R Markdown workshop

JA, CCO

October 12, 2015

Contents

What is R Markdown?

From rmarkdown.rstudio.com:

> R Markdown is an authoring format that enables easy creation of dynamic documents, presentations, and reports from R. It combines the core syntax of markdown (an easy-to-write plain text format) with embedded R code chunks that are run so their output can be included in the final document. R Markdown documents are fully reproducible (they can be automatically regenerated whenever underlying R code or data changes).

Here's a handy presentation that covers many of the features of Markdown and R Markdown: http://cpsievert.github.io/slides/markdown/

This tutorial/workshop will focus on creating R Markdown documents, rather than presentations, but the ideas are the same.

Quick start!

In RStudio, create a blank .Rmd: go to File -> New File -> R Markdown...

What are the differences between this and a standard new R script?

- the file extension
- the header
- white spaces (where text will be formatted using Markdown codes) and grey spaces (where functional R code lives)

With your new file, hit the 'Knit HTML' button at the top of the script pane in RStudio. You will need to save the file before it will knit.

Check out what it looks like! pretty nice, pretty easy.

How can you use R Markdown?

Document your code

Use R Markdown to make your code pretty and share it with others. R code in the .Rmd function exactly the same as in a regular .R script. A reader can see your code, and what it does, and in fancy cases, interact with the data directly in the document.

- Explain your code or make it available to others
- Show all code with outputs
- Create human-readable coding

Check out examples here!

Document your workflow

Use R Markdown to highlight the workflow of your code, to share your process. The code itself is not so important; instead, you can focus on the steps you used to get from input A to output B.

- Promote reproducibility of results and provenance of data
- Focus on the process: inputs, outputs, and the steps in between
- Downplay specific code except where critical

Document your results

Use R Markdown to document your results such as for publication. The code itself is not important; even the intermediate steps are not all that interesting. Instead, you can focus on your research question and present the results in nicely-formatted text, equations, tables, and graphs.

- Hide the work: if they want to see the process, they can dig into the document.
- Focus on the big picture: formatted writeup and fancy graphs
- Present results in a format that doesn't even look like code - looks more like a publication.

See these example files

- the `rmarkdown_examples.Rmd` file and `rmarkdown_examples.html` file are in this repository.

Markdown style basics (outside of code chunks):

Here's a great R Markdown reference sheet and a similar but more colorful R Markdown cheat sheet, including basic Markdown formatting. No need to recreate all the different formatting options here; but you should get to know and love the following, which are all easy:

- bulleted lists

    - including multi-level bullets and numbered lists

- *italics*, **bold**, and even ***bold italics***
- headers - six levels, from # (HUGE) to ###### (small)
- `inline monospaced text` and inline executable R code
- code blocks, non-executing (just for display) and executing (R code chunks)

Print out this awesome reference sheet or this awesome cheat sheet and nail it to your monitor.

Note on spaces and line breaks

Markdown is finicky, but R Markdown is especially finicky about line breaks and spaces. A single line break is treated about the same as a space. Between sections with different formatting, it's good to always put a blank line (two line breaks). If your knitted formattin looks wrong, you probably forgot to space it properly.

.Rmd documents and code chunks

Document setup: the .Rmd file header

The top of the .Rmd file should have a header; when you create a new .Rmd this is added automatically. At very least, include the dashes, a title, and output format. Note that the left alignment and tab spacing are important.

```
---
title: "R Markdown workshop"
output: html_document
---
```

more options can go in the header; more on these later:

```
---
title: "R Markdown workshop"
author: "CCO, JA"
date: "October 12, 2015"
output:
  html_document:
    toc: true
    theme: spacelab
    highlight: haddock
  pdf_document:
    toc: true
    theme: spacelab
    highlight: haddock
---
```

Inline code

Outside of code blocks, you can include R code directly in the formatted text. This is helpful if you want to display a value or output in a nicely formatted way. This can access values calculated locally or in other code chunks earlier in the document. To do this, start with a single back-tick followed by 'r', and end with another single back-tick. For example:

```
The `r animals = c('armadillo', 'platypus'); animals[2]` is a very unusual animal.
```

Will display as: The platypus is a very unusual animal.

Code chunks

Code chunks allow for larger pieces of executable R code within your document - basically, mini scripts or pieces of a larger script, with regular Markdown text in between. Each code chunk can access values, functions, and packages defined earlier in the .Rmd, but beyond that, each chunk needs to be self-contained (e.g. if you define a function or a loop, the entire code for that must be within a single code chunk).

Each code chunk starts with three back-ticks (“‘) at the start of a line, followed by ‘r’ in braces:

```
"‘ {r}
```

Next comes the code; then the code block ends with a triple backtick on its own line:

```
"‘
```

The `{r}` is the code chunk header, and can be modified in helpful ways.

