available. The result is cached alongside the package's topic index, so it remains valid until `pkg_cache_reset()` is called.

Usage

```
pkg_search_deps(package)
```

Arguments

`package`	A package name, or a path to the source directory of a package.

Value

A character vector of package names.

Examples

```
pkg_search_deps("stats")
```

`pkg_topics` *List the topics documented by a package*

Description

`pkg_topics()` returns a named character vector mapping every alias (i.e. everything you can type after ?) to the name of the Rd file (without extension) that documents it. It understands three kinds of package:

- **Installed** packages, read from their `help/aliases.rds` index.
- **In-development** packages loaded with `pkgload::load_all()`, indexed from the `\alias{}` commands in their source `man/` directory.
- **Source** packages, when `package` is a path to a package directory rather than a name.

Indexes remain cached until `pkg_cache_reset()` is called. Installed package indexes are also reset automatically when their namespace is unloaded.

For source packages, `\alias{}` extraction is line-based: any number of aliases may appear anywhere on a line, but an alias must open and close on the same line, and aliases produced by Rd macros are not seen.

Usage

```
pkg_topics(package)
```

Arguments

package	A package name, or a path to the source directory of a package.

Value

A named character vector mapping alias to Rd file name.

Examples

```
head(pkg_topics("stats"))
```

---

topic_exists *Is a topic documented?*

Description

Checks whether any of packages has an exact alias matching topic.

Usage

```
topic_exists(topic, packages = pkg_search_attached())
```

Arguments

topic	A single string naming an alias, matched exactly. Use topic_split() first if you need to handle qualified topics like "pkg::foo".
packages	A character vector of package names (and/or source package paths) to search, in order. Defaults to pkg_search_attached(). Unavailable packages are skipped.

Value

A single TRUE or FALSE.

Examples

```
topic_exists("rnorm", "stats")
topic_exists("rnorm", c("base", "stats"))
topic_exists("median")
topic_exists("not-a-topic", "stats")
```

`topic_find` *Find packages that document a topic*

Description

`topic_find()` looks `topic` up in each of `packages` in order and returns the first hit. `topic_find_all()` returns every hit. Lookups use the cached per-package indexes built by `pkg_topics()`, so scanning even a long search set is cheap.

Usage

```
topic_find(topic, packages = pkg_search_attached())

topic_find_all(topic, packages = pkg_search_attached())
```

Arguments

`topic`	A single string naming an alias, matched exactly. Use `topic_split()` first if you need to handle qualified topics like `"pkg::foo"`.
`packages`	A character vector of package names (and/or source package paths) to search, in order. Defaults to `pkg_search_attached()`. Unavailable packages are skipped.

Value

`topic_find()` returns `NULL` if the topic isn't found; otherwise a list with elements:

- `package`: the package that documents the topic.
- `file`: the name of the Rd file (without extension).

`topic_find_all()` returns a data frame with columns `package` and `file`, containing one row per hit (and no rows if the topic isn't found).

Examples

```
topic_find("rnorm")
topic_find("mean", c("stats", "base"))
topic_find_all("plot", c("graphics", "base"))
topic_find("no-such-topic")
```

topic_origin *Find the package where a topic's object originates*

Description

Determines which package supplies the object associated with a documented topic. This resolves re-exported functions and imported objects to the package where they originate. Results are cached alongside the package's topic index, so repeated lookups are cheap.

Usage

```
topic_origin(topic, package)
```

Arguments

topic	A single string naming a topic.
package	A package name.

Value

A single package name.

Examples

```
topic_origin("rnorm", "stats")
```

topic_qualifier *Find the package qualifier for a topic*

Description

Determines whether a topic needs a package qualifier when linking from the documentation of another package. The `from` package is checked first, followed by `packages`, and then the base packages. Re-exported objects are attributed to their original package. Results are cached alongside the `from` package's topic index, so repeated lookups are cheap.

Usage

```
topic_qualifier(topic, from, packages = character())
```

Arguments

topic	A single string naming an alias, matched exactly.
from	The name or source directory of the package you are linking from. If it documents `topic`, no qualifier is needed.
packages	A character vector of additional packages to search, typically the dependencies of `from` (as computed by `pkg_search_deps()`). Base packages are always included.

Value

- NULL if the topic isn't documented by from, packages, or the base packages.
- NA_character_ if the topic needs no qualifier because it's documented by from or a base package.
- Otherwise, the package name(s) that could qualify the topic: usually one, but more if the topic is ambiguous, in which case it's up to the caller to decide how to respond.

Examples

```
topic_qualifier("rnorm", "stats")
```

---

topic_rd *Get the parsed Rd for a topic*

Description

Retrieves the parsed Rd object documenting topic in package. For installed packages the topic is fetched lazily from the package's help database (no parsing needed); for source and in-development packages the Rd file is parsed with tools::parse_Rd(), with the package's Rd macros loaded. Results are cached per topic, so repeated access (e.g. roxygen2 inheriting several fields from one topic) only pays once.

Usage

```
topic_rd(topic, package)
```

Arguments

topic	A single string.
package	A package name, or a path to the source directory of a package.

Value

An Rd object: a recursive structure of class "Rd", as returned by tools::parse_Rd(). Returns NULL if the package or topic doesn't exist; use topic_exists() first if you need to distinguish that from other problems.

Examples

```
rd <- topic_rd("rnorm", "stats")
class(rd)
```

topic_rd_path *Get the path to the Rd file for a topic*

Description

Returns the path to the `.Rd` file that documents `topic`. Rd files only exist on disk for source and in-development packages; installed packages store their documentation in a binary database, so `topic_rd_path()` returns `NULL` for them. Use `topic_rd()` if you want the parsed contents regardless of where they are stored.

Usage

```
topic_rd_path(topic, package)
```

Arguments

`topic`	A single string.
`package`	A package name, or a path to the source directory of a package.

Value

The path to a `.Rd` file, or `NULL` if the package or topic doesn't exist, or if the package doesn't have Rd files on disk.

Examples

```
# Rd files only exist on disk for source packages, so make a minimal one
pkg <- tempfile()
dir.create(file.path(pkg, "man"), recursive = TRUE)
writeLines("Package: demo", file.path(pkg, "DESCRIPTION"))
writeLines(
  c("\\name{foo}", "\\alias{foo}", "\\title{Foo}", "\\description{Foo.}"),
  file.path(pkg, "man", "foo.Rd")
)

topic_rd_path("foo", pkg)
```

topic_split *Split a qualified topic into package and topic*

Description

Splits `"pkg::topic"` (or `"pkg:::topic"`) into its package and topic components. The prefix is only treated as a qualifier when it is a syntactically valid package name, so aliases that merely contain `::` (e.g. S7 method aliases like `"speak,foo::Dog-method"`) are left intact.

Usage

```
topic_split(topic)
```

Arguments

topic	A single string.

Value

A list with elements package (a string, or NULL if the topic is unqualified) and topic.

Examples

```
topic_split("stats::rnorm")
topic_split("rnorm")
topic_split("speak,foo::Dog-method")
```

Index