ENV 316: R Markdown Tutorial

David Hall and Hui Peng

Contents

A quick introduction to R markdown **1**

First off, what is R Markdown? 2

How do I get started with R markdown? 2

Great, now what's going on with this R markdown document? 3

How to run R code in R Markdown 4

How do I go from R markdown to something I can hand-in 4

So now what do I do with R Markdown? **5**

R Markdown resources and further reading 5

R code chunk options 6

Inserting images into markdown documents 6

Generating Tables 6

Spellcheck in R Markdown 7

Quick reference on R markdown syntax 7

Example Header Level 1 **7**

Example Header Level 2 7

A quick introduction to R markdown

Howdy!

We understand that virtual classes may not have been what you had in mind when you signed up for a course titled *Laboratory and Field Methods in Environmental Science*, but it does provide us with a great opportunity to really dive into what separates science from a walk in the woods: writing stuff down! As you've probably already heard, the ENV 316 course will make prodigious use of R and R Studio as we explore concepts of environmental chemistry and ecology. Since you're already using R and R Studio for your data analysis and manipulations, we're encouraging you to submit your work using R Markdown.

The aim of this document is to briefly explain what R Markdown is, why you should use it (hint: it'll make everyone's lives easier), and how to create simple documents for this course.

First off, what is R Markdown?

In a nutshell, R Markdown allows you to analyse your data with R and write your report in the same place (this document is written with R Markdown). This has loads of benefits including increased reproducibility, and streamlined thinking. No more flipping back and forth between coding and writing to figure out what's going on. Let's run some simple code as an example:

```
# Look at me go mom
x <- 2+2
x
```

```
## [1] 4
```

What we've done here is write a snippet of R code, ran it, and printed the results (as they would appear in the console). While the above code isn't anything special, we can extend this concept so that our R markdown document contains any data, figures or plots we generate throughout our analysis in R. For example:

```
library(tidyverse)
library(knitr)
airPol <- read_csv("./data/Toronto_60433_2018_Jan2to8.csv",
                   na = "-999")
kable(airPol[1:5, ],
      caption = "Example table of airborne pollutant levels used for Figure 1.")
```

Table 1: Example table of airborne pollutant levels used for Figure 1.

temperature	pollutant	concentration	date
-11.7	NO2	41	2018-01-01 19:00:00
-11.7	O3	2	2018-01-01 19:00:00
-11.3	NO2	28	2018-01-01 20:00:00
-11.3	O3	14	2018-01-01 20:00:00
-11.6	NO2	20	2018-01-01 20:59:59

```
ggplot(airPol, aes(date, concentration, colour = pollutant)) +
  geom_line() +
  theme_classic()
```

Pretty neat, eh? You might not think so, but let's imagine a scenario you'll encounter soon enough. You're about to submit your assignment, you've spent hours analyzing your data and beautifying your plots. Everything is good to go until you notice at the last minute you were supposed to *subtract* value `x` and not value `y` in your analysis. If you did all your work in *Excel* (tsk tsk), you'll need to find the correct worksheet, apply the changes, reformat your plots, and import them into word (assuming everything is going well, which is never does with looming deadlines). Now if you did all your work in R markdown, you go to your one `.rmd` document, briefly apply the changes and re-compile your document.

How do I get started with R markdown?

As you've already guessed, R markdown documents use R and are most easily written and assembled in the R Studio IDE. If you have not done so, download R from the comprehensive R

