Package 'quartify'

May 9, 2026

Type Package

Title Convert R Scripts to 'Quarto' Markdown Documents

Version 1.1.1

Description Converts R scripts (.R) into 'Quarto' markdown documents (.qmd) with automatic formatting. Recognizes 'RStudio' code sections, preserves comments as narrative text, extracts metadata from special comments, and provides both programmatic functions and an interactive 'RStudio' add-in for easy conversion.

Language en-US

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

URL https://ddotta.github.io/quartify/,
https://github.com/ddotta/quartify

BugReports https://github.com/ddotta/quartify/issues

Imports rstudioapi, cli, shiny, miniUI, later, base64enc, shinyFiles, quarto, utils, styler, lintr

Suggests testthat (>= 3.0.0), dplyr, shinyalert

Config/testthat/edition 3

Config/Needs/website pkgdown

NeedsCompilation no

Author Damien Dotta [aut, cre, cph]

Maintainer Damien Dotta <damien.dotta@live.fr>

Repository CRAN

Date/Publication 2026-01-21 18:50:02 UTC

Contents

`install_quartify_snippets`
Install quartify RStudio Snippets

Description

Installs useful RStudio snippets for working with quartify. These snippets help you quickly insert common structures when writing R scripts that will be converted to Quarto documents.

Usage

```
install_quartify_snippets(
  backup = TRUE,
  path = file.path(tempdir(), "r.snippets")
)
```

Arguments

`backup`	Logical. If TRUE (default), creates a backup of your existing snippets file before modifying it.
`path`	Character. Path for the snippets file. Defaults to writing in ‘tempdir()‘ to comply with CRAN policies. Provide an explicit path when you intentionally want to install into your RStudio snippets directory.

Details

The following snippets are installed:

- **header**: Insert a standard R script header with Title, Author, Date, and Description - **callout**: Insert a Quarto callout structure - **mermaid**: Insert a Mermaid diagram chunk - **tabset**: Insert a tabset structure

By default the snippets are written to a temporary file (inside ‘tempdir()‘) to comply with CRAN’s restriction on writing to the user’s filespace during examples, vignettes, and tests. To install in your actual RStudio snippets file, provide the explicit path via ‘path‘, for example: - Windows: ‘ - Mac/Linux: ‘~/.config/rstudio/snippets/r.snippets‘

If you already have custom snippets, this function will append the quartify snippets to your existing file. If quartify snippets were previously installed, they will be automatically removed and replaced with the new version.

The function will automatically open the snippets file in RStudio if available. Simply save the file (Ctrl+S / Cmd+S) to reload the snippets immediately without restarting RStudio. Then type the snippet name (e.g., 'header') and press Tab to insert the template.

Value

Invisibly returns the path to the snippets file.

Examples

```
if (interactive()) {
  # Install quartify snippets to RStudio
  install_quartify_snippets()

  # Install without backup
  install_quartify_snippets(backup = FALSE)
}

# For testing: install to temp directory
temp_snippets <- file.path(tempdir(), "r.snippets")
install_quartify_snippets(path = temp_snippets)
```

---

quartify_app *Launch Quartify Standalone Application*

Description

Standalone Shiny application for converting R scripts to Quarto markdown documents. Works in any R environment (RStudio, Positron, VS Code, etc.) without requiring the RStudio API.

Usage

```
quartify_app(launch.browser = TRUE, port = NULL)
```

Arguments

launch.browser	Logical, whether to launch browser (default: TRUE)
port	Integer, port number for the application (default: NULL for random port)

Value

No return value, called for side effects (launches a Shiny application).

Examples

```
if (interactive()) {
  # Launch the Shiny app in browser (works in any IDE)
  quartify_app()

  # Use in Positron or VS Code
  library(quartify)
  quartify_app()

  # Specify a port
  quartify_app(port = 3838)
}
```

`quartify_app_web` *Launch Quartify Web Application*

Description

Web-deployable Shiny application with file upload/download capabilities for converting R scripts to Quarto markdown documents. Suitable for deployment on Shiny Server, ShinyApps.io, or other web hosting platforms.

Usage

```
quartify_app_web(launch.browser = TRUE, port = NULL)
```

Arguments

`launch.browser`	Logical, whether to launch browser (default: TRUE)
`port`	Integer, port number for the application (default: NULL for random port)

Value

No return value, called for side effects (launches a Shiny application).

