

Package ‘officedown’

March 7, 2022

Type Package

Title Enhanced 'R Markdown' Format for 'Word' and 'PowerPoint'

Version 0.2.4

Description Allows production of 'Microsoft' corporate documents from 'R Markdown' by reusing formatting defined in 'Microsoft Word' documents. You can reuse table styles, list styles but also add column sections, landscape oriented pages. Table and image captions as well as cross-references are transformed into 'Microsoft Word' fields, allowing documents edition and merging without issue with references; the syntax conforms to the 'bookdown' cross-reference definition. Objects generated by the 'officer' package are also supported in the 'knitr' chunks. 'Microsoft PowerPoint' presentations also benefit from this as well as the ability to produce editable vector graphics in 'PowerPoint' and also to define placeholder where content is to be added.

License MIT + file LICENSE

Encoding UTF-8

Imports knitr, rmarkdown, officer (>= 0.4.1), xml2, rlang, uuid, grDevices, yaml, utils, memoise, rvg (>= 0.2.2)

Suggests ggplot2, flextable (>= 0.7.0), bookdown (>= 0.13)

URL <https://ardata-fr.github.io/officeverse/>,
<https://davidgohel.github.io/officedown/>

BugReports <https://github.com/davidgohel/officedown/discussions>

RoxygenNote 7.1.2

SystemRequirements pandoc (>= 2.0) - <http://pandoc.org>

VignetteBuilder knitr

NeedsCompilation no

Author David Gehel [aut, cre, cph],
Institut für Qualitätssicherung und Transparenz im Gesundheitswesen
[fnd],
Noam Ross [aut] (rmarkdown implementation for rvg),
ArData [cph],
Martin Camitz [ctb]

Maintainer David Gohel <david.gohel@ardata.fr>

Repository CRAN

Date/Publication 2022-03-07 10:10:02 UTC

R topics documented:

knit_print_block	2
knit_print_run	3
rdocx_document	4
rpptx_document	9

Index	11
--------------	-----------

knit_print_block	<i>Force Block Printing while Knitting</i>
------------------	--

Description

When used in a loop, calls to blocks do not generate output because knit_print method is not called. Use the function to force printing. Also you should tell the chunk to use results 'as-is' (by adding results='asis' to your chunk header).

Usage

```
knit_print_block(x, ...)
```

Arguments

x	a block object, result of a block function from officer package
...	unused arguments

Value

None. the function only print XML code.

See Also

Other functions that force printing: [knit_print_run\(\)](#)

Examples

```
library(rmarkdown)
rmd_file_src <- system.file(
  package = "officedown", "examples", "word_loop.Rmd")
rmd_file_des <- tempfile(fileext = ".Rmd")
if(pandoc_available()){
  file.copy(rmd_file_src, to = rmd_file_des)
```

```
docx_file_1 <- tempfile(fileext = ".docx")
render(rmd_file_des, output_file = docx_file_1, quiet = TRUE)

if(file.exists(docx_file_1)){
  message("file ", docx_file_1, " has been written.")
}
}
```

knit_print_run	<i>Force Run Printing while Knitting</i>
----------------	--

Description

When used in a loop, runs do not outputs because knit_print method is not called. Use the function to force printing. Also you should tell the chunk to use results 'as-is' (by adding results='asis' to your chunk header).

Usage

```
knit_print_run(x, ...)
```

Arguments

x	a run object, result of a run function from officer package
...	unused arguments

Value

None. the function only print XML code.

See Also

Other functions that force printing: [knit_print_block\(\)](#)

Examples

```
library(rmarkdown)
rmd_file_src <- system.file(
  package = "officedown", "examples", "word_loop.Rmd")
rmd_file_des <- tempfile(fileext = ".Rmd")
if(pandoc_available()){

  file.copy(rmd_file_src, to = rmd_file_des)
  docx_file_1 <- tempfile(fileext = ".docx")
  render(rmd_file_des, output_file = docx_file_1, quiet = TRUE)

  if(file.exists(docx_file_1)){
    message("file ", docx_file_1, " has been written.")
  }
}
```

Description

Format for converting from R Markdown to an MS Word document. The function comes also with improved output options.

Usage

```
rdocx_document(
  base_format = "rmarkdown::word_document",
  tables = list(),
  plots = list(),
  lists = list(),
  mapstyles = list(),
  page_size = list(),
  page_margins = list(),
  reference_num = TRUE,
  ...
)
```

