Package 'readMDTable'

June 24, 2026

Title Read Markdown Tables into Tibbles

Version 0.4.0

Description Efficient reading of raw markdown tables into tibbles. Designed to accept content from strings, files, and URLs with the ability to extract and read multiple tables from markdown for analysis.

Depends R (>= 4.1.0)

Imports cli, httr2, lifecycle, readr (>= 2.2.0), purrr, stringr

URL https://github.com/jrdnbradford/readMDTable, https://jrdnbradford.github.io/readMDTable/

BugReports https://github.com/jrdnbradford/readMDTable/issues

License GPL (>= 3)

Encoding UTF-8

Config/roxygen2/version 8.0.0

Config/roxygen2/markdown TRUE

Suggests covr, devtools, dplyr, ggplot2, knitr, lubridate, microbenchmark, precommit, rmarkdown, rvest, testthat (>= 3.0.0), tibble, usethis

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Jordan Bradford [aut, cre, cph] (ORCID: <https://orcid.org/0009-0000-8570-3474>)

Maintainer Jordan Bradford <jrdnbradford@gmail.com>

Repository CRAN

Date/Publication 2026-06-24 05:20:02 UTC

Contents

`read_md_table` *Read Markdown Tables into Tibbles*

Description

Read Markdown Tables into Tibbles

Usage

```
read_md_table(file, warn = TRUE, force = FALSE, ...)

extract_md_tables(file, warn = TRUE, force = FALSE, ...)

extract_md_table(file, warn = TRUE, force = FALSE, ...)
```

Arguments

`file`	Either a path to a file, a connection, or literal data (either a single string or a raw vector). Files starting with `http://`, `https://`, `ftp://`, or `ftps://` will be automatically downloaded.
`warn`	Boolean. Should warnings be raised if `file` does not appear to contain a markdown table? Defaults to `TRUE`.
`force`	Boolean. Should `read_md_table` attempt to read in content that does not fit the regex? This param should be used carefully as it may cause `read_md_table` to return unexpected data. Defaults to `FALSE`.
`...`	Arguments passed on to `readr::read_delim` `quote` Single character used to quote strings. `escape_backslash` Does the file use backslashes to escape special characters? This is more general than `escape_double` as backslashes can be used to escape the delimiter character, the quote character, or to add special characters like `\\n`. `escape_double` Does the file escape quotes by doubling them? i.e. If this option is `TRUE`, the value `""""` represents a single quote, `\"`. `col_names` Either `TRUE`, `FALSE` or a character vector of column names. If `TRUE`, the first row of the input will be used as the column names, and will not be included in the data frame. If `FALSE`, column names will be generated automatically: X1, X2, X3 etc. If `col_names` is a character vector, the values will be used as the names of the columns, and the first row of the input will be read into the first row of the output data frame. Missing (`NA`) column names will generate a warning, and be filled in with dummy names `...1`, `...2` etc. Duplicate column names will generate a warning and be made unique, see `name_repair` to control how this is done.

`col_types` One of `NULL`, a `cols()` specification, or a string. See `vignette("readr")` for more details.

If `NULL`, all column types will be inferred from `guess_max` rows of the input, interspersed throughout the file. This is convenient (and fast), but not robust. If the guessed types are wrong, you'll need to increase `guess_max` or supply the correct types yourself.

Column specifications created by `list()` or `cols()` must contain one column specification for each column. If you only want to read a subset of the columns, use `cols_only()`.

Alternatively, you can use a compact string representation where each character represents one column:

- c = character
- i = integer
- n = number
- d = double
- l = logical
- f = factor
- D = date
- T = date time
- t = time
- ? = guess
- _ or - = skip

By default, reading a file without a column specification will print a message showing what `readr` guessed they were. To remove this message, set `show_col_types = FALSE` or set `options(readr.show_col_types = FALSE)`.

`col_select` Columns to include in the results. You can use the same mini-language as `dplyr::select()` to refer to the columns by name. Use `c()` to use more than one selection expression. Although this usage is less common, `col_select` also accepts a numeric column index. See `?tidyselect::language` for full details on the selection language.

`id` The name of a column in which to store the file path. This is useful when reading multiple input files and there is data in the file paths, such as the data collection date. If `NULL` (the default) no extra column is created.

`locale` The locale controls defaults that vary from place to place. The default locale is US-centric (like R), but you can use `locale()` to create your own locale that controls things like the default time zone, encoding, decimal mark, big mark, and day/month names.

`na` Character vector of strings to interpret as missing values. Set this option to `character()` to indicate no missing values.

`quoted_na` **[Deprecated]** Should missing values inside quotes be treated as missing values (the default) or strings. This argument is deprecated and only works when using the legacy first edition parser. See `with_edition()` for more.