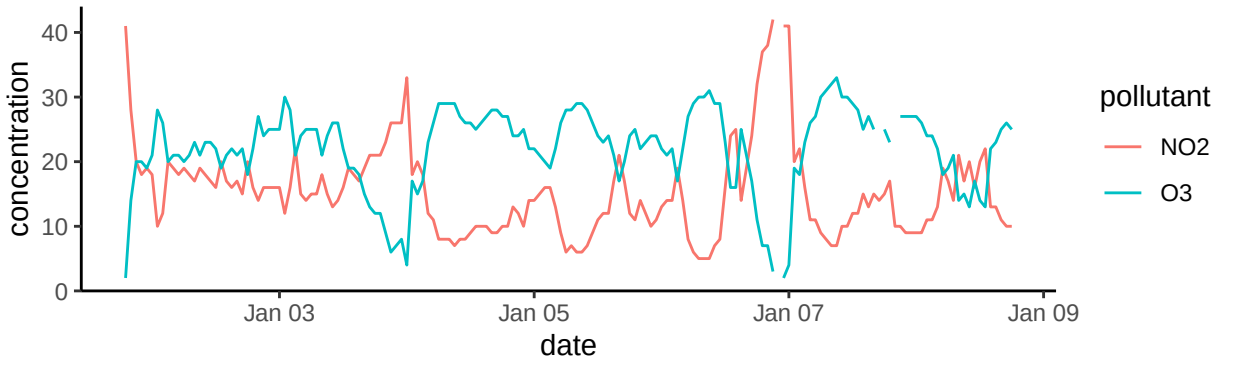

Figure 1: Time series of 2018 ambient atmospheric O_3 and NO_2 concentrations (ppb) in downtown Toronto

archive network (CRAN), link here: `http://cran.utstat.utoronto.ca/`, and R Studio, link here: `https://rstudio.com/products/rstudio/download/`). Follow the listed instructions and you should be well on your way. You can also see the accompanying *Working with RStudio* document on Quercus for additional top tips.

Once setup with R and R Studio, we'll need to install the `rmarkdown` and `tinytex` packages. In the console, simply run the following code:

```
install.packages("rmarkdown") # downloaded from CRAN

install.packages("tinytex")
tinytex::install_tinytex()  # install TinyTeX
```

The `rmarkdown` package is what we'll use to generate our documents, and the `tinytex` package enables compiling documents as PDFs. There's a lot more going on behind the scenes, but you shouldn't need to worry about it.

Now that everything is setup, you can create your first R Markdown document by opening up R Studio, selecting `FILE -> NEW FILE -> Rmarkdown`. A dialog box will appear asking for some basic input parameters for your R markdown document. Add your title and select PDF as your default output format (you can always change these later if you want). A new file should appear that's already populated with some basic script illustrating the key components of an R markdown document.

Great, now what's going on with this R markdown document?

Your first reaction when you opened your newly created R markdown document is probably that it doesn't look anything at all like something you'd show your TA. You're right, what you're seeing is the plain text code which needs to be compiled (called *knit* in R Studio) to create the final document. Let's break down what the R markdown syntax means then let's knit our document.

When you create a R markdown document like this in R Studio a bunch of example code is already written. You can compile this document (see below) to see what it looks like, but let's break down the primary components. At the top of the document you'll see something that looks like this:

```
---
title: "Untitled"
author: "David Hall"
date: "24/08/2020"
```

```
output: pdf_document
---
```

This section is known as the *preamble* and it's where you specify most of the document parameters. In the example we can see that the document title is "Untitled", it's written by yours truly, on the 24th of August, and the default output is a PDF document. You can modify the preamble to suit your needs. For example, if you wanted to change the title you would write `title: "Your Title Here"` in the preamble. Note that none of this is `R` code, rather it's `YAML`, the syntax for the document's metadata. Apart from what's shown you shouldn't need to worry about this much, just remember that indentation in YAML matters.

Reading further down the default R markdown code, you'll see different blocks of text. In R markdown anything you write will be interpreted as body text (i.e .the stuff you want folks reading like this) in the knitted document. **To actually run R code** you'll need to see the next section.

How to run R code in R Markdown

There's two ways to write `R` code in markdown:

- **Setup a code chunk**. Code chunks start with three back-ticks like this: ` ```{r} `, where `r` indicates you're using the R languauge. You end a code chunk using three more backticks like this ` ``` `.
  - Specify code chunks options in the curly braces. i.e. ` ```{r, fig.height = 2} ` sets figure height to 2 inches. See the *Code Chunk Options* section below for more details.
- **Inline code expression**, which starts with `` `r `` and ends with `` ` `` in the body text.
  - Earlier we calculated `x <- 2 + 2`, we can use inline expressions to recall that value (ex. We found that x is 4)

A screenshot of how this document, the one you're reading, appeared in R Studio is shown in Figure 2.