Examples

```
if (interactive()) {
  quartify_app_web()
}
```

`rtoqmd` *Convert R Script to Quarto Markdown*

Description

This function converts an R script to Quarto markdown format (.qmd), enabling you to leverage all modern Quarto features. Unlike `knitr::spin()` which generates R Markdown (.Rmd), `rtoqmd()` creates Quarto documents with access to advanced publishing capabilities, modern themes, native callouts, Mermaid diagrams, and the full Quarto ecosystem.

Usage

```
rtoqmd(
  input_file,
  output_file = NULL,
  title = "My title",
  author = "Your name",
  format = "html",
  theme = NULL,
  render_html = TRUE,
  output_html_file = NULL,
  open_html = FALSE,
  code_fold = FALSE,
  number_sections = TRUE,
  lang = "en",
  show_source_lines = TRUE,
  use_styler = FALSE,
  use_lintr = FALSE,
  apply_styler = FALSE
)
```

Arguments

`input_file`	Path to the input R script file
`output_file`	Path to the output Quarto markdown file (optional, defaults to same name with .qmd extension)
`title`	Title for the Quarto document (default: "My title"). Can be overridden by `# Title :` or `# Titre :` in the script
`author`	Author name (default: "Your name"). Can be overridden by `# Author :` or `# Auteur :` in the script
`format`	Output format - always "html" (parameter kept for backward compatibility)
`theme`	Quarto theme for HTML output (default: NULL uses Quarto's default). See https://quarto.org/docs/output-formats/html-themes.html for available themes (e.g., "cosmo", "flatly", "darkly", "solar", "united")
`render_html`	Logical, whether to render the .qmd file to HTML after creation (default: TRUE)

`output_html_file`
: Path to the output HTML file (optional, defaults to same name as .qmd file with .html extension)

`open_html`
: Logical, whether to open the HTML file in browser after rendering (default: FALSE, only used if render_html = TRUE)

`code_fold`
: Logical, whether to fold code blocks in HTML output (default: FALSE)

`number_sections`
: Logical, whether to number sections automatically in the output (default: TRUE)

`lang`
: Language for interface elements like table of contents title - "en" or "fr" (default: "en")

`show_source_lines`
: Logical, whether to add comments indicating original line numbers from the source R script at the beginning of each code chunk (default: TRUE). This helps maintain traceability between the documentation and the source code.

`use_styler`
: Logical, whether to apply styler code formatting and show differences in tabsets (default: FALSE). Requires the styler package to be installed.

`use_lintr`
: Logical, whether to run lintr code quality checks and display issues in tabsets (default: FALSE). Requires the lintr package to be installed.

`apply_styler`
: Logical, whether to apply styler formatting directly to the source R script file (default: FALSE). If TRUE, the input file will be modified with styled code. Requires use_styler = TRUE to take effect.

Details

It recognizes RStudio code sections with different levels: - ## Title #### creates a level 2 header - ### Title ==== creates a level 3 header - #### Title —- creates a level 4 header Regular comments are converted to plain text. Code blocks are wrapped in standard R code chunks. The YAML header includes `execute: eval: false` and `execute: echo: true` options for static documentation purposes, and `embed-resources: true` to create self-contained HTML files. See https://quarto.org/docs/output-formats/html-basics.html#self-contained.

Value

Invisibly returns NULL. Creates a .qmd file and optionally renders it to HTML.

Metadata Detection

The function automatically extracts metadata from special comment lines in your R script:

- **Title**: Use `# Title : Your Title` or `# Titre : Votre Titre`
- **Author**: Use `# Author : Your Name` or `# Auteur : Votre Nom`
- **Date**: Use `# Date : YYYY-MM-DD`
- **Description**: Use `# Description : Your description` (also accepts `# Purpose` or `# Objectif`)

If metadata is found in the script, it will override the corresponding function parameters. These metadata lines are removed from the document body and only appear in the YAML header.

The Description field supports multi-line content. Continuation lines should start with # followed by spaces and the text. The description ends at an empty line or a line without #.

Hidden Comments

Comments that start with # immediately followed by a non-space character (e.g., `#NOTE:`, `#TODO:`, `#DEBUG`) are completely ignored during conversion and will not appear in the Quarto output. This allows you to include private notes, debugging comments, or development annotations in your R scripts that won't be visible in the rendered documentation.

Only comments with a space after # (e.g., `# This is a comment`) are converted to text in the output.

