Package: accessr (via r-universe)

September 4, 2024

Type Package

Title Command Line Tools to Produce Accessible Documents using 'R Markdown'

Version 1.0.1.9000

Date 2024-05-07

Description Provides functions to produce accessible 'HTML' slides, 'HTML', 'Word' and 'PDF' documents from input 'R markdown' files. Accessible 'PDF' files are produced only on a 'Windows' Operating System. One aspect of accessibility is providing a headings structure that is recognised by a screen reader, providing a navigational tool for a blind or partially-sighted person. A key aim is to produce documents of different formats easily from each of a collection of 'R markdown' source files. Input 'R markdown' files are rendered using the render() function from the 'rmarkdown' package <https://cran.r-project.org/package=rmarkdown>. A 'zip' file containing multiple output files can be produced from one function call. A user-supplied template 'Word' document can be used to determine the formatting of an output 'Word' document. Accessible 'PDF' files are produced from 'Word' documents using 'OfficeToPDF' <https://github.com/cognidox/OfficeToPDF>. A convenience function, install_otp() is provided to install this software. The option to print 'HTML' output to (non-accessible) 'PDF' files is also available.

Depends R (>= 3.3.0)

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.2.3

Imports rmarkdown

Suggests dplyr, knitr, flextable, htmltools, huxtable, jpeg, officer, officedown, pagedown, png, tools, utils, xfun, zip, testthat (>= 3.0.0)

SystemRequirements OfficeToPDF - https://github.com/cognidox/OfficeToPDF, pandoc (>= 1.14) - https://pandoc.org

URL https://paulnorthrop.github.io/accessr/, https://github.com/paulnorthrop/accessr

BugReports https://github.com/paulnorthrop/accessr/issues

Roxygen list(markdown = TRUE)

Config/testthat/edition 3

Repository https://paulnorthrop.r-universe.dev

RemoteUrl https://github.com/paulnorthrop/accessr

RemoteRef HEAD

RemoteSha 707c584a7a0ecb57873b32e34f5145b48dd030e2

Contents

`accessr-package` *accessr: Command Line Tools to Produce Accessible Documents using R markdown*

Description

Provides functions to produce accessible HTML and PDF documents from input R markdown files. Currently, **accessr** only provides the option to produce accessible PDF files on a Windows Operating System. One aspect of accessibility is providing a headings structure that is recognised by a screen reader, providing a navigational tool for a blind or partially-sighted person. A key aim is to produce documents of different formats from each of a collection of R markdown source files. A user-supplied template Word document can be used to determine the formatting of an output Word document. Similar functions produce HTML slides and HTML documents. A zip file containing multiple files can be produced. The option to print HTML output to (non-accessible) PDF files is also available.

Details

See the accessr package page on Github for more information. An example Rmd file is available at `system.file(package = "accessr", "examples", "example.Rmd")`.

On a Windows Operating System, Accessible PDF documents are produced by creating Word documents from R markdown files and then PDF documents from these Word documents. The first step uses the `render` function from the rmarkdown package and the `rdocx_document` function from the officedown package. The second step uses OfficeToPDF.

The main functions in `accessr` are:

- `rmd2many`: create HTML slides, PDF slides, Word and PDF documents from a single R markdown file.
- `rmd2word`: create Word documents and accessible PDF files.

  `install_otp`: convenience function to install OfficeToPDF, to create PDF files from Word documents in `rmd2word`. `ext_img`: a function to enable the `knitr` chunk options `out.width` and/or `out.height` to work when the output format is a Word document
- `rmd2ioslides`: create ioslides presentations and perhaps print to (non-accessible) PDF documents.
- `rmd2slidy`: create slidy presentations and perhaps print to (non-accessible) PDF documents..
- `rmd2html`: create html documents and perhaps print to (non-accessible) PDF documents.

The `rmd2?` functions provide the option to create a zip archive containing the output files. All the .Rmd files in a directory can be processed with one function call. Information such as `title`, `author`, `lang` etc in the YAML header in the Rmd file are used but `output` is ignored.

