Getting started with R Markdown

PSTAT126

Lab 1

Objectives

This short getting started guide introduces the basics of R Markdown (.Rmd) files, which 'knit' together text and code, and some style conventions.

- Elements of .Rmd: code chunks and text
- Executing codes within .Rmd files in RStudio
- Code style guidelines
- Controlling code chunk display in knitted documents
- Controlling text display in knitted documents
- Math display
- Report formatting and code appendices

R Markdown: integrating text and code

R Markdown allows for easy building of documents that integrate codes and code results with text. R Markdown files (.Rmd) contain code chunks and regular text. The code chunks are executable from within RStudio as the document is being composed and edited. This makes .Rmd a great choice for projects involving data analysis in R, because one can simultaneously compose codes *and* narrative without having to keep track of multiple files or documents. This is very conducive to reproducible research and version control.

When an R Markdown file is 'knitted' in RStudio, a report is generated that displays text and code and code results according to author-specified parameters in the document. Your first task in this lab is to experiment with code chunk options and text markdown in the .Rmd file that control the display of codes, text, and results in the knitted document.

Codes: the "R" in R Markdown

First we'll examine aspects of the .Rmd file that pertain to R codes, output, and graphics.

Codes and output

While composing a document, code chunks can be written and executed from within the .Rmd file in RStudio. Below this paragraph in the .Rmd file is an example code chunk with some simple commands. Try running individual lines from the .Rmd file for this document within RStudio. The results will appear below the chunk and in the console.

Reminder: you can execute code by placing your cursor on the line you wish to run and pressing CTRL + ENTER or CMD + ENTER.

Tip: If you wish to run an entire chunk, use the 'play' icon in green that appears in the upper right corner of the chunk.

Now notice that the code chunk is set off by three back-ticks and some text in braces ``` `{some text}` ```. The first part of the text in braces is *always* `r`; optionally, a name for the chunk (above, `example-chunk`); then a comma, followed by **code chunk options** that control how the code will be displayed in the knitted document.

Click the 'Knit' button with the ball-of-yarn icon and have a look at the knitted document. For the code chunk above, nothing appears! To change this behavior we can change the code chunk options. To get started, let's tinker with those options.

Your turn

Make each of the following changes one by one, and re-knit after each change to examine the effect on the knitted document.

i. Change the name of the chunk to your favorite author's last name. After knitting, examine the 'R Markdown' tab in RStudio and look for where the new name appears. Try introducing a command in the code chunk that will produce an error, and knit again.
ii. Change the `results = 'hide'` option to `results = 'markup'`. Also try `results = 'asis'`.
iii. Remove the `results` option altogether and notice the default behavior. Now go all the way to the top of the .Rmd file and examine the `setup` chunk. Try changing the `results` option there to one of the other options. After examining the effect on the knitted document, revert to `results = 'markup'`.
iv. Try adding `echo = T` to the example code chunk.
v. Configure the code chunk options so that the knitted document shows the codes themselves but not the results.

Plots

The chunk below produces the scatterplot you saw in the introductory lecture. However, the title is missing and the axis labels are not clear. See the documentation for the `labs()` function (run `?labs` in the console to open the help file), especially the examples, and figure out how to change the axis labels and title. Give the plot labels that make sense to you. If you like, add a subtitle or a caption!

Now once the plot looks nice, try knitting. Notice that the plot *does not* appear in the knitted document. This is due to the `fig.show = 'hide'` option; remove this and re-knit, and the plot should appear.

Your turn

Often it is necessary to resize plots for knitting to avoid oversize or cut-off images. Try the following chunk options one by one and knit between each change to take note of the effect in the knitted document.

i. Add the option `fig.width = 2` and examine the output. Tinker with the option by changing the number until you find a display width that you like.
ii. Add the option `fig.height = ...`, specifying a number of your choosing for the height. Play until you find a display height that you like.
iii. Notice that the alignment is off-center. Try adding `fig.align = 'center'`.

There are many other code chunk options, but those come up particularly often. The above and other options are summarized in the R Markdown cheatsheet.

Code style guidelines

A few points of style can dramatically improve readability:

- `<-` is used for assignment (not `=`, used to specify function arguments)
- every comma is followed by a space
- spaces on either side of `=` and `<-`, logical operations, and arithmetic operators ('infix' operators)
- short explanatory comments above groups of lines preceded by a blank line
- use descriptive names with `_` to introduce spaces separating name components
- break long commands over multiple lines
- specify function arguments explicitly by name

Here's a simple example. Notice how much easier it is to understand what is done from the code alone when well-styled.

```
# poor style
data1=matrix(c(36.67,35.91,119.42,119.22),2)

# better style
data1 <- matrix(data = c(36.67, 35.91, 119.42, 119.22), nrow = 2)

# good style
latlong_mx <- matrix(data = c(36.67, 35.91, 119.42, 119.22),
                     nrow = 2)
```

Here is a nice short style guide.