To actually run your R code you have two options. The first is to run the individual chunks using the *Run current chunk* button (See figure 2). This is a great way to tinker with your code before you compile your document. The second option is to compile your entire document using the *Knit document* button (see Figure 2). Knitting will sequentially run all of your code chunks, generate all the text, knit the two together and output a PDF. You'll basically save this for the end. **Note all the code chunks in a single markdown document work together like a normal R script.** That is if you assign a value to a variable in the first chunk, you can call this variable in the second chunk; the same applies for libraries. **Also note that every time you compile a markdown document, it's done in a "fresh" R session.** If you're calling a variable that exist in your working environment, but isn't explicitly created in the markdown document you'll get an error.

How do I go from R markdown to something I can hand-in

To create a PDF to hand in you'll need to compile, or knit, your entire markdown document as mentioned above. To knit (or compile) your R markdown script, simply click the *knit* button in R Studio (yellow box, Figure 2). You can specify what output you would like and R Studio will (hopefully) compile your script.

If you want to test how your code chunks will run, R Studio shows a little green 'play button' on the top right of every code chunk. this is the 'run current chunk' button, and clicking it will run your code chunk and output whatever it would in the final R markdown document. This is a great way to tweak figures and codes as it avoids the need to compile the entire document to check if you managed to change the lines from 'black' to 'blue' in your plot.

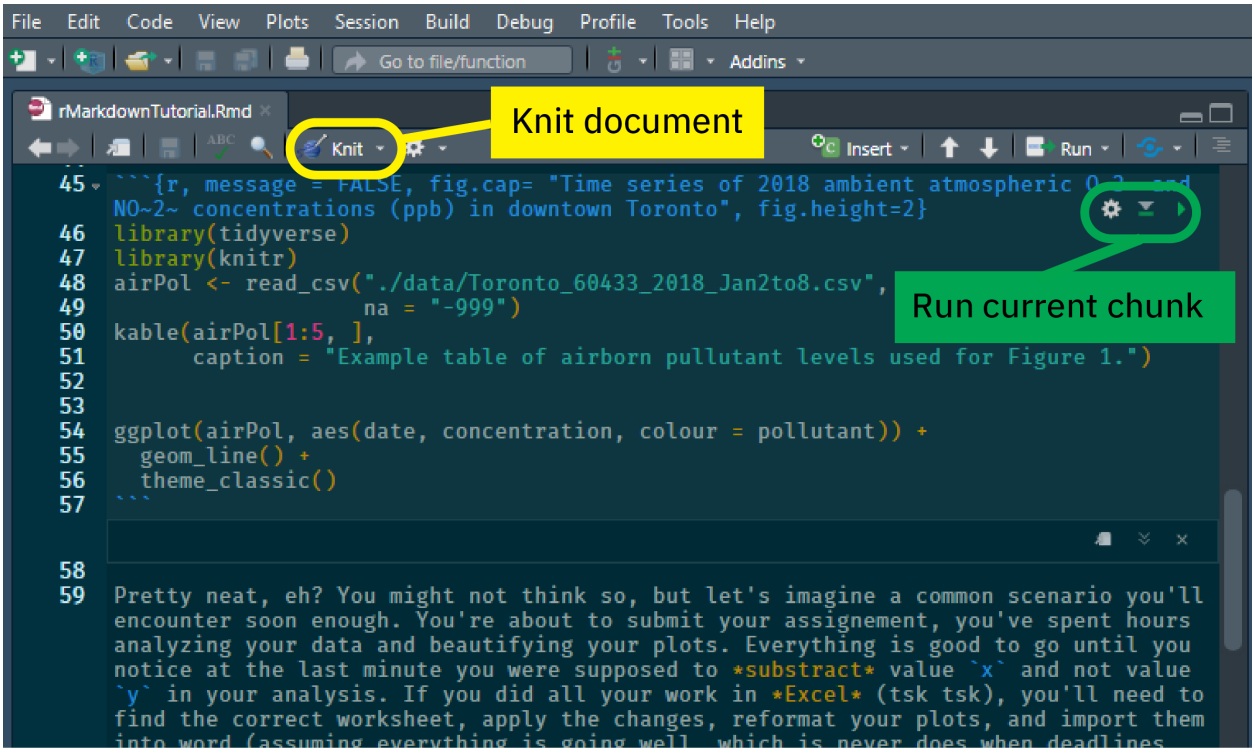

Figure 2: How this document, the one you're reading, appeared in RStudio; to see the final results scroll up to Figure 1. Note the "knit" and "run current chunk" buttons.

