Package: tinyrox (via r-universe)

May 24, 2026

Title Minimal R Documentation Generator

Version 0.3.3

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

Suggests tinytest

Repository https://cornball-ai.r-universe.dev

Date/Publication 2026-05-19 18:53:12 UTC

RemoteUrl https://github.com/cornball-ai/tinyrox

RemoteRef HEAD

RemoteSha 27c1587a1b0cb7d68bdeaffd8265375b6ea54c6a

Contents

`check_code_cran` *Check R Code for CRAN Issues*

Description

Scans R files for common CRAN policy violations.

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_code_lines *Check Code Lines for Issues*

Description

Check Code Lines for Issues

Usage

```
check_code_lines(lines, filename)
```

Arguments

lines	Character vector of code lines
filename	Filename for reporting

Value

List of issues

check_cran *Full CRAN Compliance Check*

Description

Runs all CRAN compliance checks (DESCRIPTION + code).

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_description_cran

Check DESCRIPTION for CRAN Compliance

Description

Validates Title and Description fields for common CRAN issues: unquoted package names, missing web service links, etc.

Usage

```
check_description_cran(path = ".", fix = FALSE)
```

Arguments

path	Path to package root directory
fix	If TRUE, return fixed text. If FALSE, just warn.

Value

List with validation results and optionally fixed text

Examples

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

check_description_cran(pkg)

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

check_examples_cran *Check for Missing Examples*

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)
```

check_webservice_links *Check for Missing Web Service Links*

Description

Checks if packages that typically use web services have corresponding URLs in the Description.

Usage

```
check_webservice_links(description, packages)
```

Arguments

description	Description text
packages	Package names from dependencies

Value

Named list of packages missing links (name = URL)

---

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)
```

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 (DESCRIPTION quoting, web service links, code issues, missing examples). Default TRUE.

Value

Invisibly returns a list with: - rd_files: character vector of generated Rd file paths - 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)
```

escape_regex *Escape Regex Special Characters*

Description

Escape Regex Special Characters

Usage

```
escape_regex(x)
```

Arguments

x	String to escape

Value

Escaped string safe for use in regex

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)

find_unquoted_names *Find Unquoted Names in Text*

Description

Finds package/software names that appear without single quotes.

Usage

```
find_unquoted_names(text, names)
```

Arguments

text	Text to search
names	Names to look for

Value

Character vector of unquoted names found

fix_description_cran *Fix DESCRIPTION File*

Description

Automatically fixes common CRAN issues in DESCRIPTION.

Usage

```
fix_description_cran(path = ".", backup = TRUE)
```

Arguments

path	Path to package root directory
backup	Create backup file? Default TRUE.

Value

Invisibly returns TRUE if changes were made

Examples

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

fix_description_cran(pkg, backup = FALSE)

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

get_dependency_packages *Extract Package Names from Dependencies*

Description

Parses Imports, Suggests, Depends, and Enhances fields.

Usage

```
get_dependency_packages(desc)
```

Arguments

desc	DESCRIPTION matrix from read.dcf()

Value

Character vector of package names

---

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
```

quote_names_in_text *Quote Names in Text*

Description

Wraps unquoted package/software names in single quotes.

Usage

```
quote_names_in_text(text, names)
```

Arguments

text	Text to modify
names	Names to quote

Value

Modified text with names quoted

Index