Package: rmarkdown (via r-universe)

November 5, 2024

Type Package

Title Dynamic Documents for R

Version 2.29

Description Convert R Markdown documents into a variety of formats.

License GPL-3

URL https://github.com/rstudio/rmarkdown, https://pkgs.rstudio.com/rmarkdown/

BugReports https://github.com/rstudio/rmarkdown/issues

Depends R (>= 3.0)

Imports bslib (>= 0.2.5.1), evaluate (>= 0.13), fontawesome (>= 0.5.0), htmltools (>= 0.5.1), jquerylib, jsonlite, knitr (>= 1.43), methods, tinytex (>= 0.31), tools, utils, xfun (>= 0.36), yaml (>= 2.1.19)

Suggests digest, dygraphs, fs, rsconnect, downlit (>= 0.4.0), katex (>= 1.4.0), sass (>= 0.4.0), shiny (>= 1.6.0), testthat (>= 3.0.3), tibble, vctrs, cleanrmd, withr (>= 2.4.2), xml2

VignetteBuilder knitr

Config/Needs/website rstudio/quillt, pkgdown

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

SystemRequirements pandoc (>= 1.14) - http://pandoc.org

NeedsCompilation no

Author JJ Allaire [aut], Yihui Xie [aut, cre] (<https://orcid.org/0000-0003-0645-5666>), Christophe Dervieux [aut] (<https://orcid.org/0000-0003-4474-2498>), Jonathan McPherson [aut], Javier Luraschi [aut], Kevin Ushey [aut], Aron Atkins [aut], Hadley Wickham [aut], Joe Cheng [aut], Winston Chang [aut], Richard Iannone [aut] (<https://orcid.org/0000-0003-3925-190X>), Andrew Dunning [ctb]

(<https://orcid.org/0000-0003-0464-5036>), Atsushi Yasumoto [ctb, cph] (<https://orcid.org/0000-0002-8335-495X>, Number sections Lua filter), Barret Schloerke [ctb], Carson Sievert [ctb] (<https://orcid.org/0000-0002-4958-2844>), Devon Ryan [ctb] (<https://orcid.org/0000-0002-8549-0971>), Frederik Aust [ctb] (<https://orcid.org/0000-0003-4900-788X>), Jeff Allen [ctb], JooYoung Seo [ctb] (<https://orcid.org/0000-0002-4064-6012>), Malcolm Barrett [ctb], Rob Hyndman [ctb], Romain Lesur [ctb], Roy Storey [ctb], Ruben Arslan [ctb], Sergio Oller [ctb], Posit Software, PBC [cph, fnd], jQuery UI contributors [ctb, cph] (jQuery UI library; authors listed in inst/rmd/h/jqueryui/AUTHORS.txt), Mark Otto [ctb] (Bootstrap library), Jacob Thornton [ctb] (Bootstrap library), Bootstrap contributors [ctb] (Bootstrap library), Twitter, Inc [cph] (Bootstrap library), Alexander Farkas [ctb, cph] (html5shiv library), Scott Jehl [ctb, cph] (Respond.js library), Ivan Sagalaev [ctb, cph] (highlight.js library), Greg Franko [ctb, cph] (tocify library), John MacFarlane [ctb, cph] (Pandoc templates), Google, Inc. [ctb, cph] (ioslides library), Dave Raggett [ctb] (slidy library), W3C [cph] (slidy library), Dave Gandy [ctb, cph] (Font-Awesome), Ben Sperry [ctb] (Ionicons), Drifty [cph] (Ionicons), Aidan Lister [ctb, cph] (jQuery StickyTabs), Benct Philip Jonsson [ctb, cph] (pagebreak Lua filter), Albert Krewinkel [ctb, cph] (pagebreak Lua filter)

Maintainer Yihui Xie <xie@yihui.name>

Repository CRAN

Date/Publication 2024-11-04 12:30:09 UTC

Config/pak/sysreqs make

Contents

`rmarkdown-package` *R Markdown Document Conversion*

Description

Convert R Markdown documents into a variety of formats including HTML, MS Word, PDF, and Beamer.

Details

The **rmarkdown** package includes high level functions for converting to a variety of formats. For example:

```
render("input.Rmd", html_document())
render("input.Rmd", pdf_document())
```

You can also specify a plain markdown file in which case knitting will be bypassed:

```
render("input.md", html_document())
```

Additional options can be specified along with the output format:

```
render("input.Rmd", html_document(toc = TRUE))
render("input.Rmd", pdf_document(latex_engine = "lualatex"))
render("input.Rmd", beamer_presentation(incremental = TRUE))
```

You can also include arbitrary pandoc command line arguments along with the other options:

```
render("input.Rmd", pdf_document(toc = TRUE, pandoc_args = "--listings"))
```

Author(s)

Maintainer: Yihui Xie <xie@yihui.name> (ORCID)

Authors:

- JJ Allaire <jj@posit.co>
- Christophe Dervieux <cderv@posit.co> (ORCID)
- Jonathan McPherson <jonathan@posit.co>
- Javier Luraschi
- Kevin Ushey <kevin@posit.co>
- Aron Atkins <aron@posit.co>
- Hadley Wickham <hadley@posit.co>

- Joe Cheng <joe@posit.co>
- Winston Chang <winston@posit.co>
- Richard Iannone <rich@posit.co> (ORCID)

Other contributors:

- Andrew Dunning (ORCID) [contributor]
- Atsushi Yasumoto (ORCID) (Number sections Lua filter) [contributor, copyright holder]
- Barret Schloerke [contributor]
- Carson Sievert (ORCID) [contributor]
- Devon Ryan <dpryan79@gmail.com> (ORCID) [contributor]
- Frederik Aust <frederik.aust@uni-koeln.de> (ORCID) [contributor]
- Jeff Allen <jeff@posit.co> [contributor]
- JooYoung Seo (ORCID) [contributor]
- Malcolm Barrett [contributor]
- Rob Hyndman <Rob.Hyndman@monash.edu> [contributor]
- Romain Lesur [contributor]
- Roy Storey [contributor]
- Ruben Arslan <ruben.arslan@uni-goettingen.de> [contributor]
- Sergio Oller [contributor]
- Posit Software, PBC [copyright holder, funder]
- jQuery UI contributors (jQuery UI library; authors listed in inst/rmd/h/jqueryui/AUTHORS.txt) [contributor, copyright holder]
- Mark Otto (Bootstrap library) [contributor]
- Jacob Thornton (Bootstrap library) [contributor]
- Bootstrap contributors (Bootstrap library) [contributor]
- Twitter, Inc (Bootstrap library) [copyright holder]
- Alexander Farkas (html5shiv library) [contributor, copyright holder]
- Scott Jehl (Respond.js library) [contributor, copyright holder]
- Ivan Sagalaev (highlight.js library) [contributor, copyright holder]
- Greg Franko (tocify library) [contributor, copyright holder]
- John MacFarlane (Pandoc templates) [contributor, copyright holder]
- Google, Inc. (ioslides library) [contributor, copyright holder]
- Dave Raggett (slidy library) [contributor]
- W3C (slidy library) [copyright holder]
- Dave Gandy (Font-Awesome) [contributor, copyright holder]
- Ben Sperry (Ionicons) [contributor]
- Drifty (Ionicons) [copyright holder]
- Aidan Lister (jQuery StickyTabs) [contributor, copyright holder]
- Benct Philip Jonsson (pagebreak Lua filter) [contributor, copyright holder]
- Albert Krewinkel (pagebreak Lua filter) [contributor, copyright holder]

See Also

render, html_document, pdf_document, word_document, beamer_presentation

---

`all_output_formats` *Determine all output formats for an R Markdown document*

Description

Read the YAML metadata (and any common output YAML file) for the document and return the output formats that will be generated by a call to `render`.

Usage

```
all_output_formats(input, output_yaml = NULL)
```

Arguments

`input`	Input file (Rmd or plain markdown)
`output_yaml`	Paths to YAML files specifying output formats and their configurations. The first existing one is used. If none are found, then the function searches YAML files specified to the `output_yaml` top-level parameter in the YAML front matter, _output.yml or _output.yaml, and then uses the first existing one.

Details

This function is useful for front-end tools that require additional knowledge of the output to be produced by `render` (e.g. to customize the preview experience).

Value

A character vector with the names of all output formats.

`available_templates` *List available R Markdown template in a package*

Description

List the available template by name that can be used with `draft()` to create a new document for R Markdown from a package.

Usage

```
available_templates(package = "rmarkdown", full_path = FALSE)
```

Arguments

package	Package to list template from. Default to **rmarkdown**
full_path	Set to TRUE to get the full path to the available templates

Value

A character vector of templates name to use within `draft()`. If `full_path = TRUE`, it returns the full path to the templates.

Examples

```
# List rmarkdown templates & create a draft
available_templates()

# List rticles templates
available_templates("rticles")
```

---

beamer_presentation *Convert to a Beamer presentation*

Description

Format for converting from R Markdown to a Beamer presentation.

Usage

```
beamer_presentation(
  toc = FALSE,
  slide_level = NULL,
  number_sections = FALSE,
  incremental = FALSE,
  fig_width = 10,
  fig_height = 7,
  fig_crop = "auto",
  fig_caption = TRUE,
  dev = "pdf",
  df_print = "default",
  theme = "default",
  colortheme = "default",
  fonttheme = "default",
  highlight = "default",
  template = "default",
  keep_tex = FALSE,
  keep_md = FALSE,
  latex_engine = "pdflatex",
  citation_package = c("default", "natbib", "biblatex"),
  self_contained = TRUE,
```

```
  includes = NULL,
  md_extensions = NULL,
  pandoc_args = NULL,
  extra_dependencies = NULL
)
```

Arguments

toc	TRUE to include a table of contents in the output (only level 1 headers will be included in the table of contents).
slide_level	The heading level which defines individual slides. By default this is the highest header level in the hierarchy that is followed immediately by content, and not another header, somewhere in the document. This default can be overridden by specifying an explicit slide_level.
number_sections	TRUE to number section headings
incremental	TRUE to render slide bullets incrementally. Note that if you want to reverse the default incremental behavior for an individual bullet you can precede it with >. For example: > - Bullet Text. See more in Pandoc's Manual
fig_width	Default width (in inches) for figures
fig_height	Default height (in inches) for figures
fig_crop	Whether to crop PDF figures with the command pdfcrop. This requires the tools pdfcrop and ghostscript to be installed. By default, fig_crop = TRUE if these two tools are available.
fig_caption	TRUE to render figures with captions
dev	Graphics device to use for figure output (defaults to pdf)
df_print	Method to be used for printing data frames. Valid values include "default", "kable", "tibble", and "paged". The "default" method uses a corresponding S3 method of print, typically print.data.frame. The "kable" method uses the knitr::kable function. The "tibble" method uses the **tibble** package to print a summary of the data frame. The "paged" method creates a paginated HTML table (note that this method is only valid for formats that produce HTML). In addition to the named methods you can also pass an arbitrary function to be used for printing data frames. You can disable the df_print behavior entirely by setting the option rmarkdown.df_print to FALSE. See Data frame printing section in bookdown book for examples.
theme	Beamer theme (e.g. "AnnArbor").
colortheme	Beamer color theme (e.g. "dolphin").
fonttheme	Beamer font theme (e.g. "structurebold").
highlight	Syntax highlighting style passed to Pandoc. Supported built-in styles include "default", "tango", "pygments", "kate", "monochrome", "espresso", "zenburn", "haddock", and "breezedark". Two custom styles are also included, "arrow", an accessible color scheme, and "rstudio", which mimics the default IDE theme. Alternatively, supply a path to

a '.theme' file to use a custom Pandoc style. Note that custom theme requires Pandoc 2.0+.

Pass NULL to prevent syntax highlighting.

`template` Pandoc template to use for rendering. Pass "default" to use the rmarkdown package default template; pass NULL to use pandoc's built-in template; pass a path to use a custom template that you've created. See the documentation on pandoc online documentation for details on creating custom templates.

`keep_tex` Keep the intermediate tex file used in the conversion to PDF. Note that this argument does not control whether to keep the auxiliary files (e.g., '.aux') generated by LaTeX when compiling '.tex' to '.pdf'. To keep these files, you may set `options(tinytex.clean = FALSE)`.

`keep_md` Keep the markdown file generated by knitting.

`latex_engine` LaTeX engine for producing PDF output. Options are "pdflatex", "lualatex", "xelatex" and "tectonic".

`citation_package`
The LaTeX package to process citations, natbib or biblatex. Use default if neither package is to be used, which means citations will be processed via the command pandoc-citeproc.

`self_contained` Whether to generate a full LaTeX document (TRUE) or just the body of a LaTeX document (FALSE). Note the LaTeX document is an intermediate file unless `keep_tex = TRUE`.

`includes` Named list of additional content to include within the document (typically created using the `includes` function).

`md_extensions` Markdown extensions to be added or removed from the default definition of R Markdown. See the `rmarkdown_format` for additional details.

`pandoc_args` Additional command line options to pass to pandoc

`extra_dependencies`
A LaTeX dependency `latex_dependency()`, a list of LaTeX dependencies, a character vector of LaTeX package names (e.g. `c("framed", "hyperref")`), or a named list of LaTeX package options with the names being package names (e.g. `list(hyperef = c("unicode=true", "breaklinks=true"), lmodern = NULL)`). It can be used to add custom LaTeX packages to the .tex header.

Details

See the online documentation for additional details on using the `beamer_presentation` format.

Creating Beamer output from R Markdown requires that LaTeX be installed.

R Markdown documents can have optional metadata that is used to generate a document header that includes the title, author, and date. For more details see the documentation on R Markdown metadata.

R Markdown documents also support citations. You can find more information on the markdown syntax for citations in the Bibliographies and Citations article in the online documentation.

Value

R Markdown output format to pass to `render()`

Examples

```
## Not run:

library(rmarkdown)

# simple invocation
render("pres.Rmd", beamer_presentation())

# specify an option for incremental rendering
render("pres.Rmd", beamer_presentation(incremental = TRUE))

## End(Not run)
```

`compile_notebook` *Compiling R scripts to a notebook*

Description

R Markdown can also compile R scripts to a notebook which includes commentary, source code, and script output. Notebooks can be compiled to any output format including HTML, PDF, and MS Word.

Overview

To compile a notebook from an R script you simply pass the script to `render`. For example:

```
rmarkdown::render("analysis.R")
rmarkdown::render("analysis.R", "pdf_document")
```

The first call to `render` creates an HTML document, whereas the second creates a PDF document.

By default the name of the script, username, and current date and time are included in the header of the generated notebook. You can override this default behavior by including explicit metadata in a specially formatted R comment:

```
#' ---
#' title: "Crop Analysis Q3 2013"
#' author: "John Smith"
#' date: "May 3rd, 2014"
#' ---
```

Including Markdown

Note that the R comment used above to add a title, author, and date includes a single-quote as a special prefix character. This is a **roxygen2** style comment, and it's actually possible to include many such comments in an R script, all of which will be converted to markdown content within the generated notebook. For example:

```
#' A script comment that includes **markdown** formatting.
```

Rather than displaying as an R comment in the compiled notebook any **roxygen2** style comment will be treated as markdown and rendered accordingly.

knitr Spin

Including markdown within R comments is possible because render calls the knitr spin function to convert the R script to an Rmd file. The spin function also enables you to add knitr chunk options with another special comment prefix (#+).

Here's an example of a script that uses the various features of spin:

https://github.com/yihui/knitr/blob/master/inst/examples/knitr-spin.R

For more details on knitr::spin see the following documentation:

https://yihui.org/knitr/demo/stitch/

---

context_document *Convert to a ConTeXt document*

Description

Format for converting from R Markdown to PDF using ConTeXt.