Author(s)

Maintainer: Paul J. Northrop <p.northrop@ucl.ac.uk> [copyright holder]

References

David Gohel and Noam Ross (2021). officedown: Enhanced 'R Markdown' Format for 'Word' and 'PowerPoint'. R package version 0.3.1. https://CRAN.R-project.org/package=officedown

JJ Allaire, Yihui Xie, Christophe Dervieux, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, Winston Chang, and Richard Iannone (2024). rmarkdown: Dynamic Documents for R. R package version 2.26, https://rmarkdown.rstudio.com.

See Also

`install_otp`, `rmd2many`, `rmd2word`, `rmd2ioslides`, `rmd2slidy`, `rmd2html`.

ext_img *External images for Word output*

Description

This is a modified version of the external_img function from the officer package for use in rmd2word. The modification is to allow the use of the knitr chunk options out.width and/or out.height to set the dimensions of a figure (R-generated or external image) when the output format is a Word document. This functionality is not normally available.

Usage

```
ext_img(src, width = 0.5, height = 0.2, alt = "", ref_docx_dim)
```

Arguments

src	image file path
width	width in inches
height	height in inches
alt	alternative text for images
ref_docx_dim	A list containing the components page (a named numeric vector containing the width and height), landscape (a logical vector) and margins (a named numeric vector containing the top, bottom, left, right, header and footer margins). See **Details**.

Details

If in the current R code chunk out.width or out.height have been set then the corresponding values of fig.width and fig.height are inferred from the dimensions of the figure and the page dimensions of the output Word document. The former are obtained using readPNG or readJPEG as appropriate. If only one of out.width or out.height are set then the aspect ratio of the figure is preserved.

ref_docx_dim can be produced using officer::docx_dim(officer::read_docx(doc)), where doc is an docx_dim object and docx_dim and read_docx are functions in the officer package.

Value

An object of class c("external_img", "cot", "run") with attributes "dims" (a named list containing the figure's width and height) and "alt" (a character scalar containing the alternative text for the figure).

Note

This function has been designed for use inside rmd2word but the object returned has the same structure as that returned from external_img. Outside the context of a knitr R code chunk the outputs from external_img and ext_img are identical.

Examples

```
# Example from the officer::external_img() documentation
# Wrap R logo with external_img ----
srcfile <- file.path( R.home("doc"), "html", "logo.jpg" )
extimg <- ext_img(src = srcfile, height = 1.06 / 2, width = 1.39 / 2)
```

`install_otp` *Install OfficeToPDF*

Description

Downloads `OfficeToPDF.exe` from https://github.com/cognidox/OfficeToPDF/releases and copies it to a suitable location, by default `system.file(package = "accessr")`.

Usage

```
install_otp(dir, url, method, quiet = TRUE, ...)
```

Arguments

`dir`	Directory into which to download the executable file `OfficeToPDF.exe`. **This argument must be supplied**. Pass `dir = "accessr"` to choose the `accessr` package directory, that is, `system.file(package = "accessr")`. This choice should enable `OfficeToPDF.exe` to be found but there may be a problem if `.libPaths` refers to a cloud storage directory, such as OneDrive. If `dir` is not supplied, e.g. the call is `install_otp()`, then an error is thrown and `dir = "accessr"` is suggested.
`url`	URL from which to download `OfficeToPDF.exe`. If this is missing then https://github.com/cognidox/OfficeToPDF/releases/download/v1.9.0.2/OfficeToPDF.exe is used.
`method`	Passed to `download.file`. If missing then `getOption("download.file.method")` is used to set it.
`quiet`	Passed to `download.file`. `quiet` controls whether messages are printed during the download.
`...`	Additional arguments passed to `download.file`.

Details

The argument `mode = "wb"` to `download.file` is hard-coded, so that binary transfer is forced.

Value

See the **Value** section of `download.file`.

Note

`OfficeToPDF.exe` is used to create PDF files from Word documents in `rmd2word`.

rmd2html *Converts R markdown code to html documents*

Description

Creates accessible html documents using R markdown's html_document argument to render. Zip archives of the html files may be created.

