

Package ‘stationery’

October 14, 2022

Type Package

Title Working Examples for Reproducible Research Documents

Version 1.0

Date 2021-10-08

Maintainer Paul Johnson <pauljohn@ku.edu>

Description Templates, guides, and scripts for writing documents in 'LaTeX' and 'R markdown' to produce guides, slides, and reports. Special care is taken to illustrate use of templates and customization opportunities. Challenges and opportunities of 'HTML' output from 'R markdown' receive special attention. Includes several vignettes to assist new users of literate programming.

License CC BY-SA 4.0

Imports kutils, rmarkdown, knitr

Suggests rockchalk, xtable, lavaan, utils, psych, pander

RoxygenNote 7.1.2

VignetteBuilder knitr, utils

NeedsCompilation no

Author Paul Johnson [aut, cre],
Brent Kaplan [ctb],
Meghan Sullivan [ctb],
Charles Redmon [ctb],
Zack Roman [ctb],
Chong Xing [ctb],
Longcan 'Emma' Huang [ctb],
Garrett Mills [ctb]

Repository CRAN

Date/Publication 2021-10-08 16:00:02 UTC

R topics documented:

blankPDF	2
blankPNG	3
crmda_html_document	4
getFiles	5
initWriteup	6
rmd2html	8
rmd2pdf	10
rnw2pdf	11

Index	14
--------------	-----------

blankPDF	<i>create white background empty PDF</i>
----------	--

Description

Sometimes you need a placeholder. If you do, this is it!

Usage

```
blankPDF(
  file,
  height = 1,
  width = 3.5,
  messg = "Your Logo Could Be Here",
  fontsize = 12,
  col = "gray50"
)
```

Arguments

file	A file name for output
height	inches
width	inches
messg	Default is a reminder statement, but set "" if you don't want it.
fontsize	Default is 12
col	Color for text in messg parameter. Default is gray50

Value

NULL is returned if file was created. Otherwise error is reported.

Author(s)

Paul Johnson

Examples

```
tdir <- tempdir()
dir.create(file.path(tdir, "blanks"), recursive = TRUE)
fn1 <- file.path(tdir, "blanks", "blank1.pdf")
blankPDF(file = fn1, messg = "Do you want a message?")
## Please inspect
if(interactive()) browseURL(fn1)
fn2 <- file.path(tdir, "blanks", "blank2.pdf")
blankPDF(file = fn2, height = 2, width = 3, messg = "")
if(interactive()) browseURL(fn2)
## delete test directory
unlink(file.path(tdir, "blanks"), recursive = TRUE)
```

blankPNG

create white background empty PNG

Description

Sometimes you need a placeholder. If you do, this is it!

Usage

```
blankPNG(
  file,
  height = 250,
  width = 250,
  messg = "Your Logo Could Be Here",
  pointsize = 12,
  col = "gray50"
)
```

Arguments

file	A file name for output
height	pixels
width	pixels
messg	Default is a reminder statement inside the resulting image, but set "" if you don't want it.
pointsize	pointsize 12 default
col	Color for text in messg parameter. Default is gray50

Value

NULL is returned if file was created. Otherwise error is reported.

Author(s)

Paul Johnson

Examples

```
tdir <- tempdir()
dir.create(file.path(tdir, "blanks"), recursive = TRUE)
fn1 <- file.path(tdir, "blanks", "blank1.png")
blankPNG(file = fn1, messg = "Do you want a message?")
## Please inspect
if(interactive()) browseURL(fn1)
fn2 <- file.path(tdir, "blanks", "blank2.png")
blankPNG(file = fn2, height = 2, width = 3, messg = "")
if(interactive()) browseURL(fn2)
## delete test directory
unlink(file.path(tdir, "blanks"), recursive = TRUE)
```

crmda_html_document *Custom output_format object to make MathJax work with custom HTML template*

Description

The `rmarkdown::html_document` fails to use MathJax if a custom HTML template is supplied. This is described here <https://github.com/rstudio/rmarkdown/issues/727>. The workaround is to provide this wrapper.