`comment` A string used to identify comments. Any text after the comment characters will be silently ignored.

`skip` Number of lines to skip before reading data. If `comment` is supplied any commented lines are ignored *after* skipping.

`n_max` Maximum number of lines to read.

`guess_max` Maximum number of lines to use for guessing column types. Will never use more than the number of lines read. See `vignette("column-types", package = "readr")` for more details.

`name_repair` Handling of column names. The default behaviour is to ensure column names are `"unique"`. Various repair strategies are supported:

- `"minimal"`: No name repair or checks, beyond basic existence of names.
- `"unique"` (default value): Make sure names are unique and not empty.
- `"check_unique"`: No name repair, but check they are `unique`.
- `"unique_quiet"`: Repair with the `unique` strategy, quietly.
- `"universal"`: Make the names `unique` and syntactic.
- `"universal_quiet"`: Repair with the `universal` strategy, quietly.
- A function: Apply custom name repair (e.g., `name_repair = make.names` for names in the style of base R).
- A purrr-style anonymous function, see `rlang::as_function()`.

This argument is passed on as `repair` to `vctrs::vec_as_names()`. See there for more details on these terms and the strategies used to enforce them.

`num_threads` The number of processing threads to use for initial parsing and lazy reading of data. If your data contains newlines within fields the parser should automatically detect this and fall back to using one thread only. However if you know your file has newlines within quoted fields it is safest to set `num_threads = 1` explicitly.

`progress` Display a progress bar? By default it will only display in an interactive session and not while knitting a document. The automatic progress bar can be disabled by setting option `readr.show_progress` to `FALSE`.

`show_col_types` If `FALSE`, do not show the guessed column types. If `TRUE` always show the column types, even if they are supplied. If `NULL` (the default) only show the column types if they are not explicitly supplied by the `col_types` argument.

`skip_empty_rows` Should blank rows be ignored altogether? i.e. If this option is `TRUE` then blank rows will not be represented at all. If it is `FALSE` then they will be represented by `NA` values in all the columns.

`lazy` Read values lazily? By default, this is `FALSE`, because there are special considerations when reading a file lazily that have tripped up some users. Specifically, things get tricky when reading and then writing back into the same file. But, in general, lazy reading (`lazy = TRUE`) has many benefits, especially for interactive use and when your downstream work only involves a subset of the rows or columns.

Learn more in `should_read_lazy()` and in the documentation for the `altrep` argument of `vroom::vroom()`.

Details

`read_md_table` reads all markdown tables from a string, file, or URL and returns them as a named

list of tibbles. It uses `readr::read_delim` to efficiently read in data.

If `warn` is `TRUE`, `read_md_table` will warn if no markdown tables are detected in the content. When `force` is also `TRUE`, it will attempt to read the content as a table anyway. `readr::read_delim` will provide its own warnings if there are potential issues with a table.

Value

A named list of tibbles (names `"table_1"`, `"table_2"`, ...), or `NULL` if no tables are found and `force` is `FALSE`.

Examples

```
# Read a single table from a file
read_md_table(read_md_table_example("mtcars.md"))

# Read multiple tables from a file
read_md_table(read_md_table_example("mtcars-split.md"), show_col_types = FALSE)

# Read from a string
read_md_table(
  "| H1 | H2 | \n|-----|-----|\n| R1C1 | R1C2 |\n| R2C1 | R2C2 |",
  warn = FALSE,
  force = TRUE
)


# Read from a URL
read_md_table(
  "https://raw.githubusercontent.com/jrdnbradford/readMDTable/main/inst/extdata/iris.md"
)


# Get warning for malformed tables
read_md_table(
  "| Name  | Age | City        | Date       |
   |-------|-----|-------------|------------|
   | Alice | 30  | New York    | 2021/01/08 |
   | Bob   | 25  | Los Angeles | 2023/07/22 |
     Carol | 27  | Chicago     | 2022/11/01  ",
  force = TRUE
)
```

---

read_md_table_example *Get Path to readMDTable Examples*

Description

Get Path to readMDTable Examples

Usage

```
read_md_table_example(file = NULL)
```

Arguments

file	Name of file. If NULL, the example files will be listed.

Details

readMDTable comes with a number of well-known datasets as example markdown tables in the inst/extdata directory. read_md_table_example will list the file names or return the path of a specified file.

Value

Vector of example file names if file is NULL, else the path to the example markdown table file.

Examples

```
# List the available example files
read_md_table_example()

# Get the path to the mtcars example file
read_md_table_example("mtcars.md")

# Read in an example file
mtcars_path <- read_md_table_example("mtcars.md")
read_md_table(mtcars_path)
```

Index