# Package ‘ariExtra’

July 22, 2020

**Version** 0.2.11

**Title** Tools for Creating Automated Courses

**Description** Leverages the 'ari' package and other tools to create automated courses from slides and a script. Also, uploads these to 'YouTube' and other services using 'tuber' package.

**License** GPL-3

**Imports** xml2, httr, utils, ari (>= 0.2.3), rmarkdown, pdftools, rvest, magrittr, mime, tools, stats, jsonlite, text2speech (>= 0.2.8), docxtractr (>= 0.6.2), tuneR, yaml

**Suggests** knitr, covr, testthat, pagedown, webshot, rstudioapi, base64enc, xaringan

**Encoding** UTF-8

**LazyData** true

**ByteCompile** true

**Type** Package

**VignetteBuilder** knitr

**URL** https://github.com/muschellij2/ariExtra

**BugReports** https://github.com/muschellij2/ariExtra/issues

**RoxygenNote** 7.1.1

**NeedsCompilation** no

**Author** John Muschelli [aut, cre] (<https://orcid.org/0000-0001-6469-1750>)

**Maintainer** John Muschelli <muschellij2@gmail.com>

**Repository** CRAN

**Date/Publication** 2020-07-22 13:50:29 UTC

## R topics documented:

## `ari_document` *Ari Document Class for Rendering*

### Description

Ari Document Class for Rendering

### Usage

```
ari_document(...)
```

### Arguments

| | |
|---|---|
| `...` | arguments to pass to ari::ari_spin |

## `get_slide_id` *Get Slide ID from URL*

### Description

Get Slide ID from URL

### Usage

```
get_slide_id(x)

make_slide_url(x)
```

### Arguments

| | |
|---|---|
| `x` | URL of slide |

### Value

A character vector

### Examples

```
x = paste0("https://docs.google.com/presentation/d/",
"1Tg-GTGnUPduOtZKYuMoelqUNZnUp3vvg_7TtpUPL7e8",
"/edit#slide=id.g154aa4fae2_0_58")
get_slide_id(x)
```

`gs_to_ari` *Convert Google Slides and notes to video with ari*

## Description

Convert Google Slides and notes to video with ari

## Usage

```
gs_to_ari(path, script = NULL, ..., verbose = TRUE)

pptx_to_pdf(path, verbose = TRUE)

pptx_to_pngs(path, verbose = TRUE, dpi = 600)

pptx_to_ari(path, script = NULL, ..., verbose = TRUE)

pdf_to_ari(path, script = NULL, dpi = 300, ..., verbose = TRUE)

html_to_ari(path, script = NULL, ..., verbose = TRUE)

pdf_to_pngs(path, verbose = TRUE, dpi = 600)

images_to_ari(path, script = NULL, dpi = 300, ..., verbose = TRUE)

to_ari(path, script = NULL, ..., verbose = TRUE)
```

## Arguments

| | |
|---|---|
| `path` | Identifier of google slides presentation, or PPTX filename |
| `script` | passed to `make_ari_document()` |
| `...` | Arguments passed to make_ari_document |
| `verbose` | print diagnostic messages |
| `dpi` | resolution (dots per inch) to render images |

## Value

The output from make_ari_document

## Examples

```
# takes > 5 seconds to run
  id = "1Opt6lv7rRi7Kzb9bI0u3SWX1pSz1k7botaphTuFYgNs"
  res = gs_to_ari(id, verbose = 2, open = FALSE)
  if (interactive()) {
    file.edit(res$output_file)
  }
  # replicates same thing as above without verbosity
  res2 = to_ari(id, open = FALSE)


ex_file = system.file("extdata", "example.pptx", package = "ariExtra")
have_soffice = try(docxtractr:::lo_assert())
if (!inherits(have_soffice, "try-error")) {
  pngs = try({
    pptx_to_pngs(ex_file)
  }, silent = TRUE)

  soffice_config_issue = inherits(pngs, "try-error")
  if (soffice_config_issue) {
    warning(
      paste0("soffice does not seem configured properly, may need to ",
             "adapt LD_LIBRARY_PATH, ",
             "try ariExtra:::fix_soffice_library_path()")
    )
    # this can be due to a library issue:
    url = paste0("https://codeyarns.github.io/tech/2019-09-05",
                 "-libregloso-cannot-open-shared-object-file.html")
    if (interactive()) {
      utils::browseURL(url)
    }
  }
  if (!soffice_config_issue) {
    res = pptx_to_ari(ex_file, open = FALSE)
    if (interactive()) {
      file.edit(res$output_file)
    }
    res2 = to_ari(ex_file, open = FALSE)
  }
}
ex_file = system.file("extdata", "example.pdf", package = "ariExtra")
res = pdf_to_ari(ex_file, script = c("hey", "ho"), open = FALSE)
if (interactive()) {
file.edit(res$output_file)
}

res2 = to_ari(ex_file,  script = c("hey", "ho"), open = FALSE)
```

