# The Automated R Instructor

*by Sean Kross, John Muschelli, Jeffrey T. Leek*

**Abstract** We present the `ari` package for video generation with audio overlaid. The goal of the package is to be able to generate reproducible videos, likely with the goal of education. We present an example of generating videos with RMarkdown slide decks with inline comments as the spoken script.

## Introduction

Videos are a crucial way people learn. Creating videos of an speaker with slides take time, energy, and usually a bit if 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 hope to be able 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. Synthesizing the audio from the script is done by Amazon Polly, which is a text to speech voice generation engine with over 21 languages, including a total of 29 dialects, implemented in the **aws.polly** package (Leeper, 2017). In addition to multiple languages, Amazon Polly provides voices of different genders within the same language. Amazon Polly API has a 1500 character limit when synthesizing audio; **ari** splices scripts into sufficiently-small chunks and creates the audio. 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, Krey, Mersmann, and Schnackenberg, 2018).

Once the audio is generated, it much be spliced with the images. 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. We have tested the videos on YouTube and the Coursera platform viewers. A default specification is used in **ari**, such as bitrate, audio and video codecs used, and output video format. Additional video specifications can be applied to command-line arguments `ffmpeg` through **ari**.

With these tools together, we can generate educational videos. The scripts for these videos can be stored in plain text, and therefore be version controlled. 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.

## Amazon authentication

In order to use Amazon Polly, one must set up R to use Amazon Web Services, which uses **aws.signature** for authentication (Leeper, 2018). In addition to **aws.signature** documentation, we have provided as a tutorial here. In order to test the authentication required for **ari**, you can run `aws.polly::list_voices()`. The `aws.polly::synthesize` function powers the backend authentication. NB: Amazon Polly is a paid service, but has a free tier for the first year (`https://aws.amazon.com/polly/pricing/`).

## Making videos with `ari`

Videos can be produced form `ari` using a simple set of images and a script, a Google Slide deck with the script in the notes section, or an HTML slide presentation based in RMarkdown, where the script is in the HTML comments.

The main workhorse of **ari** is the `ari_stitch` function. This function takes in a vector of images and a series of Wav audio objects or audio filenames. Though most times a user has the text script and needs to generate the audio narration, this function is useful if the text to speech generation is done using another service, such as Google in the **googleLanguageR** package or Microsoft in the **mscstts** package (Edmondson, 2019; Muschelli, 2019).

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

When the user has a text scrip, the higher-level `ari_spin` functioni is commonly used, which takes in a character vector of images and a character vector of text (`paragraphs` argument). 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. The output video format is MP4 by default, but can be any format (aka "muxers") that the `ffmpeg` installation support. Supported codecs can be founded using the functions `ffmpeg_audio_codecs` and `ffmpeg_video_codecs`.

The HTML slides are rendered using **webshot**.

Some html slides take a bit to render on webshot, and can be dark gray instead of white. If you change the `delay` argument in `ari_narrate`, passed to `webshot`, this can resolve some issues, but may take a bit longer to run. Also, using `capture_method = "vectorized"` is faster, but may have some issues, so run with `capture_method = "iterative"` if this is the case as so:

### 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")
```

*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