Package: ari (via r-universe)

June 4, 2024

Type Package

Title Automated R Instructor

Version 0.4.1

Description Create videos from 'R Markdown' documents, or images and audio files. These images can come from image files or HTML slides, and the audio files can be provided by the user or computer voice narration can be created using 'Amazon Polly'. The purpose of this package is to allow users to create accessible, translatable, and reproducible lecture videos. See <https://aws.amazon.com/polly/> for more information.

License MIT + file LICENSE

URL http://github.com/seankross/ari

BugReports http://github.com/seankross/ari/issues

Depends R (>= 3.1.0)

Imports cli, hms, progress, purrr, rmarkdown, rvest, text2speech (>= 0.3.0), tools, tuneR, webshot, xml2

Suggests aws.polly, aws.signature, grDevices, knitr, testthat, xaringan

VignetteBuilder knitr

Remotes cloudyr/aws.polly, cloudyr/aws.signature, jhudsl/text2speech

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

SystemRequirements ffmpeg (>= 3.2.4)

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

RemoteUrl https://github.com/jhudsl/ari

RemoteRef HEAD

RemoteSha 464a33927e9a42862f5584ddd5424ac20f0ae2cb

Contents

`ari_burn_subtitles` *Burn Subtitles into a video*

Description

Burn Subtitles into a video

Usage

```
ari_burn_subtitles(video, srt, verbose = FALSE)
```

Arguments

`video`	Video in `mp4` format
`srt`	Subtitle file in `srt` format
`verbose`	print diagnostic messages. If > 1, then more are printed

Value

Name of output video

Note

This needs `ffmpeg` that was compiled with `--enable-libass` as per https://trac.ffmpeg.org/wiki/HowToBurnSubtitlesIntoVideo

ari_example *Get the path to an ari example file*

Description

This function allows you to quickly access files that are used in the ari documentation.

Usage

```
ari_example(path = NULL)
```

Arguments

path	The name of the file. If no argument is provided then all of the example files will be listed.

Value

A character string

Examples

```
ari_example("ari_intro.Rmd")
```

ari_narrate *Create a video from slides and a script*

Description

`ari_narrate` creates a video from a script written in markdown and HTML slides created with `rmarkdown` or a similar package. This function uses Amazon Polly via `ari_spin`.

Usage

```
ari_narrate(
  script,
  slides,
  output = tempfile(fileext = ".mp4"),
  voice = text2speech::tts_default_voice(service = service),
  service = "amazon",
  capture_method = c("vectorized", "iterative"),
  subtitles = FALSE,
  ...,
  verbose = FALSE,
  audio_codec = get_audio_codec(),
  video_codec = get_video_codec(),
  cleanup = TRUE
)
```

Arguments

script	Either a markdown file where every paragraph will be read over a corresponding slide, or an .Rmd file where each HTML comment will be used for narration.
slides	A path or URL for an HTML slideshow created with rmarkdown, xaringan, or a similar package.
output	The path to the video file which will be created.
voice	The voice you want to use. See tts_voices for more information about what voices are available.
service	speech synthesis service to use, passed to tts. Either "amazon" or "google".
capture_method	Either "vectorized" or "iterative". The vectorized mode is faster though it can cause screens to repeat. If making a video from an ioslides_presentation you should use "iterative".
subtitles	Should a .srt file be created with subtitles? The default value is FALSE. If TRUE then a file with the same name as the output argument will be created, but with the file extension .srt.
...	Arguments that will be passed to webshot.
verbose	print diagnostic messages. If > 1, then more are printed
audio_codec	The audio encoder for the splicing. If this fails, try copy.
video_codec	The video encoder for the splicing. If this fails, see ffmpeg -codecs
cleanup	If TRUE, interim files are deleted

Value

The output from ari_spin

Examples

```
## Not run:

#
ari_narrate(system.file("test", "ari_intro_script.md", package = "ari"),
  system.file("test", "ari_intro.html", package = "ari"),
  voice = "Joey"
)

## End(Not run)
```

---

ari_spin *Create a video from images and text*

Description

Given equal length vectors of paths to images (preferably .jpgs or .pngs) and strings which will be synthesized by Amazon Polly or any other synthesizer available in tts, this function creates an .mp4 video file where each image is shown with its corresponding narration. This function uses ari_stitch to create the video.