Arguments

- | | |
|-------------|---|
| base_format | <p>a scalar character, format to be used as a base document for officedown. default to word_document but can also be <code>word_document2</code> from <code>bookdown</code>.</p> <p>When the <code>base_format</code> used is <code>bookdown::word_document2</code>, the <code>number_sections</code> parameter is automatically set to <code>FALSE</code>. Indeed, if you want numbered titles, you are asked to use a Word document template with auto-numbered titles (the title styles of the default 'rdocx_document' template are already set to <code>FALSE</code>).</p> |
| tables | <p>a list that can contain few items to style tables and table captions. Missing items will be replaced by default values. Possible items are the following:</p> <ul style="list-style-type: none"> • <code>style</code>: the Word stylename to use for tables. • <code>layout</code>: 'autofit' or 'fixed' algorithm. See table_layout. • <code>width</code>: value of the preferred width of the table in percent (base 1). • <code>topcaption</code>: caption will appear before (on top of) the table, • <code>tab.lp</code>: caption table sequence identifier. All table captions are supposed to have the same identifier. It makes possible to insert list of tables. It is also used to prefix your 'bookdown' cross-reference call; if <code>tab.lp</code> is set to "tab:", a cross-reference to table with id "xxxxx" is written as <code>\@ref(tab:xxxxx)</code>. It is possible to set the value to your default Word value (in French for example it is "Tableau", in German it is "Tabelle"), you can then add manually a list of tables (go to the "References" tab and select menu "Insert Table of Figures"). • <code>caption</code>; caption options, i.e.: |

- style: Word stylename to use for table captions.
- pre: prefix for numbering chunk (default to "Table ").
- sep: suffix for numbering chunk (default to ": ").
- tnd: (only applies if positive.)Inserts the number of the last title of level tnd (i.e. 4.3-2 for figure 2 of chapter 4.3).
- tns: separator to use between title number and table number. Default is "-".
- fp_text: text formatting properties to apply to caption prefix - see `fp_text_lite()`.
- conditional: a list of named logical values:
 - first_row and last_row: apply or remove formatting from the first or last row in the table
 - first_column and last_column: apply or remove formatting from the first or last column in the table
 - no_hband and no_vband: don't display odd and even rows or columns with alternating shading for ease of reading.

Default value is (in YAML format):

```

style: Table
layout: autofit
width: 1.0
topcaption: true
tab.lp: 'tab:'
caption:
  style: Table Caption
  pre: 'Table '
  sep: ': '
  tnd: 0
  tns: '-'
  fp_text: !expr officer::fp_text_lite(bold = TRUE)
conditional:
  first_row: true
  first_column: false
  last_row: false
  last_column: false
  no_hband: false
  no_vband: true

```

plots

a list that can contain few items to style figures and figure captions. Missing items will be replaced by default values. Possible items are the following:

- style: the Word stylename to use for plots.
- align: alignment of figures in the output document (possible values are 'left', 'right' and 'center').
- topcaption: caption will appear before (on top of) the figure,
- fig.lp: caption figure sequence identifier. All figure captions are supposed to have the same identifier. It makes possible to insert list of figures. It is also used to prefix your 'bookdown' cross-reference call; if fig.lp

is set to "fig:", a cross-reference to figure with id "xxxxx" is written as \@ref(fig:xxxxx). It is possible to set the value to your default Word value (in French for example it is "Figure"), you can then add manually a list of figures (go to the "References" tab and select menu "Insert Table of Figures").

- caption; caption options, i.e.:
 - style: Word stylename to use for figure captions.
 - pre: prefix for numbering chunk (default to "Figure ").
 - sep: suffix for numbering chunk (default to ": ").
 - tnd: (only applies if positive.)Inserts the number of the last title of level tnd (i.e. 4.3-2 for figure 2 of chapter 4.3).
 - tns: separator to use between title number and figure number. Default is "-".
 - fp_text: text formatting properties to apply to caption prefix - see [fp_text_lite\(\)](#).

Default value is (in YAML format):

```
style: Normal
align: center
topcaption: false
fig.lp: 'fig:'
caption:
  style: Image Caption
  pre: 'Figure '
  sep: ': '
  tnd: 0
  tns: '-'
  fp_text: !expr officer::fp_text_lite(bold = TRUE)
```

lists	<p>a list containing two named items <code>ol.style</code> and <code>ul.style</code>, values are the style-names to be used to replace the style of ordered and unordered lists created by pandoc. If NULL, no replacement is made.</p> <p>Default value is <code>list(ol.style = NULL, ul.style = NULL)</code>:</p> <pre>ol.style: null ul.style: null</pre>
mapstyles	<p>a named list of style to be replaced in the generated document. <code>list("Normal" = c("Author", "Date"))</code> will result in a document where all paragraphs styled with stylename "Date" and "Author" will be then styled with stylename "Normal".</p>
page_size, page_margins	<p>default page and margins dimensions, these values are used to define the default Word section. See page_size() and page_mar().</p>
reference_num	<p>if TRUE, text for references to sections will be the section number (e.g. '3.2'). If FALSE, text for references to sections will be the text (e.g. 'section title').</p>
...	<p>arguments used by word_document</p>

Value

R Markdown output format to pass to [render](#)

Finding stylenames

You can access them in the Word template used. Function `styles_info()` can let you read these styles.

