R Markdown for Social Science

Michael DeCrescenzo
University of Wisconsin–Madison

September 19, 2018

Abstract

This document introduces R Markdown and describes its utility in a social science workflow. In additional to a conceptual introduction to R Markdown and its syntax, we also cover similar ground as we did with LaTeX, focusing on bibliographies and statistical software integration. In a separate folder we discuss Xaringan, a slideshow system built on R Markdown.

Contents

1 What is R Markdown? 2
1.1 The `'R'` Component 2
1.2 The *'Markdown'* Component 3

2 How R Markdown works 4
2.1 Header (YAML) 4
2.2 Body (Markdown) 5
2.3 Code chunks 5
2.4 LaTeX (Math and Beyond) 6
2.5 Compilation 7

3 Writing with R Markdown 7
3.1 Software 7
3.2 Strategy 8
3.3 When to use R Markdown 8

4 Content control with R Markdown 8
4.1 Examples using Pokémon data 8
4.2 Tables (with `kable`) 9
4.3 Figures 11
4.4 Bibliographies 12

5 Final words **13**

1 What is R Markdown?

R Markdown is a document format that combines the simple, pleasurable writing experience of *Markdown* with the technical capabilities of `R`. R Markdown files are saved with the file extension `.rmd`.

Install it using R:

```
install.packages("rmarkdown")
```

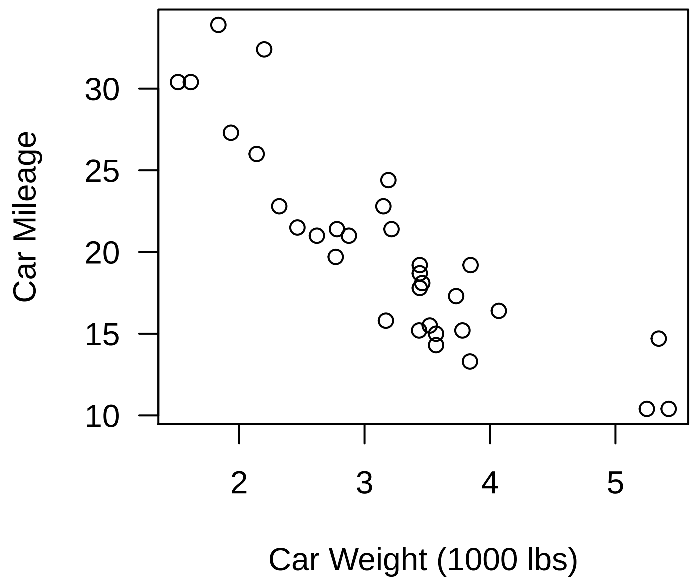
1.1 The 'R' Component

R Markdown integrates `R` code with your output document. Let's demonstrate with some commonly used data on automobiles.

```
# summarize the first four columns in the dataset
summary(mtcars[ , 1:4])
```

```
##       mpg             cyl             disp             hp
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0
```

```
# plot the relationship between mpg (y) and weight (x)
plot(mpg ~ wt, data = mtcars,
     xlab = "Car Weight (1000 lbs)",
     ylab = "Car Mileage",
     las = 1)
```



This may seem unhelpful at first, but think about the LaTeX workflow processes that we have already discussed. To pass a table from R to LaTeX, we must…

- Create the table with R
- Convert the table to LaTeX
- Save the table as a `.tex` file
- Import the table into your main `paper.tex` file

While powerful, there's no denying that this process has you juggling between programs, saving a number of auxiliary files, futzing with file pathways, and so on. A similar process exists for figures.

In R Markdown, however, you can write R code for tables and figures alongside the text of your paper. This allows you to include manage your R code and your output all in one file.

1.2 The *'Markdown'* Component

LaTeX, for all its benefits, is an **ugly** language. It takes *a lot* of code overhead to produce some simple text. If I were to write the previous sentence in LaTeX, I would write…

```
{\LaTeX}, for all its benefits, is an \textbf{ugly} language.
It takes \emph{a lot} of code overhead to produce some simple text.
```

Notice how Markdown achieves similar formatting with slimmer syntax:

```
\LaTeX\, for all its benefits, is an **ugly** language.
It takes _a lot_ of code overhead to produce some simple text.
```

Markdown is a plain text markup format. It is most commonly used to write documents that are compiled to `HTML`, but it can be used for `PDF` as well. For our purposes, Markdown still allows you to control the *content* of the writing using a text markup style, but the markup syntax itself is less verbose and easier to write with. The following section has helpful links for learning about Markdown.

