Advanced RMarkdown

Mark Edmondson

13 February 2017

Before you begin

RMarkdown has a rich ecosystem that is constantly being added to, so keep an eye on the new formats as they occur on the website. All start with an RMarkdown file.

HTML

The most flexible format is HTML, as it can run JavaScript and therefore HTML widgets, which looks to link R code with JavaScript. Since most of the sexy visualisation libraries are JavaScript this opens up a lot of possibilities, as well as letting users use CSS and all the other website tricks.

Most of the cool RMarkdown stuff is done by Yihui Xie

Starting with HTML Documents

Websites

With a collection of HTML documents, you can make a website, like this one.

In addition to being able to create R visualisations, code and text, coupled with GitHub's free hosting you can also have a live website for use. Like this one.

To create a website, start a project and collect all your `.Rmd` pages with the HTML flag. You then add a metadata file called `_site.yml` which takes care of the navigation bar and other details. You also need to specify an `index.Rmd` to generate the front page.

RStudio gives you extra tools when creating a website in its options, including a `Build` button that will render the entire website for you.

Exercise

Create a website in R:

1. Make a new project
2. Create an index.Rmd file
3. Create a `_site.yml` file
4. Build the website (you may need to close and open the project again to see build tools)

Bookdown

Bookdown is another format that will work with HTML too. Most of the great R resources use it, often as free versions of real R help books.

Bookdown also offers free hosting on the Bookdown website.

Slidedecks - xaringan

There are many slide deck formats for `.Rmd` but our current favourite is `xaringan`

`xaringan` is an example of `Document templates`, which allows you to specify a skeleton document that incudes elements such as your business logo and font. In this case it also helps support an extensive JavaScript library, remark.js

PDF

For professional looking reports offline, the PDF output is hard to beat. You need to obviously not use JavaScript in your code so will need to use `ggplot` or similar, as well as install a weighty `TeX` plugin [3GB]

A PDF version of this page is included with the basic settings so you can take a look.

Notebooks

Notebooks are an R implementation of the popular live code and text format from Python, Jupyter. It allows you to execute code live in the document.

You can save the state of the notebook and it then looks like a normal HTML document from above, but you also get a more dynamic file you can edit as you go, and share with other users.

Notebooks (and RMarkdown) don't need to run just R - Python, SQL and Bash are all supported. This means you can create very dynamic workflows all in one document.

A demo of running Python, R and Tensorflow in an R notebook is available here.

Exercise

1. Create an RMarkdown Notebook
2. Use the examples here to transfer files between an R and Python session
3. Run this code chunk of CSS in your notebook and preview it

```
body {
  color: red;
}
```

RMarkdown options

There are many options that can be passed through to `knitr` which is the engine that turns `.Rmd` code into `.md` which in turn is turned into the specified final format. These options allow you to customise heavily such as figure alignment, etc.

Uploading images

One useful option relevant for us is the ability to run a function over every image the RMarkdown produces. This means we can do things like upload the images to a service and then take that link as the image source instead.

If using `googleCloudStorageR` this means you can also host the images on Google Cloud Storage, limiting those images to just certain users. An example is available here.

Exercise

Add the option `opts_knit$set(upload.fun = imgur_upload)` to your `.Rmd` to host the images online via `imgur`

Emailing RMarkdown reports

The above is useful for emailing the results of `.Rmd` files to users.

E-mail clients are fickle beasts, so embedding images along in a file can have varying results according to if a reciever is reading the email on Outlook, Gmail or MacMail. Hosting the images gives a way to be more consistent over the email clients (if they "allow images" from your server.)

Here is a script that lets you email a file if you are using Mailgun:

```
sendEmail <- function(from_name, email, subject_line, html_file){

  html_message <- readChar(html_file, file.info(html_file)$size)

  url <- "https://api.mailgun.net/v3/sandbox5f2XXXXXXXa.mailgun.org/messages"
  api_key <- "key-XXXXX"
  the_body <-
    list(
      from = from_name,
      to = email,
      subject = subject_line,
      html = html_message
    )

  req <- httr::POST(url,
                    httr::authenticate("api", api_key),
                    encode = "form",
                    body = the_body)

  httr::stop_for_status(req)

  TRUE
}
```

To render the `.Rmd` file via a script, you need the `render()` function:

```
## takes your my_file.Rmd and turns it into HTML called my_email.html
library(rmarkdown)
render("my_file.Rmd", output_file = "my_email.html")
```

You then have all the ingredients to create an automated reporting email:

1. Run an `.Rmd` file that gathers your data, makes plots etc.
2. Set the options so the images are uploaded to a cloud service
3. Render the file via `render()` to a HTML file
4. Send an email with the HTML file