Package 'simplermarkdown'

October 14, 2022

Title Simple Engine for Generating Reports using R

Version 0.0.4

Description Runs R-code present in a pandoc markdown file and includes the resulting output in the resulting markdown file. This file can then be converted into any of the output formats supported by pandoc. The package can also be used as an engine for writing package vignettes.

BugReports `https://github.com/djvanderlaan/simplermarkdown/issues`

URL `https://github.com/djvanderlaan/simplermarkdown`

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.1.1

Imports rjson, tools

Suggests MASS

VignetteBuilder simplermarkdown

SystemRequirements Pandoc (http://pandoc.org) needs to installed and available in the search path.

NeedsCompilation no

Author Jan van der Laan [aut, cre] (<`https://orcid.org/0000-0002-0693-1514`>)

Maintainer Jan van der Laan <`r@eoos.dds.nl`>

Repository CRAN

Date/Publication 2022-01-03 20:20:02 UTC

R topics documented:

`file_subs_ext` *Replace file extension by another file extension*

Description

Replace file extension by another file extension

Usage

```
file_subs_ext(fn, new_ext, check = TRUE)
```

Arguments

`fn`	character vector with file names
`new_ext`	character vector of length one with the new extension (when it does not start with a period a period is added).
`check`	check if the new file name is not equal to the original filename. If so, generate an error.

Value

Returns a character vector of the same length of `fn` with the extension of the file names in `fn` replaced by `new_ext`.

`markdown_block` *Return a code block object that can be included in the pandoc parse tree*

Description

Return a code block object that can be included in the pandoc parse tree

Usage

```
markdown_block(content, language, id = "", ...)
```

Arguments

`content`	a character vector containing the code
`language`	language of the code in the code block
`id`	optional id of the code block
`...`	ignored.

Value

Returns a `list` with the correct structure for a code block in the pandoc parse tree.

`mdtangle` *Extract code from the code blocks in a markdown file*

Description

Extract code from the code blocks in a markdown file

Usage

```
mdtangle(
  fn,
  ofn = file_subs_ext(basename(fn), ".R"),
  extra_arguments = "",
  cmd = "pandoc %3$s -s \"%1$s\" -t json -o \"%2$s\"",
  ...
)
```

Arguments

`fn`	filename of the markdown file (should use pandoc markdown).
`ofn`	name of the resulting R-script
`extra_arguments`	extra arguments passed on to pandoc. Should be a length 1 character vector.
`cmd`	command used to run pandoc. See details.
`...`	ignored

Details

`mdtangle` calls pandoc. Pandoc will parse the markdown document and write the parsed file to temporary file. This file is read by `mdtangle` and the code is extracted from it and written to `ofn`.

Using the `cmd` argument the exact command used to run pandoc can be modified. It is passed on to `sprintf` and uses positional arguments: (1) name of the input file, (2) location of the temporary file to which the parsed document is written, (3) the value of `extra_arguments`.

Value

Returns the filename of the generated file.

`mdweave` *Run the code in a markdown file and generate a new markdown file*

Description

Run the code in a markdown file and generate a new markdown file

Usage

```
mdweave(
  fn,
  ofn = file_subs_ext(basename(fn), ".md", FALSE),
  cmd1 = "pandoc -s \"%1$s\" -t json -o \"%2$s\"",
  cmd2 = "pandoc -s \"%1$s\" -t markdown -o \"%2$s\"",
  ...
)
```

Arguments

`fn`	filename of the markdown file (should use pandoc markdown).
`ofn`	name of the resulting markdown file.
`cmd1`	command used to run pandoc. See details.
`cmd2`	command used to run pandoc. See details.
`...`	ignored

Details

`mdweave` calls pandoc twice. In the first call the markdown file is parsed by pandoc and the parse tree is written to a temporary file. This parse tree is the read by mdweave and any R-code in the tree is executed resulting in a modified parse tree. This file is then stored to a new temporary file. Pandoc is the called a second time to convert the new parse tree to a markdown file.

The arguments `cmd1` and `cmd2` contain the calls used to run pandoc. The arguments can be used to, for example pas additional arguments to pandoc. They use positional arguments. In `cmd1`, the first argument (`%1$s`) is the input file name and the second (`%2$s`) the temporary file containing the parsed tree. In `cmd2`, the first argument is the temporary file with the modified parse tree and the second argument the output file.

Value

Returns the file name of the file generated (`ofn`). Called mainly for the side effect of parsing and generating a markdown file (and possibly secondary files such as figures).