2 How R Markdown works

This section will walk through some basics of R Markdown. For your future reference:

- learn more about R Markdown on the RStudio website.
- A comprehensive resource on the R Markdown ecosystem is the bookdown web page by Yihui Xie (the primary developer).[1]

2.1 Header (YAML)

R Markdown has a header, similar to a LaTeX preamble, but it is controlled by the language `YAML`. YAML has a very minimal syntax for supplying values to variables. If we use **RStudio's** File menu to create a new R Markdown document, we get a YAML header that looks something like this.

```
---
title: "Title goes here"
author: "Janet Authorwoman"
date: "9/18/2018"
output: pdf_document
---
```

YAML headers are bracketed by two sets of `---`, comments are denoted by # symbols.

The current file's YAML header is more complicated, because it takes some work to make a PDF created with R Markdown look better. Open up this document's source file to take a look. One important component is that we use an output format called `pdf_document2` which is located within the `bookdown` package for R. The reason why we use this is because it has enhanced capabilities for referencing tables and figures, so it's pretty important for our purposes. Install bookdown like so:

[1] I refer to "R Markdown ecosystem" as the set of packages that incorporate R Markdown in some capacity, which is large. This includes `bookdown` (a package for writing books with R Markdown), `blogdown` (a package for blogging), `xaringan` (for slideshows), and so on.

```r
install.packages("bookdown")
```

If you examine the source code, you will notice that some options are indented differently. That is because some variables operate globally, and others operate as sub-components of other options, so they are nested “within” other options. You see this most with the `output` variable in the current document.

If we wanted, we could import a `.tex` file of preamble code to modify the final PDF document, or we could style the whole document according to a Pandoc template.

2.2 Body (Markdown)

Once the YAML header is complete (we close the second `---`), we can begin writing in Markdown. I won’t exhaustively describe how Markdown syntax becomes there are a handful of helpful websites that can take care of the task. This one by Adam Prichard is pretty good. You can also check the R Markdown “Cheat Sheet” and the R Markdown “Reference Guide”, both by RStudio.

2.3 Code chunks

R code can be included (and executed) using *code chunks*. A code chunk is set aside between two sets of triple backticks: `` ``` ``. Let’s say we enter the following into our `.rmd` file…

~~~
```{r chunk-demonstration}
x <- 1 + 1
x
```
~~~

In this chunk, we create an object called `x` and then ask R to print it. Here’s what it looks like when we evaluate the chunk.

```r
x <- 1 + 1
x
```

```
## [1] 2
```

Here’s how the chunk works. Within the curly brackets, we specify that the language is `R`, and we give the code chunk a name (in this case, the name is `chunk-demonstration`). Chunks don’t need names, but they turn out to be useful for data caching and cross-referencing tables or figures. Within the curly brackets is where we would also place any *chunk options*, which control…

- whether the code chunk is executed (`eval`),
- whether the code is printed in the output document (option `echo`),

- whether the results of the code chunk are displayed in the output document (`include`),
- whether the results of the code should be saved in a cache (`cache`),
- parameters of any outputted figures, and so on (`fig.width`, `fig.height`, `fig.cap` [caption]).

For example, a chunk that is not executed (but is printed) would be given the option `eval = FALSE`:

~~~
```{r, eval = FALSE}
1 + 1
```
~~~

Notice how the code below is printed but we don't see the results.

```
1 + 1
```

Similarly, a chunk whose code is not printed, but whose results are printed, would be given the option `echo = FALSE`.

~~~
```{r, echo = FALSE}
names(mtcars)
```
~~~

If we evaluate this code chunk, the results are printed even though the code is not. Notice:

```
##  [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear"
## [11] "carb"
```

Read all about chunk options here.

2.4 LaTeX (Math and Beyond)

Here's a weird thing. LaTeX works in R Markdown. R Markdown provides many capabilities for text styling that are much less verbose than LaTeX code, so that is convenient. But it is possible to use LaTeX code to *emphasize* or **embolden** some text.

LaTeX is less redundant for math, of course. We can do inline math ($y_i = \alpha + \beta x_i$) or display math:

$$p\left(y_i = 1\right) \sim \text{Bernoulli}\left(\pi_i\right) \tag{1}$$

$$\log\left(\frac{\pi_i}{1-\pi_i}\right) = X_i\beta \tag{2}$$

When you are compiling to PDF, R Markdown translates your `.rmd` file into a `.tex` file, so your math is created with actual LaTeX code. When you are compiling to HTML, the system uses an online library called `MathJax` to typeset math. (I recommend researching MathJax if you want to create a blog that features any math.)

