R Markdown

STAE04: Data Visualization

Johan Larsson

2020-10-16

Contents

1 R Markdown

This document is written using R Markdown. R Markdown is a syntax for formatting documents that lets you focus on content. You write text (including R code) in a standard text document with the ending `.Rmd` and then R Markdown (plus a handful of very useful packages) turns your text into a neatly formatted document.

This document serves as both a template for your submissions during the data visualization course as well as an introduction into R Markdown.

If you are here for the introduction, simply open this file in R Studio and keep reading.

1.1 Installation

To get started, you are going to need two packages: **rmarkdown** and **knitr**.

```
install.packages(c("rmarkdown", "knitr"))
```

You will also need a distribution of LaTeX. Installing LaTeX can be installed easily using the **tinytex** package (if you don't already have LaTeX installed). Do so right now by calling the following lines of code. If you are currently in the R Studio editor when you read this, you can simply highlight the text and

hit `Ctrl/Cmd + Enter` or put the cursor inside the code chunk below and hit `Ctrl/Cmd + Shift + Enter`.

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

1.2 Knitting

Now that you have LaTeX installed, we can turn this document into a pdf report by **knitting** it. To do so in R Studio, simply hit `Ctrl/Cmd + Shift + k`. Doing so will prompt R to run through all of your code blocks and text and pass this on to LaTeX to render your document into a pdf file, which should open up on your screen. As you keep on reading through this `.Rmd` file, try to keep an eye on the pdf document to see what the output looks like.

R Markdown files can be knitted to a wide range of medium, including PDF files, HTML, and word documents. In this course, however, you will always produce PDF files from your R Markdown documents.

1.3 YAML

An R Markdown file ends with `.Rmd` and is best edited using RStudio. Each Rmarkdown file starts with a so called YAML block; here is a bare-bones one:

```
---
title: "An Awesome Title"
author: "Fantastic Me"
date: "2020-09-28"
output: pdf_document
---
```

For this course, please use the YAML block supplied in this template, modifying only the `author`, `date`, and `title` fields. You should also remove the `references` item in the YAML block unless you need references in your report.

1.4 Formatting

R Markdown is an extension of Pandoc markdown, which uses a special—but very simple—syntax for formatting text. The following are some ways in which you can format text in R Markdown.

Start a new paragraph by surrounding text with blank lines.

Single asterisks italicize text, **double asterisks format text in bold**, and `grave accents formats text in monospace` (which is suitable whenever you want to include code).

To create lists in markdown, you add a

- dash before each item in the list, and
  - possibly asterisks for sub-items in the list,
- placing items each on its own line, and
- adding blank lines before and after the list.

Numbered lists are

1. created similarly, but
2. using numbers instead of dashes.

Sections are created by prefacing the section title with a hash tag (`#`):

```
# One Hashtag Creates a Section

## Two Hashtags Creates a Subsection

### Three Hastags Creates a Subsubsection
```

R Markdown can even format quotes for you!

> If you want to quote something, adding a `>` before the text creates a block quote —Johan Larsson

There are many ways to format tables in markdown, but the simplest one is two simply create columns of text with dashes (`---`) separating the title of each column from the cells of the table.

Table 1: A caption for the table can be added like this.

Header 1	Header 2
Cell 1	Cell 2
Cell 3	Cell 4

To add a link in markdown, you can either simply surround the URL with brackets (as in this link to a R Markdown cheatsheet): https://rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf or you can use brackets and parentheses to provide your own text for the link.

Images can be added with syntax similar to the one for links, with the text inside brackets indicating the caption for the figure. Let's first download an image to our working directory.

```
download.file(
  "https://upload.wikimedia.org/wikipedia/commons/2/29/Minard.png",
  "minard.png"
)
```

Footnotes can be useful to provide additional information[1]. For a longer footnote, it might be better to refer to it with a number[2].

It is also possible to add citations in R Markdown but this is somewhat complicated if you are not familiar with markdown, pandoc, or LaTeX since before. In this course you will usually never need more than one or two citations, so we suggest you write the citations manually, which means you can skip the next paragraph (unless you are keen on getting deeper into R Markdown and pandoc).

[1] If the footnote is short, it is often best to write it in-line, like this.
[2] Then you can add your text separately in the document.

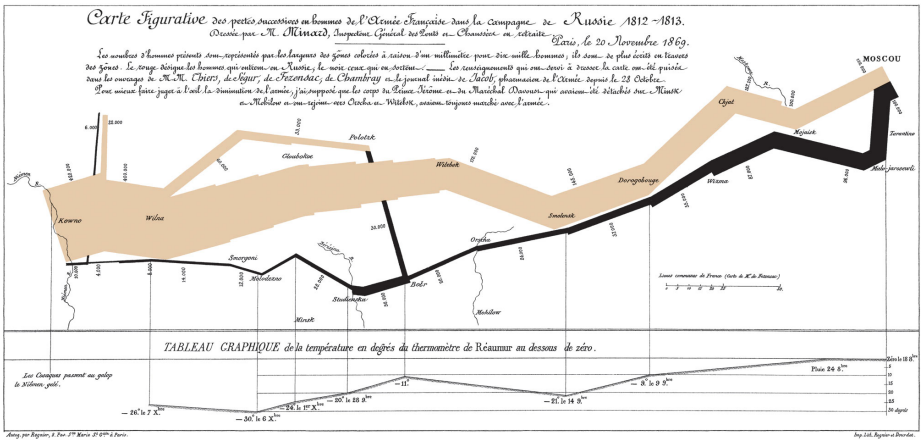


Figure 1: Here is Charle's Minard famous Napoleon chart, for instance. (Public domain.)