Usage

```
rmd2html(
  x,
  zip = if (length(x) == 1 & !add) FALSE else TRUE,
  pdf = FALSE,
  pdf_args = list(),
  zip_pdf = zip,
  add = FALSE,
  quiet = TRUE,
  rm_html = FALSE,
  rm_pdf = FALSE,
  ...
)
```

Arguments

x	A character vector containing the names (**no extension**) of the .Rmd files to convert if they are in the current working directory, or paths to the files, either absolute or relative to the current working directory, e.g., DIRECTORY/file1. The .html files are created in the same directory as their respective .Rmd file. If x is missing then an html file is created from each of the .Rmd files in the current working directory.
zip	A logical scalar or character vector indicating whether html files should be put into a zip archive. If zip = FALSE then no zip archive is created. Otherwise, an archive is created in each unique directory involved in x. If zip = TRUE each archive of html files is named accessr_html.zip. If zip is a character vector of zip file names (no extension) then these names are used to name the zip archives. The names are recycled to the length of the number of unique directories, if necessary.
pdf	A logical scalar. If pdf = TRUE then each html file is printed to a PDF file using chrome_print. Google Chrome (or an alternative browser specified in pdf_args by the browser argument to chrome_print must be installed prior to use of this option. An error message like Error in servr::random_port(NULL) : Cannot find an available TCP port means that the random_port function in the servr package could not find an internet connection that Chrome considers secure. Perhaps you are using a coffee shop's wifi.
pdf_args	A list of arguments passed to chrome_print. input cannot be passed because it is set inside rmd2html.

zip_pdf	As zip, but relates to the creation of zip archives for any PDF files created. If zip_pdf = TRUE then each archive is named accessr_html_pdf.zip.
add	A logical scalar that determines what happens if an output zip file already exists. If add = TRUE then files are added to the zip file and if add = FALSE then the zip file is deleted and will only contain newly-created files.
quiet	Argument of the same name passed to render to determine what is printed during rendering from knitr.
rm_html	A logical scalar. If rm_html = TRUE and a zip archive of html files is produced then the individual html files are deleted. Otherwise, they are not deleted.
rm_pdf	A logical scalar. If rm_pdf = TRUE and a zip archive of pdf files is produced then the individual pdf files are deleted. Otherwise, they are not deleted.
...	Additional arguments passed to html_document or render.

Details

Information such as title, author, lang etc in the YAML header in the Rmd file are used but output is ignored.

The simplest setup is to have the .Rmd files in the current working directory, in which case rmd2html() will create HTML documents from all these Rmd files, the .Rmd files may be in different directories.

The render function, with the argument output_file = html_document creates the html files.

Value

In addition to creating the html files, and perhaps zip files, a list containing the following (character vector) components is returned invisibly:

files	(absolute) paths and file names of the files added to a zip file.
zips	(relative) paths and names of all the zip files.

See Also

rmd2many, rmd2word, rmd2ioslides, rmd2slidy for other output formats.

The accessr package page on Github.

Examples

```
# Create an HTML document from example.Rmd
got_hux <- requireNamespace("huxtable", quietly = TRUE)
got_flex <- requireNamespace("flextable", quietly = TRUE)
got_pandoc <- rmarkdown::pandoc_available("1.14")
got_all <- got_hux && got_flex && got_pandoc
# This example needs packages huxtable and flextable
if (got_all) {
  ex_file <- system.file(package = "accessr", "examples", "example.Rmd")
  file.copy(ex_file, tdir <- tempdir(check = TRUE), overwrite = TRUE)
  ex_file <- list.files(tdir, pattern = "example.Rmd", full.names = TRUE)
  ex_file <- sub(".Rmd", "", ex_file)
  rmd2html(ex_file)
}
```

`rmd2ioslides` *Converts R markdown code to ioslides html presentations*

Description

Creates accessible html ioslides presentations using R markdown's `ioslides_presentation` argument to `render`. Zip archives of the html files may be created.