Your turn

Correct the following code, and then configure the chunk to display the codes only.

```
##          var1 var2       V3
## 1      adult   32     jen
## 2      adult   65     lin
## 3 adolescent   13 georgia
## 4     infant    2   niall
```

Text: the 'Markdown' in R Markdown

Now we'll look over the text aspects of .Rmd documents.

Narrative text

You've probably noticed by now that headers are set off by `#`, `##`, `###` in the .Rmd document, and these are rendered in nice format in the knitted document. These are examples of text markdown elements, which are used to control the display of text in the knitted document. Some common ones are:

- `#` header, number of hashes specifies header level
- `*italic text*` renders in *italic*
- `**bold text**` renders in **bold**
- `*`, `-`, `+` set off itemize list elements
  - sub-item
- `1.`, `2.`, `..` set off numbered lists
- `[link text](url)` creates hyperlinks

These and other options are summarized in the R Markdown cheatsheet.

Mathematical notation

R Markdown documents also integrate math display using $\LaTeX$ math expressions. For inline math environments, enclose expressions in `$`, such as `$\left(\frac{1}{x}\right)$`, which appears as $\left(\frac{1}{x}\right)$. The .Rmd file has dynamic previewing while building a document; hovering your cursor over the expression while editing will show you what it will look like in the generated document.

Equation blocks can be created by enclosing math expressions in `$$`. For example, `$$Y = \mathbf{x}\beta + \epsilon$$` displays as:

$$Y = \mathbf{X}\beta + \epsilon$$

For a list of LaTeX math syntax, see wikibooks. (These expressions also work on Nectir using the same syntax!)

Your turn

Write the density of a Gaussian random variable in an equation block. Below that, write the mean and variance in inline math mode (preferably in a short sentence).

Output type

Throughout, you've been knitting to html; this is because the html output knits fastest. However, for assignments you will need to submit pdf documents. To knit to pdf, go all the way to the top of the document, and change the `output` part of the header block from `html_document` to `pdf_document`.

Your turn

Try knitting to pdf. Look at the resulting document, and see if there are any formatting issues that need to be fixed – awkward displays, code chunks running off page, etc. – and fix them. Then add your name to the author field (move 'PSTAT126' to the date field), knit, and save. This is your completed copy of lab 1!

Extra: code appendix

Imagine you're writing a report of some data analysis (*e.g.*, a homework for this class). While it is lovely that R Markdown allows you to integrate your analysis and results directly into your written report, very likely there are many code chunks that need to be run but not necessarily shown in your report to answer the questions of interest.

It is good practice to suppress codes or output that your readers don't need to see (with the `results = 'hide'` and/or `include = F` chunk options) in the body of your report. Examples include:

- setup commands like package loads
- data import
- printed data frames

Consider setting the global options to suppress all code and output, so that you have to specifically choose to show a code chunk or result. This is a better policy than defaulting to showing all code and output, which typically results in an unreadable document.

However, your TA might need to check some of those codes if they spot an error somewhere. In that case, including an appendix of all codes at the end can be helpful.

```
# default code chunk options
knitr::opts_chunk$set(echo = F,
                      results = 'markup',
                      message = F,
                      warning = F)

# load packages
library(faraway)
library(tidyverse)
# concatenation `c()`
c(1, 3, 2, 5)

# assignment `<-`
x <- c(1, 3, 2, 5)
x

# check object structure `str()`
str(x)

# construct a matrix
x_mx <- matrix(x, nrow = 2, byrow = F)
x_mx

# vectorized arithmetic and logic
y <- x + 1
x > y
# scatterplot
ggplot(data = eco, mapping = aes(x = usborn, y = income)) +
  geom_point() +
  labs()
# poor style
data1=matrix(c(36.67,35.91,119.42,119.22),2)

# better style
data1 <- matrix(data = c(36.67, 35.91, 119.42, 119.22), nrow = 2)

# good style
latlong_mx <- matrix(data = c(36.67, 35.91, 119.42, 119.22),
                     nrow = 2)
# make me look pretty!
df =data.frame(var1=c('adult', 'adult', 'adolescent', 'infant'),var2=c(32, 65, 13, 2))
df[,3]=c('jen', 'lin','georgia','niall')
df
```