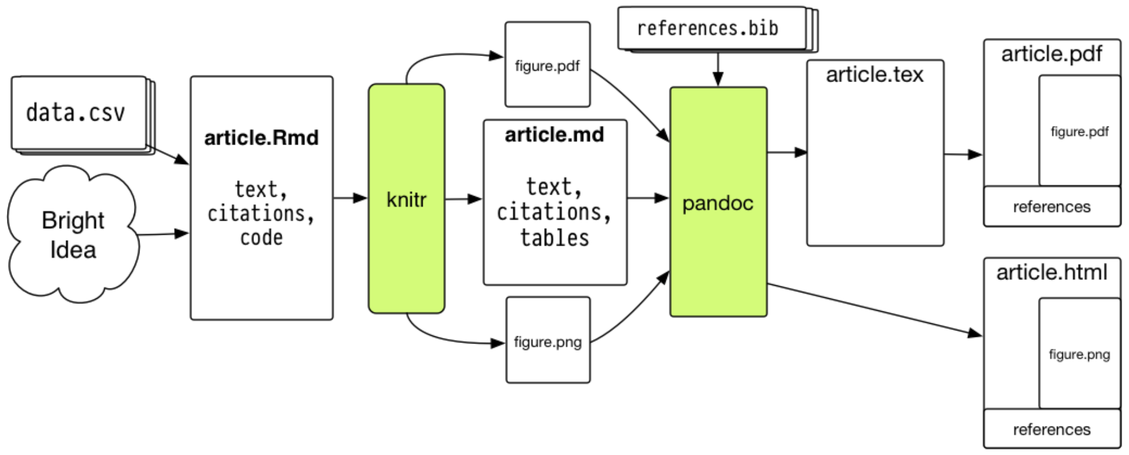


Figure 1: A diagram of the R Markdown compilation process, by Kieran Healy

2.5 Compilation

How does an R Markdown document get interpreted and built? Once you have an `.rmd` document file, here's the process (see also Figure 1).

- The `knitr` package (for R) processes any R code, creating a markdown file `.md`. During the "knitting" process, the code blocks are formatted, output files (tables and figures) are generated using R and saved.
- `Pandoc`, a document conversation program that predates R Markdown, converts the Markdown file into your final output. Pandoc can create PDFs (by way of LaTeX), HTML documents, and even Microsoft Word documents.

The system is a little complex, but the developers of R Markdown have done a great job hiding the messy details from you. The main downside is debugging—it is sometimes unclear if you have a `knitr` problem, a `pandoc` problem, or what.

3 Writing with R Markdown

3.1 Software

As you write an R Markdown document, your text and your code can be sitting side-by-side. This can be difficult to decipher if your code editing software isn't very powerful.

I recommend using **RStudio** to write with R Markdown, as it's designed with lots of helpful capabilities out of the box (including pre-formatted R Markdown documents, helpful keyboard shortcuts, easy graphical interface for "knitting" R Markdown documents, inline and instant output displays, Git capabilities, and so on).

3.2 Strategy

One concern about R Markdown is that a document that contains R code and writing is just too much. How can I include all of my stats, *and* all of my writing in the same file?

Well… you wouldn't! It's okay to have some analysis lurking behind the scenes, as long as your project builds those files along the way.

For example, imagine a project where I need to clean data, run models, create tables and figures, and then do writing. I could keep everything but the results in `clean.R`, `models.R`, and `tables-and-figs.R`. I would then write my paper (`paper.rmd`), which could call each of those files as part of the building process.[2]

3.3 When to use R Markdown

R Markdown seems like it fills a strange niche, but it's useful for problem sets, short assignments, papers, and even dissertations. The R Markdown ecosystem contains packages for books and theses (`bookdown`), but also slide shows (`xaringan`), and even websites (`blogdown`).

I use `xaringan` and `blogdown` regularly. All workshop slideshows you have seen from me have been built with `xaringan` (except for the Beamer demonstration). My own personal website uses files that are created with `blogdown` (e.g. this blog post).

4 Content control with R Markdown

4.1 Examples using Pokémon data

To demonstrate content control features, we will use the same Pokémon data as we did during the LaTeX workflow demonstration. We will include R code along with the output to demonstrate the capabilities of R Markdown.

