Welcome to Reproducible Research in R

Tad Dallas

Reading for this week:

> Munafò, M. R., Nosek, B. A., Bishop, D. V., Button, K. S., Chambers, C. D., Du Sert, N. P., ... & Ioannidis, J. P. (2017). A manifesto for reproducible science. Nature human behaviour, 1(1), 1-9.

Structure of the class

Syllabus

We will go over this the first day of class. The first week will be no recorded lectures, and then the second week we will dive into the course design of 1 recorded lecture and 1 discussion/deep dive/clarification lecture.

My expectations

- I expect that you apply yourself in an earnest attempt to learn the material. Everyone learns at their own pace, but I take it that if you were not interested in the material, you would not have signed up for the class? I think everyone here is excited to learn R and more about issues around reproducibility.
- I expect that you treat other students in the course fairly. I will not tolerate any form of discrimination or disparaging remarks to your colleagues.
- I expect that you do your own work. Every coder writes code differently. If I see the same code from two different people, it is a huge red flag. I will not accept copying of code, even if the original code was in the creative commons (you will get this joke later in the course if you do not understand now).

Your expectations

We will discuss these in the lecture. I would say 'think-pair-share', but I have no idea how to do that with Zoom.

What is reproducible research?

Reproducibility is obtaining consistent results using the same input data; computational steps, methods, and code; and conditions of analysis. This definition is synonymous with "computational reproducibility," and the terms are used interchangeably in this report.

Replicability is obtaining consistent results across studies aimed at answering the same scientific question, each of which has obtained its own data.

What are some bad things that can happen when we fail to focus on reproducibility?

Science is evidence-based by definition, and findings build on one another. Being able to reproduce the findings of other researchers is therefore nearly as important as contributing 'new' evidence. Researchers are fallable in their ability to write bug-free code, and by releasing code and data, these errors can be detected and corrected. Apart from this, reproducibiltiy and open-science sort of go hand-in-hand, in that code that is reproducible is also ideally open access, meaning that any researher has the ability to take the code and data and reproduce the analyses. We will go over many open science tools in this course (e.g., `R`). Some more examples of this given below.

Linux operating system

You do not have to use a Linux operating system in order to do reproducible research. Ideally, the point of reproducible research is such that the analyses will run on any operating system. However, often the tools that researchers use vary by operating system, and this is a huge roadblock to doing reproducible research.

To get everyone on the same page, and to promote the use of open source software, I encourage you to use a Linux operating system called Ubuntu. I have flash drives that I can pass along that are persistent Ubuntu operating systems. If you opt to not use Ubuntu, you will have to be able to download and implement the following in your native OS (e.g., Windows, Mac OS, etc.):

- bash shell
- make
- git
- R

Windows

Windows 10 should be able to run a Linux subsystem, which is necessary for getting at least one or two of the above working.

https://www.howtogeek.com/249966/how-to-install-and-use-the-linux-bash-shell-on-windows-10/

Mac OS

Mac OS is basically a Linux ripoff, so it has the bash shell and everything needed. However, if you are not using Linux, I will not be able to help troubleshoot, and you will be on your own there.

An important note for the brave:

I have 10 USB drives containing a persistent Ubuntu 18.04 operating system. This would be a nice way to familiarize yourself with a Linux environment without having to change anything on your computer. Please reach out if you want one. I ask that you return them after the semester is over, but honestly, it is yours if you promise to keep using it

The tools you will learn

The software listed above (i.e., bash shell, make, git, and R) can be used in concert to create reproducible workflows for analyses.

What is markdown?

> Markdown is a lightweight markup language that you can use to add formatting elements to plaintext text documents.

Why is it useful?

Independent of operating system (portable and platform independent)

Markdown is "future-proof" (plain text with some sprinkles, so it can be read by most any text editor)

Can be used to make websites, presentations, email, documentation, and academic papers.

Markdown syntax

Markdown reads like plain-text, but can be compiled into hmtl, pdf, and other useful formats. It has a bunch of benefits over other text formats (e.g., docx) in that it can do syntax formatting quite easily, is incredibly simple, and is independent of operating system. In an application like Microsoft Word, you click buttons to format words and phrases, and the changes are visible immediately. Markdown isn't like that. When you create a Markdown-formatted file, you add Markdown syntax to the text to indicate which words and phrases should look different.

We will first go over the syntax of markdown, then introduce embedding code chunks and making a reproducible document. This last part is probably the biggest benefit of using markdown.

Headers

Creating headings (like the one above that says "Headers") is quite easy, and is accomplished with nested "#" symbols. Heading level 1 (the largest text heading) is acheived with 1 "#" followed by a space and the heading text.

```
# Heading 1
## Heading 2
### Heading 3
#### Heading 4
##### Heading 5
```

Heading 1

Heading 2

Heading 3

Heading 4

Heading 5

Paragraphs

Paragraphs are started by just typing in text with one hard return after the heading title. For many text editors, this is not even necessary, and much of markdown writing handles blank space really well (try this out for yourself by increasing the number of lines between blocks and text and observing the lack of difference it makes).

There is no need to format the text yourself in terms of indentation or anything. Markdown will do this for you, and you can control how the document appears using the *yaml* header. *yaml* stands for "yet another markup language" and is used to template how the markdown appears. For instance, in this document, the *yaml* header is

```
---
title: "An introduction to R and R markdown"
author: Tad Dallas
date:
output: pdf_document
---
```

This provides title and author information, as well as information on how to output the file. The raw markdown will be formatted into one of many different formats, with the two most common being "pdf" or "html".

Emphasis

Bold and italic characters are straightforward in markdown, with italics denoted with a "_" character on either side of the italicized text (e.g., "*word*" becomes *word*).

Bold text is just as simple, with two "*" characters surrounding whatever bold text you would like (e.g., "**word**" becomes **word**).

Block quotes

Indented and pretty block quotes are easy to add as well, by just using the ">" operator.

```
> Everything was beautiful, and nothing hurt.
```

becomes

> Everything was beautiful, and nothing hurt.

Lists

Bulleted or enumerated lists are simple as well.

```
1. Break an egg
2. Make an omelette
3. Eat the omelette
```

becomes

1. Break an egg
2. Make an omelette
3. Eat the omelette

Itemized lists without the enumeration (no numbers, so just a bulleted list) are acheived with similar syntax. Here any of the following can be used to delineate an item in a list ("*, -, +"), so

```
+ Eggs
* Grits
- Cheese
```

becomes

- Eggs
- Grits
- Cheese

Hyperlinks

Hyperlinks can be embedded using the syntax

`[link](url)`

For instance, if you want to download R click here

Images

Images can be embedded into markdown without the copy-paste approach of things like Microsoft Word, which can butcher images depending on OS (have you ever tried to open a Powerpoint on a different computer and all the formatting and images are messed up?)

To embed an image, the syntax is simply

`![caption]{image}`

an example

Figure 1: cat

Code

Now we can get into why markdown is really useful, and how we will use it in this course. Markdown allows for the embedding of code chunks, and will highlight their syntax.

```
y <- 1
y
```

This is nice for tutorials or when we need to clearly show code. But what if we also want the code to be executable (i.e., to run and produce output)? This is where R markdown comes in.