Package 'frontmatter'

January 14, 2026

Title Parse Front Matter from Documents

Version 0.1.0

Description Extracts and parses structured metadata ('YAML' or 'TOML') from the beginning of text documents. Front matter is a common pattern in 'Quarto' documents, 'R Markdown' documents, static site generators, documentation systems, content management tools and even 'Python' and 'R' scripts where metadata is placed at the top of a document, separated from the main content by delimiter fences.

License MIT + file LICENSE

URL https://github.com/posit-dev/frontmatter, https://posit-dev.github.io/frontmatter/

BugReports https://github.com/posit-dev/frontmatter/issues

Imports cpp11, rlang, tomledit, yaml12

Suggests testthat (>= 3.0.0), withr, yaml

LinkingTo cpp11

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation yes

Author Garrick Aden-Buie [aut, cre] (ORCID: <https://orcid.org/0000-0002-7111-0077>), Posit Software, PBC [cph, fnd] (ROR: <https://ror.org/03wc8by49>)

Maintainer Garrick Aden-Buie <garrick@posit.co>

Repository CRAN

Date/Publication 2026-01-14 18:10:07 UTC

Contents

parse_front_matter *Parse YAML or TOML Front Matter*

Description

Extract and parse YAML or TOML front matter from a file or a text string. Front matter is structured metadata at the beginning of a document, delimited by fences (`---` for YAML, `+++` for TOML). `parse_front_matter()` processes a character string, while `read_front_matter()` reads from a file. Both functions return a list with the parsed front matter and the document body.

Usage

```
parse_front_matter(text, parse_yaml = NULL, parse_toml = NULL)

read_front_matter(path, parse_yaml = NULL, parse_toml = NULL)
```

Arguments

`text`	A character string or vector containing the document text. If a vector with multiple elements, they are joined with newlines (as from `readLines()`).
`parse_yaml`, `parse_toml`	A function that takes a string and returns a parsed R object, or `NULL` to use the default parser. Use `identity` to return the raw string without parsing.
`path`	A character string specifying the path to a file. The file is assumed to be UTF-8 encoded. A UTF-8 BOM (byte order mark) at the start of the file is automatically stripped if present.

Value

A named list with two elements:

- `data`: The parsed front matter as an R object, or `NULL` if no valid front matter was found.
- `body`: The document content after the front matter, with leading empty lines removed. If no front matter is found, this is the original text.

Functions

- `parse_front_matter()`: Parse front matter from text
- `read_front_matter()`: Parse front matter from a file.

Custom Parsers

By default, the package uses `yaml12::parse_yaml()` for YAML and `tomledit::parse_toml()` for TOML. You can provide custom parser functions via `parse_yaml` and `parse_toml` to override these defaults.

Use `identity` to return the raw YAML or TOML string without parsing.

YAML Specification Version

The default YAML parser uses YAML 1.2 via `yaml12::parse_yaml()`. To use YAML 1.1 parsing instead (via `yaml::yaml.load()`), set either:

- The R option `frontmatter.parse_yaml.spec` to `"1.1"`
- The environment variable `FRONTMATTER_PARSE_YAML_SPEC` to `"1.1"`

The option takes precedence over the environment variable. Valid values are `"1.1"` and `"1.2"` (the default).

YAML 1.1 differs from YAML 1.2 in several ways, most notably in how it handles boolean values (e.g., yes/no are booleans in 1.1 but strings in 1.2).

Examples

```
# Parse YAML front matter
text <- "---
title: My Document
date: 2024-01-01
---
Document content here"

result <- parse_front_matter(text)
result$data$title  # "My Document"
result$body        # "Document content here"

# Parse TOML front matter
text <- "+++
title = 'My Document'
date = 2024-01-01
+++
Document content"

result <- parse_front_matter(text)

# Get raw YAML without parsing
result <- parse_front_matter(text, parse_yaml = identity)

# Use a custom parser that adds metadata
result <- parse_front_matter(
  text,
  parse_yaml = function(x) {
    data <- yaml12::parse_yaml(x)
    data$parsed_at <- Sys.time()
    data
  }
)

# Or read from a file
tmpfile <- tempfile(fileext = ".md")
writeLines(text, tmpfile)
```

```
read_front_matter(tmpfile)
```

Index