Package 'doconv'

August 19, 2022

Type Package

Title Document Conversion to 'PDF' or 'PNG'

Version 0.1.4

Description Functions to convert 'Microsoft Word' or 'Microsoft PowerPoint' documents to 'PDF' format and also for converting them into a thumbnail. In order to work, 'LibreOffice' must be installed on the machine and or 'Microsoft Word'. If the latter is available, it can be used to produce PDF documents identical to the originals, otherwise, 'LibreOffice' is used. A function is also provided to update all fields and table of contents of a Word document using 'Microsoft Word'.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.1

Imports magick, pdftools, locatexec, processx, tools

Depends R (>= 4.0.0)

Suggests officer

BugReports https://github.com/ardata-fr/doconv/issues

SystemRequirements LibreOffice, Microsoft Word

NeedsCompilation no

Author David Gohel [aut, cre],
ArData [cph],
David Hajage [ctb] (initial powershell code)

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

Repository CRAN

Date/Publication 2022-08-19 09:30:05 UTC

R topics documented:

`check_libreoffice_export` *Check if PDF export is functional*

Description

Test if 'LibreOffice' can export to PDF. An attempt to export to PDF is made to confirm that the PDF export is functional.

Usage

```
check_libreoffice_export(UserInstallation = NULL)
```

Arguments

`UserInstallation`
: use this value to set a non-default user profile path for "LibreOffice". If not provided a temporary dir is created. It makes possibles to use more than a single session of "LibreOffice."

Value

a single logical value.

Examples

```
library(locatexec)
if(exec_available("libreoffice")){
  check_libreoffice_export()
}
```

`docx2pdf` *Convert docx to pdf*

Description

Convert docx to pdf directly using "Microsoft Word". This function will not work if "Microsoft Word" is not available on your machine.

The calls to "Microsoft Word" are made differently depending on the operating system:

- On "Windows", a "PowerShell" script using COM technology is used to control "Microsoft Word". The resulting PDF is containing a browsable TOC.
- On macOS, an "AppleScript" script is used to control "Microsoft Word". The resulting PDF is not containing a browsable TOC as when on 'Windows'.

Usage

```
docx2pdf(input, output = gsub("\\.docx$", ".pdf", input))
```

Arguments

`input, output`	file input and optional file output (default to input with pdf extension).

Value

the name of the produced pdf (the same value as `output`)

Macos manual authorizations

On macOS the call is happening into a working directory managed with function `working_directory()`.

Manual interventions are necessary to authorize 'Word' and 'PowerPoint' applications to write in a single directory: the working directory. These permissions must be set manually, this is required by the macOS security policy. We think that this is not a problem because it is unlikely that you will use a Mac machine as a server.

You must click "allow" two times to:

1. allow R to run 'AppleScript' scripts that will control Word
2. allow Word to write to the working directory.

This process is a one-time operation.

Examples

```
library(locatexec)
if (exec_available('word')) {
  file <- system.file(package = "doconv",
    "doc-examples/example.docx")
```

```
  out <- docx2pdf(input = file,
    output = tempfile(fileext = ".pdf"))

  if (file.exists(out)) {
    message(basename(out), " is existing now.")
  }
}
```

---

docx_update *Update docx fields*

Description

Update all fields and table of contents of a Word document using "Microsoft Word". This function will not work if "Microsoft Word" is not available on your machine.

The calls to "Microsoft Word" are made differently depending on the operating system. On "Windows", a "PowerShell" script using COM technology is used to control "Microsoft Word". On macOS, an "AppleScript" script is used to control "Microsoft Word".

Usage

```
docx_update(input)
```

Arguments

input	file input

Value

the name of the produced pdf (the same value as output)

Examples

```
library(locatexec)
if (exec_available('word')) {
  file <- system.file(package = "doconv",
    "doc-examples/example.docx")
  docx_out <- tempfile(fileext = ".docx")
  file.copy(file, docx_out)
  docx_update(input = docx_out)

  if (require("officer")) {
    doc <- read_docx()
    doc <- body_add_fpar(doc,
        value = fpar(
          run_word_field("DOCPROPERTY \"coco\" \\* MERGEFORMAT")))
    doc <- set_doc_properties(doc, coco = "test")

    docx_out <- tempfile(fileext = ".docx")
```

```
    file <- print(doc, target = docx_out)
    docx_update(docx_out)
  }
}
```

`pptx2pdf` *Convert pptx to pdf*

Description

Convert pptx to pdf directly using "Microsoft PowerPoint". This function will not work if "Microsoft PowerPoint" is not available on your machine.

The calls to "Microsoft PowerPoint" are made differently depending on the operating system. On "Windows", a "PowerShell" script using COM technology is used to control "Microsoft PowerPoint". On macOS, an "AppleScript" script is used to control "Microsoft PowerPoint".

Usage

```
pptx2pdf(input, output = gsub("\\.pptx$", ".pdf", input))
```

Arguments

`input, output`	file input and optional file output (default to input with pdf extension).

Value

the name of the produced pdf (the same value as `output`)

Macos manual authorizations

On macOS the call is happening into a working directory managed with function `working_directory()`.

Manual interventions are necessary to authorize 'PowerPoint' applications to write in a single directory: the working directory. These permissions must be set manually, this is required by the macOS security policy. We think that this is not a problem because it is unlikely that you will use a Mac machine as a server.