Usage

```
crmda_html_document(template = "theme/guide-template.html", ...)
```

Arguments

template	Name of template file.
...	Any arguments passed along to <code>html_document</code> in <code>rmarkdown</code>

Details

This function is needed only so as to "fool" pandoc in order to make an HTML document compatible with MathJax. See the example files created by `initWriteup("rmd2html-guide")` for a usage example.

Value

`html_document` object with custom template

Author(s)

Paul Johnson

Examples

```
tdir <- tempdir()
doctype <- "rmd2html-guide"
dirout <- initWriteup(doctype, dir = file.path(tdir, doctype))
list.files(dirout)
MESSG <- paste("Inspect the YAML header output section of 'skeleton.Rmd'.",
               "It makes use of 'crmda_html_document'.")
cat(MESSG)
```

getFiles

*Retrieve theme files***Description**

A wrapper for `file.copy` that retrieves files from the `dn = "theme"` directory of a package. It ties in the `file.backup` function from `kutils` so that if a file is overwritten, then a backup copy is created, using the last modification time of the file in the backup file name.

Usage

```
getFiles(
  fn,
  pkg = "stationery",
  overwrite = FALSE,
  backup = overwrite,
  pkgdir = "theme",
  outputdir = "theme"
)
```

Arguments

<code>fn</code>	A file name vector. In <code>stationery</code> , this is typically retrieving logo images or theme files.
<code>pkg</code>	Package name, default <code>"stationery"</code>
<code>overwrite</code>	Default <code>FALSE</code> , function returns <code>TRUE</code> , meaning "my work was done before." If <code>TRUE</code> , old file will be replaced.
<code>backup</code>	Default is <code>TRUE</code> , if a file is to be overwritten, then a backup will be created with <code>kutils::file.backup</code> .
<code>pkgdir</code>	package directory name where files are found. Default is <code>"theme"</code> because that is used in <code>stationery</code> and related CRMDA packages. It is the name of a directory in the package's <code>inst</code> folder, the same place where R <code>system.file</code> looks for files and directories.
<code>outputdir</code>	Default is same as <code>pkgdir</code> . A directory where file file will be copied within the current working directory.

Details

This was originally created because we wanted to protect user logo and address files from accidental erasure. Now it has a little more versatility because `overwrite=TRUE` and `backup=TRUE` add a little bit of valuable functionality. When a file is overwritten, it should never be lost if `backup=TRUE`.

Value

TRUE if succeeded, either a file existed before (and `overwrite=FALSE`) or a file was copied.

Author(s)

Paul Johnson<pauljohn@ku.edu>

Examples

```
## To demonstrate, we use a temporary directory. Usage of setwd is discouraged
## in examples, so this is a little bit more indirect than a real usage would be:
tdir <- tempdir()
list.files(file.path(tdir, "theme"))
getFiles("logo.pdf", pkg = "stationery", overwrite = TRUE, outputdir = file.path(tdir, "theme"))
list.files(file.path(tdir, "theme"))
getFiles("logo.pdf", pkg = "stationery", overwrite = TRUE, outputdir = file.path(tdir, "theme"),
        backup = TRUE)
list.files(file.path(tdir, "theme"))
unlink(file.path(tdir, "theme"), recursive = TRUE)
```

initWriteup

Create skeleton for a report or guide.

Description

The installed package includes a set of folders with document templates. `initWriteup` is simply an automated way to copy the template folder and skeleton document file into a new directory.

The current formats are:

1. `rmd2html-guide`
2. `rmd2pdf-report`
3. `rnw2pdf-guide-sweave`
4. `rnw2pdf-report-sweave`
5. `rmd2pdf-guide`
6. `rnw2pdf-guide-knit`
7. `rnw2pdf-report-knit`
8. `rnw2pdf-slides-sweave`

The names represent the "from" format (rmd or rnw), the "to" format (html or pdf), the document type (guide or report), and the chunk processing program (Sweave or knitr).

