Introduction to R Markdown

Jennifer Lin

2022-08-30

Contents

Overview of Markdwon

Throughout your life, you have likely written documents using software like Microsoft Word, Google Docs, and, for the Mac users among us, Pages. Each of these, in their unique, albeit similar, ways allows you to construct a document with proper formatting. When you use these, you likely click a button to do the various things you want, such as making text bold, changing font sizes, and so on. The software does the markup on the back end for you. While simple, these programs are just an overlay for markup languages like HTML, LaTeX and Markdown. In a markup language, you are writing in a plain text editor and you are in control of writing the code that allows things like bold text and font size changes to occur. Plain text simply means that the file contains text only, without any back end coding to enhance the layout and styles of the page. Of the three I mentioned, Markdown is perhaps the easiest to pick up. In this memo, I introduce Markdown markup and its implementation in R.

General Formatting

The conventions to format a markdown document is quite easy to learn in comparison to other markup languages. While there is a slight learning curve, once you get the hang of it, the syntax can come quite naturally, and you can apply this to other languages. In this section, I discuss some of the general formatting conventions that work for any markdown (`.md`) or R markdown (`.Rmd` – markdown that can take R code) files. In the next section, I will go into greater detail about R markdown and its specific components.

Bold, Italics and Others

Markdown can take headings, and you should make use of this to structure your document. Use the `#` to specify headings in your paper. The number of `#` denotes what level of heading you have.

```
# Heading 1

## Heading 2

### Heading 3
```

Within the text itself, markdown provides many simple ways to do text markup. Here are many of the things you will need. In the below section, I will present the same sentence twice, first as formatted text and then again as raw code. Formatted text comes in the serif font while code is in the typewriter font. This way, you can see the formatting that went into creating the outcome.

You can **bold** text by putting two `**` before and after the words you want to bold.

```
You can **bold** text by putting two `**` before and after
the words you want to bold.
```

You can *italicize* text by putting one `*` before and after the words you want to italicize.

```
You can *italicize* text by putting one `*` before and
after the words you want to italicize.
```

You can generate a number list by adding a `1.` and inserting your text. For example, below is my sample list:

1. Point 1
2. Point 2
3. Point 3

```
You can generate a number list by adding a `1.` and
inserting your text. For example, below is my sample list:

1. Point 1
2. Point 2
3. Point 3
```

You can generate a bulleted list by using a `-` in a new line followed by a space and then your text. For example, here is my grocery list for next week:

- Apples
- Broccoli
- Rice
- Bananas
- Yellow Onion
- Spinach

```
You can generate a bulleted list by using a `-` in a new
line followed by a space and then your text. For example,
here is my grocery list for next week:

- Apples
- Broccoli
- Rice
- Bananas
- Yellow Onion
- Spinach
```

How have I been generating the verbatim text that looks like code? You can use the back ticks to generate `inline code` or use three of them to give you a block of code

```
Code blocks look like this.
```

````
How have I been generating the verbatim text that looks
like code? You can use the backticks to generate `inline
code` or use three of them to give you a block of code

```
Code blocks look like this.
```
````

In markdown, you can also quote someone using > to insert a block quote. For example:

> "Sometimes I'll start a sentence and I don't even know where it's going. I just hope I find it along the way." – Michael Scott, Season 5, "The Duel"

```
In markdown, you can also quote someone using `>` to
insert a block quote. For example:

> "Sometimes I'll start a sentence and I don't even know
where it's going. I just hope I find it along the way." -
Michael Scott, Season 5, "The Duel"
```

You can insert hyperlinks using `[Google](https://www.google.com/)` where the square bracket contains the information for the text and the information in the parentheses is the URL. The Google link complies to Google.

To insert images, simply add a `!` beforehand. The text in the square brackets will serve as the caption and the information in the parentheses is the file path for the image of interest. You can control the size of the image using the `width=` in the curly brackets next to the image command. For example, here is a picture of a cute puppy.[1]

Figure 1: Cute Puppy

```
You can insert hyperlinks using
`[Google](https://www.google.com/)` where the square
```