Callouts

The function converts special comment patterns into Quarto callouts. Callouts are special blocks that highlight important information. Supported callout types: `note`, `tip`, `warning`, `caution`, `important`.

Syntax:

- **With title**: `# callout-tip - Your Title`
- **Without title**: `# callout-tip`

All subsequent comment lines become the callout content until an empty line or code is encountered.

Example in R script:

```
# callout-note - Important Note
# This is the content of the note.
# It can span multiple lines.

x <- 1
```

Becomes in Quarto:

```
::: {.callout-note title="Important Note"}
This is the content of the note.
It can span multiple lines.
:::
```

Mermaid Diagrams

The function supports Mermaid diagrams for flowcharts, sequence diagrams, and visualizations. Mermaid chunks start with a special comment, followed by options and diagram content. Options use hash-pipe syntax and are converted to percent-pipe in the Quarto output. Diagram content should not start with hash symbols. The chunk ends at a blank line or comment. Supported types: flowchart, sequence, class, state, etc. See example file in inst/examples/example_mermaid.R.

Tabsets

Create tabbed content panels for interactive navigation between related content. Use hash tabset to start a tabset container, then define individual tabs with hash tab - Title. Each tab can contain text, code, and other content. The tabset closes automatically when a new section starts. Example: hash tabset, hash tab - Plot A, code or text content, hash tab - Plot B, more content.

Roxygen2 Documentation

The function automatically detects and formats roxygen2 documentation blocks (starting with `#'`) into structured callouts that resemble pkgdown reference pages. The formatted documentation includes:

- **Title**: Extracted from `@title` tag or first roxygen comment line
- **Description**: From `@description` tag or initial paragraph
- **Usage**: Function signature with parameters
- **Arguments**: Each parameter from `@param` tags, formatted with parameter name in bold
- **Value**: Return value description from `@return` tag
- **Details**: Additional details from `@details` tag
- **Examples**: Code examples from `@examples` tag, displayed in R code blocks

LaTeX-style formatting is automatically converted to Markdown: `\href{url}{text}` becomes `[text](url)`, `\code{text}` becomes `` `text` ``, `\strong{text}` becomes `**text**`, and `\emph{text}` becomes `*text*`. Section headers within the callout use bold text instead of Markdown headers to avoid interfering with the document's table of contents. See example file in inst/examples/example_roxygen.R.

Examples

```
# Use example file included in package
example_file <- system.file("examples", "example.R", package = "quartify")

# Convert and render to HTML (output in temp directory)
output_qmd <- file.path(tempdir(), "output.qmd")
rtoqmd(example_file, output_qmd)

# Convert only, without rendering
rtoqmd(example_file, output_qmd, render_html = FALSE)

# Example with metadata in the R script:
# Create a script with metadata
script_with_metadata <- tempfile(fileext = ".R")
writeLines(c(
  "# Title : My Analysis",
  "# Author : Jane Doe",
  "# Date : 2025-11-28",
  "# Description : Analyze iris dataset",
  "",
  "library(dplyr)",
  "iris %>% head()"
), script_with_metadata)

# Convert - metadata will override function parameters
output_meta <- file.path(tempdir(), "output_with_metadata.qmd")
rtoqmd(script_with_metadata, output_meta)

# Example with code quality checks (requires styler and lintr packages)
script_with_style_issues <- tempfile(fileext = ".R")
writeLines(c(
```

```
  "# Script with style issues",
  "",
  "x = 3  # Should use <- instead of =",
  "y <- 2",
  "",
  "z <- 10"
), script_with_style_issues)

# Convert with styler formatting
output_styled <- file.path(tempdir(), "output_styled.qmd")
rtoqmd(script_with_style_issues, output_styled, use_styler = TRUE)

# Convert with both styler and lintr
output_quality <- file.path(tempdir(), "output_quality.qmd")
rtoqmd(script_with_style_issues, output_quality,
       use_styler = TRUE, use_lintr = TRUE)
```

`rtoqmd_addin` *Convert Active R Script to Quarto Markdown*

Description

RStudio add-in that converts the currently active R script in the editor to a Quarto markdown document. Uses a Shiny interface for parameter input.

Usage

```
rtoqmd_addin()
```

Value

No return value, called for side effects (launches an interactive Shiny gadget).

`rtoqmd_dir` *Convert All R Scripts in a Directory to Quarto Markdown*

Description

This function recursively searches for all R script files (.R) in a directory and its subdirectories, and converts each one to a Quarto markdown document (.qmd). The output files are created in the same directories as the input files.