Each selection offers a self-contained working document and enough information to compile that document. It also includes a shell script that can compile the document.

Usage

```
initWriteup(type, dir = type)
```

Arguments

type	One of these character strings: c("rmd2html-guide", "rmd2pdf-report", "rnw2pdf-guide-knit", "rnw2pdf-guide-sweave", "rmd2pdf-guide", "rnw2pdf-report-knit", "rnw2pdf-report-sweave")
dir	Directory into which files are inserted. Type "." for current working directory. Default is a new directory with name equal to the type parameter

Value

The normalized path of the new directory

Details

The examples demonstrate all of the mix-and-match combinations of input document formats, output formats, and document types. The issues involved are more fully explained in the vignettes provided with the package, but a nutshell summary would be as follows.

Report or Guide?

In our terminology, a guide is a document that includes "raw" code examples and possibly output. A guide is intended for training/teaching purposes, it might be offered on <http://crmda.ku.edu/guides>. In contrast, a report generally will now include "raw" code and only very rarely will it have raw output. Calculations should create tables and plots that are inserted as (in LaTeX, floating) tables and figures.

Markdown ("*.Rmd") or Noweb ("*.Rnw")?

Documents can be prepared in markdown "Rmd" or R noweb Rnw. Either type of document can include code chunks. For creating HTML output, markdown is perhaps the most workable format. For creating PDF output, the noweb document format is better. When PDF output is desired, a markdown document must be converted to tex and then to PDF. Some LaTeX features are lost in that process, but they are not lost if we prepare the document in a noweb format (which *is* a LaTeX file). The details are discussed in the vignettes available with this package.

PDF or HTML?

The allowed components in a report or guide depend on whether the eventual output is HTML or PDF. Many features intended for PDF output will not work if the backend is changed to HTML. The converse is also true. While this may change in the future, at the current time, only the most basic document which does not include most of the features that we truly need will be compatible with both PDF and HTML output. That's why we need to have so many mix-and match combinations of document types and output formats.

We may have slide templates as well, at some point in future.

Author(s)

Paul Johnson <pauljohn@ku.edu>

Examples

```
tdir <- tempdir()
cat("Will Create All Doc Types in Temp Dir:\n ", tdir, "\n")

doctype <- c("rmd2html-guide", "rmd2pdf-report",
             "rnw2pdf-guide-knit", "rnw2pdf-guide-sweave",
             "rmd2pdf-guide", "rnw2pdf-report-knit",
             "rnw2pdf-report-sweave", "rnw2pdf-slides-sweave")
folders <- vapply(doctype, function(x){
  initWriteup(x, dir = file.path(tdir, "todaytest", x))},
  character(1))
folders
list.files(file.path(tdir, "todaytest"), recursive = TRUE)
## Compile one example in the rnw2pdf-report-sweave folder
list.files(file.path(tdir, "todaytest/rnw2pdf-report-sweave"))

## requires pdflatex on system
rnw2pdf("skeleton.Rnw", wd = file.path(tdir, "todaytest/rnw2pdf-report-sweave"),
        engine = "Sweave")
## Check the pdf was created
list.files(file.path(tdir, "todaytest/rnw2pdf-report-sweave"))
if(interactive()) browseURL(file.path(tdir, "todaytest/rnw2pdf-report-sweave", "skeleton.pdf"))
unlink(file.path(tdir, "todaytest"), recursive = TRUE)
```

rmd2html

Convert an Rmd file into an HTML file

Description

This is a very simple wrapper around the `rmarkdown::render` function. It makes sure that the style sheet we want to use is applied to the data.