Usage

```
rmd2ioslides(
  x,
  zip = if (length(x) == 1 & !add) FALSE else TRUE,
  pdf = FALSE,
  zip_pdf = zip,
  pdf_args = list(),
  add = FALSE,
  quiet = TRUE,
  rm_html = FALSE,
  rm_pdf = FALSE,
  inc_rmd = FALSE,
  params = NULL,
  ...
)
```

Arguments

`x`	A character vector containing the names (**no extension**) of the `.Rmd` files to convert if they are in the current working directory, or paths to the files, either absolute or relative to the current working directory, e.g., `DIRECTORY/file1`. The `.html` files are created in the same directory as their respective `.Rmd` file. If `x` is missing then an html file is created from each of the `.Rmd` files in the current working directory.
`zip`	A logical scalar or character vector indicating whether html files should be put into a zip archive. If `zip = FALSE` then no zip archive is created. Otherwise, an archive is created in each unique directory involved in `x`. If `zip = TRUE` then any archive created has the name `accessr_ioslides.zip`. If `zip` is a character vector of zip file names (no extension) then these names are used to name the zip archives. The names are recycled to the length of the number of unique directories, if necessary.
`pdf`	A logical scalar. If `pdf = TRUE` then each html file is printed to a PDF file using `chrome_print`. Google Chrome (or an alternative browser specified in `pdf_args` by the `browser` argument to `chrome_print` must be installed prior to use of this option. An error message like `Error in servr::random_port(NULL) : Cannot find an available TCP port` means that the `random_port` function in the `servr` package could not find an internet connection that Chrome considers secure. Perhaps you are using a coffee shop's wifi.

zip_pdf	As zip, but relates to the creation of zip archives for any PDF files created. If zip_pdf = TRUE then each archive is named accessr_ioslides_pdf.zip.
pdf_args	A list of arguments passed to chrome_print. input cannot be passed because it is set inside rmd2html.
add	A logical scalar that determines what happens if the output zip file already exists. If add = TRUE then files are added to the zip file and if add = FALSE then the zip file is deleted and will only contain newly-created files.
quiet	Argument of the same name passed to render to determine what is printed during rendering from knitr.
rm_html	A logical scalar. If rm_html = TRUE and a zip archive of html files is produced then the individual html files are deleted. Otherwise, they are not deleted.
rm_pdf	A logical scalar. If rm_pdf = TRUE and a zip archive of pdf files is produced then the individual pdf files are deleted. Otherwise, they are not deleted.
inc_rmd	A logical scalar. If inc_rmd = TRUE then the source Rmd files are included in the zip file created. Otherwise, they are not included.
params	A list of named parameters to pass as the argument params to render.
...	Additional arguments passed to ioslides_presentation. Pass (the default) slide_level = 1 if you want a level one header # to create a new non-segue slide. If css = "black" is passed then accessr's css file black.css is used, which results in black text being used in the slides. This function is **not** vectorised with respect to arguments in ....

Details

Information such as title, author, lang etc in the YAML header in the Rmd file are used but output is ignored.

The simplest setup is to have the .Rmd files in the current working directory, in which case rmd2ioslides() will create ioslides presentations from all these Rmd files, but the .Rmd files may be in different directories.

The render function, with the argument output_file = ioslides_presentation creates the ioslides html files.

The function ioslides_presentation has an argument slide_level that sets the header level used as a slide separator. The Lua filter ioslides_presentation.lua in the rmarkdown package uses any content between headers of level slide_level to create a segue slide, which has a grey background and is intended only to contain a section heading.

In particular, under the default, slide_level = 2, content between a level one header # and the next level two header ## is formatted as a separate grey segue slide. If we do not want segue slides then we must avoid using level one headers. However, for reasons of document accessibility, we may want to use level one headers to separate slides. For example, if we wish to create an ioslides presentation and a Word document from the same source Rmd file then the Word document will only meet fully accessibility requirements if the headings in the document start at level one.

In rmarkdown version 2.26, passing slide_level = 1 to ioslides_presentation does **not** force a new non-segue slide when a level one header # is used: it places all content between # and the next

on a grey segue slide and the behaviour content of the resulting slides is not as intended. Passing `slide_level = 1` to `rmd2ioslides()` replaces `rmarkdown`'s Lua filter `ioslides_presentation.lua` with one that has been modified, so that passing `slide_level = 1` **will** start a new non-segue slide. A modified `default.css` css file is also used that adjusts the font sizes for the header levels accordingly, so that the level one header has a larger font than the level two header. For values of `slide_level` greater than or equal to 2, `ioslides_presentation` will behave as usual.