[1]The dog in this picture belongs to my mentor when I interned at UW Madison. Isn't he the cutest? His name is Goguma, which means sweet potato in Korean because he looks like a sweet potato. And now that you are here, this is how you add a footnote in markdown. In the main text, insert a key in square brackets (`[^key]`) and somewhere below that text, insert the same key, followed by a colon, then insert the text of the footnote.

```
bracket contains the information for the text and the
information in the parentheses is the URL. The google link
complies to [Google](https://www.google.com/).

To insert images, simply add a `!` beforehand. The text in
the square brackets will serve as the caption and the
information in the parentheses is the file path for the
image of interest. You can control the size of the image
using the `width=` in the curly brackets next to the image
command. For example, here is a picture of a cute
puppy[^Goguma].

[^Goguma]: The dog in this picture belongs to my mentor
when I interned at UW Madison. Isn't he the cutest? His
name is Goguma, which means sweet potato in Korean because
he looks like a sweet potato. And now that you are here,
this is how you add a footnote in markdown. In the main
text, insert a key in square brackets (`[^key]`) and
somewhere below that text, insert the same key, followed
by a colon, then insert the text of the footnote.

![Cute Puppy](Images/Goguma.jpg){width=50%}
```

Finally, you can make tables in markdown, though they might not be like the way you do it in Word. Table cells are separated using `|`. The first line contains the table headings, the second contains the table justification and the rest are the table contents itself. Here is an example table. Text is justified in each column based on the column name. As you can see, you can format markdown text within table environments too.

Left	Center	Right
1	Red	40
2	Orange	25
3	*Blue*	46

```
Finally, you can make tables in markdown, though they
might not be like the way you do it in Word. Table cells
are separated using `|`. The first line contains the table
headings, the second contains the table justification and
the rest are the table contents itself. Here is an example
table. Text is justified in each column based on the
column name. As you can see, you can format markdown text
within table environments too.
```

```
|Left|Center|Right|
|:---|:-----:|----:|
|1|Red|40|
|2|Orange|25|
|**3**|*Blue*|`46`|
```

Math Language

A special part of markdown is its ability to support beautifully formatted math equations that are formatted similar to the way they are in LaTeX. The catch here is that you will need to have some familiarity with LaTeX syntax. Here is a great cheat sheet on math in LaTeX. In this section, I will show you some of the ones that are more frequently used through popular math equations. Like the previous section, I will format it to show both the final product and the code. This is by no way exhaustive, however. I am getting my inspirations from some of the equations featured in this Business Insider article on "The 17 Equations that Changed the World". You can reference math code easily by googling what you need.

Pythagorean Therom

Starting off simple, we have the Pythagorean Theorem. I include this to show you how to write math in a line of text rather than in a block. I also include this to show you how to do superscripts. The Pythagorean Theorem is $a^2 + b^2 = c^2$, or `$a^2 + b^2 = c^2$`. Math in text is enclosed by one dollar sign on each end of the equation.

The Fundimental Theorem of Calculus

Calculus is essential for understanding optimization, but I am not presenting this to show you how to optimize. I am presenting this equation to show you how to do

- Special math symbols such as the limit and the integral
- Fractions
- Math in an equation block (as opposed to inline)

```
$$
\frac{df}{dt} = \lim_{h \rightarrow 0}\frac{f(t + h) - f(t)}{h}
$$
```

$$\frac{df}{dt} = \lim_{h \rightarrow 0}\frac{f(t + h) - f(t)}{h}$$

```
$$
\int^{b}_{a} f(x) \ dx = F(b) - F(a)
$$
```

$$\int^{b}_{a} f(x) \ dx = F(b) - F(a)$$

As you can see, surrounding the equation with two dollar signs on either end creates equations that print in the center of the page.

From the examples here, we also learn that

- `\frac{numerator}{denominator}` is for fractions
- Superscript are introduced using a `^` and subscripts are introduced with `_`

The Normal Distribution

I am introducing the normal distribution equation to show you how to do

1. Exponents with complex formatting
2. Greek letters
3. Square roots
4. Fractions in exponents and in the main body

```
$$
\Phi(x) = \frac{1}{\sqrt{2\pi\sigma}}e^{\frac{(x - \mu)^{2}}{2\sigma^{2}}}
$$
```

$$\Phi(x) = \frac{1}{\sqrt{2\pi\sigma}}e^{\frac{(x-\mu)^2}{2\sigma^2}}$$

From this demonstration, here are a few takeaway points

- When it comes to complex math equations, it is *crucial* to keep your curly braces `{}` in order. Some text editors will open and close braces for you, but I would recommend double checking. If you make edits that involve deleting braces, be sure to delete the appropriate number so that each opener has a close.
- Greek letters in LaTeX are denoted by their name preceded by a backslash. Capital Greek letters have capital commands. For example, Δ (`$\Delta$`) and δ (`$\delta$`) are the Greek letter delta, with the former is the capital and the latter is the lowercase. Detexify (https://detexify.kirelabs.org/classify.html) is a good tool to help you find code too!

