R Markdown Demo

PHAR G8012: Statistics for the Basic Sciences

Jared Sampson

2/20/2018

Before We Begin

The instructions here are for working with R Markdown documents within RStudio. If you choose to use another editor or work from the command line you will probably need to make some adjustments to the workflow.

Before creating your first `.Rmd` document, you need to install the R Markdown package into your R environment. This can be done easily by running the command `install.packages("rmarkdown")` at the RStudio console command prompt. This may take a few moments as dependencies are installed.

An R Markdown Exercise

Create a new RStudio project

1. File > New Project… > Existing Directory… > Browse… (to this directory) > Create Project.
2. File > New File > R Markdown…
3. Select 'Document', add Title and select HTML as an output format.

Inspect your .Rmd document

1. Save the document as **demo.Rmd** to the project folder.
2. Press **Knit** (or `Shift+Cmd+K` for Mac / `Shift+Ctrl+K` for Windows) to compile your document. You should see the formatted HTML document appear in a new RStudio window.
3. Look at the source code to see how it defines e.g. `**bold**` text and "fenced" (```) code blocks.
4. Look at the first code block. Click the green arrow to run the code in the block. Note there is no output due to the `include=FALSE` option.
5. Look at the second code block. Run this code block, too, and see how the output gets printed just below it. Try the third code block as well.

6. Note that R code blocks are shown with a grey background and syntax highlighting as in the RStudio source editor, but plain (i.e. non-R) code blocks, which don't have the `{r block_name}` element directly the opening fence, are shown in a fixed-width font but not shaded grey or syntax-highlighted. I'll use these plain code blocks to indicate non-R text for your .Rmd document that is not specifically R code.
7. Delete everything after the first ```` ```{r setup}``` ```` code block (or don't, if you want to keep it for reference), and let's work through a problem.

A "real" problem

You'll find the `bodytempandheartrate.csv` dataset in this `rmarkdown_demo` project folder. We're going to do problem #1a from Problem Set 2: Calculate 95% confidence intervals for body temperature and heart rate for the sample as a whole and for men and women separately.

1. Add a header by typing the following on a line by itself.

   ```
   # Problem 1a.
   ```

   See what happens if you add extra hash/pound/octothorpe symbols—use 1-6 of these these to get nested heading levels 1 through 6.

2. Add a description of the question you are answering under the header. Try making it bold. Where do you have to put the asterisks?

3. Insert an R code block via "Insert > R".

4. Write some code in the code block. Include comments for anything that might be unclear.

```r
# Import the data
bthr <- read.csv("bodytempandheartrate.csv")

# Convenience variables
temp <- bthr$T
hr <- bthr$HR
sex <- bthr$MF  # 1 = Male, 2 = Female
temp_m <- temp[sex == 1]
temp_f <- temp[sex == 2]
hr_m <- hr[sex == 1]
hr_f <- hr[sex == 2]

# Calculate confidence intervals
temp_all_conf <- t.test(temp)$conf.int
temp_m_conf <- t.test(temp_m)$conf.int
temp_f_conf <- t.test(temp_f)$conf.int
hr_all_conf <- t.test(hr)$conf.int
hr_m_conf <- t.test(hr_m)$conf.int
hr_f_conf <- t.test(hr_f)$conf.int
```

5. We can access the values R has calculated for these stored confidence intervals using the notation `` `r hr_all_conf` `` inline in a regular line of text.
6. After the code block, add a paragraph or other text (here, a bulleted list) to answer the question.

```
The confidence intervals for heart rate are:
- All: `r hr_all_conf`
- Men: `r hr_m_conf`
- Women: `r hr_f_conf`
```

7. When you knit your document, this will show up like this:

> The confidence intervals for heart rate are:
>
> - All: 72.5360699, 74.9870071
> - Men: 71.9134314, 74.8250301
> - Women: 72.1454691, 76.1622232

8. That seems like a lot of unnecessary digits. Let's adjust that output option. In the `setup` block at the top of the document, add the following line and re-knit.

```
options(digits = 2)
```

Note that `options()` is not specific to R Markdown, but is a general R function, so you can use it in any R script.

Make a figure

1. Insert a new R code block and give it a name.
2. Add some histograms or boxplot of the temperature data (always a good idea to do this before even calculating any statistics).

```
hist(temp_m)
hist(temp_f)
boxplot(hr_m, hr_f, names = c("Men", "Women"), notch = TRUE)
```

3. That takes up too much space. Combine them into one row by adding the following line before the first `hist` statement.

```
par(mfrow = c(1, 3))  # 'multi-figure by row'
```

4. But now the aspect ratio is too tall. Correct it by adding `fig.width` and `fig.asp` options after the block name, so it looks like `{r some_figs, fig.width=3, fig.asp=0.4}`.
5. What if we want it to be a floating figure (i.e. not move with the text but be anchored to the top or bottom of a page like a journal figure)? Add `fig.cap="Some figure caption."` to the block options as well. Note that this functions differently depending on the output type (HTML, PDF, Word).

Other things you may want to try

1. Use the down arrow next to the Knit button to knit a Word document. Notice the extra output format that appears in the YAML frontmatter (header). If you want to customize the appearance, edit the styles in the generated .docx document to your liking, save it as e.g. `template.docx`, and add the following to the YAML frontmatter.

   ```
   output:
       word_document:
           reference_docx: path/to/template.docx
   ```

   See https://vimeo.com/110804387 for a video example.

2. If you've installed LaTeX, try creating a PDF. Perhaps you don't like the default font in the RStudio PDF template, and would like to change the fonts. You can do so like this within the YAML header:

   ```
   output:
       pdf_document:
           latex_engine: xelatex
   mainfont: Times New Roman
   monofont: Courier
   fontsize: 11pt
   ```

   This asks Pandoc to generate the PDF using a custom LaTeX engine that recognizes the font-related frontmatter options. XeLaTeX also is better able than the default engine to handle non-ASCII characters such as Greek letters. Although the default engine does handle LaTeX math mode expressions just fine: `$\pi$` turns into π ; and the expression

   ```
   $$ \sum_{n=1}^{\infty} 2^{-n} = 1 $$
   ```

   results in the following nice-looking equation.

   $$\sum_{n=1}^{\infty} 2^{-n} = 1$$

3. Write a paper or (gasp) your thesis in R Markdown. It's been done!

- https://rosannavanhespenresearch.wordpress.com/2016/02/03/writing-your-thesis-with-r-markdown-1-getting-
- https://paulvanderlaken.com/2017/09/01/writing-your-thesis-with-r-markdown-1-getting-started/
- http://robertmyles.github.io/2016/04/15/write-your-thesis-or-paper-in-r-markdown/

  and someone has even written a LaTeX template that adheres to Columbia's thesis guidelines:

- https://charlesmcnamara.com/latex-template-for-columbia-university-dissertations/

That's it. Have fun using R Markdown!