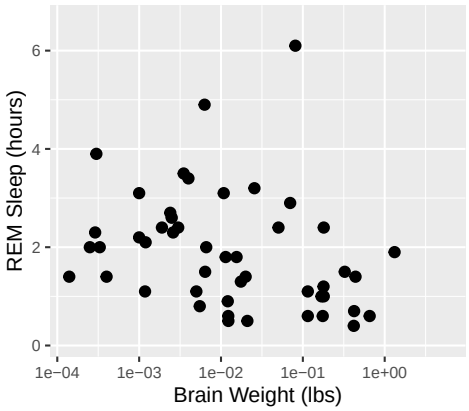
To cite in R Markdown, you will need either 1) a `.bib` file (with bibtex-formatted references) somewhere in your working directory or 2) a `references` field in the YAML block; in this document we use the latter but only to keep this document self-contained. Using a `.bib` file is recommended. Find the key of the reference you are looking for and preface it with an `@`, like the following reference to Wickham (2010), which creates a text reference. If you surround the key with brackets, you instead get a non-text citation (Wickham 2010). If you've done everything right, the final document will get a bibliography at the end (as in this one).

1.5 Code Chunks

So far we've only really talked about features that are made possible by pandoc and its flavor of markdown. But what makes R Markdown special is that is allows us to write code chunks containing R code. This is a code chunk that plots a simple plot using ggplot2:

```
library(tidyverse)

ggplot(msleep, aes(brainwt, sleep_rem)) +
  geom_point() +
  scale_x_log10() +
  labs(x = "Brain Weight (lbs)", y = "REM Sleep (hours)")
```



As you can see, we've started the code chunk with ```{r} and ended it with ```. Everything in between will be treated as R code, just as if you would have written in in an R script of the R terminal. When you compile this document all this code will be run and if it produces any output (text, plot, tables) then that output will make it into the final document.

You can control many settings by changing the parameters in the header of the code chunk. In the following chunk we've changed the width and height of the figure as well as added a caption to the figure. These are settings that will be **incredibly** useful to you during the course.

```
ggplot(msleep, aes(brainwt, sleep_rem)) +
  geom_point() +
  scale_x_log10() +
  labs(x = "Brain Weight (kg)", y = "REM Sleep (hours)")
```

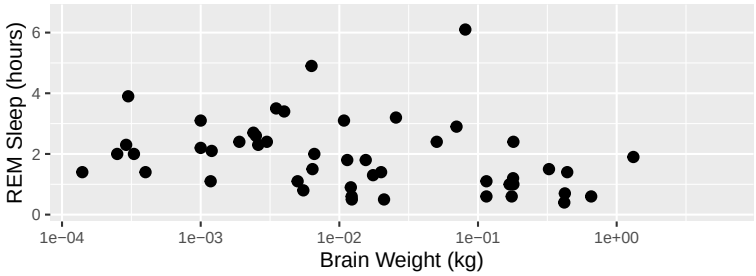


Figure 2: Brain weight and REM sleep duration for mammals.

There are a few other settings that will also come in handy: `include`, `eval`, `results`, and `echo`.

`echo` Setting `echo = TRUE` means that the code in your code chunk will be included in the output (your pdf report). In this template, `echo` is set to `TRUE` by default (see the first code hunk header "setup" in this document), but sometimes you only want the output of your code and not the code itself, which is often the case when you create figures.

`eval` Setting `eval = FALSE` means that the code in your code chunk will not be evaluated at all. This is often useful when you want to show how something

is done (like installing a package) but don't want to do this every time you knitr your document.

`results` Setting `results = "hide"` means that the results generated from the code won't make it into the output (your pdf report). This is sometimes useful if you want to run code interactively but not have it end up in your output. It can also be useful if the code you run produces output as a side-effect.

`include` Setting `include = FALSE` mean that the code will be evaluated but neither the output not the code will be included. This is equivalent to setting `echo = FALSE`, `eval = TRUE`, and `results = "hide"`.

Here are a couple of examples of these arguments (note that `echo = TRUE`, `include = TRUE`, `eval = TRUE`, and `results = "show"` are the defaults):

```
## [1] -2.028 -0.859 -0.138  0.398  0.904
```

```
# this code block will show in the output,
# but the results of `rnorm(50)` will not
rnorm(5)
```

```
# this code will not be evaluated at all
rnorm(5)
```

1.6 Reproducibility

Authoring your documents using R Markdown facilitates reproducibility. Because you need to supply all the code used to produce your paper in the `.Rmd` file, this makes it much easier for other people to re-run your analysis and use your code.

2 Learning More About R Markdown

If you want to learn more about R Markdown, we recommend the R Markdown Cookbook. If you run into any issues with R Markdown, please use the course's discussion board on Canvas or search stack overflow with the `[r-markdown]` or `[knitr]` tag.

References

Wickham, Hadley. 2010. "A Layered Grammar of Graphics." *Journal of Computational and Graphical Statistics* 19 (1): 3–28. https://doi.org/10.1198/jcgs.2009.07098.