Usage

```
context_document(
  toc = FALSE,
  toc_depth = 2,
  number_sections = FALSE,
  fig_width = 6.5,
  fig_height = 4.5,
  fig_crop = "auto",
  fig_caption = TRUE,
  dev = "pdf",
  df_print = "default",
  template = NULL,
  keep_tex = FALSE,
  keep_md = FALSE,
  citation_package = c("default", "natbib", "biblatex"),
  includes = NULL,
  md_extensions = NULL,
  output_extensions = NULL,
  pandoc_args = NULL,
  context_path = NULL,
  context_args = NULL,
  ext = c(".pdf", ".tex")
)
```

Arguments

toc	TRUE to include a table of contents in the output
toc_depth	Depth of headers to include in table of contents

number_sections
: TRUE to number section headings

fig_width
: Default width (in inches) for figures

fig_height
: Default height (in inches) for figures

fig_crop
: Whether to crop PDF figures with the command pdfcrop. This requires the tools pdfcrop and ghostscript to be installed. By default, fig_crop = TRUE if these two tools are available.

fig_caption
: TRUE to render figures with captions

dev
: Graphics device to use for figure output (defaults to pdf)

df_print
: Method to be used for printing data frames. Valid values include "default", "kable", "tibble", and "paged". The "default" method uses a corresponding S3 method of print, typically print.data.frame. The "kable" method uses the knitr::kable function. The "tibble" method uses the **tibble** package to print a summary of the data frame. The "paged" method creates a paginated HTML table (note that this method is only valid for formats that produce HTML). In addition to the named methods you can also pass an arbitrary function to be used for printing data frames. You can disable the df_print behavior entirely by setting the option rmarkdown.df_print to FALSE. See Data frame printing section in bookdown book for examples.

template
: Pandoc template to use for rendering. Pass "default" to use the rmarkdown package default template; pass NULL to use pandoc's built-in template; pass a path to use a custom template that you've created. See the documentation on pandoc online documentation for details on creating custom templates.

keep_tex
: Keep the intermediate tex file used in the conversion to PDF. Note that this argument does not control whether to keep the auxiliary files (e.g., '.aux') generated by LaTeX when compiling '.tex' to '.pdf'. To keep these files, you may set options(tinytex.clean = FALSE).

keep_md
: Keep the markdown file generated by knitting.

citation_package
: The LaTeX package to process citations, natbib or biblatex. Use default if neither package is to be used, which means citations will be processed via the command pandoc-citeproc.

includes
: Named list of additional content to include within the document (typically created using the includes function).

md_extensions
: Markdown extensions to be added or removed from the default definition of R Markdown. See the rmarkdown_format for additional details.

output_extensions
: Pandoc extensions to be added or removed from the output format, e.g., "-smart" means the output format will be latex-smart.

pandoc_args
: Additional command line options to pass to pandoc

context_path
: Path of the ConTeXt executable. If not provided, ConTeXt has to be available from the PATH environment variable.

context_args
: Command line arguments passed to ConTeXt.

ext
: Format of the output document (defaults to ".pdf").

Details

ConTeXt needs to be installed, e.g., you can install it with `tinytex::tlmgr_install("context")`.

R Markdown documents can have optional metadata that is used to generate a document header that includes the title, author, and date. For more details see the documentation on R Markdown metadata.

R Markdown documents also support citations. You can find more information on the markdown syntax for citations in the Bibliographies and Citations article in the online documentation.

Value

R Markdown output format to pass to `render`.

Examples

```
## Not run:
library(rmarkdown)

# simple invocation
render("input.Rmd", context_document())

## End(Not run)
```

---

convert_ipynb *Convert a Jupyter/IPython notebook to an R Markdown document*

Description

Read a Jupyter/IPython notebook file ('`.ipynb`') via `jsonlite::fromJSON()`, convert its code cells to R Markdown code chunks, preserve Markdown cells, and write out the results to an Rmd file.

Usage

```
convert_ipynb(input, output = xfun::with_ext(input, "Rmd"))
```

Arguments

`input`	Path to the input '`.ipynb`' file.
`output`	The output file path.

Details

This simple converter may have some rough edges, depending on how many IPython-specific features are used in a notebook. For example, line magics are not automatically converted (warnings will be issued if line magics are detected), but you may consider using or writing R functions to replace them in R Markdown (e.g., the `%load` magic may be replaced by `reticulate::source_python()`). Cell magics will be converted to code chunks with the (**knitr**) language engine names being the magic names. For example, the cell magic `%%js` is converted to ```` ```{js} ```` in R Markdown. This

does not always work because not all IPython cell magics have their counterparts in **knitr**'s language engines, but common cell magics like `%%bash`, `%%sh`, `%%js`, `%%perl`, `%%python`, and `%%ruby` should work.

Value

The output file path (invisibly).

Examples

```
# this is not a real ipynb file, but illustrates what convert_ipynb() does
nb_data <- list(
  cells = list(
    list(cell_type = 'markdown', source = 'Hi **Markdown**!'),
    list(cell_type = 'code', source = 'print("Hi R Markdown!")')
  ),
  metadata = list(
    kernelspec = list(language = 'python')
  )
)
nb_file = tempfile(fileext = '.ipynb')
jsonlite::write_json(nb_data, nb_file, auto_unbox = TRUE, pretty = TRUE)
xfun::file_string(nb_file)  # show file content

# convert to R Markdown
nb_rmd = rmarkdown:::convert_ipynb(nb_file)
xfun::file_string(nb_rmd)
```

`default_output_format` *Determine the default output format for an R Markdown document*

Description

Read the YAML metadata (and any common output YAML file) for the document and return the output format that will be generated by a call to `render`.

Usage

```
default_output_format(input, output_yaml = NULL)
```

Arguments

`input`	Input file (Rmd or plain markdown)
`output_yaml`	Paths to YAML files specifying output formats and their configurations. The first existing one is used. If none are found, then the function searches YAML files specified to the `output_yaml` top-level parameter in the YAML front matter, _output.yml or _output.yaml, and then uses the first existing one.

Details

This function is useful for front-end tools that require additional knowledge of the output to be produced by `render` (e.g. to customize the preview experience).

Value

A named list with a `name` value containing the format name and an `options` value that is a list containing all the options for the format and their values. An option's default value will be returned if the option isn't set explicitly in the document.

`draft` *Create a new document based on a template*

Description

Create (and optionally edit) a draft of an R Markdown document based on a template.

Usage

```
draft(file, template, package = NULL, create_dir = "default", edit = TRUE)
```

Arguments

`file`	File name for the draft
`template`	Template to use as the basis for the draft. This is either the full path to a template directory or the name of a template directory within the `rmarkdown/templates` directory of a package.
`package`	(Optional) Name of package where the template is located.
`create_dir`	`TRUE` to create a new directory for the document (the "default" setting leaves this behavior up to the creator of the template).
`edit`	`TRUE` to edit the template immediately

Details

The `draft` function creates new R Markdown documents based on templates that are either located on the filesystem or within an R package. The template and its supporting files will be copied to the location specified by `file`.

Value

The file name of the new document (invisibly).

Note

An R Markdown template consists of a directory that contains a description of the template, a skeleton Rmd file used as the basis for new documents, and optionally additional supporting files that are provided along with the skeleton (e.g. a logo graphic).

If the template directory is contained within a package then it should be located at `inst/rmarkdown/templates`. For example, a package named **pubtools** that wanted to provide a template named `quarterly_report` would need to provide the following files within the `pubtools/inst/rmarkdown/templates` directory:

```
quarterly_report/template.yaml
quarterly_report/skeleton/skeleton.Rmd
```

The `template.yaml` file should include a `name` field. If you want to ensure that a new directory is always created for a given template, then you can add the `create_dir` field to the `template.yaml` file. For example:

```
create_dir: true
```

The `skeleton/skeleton.Rmd` file should include the initial contents you want for files created from this template. Additional files can be added to the `skeleton` directory, for example:

```
skeleton/logo.png
```

These files will automatically be copied to the directory containing the new R Markdown draft.

Examples

```
## Not run:
rmarkdown::draft("Q4Report.Rmd",
                 template="/opt/rmd/templates/quarterly_report")

rmarkdown::draft("Q4Report.Rmd",
                 template="quarterly_report", package="pubtools")

## End(Not run)
```

---

find_external_resources

Find External Resource References

Description

Given an R Markdown document or HTML file, attempt to determine the set of additional files needed in order to render and display the document.

Usage

```
find_external_resources(input_file, encoding = "UTF-8")
```

Arguments

input_file	path to the R Markdown document or HTML file to process
encoding	Ignored. The encoding is always assumed to be UTF-8.

Details

This routine applies heuristics in order to scan a document for possible resource references.

In R Markdown documents, it looks for references to files implicitly referenced in Markdown (e.g. `![alt](img.png)`), in the document's YAML header, in raw HTML chunks, and as quoted strings in R code chunks (e.g. `read.csv("data.csv")`).

Resources specified explicitly in the YAML header for R Markdown documents are also returned. To specify resources in YAML, use the `resource_files` key:

```
---
title: My Document
author: My Name
resource_files:
  - data/mydata.csv
  - images/figure.png
---
```

Each item in the `resource_files` list can refer to:

1. A single file, such as `images/figure.png`, or
2. A directory, such as `resources/data`, in which case all of the directory's content will be recursively included, or
3. A wildcard pattern, such as `data/*.csv`, in which case all of the files matching the pattern will be included. No recursion is done in this case.

In HTML files (and raw HTML chunks in R Markdown documents), this routine searches for resources specified in common tag attributes, such as `<img src="...">`, `<link href="...">`, etc.

In all cases, only resources that exist on disk and are contained in the document's directory (or a child thereof) are returned.

Value

A data frame with the following columns:

path The relative path from the document to the resource

explicit Whether the resource was specified explicitly (`TRUE`) or discovered implicitly (`FALSE`)

web Whether the resource is needed to display a Web page rendered from the document

find_pandoc *Find the* pandoc *executable*

Description

Searches for the pandoc executable in a few places and use the highest version found, unless a specific version is requested.

Usage

```
find_pandoc(cache = TRUE, dir = NULL, version = NULL)
```

Arguments

cache	Whether to search for pandoc again if a Pandoc directory containing the pandoc executable of the expected version (if provided) has been found previously. Search again if cache = FALSE.
dir	A character vector of potential directory paths under which pandoc may be found. If not provided, this function searches for pandoc from the environment variable *RSTUDIO_PANDOC* (the RStudio IDE will set this variable to the directory of Pandoc bundled with the IDE), the environment variable *PATH*, and the directory '~/opt/pandoc/'.
version	The version of Pandoc to look for (e.g., "2.9.2.1"). If not provided, this function searches for the highest version under the potential directories.

Value

A list containing the directory and version of Pandoc (if found).

Note

Usually you do not need to install Pandoc if you use the RStudio IDE, because the IDE has bundled a version of Pandoc. If you have installed a version of Pandoc by yourself and want to use this version instead, you may use the dir argument of this function.

Examples

```
rmarkdown::find_pandoc()
rmarkdown::find_pandoc(dir = '~/Downloads/Pandoc')
rmarkdown::find_pandoc(version = '2.7.3')
```

`github_document` *Convert to GitHub Flavored Markdown*

Description

Format for converting from R Markdown to GitHub Flavored Markdown.

Usage

```
github_document(
  toc = FALSE,
  toc_depth = 3,
  number_sections = FALSE,
  math_method = "default",
  preserve_yaml = FALSE,
  fig_width = 7,
  fig_height = 5,
  dev = "png",
  df_print = "default",
  includes = NULL,
  md_extensions = NULL,
  hard_line_breaks = TRUE,
  pandoc_args = NULL,
  html_preview = TRUE,
  keep_html = FALSE
)
```

Arguments

`toc`	`TRUE` to include a table of contents in the output
`toc_depth`	Depth of headers to include in table of contents
`number_sections`	`TRUE` to number section headings
`math_method`	`"default"` means that native Github support for math notations using Mathjax syntax will be used. Other possible value is `"webtex"` where equation will be rendered to an image in the resulting Markdown. See `html_document()` for option to change webtex URL. Set `math_method`to `NULL` to opt-out any math treatment.
`preserve_yaml`	Preserve YAML front matter in final document.
`fig_width`	Default width (in inches) for figures
`fig_height`	Default height (in inches) for figures
`dev`	Graphics device to use for figure output (defaults to png)
`df_print`	Method to be used for printing data frames. Valid values include "default", "kable", "tibble", and "paged". The "default" method uses a corresponding S3

method of print, typically print.data.frame. The "kable" method uses the knitr::kable function. The "tibble" method uses the **tibble** package to print a summary of the data frame. The "paged" method creates a paginated HTML table (note that this method is only valid for formats that produce HTML). In addition to the named methods you can also pass an arbitrary function to be used for printing data frames. You can disable the df_print behavior entirely by setting the option rmarkdown.df_print to FALSE. See Data frame printing section in bookdown book for examples.

includes — Named list of additional content to include within the document (typically created using the includes function).

md_extensions — Markdown extensions to be added or removed from the default definition of R Markdown. See the rmarkdown_format for additional details.

hard_line_breaks — TRUE to generate markdown that uses a simple newline to represent a line break (as opposed to two-spaces and a newline).

pandoc_args — Additional command line options to pass to pandoc

html_preview — TRUE to also generate an HTML file for the purpose of locally previewing what the document will look like on GitHub.

keep_html — TRUE to keep the preview HTML file in the working directory. Default is FALSE.

Details

See the online documentation for additional details on using the github_document() format.

Value

R Markdown output format to pass to render()

About Math support

Default behavior is to keep any inline equation using $ and any block equation using $$ in the resulting markdown as Github will process those using Mathjax. **This feature is only available with Pandoc 2.10.1 and above**

When using "webtex", PNG images with a white background are used by default so that it shows correctly on Github on both light and dark theme. You can choose to only output SVG for better quality by changing the URL used:

```
output:
  github_document:
    math_method:
      engine: webtex
      url: https://latex.codecogs.com/svg.image?
```

Background or fonts color cannot be changed for now and your equation may not be visible on dark theme.

Using "webtex" will be the default with Pandoc 2.0.4 until Pandoc 2.10. Before 2.0.4, Github document output does not support math.

`html-dependencies` *Provide common HTML dependencies for R Markdown formats*

Description

These functions provide common HTML dependencies (e.g. jquery, bootstrap) for re-use by other R Markdown formats.

Usage

```
html_dependency_jquery()

html_dependency_jqueryui()

html_dependency_bootstrap(theme)

html_dependency_tocify()

html_dependency_font_awesome()

html_dependency_ionicons()

html_dependency_pagedtable()

html_dependency_highlightjs(highlight)

html_dependency_tabset()

html_dependency_codefolding_lua()
```

Arguments

`theme`	One of the following:
	• A `bslib::bs_theme()` object (or a list of `bslib::bs_theme()` argument values)
	  – Use this option for custom themes using Bootstrap 4 or 3.
	  – In this case, any `.scss`/`.sass` files provided to the `css` parameter may utilize the `theme`'s underlying Sass utilities (e.g., variables, mixins, etc).
	• `NULL` for no theme (i.e., no `html_dependency_bootstrap()`).
	• A character string specifying a Bootswatch 3 theme name (for backwards-compatibility).

`highlight`	Highlighter to use

`html_document` *Convert to an HTML document*

Description

Format for converting from R Markdown to an HTML document.

Usage

```
html_document(
  toc = FALSE,
  toc_depth = 3,
  toc_float = FALSE,
  number_sections = FALSE,
  anchor_sections = FALSE,
  section_divs = TRUE,
  fig_width = 7,
  fig_height = 5,
  fig_retina = 2,
  fig_caption = TRUE,
  dev = "png",
  df_print = "default",
  code_folding = c("none", "show", "hide"),
  code_download = FALSE,
  self_contained = TRUE,
  theme = "default",
  highlight = "default",
  highlight_downlit = FALSE,
  math_method = "default",
  mathjax = "default",
  template = "default",
  extra_dependencies = NULL,
  css = NULL,
  includes = NULL,
  keep_md = FALSE,
  lib_dir = NULL,
  md_extensions = NULL,
  pandoc_args = NULL,
  ...
)
```

Arguments

`toc`	`TRUE` to include a table of contents in the output
`toc_depth`	Depth of headers to include in table of contents

`toc_float` `TRUE` to float the table of contents to the left of the main document content. Rather than `TRUE` you may also pass a list of options that control the behavior of the floating table of contents. See the *Floating Table of Contents* section below for details.

`number_sections` `TRUE` to number section headings

`anchor_sections` `TRUE` to show section anchors when mouse hovers for all headers. A list can also be passed with `style` and/or `depth` to customize the behavior. See Anchor Sections Customization section.

`section_divs` Wrap sections in `<div>` tags, and attach identifiers to the enclosing `<div>` rather than the header itself.

`fig_width` Default width (in inches) for figures

`fig_height` Default height (in inches) for figures

`fig_retina` Scaling to perform for retina displays (defaults to 2, which currently works for all widely used retina displays). Set to `NULL` to prevent retina scaling. Note that this will always be `NULL` when `keep_md` is specified (this is because `fig_retina` relies on outputting HTML directly into the markdown document).

`fig_caption` `TRUE` to render figures with captions

`dev` Graphics device to use for figure output (defaults to png)

`df_print` Method to be used for printing data frames. Valid values include "default", "kable", "tibble", and "paged". The "default" method uses a corresponding S3 method of `print`, typically `print.data.frame`. The "kable" method uses the `knitr::kable` function. The "tibble" method uses the **tibble** package to print a summary of the data frame. The "paged" method creates a paginated HTML table (note that this method is only valid for formats that produce HTML). In addition to the named methods you can also pass an arbitrary function to be used for printing data frames. You can disable the `df_print` behavior entirely by setting the option `rmarkdown.df_print` to `FALSE`. See Data frame printing section in bookdown book for examples.

`code_folding` Enable document readers to toggle the display of R code chunks. Specify `"none"` to display all code chunks. Specify `"hide"` or `"show"` to hide or show all R code chunks by default, and let readers toggle the states on browsers. See the *Code folding*

`code_download` Embed the Rmd source code within the document and provide a link that can be used by readers to download the code.

`self_contained` Produce a standalone HTML file with no external dependencies, using data: URIs to incorporate the contents of linked scripts, stylesheets, images, and videos. Note that even for self contained documents MathJax is still loaded externally (this is necessary because of its size).