Usage

```
ari_spin(
  images,
  paragraphs,
  output = tempfile(fileext = ".mp4"),
  voice = text2speech::tts_default_voice(service = service),
  service = ifelse(have_polly(), "amazon", "google"),
  subtitles = FALSE,
  duration = NULL,
  tts_args = NULL,
  key_or_json_file = NULL,
  ...
)

have_polly()
```

Arguments

images	A vector of paths to images.
paragraphs	A vector strings that will be spoken by Amazon Polly.
output	A path to the video file which will be created.
voice	The voice you want to use. See tts_voices for more information about what voices are available.
service	speech synthesis service to use, passed to tts, Either "amazon", "microsoft", or "google".
subtitles	Should a .srt file be created with subtitles? The default value is FALSE. If TRUE then a file with the same name as the output argument will be created, but with the file extension .srt.
duration	a vector of numeric durations for each audio track. See pad_wav
tts_args	list of arguments to pass to tts
key_or_json_file	access key or JSON file to pass to tts_auth for authorization
...	additional arguments to ari_stitch

Details

This function needs to connect to Amazon Web Services in order to create the narration. You can find a guide for accessing AWS from R here. For more information about how R connects to Amazon Polly see the aws.polly documentation here.

Value

The output from ari_stitch

Examples

```
## Not run:

slides <- system.file("test", c("mab2.png", "mab1.png"),
  package = "ari"
)
sentences <- c(
  "Welcome to my very interesting lecture.",
  "Here are some fantastic equations I came up with."
)
ari_spin(slides, sentences, voice = "Joey")

## End(Not run)
```

`ari_stitch` *Create a video from images and audio*

Description

Given a vector of paths to images (preferably `.jpg`s or `.png`s) and a flat list of `Wave`s of equal length this function will create an `.mp4` video file where each image is shown with its corresponding audio. Take a look at the `readWave` function if you want to import your audio files into R. Please be sure that all images have the same dimensions.

Usage

```
ari_stitch(
  images,
  audio,
  output = tempfile(fileext = ".mp4"),
  verbose = FALSE,
  cleanup = TRUE,
  ffmpeg_opts = "",
  divisible_height = TRUE,
  audio_codec = get_audio_codec(),
  video_codec = get_video_codec(),
  video_sync_method = "2",
  audio_bitrate = NULL,
  video_bitrate = NULL,
  pixel_format = "yuv420p",
  fast_start = FALSE,
  deinterlace = FALSE,
  stereo_audio = TRUE,
  duration = NULL,
  video_filters = NULL,
  frames_per_second = NULL,
  check_inputs = TRUE
)
```

Arguments

images	A vector of paths to images.
audio	A list of Waves from tuneR.
output	A path to the video file which will be created.
verbose	print diagnostic messages. If > 1, then more are printed
cleanup	If TRUE, interim files are deleted
ffmpeg_opts	additional options to send to ffmpeg. This is an advanced option, use at your own risk
divisible_height	Make height divisible by 2, which may be required if getting "height not divisible by 2" error.
audio_codec	The audio encoder for the splicing. If this fails, try copy.
video_codec	The video encoder for the splicing. If this fails, see ffmpeg -codecs
video_sync_method	Video sync method. Should be "auto" or '"vfr"' or a numeric. See https://ffmpeg.org/ffmpeg.html.
audio_bitrate	Bit rate for audio. Passed to -b:a.
video_bitrate	Bit rate for video. Passed to -b:v.
pixel_format	pixel format to encode for 'ffmpeg'.
fast_start	Adding 'faststart' flags for YouTube and other sites, see https://trac.ffmpeg.org/wiki/Encode/YouTube
deinterlace	should the video be de-interlaced, see https://ffmpeg.org/ffmpeg-filters.html, generally for YouTube
stereo_audio	should the audio be forced to stereo, corresponds to '-ac 2'
duration	a vector of numeric durations for each audio track. See pad_wav
video_filters	any options that are passed to -vf arguments for ffmpeg
frames_per_second	frames per second of the video, should be an integer
check_inputs	Should the inputs be checked? Almost always should be TRUE, but may be useful if trying to do customized stuff.

Details

This function uses FFmpeg which you should be sure is installed before using this function. If running Sys.which("ffmpeg") in your R console returns an empty string after installing FFmpeg then you should set the path to FFmpeg on you computer to an environmental variable using Sys.setenv(ffmpeg = "path/to/ffmpeg"). The environmental variable will always override the result of Sys.which("ffmpeg").

Value

A logical value, with the attribute outfile for the output file.

Examples

```
## Not run:
if (ffmpeg_version_sufficient()) {
  result <- ari_stitch(
    ari_example(c("mab1.png", "mab2.png")),
    list(tuneR::noise(), tuneR::noise())
  )
  result <- ari_stitch(
    ari_example(c("mab1.png", "mab2.png")),
    list(tuneR::noise(), tuneR::noise()),
    ffmpeg_opts = "-qscale 0",
    verbose = 2
  )
  # system2("open", attributes(result)$outfile)
}

## End(Not run)
```

`ari_talk` *Create spoken audio files*

Description

A simple function for demoing how spoken text will sound.

Usage

```
ari_talk(
  paragraphs,
  output = tempfile(fileext = ".wav"),
  voice = text2speech::tts_default_voice(service = service),
  service = "amazon"
)
```

Arguments

`paragraphs`	A vector strings that will be spoken by Amazon Polly.
`output`	A path to the audio file which will be created.
`voice`	The voice you want to use. See `tts_voices` for more information about what voices are available.
`service`	speech synthesis service to use, passed to `tts` Either `"amazon"` or `"google"`.