Usage

```
rtoqmd_dir(
  dir_path,
  title_prefix = NULL,
  author = "Your name",
  format = "html",
  theme = NULL,
  render_html = FALSE,
  output_html_dir = NULL,
  open_html = TRUE,
  code_fold = FALSE,
  number_sections = TRUE,
  recursive = TRUE,
  pattern = "\\.R$",
  exclude_pattern = NULL,
  create_book = TRUE,
  book_title = "R Scripts Documentation",
  output_dir = NULL,
  language = "en",
  use_styler = FALSE,
  use_lintr = FALSE,
  apply_styler = FALSE
)
```

Arguments

dir_path	Path to the directory containing R scripts
title_prefix	Optional prefix to add to all document titles (default: NULL)
author	Author name for all documents (default: "Your name")
format	Output format - always "html" (parameter kept for backward compatibility)
theme	Quarto theme for HTML output (default: NULL uses Quarto's default). See https://quarto.org/docs/output-formats/html-themes.html
render_html	Logical, whether to render the .qmd files to HTML after creation (default: FALSE)
output_html_dir	Directory path for HTML output files (optional, defaults to same directory as .qmd files)
open_html	Logical, whether to open the HTML files in browser after rendering (default: FALSE)
code_fold	Logical, whether to fold code blocks in HTML output (default: FALSE)
number_sections	Logical, whether to number sections automatically (default: TRUE)
recursive	Logical, whether to search subdirectories recursively (default: TRUE)
pattern	Regular expression pattern to filter R files (default: "\.R$")
exclude_pattern	Optional regular expression pattern to exclude certain files (default: NULL)

create_book	Logical, whether to create a Quarto book structure with _quarto.yml (default: TRUE)
book_title	Title for the Quarto book (default: "R Scripts Documentation")
output_dir	Output directory for the book (required if create_book=TRUE, default: NULL uses input_dir/output)
language	Language for the documentation ("en" or "fr", default: "en")
use_styler	Logical, whether to apply styler code formatting and show differences in tabsets (default: FALSE). Requires the styler package to be installed.
use_lintr	Logical, whether to run lintr code quality checks and display issues in tabsets (default: FALSE). Requires the lintr package to be installed.
apply_styler	Logical, whether to apply styler formatting directly to the source R script files (default: FALSE). If TRUE, all input files will be modified with styled code. Requires use_styler = TRUE to take effect.

Details

Supports all features of rtoqmd, including:

- Metadata detection (Title, Author, Date, Description)
- RStudio section headers
- Callouts (note, tip, warning, caution, important)
- Code blocks and comments

See rtoqmd for details on callout syntax and metadata detection.

Value

Invisibly returns a data frame with conversion results (file paths and status)

Note

Existing .qmd and .html files will be automatically overwritten during generation to ensure fresh output.

Examples

```
## Not run:
# Convert all R scripts in a directory
rtoqmd_dir("path/to/scripts")

# Convert and render all scripts
rtoqmd_dir("path/to/scripts", render_html = TRUE)

# Create a Quarto book with automatic navigation
rtoqmd_dir(
  dir_path = "path/to/scripts",
  output_html_dir = "path/to/scripts/documentation",
  render_html = TRUE,
```

```
  author = "Your Name",
  book_title = "My R Scripts Documentation",
  open_html = TRUE
)

# Create a Quarto book in French
rtoqmd_dir(
  dir_path = "path/to/scripts",
  output_html_dir = "path/to/scripts/documentation",
  render_html = TRUE,
  author = "Votre Nom",
  book_title = "Documentation des Scripts R",
  language = "fr"
)

# Convert with custom author and title prefix
rtoqmd_dir("path/to/scripts",
           title_prefix = "Analysis: ",
           author = "Data Team")

# Exclude certain files (e.g., test files)
rtoqmd_dir("path/to/scripts",
           exclude_pattern = "test_.*\\.R$")

# Non-recursive (only current directory)
rtoqmd_dir("path/to/scripts", recursive = FALSE)

# Reproducible example with sample scripts
example_dir <- system.file("examples", "book_example", package = "quartify")
if (example_dir != "") {
  rtoqmd_dir(
    dir_path = example_dir,
    output_html_dir = file.path(example_dir, "documentation"),
    render_html = TRUE,
    open_html = TRUE
  )
}

## End(Not run)
```

Index