`theme` One of the following:

- A `bslib::bs_theme()` object (or a list of `bslib::bs_theme()` argument values)
  - Use this option for custom themes using Bootstrap 4 or 3.

  - In this case, any `.scss/.sass` files provided to the `css` parameter may utilize the `theme`'s underlying Sass utilities (e.g., variables, mixins, etc).
- `NULL` for no theme (i.e., no `html_dependency_bootstrap()`).
- A character string specifying a Bootswatch 3 theme name (for backwards-compatibility).

`highlight` Syntax highlight engine and style. See the *Highlighting* section below for details.

"default" (and "textmate") will use highlightjs as syntax highlighting engine instead of Pandoc.

Any other value will be passed as Pandoc's highlighting style. Pandoc's built-in styles include "tango", "pygments", "kate", "monochrome", "espresso", "zenburn", "haddock" and "breezedark".

Two custom styles are also included, "arrow", an accessible color scheme, and "rstudio", which mimics the default IDE theme. Alternatively, supply a path to a '`.theme`' to use a custom Pandoc style. Note that custom theme requires Pandoc 2.0+.

Pass `NULL` to prevent syntax highlighting.

`highlight_downlit`

`TRUE` to use the **downlit** package as syntax highlight engine to highlight inline code and R code chunks (including providing hyperlinks to function documentation). The package needs to be installed to use this feature.

Only Pandoc color schemes are supported with this engine. With `highlight = "default"`, it will use the accessible theme called "arrow". To learn more about **downlit** highlighting engine, see `https://downlit.r-lib.org/`.

`math_method` Math rendering engine to use. This will define the math method to use with Pandoc.

- It can be a string for the engine, one of "mathjax", "mathml", "webtex", "katex", "gladtex", or "r-katex" or "default" for `mathjax`.
- It can be a list of
  - `engine`: one of "mathjax", "mathml", "webtex", "katex", or "gladtex".
  - `url`: A specific url to use with `mathjax`, `katex` or `webtex`. Note that for `engine = "mathjax"`, `url = "local"` will use a local version of MathJax (which is copied into the output directory).

For example,

```
output:
  html_document:
    math_method:
      engine: katex
      url: https://cdn.jsdelivr.net/npm/katex@0.11.1/dist
```

See Pandoc's Manual about Math in HTML for the details about Pandoc supported methods.

Using `math_method = "r-katex"` will opt-in server side rendering using KaTeX thanks to katex R package. This is useful compared to `math_method = "katex"` to have no JS dependency, only a CSS dependency for styling equation.

`mathjax` Include mathjax. The "default" option uses an https URL from a MathJax CDN. The "local" option uses a local version of MathJax (which is copied into the output directory). You can pass an alternate URL or pass `NULL` to exclude MathJax entirely.

`template` Pandoc template to use for rendering. Pass "default" to use the rmarkdown package default template; pass `NULL` to use pandoc's built-in template; pass a path to use a custom template that you've created. Note that if you don't use the "default" template then some features of `html_document` won't be available (see the Templates section below for more details).

`extra_dependencies` Extra dependencies as a list of the `html_dependency` class objects typically generated by `htmltools:htmlDependency()`.

`css` CSS and/or Sass files to include. Files with an extension of .sass or .scss are compiled to CSS via `sass::sass()`. Also, if `theme` is a `bslib::bs_theme()` object, Sass code may reference the relevant Bootstrap Sass variables, functions, mixins, etc.

`includes` Named list of additional content to include within the document (typically created using the `includes` function).

`keep_md` Keep the markdown file generated by knitting.

`lib_dir` Directory to copy dependent HTML libraries (e.g. jquery, bootstrap, etc.) into. By default this will be the name of the document with `_files` appended to it.

`md_extensions` Markdown extensions to be added or removed from the default definition of R Markdown. See the `rmarkdown_format` for additional details.

`pandoc_args` Additional command line options to pass to pandoc

`...` Additional function arguments to pass to the base R Markdown HTML output formatter `html_document_base`

Details

See the online documentation for additional details on using the `html_document` format.

R Markdown documents can have optional metadata that is used to generate a document header that includes the title, author, and date. For more details see the documentation on R Markdown metadata.

R Markdown documents also support citations. You can find more information on the markdown syntax for citations in the Bibliographies and Citations article in the online documentation.

Value

R Markdown output format to pass to `render`

Highlighting

There are three highlighting engines available to HTML documents:

highlightjs It does highlighting in the browser, using javascript It can only be used with the default template (i.e `template = "default"`) and it has two styles ("default" and "textmate"). When

activated, it adds two additional dependencies to the output file: a JS script and a CSS file. For now, this is the default engine for the default template - this could change in the future.

Pandoc Pandoc's built-in highlighting.engine works with any template, default or custom, and style can be chosen among the built-in ones ("tango", "pygments", "kate", "monochrome", "espresso", "zenburn", "haddock" and "breezedark") or a path to a custom theme ".theme" file (see Details in the Pandoc Manual). **rmarkdown** includes two custom themes to select with `highlight` parameter:

- "arrow", an accessible style using colors optimized for accessibility and color contrast
- "rstudio", a color scheme close to RStudio's default highlighting and highglightjs's textmate.

Custom themes are only available for Pandoc 2.0 and above.

downlit **downlit** is an R package that provides a syntax highlighting engine in R. It will also do automatic linking of R code (requires internet connectivity). It is activated only if `highlight_downlit = TRUE` and only affects R code, leaving highlighting for other languages unchanged. The default color scheme is the accessible theme "arrow".

It requires some CSS in the template to correctly style links. This is included in the default template, but if you want to use with a custom template, you will need to add this to your template:

```
$if(highlight-downlit)$
<style type="text/css">
  code a:any-link {
   color: inherit; /* use colour from syntax highlighting */
   text-decoration: underline;
   text-decoration-color: #ccc;
  }
 </style>
 $endif$
```

Anchor Sections Customization

This will be the default to activate anchor sections link on header

```
output:
  html_document:
    anchor_sections: TRUE
```

There are currently two options to modify the default behavior

`style` Select a predefined visual style:

- `style = "dash"`, the default, uses '#', a minimalist choice that evokes the id selector from HTML and CSS.
- `style = "symbol"` will use a link symbol
- `style = "icon"` will use an svg icon.

You can also customize using a css rule in your document. For example, to get a pictogram :

```
a.anchor-section::before {
  content: '\\01F517';
}
```

About how to apply custom CSS in R Markdown document, see https://bookdown.org/yihui/rmarkdown-cookbook/html-css.html

`depth` Select the maximum header level to add the anchor link to. For example, this yaml will use the symbol style and only with level 1 and 2 headings:

```
output:
  html_document:
    anchor_sections:
      style: icon
      depth: 2
```

If omitted, anchor will be added to all headers (equivalent of `depth=6`). You can also set anchors manually with `depth = 0` using this syntax

```
# my header {.hasAnchor}
```

Using anchor sections will add some CSS to your document output for the styling, and a JS script if `section_divs = TRUE`. The anchor link itself is added using a Lua filter, and hence requires Pandoc 2.0+

Navigation Bars

If you have a set of html documents which you'd like to provide a common global navigation bar for, you can include a "_navbar.yml" or "_navbar.html" file within the same directory as your html document and it will automatically be included at the top of the document.

The "_navbar.yml" file includes `title`, `type`, `left`, and `right` fields (to define menu items for the left and right of the navbar respectively). Menu items include `title` and `href` fields. For example:

```
title: "My Website"
type: default
left:
  - text: "Home"
    href: index.html
  - text: "Other"
    href: other.html
right:
  - text: GitHub
    href: https://github.com
```

The `type` field is optional and can take the value "default" or "inverse" (which provides a different color scheme for the navigation bar).

Alternatively, you can include a "_navbar.html" file which is a full HTML definition of a bootstrap navigation bar. For a simple example of including a navigation bar see https://github.com/rstudio/rmarkdown-website/blob/master/_navbar.html. For additional documentation on creating Bootstrap navigation bars see https://getbootstrap.com/docs/4.5/components/navbar/.

Floating Table of Contents

You may specify a list of options for the `toc_float` parameter which control the behavior of the floating table of contents. Options include:

- `collapsed` (defaults to `TRUE`) controls whether the table of contents appears with only the top-level (H2) headers. When collapsed the table of contents is automatically expanded inline when necessary.
- `smooth_scroll` (defaults to `TRUE`) controls whether page scrolls are animated when table of contents items are navigated to via mouse clicks.
- `print` (defaults to `TRUE`) controls whether the table of contents appears when user prints out the HTML page.

Code folding

Code blocks become foldable by specifying "show" or "hide" to the `code_folding` parameter. The state can be toggled individually on browsers. The document-wide toggle button is also provided for `html_document` and some of its extensions such as `html_notebook`. Note that this feature applies not only to source codes of chunks, but also markdown code blocks.

Supported languages are R, Python, Bash, SQL, C++, Stan, and Julia. To support code blocks with other languages, add `foldable` class to them (i.e., `class.source = "foldable"` as a chunk option).

The default initial state of code folding respects the value given to the `code_folding` parameter. To override the behavior individually, add `fold-none` to disable, `fold-hide` to initially hide, `fold-show` to initially show.

Tabbed Sections

You can organize content using tabs by applying the `.tabset` class attribute to headers within a document. This will cause all sub-headers of the header with the `.tabset` attribute to appear within tabs rather than as standalone sections. For example:

```
## Quarterly Results {.tabset}

### By Product

### By Region
```

With `html_document()`, you can also specify two additional attributes to control the appearance and behavior of the tabs. The `.tabset-fade` attributes causes the tabs to fade in and out when switching. The `.tabset-pills` attribute causes the visual appearance of the tabs to be "pill" rather than traditional tabs. For example:

```
## Quarterly Results {.tabset .tabset-fade .tabset-pills}
```

If tabbed sections relies on `html_dependency_tabset()`, for example by `html_vignette()`, these two attributes are not supported.

Templates

You can provide a custom HTML template to be used for rendering. The syntax for templates is described in the pandoc documentation. You can also use the basic pandoc template by passing `template = NULL`.

Note however that if you choose not to use the "default" HTML template then several aspects of HTML document rendering will behave differently:

- The `theme` parameter does not work (you can still provide styles using the `css` parameter).
- For the `highlight` parameter, the default highlighting engine will resolve to Pandoc instead of highlightjs and highlighting style will default to "pygments". "textmate" style is not available as related to highlightjs
- The `toc_float` parameter will not work.
- The `code_folding` parameter will not work.
- Tabbed sections (as described above) will not work.
- Navigation bars (as described above) will not work.
- MathJax will not work if `self_contained` is `TRUE` (these two options can't be used together in normal pandoc templates).

Due to the above restrictions, you might consider using the `includes` parameter as an alternative to providing a fully custom template.

Examples

```
## Not run:
library(rmarkdown)

render("input.Rmd", html_document())

render("input.Rmd", html_document(toc = TRUE))

## End(Not run)
```

`html_document_base` *Base output format for HTML-based output formats*

Description

Creates an HTML base output format suitable for passing as the `base_format` argument of the `output_format` function.

Usage

```
html_document_base(
  theme = NULL,
  self_contained = TRUE,
  lib_dir = NULL,
  math_method = "default",
  mathjax = "default",
  pandoc_args = NULL,
  template = "default",
  dependency_resolver = NULL,
  copy_resources = FALSE,
  extra_dependencies = NULL,
  css = NULL,
  bootstrap_compatible = FALSE,
  ...
)
```

Arguments

theme	One of the following: • A bslib::bs_theme() object (or a list of bslib::bs_theme() argument values) – Use this option for custom themes using Bootstrap 4 or 3. – In this case, any .scss/.sass files provided to the css parameter may utilize the theme's underlying Sass utilities (e.g., variables, mixins, etc). • NULL for no theme (i.e., no html_dependency_bootstrap()). • A character string specifying a Bootswatch 3 theme name (for backwards-compatibility).
self_contained	Produce a standalone HTML file with no external dependencies, using data: URIs to incorporate the contents of linked scripts, stylesheets, images, and videos. Note that even for self contained documents MathJax is still loaded externally (this is necessary because of its size).
lib_dir	Directory to copy dependent HTML libraries (e.g. jquery, bootstrap, etc.) into. By default this will be the name of the document with _files appended to it.
math_method	Math rendering engine to use. This will define the math method to use with Pandoc. • It can be a string for the engine, one of "mathjax", "mathml", "webtex", "katex", "gladtex", or "r-katex" or "default" for mathjax. • It can be a list of – engine: one of "mathjax", "mathml", "webtex", "katex", or "gladtex". – url: A specific url to use with mathjax, katex or webtex. Note that for engine = "mathjax", url = "local" will use a local version of MathJax (which is copied into the output directory). For example,

```
output:
  html_document:
    math_method:
      engine: katex
      url: https://cdn.jsdelivr.net/npm/katex@0.11.1/dist
```

See Pandoc's Manual about Math in HTML for the details about Pandoc supported methods.

Using `math_method = "r-katex"` will opt-in server side rendering using KaTeX thanks to katex R package. This is useful compared to `math_method = "katex"` to have no JS dependency, only a CSS dependency for styling equation.

`mathjax` Include mathjax. The "default" option uses an https URL from a MathJax CDN. The "local" option uses a local version of MathJax (which is copied into the output directory). You can pass an alternate URL or pass `NULL` to exclude MathJax entirely.

`pandoc_args` Additional command line options to pass to pandoc

`template` Pandoc template to use for rendering. Pass "default" to use the rmarkdown package default template; pass `NULL` to use pandoc's built-in template; pass a path to use a custom template that you've created. Note that if you don't use the "default" template then some features of `html_document` won't be available (see the Templates section below for more details).

`dependency_resolver`
A dependency resolver

`copy_resources` Copy resources

`extra_dependencies`
Extra dependencies as a list of the `html_dependency` class objects typically generated by `htmltools:htmlDependency()`.

`css` CSS and/or Sass files to include. Files with an extension of .sass or .scss are compiled to CSS via `sass::sass()`. Also, if `theme` is a `bslib::bs_theme()` object, Sass code may reference the relevant Bootstrap Sass variables, functions, mixins, etc.

`bootstrap_compatible`
Bootstrap compatible

`...` Ignored

Value

HTML base output format.

`html_fragment` *Convert to an HTML fragment.*

Description

An html fragment is suitable for inclusion into an external html page. See `html_document` for full details - this is a minor variation that assumes you will include the output into an existing document (e.g. a blog post).

Usage

```
html_fragment(
  number_sections = FALSE,
  section_divs = TRUE,
  fig_width = 7,
  fig_height = 5,
  fig_retina = 2,
  fig_caption = TRUE,
  dev = "png",
  df_print = "default",
  mathjax = TRUE,
  includes = NULL,
  keep_md = FALSE,
  md_extensions = NULL,
  pandoc_args = NULL,
  ...
)
```

Arguments

number_sections	TRUE to number section headings
section_divs	Wrap sections in tags, and attach identifiers to the enclosing rather than the header itself.
fig_width	Default width (in inches) for figures
fig_height	Default height (in inches) for figures
fig_retina	Scaling to perform for retina displays (defaults to 2, which currently works for all widely used retina displays). Set to NULL to prevent retina scaling. Note that this will always be NULL when keep_md is specified (this is because fig_retina relies on outputting HTML directly into the markdown document).
fig_caption	TRUE to render figures with captions
dev	Graphics device to use for figure output (defaults to png)
df_print	Method to be used for printing data frames. Valid values include "default", "kable", "tibble", and "paged". The "default" method uses a corresponding S3 method of print, typically print.data.frame. The "kable" method uses the knitr::kable function. The "tibble" method uses the **tibble** package to print a summary of the data frame. The "paged" method creates a paginated HTML table (note that this method is only valid for formats that produce HTML). In addition to the named methods you can also pass an arbitrary function to be used for printing data frames. You can disable the df_print behavior entirely by setting the option rmarkdown.df_print to FALSE. See Data frame printing section in bookdown book for examples.
mathjax	TRUE to convert $ and $$ math blocks into MathJax compatible output. Note that you'll still need to ensure that the page where the fragment is included loads the required MathJax scripts.

includes	Named list of additional content to include within the document (typically created using the includes function).
keep_md	Keep the markdown file generated by knitting.
md_extensions	Markdown extensions to be added or removed from the default definition of R Markdown. See the rmarkdown_format for additional details.
pandoc_args	Additional command line options to pass to pandoc
...	Additional arguments passed to html_document

Details

See the online documentation for additional details on using the html_fragment format.

Value

R Markdown output format to pass to render

---

html_notebook *Convert to an HTML notebook*

Description

Format for converting from R Markdown to an HTML notebook.