## make_ari_document *Make an ari document*

### Description

Make an ari document

## Usage

```
make_ari_document(
  images,
  script,
  output_file = NULL,
  open = interactive(),
  use_knitr = FALSE,
  ...,
  verbose = TRUE
)

pngs_to_ari(
  images,
  script,
  output_file = NULL,
  open = interactive(),
  use_knitr = FALSE,
  ...,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| `images` | a vector of paths to images. |
| `script` | a file or vector strings that will be spoken |
| `output_file` | a path to the Rmd file which will be created. |
| `open` | should the Rmd be opened after creating? |
| `use_knitr` | use an Rmarkdown type syntax for including the images |
| `...` | additional arguments to pass to ari::ari_spin |
| `verbose` | print diagnostic messages and also passed to ari::ari_stitch |

## Value

A path to the document

## Examples

```
images = system.file("extdata", c("example_1.png", "example_2.png"),
package = "ariExtra")
res = make_ari_document(images, script = c("hello", "how are you"))
res$output_file
res = make_ari_document(images, script = c("hello", "how are you"),
use_knitr = TRUE)
res$output_file

res = to_ari(images, script = c("hello", "how are you"), open = FALSE)
```

## pptx_notes *Get Notes from a PowerPoint (usually from Google Slides)*

### Description

Get Notes from a PowerPoint (usually from Google Slides)

### Usage

```
pptx_notes(file)

pptx_slide_text_df(file)

pptx_slide_note_df(file)

unzip_pptx(file)
```

### Arguments

| | |
|---|---|
| `file` | Character. Path for `PPTX` file |

### Value

Either a character vector or `NULL`

### Examples

```
file = ex_file = system.file("extdata", "example.pptx",
package = "ariExtra")
pptx_notes(ex_file)
pptx_slide_note_df(ex_file)
pptx_slide_text_df(ex_file)
```

## rmd_to_ari *Convert a Rmd to an Ari Document*

### Description

Convert a Rmd to an Ari Document

## Usage

```
rmd_to_ari(
  path,
  script = NULL,
  capture_method = c("iterative", "vectorized"),
  capturer = c("chrome_print", "webshot", "decktape"),
  capturer_args = list(),
  ...,
  rendered_file = NULL,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| `path` | path to Rmd file |
| `script` | optional spoken script, otherwise taken from the HTML comments |
| `capture_method` | Either `"vectorized"` or `"iterative"`. The vectorized mode is faster though it can cause screens to repeat. If making a video from an rmarkdown::ioslides_presentation you should use `"iterative"`. |
| `capturer` | Methods for capturing the HTML slides |
| `capturer_args` | a list of arguments to pass to webshot::webshot or pagedown::chrome_print |
| `...` | additional arguments to pass to make_ari_document |
| `rendered_file` | the HTML output already from render |
| `verbose` | print diagnostic messages |

## Value

The output of make_ari_document

## Examples

```
if (rmarkdown::pandoc_available("1.12.3")) {
  path = system.file("extdata", "example.Rmd", package = "ariExtra")
  tfile = tempfile(fileext = ".pdf")
  out = try({
    output_file = tempfile(fileext = ".html")
    rmarkdown::render(path, output_file = output_file)
    pagedown::chrome_print(output_file,
                           output = tfile)
  }, silent = TRUE)
  if (!inherits(out, "try-error")) {
    res = rmd_to_ari(path, open = FALSE)
    res$output_file
  }
}
```

```
# xaringan example
if (requireNamespace("xaringan", quietly = TRUE)) {
  path  = system.file("examples", "lucy-demo.Rmd", package = "xaringan")

  # get rid of ggplot2 dependency
  x = readLines(path)
  x = gsub("library\\(ggplot2\\)", "", x)
  x = gsub("^\\s*ggplot.*", "", x)
  x = gsub("^\\s*geom_bar.*", "barplot(table(mtcars$am))", x)
  path = tempfile(fileext = ".Rmd")
  writeLines(x, path)
  rendered_file = tempfile(fileext = ".html")

  required_pandoc <- "1.12.3"
  have_pandoc_version = rmarkdown::pandoc_available(required_pandoc)
  if (have_pandoc_version) {
    rmarkdown::render(path,
                      output_format = xaringan::moon_reader(),
                      output_file = rendered_file)
  } else {
    rendered_file = system.file("extdata",
                                "lucy-demo-noggplot2.html",
                                package = "ariExtra")
  }


  script = c("this", "is", "one", "word", "per slide")

  have_decktape = nzchar(Sys.which("decktape"))
  if (have_decktape) {
    pdf_file = tempfile(fileext = ".pdf")
    xaringan::decktape(rendered_file, pdf_file, docker = FALSE)
    res = pdf_to_ari(pdf_file, script = script, open = FALSE)
    result = rmd_to_ari(path = path,
               script = script,
               rendered_file = rendered_file,
               capturer = "decktape")
  }
}
```

## xml_notes    *Get Notes from XML*

### Description

Get Notes from XML

## Usage

```
xml_notes(file, collapse_text = TRUE)
```

## Arguments

| | |
|---|---|
| `file` | XML file from a PPTX |
| `collapse_text` | should text be collapsed by spaces? |

## Value

A character vector

# Index