You need `officer` to read the stylenames (to get information from a specific "reference_docx", change `ref_docx_default` in the example below).

```
library(officer)
docx_file <- system.file(package = "officer", "template", "template.docx")
doc <- read_docx(docx_file)
```

To read paragraph stylenames:

```
styles_info(doc, type = "paragraph")
```

To read table stylenames:

```
styles_info(doc, type = "table")
```

To read list stylenames:

```
styles_info(doc, type = "numbering")
```

R Markdown yaml

The following demonstrates how to pass arguments in the R Markdown yaml:

```
---
output:
  officedown::rdocx_document:
    reference_docx: pandoc_template.docx
    tables:
      style: Table
      layout: autofit
      width: 1.0
      topcaption: true
      tab.lp: 'tab:'
      caption:
        style: Table Caption
        pre: 'Table '
        sep: ': '
        tnd: 0
        tns: '-'
      fp_text: !expr officer::fp_text_lite(bold = TRUE)
    conditional:
      first_row: true
```

```

    first_column: false
    last_row: false
    last_column: false
    no_hband: false
    no_vband: true
plots:
  style: Normal
  align: center
  fig.lp: 'fig:'
  topcaption: false
  caption:
    style: Image Caption
    pre: 'Figure '
    sep: ': '
    tnd: 0
    tns: '-'
    fp_text: !expr officer::fp_text_lite(bold = TRUE)
lists:
  ol.style: null
  ul.style: null
mapstyles:
  Normal: ['First Paragraph', 'Author', 'Date']
page_size:
  width: 8.3
  height: 11.7
  orient: "portrait"
page_margins:
  bottom: 1
  top: 1
  right: 1.25
  left: 1.25
  header: 0.5
  footer: 0.5
  gutter: 0.5
reference_num: true
---
```

Examples

```

library(rmarkdown)
run_ok <- pandoc_available() &&
  pandoc_version() >= numeric_version("2.0")

if(run_ok){

# minimal example -----
example <- system.file(package = "officedown",
  "examples/minimal_word.Rmd")
rmd_file <- tempfile(fileext = ".Rmd")
file.copy(example, to = rmd_file)

```



```

docx_file_1 <- tempfile(fileext = ".docx")
render(rmd_file, output_file = docx_file_1, quiet = TRUE)
render(rmd_file, output_file = docx_file_1, quiet = TRUE,
       intermediates_dir = tempfile())

# bookdown example -----
if(require("bookdown")){

bookdown_loc <- system.file(package = "officedown", "examples/bookdown")

temp_dir <- tempfile()
# uncomment next line to get the result in your working directory
# temp_dir <- "./bd_example"

dir.create(temp_dir, showWarnings = FALSE, recursive = TRUE)
file.copy(
  from = list.files(bookdown_loc, full.names = TRUE),
  to = temp_dir,
  overwrite = TRUE, recursive = TRUE)

render_site(
  input = temp_dir, encoding = 'UTF-8',
  envir = new.env(), quiet = TRUE)

docx_file_2 <- file.path(temp_dir, "_book", "bookdown.docx")

if(file.exists(docx_file_2)){
  message("file ", docx_file_2, " has been written.")
}
}
}

```

Description

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

The function will allow you to specify the destination of your chunks in the output PowerPoint file. In this case, you must specify the layout and master for the layout you want to use, as well as the ph argument, which will allow you to specify the placeholder to be generated to place the result. Use the officer package to help you choose the identifiers to use.

This function also support Vector graphics output in an editable format (using package rvg). Wrap you R plot commands with function dml to use this graphic capability.

Usage

```
rpptx_document(
  base_format = "rmarkdown::powerpoint_presentation",
  layout = "Title and Content",
  master = "Office Theme",
  tcf = list(),
  ...
)
```

Arguments

base_format	a scalar character, format to be used as a base document for officedown. default to powerpoint_presentation but can also be powerpoint_presentation2 from bookdown
layout	default slide layout name to use
master	default master layout name where layout is located
tcf	default conditional formatting settings defined by officer::table_conditional_formatting()
...	arguments used by powerpoint_presentation

Value

R Markdown output format to pass to [render](#)

Examples

```
library(rmarkdown)
run_ok <- pandoc_available() && pandoc_version() > numeric_version("2.4")

if(run_ok){
  example <- system.file(package = "officedown",
    "examples/minimal_powerpoint.Rmd")
  rmd_file <- tempfile(fileext = ".Rmd")
  file.copy(example, to = rmd_file)

  pptx_file_1 <- tempfile(fileext = ".pptx")
  render(rmd_file, output_file = pptx_file_1)
}

if(run_ok && require("ggplot2")){
  skeleton <- system.file(package = "officedown",
    "rmarkdown/templates/powerpoint/skeleton/skeleton.Rmd")
  rmd_file <- tempfile(fileext = ".Rmd")
  file.copy(skeleton, to = rmd_file)

  pptx_file_2 <- tempfile(fileext = ".pptx")
  render(rmd_file, output_file = pptx_file_2)
}
```

Index

* functions that force printing

knit_print_block, [2](#)

knit_print_run, [3](#)

fp_text_lite(), [5](#), [6](#)

knit_print_block, [2](#), [3](#)

knit_print_run, [2](#), [3](#)

officer::table_conditional_formatting(),
[10](#)

page_mar(), [6](#)

page_size(), [6](#)

powerpoint_presentation, [10](#)

rdocx_document, [4](#)

render, [7](#), [10](#)

rpptx_document, [9](#)

styles_info(), [7](#)

table_layout, [4](#)

word_document, [4](#), [6](#)