If `slide_level` is not supplied then `slide_level = 1` is used.

Value

In addition to creating the html files, and perhaps zip files, a list containing the following (character vector) components is returned invisibly:

`files`	(absolute) paths and file names of the files added to a zip file.
`zips`	(relative) paths and names of all the zip files.

See Also

`rmd2many`, `rmd2word`, `rmd2slidy`, `rmd2html` for other output formats.

The accessr package page on Github.

Examples

```
# Create an ioslides presentation from example.Rmd
got_hux <- requireNamespace("huxtable", quietly = TRUE)
got_flex <- requireNamespace("flextable", quietly = TRUE)
got_pandoc <- rmarkdown::pandoc_available("1.14")
got_all <- got_hux && got_flex && got_pandoc
# This example needs packages huxtable and flextable
if (got_all) {
  ex_file <- system.file(package = "accessr", "examples", "example.Rmd")
  file.copy(ex_file, tdir <- tempdir(check = TRUE), overwrite = TRUE)
  ex_file <- list.files(tdir, pattern = "example.Rmd", full.names = TRUE)
  ex_file <- sub(".Rmd", "", ex_file)
  rmd2ioslides(ex_file)
}
```

`rmd2many` *Create content in multiple formats*

Description

From a single R markdown file create HTML slides, PDF slides, Word and PDF documents.

Usage

```
rmd2many(
  x,
  outputs = c("ioslides", "word"),
  slide_level = 1,
  css = "black",
  add18 = TRUE,
  pdf = TRUE,
 highlight = list(word = "monochrome", ioslides = NULL, slidy = NULL, html = NULL),
  params = NULL,
  zip = TRUE,
  ...
)
```

Arguments

x	A character vector containing the names (**no extension**) of the .Rmd files to convert if they are in the current working directory, or paths to the files, either absolute or relative to the current working directory, e.g., DIRECTORY/file1. The output files are created in the same directory as their respective .Rmd file.
outputs	A character vector. Specifies the output formats required. A subset of c("word", "ioslides", "slidy", "html"). If more than one of "ioslides", "slidy" and "html" are present then only one of these is used with the order of preference "ioslides", "slidy" then "html".
slide_level	Passed to rmd2ioslides via .... The default slide_level = 1 means that a level one header # create a new non-segue slide for an ioslides presentation.
css	The argument css passed to ioslides_presentation or slidy_presentation. If css = "black" then accessr's css file black.css is used, which results in black text being used in the slides. css is not used if outputs = html.
add18	A logical scalar. If TRUE then we also create Word documents with 18pt text.
pdf	A logical scalar. If TRUE then we use chrome_print to print PDF versions of HTML files produced using the output "ioslides" or "slidy". and/or OfficeToPDF.exe to create PDF files from any Word documents that are produced.
highlight	A named list, with names a subset of c("word", "ioslides", "slidy"), providing the respective syntax highlighting styles passed to Pandoc for the output formats. Any syntax highlighting provided in css will take precedence. highlight is not used if outputs = html.
params	A list of named parameters to pass as the argument params to render. In the example below, the file example.Rmd has a parameter hide. If hide = TRUE then parts of the output are hidden using the knitr chunk options echo and eval.
zip	A logical scalar or character vector indicating whether PDF files should be put into a zip archive. If zip = FALSE then no zip archive is created. Otherwise, an archive is created in each unique directory involved in x. If zip = TRUE then any archive created is named after the first filename in x from the relevant directory. If zip is a character vector of zip file names (no extension) then these names

are used to name the zip archives. The names are recycled to the length of the number of unique directories if necessary.

... Additional arguments to be passed to rmd2ioslides, rmd2slidy, rmd2word or rmd2html.

Details

The default setting creates, for each valid filename in x, the following files

- filename.html: lecture slides in ioslides format.
- filename_slides.pdf: a PDF document containing the content in filename.html.
- filename.pdf: a PDF document created from a Word document produced by rmd2word.
- filename.docx: a Word document.
- filename18pt.docx: a Word document. If add18 = TRUE then a template Word document with 18pt bold text is used.
- filename.zip: a zip file containing all the files produced.