- The text after the `r` but before a comma becomes the name of the code chunk. The name appears in the list of sections in the RStudio script editing pane, so you can navigate right to the code chunk of your choice.
  - e.g. `"‘ {r chunk name goes here}`
- Code chunk options can be added to the header, to control how the code chunk is executed and knitted. Code chunk options are separated by commas.
  - e.g. `"‘ {r chunk name, echo = FALSE, eval = TRUE, message = FALSE, warning = FALSE}`
- Helpful code chunk options:
  - `echo`: when FALSE, the code in this chunk will not be displayed in the knitted document. Any outputs due to the code will still be displayed (e.g. `x = 10; x` or `cat(sprintf('The value of x is %s\n', 10))`.
  - `eval`: when FALSE, this code chunk will not run. You can use `eval = FALSE, echo = TRUE` to display but not run the code.
  - `message` or `warning`: setting these to false will prevent the display of outputs from `message()` or `warning()` - such as those that often show up (in red) when loading packages with `library()`.
- You can skip the name and go straight to the options, if you like, just use a comma after the `r`:
  - e.g. `"‘ {r, echo = FALSE}`

More code chunk options can be found on the R Markdown reference sheet or R Markdown cheat sheet, or at http://rmarkdown.rstudio.com/authoring_rcodechunks.html. Some of these are also shown later in this document.

Knitting and previewing .Rmd

Once your .Rmd is done, you can try to knit it. At the top of the script editing pane in RStudio, you’ll see an option to “Knit HTML” (or maybe “Knit PDF” or other formats). That kicks in the `knitr` library to interpret your .Rmd code, and save it as a new file in the same directory as your .Rmd, with the same base name, and a new extension.

If your code has errors, you’ll see the error reported in the R Markdown pane in RStudio. Otherwise, you can track the progress of the knitting, and see the resulting document.

Previewing .html on GitHub

RStudio can't display .html, but if you click on the file in the files pane, it gives you an option to open in a web browser.

GitHub also can't display .html. But once you commit and push your .html to GitHub, you can preview it by following these steps:

- Browse to the .html doc, and click it.
  - It probably gives you some message like "(Sorry about that, but we can't show files that are this big right now.)"
- Ignore the message, and in the URL window, do this:
  - replace `github.com` with `rawgit.com`
  - delete `blob` and one of the `/` next to it
  - hit enter to load the modified URL. Done!
- Copy the new URL, and you can send people this as the link to your fancy new document.
  - Assuming you don't change the file name, every time you re-knit, re-commit, and re-push your .Rmd and .html, this link will still be live and point to the most recent version.
- A second way is using `htmlpreview` - https://htmlpreview.github.io/

Common issues in R Markdown

`setwd()`: don't do it! The `knitr` package isn't friendly about setting working directories. It expects that the working directory is the directory in which the .Rmd file lives; if you change that through `setwd()` in one chunk of code, it resets it for the next chunk of code.

It's generally a good practice to avoid using `setwd()` in your code anyway - use variables and `file.path()` to store and set file locations.

Previewing on GitHub with rawgit The rawgit method (see above) of previewing .html on GitHub only works for public repositories, not private repositories. Probably also true of the htmlpreview.github.io method.

Spaces/line breaks As noted above: Knitting an .Rmd properly is very sensitive to line breaks. Whenever you want to change the format from one line to the next, include an extra line break in between. Note that knitting ignores multiple line breaks, so one blank line and ten blank lines will render the same in the final output document.

Printing plots in ggplot2 You have awesome code that plots the most wonderful and informative graph ever beheld by human eyes. But it doesn't show up in your knitted file!