Usage

```
html_notebook(
  toc = FALSE,
  toc_depth = 3,
  toc_float = FALSE,
  number_sections = FALSE,
  fig_width = 7,
  fig_height = 5,
  fig_retina = 2,
  fig_caption = TRUE,
  code_folding = "show",
  theme = "default",
  highlight = "textmate",
  highlight_downlit = FALSE,
  math_method = "default",
  mathjax = "default",
  extra_dependencies = NULL,
  css = NULL,
  includes = NULL,
  md_extensions = NULL,
  pandoc_args = NULL,
  output_source = NULL,
```

```
  self_contained = TRUE,
  ...
)
```

Arguments

toc	TRUE to include a table of contents in the output
toc_depth	Depth of headers to include in table of contents
toc_float	TRUE to float the table of contents to the left of the main document content. Rather than TRUE you may also pass a list of options that control the behavior of the floating table of contents. See the *Floating Table of Contents* section below for details.
number_sections	TRUE to number section headings
fig_width	Default width (in inches) for figures
fig_height	Default height (in inches) for figures
fig_retina	Scaling to perform for retina displays (defaults to 2, which currently works for all widely used retina displays). Set to NULL to prevent retina scaling. Note that this will always be NULL when keep_md is specified (this is because fig_retina relies on outputting HTML directly into the markdown document).
fig_caption	TRUE to render figures with captions
code_folding	Enable document readers to toggle the display of R code chunks. Specify "none" to display all code chunks. Specify "hide" or "show" to hide or show all R code chunks by default, and let readers toggle the states on browsers. See the *Code folding*
theme	One of the following: • A bslib::bs_theme() object (or a list of bslib::bs_theme() argument values) – Use this option for custom themes using Bootstrap 4 or 3. – In this case, any .scss/.sass files provided to the css parameter may utilize the theme's underlying Sass utilities (e.g., variables, mixins, etc). • NULL for no theme (i.e., no html_dependency_bootstrap()). • A character string specifying a Bootswatch 3 theme name (for backwards-compatibility).
highlight	Syntax highlight engine and style. See the *Highlighting* section below for details. "default" (and "textmate") will use highlightjs as syntax highlighting engine instead of Pandoc. Any other value will be passed as Pandoc's highlighting style. Pandoc's built-in styles include "tango", "pygments", "kate", "monochrome", "espresso", "zenburn", "haddock" and "breezedark". Two custom styles are also included, "arrow", an accessible color scheme, and "rstudio", which mimics the default IDE theme. Alternatively, supply a path to a

'.theme' to use a custom Pandoc style. Note that custom theme requires Pandoc 2.0+.

Pass NULL to prevent syntax highlighting.

highlight_downlit

TRUE to use the **downlit** package as syntax highlight engine to highlight inline code and R code chunks (including providing hyperlinks to function documentation). The package needs to be installed to use this feature.

Only Pandoc color schemes are supported with this engine. With highlight = "default", it will use the accessible theme called "arrow". To learn more about **downlit** highlighting engine, see https://downlit.r-lib.org/.

math_method Math rendering engine to use. This will define the math method to use with Pandoc.

- It can be a string for the engine, one of "mathjax", "mathml", "webtex", "katex", "gladtex", or "r-katex" or "default" for mathjax.
- It can be a list of
  - engine: one of "mathjax", "mathml", "webtex", "katex", or "gladtex".
  - url: A specific url to use with mathjax, katex or webtex. Note that for engine = "mathjax", url = "local" will use a local version of MathJax (which is copied into the output directory).

For example,

```
output:
  html_document:
    math_method:
      engine: katex
      url: https://cdn.jsdelivr.net/npm/katex@0.11.1/dist
```

See Pandoc's Manual about Math in HTML for the details about Pandoc supported methods.

Using math_method = "r-katex" will opt-in server side rendering using KaTeX thanks to katex R package. This is useful compared to math_method = "katex" to have no JS dependency, only a CSS dependency for styling equation.

mathjax Include mathjax. The "default" option uses an https URL from a MathJax CDN. The "local" option uses a local version of MathJax (which is copied into the output directory). You can pass an alternate URL or pass NULL to exclude MathJax entirely.

extra_dependencies

Extra dependencies as a list of the html_dependency class objects typically generated by htmltools:htmlDependency().

css CSS and/or Sass files to include. Files with an extension of .sass or .scss are compiled to CSS via sass::sass(). Also, if theme is a bslib::bs_theme() object, Sass code may reference the relevant Bootstrap Sass variables, functions, mixins, etc.

includes Named list of additional content to include within the document (typically created using the includes function).

md_extensions	Markdown extensions to be added or removed from the default definition of R Markdown. See the rmarkdown_format for additional details.
pandoc_args	Additional command line options to pass to pandoc
output_source	Define an output source for R chunks (ie, outputs to use instead of those produced by evaluating the underlying R code). See html_notebook_output for more details.
self_contained	Produce a standalone HTML file with no external dependencies. Defaults to TRUE. In notebooks, setting this to FALSE is not recommended, since the setting does not apply to embedded notebook output such as plots and HTML widgets.
...	Additional function arguments to pass to the base R Markdown HTML output formatter html_document_base

Details

See the online documentation for additional details on using the html_notebook format.

---

html_notebook_metadata

Generate R Notebook Metadata

Description

A structured helper for the construction of metadata used by the R Notebook output functions. See html_notebook_output for more details.

Usage

```
html_notebook_metadata(iframe = TRUE)
```

Arguments

iframe	Boolean; should output be shown in an <iframe>?

---

html_notebook_output *Generate R Notebook Output*

Description

Utilities for generating output for the html_notebook format, through the output_source function attached to a output_format.

Usage

```
html_notebook_output_html(html, meta = NULL)

html_notebook_output_img(
  path = NULL,
  bytes = NULL,
  attributes = NULL,
  meta = NULL,
  format = c("png", "jpeg")
)

html_notebook_output_png(
  path = NULL,
  bytes = NULL,
  attributes = NULL,
  meta = NULL,
  format = c("png", "jpeg")
)

html_notebook_output_code(code, attributes = list(class = "r"), meta = NULL)
```

Arguments

`html`	Arbitrary HTML content to insert.
`meta`	An R list of arbitrary meta-data. The data will be converted to JSON, base64-encoded, and injected into the header comment.
`path`	A path to a file. For functions accepting both `path` and `bytes`, if `bytes` is `NULL`, the bytewise contents will be obtained by reading the file.
`bytes`	The bytewise representation of content.
`attributes`	A named R list of HTML attributes. These will be escaped and inserted into the generated HTML as appropriate.
`format`	The image format; one of `"png"` or `"jpeg"`.
`code`	Source code.

Details

See the online documentation for additional details on using the `html_notebook` format.

`html_vignette` *Convert to an HTML vignette*

Description

A HTML vignette is a lightweight alternative to `html_document()` suitable for inclusion in packages to be released to CRAN. It reduces the size of a basic vignette from 100k to around 10k.

Usage

```
html_vignette(
  fig_width = 3,
  fig_height = 3,
  dev = "png",
  df_print = "default",
  css = NULL,
  highlight = "pygments",
  keep_md = FALSE,
  readme = FALSE,
  self_contained = TRUE,
  tabset = FALSE,
  code_folding = c("none", "show", "hide"),
  extra_dependencies = NULL,
  pandoc_args = NULL,
  ...
)
```

Arguments

fig_width	Default width (in inches) for figures
fig_height	Default height (in inches) for figures
dev	Graphics device to use for figure output (defaults to png)
df_print	Method to be used for printing data frames. Valid values include "default", "kable", "tibble", and "paged". The "default" method uses a corresponding S3 method of `print`, typically `print.data.frame`. The "kable" method uses the `knitr::kable` function. The "tibble" method uses the **tibble** package to print a summary of the data frame. The "paged" method creates a paginated HTML table (note that this method is only valid for formats that produce HTML). In addition to the named methods you can also pass an arbitrary function to be used for printing data frames. You can disable the `df_print` behavior entirely by setting the option `rmarkdown.df_print` to `FALSE`. See Data frame printing section in bookdown book for examples.
css	One or more css files to include.
highlight, ...	Additional arguments passed to `html_document()`. Please note that `theme` and `fig_retina` are hard-coded. Setting any of those will yield an error.
keep_md	Keep the markdown file generated by knitting.
readme	Use this vignette as the package README.md file (i.e. render it as README.md to the package root). Note that if there are image files within your vignette you should be sure to add '`README_files`' to '`.Rbuildignore`'.
self_contained	Produce a standalone HTML file with no external dependencies, using data: URIs to incorporate the contents of linked scripts, stylesheets, images, and videos. Note that even for self contained documents MathJax is still loaded externally (this is necessary because of its size).

`tabset`	Opt-in tabbed-sections feature inspired by `html_document()`. See section "Tabbed Sections" for the detail. This feature also allows navigation to the tab from table of contents and URL.
`code_folding`	Enable document readers to toggle the display of R code chunks. Specify `"none"` to display all code chunks. Specify `"hide"` or `"show"` to hide or show all R code chunks by default, and let readers toggle the states on browsers. See the *Code folding*
`extra_dependencies`	Extra dependencies as a list of the `html_dependency` class objects typically generated by `htmltools:htmlDependency()`.
`pandoc_args`	Additional command line options to pass to pandoc

Details

Compared to `html_document()`, it:

- never uses retina figures
- never uses a theme
- has a smaller default figure size
- uses a custom css stylesheet

See the online documentation for additional details on using the `html_vignette()` format.

Value

R Markdown output format to pass to `render()`

Tabbed Sections

You can organize content using tabs by applying the `.tabset` class attribute to headers within a document. This will cause all sub-headers of the header with the `.tabset` attribute to appear within tabs rather than as standalone sections. For example:

```
## Quarterly Results {.tabset}

### By Product

### By Region
```

With `html_document()`, you can also specify two additional attributes to control the appearance and behavior of the tabs. The `.tabset-fade` attributes causes the tabs to fade in and out when switching. The `.tabset-pills` attribute causes the visual appearance of the tabs to be "pill" rather than traditional tabs. For example:

```
## Quarterly Results {.tabset .tabset-fade .tabset-pills}
```

If tabbed sections relies on `html_dependency_tabset()`, for example by `html_vignette()`, these two attributes are not supported.

Code folding

Code blocks become foldable by specifying "show" or "hide" to the `code_folding` parameter. The state can be toggled individually on browsers. The document-wide toggle button is also provided for `html_document` and some of its extensions such as `html_notebook`. Note that this feature applies not only to source codes of chunks, but also markdown code blocks.

Supported languages are R, Python, Bash, SQL, C++, Stan, and Julia. To support code blocks with other languages, add `foldable` class to them (i.e., `class.source = "foldable"` as a chunk option).

The default initial state of code folding respects the value given to the `code_folding` parameter. To override the behavior individually, add `fold-none` to disable, `fold-hide` to initially hide, `fold-show` to initially show.

`includes` *Include content within output*

Description

Specify additional content to be included within an output document.

Usage

```
includes(in_header = NULL, before_body = NULL, after_body = NULL)

includes_to_pandoc_args(includes, filter = identity)
```

Arguments

`in_header`	One or more files with content to be included in the header of the document.
`before_body`	One or more files with content to be included before the document body.
`after_body`	One or more files with content to be included after the document body.
`includes`	Includes to convert to pandoc args.
`filter`	Filter to pre-process includes with.

Details

Non-absolute paths for resources referenced from the `in_header`, `before_body`, and `after_body` parameters are resolved relative to the directory of the input document.

Value

Includes list or pandoc args

Examples

```
## Not run:
library(rmarkdown)

html_document(includes = includes(before_body = "header.htm"))

pdf_document(includes = includes(after_body = "footer.tex"))

## End(Not run)
```

`ioslides_presentation` *Convert to an ioslides Presentation*

Description

Format for converting from R Markdown to an ioslides presentation.

Usage

```
ioslides_presentation(
  number_sections = FALSE,
  logo = NULL,
  slide_level = 2,
  incremental = FALSE,
  fig_width = 7.5,
  fig_height = 4.5,
  fig_retina = 2,
  fig_caption = TRUE,
  dev = "png",
  df_print = "default",
  smart = TRUE,
  self_contained = TRUE,
  widescreen = FALSE,
  smaller = FALSE,
  transition = "default",
  math_method = "mathjax",
  mathjax = "default",
  analytics = NULL,
  template = NULL,
  css = NULL,
  includes = NULL,
  keep_md = FALSE,
  lib_dir = NULL,
  md_extensions = NULL,
  pandoc_args = NULL,
  extra_dependencies = NULL,
  ...
)
```

Arguments

number_sections	TRUE to number section headings
logo	Path to file that includes a logo for use in the presentation (should be square and at least 128x128).
slide_level	Header level to consider as slide separator (Defaults to header 2).
incremental	TRUE to render slide bullets incrementally. Note that if you want to reverse the default incremental behavior for an individual bullet you can preceded it with >. For example: `> - Bullet Text`.
fig_width	Default width (in inches) for figures
fig_height	Default height (in inches) for figures
fig_retina	Scaling to perform for retina displays (defaults to 2, which currently works for all widely used retina displays). Set to NULL to prevent retina scaling. Note that this will always be NULL when keep_md is specified (this is because fig_retina relies on outputting HTML directly into the markdown document).
fig_caption	TRUE to render figures with captions
dev	Graphics device to use for figure output (defaults to png)
df_print	Method to be used for printing data frames. Valid values include "default", "kable", "tibble", and "paged". The "default" method uses a corresponding S3 method of print, typically print.data.frame. The "kable" method uses the knitr::kable function. The "tibble" method uses the **tibble** package to print a summary of the data frame. The "paged" method creates a paginated HTML table (note that this method is only valid for formats that produce HTML). In addition to the named methods you can also pass an arbitrary function to be used for printing data frames. You can disable the df_print behavior entirely by setting the option rmarkdown.df_print to FALSE. See Data frame printing section in bookdown book for examples.
smart	Produce typographically correct output, converting straight quotes to curly quotes, --- to em-dashes, -- to en-dashes, and ... to ellipses.
self_contained	Produce a standalone HTML file with no external dependencies, using data: URIs to incorporate the contents of linked scripts, stylesheets, images, and videos. Note that even for self contained documents MathJax is still loaded externally (this is necessary because of its size).
widescreen	Display presentation with wider dimensions.
smaller	Use smaller text on all slides. You can also enable this for individual slides by adding the .smaller attribute to the slide header (see *Presentation Size* below for details).
transition	Speed of slide transitions. This can be "default", "slower", "faster", or a numeric value with a number of seconds (e.g. 0.5).
math_method	Math rendering engine to use. This will define the math method to use with Pandoc. • It can be a string for the engine, one of "mathjax", "mathml", "webtex", "katex", "gladtex", or "r-katex" or "default" for mathjax.

- It can be a list of
  - `engine`: one of "mathjax", "mathml", "webtex", "katex", or "gladtex".
  - `url`: A specific url to use with `mathjax`, `katex` or `webtex`. Note that for `engine = "mathjax"`, `url = "local"` will use a local version of MathJax (which is copied into the output directory).

For example,

```
output:
  html_document:
    math_method:
      engine: katex
      url: https://cdn.jsdelivr.net/npm/katex@0.11.1/dist
```

See Pandoc's Manual about Math in HTML for the details about Pandoc supported methods.

Using `math_method = "r-katex"` will opt-in server side rendering using KaTeX thanks to katex R package. This is useful compared to `math_method = "katex"` to have no JS dependency, only a CSS dependency for styling equation.

`mathjax` Include mathjax. The "default" option uses an https URL from a MathJax CDN. The "local" option uses a local version of MathJax (which is copied into the output directory). You can pass an alternate URL or pass `NULL` to exclude MathJax entirely.

`analytics` A Google analytics property ID.

`template` Pandoc template to use for rendering. Pass "default" to use the rmarkdown package default template; pass `NULL` to use pandoc's built-in template; pass a path to use a custom template that you've created. Note that if you don't use the "default" template then some features of `html_document` won't be available (see the Templates section below for more details).

`css` One or more css files to include.

`includes` Named list of additional content to include within the document (typically created using the `includes` function).

`keep_md` Keep the markdown file generated by knitting.

`lib_dir` Directory to copy dependent HTML libraries (e.g. jquery, bootstrap, etc.) into. By default this will be the name of the document with `_files` appended to it.

`md_extensions` Markdown extensions to be added or removed from the default definition of R Markdown. See the `rmarkdown_format` for additional details.

`pandoc_args` Additional command line options to pass to pandoc

`extra_dependencies`
Extra dependencies as a list of the `html_dependency` class objects typically generated by `htmltools:htmlDependency()`.

`...` Additional function arguments to pass to the base R Markdown HTML output formatter `html_document_base`

Details

See the online documentation for additional details on using the `ioslides_presentation` format.

Note that, if a `before_body` include is specified in `includes`, then it will replace the standard title slide entirely.

Regarding previewing slide in RStudio IDE, `ioslides_presentation()` will always open preview in a new Window and the RStudio IDE configuration "Open in Viewer Pane" will have no effect for this format.

Value

R Markdown output format to pass to `render()`.