Let's start by attaching packages and reading the data into R.

```
# data wrangling packages
library("here")
library("magrittr")
library("tidyverse")
library("broom")

# table pkgs
library("knitr")
library("texreg")
```

[2]It would be smart to cache the results of each `.R` file, so I don't have to re-run every file each time I build my paper document.

```
# graphics
library("ggplot2")
theme_set(theme_minimal())

# read data
pm <- read_csv(here("data/pokemon.csv"))
```

4.2 Tables (with kable)

The following code creates a table of means, just as we did in the LaTeX workflow document. We will see the code and the resulting table.

```
mean_stats_tab <- pm %>%
  select(Generation, HP:Speed) %>%
  group_by(Generation) %>%
  summarize_all(mean) %>%
  mutate_if(is.numeric, round) %>%
  print()
```

```
## # A tibble: 6 x 7
##   Generation    HP Attack Defense `Sp. Atk` `Sp. Def` Speed
##        <dbl> <dbl>  <dbl>   <dbl>     <dbl>     <dbl> <dbl>
## 1          1    66     77      71        72        69    73
## 2          2    71     72      73        66        74    62
## 3          3    67     82      74        76        71    67
## 4          4    73     83      78        76        77    71
## 5          5    72     82      72        72        69    68
## 6          6    68     76      77        74        75    66
```

Once we have this data frame, we can turn it into a table using the `knitr::kable()` (which is to say, the `kable()` function from the `knitr` package for R). The following code creates Table 1. The `booktabs` option controls whether the table is created using "booktabs" style.

```
mean_stats_tab %>%
  kable(caption = "Mean Pok\\'emon Stats",
        booktabs = TRUE)
```

In order to reference Table 1, the table needs a `\label{}`. It turns out that the label is created by naming the R chunk in your `.rmd` file. We are then able to reference the label using `\@ref(tab:chunk-label)`. I use non-breaking spaces when I type a Table or Figure name, which prevents "Table" and "1" from printing on different lines. In the code, this uses

Table 1: Mean Pokémon Stats

Generation	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed
1	66	77	71	72	69	73
2	71	72	73	66	74	62
3	67	82	74	76	71	67
4	73	83	78	76	77	71
5	72	82	72	72	69	68
6	68	76	77	74	75	66

Table 2: Choices for your first Pokémon

Species	Type	Weakness	Gary Picks
Bulbasaur	Grass	Fire	Charmander
Squirtle	Water	Grass	Bulbasaur
Charmander	Fire	Water	Squirtle

the non-breaking space HTML character (`&nbsp;`) which looks a little ugly but gets the job done. If you open the source code, you will see what I mean.

In Table 2, we recreate the text-only table.

```
# create a data frame of pokemon info
starters <-
  data_frame(Species = c("Bulbasaur", "Squirtle", "Charmander"),
             Type = c("Grass", "Water", "Fire"),
             Weakness = c("Fire", "Grass", "Water"),
             `Gary Picks` = c("Charmander", "Bulbasaur", "Squirtle"))

# print using kable()
starters %>%
  kable(caption = "Choices for your first Pok\\'emon",
        booktabs = TRUE)
```

In Table 3, we show how to use the `texreg` package to print regression tables.

```
# estimate regressions
def_model <- lm(HP ~ Defense, data = pm)
atk_def_model <- lm(HP ~ Defense + Attack, data = pm)
```

```r
# regression table
texreg(list(def_model, atk_def_model),
       custom.model.names = c("Restricted Model", "Full Model"),
       caption.above = TRUE,
       caption = "Regressions predicting Pok\\'emon HP",
       label = "tab:hp-regs",
       booktabs = TRUE, use.packages = FALSE,
       float.pos = "ht")
```

Table 3: Regressions predicting Pokémon HP

	Restricted Model	Full Model
(Intercept)	54.77^{***}	40.77^{***}
	(2.26)	(2.46)
Defense	0.20^{***}	0.06
	(0.03)	(0.03)
Attack		0.31^{***}
		(0.03)
R^2	0.06	0.18
Adj. R^2	0.06	0.18
Num. obs.	800	800
RMSE	24.81	23.12

$^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$

4.3 Figures

We can also use R code chunks to include figures directly in our output. This is also great for reproducibility.