`mdweave_to_pdf` *Run the code in a markdown file and generate a new document*

Description

Run the code in a markdown file and generate a new document

Usage

```
mdweave_to_pdf(
  fn,
  ofn = file_subs_ext(basename(fn), ".pdf", FALSE),
  extra_arguments2 = "--self-contained",
  run_in_temp = TRUE,
  cmd2 = "pandoc %3$s -s \"%1$s\" -t latex -o \"%2$s\"",
  ...
)

mdweave_to_tex(
  fn,
  ofn = file_subs_ext(basename(fn), ".tex", FALSE),
  extra_arguments2 = "--self-contained",
  run_in_temp = TRUE,
  cmd2 = "pandoc %3$s -s \"%1$s\" -t latex -o \"%2$s\"",
  ...
)

mdweave_to_html(
  fn,
  ofn = file_subs_ext(basename(fn), ".html", FALSE),
  extra_arguments2 = "--self-contained",
  run_in_temp = TRUE,
  cmd2 = "pandoc %3$s -s \"%1$s\" -t html -o \"%2$s\"",
  ...
)
```

Arguments

`fn`	filename of the markdown file (should use pandoc markdown).
`ofn`	name of the resulting file.
`extra_arguments2`	extra arguments passed on to pandoc. Should be a length 1 character vector.
`run_in_temp`	When TRUE the intermediary markdown file and generated figures (when not using custom paths) are created in a temporary directory. Otherwise these will be generated in the same directory as the output file.
`cmd2`	command used to run pandoc. See details.
`...`	additional arguments are passed on to `mdweave`.

Details

These functions first call mdweave to run the code in the original file and convert the original markdown file to a new markdown file. This second markdown file is then converted to the desired output format using a second run of pandoc.

In case of converting to pdf the file is required to have the extension .pdf. In case of converting to LaTeX, the file cannot have the extension .pdf. That is because in both cases the file is first converted to LaTeX. In case of a file with the extension .pdf the file is than further converted to PDF.

Value

Returns the name of the resulting outout file.

---

md_figure *Generate a figure and generate the markdown to include the figure*

Description

Will evaluate the expressions in expr and capture the output on the given plotting device in the given file. It will then generate the markdown needed to include that figure in a markdown document.

Usage

```
md_figure(
  expr,
  name,
  caption = "",
  id = "",
  dir = file.path(Sys.getenv("MDOUTDIR", "."), "figures"),
  device = c("png", "pdf"),
  ...,
  as_character = FALSE,
  echo = FALSE,
  results = FALSE
)
```

Arguments

expr	the expressions to evaluate. Will generally contain plotting commands. The expressions are evaluated in the global environment.
name	the name of the figure.
caption	text of the caption. When omitted no caption is added to the figure.
id	id of the figure. When omitted or equal to NULL or an empty character, no id is added to the figure.
dir	name of the directory in which to store the file.

`device`	the graphics device to use for creating the image.
`...`	passed on to the graphics device.
`as_character`	return the figure as a character vector. If `FALSE` the figure will be written to the standard output.
`echo`	the code in `code` is repeated in the output.
`results`	include the results of running the code in the output. The output of code that explicitly writes to standard output is always included.

Details

The image is stored in the file `dir/name.device`.

Value

When `as_character = FALSE` a character vector with the markdown needed to include the generated figure in a markdown file is returned. Otherwise, nothing is returned; the markdown is written to the console.

`md_table` *Generate a markdown table from a data.frame*

Description

Generate a markdown table from a data.frame

Usage

```
md_table(tab, caption, as_character = FALSE, ...)
```

Arguments

`tab`	a data frame
`caption`	text of the caption. When omitted no caption is added to the table.
`as_character`	return the table as a character vector. If `FALSE` the table will be written to the standard output.
`...`	unused.

Value

Then `as_character = FALSE` a character vector with the markdown containing the table is returned. Otherwise, nothing is returned; the markdown is then written to the console.

output_table *Output filters for code blocks in markdown*

Description

Output filters for code blocks in markdown

Usage

```
output_table(code, language = "R", id = "", ...)

output_figure(code, language = "R", id = "", ...)

output_eval(code, language = "R", id = "", echo = TRUE, results = TRUE, ...)

output_raw(code, language = "R", id = "", ...)

output_str(code, language = "R", id = "", ...)
```

Arguments

`code`	character vector containing the code of the code block
`language`	the language in which the code is written (as specified in the markdown file).
`id`	the identifier of the code block.
`...`	additional arguments specified as arguments in the code block are passed on to the filter function. Often these are ignored, or they are passed on to other functions (see 'Details').
`echo`	the code in `code` is repeated in the output.
`results`	include the results of running the code in the output. The output of code that explicitly writes to standard output is always included.

Details

The filter functions `tab` and `fig` call `md_table` and `md_figure` respectively; additional arguments are passed on to those functions. Other filter functions ignore the additional arguments.

It is also possible to write custom output filter. An output filter should have `code`, `language` and `id` as its first three arguments. It should either return a character vector containing the markdown that should be included in the resulting markdown file or an object that can be directly included in the pandoc parse tree. If the function does not return a character vector it is assumed the latter is returned. `simplermarkdown` defines a small number of valid object constructors: `raw_block` and `markdown_block`.

The custom function should be available when running the markdown document through pandoc. The easiest way is to `source` or define the function in the markdown document before using it.

Value

The functions either return a character vector with markdown, or return a list with the correct structure to include in the pandow parse tree.

`raw_block` *Return a raw chunk of text that can be included in the pandoc parse tree*

Description

Return a raw chunk of text that can be included in the pandoc parse tree

Usage

```
raw_block(content, language = "markdown")
```

Arguments

`content`	a character vector containing the content to include in the final document.
`language`	language of the content

Details

A raw block is included as is into the final markdown document. This can be used for example to include raw chunks of markdown.

Value

Returns a `list` with the correct structure for a `RowBlock` in the pandoc parse tree.

Index