Slide Basics

You can create a slide show broken up into sections by using the # and ## heading tags (you can also create a new slide without a header using a horizontal rule (`----------`). For example here's a simple slide show:

```
---
title: "Habits"
author: John Doe
date: March 22, 2005
output: ioslides_presentation
---

# In the morning

## Getting up

- Turn off alarm
- Get out of bed

## Breakfast

- Eat eggs
- Drink coffee

# In the evening

## Dinner

- Eat spaghetti
- Drink wine

----------

![picture of spaghetti](images/spaghetti.jpg)
```

```
## Going to sleep

- Get in bed
- Count sheep
```

You can add a subtitle to a slide or section by including text after the pipe (|) character. For example:

```
## Getting up | What I like to do first thing
```

Display Modes

The following single character keyboard shortcuts enable alternate display modes:

`'f'` enable fullscreen mode

`'w'` toggle widescreen mode

`'o'` enable overview mode

`'h'` enable code highlight mode

`'p'` show presenter notes

Pressing `Esc` exits all of these modes. See the sections below on *Code Highlighting* and *Presenter Mode* for additional detail on those modes.

Incremental Bullets

You can render bullets incrementally by adding the `incremental` option:

```
---
output:
  ioslides_presentation:
    incremental: true
---
```

If you want to render bullets incrementally for some slides but not others you can use this syntax:

```
> - Eat eggs
> - Drink coffee
```

Presentation Size

You can display the presentation using a wider form factor using the `widescreen` option. You can specify that smaller text be used with the `smaller` option. For example:

```
---
output:
  ioslides_presentation:
    widescreen: true
    smaller: true
---
```

You can also enable the `smaller` option on a slide-by-slide basis by adding the `.smaller` attribute to the slide header:

```
## Getting up {.smaller}
```

Adding a Logo

You can add a logo to the presentation using the `logo` option (the logo should be square and at least 128x128). For example:

```
---
output:
  ioslides_presentation:
    logo: logo.png
---
```

A 128x128 version of the logo graphic will be added to the title slide and an icon version of the logo will be included in the bottom-left footer of each slide.

Build Slides

Slides can also have a `.build` attribute that indicate that their content should be displayed incrementally. For example:

```
## Getting up {.build}
```

Slide attributes can be combined if you need to specify more than one, for example:

```
## Getting up {.smaller .build}
```

Code Highlighting

It's possible to select subsets of code for additional emphasis by adding a special "highlight" comment around the code. For example:

```
### <b>
x <- 10
y <- x * 2
### </b>
```

The highlighted region will be displayed with a bold font. When you want to help the audience focus exclusively on the highlighted region press the `'h'` key and the rest of the code will fade away.

Tables

The ioslides template has an attractive default style for tables so you shouldn't hesitate to add tables for presenting more complex sets of information. Pandoc markdown supports several syntaxes for defining tables which are described in the pandoc online documentation.

Advanced Layout

You can center content on a slide by adding the `.flexbox` and `.vcenter` attributes to the slide title. For example:

```
## Dinner {.flexbox .vcenter}
```

You can horizontally center content by enclosing it in a `div` tag with class `centered`. For example:

```
<div class="centered">
This text is centered.
</div>
```

You can do a two-column layout using the `columns-2` class. For example:

```
<div class="columns-2">
  ![Image](image.png)

  - Bullet 1
  - Bullet 2
  - Bullet 3
</div>
```

Note that content will flow across the columns so if you want to have an image on one side and text on the other you should make sure that the image has sufficient height to force the text to the other side of the slide.

Text Color

You can color content using base color classes red, blue, green, yellow, and gray (or variations of them e.g. red2, red3, blue2, blue3, etc.). For example:

```
<div class="red2">
This text is red
</div>
```

Presenter Mode

A separate presenter window can also be opened (ideal for when you are presenting on one screen but have another screen that's private to you). The window stays in sync with the main presentation window and also shows presenter notes and a thumbnail of the next slide. To enable presenter mode add `?presentme=true` to the URL of the presentation, for example:

```
mypresentation.html?presentme=true
```

The presenter mode window will open and will always re-open with the presentation until it's disabled with:

```
mypresentation.html?presentme=false
```

To add presenter notes to a slide you include it within a "notes" `div`. For example:

```
<div class="notes">
This is my *note*.

- It can contain markdown
- like this list

</div>
```

Printing and PDF Output

You can print an ioslides presentation from within browsers that have good support for print CSS (i.e. as of this writing Google Chrome has the best support). Printing maintains most of the visual styles of the HTML version of the presentation.

To create a PDF version of a presentation you can use Print to PDF from Google Chrome.

---

knitr_options *Knitr options for an output format*

Description

Define the knitr options for an R Markdown output format.

Usage

```
knitr_options(
  opts_knit = NULL,
  opts_chunk = NULL,
  knit_hooks = NULL,
  opts_hooks = NULL,
  opts_template = NULL
)
```

Arguments

`opts_knit`	List of package level knitr options (see `opts_knit`)
`opts_chunk`	List of chunk level knitr options (see `opts_chunk`)
`knit_hooks`	List of hooks for R code chunks, inline R code, and output (see `knit_hooks`)
`opts_hooks`	List of hooks for code chunk options (see `opts_hooks`)
`opts_template`	List of templates for chunk level knitr options (see `opts_template`)

Value

An list that can be passed as the `knitr` argument of the `output_format` function.

See Also

output_format

`knitr_options_html` *Knitr options for an HTML output format*

Description

Define knitr options for an R Markdown output format that creates HTML output.

Usage

```
knitr_options_html(fig_width, fig_height, fig_retina, keep_md, dev = "png")
```

Arguments

`fig_width`	Default width (in inches) for figures
`fig_height`	Default height (in inches) for figures
`fig_retina`	Scaling to perform for retina displays (defaults to 2, which currently works for all widely used retina displays). Set to `NULL` to prevent retina scaling. Note that this will always be `NULL` when `keep_md` is specified (this is because `fig_retina` relies on outputting HTML directly into the markdown document).
`keep_md`	Keep the markdown file generated by knitting.
`dev`	Graphics device to use for figure output (defaults to png)

Value

An list that can be passed as the `knitr` argument of the `output_format` function.

See Also

knitr_options, output_format

knitr_options_pdf *Knitr options for a PDF output format*

Description

Define knitr options for an R Markdown output format that creates PDF output.

Usage

```
knitr_options_pdf(fig_width, fig_height, fig_crop, dev = "pdf")
```

Arguments

fig_width	Default width (in inches) for figures
fig_height	Default height (in inches) for figures
fig_crop	Whether to crop PDF figures with the command pdfcrop. This requires the tools pdfcrop and ghostscript to be installed. By default, fig_crop = TRUE if these two tools are available.
dev	Graphics device to use for figure output (defaults to png)

Value

An list that can be passed as the knitr argument of the output_format function.

See Also

knitr_options, output_format

knit_params_ask *Run a shiny application asking for parameter configuration for the given document.*

Description

Run a shiny application asking for parameter configuration for the given document.

Usage

```
knit_params_ask(
  file = NULL,
  input_lines = NULL,
  params = NULL,
  shiny_args = NULL,
  save_caption = "Save",
  encoding = "UTF-8"
)
```

Arguments

file	Path to the R Markdown document with configurable parameters.
input_lines	Content of the R Markdown document. If NULL, the contents of file will be read.
params	A named list of optional parameter overrides used in place of the document defaults.
shiny_args	Additional arguments to runApp.
save_caption	Caption to use use for button that saves/confirms parameters.
encoding	Ignored. The encoding is always assumed to be UTF-8.

Value

named list with overridden parameter names and value.

---

latex-dependencies *Provide common LaTeX dependencies*

Description

These functions provide common LaTeX dependencies (e.g. tikz) for R Markdown formats that use LaTeX.

Usage

```
latex_dependency_tikz(libraries, options = NULL, extra_lines = NULL)
```

Arguments

libraries	A character vector of tikz libraries to load
options	The LaTeX options for the package
extra_lines	LaTeX code related to the package added to the preamble

latex_dependency *Define a LaTeX package dependency*

Description

Define a LaTeX package dependency

Usage

```
latex_dependency(name, options = NULL, extra_lines = NULL)
```

Arguments

name	The LaTeX package name
options	The LaTeX options for the package
extra_lines	LaTeX code related to the package added to the preamble

md_document *Convert to a markdown document*

Description

Format for converting from R Markdown to another variant of markdown (e.g. strict markdown or github flavored markdown)

Usage

```
md_document(
  variant = "markdown_strict",
  preserve_yaml = FALSE,
  toc = FALSE,
  toc_depth = 3,
  number_sections = FALSE,
  standalone = FALSE,
  fig_width = 7,
  fig_height = 5,
  fig_retina = NULL,
  dev = "png",
  df_print = "default",
  includes = NULL,
  md_extensions = NULL,
  pandoc_args = NULL,
  ext = ".md"
)
```

Arguments

variant	Markdown variant to produce (defaults to "markdown_strict"). Other valid values are "commonmark", "gfm", "commonmark_x", "markdown_mmd", markdown_phpextra", "markdown_github", or even "markdown" (which produces pandoc markdown). You can also compose custom markdown variants, see the pandoc online documentation for details.
preserve_yaml	Preserve YAML front matter in final document.
toc	TRUE to include a table of contents in the output
toc_depth	Depth of headers to include in table of contents
number_sections	TRUE to number section headings
standalone	Set to TRUE to include title, date and other metadata field in addition to Rmd content as a body.
fig_width	Default width (in inches) for figures
fig_height	Default height (in inches) for figures
fig_retina	Scaling to perform for retina displays. Defaults to NULL which performs no scaling. A setting of 2 will work for all widely used retina displays, but will also result in the output of <img> tags rather than markdown images due to the need to set the width of the image explicitly.
dev	Graphics device to use for figure output (defaults to png)
df_print	Method to be used for printing data frames. Valid values include "default", "kable", "tibble", and "paged". The "default" method uses a corresponding S3 method of print, typically print.data.frame. The "kable" method uses the knitr::kable function. The "tibble" method uses the **tibble** package to print a summary of the data frame. The "paged" method creates a paginated HTML table (note that this method is only valid for formats that produce HTML). In addition to the named methods you can also pass an arbitrary function to be used for printing data frames. You can disable the df_print behavior entirely by setting the option rmarkdown.df_print to FALSE. See Data frame printing section in bookdown book for examples.
includes	Named list of additional content to include within the document (typically created using the includes function).
md_extensions	Markdown extensions to be added or removed from the default definition of R Markdown. See the rmarkdown_format for additional details.
pandoc_args	Additional command line options to pass to pandoc
ext	Extension of the output file (defaults to ".md").

Details

See the online documentation for additional details on using the md_document() format.

R Markdown documents can have optional metadata that is used to generate a document header that includes the title, author, and date. For more details see the documentation on R Markdown metadata.

Value

R Markdown output format to pass to `render()`

Examples

```
## Not run:
library(rmarkdown)

render("input.Rmd", md_document())

render("input.Rmd", md_document(variant = "markdown_github"))

## End(Not run)
```

`metadata` *The YAML metadata of the current R Markdown document*

Description

The object `metadata` stores the YAML metadata of the current R Markdown document as a list, which you may use in the R code chunks, e.g. `rmarkdown::metadata$title` (the title of the document), `rmarkdown::metadata$author`, and `rmarkdown::metadata$foo` (if you have a YAML field named `foo`), etc.

Format

An object of class `list` of length 0.

Examples

```
rmarkdown::metadata
```

`odt_document` *Convert to an OpenDocument Text (ODT) document*

Description

Format for converting from R Markdown to an ODT document.

Usage

```
odt_document(
  number_sections = FALSE,
  fig_width = 5,
  fig_height = 4,
  fig_caption = TRUE,
  template = "default",
  reference_odt = "default",
  includes = NULL,
  keep_md = FALSE,
  md_extensions = NULL,
  pandoc_args = NULL
)
```

Arguments

number_sections	TRUE to number section headings
fig_width	Default width (in inches) for figures
fig_height	Default height (in inches) for figures
fig_caption	TRUE to render figures with captions
template	Pandoc template to use for rendering. Pass "default" to use the rmarkdown package default template; pass NULL to use pandoc's built-in template; pass a path to use a custom template that you've created. See the documentation on pandoc online documentation for details on creating custom templates.
reference_odt	Use the specified file as a style reference in producing an odt file. For best results, the reference odt should be a modified version of an odt file produced using pandoc. Pass "default" to use the rmarkdown default styles.
includes	Named list of additional content to include within the document (typically created using the includes function).
keep_md	Keep the markdown file generated by knitting.
md_extensions	Markdown extensions to be added or removed from the default definition of R Markdown. See the rmarkdown_format for additional details.
pandoc_args	Additional command line options to pass to pandoc

Details

See the online documentation for additional details on using the odt_document format.

R Markdown documents can have optional metadata that is used to generate a document header that includes the title, author, and date. For more details see the documentation on R Markdown metadata.

R Markdown documents also support citations. You can find more information on the markdown syntax for citations in the Bibliographies and Citations article in the online documentation.

Value

R Markdown output format to pass to render

Examples

```
## Not run:
library(rmarkdown)

# simple invocation
render("input.Rmd", odt_document())

# specify an option for syntax highlighting
render("input.Rmd", odt_document(highlight = "zenburn"))

## End(Not run)
```

---

output_format *Define an R Markdown output format*

Description

Define an R Markdown output format based on a combination of knitr and pandoc options.

Usage

```
output_format(
  knitr,
  pandoc,
  keep_md = FALSE,
  clean_supporting = TRUE,
  df_print = NULL,
  pre_knit = NULL,
  post_knit = NULL,
  pre_processor = NULL,
  intermediates_generator = NULL,
  post_processor = NULL,
  on_exit = NULL,
  file_scope = NULL,
  base_format = NULL
)
```

Arguments

knitr	Knitr options for an output format (see knitr_options)
pandoc	Pandoc options for an output format (see pandoc_options)
keep_md	Keep the markdown file generated by knitting. Note that if this is TRUE then clean_supporting will always be FALSE.

`clean_supporting`
: Cleanup any supporting files after conversion see `render_supporting_files`

`df_print`
: Method to be used for printing data frames. Valid values include "default", "kable", "tibble", and "paged". The "default" method uses a corresponding S3 method of `print`, typically `print.data.frame`. The "kable" method uses the `knitr::kable` function. The "tibble" method uses the **tibble** package to print a summary of the data frame. The "paged" method creates a paginated HTML table (note that this method is only valid for formats that produce HTML). In addition to the named methods you can also pass an arbitrary function to be used for printing data frames. You can disable the `df_print` behavior entirely by setting the option `rmarkdown.df_print` to `FALSE`. See Data frame printing section in bookdown book for examples.

`pre_knit`
: An optional function that runs before knitting which receives the `input` (input filename passed to `render`), `metadata` (the parsed front matter of the Rmd file) and `...` (for future expansion) arguments. This function can be used to add side effects before knitting step.

`post_knit`
: An optional function that runs after knitting which receives the `metadata`, `input_file`, `runtime`, and `...` (for future expansion) arguments. This function can return additional arguments to pass to pandoc and can call `knitr::knit_meta_add` to add additional dependencies based on the contents of the input_file or on other assets side by side with it that may be used to produce html with dependencies during subsequent processing.

`pre_processor`
: An optional pre-processor function that receives the `metadata`, `input_file`, `runtime`, `knit_meta`, `files_dir`, and `output_dir` and can return additional arguments to pass to pandoc.

`intermediates_generator`
: An optional function that receives the original `input_file`, and the intermediates directory (i.e. the `intermediates_dir` argument to `render`). The function should generate and return the names of any intermediate files required to render the `input_file`.

`post_processor`
: An optional post-processor function that receives the `metadata`, `input_file`, `output_file`, `clean`, and `verbose` parameters, and can return an alternative `output_file`.

`on_exit`
: A function to call when `rmarkdown::render()` finishes execution (as registered with a `on.exit` handler).

`file_scope`
: A function that will split markdown input to pandoc into multiple named files. This is useful when the caller has concatenated a set of Rmd files together (as **bookdown** does), and those files may need to processed by pandoc using the `--file-scope` option. The first argument is input file paths and the second is `NULL` or current file scope which is a named list of files w/ `name` and `content` for each file. The return is the new file scope. Also, the arguments should include `...` for the future extensions.

`base_format`
: An optional format to extend.

Value

An R Markdown output format definition that can be passed to `render`.

See Also

render, knitr_options, pandoc_options

Examples

```
## Not run:
output_format(knitr = knitr_options(opts_chunk = list(dev = 'png')),
              pandoc = pandoc_options(to = "html"))

## End(Not run)
```

`output_format_dependency` *Define and merge an R Markdown's output format dependency*

Description

Define and merge a dependency such as pre/post-processors from within chunks. The merge happens explicitly when a list of dependencies are passed to `knitr::knit_meta_add()` or implicitly when a dependency is `knitr::knit_printed`. Defining a function that does the former is the best way for package developers to share the dependency. On the contrary, the latter is useful to declare a document-specific dependency. This function shares some arguments with `output_format`, but lacks the others because dependency is resolved after `post_knit` and before `pre_processor`.

Usage

```
output_format_dependency(
  name,
  pandoc = list(),
  pre_processor = NULL,
  post_processor = NULL,
  file_scope = NULL,
  on_exit = NULL
)
```

Arguments

`name`	A dependency name. If some dependencies share the same name, then only the first one will be merged to the output format.
`pandoc`	Pandoc options for an output format (see `pandoc_options`)
`pre_processor`	An optional pre-processor function that receives the `metadata`, `input_file`, `runtime`, `knit_meta`, `files_dir`, and `output_dir` and can return additional arguments to pass to pandoc.
`post_processor`	An optional post-processor function that receives the `metadata`, `input_file`, `output_file`, `clean`, and `verbose` parameters, and can return an alternative `output_file`.