There are likely some things that you may need that I am not showing you here. Regardless of your current needs, know that the way you import math into a markdown document is the same way that you would do it in LaTeX.

R Markdown

R Markdown is a nifty package in R that allows you to compile markdown document with R code in R studio. Documents can be exported as HTML, PDF (tinytex or LaTeX distribution required because this is how markdown processes PDF files under the hood) or Word (Microsoft Word required). To use it, you need to install the package using `install.packages("rmarkdown")`. There is no need to explicitly load the package. You can open a new R Markdown document using File > New File > R Markdown on OSX.

Here is a walk through of the R Markdown interface in R Studio.

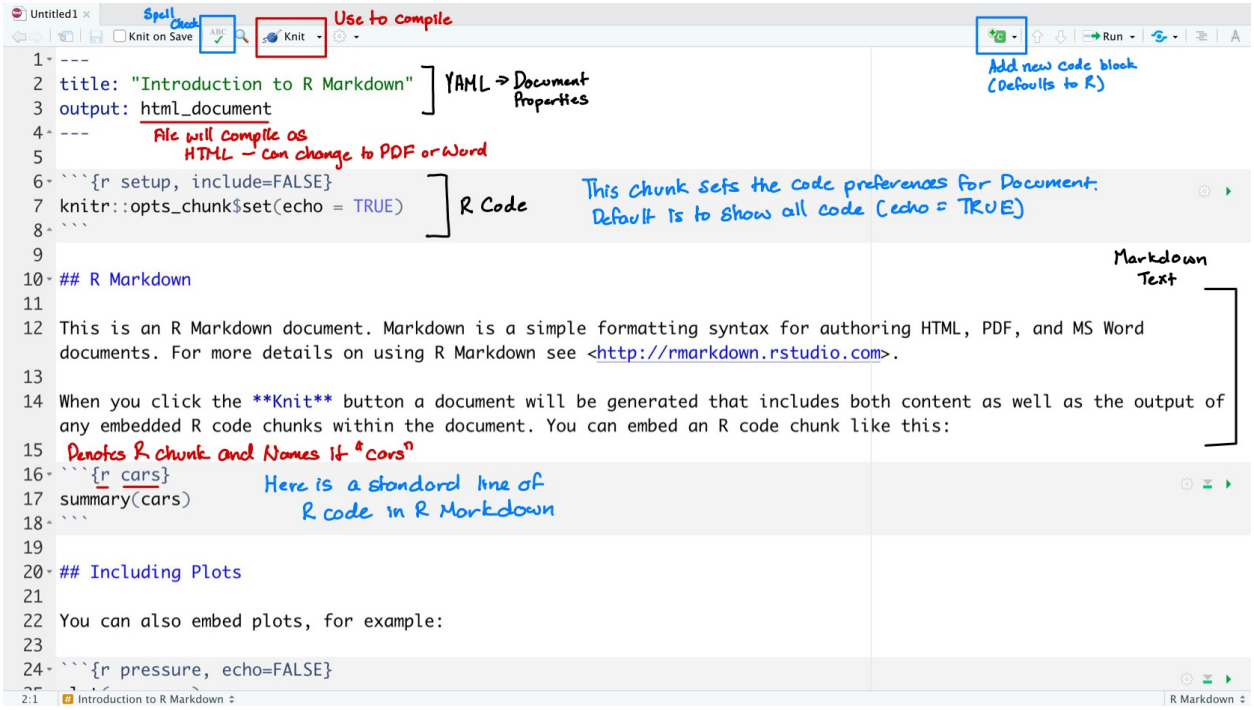


Figure 2: R Narkdown Interface in R Studio

Rendering R Code

The unique piece of R Markdown is the ability to use R code in text. To do this, you insert a code block with three tick marks, followed by `{r}`. For example:

```
```{r}
# Write some R Code here

```
```

On OSX, you can initiate this using Command+Option+I. Within these R chunks, you can write and run R code as you would in R Studio. While demonstrating that, I will also demonstrate some options that R chunks allow so you can control what outputs and codes are shown in the final product. Like the previous sections, I will include both he final product and the input.

