Package 'doconv'

May 19, 2021

Type Package

Title Document Conversion to 'PDF' or 'PNG'

Version 0.1.3

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 possibly 'python' and 'Microsoft Word'. If the latter is available, it can be used to produce PDF documents identical to the originals, otherwise, 'LibreOffice' is used.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.1.1

Imports magick, pdftools, locatexec, rappdirs

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

SystemRequirements LibreOffice, Microsoft Word

NeedsCompilation no

Author David Gohel [aut, cre], ArData [cph]

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

Repository CRAN

Date/Publication 2021-05-19 10:00:03 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.

On Windows, this is implemented via win32com while on macOS this is implemented via JXA (Javascript for Automation, aka AppleScript in JS).

This is a simple call to python module 'docx2pdf' into a working directory managed with function `working_directory()`.

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)

See Also

Other tools for docx2pdf: docx2pdf_available(), docx2pdf_install(), docx2pdf_uninstall()

Examples

```
library(locatexec)
if (exec_available('python') && docx2pdf_available()) {
  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.")
  }
}
```

---

docx2pdf_available *Is 'docx2pdf' available*

Description

Checks if 'docx2pdf' is available within a given 'python' distribution.

Usage

```
docx2pdf_available(error = FALSE)
```

Arguments

error	Whether to signal an error if 'docx2pdf' is not found

Value

a single logical value.

See Also

Other tools for docx2pdf: `docx2pdf_install()`, `docx2pdf_uninstall()`, `docx2pdf()`

Examples

```
library(locatexec)

if(exec_available("python") &&
   exec_version("python") > numeric_version("3")) {
  docx2pdf_available()
}
```

`docx2pdf_install` *Install 'docx2pdf'*

Description

Downloads and installs 'docx2pdf' within a given 'python' distribution.

Usage

```
docx2pdf_install(force = FALSE)
```

Arguments

`force`	Whether to force to install (override) 'docx2pdf'.

Value

a single logical value, FALSE if the operation failed, TRUE otherwise.

See Also

Other tools for docx2pdf: `docx2pdf_available()`, `docx2pdf_uninstall()`, `docx2pdf()`

Examples

```
library(locatexec)

if(exec_available("python") &&
   exec_version("python") > numeric_version("3") &&
   !docx2pdf_available()) {

  # this will install and uninstall 'docx2pdf' (i.e. `pip install docx2pdf`)
  # in your python environnement. It can run during 10s and
  # require user to have write permission in python environnement.
  docx2pdf_install()
  docx2pdf_uninstall()

}
```

docx2pdf_uninstall *Uninstall 'docx2pdf'*

Description

Removes 'docx2pdf' within a given 'python' distribution.

Usage

```
docx2pdf_uninstall()
```

Value

a single logical value, FALSE if the operation failed, TRUE otherwise.

See Also

Other tools for docx2pdf: `docx2pdf_available()`, `docx2pdf_install()`, `docx2pdf()`

Examples

```
library(locatexec)

if(exec_available("python") &&
   exec_version("python") > numeric_version("3") &&
   docx2pdf_available()) {

  # this will uninstall and install 'docx2pdf' (i.e. `pip install docx2pdf`)
  # in your python environnement. It can run during 10s and
  # require user to have write permission in python environnement.
  docx2pdf_uninstall()
  docx2pdf_install()

}
```

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",
  fileout = NULL,
  use_docx2pdf = FALSE,
  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()`.
`fileout`	if not NULL, result is saved in a png file whose filename is defined by this argument.
`use_docx2pdf`	if TRUE (and if 'Microsoft Word' executable can be found as well as 'docx2pdf'), docx2pdf will be used to convert 'Word' documents to PDF. This makes it possible to have a PDF identical to the 'Word' display whereas with 'LibreOffice', this is not always the case.
`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("python") && docx2pdf_available())
  to_miniature(docx_file, use_docx2pdf = TRUE)

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),
  use_docx2pdf = FALSE,
  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".
use_docx2pdf	if TRUE (and if 'Microsoft Word' executable can be found as well as 'docx2pdf'), docx2pdf will be used to convert 'Word' documents to PDF. This makes it possible to have a PDF identical to the 'Word' display whereas with 'LibreOffice', this is not always the case.
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.

The operation may require writing rights to the directory by the Word program. On some operating systems (Mac OS), Word (via docx2pdf) must be authorized to write in the directories, if the authorization does not exist, a manual confirmation window is launched, thus preventing automation.

The package chooses to use only one directory in order to have only one time to click this confirmation. This directory is managed by the rappdirs package. 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()`.

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