`file_scope` A function that will split markdown input to pandoc into multiple named files. This is useful when the caller has concatenated a set of Rmd files together (as **bookdown** does), and those files may need to processed by pandoc using the `--file-scope` option. The first argument is input file paths and the second is `NULL` or current file scope which is a named list of files w/ `name` and `content` for each file. The return is the new file scope. Also, the arguments should include `...` for the future extensions.

`on_exit` A function to call when `rmarkdown::render()` finishes execution (as registered with a `on.exit` handler).

Value

An list of arguments with the "rmd_dependency" class.

Examples

```
# Implicitly add lua filters from within a chunk
# This relies on (implicit) printing of the dependency in a chunk via
# knitr::knit_print()`
output_format_dependency(
  "lua_filter1",
  pandoc = list(lua_filters = "example1.lua")
)

# Explicitly add lua filters from within a chunk
knitr::knit_meta_add(list(output_format_dependency(
  "lua_filter2",
  pandoc = list(lua_filters = "example2.lua")
)))

# List the available dependencies
# Note that the list may include dependencies with duplicated names. In that
# case, the first one is merged to the output format and the others are
# discarded.
str(knitr::knit_meta("output_format_dependency", clean = FALSE))
```

`output_metadata` *The output metadata object*

Description

This object provides a mechanism for users to attach metadata as an attribute (named `rmd_output_metadata`) of the returned value of `render()`. The initial value of the metadata comes from in the `rmd_output_metadata` field of the YAML frontmatter of an R Markdown document. The metadata can be queried via the `output_metadata$get()` method, and modified via the `output_metadata$set()` method.

paged_table *Create a table in HTML with support for paging rows and columns*

Description

Create a table in HTML with support for paging rows and columns

Usage

```
paged_table(x, options = NULL)
```

Arguments

x	a data frame to be rendered as a paged table.
options	options for printing the paged table. See details for specifics.

Details

Below are the recognized table pagination options.

Option	Description	Default
max.print	The number of rows to print.	1000
sql.max.print	The number of rows to print from a SQL data table.	1000
rows.print	The number of rows to display.	10
cols.print	The number of columns to display.	10
cols.min.print	The minimum number of columns to display.	-
pages.print	The number of pages to display under page navigation.	-
paged.print	When set to FALSE turns off paged tables.	TRUE
rownames.print	When set to FALSE turns off row names.	TRUE

Note: There is a hard cap of 10,000 rows to ensure that pandoc will not fail when rendering the document.

pandoc_args *Functions for generating pandoc command line arguments*

Description

Functions that assist in creating various types of pandoc command line arguments (e.g. for templates, table of contents, highlighting, and content includes).

Usage

```
pandoc_variable_arg(name, value)

pandoc_metadata_arg(name, value)

pandoc_metadata_file_arg(file)

pandoc_include_args(in_header = NULL, before_body = NULL, after_body = NULL)

pandoc_highlight_args(highlight, default = "tango")

pandoc_latex_engine_args(latex_engine)

pandoc_toc_args(toc, toc_depth = 3)

pandoc_citeproc_args()

pandoc_lua_filter_args(lua_files)
```

Arguments

name	Name of template variable to set.
value	Value of template variable (defaults to true if missing).
file	string. Path to a file
in_header	One or more files with content to be included in the header of the document.
before_body	One or more files with content to be included before the document body.
after_body	One or more files with content to be included after the document body.
highlight	The name of a pandoc syntax highlighting theme.
default	The highlighting theme to use if "default" is specified.
latex_engine	LaTeX engine for producing PDF output. Options are "pdflatex", "lualatex", "xelatex", and "tectonic".
toc	TRUE to include a table of contents in the output.
toc_depth	Depth of headers to include in table of contents.
lua_files	Character vector of file paths to Lua filter files. Paths will be transformed by pandoc_path_arg.

Details

Non-absolute paths for resources referenced from the in_header, before_body, and after_body parameters are resolved relative to the directory of the input document.

Value

A character vector with pandoc command line arguments.

About Pandoc citeproc

For Pandoc version before 2.11, a pandoc filter '`pandoc-citeproc`' is used. Since Pandoc 2.11, the feature is built-in and activated using '`--citeproc`' flag. '`pandoc_citeproc_arg`' will return the correct switches depending on the Pandoc version in use.

Examples

```
## Not run:
library(rmarkdown)

pandoc_include_args(before_body = "header.htm")
pandoc_include_args(before_body = "header.tex")

pandoc_highlight_args("kate")

pandoc_latex_engine_args("pdflatex")

pandoc_toc_args(toc = TRUE, toc_depth = 2)

## End(Not run)
```

`pandoc_available` *Check pandoc availability and version*

Description

Determine whether pandoc is currently available on the system (optionally checking for a specific version or greater). Determine the specific version of pandoc available.

Usage

```
pandoc_available(version = NULL, error = FALSE)

pandoc_version()
```

Arguments

`version`	Required version of pandoc
`error`	Whether to signal an error if pandoc with the required version is not found

Details

The system environment variable '`PATH`' as well as the version of pandoc shipped with RStudio (its location is set via the environment variable '`RSTUDIO_PANDOC`' by RStudio products like the RStudio IDE, RStudio Server, Shiny Server, and RStudio Connect, etc) are scanned for pandoc and the highest version available is used. Please do not modify the environment variable '`RSTUDIO_PANDOC`' unless you know what it means.

Value

pandoc_available returns a logical indicating whether the required version of pandoc is available. pandoc_version returns a numeric_version with the version of pandoc found.

Examples

```
## Not run:
library(rmarkdown)

if (pandoc_available())
  cat("pandoc", as.character(pandoc_version()), "is available!\n")

if (pandoc_available("1.12.3"))
  cat("required version of pandoc is available!\n")

## End(Not run)
```

---

pandoc_citeproc_convert

Convert a bibliograpy file

Description

Convert a bibliography file (e.g. a BibTeX file) to an R list, JSON text, or YAML text

Usage

```
pandoc_citeproc_convert(file, type = c("list", "json", "yaml"))
```

Arguments

file	Bibliography file
type	Conversion type

Value

For ‘type = "list"‘, and R list. For ‘type = "json"‘ or ‘type = "yaml"‘, a character vector with the specified format.

pandoc_convert *Convert a document with pandoc*

Description

Convert documents to and from various formats using the pandoc utility.

Usage

```
pandoc_convert(
  input,
  to = NULL,
  from = NULL,
  output = NULL,
  citeproc = FALSE,
  options = NULL,
  verbose = FALSE,
  wd = NULL
)
```

Arguments

input	Character vector containing paths to input files (files must be UTF-8 encoded)
to	Format to convert to (if not specified, you must specify output)
from	Format to convert from (if not specified then the format is determined based on the file extension of input).
output	Output file (if not specified then determined based on format being converted to).
citeproc	TRUE to run the pandoc-citeproc filter (for processing citations) as part of the conversion.
options	Character vector of command line options to pass to pandoc.
verbose	TRUE to show the pandoc command line which was executed
wd	Working directory in which code will be executed. If not supplied, defaults to the common base directory of input.

Details

Supported input and output formats are described in the pandoc user guide.

The system path as well as the version of pandoc shipped with RStudio (if running under RStudio) are scanned for pandoc and the highest version available is used.

Examples

```
## Not run:
library(rmarkdown)

# convert markdown to various formats
pandoc_convert("input.md", to = "html")
pandoc_convert("input.md", to = "latex")

# process citations
pandoc_convert("input.md", to = "html", citeproc = TRUE)

# add some pandoc options
pandoc_convert("input.md", to = "latex", options = c("--listings"))

## End(Not run)
```

---

pandoc_exec *Get the path of the pandoc executable*

Description

Returns the path of the pandoc executable used by functions in the the **rmarkdown** package. This is the most recent version of pandoc found in either the system path or shipped with RStudio.

Usage

```
pandoc_exec()
```

Details

See the pandoc manual for pandoc commands.

---

pandoc_options *Pandoc options for an output format*

Description

Define the pandoc options for an R Markdown output format.

Usage

```
pandoc_options(
  to,
  from = rmarkdown_format(),
  args = NULL,
  keep_tex = FALSE,
  latex_engine = c("pdflatex", "lualatex", "xelatex", "tectonic"),
  ext = NULL,
  lua_filters = NULL,
  convert_fun = NULL
)
```

Arguments

to	Pandoc format to convert to
from	Pandoc format to convert from
args	Character vector of command line arguments to pass to pandoc
keep_tex	Keep the intermediate tex file used in the conversion to PDF (applies only to 'latex' and 'beamer' target formats)
latex_engine	LaTeX engine to producing PDF output (applies only to 'latex' and 'beamer' target formats)
ext	File extension (e.g. ".tex") for output file (if NULL chooses default based on to). This is typically used to force the final output of a latex or beamer conversion to be .tex rather than .pdf.
lua_filters	Character vector of file paths to Lua filters to use with this format. They will be added to pandoc command line call using --lua-filter argument. See vignette("lua-filters", package = "rmarkdown") to know more about Lua filters.
convert_fun	A function to convert the input file to the desired output format in render(). If not provided, pandoc_convert() will be used. If a custom function is provided, its arguments and returned value should match the pandoc_convert() function. Note that this function does not have to use Pandoc but can also use other tools such as **commonmark**.

Details

The from argument should be used very cautiously as it's important for users to be able to rely on a stable definition of supported markdown extensions.

Value

An list that can be passed as the pandoc argument of the output_format function.

See Also

output_format, rmarkdown_format

pandoc_path_arg *Transform path for passing to pandoc*

Description

Transform a path for passing to pandoc on the command line. Calls `path.expand` on all platforms. On Windows, transform it to a short path name if it contains spaces, and then convert forward slashes to back slashes (as required by pandoc for some path references).

Usage

```
pandoc_path_arg(path, backslash = TRUE)
```

Arguments

`path`	Path to transform
`backslash`	Whether to replace forward slashes in `path` with backslashes on Windows.

Value

Transformed path that can be passed to pandoc on the command line.

pandoc_self_contained_html *Create a self-contained HTML document using pandoc.*

Description

Create a self-contained HTML document by base64 encoding images, scripts, and stylesheets referred by the input document.

Usage

```
pandoc_self_contained_html(input, output)
```

Arguments

`input`	Input html file to create self-contained version of.
`output`	Path to save output.

Value

(Invisibly) The path of the generated file.

pandoc_template *Render a pandoc template.*

Description

Use the pandoc templating engine to render a text file. Substitutions are done using the `metadata` list passed to the function.

Usage

```
pandoc_template(metadata, template, output, verbose = FALSE)
```

Arguments

`metadata`	A named list containing metadata to pass to template.
`template`	Path to a pandoc template.
`output`	Path to save output.
`verbose`	`TRUE` to show the pandoc command line which was executed.

Value

(Invisibly) The path of the generated file.

parse_html_notebook *Parse an HTML Notebook*

Description

Parse an HTML notebook, retrieving annotation information related to generated outputs in the document, as well as the original R Markdown source document.

Usage

```
parse_html_notebook(path)
```

Arguments

`path`	The path to an R Notebook file (with extension `.nb.html`).

Details

See the online documentation for additional details on using the `html_notebook` format.

pdf_document *Convert to a PDF/LaTeX document*

Description

Formats for converting from R Markdown to a PDF or LaTeX document.

Usage

```
pdf_document(
  toc = FALSE,
  toc_depth = 2,
  number_sections = FALSE,
  fig_width = 6.5,
  fig_height = 4.5,
  fig_crop = "auto",
  fig_caption = TRUE,
  dev = "pdf",
  df_print = "default",
  highlight = "default",
  template = "default",
  keep_tex = FALSE,
  keep_md = FALSE,
  latex_engine = "pdflatex",
  citation_package = c("default", "natbib", "biblatex"),
  includes = NULL,
  md_extensions = NULL,
  output_extensions = NULL,
  pandoc_args = NULL,
  extra_dependencies = NULL
)

latex_document(...)

latex_fragment(...)
```

Arguments

toc	TRUE to include a table of contents in the output
toc_depth	Depth of headers to include in table of contents
number_sections	TRUE to number section headings
fig_width	Default width (in inches) for figures
fig_height	Default height (in inches) for figures

`fig_crop` Whether to crop PDF figures with the command `pdfcrop`. This requires the tools `pdfcrop` and `ghostscript` to be installed. By default, `fig_crop = TRUE` if these two tools are available.

`fig_caption` `TRUE` to render figures with captions

`dev` Graphics device to use for figure output (defaults to pdf)

`df_print` Method to be used for printing data frames. Valid values include "default", "kable", "tibble", and "paged". The "default" method uses a corresponding S3 method of `print`, typically `print.data.frame`. The "kable" method uses the `knitr::kable` function. The "tibble" method uses the **tibble** package to print a summary of the data frame. The "paged" method creates a paginated HTML table (note that this method is only valid for formats that produce HTML). In addition to the named methods you can also pass an arbitrary function to be used for printing data frames. You can disable the `df_print` behavior entirely by setting the option `rmarkdown.df_print` to `FALSE`. See Data frame printing section in bookdown book for examples.

`highlight` Syntax highlighting style passed to Pandoc.

Supported built-in styles include "default", "tango", "pygments", "kate", "monochrome", "espresso", "zenburn", "haddock", and "breezedark".

Two custom styles are also included, "arrow", an accessible color scheme, and "rstudio", which mimics the default IDE theme. Alternatively, supply a path to a '`.theme`' file to use a custom Pandoc style. Note that custom theme requires Pandoc 2.0+.

Pass `NULL` to prevent syntax highlighting.

`template` Pandoc template to use for rendering. Pass "default" to use the rmarkdown package default template; pass `NULL` to use pandoc's built-in template; pass a path to use a custom template that you've created. See the documentation on pandoc online documentation for details on creating custom templates.

`keep_tex` Keep the intermediate tex file used in the conversion to PDF. Note that this argument does not control whether to keep the auxiliary files (e.g., '`.aux`') generated by LaTeX when compiling '`.tex`' to '`.pdf`'. To keep these files, you may set `options(tinytex.clean = FALSE)`.

`keep_md` Keep the markdown file generated by knitting.

`latex_engine` LaTeX engine for producing PDF output. Options are "pdflatex", "lualatex", "xelatex" and "tectonic".

`citation_package`

The LaTeX package to process citations, `natbib` or `biblatex`. Use `default` if neither package is to be used, which means citations will be processed via the command `pandoc-citeproc`.

`includes` Named list of additional content to include within the document (typically created using the `includes` function).

`md_extensions` Markdown extensions to be added or removed from the default definition of R Markdown. See the `rmarkdown_format` for additional details.

`output_extensions`

Pandoc extensions to be added or removed from the output format, e.g., `"-smart"` means the output format will be `latex-smart`.

`pandoc_args` Additional command line options to pass to pandoc

`extra_dependencies`
A LaTeX dependency `latex_dependency()`, a list of LaTeX dependencies, a character vector of LaTeX package names (e.g. `c("framed", "hyperref")`), or a named list of LaTeX package options with the names being package names (e.g. `list(hyperef = c("unicode=true", "breaklinks=true"), lmodern = NULL)`). It can be used to add custom LaTeX packages to the .tex header.

`...` Arguments passed to `pdf_document()`.

Details

See the online documentation for additional details on using the `pdf_document` format.

Creating PDF output from R Markdown requires that LaTeX be installed.

R Markdown documents can have optional metadata that is used to generate a document header that includes the title, author, and date. For more details see the documentation on R Markdown metadata.

R Markdown documents also support citations. You can find more information on the markdown syntax for citations in the Bibliographies and Citations article in the online documentation.

Many aspects of the LaTeX template used to create PDF documents can be customized using metadata. For example:

```
---
title: "Crop Analysis Q3 2013"
fontsize: 11pt
geometry: margin=1in
---
```

Available metadata variables include:

`lang` Document language code (e.g. "es", "fr", "pt-BR")

`fontsize` Font size (e.g. 10pt, 11pt, 12pt)

`documentclass` LaTeX document class (e.g. article)

`classoption` Option for `documentclass` (e.g. oneside); may be repeated

`geometry` Options for geometry class (e.g. margin=1in); may be repeated

`mainfont, sansfont, monofont, mathfont` Document fonts (works only with xelatex and lualatex, see the `latex_engine` option)

`linkcolor, urlcolor, citecolor` Color for internal, external, and citation links (red, green, magenta, cyan, blue, black)

`linestretch` Options for line spacing (e.g. 1, 1.5, 3)

Value

R Markdown output format to pass to `render`

Examples

```
## Not run:
library(rmarkdown)

# simple invocation
render("input.Rmd", pdf_document())

# specify an option for latex engine
render("input.Rmd", pdf_document(latex_engine = "lualatex"))

# add a table of contents and pass an option to pandoc
render("input.Rmd", pdf_document(toc = TRUE, "--listings"))

## End(Not run)
```

`pkg_file_lua` *Get the full paths of Lua filters in an R package*

Description

Lua filters stored in a source package in the 'inst/rmarkdown/lua' directory will be installed to the 'rmarkdown/lua' directory in the package path. This function finds the full paths of the Lua filters in the installed packages.

