Introduction to R Markdown

You will be submitting your homework in the form of an R Markdown file (.Rmd) and a knitted HTML file (.html). This tutorial will help you navigate the R Markdown platform.

## Markdown

Markdown is a simplified version of ‘markup’ languages. Unlike cumbersome word processing applications, text written in Markdown uses simple and intuitve formatting elements and can be easily shared between computers, mobile phones, and people. It’s quickly becoming the writing standard for academics, scientists, writers, and many more. Websites like GitHub and reddit use Markdown to style their comments.

Formatting text in Markdown has a very gentle learning curve. It doesn’t do anything fancy like change the font size, color, or type - just the essentials, using keyboard symbols you already know. All you have control over is the display of the text-stuff like making things bold, creating headers, and organizing lists.

We won’t go into a lot of detail, but run through a few basic examples. For more details and examples, click [here](https://www.markdowntutorial.com/).

### Italics and Bold

To make a phrase italic in Markdown, you can surround words with a single underscore (\_) or asterisk (\*). For example, *this* word would become italic.

Similarly, to make phrases bold in Markdown, you can surround words with two underscores (\_\_) or two asterisks ( \*\* ). This will **really** get your point across.

Most commonly, a single underscore is used for italics and two asterisks are used for bold.

Of course, you can use ***both*** italics and bold in the same line by adding both an underscore and asterisks (\*\*\_). You can also span them ***across multiple words***.

In general, it doesn’t matter which order you place the asterisks or underscores. I prefer to place the asterisks on the outside to make it ***easier to read***.

### Headers

Headers are frequently used on websites, magazine articles, and notices, to draw attention to a section. As their name implies, they act like titles or subtitles above sections.

There are six types of headers, in decreasing sizes:

# Header one

## Header two

### Header three

#### Header four

##### Header five

###### Header six

To make headers in Markdown, you preface the phrase with a hash mark (#). You place the same number of hash marks as the size of the header you want. For example, for a header one, you’d use one hash mark (# Header One), while for a header three, you’d use three (### Header Three). It’s up to you to decide when it’s appropriate to use which header. In general, headers one and six should be used sparingly.

### Links

There are two different link types in Markdown, but both of them render the exact same way. The first link style is called an *inline* link. To create an inline link, you wrap the link text in brackets ([ ]), and then you wrap the link in parenthesis (( )). For example, to create a hyperlink to www.github.com, with a link text that says, Visit GitHub!, you’d write this in Markdown: [Visit GitHub!](https://github.com/).

The other link type is called a reference link. As the name implies, the link is actually a reference to another place in the document. Here’s an example:

Here’s [a link to something else](https://github.com/).  
Here’s [yet another link](https://www.google.com/).  
And now back to [the first link](https://github.com/).

The “references” above are the second set of brackets: [another place](https://github.com/) and [another link](https://www.google.com/). At the bottom of a Markdown document, these brackets are defined as proper links to outside websites. An advantage of the reference link style is that multiple links to the same place only need to be updated once. For example, if we decide to make all of the [another place](https://github.com/) links go somewhere else, we only have to change the single reference link.

Reference links don’t appear in the rendered Markdown. You define them by providing the same tag name wrapped in brackets, followed by a colon, followed by the link.

### Images

If you know how to create links in Markdown, you can create images, too. The syntax is nearly the same.

Images also have two styles, just like links, and both of them render the exact same way. The difference between links and images is that images are prefaced with an exclamation point (!).

The first image style is called an *inline image link*. To create an inline image link, enter an exclamation point (!), wrap the alt text in brackets ([ ]), and then wrap the link in parenthesis (( )). (Alt text is a phrase or sentence that describes the image for the visually impaired.)

For example, to create an inline image link to <https://octodex.github.com/images/bannekat.png>, with an alt text that says, Benjamin Bannekat, you’d write this in Markdown: 

Although you don’t *need* to add alt text, it will make your content accessible to your audience, including people who are visually impaired, use screen readers, or do not have high speed internet connections.

For a reference image, you’ll follow the same pattern as a reference link. You’ll precede the Markdown with an exclamation point, then provide two brackets for the alt text, and then two more for the image tag. At the bottom of your Markdown page, you’ll define an image for the tag.

Here’s an example: the first reference tag is called “First Father”, and links to <http://octodex.github.com/images/founding-father.jpg>; the second image links out to <http://octodex.github.com/images/foundingfather_v2.png>.



The first father



The second first father

### Lists

There are two types of lists in the known universe: unordered and ordered. That’s a fancy way of saying that there are lists with bullet points, and lists with numbers.

To create an unordered list, you’ll want to preface each item in the list with an asterisk (\*). Each list item also gets its own line. For example, a grocery list in Markdown might look like this:

* Milk
* Eggs
* Salmon
* Butter

An ordered list is prefaced with numbers, instead of asterisks. For example:

1. Crack three eggs over a bowl
2. Pour a gallon of milk into the bowl
3. Rub the salmon vigorously with butter
4. Drop the salmon into the egg-milk bowl

Easy, right? It’s just like you’d expect a list to look.