Before writing code, you need to load some packages. However, some packages can load with warnings or messages. These are often ugly and unnecessary. Therefore, we can suppress them using `warning=` and `message=` and set both to FALSE to turn off messages. As you can see, the code still shows in the output but not the messages.

```
library(dplyr)
library(ggplot2)
library(tidyr)
```

```{r, warning=FALSE, message=FALSE}
library(dplyr)
library(ggplot2)
library(tidyr)
```

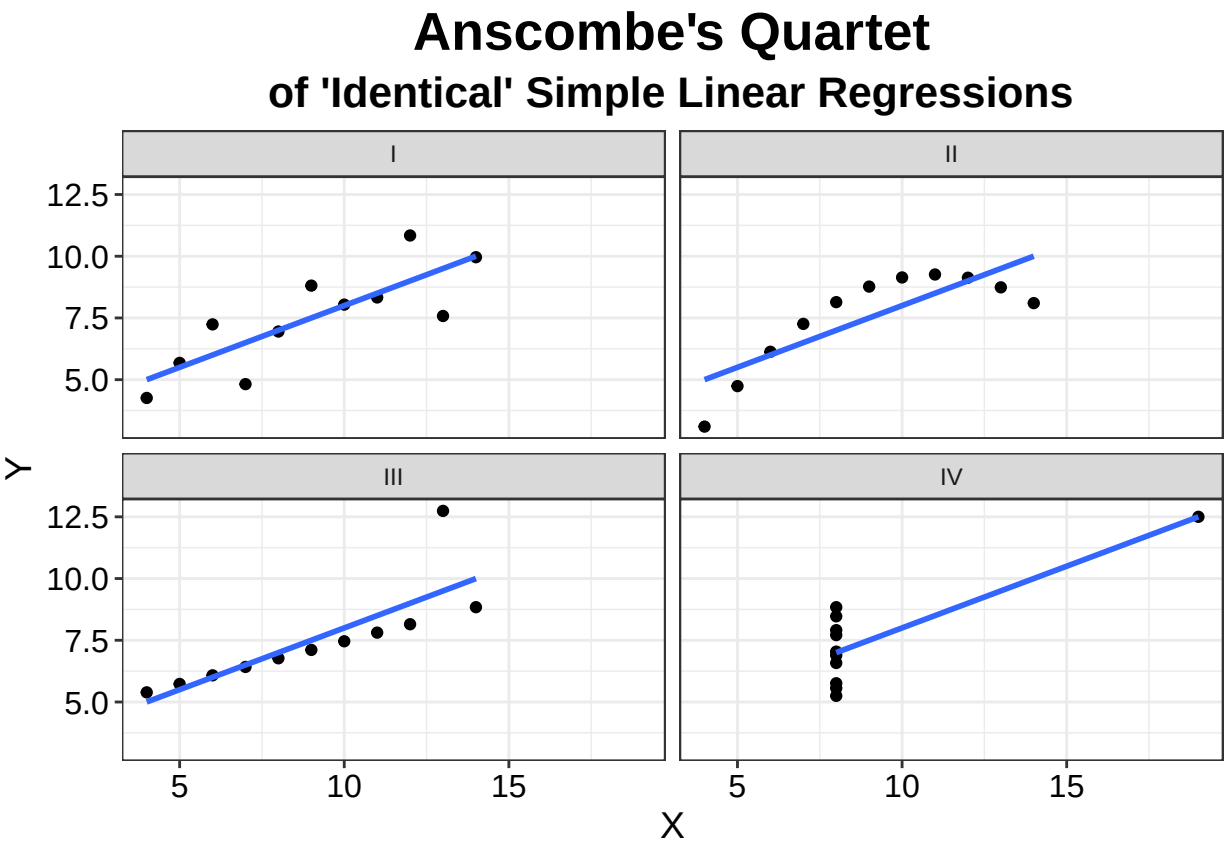
For the purposes of demonstration, I will use the anscombe data set, which is preloaded in base R. This dataset comes from the "Anscombe's Quartet of 'Identical' Simple Linear Regressions" where there are distinct groups of numbers that will all produce the same predictors in a linear regression model. We can simply call the data as follows:

```
data("anscombe")
```

```{r}
data("anscombe")
```

Note that since this step does not produce warnings or messages, we do not need to specify that warnings and messages should be set to FALSE.

Now, let's plot the data to the one that we know in the textbooks.