Usage

```
pkg_file_lua(filters = NULL, package = "rmarkdown")
```

Arguments

filters	A character vector of filenames for Lua filters to be retrieved in 'rmarkdown/lua' folder of the package. By default (NULL), if none is provided, it returns all filters in that folder.
package	The name of the package in which to look for the filters.

Value

A character vector of absolute file paths for the Lua filter from the package. The returned paths have been processed by pandoc_path_arg(), so they are ready to be used by Pandoc.

Examples

```
# list all Lua filters stored in the rmarkdown package
pkg_file_lua()
# get a specific filter
pkg_file_lua(c("pagebreak.lua", "latex_div.lua"))
```

powerpoint_presentation *Convert to a PowerPoint presentation*

Description

Format for converting from R Markdown to a PowerPoint presentation. Pandoc v2.0.5 or above is required.

Usage

```
powerpoint_presentation(
  toc = FALSE,
  toc_depth = 2,
  number_sections = FALSE,
  incremental = FALSE,
  fig_width = 5,
  fig_height = 4,
  fig_caption = TRUE,
  df_print = "default",
  keep_md = FALSE,
  md_extensions = NULL,
  slide_level = NULL,
  reference_doc = "default",
  pandoc_args = NULL
)
```

Arguments

`toc`	`TRUE` to include a table of contents in the output
`toc_depth`	Depth of headers to include in table of contents
`number_sections`	`TRUE` to number section headings
`incremental`	`TRUE` to render slide bullets incrementally. Note that if you want to reverse the default incremental behavior for an individual bullet you can precede it with >. For example: `> - Bullet Text`. See more in Pandoc's Manual
`fig_width`	Default width (in inches) for figures
`fig_height`	Default height (in inches) for figures
`fig_caption`	`TRUE` to render figures with captions
`df_print`	Method to be used for printing data frames. Valid values include "default", "kable", "tibble", and "paged". The "default" method uses a corresponding S3 method of `print`, typically `print.data.frame`. The "kable" method uses the `knitr::kable` function. The "tibble" method uses the **tibble** package to print a summary of the data frame. The "paged" method creates a paginated HTML table (note that this method is only valid for formats that produce HTML). In

	addition to the named methods you can also pass an arbitrary function to be used for printing data frames. You can disable the `df_print` behavior entirely by setting the option `rmarkdown.df_print` to `FALSE`. See Data frame printing section in bookdown book for examples.
`keep_md`	Keep the markdown file generated by knitting.
`md_extensions`	Markdown extensions to be added or removed from the default definition of R Markdown. See the `rmarkdown_format` for additional details.
`slide_level`	The heading level which defines individual slides. By default this is the highest header level in the hierarchy that is followed immediately by content, and not another header, somewhere in the document. This default can be overridden by specifying an explicit `slide_level`.
`reference_doc`	Path to a PowerPoint template.
`pandoc_args`	Additional command line options to pass to pandoc

Value

R Markdown output format to pass to `render()`

`publish_site` *Publish an R Markdown Website*

Description

Publish a website to RStudio Connect

Usage

```
publish_site(
  site_dir = ".",
  site_name = NULL,
  method = c("rsconnect"),
  server = NULL,
  account = NULL,
  render = TRUE,
  launch_browser = interactive()
)
```

Arguments

`site_dir`	Directory containing website. Defaults to current working directory.
`site_name`	Name for the site (names must be unique within an account). Defaults to the 'name' provided by the site generator (or to the name of the site_dir if there is no 'name' specified).
`method`	Publishing method (currently only "rsconnect" is available)

account, server	Uniquely identify a remote server with either your user account, the server name, or both. If neither are supplied, and there are multiple options, you'll be prompted to pick one. Use accounts() to see the full list of available options.
render	'TRUE' to render the site locally before publishing.
launch_browser	If 'TRUE', the system's default web browser will be launched automatically after the site is deployed. Defaults to 'TRUE' in interactive sessions only.

Examples

```
## Not run:
library(rmarkdown)
publish_site()

## End(Not run)
```

---

relative_to *Relative path utility function*

Description

Given a directory and a file, return a relative path from the directory to the file, or the unmodified file path if the file does not appear to be in the directory.

Usage

```
relative_to(dir, file)
```

Arguments

dir	Directory
file	File

Value

Relative path from the directory to the file (or the unmodified file path if the file does not appear to be in the directory).

render *Render R Markdown*

Description

Render the input file to the specified output format using pandoc. If the input requires knitting then `knit` is called prior to pandoc.

Usage

```
render(
  input,
  output_format = NULL,
  output_file = NULL,
  output_dir = NULL,
  output_options = NULL,
  output_yaml = NULL,
  intermediates_dir = NULL,
  knit_root_dir = NULL,
  runtime = c("auto", "static", "shiny", "shinyrmd", "shiny_prerendered"),
  clean = TRUE,
  params = NULL,
  knit_meta = NULL,
  envir = parent.frame(),
  run_pandoc = TRUE,
  quiet = FALSE,
  encoding = "UTF-8"
)
```

Arguments

`input`	The input file to be rendered. This can be an R script (.R), an R Markdown document (.Rmd), or a plain markdown document.
`output_format`	The R Markdown output format to convert to. The option `"all"` will render all formats defined within the file. The option can be the name of a format (e.g. `"html_document"`) and that will render the document to that single format. One can also use a vector of format names to render to multiple formats. Alternatively, you can pass an output format object (e.g. `html_document()`). If using `NULL` then the output format is the first one defined in the YAML frontmatter in the input file (this defaults to HTML if no format is specified there). If you pass an output format object to `output_format`, the options specified in the YAML header or `_output.yml` will be ignored and you must explicitly set all the options you want when you construct the object. If you pass a string, the output format will use the output parameters in the YAML header or `_output.yml`.
`output_file`	The name of the output file. If using `NULL` then the output filename will be based on filename for the input file. If a filename is provided, a path to the

output file can also be provided. Note that the `output_dir` option allows for specifying the output file path as well, however, if also specifying the path, the directory must exist. If `output_file` is specified but does not have a file extension, an extension will be automatically added according to the output format. To avoid the automatic file extension, put the `output_file` value in `I()`, e.g., `I('my-output')`.

`output_dir` The output directory for the rendered `output_file`. This allows for a choice of an alternate directory to which the output file should be written (the default output directory of that of the input file). If a path is provided with a filename in `output_file` the directory specified here will take precedence. Please note that any directory path provided will create any necessary directories if they do not exist.

`output_options` List of output options that can override the options specified in metadata (e.g. could be used to force `self_contained` or `mathjax = "local"`). Note that this is only valid when the output format is read from metadata (i.e. not a custom format object passed to `output_format`).

`output_yaml` Paths to YAML files specifying output formats and their configurations. The first existing one is used. If none are found, then the function searches YAML files specified to the `output_yaml` top-level parameter in the YAML front matter, _output.yml or _output.yaml, and then uses the first existing one.

`intermediates_dir`
Intermediate files directory. If a path is specified then intermediate files will be written to that path. If `NULL`, intermediate files are written to the same directory as the input file.

`knit_root_dir` The working directory in which to knit the document; uses knitr's `root.dir` knit option. If `NULL` then the behavior will follow the knitr default, which is to use the parent directory of the document.

`runtime` The runtime target for rendering. The `static` option produces output intended for static files; `shiny` produces output suitable for use in a Shiny document (see `run`). The default, `auto`, allows the `runtime` target specified in the YAML metadata to take precedence, and renders for a `static` runtime target otherwise.

`clean` Using `TRUE` will clean intermediate files that are created during rendering.

`params` A list of named parameters that override custom params specified within the YAML front-matter (e.g. specifying a dataset to read or a date range to confine output to). Pass `"ask"` to start an application that helps guide parameter configuration.

`knit_meta` (This option is reserved for expert use.) Metadata generated by **knitr**.

`envir` The environment in which the code chunks are to be evaluated during knitting (can use `new.env()` to guarantee an empty new environment).

`run_pandoc` An option for whether to run pandoc to convert Markdown output.

`quiet` An option to suppress printing during rendering from knitr, pandoc command line and others. To only suppress printing of the last "Output created: " message, you can set `rmarkdown.render.message` to `FALSE`

`encoding` Ignored. The encoding is always assumed to be UTF-8.

Details

Note that the **knitr** `error` option is set to `FALSE` during rendering (which is different from the **knitr** default value of `TRUE`).

For additional details on rendering R scripts see Compiling R scripts to a notebook.

If no `output_format` parameter is specified then the output format is read from the YAML front-matter of the input file. For example, the following YAML would yield a PDF document:

```
output: pdf_document
```

Additional format options can also be specified in metadata. For example:

```
output:
  pdf_document:
    toc: true
    highlight: zenburn
```

Multiple formats can be specified in metadata. If no `output_format` is passed to `render` then the first one defined will be used:

```
output:
  pdf_document:
    toc: true
    highlight: zenburn
  html_document:
    toc: true
    theme: united
```

Formats specified in metadata can be any one of the built in formats (e.g. `html_document`, `pdf_document`) or a format defined in another package (e.g. `pkg::custom_format`).

If there is no format defined in the YAML then `html_document` will be used.

Value

When `run_pandoc = TRUE`, the compiled document is written into the output file, and the path of the output file is returned. When `run_pandoc = FALSE`, the path of the Markdown output file, with attributes `knit_meta` (the **knitr** meta data collected from code chunks) and `intermediates` (the intermediate files/directories generated by `render()`).

R Markdown

R Markdown supports all of the base pandoc markdown features as well as some optional features for compatibility with GitHub Flavored Markdown (which previous versions of R Markdown were based on). See `rmarkdown_format` for details.

See Also

knit, output_format, `https://pandoc.org`

Examples

```
## Not run:
library(rmarkdown)

# Render the default (first) format defined in the file
render("input.Rmd")

# Render all formats defined in the file
render("input.Rmd", "all")

# Render a single format, using parameters for \code{html_document} from
# the YAML header parameters.
render("input.Rmd", "html_document")

# Render a single format, ignoring parameters for \code{html_document} in
# the YAML header. Any parameters not passed as arguments to
# \code{html_document()} will be assigned to their default values, regardless
# of anything in the YAML header
render("input.Rmd", html_document(toc = TRUE, toc_depth = 2))

# Render multiple formats
render("input.Rmd", c("html_document", "pdf_document"))

## End(Not run)
```

`render_delayed` *Delay Rendering for an Expression*

Description

In a Shiny document, evaluate the given expression after the document has finished rendering, instead of during render.

Usage

```
render_delayed(expr)
```

Arguments

`expr`	The expression to evaluate.

Details

This function is useful inside Shiny documents. It delays the evaluation of its argument until the document has finished its initial render, so that the document can be viewed before the calculation is finished.

Any expression that returns HTML can be wrapped in `render_delayed`.

Value

An object representing the expression.

Note

`expr` is evaluated in a **copy** of the environment in which the `render_delayed` call appears. Consequently, no side effects created by `expr` are visible in succeeding expressions, nor are changes to the environment after the call to `render_delayed` visible to `expr`.

`expr` must be an expression that produces HTML.

Examples

```
## Not run:
# Add the following code to an R Markdown document

div(Sys.time())

render_delayed({
 Sys.sleep(3)      # simulate an expensive computation
 div(Sys.time())
})

div(Sys.time())

## End(Not run)
```

---

render_site *Render multiple documents as a website*

Description

Render all of the R Markdown documents within a directory as a website.

Usage

```
render_site(
  input = ".",
  output_format = "all",
  envir = parent.frame(),
  quiet = FALSE,
  encoding = "UTF-8"
)

clean_site(input = ".", preview = TRUE, quiet = FALSE, encoding = "UTF-8")

site_generator(input = ".", output_format = NULL)

site_config(input = ".", encoding = "UTF-8")
```

```
default_site_generator(input, output_format_filter = NULL, ...)
```

Arguments

input	Website directory (or the name of a file within the directory).
output_format	R Markdown format to convert to (defaults to "all").
envir	The environment in which the code chunks are to be evaluated during knitting (can use new.env to guarantee an empty new environment).
quiet	TRUE to suppress messages and other output.
encoding	Ignored. The encoding is always assumed to be UTF-8.
preview	Whether to list the files to be removed rather than actually removing them. Defaulting to TRUE to prevent removing without notice.
output_format_filter	An optional function which is passed the input file and the output format, and which returns a (potentially modified) output format.
...	Currently unused.

Details

The render_site function enables you to render a collection of markdown documents within a directory as a website. There are two requirements for a directory to be rendered as a website:

1. It must contain either an "index.Rmd" or "index.md" file.
2. It must contain a site configuration file ("_site.yml").

The most minimal valid website is an empty "index.Rmd" and an empty "_site.yml". With this configuration a single empty webpage would be generated via a call to render_site. If you add additional markdown documents to the directory they will also be rendered. By default a site is rendered in the following fashion:

1. R Markdown (.Rmd) and plain markdown (.md) files in the root directory are rendered. Note however that markdown files beginning with "_" are not rendered (this is a convention to designate files that are included by top level documents).
2. All output and supporting files are copied to a "_site" subdirectory of the website directory (this is configurable, see discussion below).
3. The following files are **not** copied to the "_site" sub-directory:
   - Files beginning with "." (hidden files).
   - Files beginning with "_"
   - Files known to contain R source code (e.g. ".R", ".s", ".Rmd"), R data (e.g. ".RData", ".rds"), configuration data (e.g. ".Rproj", "rsconnect") or package project management data (e.g. "packrat", "renv").

   Note that you can override which files are included or excluded via settings in "_site.yml" (described below).

4. Normally R Markdown renders documents as self-contained HTML. However, `render_site` ensures that dependencies (e.g. CSS, JavaScript, images, etc.) remain in external files. CSS/JavaScript libraries are copied to a "site_libs" sub-directory and plots/images are copied to "_files" sub-directories.

You can remove the files generated by `render_site` using the `clean_site` function.

Value

`render_site` returns the name of the site output file (relative to the input directory). `clean_site` returns the names of the generated files removed during cleaning. `site_config` returns the contents of _site.yml as an R list. `default_site_generator` returns the default site generator for R Markdown websites.

Configuration

A "_site.yml" file can be used to configure the behavior of site generation. Here is an example configuration file:

```
name: my-website
output_dir: _site
include: ["demo.R"]
exclude: ["docs.txt", "*.csv"]
navbar:
  title: "My Website"
  left:
    - text: "Home"
      href: index.html
    - text: "About"
      href: about.html
output:
  html_document:
    toc: true
    highlight: textmate
```

The `name` field provides a suggested URL path for your website when it is published (by default this is just the name of the directory containing the site). The `output_dir` indicates which directory to copy site content into ("_site" is the default if none is specified). Note that this can be "." to keep all content within the root website directory alongside the source code.

The `include` and `exclude` fields enable you to override the default behavior vis-a-vis what files are copied into the "_site" directory (wildcards can be used as in the above example).

The `navbar` field can be used to define a navigation bar for websites based on the `html_document` format.

Finally, the `output` field enables you to specify output options that are common to all documents within the website (you can also still provide local options within each document that override any common options).

`new_session: true` causes each file to be rendered in a new R session. This prevents the masking problem that arises when different files use functions from different packages (namespaces)

that share a common name, such as `here::here` and `lubridate::here` or `dplyr::filter` and `MASS::filter`. The default behaviour of `render_site` is to use a common R session.

`autospin: true` causes `.R` files to be spinned and rendered (as well as `.Rmd` files). If `autospin` is set to false (the default), `.R` files will not be spinned nor rendered. `autospin` can also enumerate a list of .R files to be spinned and rendered.

Custom Site Generation

The behavior of the default site generation function (`rmarkdown::default_site`) is described above. It is also possible to define a custom site generator that has alternate behavior. A site generator is an R function that is bound to by including it in the "site:" field of the "index.Rmd" or "index.md" file. For example:

```
title: "My Book"
output: bookdown::gitbook
site: bookdown::bookdown_site
```

A site generation function should return a list with the following elements:

`name`**:** The name for the website (e.g. the parent directory name).

`output_dir`**:** The directory where the website output is written to. This path should be relative to the site directory (e.g. "." or "_site")

`render`**:** An R function that can be called to generate the site. The function should accept the `input_file`, `output_format`, `envir`, and `quiet` arguments.

`clean`**:** An R function that returns relative paths to the files generated by `render_site` (these files are the ones which will be removed by the `clean_site` function.

`subdirs` ***(optional)***: A logical flag that indicates if the generator supports nested source files in sub-directories of the project (`TRUE`) or only at the project root (`FALSE`). (e.g. `blogdown:::blogdown_site()`)

Note that the `input_file` argument will be `NULL` when the entire site is being generated. It will be set to a specific file name if a front-end tool is attempting to preview it (e.g. RStudio IDE via the Knit button).

