The Automated R Instructor

by Sean Kross, John Muschelli, Jeffrey T. Leek

Abstract We present the `ari` package for video generation of teaching materials. The goal of the package is to be able to generate reproducible videos, with the ability to change and update videos seamlessly. We present an example of generating videos with RMarkdown slide decks with inline comments as the spoken script along with examples using PowerPoint slides or simple images. We also discusss how these videos can be translated into a number of languages from multiple input formats.

Introduction

Videos are a crucial way people learn and pervasive in online education platforms. Creating videos of a speaker with slides take time, energy, and usually video editing skills. A large issue with such videos is that updating the materials either requires remaking the entire video or extensive editing and splicing of new segments. We present **ari**, the automated R instructor to mitigate these issues by creating reproducible presentations and videos that can be automatically generated. By using **ari**, we provide a tools for users to rapidly create and update video content.

The premise of the **ari** package is that you have visual content (e.g. slides, figures) and you want to explain them with words (i.e. a script) in a video. Voice synthesizer services are available from Google, Microsoft, and Amazon. Many of these synthesizers take make use of deep learning methods, such as WaveNet (Van Den Oord et al. 2016) and have interfaces in R (Edmondson 2019; Muschelli 2019a; Leeper 2017). Currently in **ari**, synthesis of the the audio can be rendered using any of these services through the **text2speech** (Muschelli 2019b). The default is Amazon Polly, which has text to speech voice generation in over 21 languages, including a total of 29 dialects, implemented in the **aws.polly** package (Leeper 2017). In addition to multiple languages, the speech generation services provide voices of different genders within the same language. We present the **ari** package with reproducible use case examples and the video outputs with different voices in multiple languages.

The **ari** package relies on the **tuneR** package for reading and manipulating audio output to combine split audio files and to add pauses to audio files between slides (Ligges et al. 2018). Once the audio is generated, it much be spliced with the images to make the video. Multiple open source tools for video editing and splicing exist. The `ffmpeg` (`http://www.ffmpeg.org/`) software is highly powerful, has been thoroughly tested, and has been developed for almost 20 years; **ari** uses `ffmpeg` to overlay the images over the audio. The output videos have been tested on multiple platforms, including the YouTube and Coursera players. A default specification is used in **ari**, such as bitrate, audio and video codecs used, and output video format. GThe numerous additional video specifications can be applied to command-line arguments `ffmpeg` through **ari**.

With these tools together, we can generate automated videos; we have used it for educational videos. The spoken scripts for these videos can be stored in plain text, and therefore be version controlled, edited, and updated easily. If the figures are created in a reproducible framework, such as generated using R code, the entire video can be reproducibly created and automatically updated. Thus, **ari** is the Automated R Instructor. We will provide examples of creating videos based on a slide deck in RMarkdown, a set of images and a script, and discuss how to create slides using a Google Slide deck or PowerPoint presentation.

Making videos with `ari`

The main workhorse of **ari** is the `ari_stitch` function. In this simple example, we assume we already have audio to overlay on some images. The `ari_stitch` function takes the audio and images, and "stitches" them together using `ffmpeg`. In order to use **ari**, one must have an `ffmpeg` installation to combine the audio and images, which can be obtained at `https://ffmpeg.org/`. In the example below, 2 images are overlaid withe white noise for demonstration. This example also allows users to check if the output of `ffmpeg` works with a desired video player.

```
library(tuneR)
library(ari)
result = ari_stitch(
  ari_example(c("mab1.png", "mab2.png")),
  list(noise(), noise()),
  output = "noise.mp4")
isTRUE(result)
```

```
#> [1] TRUE

outfile = attributes(result)$outfile
```

The output is a logical indicator of success and the path of the output file. The video for this output can be seen at https://youtu.be/3kgaYf-EV90.

Synthesizer authentication

In most cases, however, we do not have audio to overlay on images, but must generate it. Though one can generate the spoken audio in many ways, such as fitting a custom deep learning model, we will use the aforementioned services (e.g. Google) as they have direct APIs for use. The downside of using such services is that users must go through steps to provide authentication, whereas most of these APIs and the associated R packages do not allow for interactive authentication such as OAuth.

The **text2speech** package provides a unified interface to these 3 services, and we will focus on Amazon Polly and its authentication requirements. Polly is authenticated using the **aws.signature** package (Leeper 2019). The **aws.signature** documentation provides options and steps to create the relevant credentials; we have also provided an additional tutorial. Essentially, the user must sign up for the service and retrieve public and private API keys and put them into their R profile or other areas accesssible to R. Running `text2speech::tts_auth(service = "amazon")` will indicate if authentication was successful (if using a different service, change the `service` argument). NB: The APIs are generally paid services, but many have free tiers or limits, such as Amazon Polly's free tier for the first year (https://aws.amazon.com/polly/pricing/).