You must also click "allow" two times to:

1. allow R to run 'AppleScript' scripts that will control PowerPoint
2. allow PowerPoint to write to the working directory.

This process is a one-time operation.

Examples

```
library(locatexec)
if (exec_available('powerpoint')) {
  file <- system.file(package = "doconv",
    "doc-examples/example.pptx")

  out <- pptx2pdf(input = file,
    output = tempfile(fileext = ".pdf"))

  if (file.exists(out)) {
    message(basename(out), " is existing now.")
  }
}
```

`to_miniature` *Thumbnail of a document*

Description

Convert a file into an image (magick image) where the pages are arranged in rows, each row can contain one to several pages.

The result can be saved as a png file.

Usage

```
to_miniature(
  filename,
  row = NULL,
  width = NULL,
  border_color = "#ccc",
  border_geometry = "2x2",
  dpi = 150,
  fileout = NULL,
  timeout = 120
)
```

Arguments

`filename`	input filename, a 'Microsoft Word' or a 'Microsoft Word' or a 'PDF' document.
`row`	row index for every pages. 0 are to be used to drop the page from the final minature. • `c(1, 1)` is to be used to specify that a 2 pages document is to be displayed in a single row with two columns. • `c(1, 1, 2, 3, 3)` is to be used to specify that a 5 pages document is to be displayed as: first row with pages 1 and 2, second row with page 3, third row with pages 4 and 5.

- `c(1, 1, 0, 2, 2)` is to be used to specify that a 5 pages document is to be displayed as: first row with pages 1 and 2, second row with pages 4 and 5.

`width` width of a single image, recommanded values are:

- 650 for docx files
- 750 for pptx files

`border_color` border color, see `image_border()`.

`border_geometry` border geometry to be added around images, see `image_border()`.

`dpi` resolution (dots per inch) to use for images, see `pdf_convert()`.

`fileout` if not NULL, result is saved in a png file whose filename is defined by this argument.

`timeout` timeout in seconds that libreoffice is allowed to use in order to generate the corresponding pdf file, ignored if 0.

Value

a magick image object as returned by `image_read()`.

Examples

```
library(locatexec)
docx_file <- system.file(
  package = "doconv",
  "doc-examples/example.docx"
)
if(exec_available("word"))
  to_miniature(docx_file)

pptx_file <- system.file(
  package = "doconv",
  "doc-examples/example.pptx"
)
if(exec_available("libreoffice") && check_libreoffice_export())
  to_miniature(pptx_file)
```

---

to_pdf *Convert documents to pdf*

Description

Convert documents to pdf using Libre Office. It supports very well "Microsoft PowerPoint" to PDF. "Microsoft Word" can also be converted but some Word features are not supported such as sections.

Windows users must be warned the program is slow on your platform. Performances are not excellent but fast enough on other platform.

Usage

```
to_pdf(
  input,
  output = gsub("\\.[[:alnum:]]+$", ".pdf", input),
  timeout = 120,
  UserInstallation = NULL
)
```

Arguments

input, output	file input and optional file output. If output file is not provided, the value will be the value of input file with extension "pdf".
timeout	timeout in seconds, ignored if 0.
UserInstallation	use this value to set a non-default user profile path for "LibreOffice". If not provided a temporary dir is created. It makes possibles to use more than a single session of "LibreOffice."

Value

the name of the produced pdf (the same value as output), invisibly.

Ubuntu platforms

On some Ubuntu platforms, 'LibreOffice' require to add in the environment variable LD_LIBRARY_PATH the following path: /usr/lib/libreoffice/program (you should see the message "libreglo.so cannot open shared object file" if it is the case). This can be done with R command Sys.setenv(LD_LIBRARY_PATH = "/usr/lib/libreoffice/program/")

Examples

```
library(locatexec)
if (exec_available("libreoffice") && check_libreoffice_export()) {

  out_pptx <- tempfile(fileext = ".pdf")
  file <- system.file(package = "doconv",
    "doc-examples/example.pptx")

  to_pdf(input = file, output = out_pptx)

  out_docx <- tempfile(fileext = ".pdf")
  file <- system.file(package = "doconv",
    "doc-examples/example.docx")

  to_pdf(input = file, output = out_docx)

}
```

`working_directory` *manage docx2pdf working directory*

Description

Initialize or remove working directory used when docx2pdf create the PDF.

On 'macOS', the operation require writing rights to the directory by the Word or PowerPoint program. Word or PowerPoint program must be authorized to write in the directories, if the authorization does not exist, a manual confirmation window is launched, thus preventing automation.

Fortunately, users only have to do this once. The package implementation use only one directory where results are saved in order to have only one time to click this confirmation.

This directory is managed by R function `R_user_dir()`. Its value can be read with the `working_directory()` function. The directory can be deleted with `rm_working_directory()` and created with `init_working_directory()`. Each call will remove that directory when completed.

As a user, you do not have to use these functions because they are called automatically by the `docx2pdf()` function. They are provided to meet the requirements of CRAN policy:

"[...] packages may store user-specific data, configuration and cache files in their respective user directories [...], provided that by default sizes are kept as small as possible and the contents are actively managed (including removing outdated material)."

Usage

```
working_directory()

rm_working_directory()

init_working_directory()
```

Index