How did I get the graph up on the page? Where is the code? You can hide the code using `echo = FALSE`, which tells R to evaluate the code but not include the script. Here is the

script that I used.

```
```{r, echo=FALSE, warning=FALSE, message=FALSE}
anscombe_tidy <- anscombe %>%
    mutate(observation = seq_len(n())) %>%
    gather(key, value, -observation) %>%
    separate(key, c("variable", "set"), 1, convert = TRUE) %>%
    mutate(set = c("I", "II", "III", "IV")[set]) %>%
    spread(variable, value)

ggplot(anscombe_tidy, aes(x, y)) +
  geom_point() +
  facet_wrap(~ set) +
  geom_smooth(method = "lm", se = FALSE)+
  xlab("X")+
  ylab("Y")+
  labs(
    title = "Anscombe's Quartet",
    subtitle = "of 'Identical' Simple Linear Regressions"
  )+
  theme_bw()+
  theme(
    plot.title         = element_text(
      hjust = 0.5, size = 20, colour="black", face = "bold"),
    plot.subtitle      = element_text(
      hjust = 0.5, size = 16, colour="black", face = "bold"),
    legend.title       = element_text(
      hjust = 0.5, size = 14, colour="black", face = "bold"),
    plot.caption       = element_text(
      size = 10, colour="black"),
    axis.title         = element_text(
      size = 14, colour="black"),
    axis.text.x        = element_text(
      size = 12, colour="black", angle = 0, hjust = 0.5),
    axis.text.y        = element_text(
      size = 12, colour="black"),
    legend.position    = 'bottom',
    legend.direction   = "horizontal",
    legend.text        = element_text(
      size = 12, colour="black")
  )
```
```

You can also include code without evaluating it. To do this, simply put `eval = FALSE` in the chunk header. Here, I include code to calculate the mean of a variable in the anscombe dataset, but I am not evaluating it

```
mean(anscombe$x1)
```

````
```{r, eval = FALSE}
mean(anscombe$x1)
```
````

YAML Features

In every R markdown document, there is a YAML on the top that specifies document features. You can include this in a standard markdown document too. All it does is tell your computer how to render certain data features and in the case of R markdown, how to export your file once you compile it. Here, I will show you some typical YAML features but be aware that these might differ slightly from document to document. If you are using a document template from a package (See next section for more details), consult the template YAML to see what specifications are available for that template. You are always welcome to add more fields as needed.

Here is the YAML section of this document.

```
---
title: "Introduction to R Markdown"
author: "Jennifer Lin"
date: "2022-08-30"
output:
  pdf_document:
    toc: true
    toc_depth: 2
fontsize: 12pt
urlcolor: blue
---
```

As you can see, it starts end ends with three dashes (`---`). Below, I go through this line by line to show you what each line does.

- `title: "Introduction to R Markdown"`: This is the title of the document. You can add a subtitle using `subtitle:`
- `author: "Jennifer Lin"`: This is where you include the author name.
- `date: "2022-08-30"`: Include the date – this can be specified as a fixed date or have R insert today's date using `"2022-08-30"`.
- `output:` Specifies the output of the document. Here, if the document has further specifications, *indentations are important*. Here, I am having it compile to a PDF (indent once) with table of content specifications (each indent twice). `toc: true` specifies that I want a table of contents and `toc_depth: 2` tells it to include Headings 1 and 2 only.
- `fontsize: 12pt`: 12 point font for the main text
- `urlcolor: blue`: Text that has a URL/Hyperlink will be blue

When you are ready to compile your R markdown documents, simply click the "Knit" button at the top of the screen and it will compile the document in the format that you choose. This requires the `knitr` package, which should be installed with your `rmarkdown` package. `knitr` compiles R packages using a software known as `pandoc`, which can allow you to convert almost any document from one form to another using your computer's command line terminal (See more about `pandoc` here).

R Markdown Extensions and Possibilities

R markdown is the foundation for a suite of tools that build off of its coding parameters which can allow you to develop your academic work. Here are just a few extensions that have been developed, which build off the R markdown model.

1. `bookdown` allows you to format books. You can export them as PDF or HTML.
2. `pagedown` allows you to convert HTML output to PDF for easy distribution – also includes templates for various HTML documents, including letters and resumes
3. `blogdown` allows you to create blogs or websites
4. `hugodown` is a developmental package that allows you to integrate the `hugo` system with your `blogdown` sites
5. `xaringan` creates HTML slides
6. `rticles` contains journal templates so you can write your submission in markdown rather than LaTeX.
7. `tufte` generates Tufte style handouts and materials
8. `flexdashboard` creates interactive dashboards. It is a streamlined Shiny App package
9. `pkgdown` builds websites for R packages
10. `prettydoc` includes extension R markdown document themes
11.

There is a lot more that R Markdown can handle. You can also find ways to integrate your own JavaScript and CSS styles to customize your work. It is quite versatile in this way.

Additional Resources

Here are some additional resources for learning R Markdown.

- The R Markdown Codebook
- R Markdown Cheat Sheet