Making videos with `ari`

After the steps above, videos can be using the `ari_spin` function with a set of images and of text. This text is the "script" that is spoken over the images to create the output video. The number of elements in the text need to be equal to the number of images. Let us take the Mercutio's speech from Shakespeare's Romeo and Juliet (Shakespeare 2003) and overlay it on the same images:

```
speech =  c(
  "I will now perform part of Mercutio's speech from Shakespeare's Romeo and Juliet.",
  "O, then, I see Queen Mab hath been with you.
She is the fairies' midwife, and she comes
In shape no bigger than an agate-stone
On the fore-finger of an alderman,
Drawn with a team of little atomies
Athwart men's noses as they lies asleep;")

shakespeare_result = ari_spin(
  ari_example(c("mab1.png", "mab2.png")),
  speech, output = "romeo.mp4", voice = "Joanna")
isTRUE(shakespeare_result)
```

The speech output can be seen at https://youtu.be/ZCClmUv95iY. We chose the voice "Joanna" to the the female US-English speaker for the script. The voices are language-dependent; we can see the available voices for English for Amazon Polly below (from https://docs.aws.amazon.com/polly/latest/dg/SupportedLanguage.html):

```
#>       voice           language language_code gender service
#> 1   Russell Australian English         en-AU   Male  amazon
#> 2    Nicole Australian English         en-AU Female  amazon
#> 3      Emma    British English         en-GB Female  amazon
#> 4     Brian    British English         en-GB   Male  amazon
#> 5       Amy    British English         en-GB Female  amazon
#> 6   Raveena     Indian English         en-IN Female  amazon
#> 7     Aditi     Indian English         en-IN Female  amazon
#> 8     Salli         US English         en-US Female  amazon
#> 9   Matthew         US English         en-US   Male  amazon
#> 10 Kimberly         US English         en-US Female  amazon
#> 11   Kendra         US English         en-US Female  amazon
#> 12   Justin         US English         en-US   Male  amazon
#> 13     Joey         US English         en-US   Male  amazon
```

```
#> 14   Joanna         US English         en-US Female  amazon
#> 15      Ivy         US English         en-US Female  amazon
#> 16  Geraint      Welsh English     en-GB-WLS   Male  amazon
```

Though the voice generation is relatively clear, we would not classify the speech as passionate or with a high level of emphasis. Thus, be believe these videos may be best used for conveying information or education. We can also generate the video using the voice `Brian`, which is an British English male voice:

```
gb_result = ari_spin(
  ari_example(c("mab1.png", "mab2.png")),
  speech, output = "romeo_gb.mp4", voice = "Brian")
isTRUE(gb_result)
```

The speech output can be seen at https://youtu.be/VuaeRKvs-Y4. The output video format is MP4 by default, but can be any format (aka "muxers") that the `ffmpeg` installation support, see `ari::ffmpeg_muxers`. Supported codecs can be founded using the functions `ffmpeg_audio_codecs` and `ffmpeg_video_codecs`. The images and script can be presented in a number of ways, such as a text file and a series of PNG images. More likely, the images and script will be bundled together, such as a Google Slide deck/Powerpoint presentation with the script in the notes section, or an HTML slide presentation based in RMarkdown, where the script is in the HTML comments.

#stopped here

For most R users, we believe the most natural setting is that the user has a slide deck using RMarkdown, for example using the **rmarkdown** or **xaringan** packages (Allaire et al. 2019; Xie, Allaire, and Grolemund 2018; Xie 2018). The HTML slides are rendered using **webshot** (Chang 2018) and the script is located in HTML comments (i.e. between `<!--` and `-->`). This setup allows for one plain text, version-controllable, integrated document that can reproducibly generate a video.

Some HTML slides take a bit to render on **webshot**; for example may be rendered dark gray instead of white. If you change the `delay` argument in `ari_narrate`, passed to **webshot**, this can resolve some issues by allowing the page to fully render, but may take a bit longer to run. Also, the argument `capture_method` allows for the control on how `webshot` is run. Using the value `vectorized`, webshot is run on the entire slide deck and is faster, but may have some issues. The value `iterative` runs `webshot` for each slide separately, which can be more robust, but can be slower.

Users can pass in both the RMarkdown document and the resulting output, or simply the document, and the output will be created using `render` from **rmarkdown** (Allaire et al. 2019).

```
# Create a video from an R Markdown file with comments and slides
res = ari_narrate(
  script = ari_example("ari_comments.Rmd"),
  voice = "Kendra",
  capture_method = "iterative")
```

Accessibility

With respect to accessibility, as **ari** has the direct script that was spoken, this provides for direct subtitles for those hard of hearing rather than relying on other services, such as YouTube, to provide a speech to text translation. Though some changes to the script are required for AMazon Polly to correctly pronounce the information, these can be changed using regular expressions in the script, and then passed to `ari_subtitles`.