## knitr

knitr is an R package that is used for *statistical literate programming*, meaning you are able to integrate code and text in a single, simple document format. It supports R Markdown, R LaTex, and R HTML as documentation languages, and can export markdown, PDF and HTML documents.

Something that is really nice about using R Markdown with knitr is that your final document will not be created if there is an error in your code. Thus, it’s really easy to check if your code is running by simply knitting the .Rmd file.

Make sure to install the latest version of the knitr package: install.packages("knitr").

## rmarkdown

Documents, like this one, containing both R code (below) and markdown are R Markdown files. The similarly named rmarkdown is an R package that makes working with R Markdown easier by wrapping knitr along with other tools. For more details on both the R Markdown format and the rmarkdown package, see <http://rmarkdown.rstudio.com>.

When you run render from the rmarkdown package (or click the **Knit** button above), the .Rmd file is fed to knitr, which executes all of the code chunks and creates a new markdown (.md) document which includes the code and it’s output.

The markdown file generated by knitr is then processed by pandoc which is responsible for creating the finished format.



### Embedding Code

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R ‘code chunks’ within the document. We **ALWAYS** want to see your code and output. If you do not provide the code and output in the .Rmd file, you will lose points. In addition to the .Rmd file, we expect you to provide a knitted .html file in your GitHub homework repository. We will also knit your .Rmd file on our own computers while grading, to make sure we can reproduce your results.

#### Code Chunks

There are 3 ways to create a code chunk:

1. Typing ```{r} to start a code chunk and then ``` to end a code chunk
2. The keyboard shortcut **Ctrl + Alt + I** (OS X: **Cmd + Option + I**)
3. Clicking the Add Chunk button in the tool bar

Let’s start with the built-in data set pressure, which includes data on the vapor pressure of Mercury as a function of temperature. You can embed an R code chunk like this:

head(pressure)

## temperature pressure  
## 1 0 0.0002  
## 2 20 0.0012  
## 3 40 0.0060  
## 4 60 0.0300  
## 5 80 0.0900  
## 6 100 0.2700

summary(pressure)

## temperature pressure   
## Min. : 0 Min. : 0.0002   
## 1st Qu.: 90 1st Qu.: 0.1800   
## Median :180 Median : 8.8000   
## Mean :180 Mean :124.3367   
## 3rd Qu.:270 3rd Qu.:126.5000   
## Max. :360 Max. :806.0000

Note that both the code written and output produced are shown in the final document.

### Plots

**Assessment**: create a code chunk that makes a scatterplot of temperature and pressure. When you are done, knit your file and look at the ouput.

### Inline Text Computations

You can also include the otput of your code in the middle of a sentence. For example, I want to randomly generate two numbers, x and y, and include their values in a sentence. I would first write the following code chunk:

x <- rnorm(1)  
y <- rnorm(1)

Then I can write x = 2.1614889 and y = 1.3853053.

### Equations

If you know latex, including equations is really simple. The same syntax is used. For example, you can write an *inline* equation like this - . You can also center an equation like this:

### The cache option

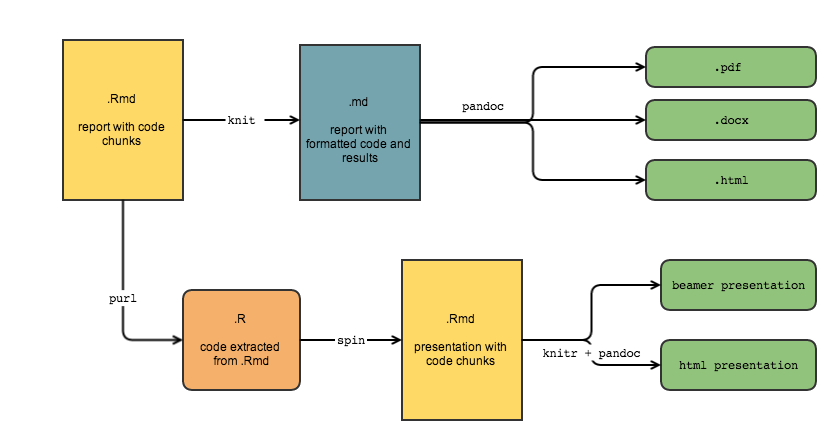
All code chunks have to be re-computed every time you re-knit the file. If you have code chunks that take a while to process, you may want to use the cache = TRUE option which stores and then loads the results from cache after the first run, and can save you considerable time. This can be done on a chunk-by-chunk basis. However, this is only useful if you have code chunks you haven’t edited since the first run. If the data or code changes, you will have to re-run the code to update the results

### Extracting R code

In knitr, you can use purl() to pull out all of the R code and put it into a single .R file. This wiil ignore all prose outside of code chunks. The following code will create a file called intro-to-rmarkdown.R in the same directory I’m working in.

library(knitr)  
purl("intro-to-rmarkdown.Rmd", documentation = 0)

### Summary



### Other Awesome Powers

R Markdown can render PDF presentations with beamer, HTML presentations with ioslides, slidy and reveal.js. It can also be used to write full academic manuscripts. You can also build your own websites and interactive documents. These powers are too complicated and time consuming for this lecture, but at least you know they’re possible!