Usage

```
rmd2html(
  fn = NULL,
  wd = NULL,
  ...,
  verbose = FALSE,
  pur1 = TRUE,
  tangle = pur1,
  backup = TRUE
)
```


Arguments

fn	One or more filenames ending in "*.Rmd".
wd	A working directory name of the Rmd file. If not specified, then the current working directory from R will be used.
...	Arguments that will be passed to <code>rmarkdown::render</code> and <code>rmarkdown::html_document</code> . We usually have customizations via parameters <code>css</code> and <code>template</code> , but many other parameters can be specified to change arguments for <code>rmarkdown::render</code> and <code>rmarkdown::html_documents</code> . These possible arguments for <code>html_document</code> : <code>c("toc", "toc_depth", "toc_float", "number_sections", "section_divs", "fig_width", "fig_height", "fig_retina", "fig_caption", "dev", "df_print", "code_folding", "code_download", "smart", "self_contained", "theme", "highlight", "mathjax", "template", "extra_dependencies", "css", "includes", "keep_md", "lib_dir", "md_extensions", "pandoc_args")</code> . These arguments intended for <code>render()</code> are allowed: <code>c("output_file", "output_dir", "output_options", "intermediates_dir", "knit_root_dir", "runtime", "clean", "params", "knit_meta", "envir", "run_pandoc", "quiet", "encoding")</code> .
verbose	The opposite of <code>render(quiet = TRUE)</code> . Shows compile commentary and pandoc command. Can be informative!
purl	Default TRUE, synonym for <code>tangle</code> . Set either one, or set both same, result is same.
tangle	Default TRUE, synonym for <code>purl</code>
backup	Default TRUE. Will create backup copies of pdf, R, Rnw files before replacing them. Uses <code>kutils::file.backup</code>

Details

Running this will be the same as running the `rmd2html.sh` script within the directory.

Value

A vector of output file names

Author(s)

Paul Johnson <pauljohn@ku.edu>

Examples

```
tdir <- tempdir()
doctype <- "rmd2html-guide"
dirout <- initWriteup(doctype, dir = file.path(tdir, doctype))
list.files(dirout)

result <- try(rmd2html("skeleton.Rmd", wd = dirout))
if(inherits(result, "try-error")){
  MESSG <- paste("Compiling the markdown file failed, perhaps",
                "your version of pandoc is not found")
  print(MESSG)
```

```

} else {
  ## Check the result file:
  MESSG <- paste("Check the directory", dirout, "for results.")
  print(MESSG)
  list.files(dirout)
  if(interactive() && file.exists(file.path(dirout, "skeleton.html"))) {
    browseURL(file.path(dirout, "skeleton.html"))
  }
}
unlink(dirout)

```

rmd2pdf

Convert Rmd to PDF

Description

Convert an Rmd file to PDF by Sweaveing or knitting and then compiling.

Usage

```

rmd2pdf(
  fn = NULL,
  wd = NULL,
  ...,
  verbose = FALSE,
  purl = TRUE,
  tangle = purl,
  backup = TRUE
)

```

Arguments

fn	One or more filenames ending in <code>"*.Rmd"</code> .
wd	A working directory in which the Rmd file exists. Leave as <code>NULL</code> if file is in current working directory,
...	Arguments that will be passed to <code>rmarkdown::render</code> and <code>rmarkdown::pdf_document</code> . Our defaults set a LaTeX template, <code>toc = TRUE</code> , and the <code>pandoc_args</code> includes use of the listings class. Users may override by specifying named arguments for <code>render()</code> : <code>c("output_file", "output_dir", "output_options", "intermediates_dir", "knit_root_dir", "runtime", "clean", "params", "knit_meta", "envir", "run_pandoc", "quiet", "encoding")</code> . Users may also specify named arguments for <code>pdf_document</code> : <code>("toc", "toc_depth", "number_sections", "fig_width", "fig_height", "fig_crop", "fig_caption", "dev", "df_print", "highlight", "template", "keep_tex", "latex_engine", "citation_package", "includes", "md_extensions", "pandoc_args", "extra_</code>
verbose	The opposite of <code>render(quiet = TRUE)</code> . Shows compile commentary and pandoc command. Can be informative!

pur1	Default TRUE
tangle	Default TRUE, synonym for pur1
backup	Default TRUE. Will create backup copies of pdf, R, Rnw files before replacing them. Uses <code>kutils::file.backup</code>

Details

Running this will be the same as running the `rmd2pdf.sh` script within the directory.

