

Package ‘tinyrox’

August 5, 2026

Title Minimal R Documentation Generator

Version 0.4.1

Date 2026-08-04

Description A deterministic, dependency-free documentation generator for R packages. Generates valid Rd files and NAMESPACE from 'roxygen2'-style comments using only base R. Supports a strict subset of tags with no markdown parsing, no inference magic, and explicit-only behavior.

License GPL-3

URL <https://github.com/cornball-ai/tinyrox>

BugReports <https://github.com/cornball-ai/tinyrox/issues>

Encoding UTF-8

Imports utils

Suggests tinytest

NeedsCompilation no

Author Troy Hernandez [aut, cre] (ORCID:
<<https://orcid.org/0009-0005-4248-604X>>),
cornball.ai [cph]

Maintainer Troy Hernandez <troy@cornball.ai>

Repository CRAN

Date/Publication 2026-08-04 22:50:02 UTC

Contents

check_code_cran	2
check_cran	3
check_examples_cran	3
clean	4
document	5
extract_function_name	6
find_dontrun_examples	7

find_exports_without_examples	7
find_long_example_lines	8
parse_tags	8
Index	10

check_code_cran	<i>Check R Code for CRAN Issues</i>
-----------------	-------------------------------------

Description

Scans R files for common CRAN policy violations. Files are parsed and checked token-wise: comments and string literals are never flagged, print()/cat() are allowed inside print./format. S3 methods, and T/F shorthand is not confused with a local variable named T or F.

Usage

```
check_code_cran(path = ".")
```

Arguments

path Path to package root directory

Value

List with issues found

Examples

```
# Create a minimal package in tempdir
pkg <- file.path(tempdir(), "mypkg")
dir.create(file.path(pkg, "R"), recursive = TRUE, showWarnings = FALSE)
writeLines("Package: mypkg\nTitle: Test\nVersion: 0.1.0",
  file.path(pkg, "DESCRIPTION"))
writeLines("add <- function(x, y) x + y",
  file.path(pkg, "R", "add.R"))

check_code_cran(pkg)

# Clean up
unlink(pkg, recursive = TRUE)
```

check_cran	<i>Full CRAN Compliance Check</i>
------------	-----------------------------------

Description

Runs the CRAN compliance checks for package code and examples.

Usage

```
check_cran(path = ".")
```

Arguments

path	Path to package root directory
------	--------------------------------

Value

List with all issues

Examples

```
# Create a minimal package in tempdir
pkg <- file.path(tempdir(), "mypkg")
dir.create(file.path(pkg, "R"), recursive = TRUE, showWarnings = FALSE)
writeLines("Package: mypkg\nTitle: Test\nVersion: 0.1.0\nDescription: A test package.",
  file.path(pkg, "DESCRIPTION"))
writeLines("add <- function(x, y) x + y",
  file.path(pkg, "R", "add.R"))

check_cran(pkg)

# Clean up
unlink(pkg, recursive = TRUE)
```

check_examples_cran	<i>Check for Missing Examples</i>
---------------------	-----------------------------------

Description

Identifies exported functions that lack examples in their documentation.

Usage

```
check_examples_cran(path = ".")
```

Arguments

path Path to package root directory

Value

Character vector of function names missing examples

Examples

```
# Create a minimal package in tempdir
pkg <- file.path(tempdir(), "mypkg")
dir.create(file.path(pkg, "R"), recursive = TRUE, showWarnings = FALSE)
writeLines("Package: mypkg\nTitle: Test\nVersion: 0.1.0",
  file.path(pkg, "DESCRIPTION"))
writeLines("'#' Add numbers\n#' @export\nadd <- function(x, y) x + y",
  file.path(pkg, "R", "add.R"))

check_examples_cran(pkg)

# Clean up
unlink(pkg, recursive = TRUE)
```

clean

Clean Generated Files

Description

Removes all Rd files from man/ directory.

Usage

```
clean(path = ".", namespace = FALSE)
```

Arguments

path Path to package root directory.

namespace Also remove NAMESPACE? Default FALSE.

Value

No return value, called for side effects.