So now what do I do with R Markdown?

You do science and you write it down!

In all seriousness though, this document was only meant to introduce you to R markdown, and to make the case that you should use it for your ENV 316 coursework. A couple of the most useful elements are talked about below, and there is a wealth of helpful resources for formatting your documents. Just remember to keep it simple, there's no need to reinvent the wheel. The default R markdown outputs are plenty fine with us.

R Markdown resources and further reading

There's a plethora of helpful online resources to help hone your R markdown skills. We'll list a couple below (the titles are links to the corresponding document):

- Chapter 2 of the *R Markdown: The Definitive Guide* by Xie, Allair & Grolemund (2020). This is the simplest, most comprehensive, guide to learning R markdown and it's available freely online. Stuff to read up on includes:
- The R markdown cheat sheet, a great resource with the most common R markdown operators; keep on hand for quick referencing.
- bookdown: Authoring Books and Technical Documents with R Markdown (2020) by Yihui Xie. Explains the `bookdown` package which greatly expands the capabilities of R markdown. The table of contents of this document is created with `bookdown`.

R code chunk options

You can specify a number of options for an individual R code chunk. You include these at the top of the code chunk. For example the following code tells markdown you're running code written in `R`, that when you compile your document this code chunk should be evaluated, and that the resulting figure should have the caption "Some Caption". A list of code chunk options is shown below:

````
```{r, eval = FALSE, fig.cap = "Some caption}

# some code to generate a plot worth captioning.

```
````

option	default	effect
eval	TRUE	whether to evaluate the code and include the results
echo	TRUE	whether to display the code along with its results
warning	TRUE	whether to display warnings
error	FALSE	whether to display errors
message	TRUE	whether to display messages
tidy	FALSE	whether to reformat code in a tidy way when displaying it
fig.width	7	witdh in inches for plots created in chunk
fig.height	7	height in inches for plots created in chunk
fig.cap	NA	include figure caption, must be in quotation makrs ("")

Inserting images into markdown documents

Images not produced by R code can easiyl be inserted into your document. Between paragraphs insert the following code. Note that compiling to PDF, the LaTeX call will place your image in the "optimal" location, so you might find your image isn't exactly where you though it would be. A quick google search can help you out if this is problem.

```
![Caption for the picture.](/path/to/image.png){width=50%, height=50%}
```

Note that in the above the use of image atributes, the `{width=50%, height=50%}` at the end. This is how you'll adjust the size of your image. Other dimensions you can use include `px, cm, mm, in, inch`, and `%`.

Generating Tables

There's multiple methods to creat tables in R markdown. Assuming you want to display results calculted through R code, you can use the `kable()` function. Please consult Chapter 10 of the *R Markdown Cookbook* for additional support.

Alternatively, if you want to create simple tables manually using the following code in the main body (you can obviously increase the number of rows/columns and the location of the horizontal lines.) To generate more complex tables, see the `kable()` function and the `kableExtra` package.

```
Header 1 | Header 2| Header 3
---------|---------|---------|
Row 1    | Data    | Some other Data
Row 2    | Data    | Some other Data
---------|---------|---------|
```

Header 1	Header 2	Header 3
Row 1	Data	Some other Data
Row 2	Data	Some other Data

Spellcheck in R Markdown

While writtin an R markdown document in R studio, go to the `Edit` tab at the top of the window and select `Check Spelling`. You can also use the `F7` key as a shortcut. The spell checker will literally go through every word it thinks you've misspelled in your document. You can add words to it so your spell checker grows with your use. **Note** that the spell check with also check your R code; be wary of changing misspelt word in your code because you may get an error down the line.

Quick reference on R markdown syntax

- **Inline formatting**; *which* is $_{\text{used}}$ to $^{\text{format}}$ `your text`.

1. Numbered lists

- Normal lists
  - Lists
- **Block-level elements**, i.e. you're section headers

Example Header Level 1

Example Header Level 2

Example Header Level 3

Example R markdown syntax used for formatting shown above:

```
- **Inline formatting**; *which* is ~used~ to ^format^ `your text`.
1. Numbered lists
- Normal lists
  - Lists

- **Block-level elements**, i.e. your section headers

# Headers
## Headers
### Headers
```