Value

A vector of output file names

Author(s)

Paul Johnson <pauljohn@ku.edu>

Examples

```
tdir <- tempdir()
fmt <- "rmd2pdf-guide"
dirout <- initWriteup(fmt, dir = file.path(tdir, fmt))
print(dirout)
list.files(dirout)

of1 <- try(rmd2pdf("skeleton.Rmd", wd = dirout))
if(inherits(of1, "try-error")){
  MESSG <- paste("Compiling the markdown file failed, perhaps",
                 "you should run with parameters verbose=TRUE",
                 "and keep_tex=TRUE")
  print(MESSG)
} else {
  ## Check the result file:
  MESSG <- paste("Check the directory", dirout, "for results.")
  print(MESSG)
  list.files(dirout)
  if(interactive() && file.exists(file.path(dirout, "skeleton.pdf"))) {
    browseURL(of1)
  }
}

unlink(dirout)
```

Description

Documents saved with suffix `".lyx"` or `".Rnw"` will be converted. Note it is very important to specify the engine for the code chunks correctly, this can be either `"Sweave"` or `"knitr"`.

Usage

```
rnw2pdf(
  fn = NULL,
  wd = NULL,
  ...,
  engine = "knitr",
  purl = TRUE,
  tangle = purl,
  backup = TRUE,
  clean = TRUE,
  quiet = TRUE,
  verbose = !quiet,
  envir = parent.frame(),
  encoding = getOption("encoding")
)
```

Arguments

fn	One or more file names, should end in either ".Rnw" or ".lyx"
wd	Directory in which the file to be converted exists. Leave NULL default if is in current working directory.
...	Other parameters, not used at the moment.
engine	"knitr" or "Sweave"
purl	Default TRUE. Synonym of tangle: extract R code chunks
tangle	Same as purl, both parameters have same result
backup	Default TRUE. Will create backup copies of pdf, R, Rnw files before replacing them. Uses <code>kutils::file.backup</code>
clean	Default TRUE. Remove intermediate LaTeX files when using <code>texi2pdf</code>
quiet	Default = TRUE. No output unless an error occurs. Antonym for verbose.
verbose	Default = !quiet. Antonym for quiet. Functions try to reduce amount of screen output. Knitr functions that use quiet flag will be set to !verbose.
envir	environment for evaluation, see <code>knitr</code> documents, defaults to <code>parent.frame()</code> .
encoding	character encoding, defaults from user options

Value

names of pdf output files

Author(s)

Paul Johnson <pauljohn@ku.edu>

Examples

```
tdir <- tempdir()
fmt <- "rnw2pdf-guide-sweave"
dirout <- initWriteup(fmt, dir = file.path(tdir, fmt))
print(dirout)
list.files(dirout)
of1 <- try(rnw2pdf("skeleton.Rnw", engine = "Sweave", wd = dirout, backup = TRUE))
if(inherits(of1, "try-error")){
  MESSG <- paste("Compiling the markdown file failed, perhaps",
                 "there is an R or LaTeX error.",
                 "Run again with parameters verbose=TRUE",
                 "and clean=FALSE")
  print(MESSG)
} else {
  ## Check the result file:
  MESSG <- paste("Check the directory", dirout, "for results.")
  print(MESSG)
  list.files(dirout)
  if(interactive() && file.exists(of1)) {
    browseURL(of1)
  }
}
unlink(dirout)
```

Index

`blankPDF`, [2](#)

`blankPNG`, [3](#)

`crmda_html_document`, [4](#)

`getFiles`, [5](#)

`initWriteup`, [6](#)

`rmd2html`, [8](#)

`rmd2pdf`, [10](#)

`rnw2pdf`, [11](#)