Value

A list containing the following components:

files	names of all the files created.
zips	names of all zip files created (if zip = TRUE).

See Also

install_otp to install OfficeToPDF.

rmd2word, rmd2ioslides, rmd2slidy, rmd2html.

The accessr package page on Github.

Examples

```
# Create documents from example.Rmd
got_hux <- requireNamespace("huxtable", quietly = TRUE)
got_flex <- requireNamespace("flextable", quietly = TRUE)
got_pandoc <- rmarkdown::pandoc_available("1.14")
got_all <- got_hux && got_flex && got_pandoc
# This example needs packages huxtable and flextable
# We pass pdf = FALSE because OfficeToPDF is needed to convert Word to PDF
# and this is only relevant on  Windows Operating System.
#
if (got_all) {
  ex_file <- system.file(package = "accessr", "examples", "example.Rmd")
  file.copy(ex_file, tdir <- tempdir(check = TRUE), overwrite = TRUE)
  ex_file <- list.files(tdir, pattern = "example.Rmd", full.names = TRUE)
  ex_file <- sub(".Rmd", "", ex_file)
  rmd2many(ex_file, params = list(hide = TRUE), pdf = FALSE, zip = TRUE)
}
```

`rmd2slidy` *Converts R markdown code to slidy html presentations*

Description

Creates accessible html slidy presentations using R markdown's `slidy_presentation` argument to `render`. Zip archives of the html files may be created.

Usage

```
rmd2slidy(
  x,
  zip = if (length(x) == 1 & !add) FALSE else TRUE,
  pdf = FALSE,
  zip_pdf = zip,
  pdf_args = list(),
  add = FALSE,
  quiet = TRUE,
  rm_html = FALSE,
  rm_pdf = FALSE,
  inc_rmd = FALSE,
  params = NULL,
  ...
)
```

Arguments

`x`	A character vector containing the names (**no extension**) of the `.Rmd` files to convert if they are in the current working directory, or paths to the files, either absolute or relative to the current working directory, e.g., `DIRECTORY/file1`. The `.html` files are created in the same directory as their respective `.Rmd` file. If `x` is missing then an html file is created from each of the `.Rmd` files in the current working directory.
`zip`	A logical scalar or character vector indicating whether html files should be put into a zip archive. If `zip = FALSE` then no zip archive is created. Otherwise, an archive is created in each unique directory involved in `x`. If `zip = TRUE` then any archive created has the name `accessr_slidy.zip`. If `zip` is a character vector of zip file names (no extension) then these names are used to name the zip archives. The names are recycled to the length of the number of unique directories, if necessary.
`pdf`	A logical scalar. If `pdf = TRUE` then each html file is printed to a PDF file using `chrome_print`. Google Chrome (or an alternative browser specified in `pdf_args` by the `browser` argument to `chrome_print` must be installed prior to use of this option. An error message like `Error in servr::random_port(NULL) : Cannot find an available TCP port` means that the `random_port` function in the `servr` package could not find an internet connection that Chrome considers secure. Perhaps you are using a coffee shop's wifi.

zip_pdf	As zip, but relates to the creation of zip archives for any PDF files created. If zip_pdf = TRUE then each archive is named accessr_html_pdf.zip.
pdf_args	A list of arguments passed to chrome_print. input cannot be passed because it is set inside rmd2html.
add	A logical scalar that determines what happens if the output zip file already exists. If add = TRUE then files are added to the zip file and if add = FALSE then the zip file is deleted and will only contain newly-created files.
quiet	Argument of the same name passed to render to determine what is printed during rendering from knitr.
rm_html	A logical scalar. If rm_html = TRUE and a zip archive of html files is produced then the individual html files are deleted. Otherwise, they are not deleted.
rm_pdf	A logical scalar. If rm_pdf = TRUE and a zip archive of pdf files is produced then the individual pdf files are deleted. Otherwise, they are not deleted.
inc_rmd	A logical scalar. If inc_rmd = TRUE then the source Rmd files are included in the zip file created. Otherwise, they are not included.
params	A list of named parameters to pass as the argument params to render.
...	Additional arguments passed to slidy_presentation. If css = "black" is passed then accessr's css file black.css is used, which results in black text being used in the slides. This function is **not** vectorised with respect to arguments in ....