Technical stuff

The **ari** package relies on FFmpeg (>= 3.2.4) to interleave the images and the audio files.

Future directions

We believe the heavy reliance on an `ffmpeg` installation can be mitigated in the future with advances in the **av** package. Though the **av** package has powerful functionality and is currently porting more from `libav` and therefore `ffmpeg`, it currently does not have the capabilities requried for **ari**. Although third party installation from https://ffmpeg.org/ can be burdensome to a user, package managers such as `brew` for OSX and `choco` for Windows provide installations.

Although we rely on Amazon Polly for voice synthesis, other packages provide voice synthesis, such as **mscstts** for Microsoft and **googleLanguageR** for Google. We aim to harmonize these synthesis options, so that users can choose to create videos with the services that they support or have access to.

Scripts can be automatically translated into other languages with services like the Google Translation API, which **googleLanguageR** provides an interface. Amazon Polly can speak languages other than English. This means you can write a lecture once and generate slides and videos in multiple languages.

We have created a Docker environment (https://github.com/seankross/bologna) with the requirements to create videos using **ari**. This Docker image increases the level of reproducibility and can be used to create standalone disk images to create content.

Examples (FROM README - edit)

These examples make use of the `ari_example()` function. In order to view the files mentioned here you should use `file.show(ari_example("[file name]"))`. You can watch an example of a video produced by Ari here.

```
library(ari)

# First set up your AWS keys
Sys.setenv("AWS_ACCESS_KEY_ID" = "EA6TDV7ASDE9TL2WI6RJ",
           "AWS_SECRET_ACCESS_KEY" = "OSnwITbMzcAwvHfYDEmk10khb3g82j04Wj8Va4AA",
           "AWS_DEFAULT_REGION" = "us-east-2")

# Create a video from a Markdown file and slides
ari_narrate(
  ari_example("ari_intro_script.md"),
  ari_example("ari_intro.html"),
  voice = "Joey")

# Create a video from an R Markdown file with comments and slides
ari_narrate(
  ari_example("ari_comments.Rmd"),
  ari_example("ari_intro.html"),
  voice = "Kendra")

# Create a video from images and strings
ari_spin(
  ari_example(c("mab1.png", "mab2.png")),
  c("This is a graph.", "This is another graph"),
  voice = "Joanna")
```

RMarkdown/HTML slide Problems

```
ari_narrate(
  ari_example("ari_comments.Rmd"),
  ari_example("ari_intro.html"),
  voice = "Kendra",
  delay = 0.5,
  capture_method = "iterative")
```

Allaire, JJ, Yihui Xie, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, Winston Chang, and Richard Iannone. 2019. *rmarkdown: Dynamic Documents for R*. https://rmarkdown.rstudio.com.

Chang, Winston. 2018. *webshot: Take Screenshots of Web Pages*. https://CRAN.R-project.org/package=webshot.

Edmondson, Mark. 2019. *googleLanguageR: Call Google's 'Natural Language' API, 'Cloud Translation' API, 'Cloud Speech' API and 'Cloud Text-to-Speech' API*.

Leeper, Thomas J. 2017. *aws.polly: Client for AWS Polly*.

———. 2019. *aws.signature: Amazon Web Services Request Signatures*.

Ligges, Uwe, Sebastian Krey, Olaf Mersmann, and Sarah Schnackenberg. 2018. *tuneR: Analysis of Music and Speech*. https://CRAN.R-project.org/package=tuneR.

Muschelli, John. 2019a. *mscstts: R Client for the Microsoft Cognitive Services 'Text-to-Speech' REST API*. https://CRAN.R-project.org/package=mscstts.

———. 2019b. *text2speech: Text to Speech*. https://github.com/muschellij2/text2speech.

Shakespeare, William. 2003. *Romeo and Juliet*. Cambridge University Press.

Van Den Oord, Aäron, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W Senior, and Koray Kavukcuoglu. 2016. "WaveNet: A Generative Model for Raw Audio." *SSW* 125.

Xie, Yihui. 2018. *xaringan: Presentation Ninja*. https://CRAN.R-project.org/package=xaringan.

Xie, Yihui, J.J. Allaire, and Garrett Grolemund. 2018. *R Markdown: The Definitive Guide*. Boca Raton, Florida: Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown.

Sean Kross
Cognitive Science, University of California, San Diego
9500 Gilman Dr.
La Jolla, CA 92093
author1@work

John Muschelli
Department of Biostatistics, Johns Hopkins Bloomberg School of Public Health
615 N Wolfe Street
Baltimore, MD 21231
jmusche1@jhu.edu

Jeffrey T. Leek
Department of Biostatistics, Johns Hopkins Bloomberg School of Public Health
615 N Wolfe Street
Baltimore, MD 21231
jtleek@jhu.edu