To demonstrate, we will plot the relationship between a Pokémon's attack points and its defense points. If some Pokémon are simply designed to be more capable than others, we might observe a positive relationship (Pokémon with high attack also have high defense). If the creators of Pokémon are interested in creating trade-offs between Pokémon traits, we might observe a negative relationship. The following code creates Figure 2.

```r
# jittered points for clarity of data
# add a diagonal reference line (abline)
# fit the linear relationship
```

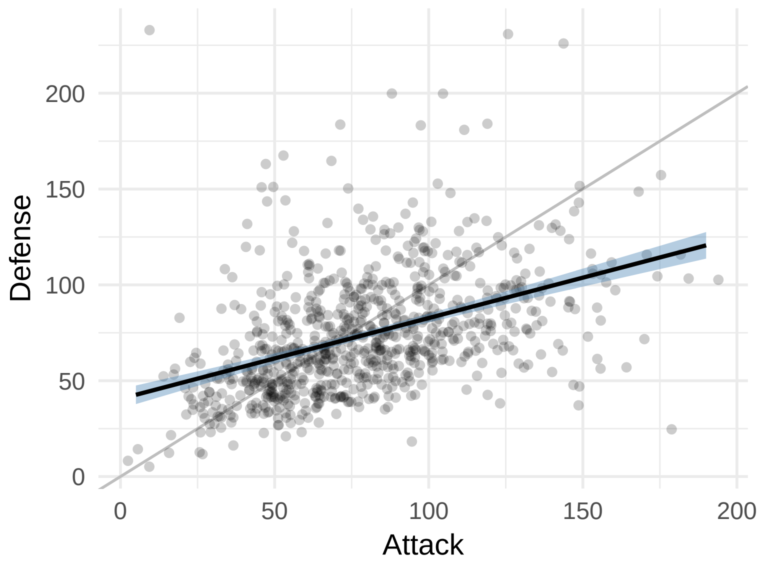


Figure 2: Relationship between Pokémon Attack and Defense points

```
ggplot(pm, aes(Attack, Defense)) +
  geom_abline(color = "gray") +
  geom_jitter(shape = 16, alpha = 0.2, width = 5, height = 5) +
  geom_smooth(method = "lm", size = 0.75,
              color = "black", fill = "steelblue")
```

4.4 Bibliographies

Bibliographies can be especially confusing because there are so many ways they can go wrong—with the R Markdown side, the Pandoc side, the LaTeX side, the YAML header, and so on. I spent a *long* time messing with the bibliography for this document—I may not have done the best job, but here is what I've got.

The most necessary component, naturally, is your `.bib` file. The highest-level of your YAML matter can take a `bibliography` variable. In our case, we have a `bib` folder that contains the file `rmd-bib.bib`. The `bib` folder is in the same folder as our `.rmd` file, so we need to navigate *into that folder* in order to find out `.bib` file.

```
bibliography: bib/rmd-bib.bib
```

Next, in order to create a bibliography in a LaTeX PDF, we need a package to manage the bibliography. We have to place this option underneath the output header in our YAML, since it modifies the LaTeX building process. We used `biblatex` for bibliographies before, so let's use it again here.

```
output:
  bookdown::pdf_document2:
    citation_package: biblatex
```

Lastly, we can set some parameters about how citations are formatted. By default, `biblatex` uses a numerical citation system. Political science uses an author-year system though, so we should change it globally.

```
biblio-style: authoryear
```

All of these options are visible in the document's YAML header.

4.4.1 Citations using Rmd's syntax

R Markdown's citation syntax uses a combination of the `@` symbol, the cite key, and square brackets.

If I wanted to do an in-text citation of Gelman and Stern (2006), I would write `@Gelman2006significance`.

If I wanted to do a parenthetical citation (Gelman and Stern 2006), I would write `[@Gelman2006significance]`.

If I wanted to cite this Gelman and Stern paper without the authors' names (2006), I could write `[-@Gelman2006significance]`.

5 Final words

R Markdown is neat, but because it is a Frankenstein's monster of software, it can be cumbersome at times. While it does combine statistical output with your writing, you still sometimes have to manage other files (`.yml` if you want to hide YAML variables in a separate file, `.csl` files for citation styles, templates if you want to use templates...).

It can also be confusing to manage cached R data. Sometimes the data that gets used in one part of the file seem "out-of-order" due to caching of previous output (read more here). For this reason, I advocate doing your complex, heavy-lifting code in `.R` files, leaving only the final output to be created in your `.rmd` file.

References

Gelman, Andrew and Hal Stern (Nov. 2006). "The Difference Between "Significant" and "Not Significant" is not Itself Statistically Significant". In: *The American Statistician* 60.4, pp. 328–331. DOI: 10.1198/000313006x152649. URL: https://doi.org/10.1198/000313006x152649.