Details

Information such as title, author, lang etc in the YAML header in the Rmd file are used but output is ignored.

The simplest setup is to have the .Rmd files in the current working directory, in which case rmd2slidy() will create slidy presentations from all these Rmd files, but the .Rmd files may be in different directories.

The render function, with the argument output_file = slidy_presentation creates the slidy html files.

Value

In addition to creating the html files, and perhaps zip files, a list containing the following (character vector) components is returned invisibly:

files	(absolute) paths and file names of the files added to a zip file.
zips	(relative) paths and names of all the zip files.

See Also

rmd2word, rmd2word, rmd2ioslides, rmd2html for other output formats.

The accessr package page on Github.

Examples

```
# Create a slidy presentation from example.Rmd
got_hux <- requireNamespace("huxtable", quietly = TRUE)
got_flex <- requireNamespace("flextable", quietly = TRUE)
got_pandoc <- rmarkdown::pandoc_available("1.14")
got_all <- got_hux && got_flex && got_pandoc
# This example needs packages huxtable and flextable
if (got_all) {
  ex_file <- system.file(package = "accessr", "examples", "example.Rmd")
  file.copy(ex_file, tdir <- tempdir(check = TRUE), overwrite = TRUE)
  ex_file <- list.files(tdir, pattern = "example.Rmd", full.names = TRUE)
  ex_file <- sub(".Rmd", "", ex_file)
  rmd2slidy(ex_file)
}
```

---

rmd2word *Converts R markdown code to Word and PDF documents*

Description

Creates Word documents from input R markdown documents. On a Windows Operating System, accessible PDF documents may be created from these Word files if the software OfficeToPDF is installed. The convenience function `install_otp` can be used to install this software. Zip archives of the Word and/or PDF files may be created.

Usage

```
rmd2word(
  x,
  doc = "accessr",
  pdf = isTRUE(.Platform$OS.type == "windows"),
  dir,
  zip = if (length(x) == 1 & !add) FALSE else TRUE,
  add = FALSE,
  quiet = TRUE,
  rm_word = FALSE,
  rm_pdf = FALSE,
  inc_word = FALSE,
  params = NULL,
  ...
)
```

Arguments

x	A character vector containing the names (**no extension**) of the `.Rmd` files to convert if they are in the current working directory, or paths to the files, either absolute or relative to the current working directory, e.g., `DIRECTORY/file1`. The `.html` files are created in the same directory as their respective `.Rmd` file. If

x is missing then an html file is created from each of the .Rmd files in the current working directory.

doc — An optional character vector (**including the file extension**) to specify template Word documents on which to base the style of the respective output Word documents. This determines what is passed as the argument reference_docx to word_document, via officedown::rdocx_document. Different templates may be used for different files. rep_len(doc, length(x)) is used to force length(doc) to have the same length as x. See **Details** for some built-in options.

pdf — A logical scalar. Should OfficeToPDF.exe be used to create PDF files from the Word documents that are produced? If pdf = FALSE then any zip archives created contain only Word files. PDF files will only be produced if the Operating System is "windows", that is, .Platform$OS.type == "windows".

dir — A path to the directory in which the file OfficeToPDF.exe sits. This is not needed if this file sits in the current working directory or a directory in the list returned by searchpaths(). Otherwise, it may be a path relative to the current working directory or an absolute path. If dir is missing then rmd2word will look in system.file(package = "accessr"), which is the default installation location of install_otp.

zip — A logical scalar or character vector indicating whether PDF files should be put into a zip archive. If zip = FALSE then no zip archive is created. Otherwise, an archive is created in each unique directory involved in x. If zip = TRUE then any archive created has the name accessr_word.zip. If zip is a character vector of zip file names (no extension) then these names are used to name the zip archives. The names are recycled to the length of the number of unique directories if necessary.

add — A logical scalar that determines what happens if the output zip file already exists. If add = TRUE then files are added to the zip file and if add = FALSE then the zip file is deleted and will only contain newly-created files.

quiet — A logical scalar. Passed to render as the argument quiet. The default, quite = TRUE suppresses all printing during rendering of the document.

rm_word — A logical scalar. If rm_word = TRUE then all the Word files created are deleted. Otherwise, they are not deleted.

rm_pdf — A logical scalar. If rm_pdf = TRUE and a zip archive of PDF files is produced then the individual PDF files are deleted. Otherwise, they are not deleted.

inc_word — A logical scalar. If inc_word = TRUE then the Word files are included in the zip file created. Otherwise, they are not included.

params — A list of named parameters to pass as the argument params to render.