Value

A `Wave` output object, with the attribute `outfile` of the output file name.

ffmpeg_codecs *Get Codecs for ffmpeg*

Description

Get Codecs for ffmpeg

Usage

```
ffmpeg_codecs()

ffmpeg_video_codecs()

ffmpeg_audio_codecs()

ffmpeg_muxers()

ffmpeg_version()

ffmpeg_version_sufficient()

check_ffmpeg_version()
```

Value

A 'data.frame' of codec names and capabilities

Examples

```
## Not run:
if (ffmpeg_version_sufficient()) {
  ffmpeg_codecs()
  ffmpeg_video_codecs()
  ffmpeg_audio_codecs()
}

## End(Not run)
```

ffmpeg_convert *Convert Files using FFMPEG*

Description

Convert Files using FFMPEG

Usage

```
ffmpeg_convert(
  file,
  outfile = tempfile(fileext = paste0(".", tools::file_ext(file))),
  overwrite = TRUE,
  args = NULL
)
```

Arguments

file	Video/PNG file to convert
outfile	output file
overwrite	should output file be overwritten?
args	arguments to pass to system2 to pass to ffmpeg

Value

A character string of the output file with different attributes

Examples

```
pngfile <- tempfile(fileext = ".png")
png(pngfile)
plot(0, 0)
dev.off()
if (have_ffmpeg_exec()) {
  res <- ffmpeg_convert(pngfile)
}
```

---

ffmpeg_error_log *Check error output from individual video*

Description

Check error output from individual video

Usage

```
ffmpeg_error_log(file, verbose = TRUE)
```

Arguments

file	path to video
verbose	print diagnostic messages

Value

The output of the error log

ffmpeg_exec *Get Path to ffmpeg Executable*

Description

Get Path to ffmpeg Executable

Usage

```
ffmpeg_exec(quote = FALSE)

have_ffmpeg_exec()
```

Arguments

quote	should shQuote be run before returning?

Value

The path to the ffmpeg executable, or an error.

Note

This looks using ‘Sys.getenv("ffmpeg")‘ and ‘Sys.which("ffmpeg")‘ to find ‘ffmpeg‘. If ‘ffmpeg‘ is not in your PATH, then please set the path to ‘ffmpeg‘ using ‘Sys.setenv(ffmpeg = "/path/to/ffmpeg")‘

Examples

```
## Not run:
if (have_ffmpeg_exec()) {
  ffmpeg_exec()
}

## End(Not run)
```

pad_wav *Pad Wave Objects*

Description

Pad Wave Objects

Usage

```
pad_wav(wav, duration = NULL)
```

Arguments

wav	list of Wave objects
duration	If NULL, the duration will simply round the Wave up to the next whole integer. If not, these are the duration to pad the Wave *to*. For example 12 means the output Wave will have a length of 12 seconds. Pass NA to those Waves that you want simple rounding.

Value

A list of Wave objects, same length as input wav

Examples

```
wavs <- list(
  tuneR::noise(duration = 1.85 * 44100),
  tuneR::noise()
)
out <- pad_wav(wavs)
dur <- sapply(out, function(x) length(x@left) / x@samp.rate)
duration <- c(2, 2)
out <- pad_wav(wavs, duration = duration)
dur <- sapply(out, function(x) length(x@left) / x@samp.rate)
stopifnot(all(dur == duration))
duration <- c(2, 2.5)
out <- pad_wav(wavs, duration = duration)
dur <- sapply(out, function(x) length(x@left) / x@samp.rate)
stopifnot(isTRUE(all.equal(dur, duration)))
```

---

set_audio_codec *Set Default Audio and Video Codecs*

Description

Set Default Audio and Video Codecs

Usage

```
set_audio_codec(codec)

set_video_codec(codec = "libx264")

get_audio_codec()

get_video_codec()

audio_codec_encode(codec)

video_codec_encode(codec)
```

Arguments

codec	The codec to use or get for audio/video. Uses the ‘ffmpeg_audio_codec‘ and ‘ffmpeg_video_codec‘ options to store this information.

Value

A ‘NULL‘ output

See Also

[ffmpeg_codecs()] for options

Examples

```
## Not run:
if (have_ffmpeg_exec()) {
  print(ffmpeg_version())
  get_audio_codec()
  set_audio_codec(codec = "libfdk_aac")
  get_audio_codec()
  set_audio_codec(codec = "aac")
  get_audio_codec()
}
if (have_ffmpeg_exec()) {
  get_video_codec()
  set_video_codec(codec = "libx265")
  get_video_codec()
  set_video_codec(codec = "libx264")
  get_video_codec()
}
## empty thing
if (have_ffmpeg_exec()) {
  video_codec_encode("libx264")

  audio_codec_encode("aac")
}

## End(Not run)
```

Index