When `quiet = FALSE` the `render` function should also print a line of output using the `message` function indicating which output file should be previewed, for example:

```
if (!quiet)
  message("\nOutput created: ", output)
```

Emitting this line enables front-ends like RStudio to determine which file they should open to preview the website.

See the source code of the `rmarkdown::default_site` function for a example of a site generation function.

render_supporting_files *Render supporting files for an input document*

Description

Render (copy) required supporting files for an input document to the `_files` directory that is associated with the document.

Usage

```
render_supporting_files(from, files_dir, rename_to = NULL)
```

Arguments

`from`	The directory from which the files should be copied.
`files_dir`	The directory that will receive the copied files.
`rename_to`	An option to rename the source directory after the copy operation is complete.

Value

The relative path to the supporting files. This path is suitable for inclusion in HTML`href` and `src` attributes.

resolve_output_format *Resolve the output format for an R Markdown document*

Description

Read the YAML metadata (and any common output YAML file) for the document and return an output format object that can be passed to the `render` function.

Usage

```
resolve_output_format(
  input,
  output_format = NULL,
  output_options = NULL,
  output_yaml = NULL
)
```

Arguments

input	Input file (Rmd or plain markdown)
output_format	Name of output format (or NULL to use the default format for the input file).
output_options	List of output options that should override the options specified in metadata.
output_yaml	Paths to YAML files specifying output formats and their configurations. The first existing one is used. If none are found, then the function searches YAML files specified to the output_yaml top-level parameter in the YAML front matter, _output.yml or _output.yaml, and then uses the first existing one.

Details

This function is useful for front-end tools that need to modify the default behavior of an output format.

Value

An R Markdown output format definition that can be passed to render.

---

rmarkdown_format *R Markdown input format definition*

Description

Compose a pandoc markdown input definition for R Markdown that can be passed as the from argument of pandoc_options.

Usage

```
rmarkdown_format(extensions = NULL)

from_rmarkdown(implicit_figures = TRUE, extensions = NULL)
```

Arguments

extensions	Markdown extensions to be added or removed from the default definition of R Markdown.
implicit_figures	Automatically make figures from images (defaults to TRUE).

Details

By default R Markdown is defined as all pandoc markdown extensions with the following tweaks for backward compatibility with the markdown package (+ features are added, - features are removed):

```
+autolink_bare_uris
+tex_math_single_backslash
```

For more on pandoc markdown see the pandoc online documentation.

Value

Pandoc markdown format specification

See Also

output_format, pandoc_options

Examples

```
## Not run:
rmarkdown_format("-implicit_figures")

## End(Not run)
```

`rmd_metadata` *R Markdown Metadata*

Description

Rmd files include a metadata section (typically located at the top of the file) that can specify (among other things) the title, author, and date of the document. Metadata adheres to the YAML format and is delimited by lines containing three dashes (`---`). Here is an example metadata section:

```
---
title: "Crop Analysis Q3 2013"
author: Martha Smith
date: October 23rd, 2013
---
```

Note that the `title` field is quoted. This is because titles often contained embedded colons (`:`) and colons followed by a space need to be quoted in YAML.

Details

When title, author, and date metadata is provided it's used to automatically create a title section within output documents. If you don't want this section included in your document then you should remove the corresponding metadata fields.

When generating PDF and Beamer output there are also a number of other metadata fields that can be included to customize the appearance and theme of PDF output. For more details see the documentation for `pdf_document` and `beamer_presentation`.

`rtf_document` *Convert to an RTF document*

Description

Format for converting from R Markdown to an RTF document.

Usage

```
rtf_document(
  toc = FALSE,
  toc_depth = 3,
  number_sections = FALSE,
  fig_width = 5,
  fig_height = 4,
  keep_md = FALSE,
  md_extensions = NULL,
  pandoc_args = NULL
)
```

Arguments

`toc`	`TRUE` to include a table of contents in the output
`toc_depth`	Depth of headers to include in table of contents
`number_sections`	`TRUE` to number section headings
`fig_width`	Default width (in inches) for figures
`fig_height`	Default height (in inches) for figures
`keep_md`	Keep the markdown file generated by knitting.
`md_extensions`	Markdown extensions to be added or removed from the default definition of R Markdown. See the `rmarkdown_format` for additional details.
`pandoc_args`	Additional command line options to pass to pandoc

Details

See the online documentation for additional details on using the `rtf_document` format.

R Markdown documents can have optional metadata that is used to generate a document header that includes the title, author, and date. For more details see the documentation on R Markdown metadata.

R Markdown documents also support citations. You can find more information on the markdown syntax for citations in the Bibliographies and Citations article in the online documentation.

Value

R Markdown output format to pass to `render`

Examples

```
## Not run:

library(rmarkdown)

# simple invocation
render("input.Rmd", rtf_document())

# specify table of contents option
render("input.Rmd", rtf_document(toc = TRUE))

## End(Not run)
```

---

run *Run a Shiny document*

Description

Start a Shiny server for the given document, and render it for display.

Usage

```
run(
  file = "index.Rmd",
  dir = dirname(file),
  default_file = NULL,
  auto_reload = TRUE,
  shiny_args = NULL,
  render_args = NULL
)
```

Arguments

file	Path to the R Markdown document to launch in a web browser. Defaults to index.Rmd in the current working directory, but may be NULL to skip launching a browser.
dir	The directory from which to to read input documents. Defaults to the parent directory of file.
default_file	The file to serve at the Shiny server's root URL. If NULL (the default), a sensible default is chosen (see Details)
auto_reload	If TRUE (the default), automatically reload the Shiny application when the file currently being viewed is changed on disk.
shiny_args	Additional arguments to runApp.
render_args	Additional arguments to render.

Details

The run function runs a Shiny document by starting a Shiny server associated with the document. The shiny_args parameter can be used to configure the server; see the runApp documentation for details.

Once the server is started, the document will be rendered using render. The server will initiate a render of the document whenever necessary, so it is not necessary to call run every time the document changes: if auto_reload is TRUE, saving the document will trigger a render. You can also manually trigger a render by reloading the document in a Web browser.

The server will render any R Markdown (.Rmd) document in dir; the file argument specifies only the initial document to be rendered and viewed. You can therefore link to other documents in the directory using standard Markdown syntax, e.g. [Analysis Page 2](page2.Rmd).

If default_file is not specified, nor is a file specified on the URL, then the default document to serve at / is chosen from (in order of preference):

- If dir contains only one Rmd, that Rmd.
- The file 'index.Rmd', if it exists in dir.
- The first Rmd that has runtime: shiny in its YAML metadata.
- The file 'index.html' (or 'index.htm'), if it exists in dir.

If you wish to share R code between your documents, place it in a file named global.R in dir; it will be sourced into the global environment.

Value

Invisible NULL.

Note

Unlike render, run does not render the document to a file on disk. In most cases a Web browser will be started automatically to view the document; see launch.browser in the runApp documentation for details.

When using an external web browser with the server, specify the name of the R Markdown file to view in the URL (e.g. http://127.0.0.1:1234/foo.Rmd). A URL without a filename will show the default_file as described above.

Examples

```
## Not run:
# Run the Shiny document "index.Rmd" in the current directory
rmarkdown::run()

# Run the Shiny document "shiny_doc.Rmd" on port 8241
rmarkdown::run("shiny_doc.Rmd", shiny_args = list(port = 8241))

## End(Not run)
```

shiny_prerendered_chunk

Add code to a shiny_prerendered context

Description

Programmatic equivalent to including a code chunk with a context in a runtime: shiny_prerendered document.

Usage

```
shiny_prerendered_chunk(context, code, singleton = FALSE)
```

Arguments

context	Context name (e.g. "server", "server-start")
code	Character vector with code
singleton	Collapse multiple identical versions of this chunk into a single chunk.

shiny_prerendered_clean

Clean prerendered content for the specified Rmd input file

Description

Remove the associated html file and supporting _files directory for a shiny_prerendered documet.

Usage

```
shiny_prerendered_clean(input)
```

Arguments

input	Rmd input file to clean content for

site_resources *Determine website resource files for a directory*

Description

Determine which files within a given directory should be copied in order to serve a website from the directory. Attempts to automatically exclude source, data, hidden, and other files not required to serve website content.

Usage

```
site_resources(site_dir, include = NULL, exclude = NULL, recursive = FALSE)
```

Arguments

site_dir	Site directory to analyze
include	Additional files to include (glob wildcards supported)
exclude	Files to exclude (glob wildcards supported)
recursive	TRUE to return a full recursive file listing; FALSE to just provide top-level files and directories.

Value

Character vector of files and directories to copy

slidy_presentation *Convert to a slidy presentation*

Description

Format for converting from R Markdown to a slidy presentation.

Usage

```
slidy_presentation(
  number_sections = FALSE,
  incremental = FALSE,
  slide_level = NULL,
  duration = NULL,
  footer = NULL,
  font_adjustment = 0,
  fig_width = 8,
  fig_height = 6,
  fig_retina = 2,
  fig_caption = TRUE,
```

```
  dev = "png",
  df_print = "default",
  self_contained = TRUE,
  highlight = "default",
  math_method = "default",
  mathjax = "default",
  template = "default",
  css = NULL,
  includes = NULL,
  keep_md = FALSE,
  lib_dir = NULL,
  md_extensions = NULL,
  pandoc_args = NULL,
  extra_dependencies = NULL,
  ...
)
```

Arguments

number_sections	TRUE to number section headings
incremental	TRUE to render slide bullets incrementally. Note that if you want to reverse the default incremental behavior for an individual bullet you can precede it with >. For example: `> - Bullet Text`. See more in Pandoc's Manual
slide_level	The heading level which defines individual slides. By default this is the highest header level in the hierarchy that is followed immediately by content, and not another header, somewhere in the document. This default can be overridden by specifying an explicit `slide_level`.
duration	Duration (in minutes) of the slide deck. This value is used to add a countdown timer to the slide footer.
footer	Footer text (e.g. organization name and/or copyright)
font_adjustment	Increase or decrease the default font size (e.g. -1 or +1). You can also manually adjust the font size during the presentation using the 'S' (smaller) and 'B' (bigger) keys.
fig_width	Default width (in inches) for figures
fig_height	Default height (in inches) for figures
fig_retina	Scaling to perform for retina displays (defaults to 2, which currently works for all widely used retina displays). Set to NULL to prevent retina scaling. Note that this will always be NULL when keep_md is specified (this is because `fig_retina` relies on outputting HTML directly into the markdown document).
fig_caption	TRUE to render figures with captions
dev	Graphics device to use for figure output (defaults to pdf)
df_print	Method to be used for printing data frames. Valid values include "default", "kable", "tibble", and "paged". The "default" method uses a corresponding S3 method of `print`, typically `print.data.frame`. The "kable" method uses the

knitr::kable function. The "tibble" method uses the **tibble** package to print a summary of the data frame. The "paged" method creates a paginated HTML table (note that this method is only valid for formats that produce HTML). In addition to the named methods you can also pass an arbitrary function to be used for printing data frames. You can disable the df_print behavior entirely by setting the option rmarkdown.df_print to FALSE. See Data frame printing section in bookdown book for examples.

self_contained Produce a standalone HTML file with no external dependencies, using data: URIs to incorporate the contents of linked scripts, stylesheets, images, and videos. Note that even for self contained documents MathJax is still loaded externally (this is necessary because of its size).

highlight Syntax highlighting style passed to Pandoc.

Supported built-in styles include "default", "tango", "pygments", "kate", "monochrome", "espresso", "zenburn", "haddock", and "breezedark".

Two custom styles are also included, "arrow", an accessible color scheme, and "rstudio", which mimics the default IDE theme. Alternatively, supply a path to a '.theme' file to use a custom Pandoc style. Note that custom theme requires Pandoc 2.0+.

Pass NULL to prevent syntax highlighting.

math_method Math rendering engine to use. This will define the math method to use with Pandoc.

- It can be a string for the engine, one of "mathjax", "mathml", "webtex", "katex", "gladtex", or "r-katex" or "default" for mathjax.
- It can be a list of
  - engine: one of "mathjax", "mathml", "webtex", "katex", or "gladtex".
  - url: A specific url to use with mathjax, katex or webtex. Note that for engine = "mathjax", url = "local" will use a local version of MathJax (which is copied into the output directory).

For example,

```
output:
  html_document:
    math_method:
      engine: katex
      url: https://cdn.jsdelivr.net/npm/katex@0.11.1/dist
```

See Pandoc's Manual about Math in HTML for the details about Pandoc supported methods.

Using math_method = "r-katex" will opt-in server side rendering using KaTeX thanks to katex R package. This is useful compared to math_method = "katex" to have no JS dependency, only a CSS dependency for styling equation.

mathjax Include mathjax. The "default" option uses an https URL from a MathJax CDN. The "local" option uses a local version of MathJax (which is copied into the output directory). You can pass an alternate URL or pass NULL to exclude MathJax entirely.

template	Pandoc template to use for rendering. Pass "default" to use the rmarkdown package default template; pass NULL to use pandoc's built-in template; pass a path to use a custom template that you've created. See the documentation on pandoc online documentation for details on creating custom templates.
css	One or more css files to include.
includes	Named list of additional content to include within the document (typically created using the includes function).
keep_md	Keep the markdown file generated by knitting.
lib_dir	Directory to copy dependent HTML libraries (e.g. jquery, bootstrap, etc.) into. By default this will be the name of the document with _files appended to it.
md_extensions	Markdown extensions to be added or removed from the default definition of R Markdown. See the rmarkdown_format for additional details.
pandoc_args	Additional command line options to pass to pandoc
extra_dependencies	A LaTeX dependency latex_dependency(), a list of LaTeX dependencies, a character vector of LaTeX package names (e.g. c("framed", "hyperref")), or a named list of LaTeX package options with the names being package names (e.g. list(hyperef = c("unicode=true", "breaklinks=true"), lmodern = NULL)). It can be used to add custom LaTeX packages to the .tex header.
...	Additional function arguments to pass to the base R Markdown HTML output formatter html_document_base

Details

See the online documentation for additional details on using the slidy_presentation format.

For more information on markdown syntax for presentations see the pandoc online documentation.

Value

R Markdown output format to pass to render

Examples

```
## Not run:
library(rmarkdown)

# simple invocation
render("pres.Rmd", slidy_presentation())

# specify an option for incremental rendering
render("pres.Rmd", slidy_presentation(incremental = TRUE))

## End(Not run)
```

word_document *Convert to an MS Word document*

Description

Format for converting from R Markdown to an MS Word document.

Usage

```
word_document(
  toc = FALSE,
  toc_depth = 3,
  number_sections = FALSE,
  fig_width = 5,
  fig_height = 4,
  fig_caption = TRUE,
  df_print = "default",
  highlight = "default",
  reference_docx = "default",
  keep_md = FALSE,
  md_extensions = NULL,
  pandoc_args = NULL
)
```

Arguments

toc	TRUE to include a table of contents in the output
toc_depth	Depth of headers to include in table of contents
number_sections	TRUE to number section headings
fig_width	Default width (in inches) for figures
fig_height	Default height (in inches) for figures
fig_caption	TRUE to render figures with captions
df_print	Method to be used for printing data frames. Valid values include "default", "kable", "tibble", and "paged". The "default" method uses a corresponding S3 method of print, typically print.data.frame. The "kable" method uses the knitr::kable function. The "tibble" method uses the **tibble** package to print a summary of the data frame. The "paged" method creates a paginated HTML table (note that this method is only valid for formats that produce HTML). In addition to the named methods you can also pass an arbitrary function to be used for printing data frames. You can disable the df_print behavior entirely by setting the option rmarkdown.df_print to FALSE. See Data frame printing section in bookdown book for examples.

highlight	Syntax highlighting style passed to Pandoc. Supported built-in styles include "default", "tango", "pygments", "kate", "monochrome", "espresso", "zenburn", "haddock", and "breezedark". Two custom styles are also included, "arrow", an accessible color scheme, and "rstudio", which mimics the default IDE theme. Alternatively, supply a path to a '.theme' file to use a custom Pandoc style. Note that custom theme requires Pandoc 2.0+. Pass NULL to prevent syntax highlighting.
reference_docx	Use the specified file as a style reference in producing a docx file. For best results, the reference docx should be a modified version of a docx file produced using pandoc. Pass "default" to use the rmarkdown default styles.
keep_md	Keep the markdown file generated by knitting.
md_extensions	Markdown extensions to be added or removed from the default definition of R Markdown. See the rmarkdown_format for additional details.
pandoc_args	Additional command line options to pass to pandoc

Details

See the online documentation for additional details on using the word_document format.

R Markdown documents can have optional metadata that is used to generate a document header that includes the title, author, and date. For more details see the documentation on R Markdown metadata.

R Markdown documents also support citations. You can find more information on the markdown syntax for citations in the Bibliographies and Citations article in the online documentation.

Value

R Markdown output format to pass to render

Examples

```
## Not run:
library(rmarkdown)

# simple invocation
render("input.Rmd", word_document())

# specify an option for syntax highlighting
render("input.Rmd", word_document(highlight = "zenburn"))

## End(Not run)
```

Index