... — Additional arguments passed to word_document.

Details

Information such as title, author, lang etc in the YAML header in the Rmd file are used but output is ignored.

The simplest setup is to have the .Rmd files in the current working directory, in which case rmd2word() will create Word documents from all these Rmd files, but the .Rmd files may be in different directories.

It is possible to have the .Rmd files in different directories, but any non-"default" values in doc must be such that the reference_docx argument of word_document finds a template Word file. If the template is in the same directory as its respective .Rmd component in x then the filename, e.g. "template.docx" will suffice. Otherwise, a path to the template should be given, either relative to the directory in which the .Rmd file sits, or an absolute path.

For information on how to create a template Word document see Richard Layton's guide Happy collaboration with Rmd to docx.

There are some built-in template options:

- doc = "officedown": uses rdocx_document's default,
- doc = "accessr": similar to "officedown" but with narrower margins and black text for titles and darker hyperlinks, to avoid contrast issues,
- doc = "18": like "accessr" but with 18pt text,
- doc = "exam": creates a Word file with a header "Examination paper for STAT0002" on the left and "Page x of n" on the right.

To use your own template(s), provide their filename(s). A component equal to "officedown" chooses rdocx_document's default. A component equal to "accessr" chooses accessr's internal template file, which has narrower margins and darker blue fonts for titles and hyperlinks, to avoid contrast issues. To use your own template(s), provide their filenames. See **Details** for more information.

The render function creates a Word file from each input .Rmd file. Then OfficeToPDF is used to convert the Word file to a PDF file. The file OfficeToPDF.exe needs to be downloaded from the OfficeToPDF releases page and placed in the directory specified by the argument dir, or in a directory that is in the list returned by searchpaths. If OfficeToPDF.exe cannot be found then an error is thrown. A warning will be given if any of the PDF files could not be produced. This will occur if there is an existing PDF file of the same name open in another application.

Value

In addition to creating the Word and PDF files, and perhaps zip files, a list containing the following vector components is returned invisibly:

error_codes	If pdf = TRUE, numeric values returned from system. If wait = FALSE then these values will be 0 (the success value) even if some of the PDF files could not be produced. The error code 17234 indicates that a PDF file was open in another application.
files	(absolute) paths and file names of all files created.
zips	(relative) paths and names of all zip files created (if zip = TRUE).

References

Layton, Richard. (2015) Happy collaboration with Rmd to docx. R Markdown from RStudio article. https://rmarkdown.rstudio.com/articles_docx.html

See Also

install_otp to install OfficeToPDF.

rmd2many, rmd2ioslides, rmd2slidy, rmd2html for other output formats.

The accessr package page on Github.

Examples

```
# Create a Word file from example.Rmd
got_hux <- requireNamespace("huxtable", quietly = TRUE)
got_flex <- requireNamespace("flextable", quietly = TRUE)
got_pandoc <- rmarkdown::pandoc_available("1.14")
got_all <- got_hux && got_flex && got_pandoc
# This example needs packages huxtable and flextable
# We pass pdf = FALSE because OfficeToPDF is needed to convert Word to PDF
# and this is only relevant on  Windows Operating System.
if (got_all) {
  ex_file <- system.file(package = "accessr", "examples", "example.Rmd")
  file.copy(ex_file, tdir <- tempdir(check = TRUE), overwrite = TRUE)
  ex_file <- list.files(tdir, pattern = "example.Rmd", full.names = TRUE)
  ex_file <- sub(".Rmd", "", ex_file)
  rmd2word(ex_file, pdf = FALSE)
}
```

Index