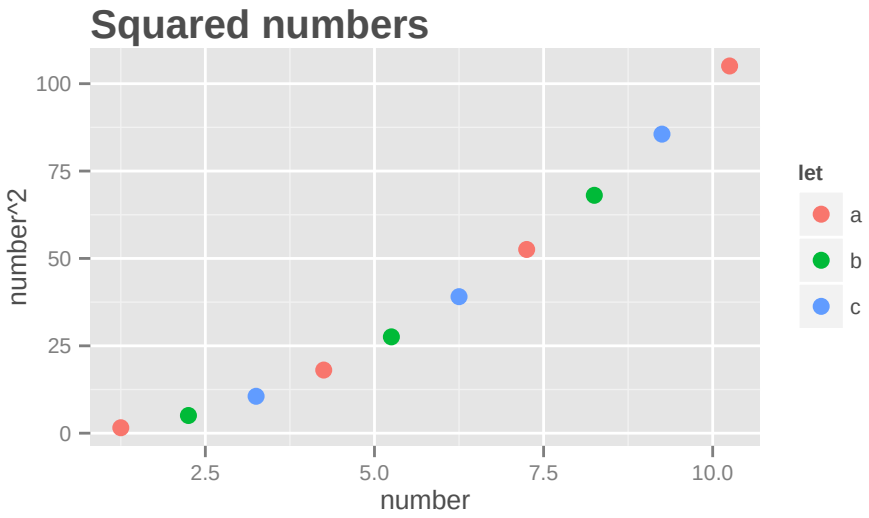
Once you get your ggplot code all worked out, assign it to a named object, and then use `print(object_name)`. Just calling `object_name` works within a regular R script, but not when knitting. In the R code chunk header, you can set `fig.align`, `fig.width`, and `fig.height` if you like.

```
nums <- c(1:10) + 0.25
sq_num <- data.frame('num' = nums, 'num_sq' = nums^2, 'let' = rep(letters[1:3], len = 10))

awesome_plot <- ggplot(data = sq_num, aes(x = num, y = num_sq, color = let)) +
```

```
  theme(text = element_text(family = 'Helvetica', color = 'gray30', size = 10),
        plot.title = element_text(size = rel(1.5), hjust = 0, face = 'bold'),
        legend.position = 'right') +
  geom_point(size = 3) +
  labs(title = 'Squared numbers', x = 'number', y = 'number^2')

print(awesome_plot)
```



Advanced topics

Tables

Tables are actually pretty easy. Use the `kable()` function in `knitr` package to make nicely formatted tables simply.

```
print(sq_num[1:4, ])
```

```
##    num  num_sq let
## 1 1.25  1.5625   a
## 2 2.25  5.0625   b
## 3 3.25 10.5625   c
## 4 4.25 18.0625   a
```

```
knitr::kable(head(sq_num, 4))
```

num	num_sq	let
1.25	1.5625	a
2.25	5.0625	b
3.25	10.5625	c
4.25	18.0625	a

Nice options for tables in `knitr::kable()`:

- `caption`: text string to add a caption at the start of your table.
- `align`: vector of characters indicating alignment of each column: 'l', 'r', or 'c'
- `digits`: vector of integers fixing the number of decimal digits to display.

```
knitr::kable(head(sq_num, 4),
             align = c('l', 'c', 'r'),
             caption = 'Squared numbers',
             digits = c(2, 3, 4))
```

Table 2: Squared numbers

num	num_sq	let
1.25	1.562	a
2.25	5.062	b
3.25	10.562	c
4.25	18.062	a

For other options for formatted tables, see `pander` or `xtable` packages. More on these at: http://kbroman.org/knitr_knutshell/pages/figs_tables.html

Interactive tables! woo! The `DT` package offers a function `datatable()` (not to be confused with the `data.table` package and function...) that makes for easy creation of interactive tables in R Markdown. It essentially embeds JavaScript code in the output HTML that adds the interactivity. Sorting by column is standard; this example adds a filter option too.

```
# install.packages('DT')
library(DT)

nums <- c(1:100) + 0.25
sq_num2 <- data.frame('num' = nums, 'num_sq' = nums^2, 'let' = rep(letters[1:3], len = length(nums)))

datatable(sq_num2,
          caption = 'Squared numbers, interactive',
          rownames = FALSE,
          filter = 'bottom')
```

Appendix 1: stuff to look into later

- caching - globally and by chunk - saving graphical outputs to reduce processing time next time around
- equations using LaTeX - create nicely formatted equations directly in R Markdown
- Themes and Highlights - make your docs look fancy with Bootswatch themes and syntax highlighting style ()
- Knit Options - more control over how your document knits
- knitr::knit_child() - knitting multiple Rmd documents into one (Gavin)

Appendix 2: preparing for the workshop/tutorial

Packages required for R Markdown:

- library(rmarkdown) ### install.packages("rmarkdown")

Optional packages:

- library(DT) ### install.packages('DT') ### for cool interactive data table in R Markdown HTML outputs

Atom: a simple (and free) markdown editor, with an option allowing you to preview your formatted file side-by-side with your Markdown code.

- Atom description: https://github.com/blog/1831-atom-free-and-open-source-for-everyone
- Atom download: https://atom.io/

PDF: To knit pdf documents, you must have one of these installed on your system. They are large files - it'll take a while to download so please download ahead of time.

- **Windows**: MiKTeX (Complete) - http://miktex.org/2.9/setup
  - NOTE: Be sure to download the Complete rather than Basic installation
- **Mac OS X**: TexLive 2013 (Full) - http://tug.org/mactex/
  - NOTE: Download with Safari rather than Chrome *strongly* recommended

Appendix 3: More R references

- https://www.rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf
- https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf
- http://www.markitdown.net/markdown
- http://cpsievert.github.io/slides/markdown/
- citations and bibliographies: http://rmarkdown.rstudio.com/authoring_bibliographies_and_citations.html