Examples

```
# Create a minimal package in tempdir
pkg <- file.path(tempdir(), "mypkg")
dir.create(file.path(pkg, "man"), recursive = TRUE, showWarnings = FALSE)
writeLines("Package: mypkg\nTitle: Test\nVersion: 0.1.0",
  file.path(pkg, "DESCRIPTION"))
writeLines("placeholder", file.path(pkg, "man", "test.Rd"))

clean(pkg)

# Clean up
unlink(pkg, recursive = TRUE)
```

document

Generate Documentation for an R Package

Description

Main function for tinyrox. Parses R source files for documentation comments and generates Rd files and NAMESPACE.

Usage

```
document(
  path = ".",
  namespace = c("overwrite", "append", "none"),
  cran_check = TRUE,
  silent = FALSE,
  prune_rd = TRUE
)
```

Arguments

path	Path to package root directory. Default is current directory.
namespace	How to handle NAMESPACE generation. One of "overwrite" Fully regenerate NAMESPACE (default) "append" Insert between ## tinyrox start/end markers "none" Don't modify NAMESPACE
cran_check	Run CRAN compliance checks (code issues and missing examples) and undocumented-parameter warnings. Default TRUE.
silent	Operate less verbose without messages. Default FALSE.
prune_rd	Remove stale Rd files for topics no longer documented (e.g. after a rename or deletion)? Only files tinyrox generated are removed; hand-written Rd is never touched. Default TRUE. Set FALSE to leave existing Rd in place.

Value

Invisibly returns a list with: - `rd_files`: character vector of generated Rd file paths - `pruned`: character vector of stale Rd file paths removed - `namespace`: path to NAMESPACE file (or NULL if `mode="none"`)

Examples

```
# Create a minimal package in tempdir
pkg <- file.path(tempdir(), "mypkg")
dir.create(file.path(pkg, "R"), recursive = TRUE, showWarnings = FALSE)
writeLines("Package: mypkg\nTitle: Test\nVersion: 0.1.0",
  file.path(pkg, "DESCRIPTION"))
writeLines(c(
  "'#' Add two numbers",
  "'#' @param x A number",
  "'#' @param y A number",
  "'#' @export",
  "add <- function(x, y) x + y",
  file.path(pkg, "R", "add.R"))

# Document the package
document(pkg, cran_check = FALSE)

# Clean up
unlink(pkg, recursive = TRUE)
```

`extract_function_name` *Extract Function Name from Definition Line*

Description

Extract Function Name from Definition Line

Usage

```
extract_function_name(line)
```

Arguments

`line` Code line potentially containing function definition

Value

Function name or NULL

`find_dontrun_examples` *Find Exported Functions Using dontrun in Examples*

Description

Parses R file content to find @export tags with \dontrun in @examples.

Usage

```
find_dontrun_examples(lines)
```

Arguments

lines Character vector of file lines

Value

Character vector of function names using dontrun

`find_exports_without_examples`
Find Exported Functions Without Examples

Description

Parses R file content to find @export tags without @examples.

Usage

```
find_exports_without_examples(lines)
```

Arguments

lines Character vector of file lines

Value

Character vector of function names missing examples

find_long_example_lines

Find Example Lines Exceeding 100 Characters

Description

Scans @examples blocks for lines that will be truncated in the PDF manual.

Usage

```
find_long_example_lines(lines, filename)
```

Arguments

lines	Character vector of file lines
filename	Filename for reporting

Value

Character vector of warnings (file:line format)

parse_tags

Parse Tags from Documentation Lines

Description

Parse Tags from Documentation Lines

Usage

```
parse_tags(lines, object_name, file = NULL, line_num = NULL)
```

Arguments

lines	Character vector of documentation lines (without #').
object_name	Name of the documented object.
file	Source file (for error messages).
line_num	Starting line number (for error messages).

Value

A list with parsed tag values.

Examples

```
lines <- c("Title Here", "", "Description text.", "", "@param x A number.",  
          "@return The number.", "@export")  
tags <- parse_tags(lines, "my_function")  
tags$title  
tags$params
```

Index

check_code_cran, [2](#)
check_cran, [3](#)
check_examples_cran, [3](#)
clean, [4](#)

document, [5](#)

extract_function_name, [6](#)

find_dontrun_examples, [7](#)
find_exports_without_examples, [7](#)
find_long_example_lines, [8](#)

parse_tags, [8](#)