#Notes for RMarkdown

By default we start in "Markdown" mode. Markdown lets us structure documents:

Title

Main Section

Subsection

We can add various inline styling: *italics*, **bold**, `monospace fonts`.

- Lists/bullets
- links
- Quotes

> This is a quoted section.

```
Large verbatim
  bits of text
    can be formatted
      using backticks.
```

Math Notation

There is also a built in system for writing mathematical expressions. For example, x^2.

You can also write "display" math like:

$$f(x) = \sin(\theta x y)$$

or

$$g(x, y) = \frac{x + y}{2}$$

Various special symbols and operators have a backslash notation. Curly braces are used instead of parethenses to indicate arguments.

$$\int_{-\infty}^{\infty} f(x)\, dx = 1$$

This languae is also known at LaTeX ("lah-tech") math notation. A really useful website is Detexify (only use "mathmode" symbols).

Often times you want to have several lines that connect together. You can use the `aligned` environment:

$$\begin{aligned} f(x) &= x^2 + 2xy + y^2 \\ &= (x + y)^2 \end{aligned}$$

There are some additional references in this week's homework. Piazza will be a great place to share tips and tricks.

Integrating R

When "knit", RMarkdown will process all the R code included and insert the results into the document (processes from top to bottom, errors stop processing.)

To start an R code block add `{r}` after three backticks:

```
# this is a comment in R, will be ignored by R
print("hello")
```

```
## [1] "hello"
```

You can also run code in your interpreter using Cmd-Return/Alt-Return.

Expressions are evaluated and results printed out by default:

```
7 + 2
```

```
## [1] 9
```

```
mean(rnorm(10))
```

```
## [1] 0.08395415
```

It is also possible to give additional instructions to the R "chunk":

```
stop("This would halt the knitting if it ran")
```

But notice it does display.

We can selectively hide the output of chunks, while keeping the code.

```
2 + 3 # will not display
```

Sometimes you want to suppress the code itself (`echo = FALSE`).

```
## [1] "Surprise!"
```

To combine both hiding code and output (but still executing):

```
print(hidden_var)
```

```
## [1] 7
```

For long calculations it can be useful to set `cache = TRUE` (just be careful if the cached chunk depends on other code or data that changes!)

```
sum(1:1e10)
```

```
## [1] 5e+19
```

Inline results can be included using (backtick)r EXP (backtick). For example the sum of 3 and 4 is 7.

```
estimate_of_mean <- 87.2
```

We estimate the mean as 87.2.

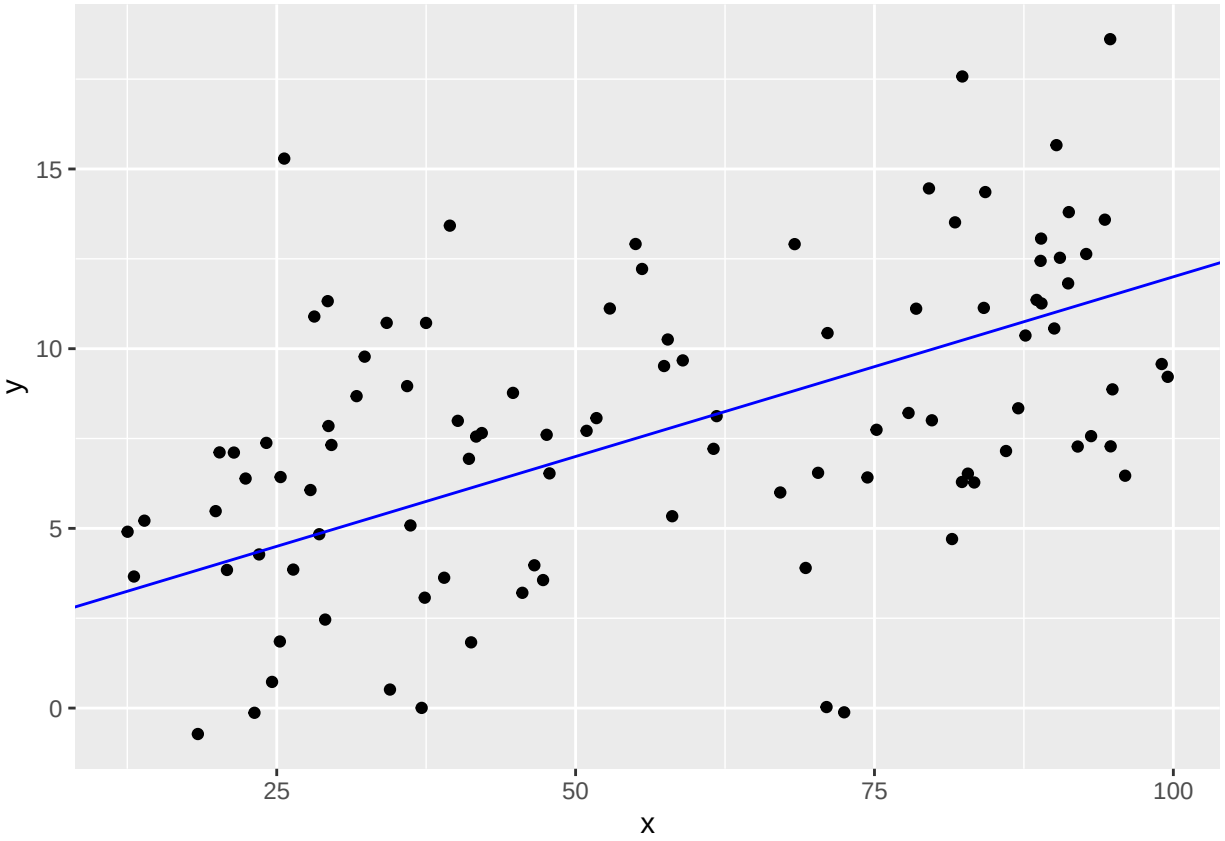
Plotting

Let's simulate the class linear model.

```
n <- 100 # number of units in the study
x <- runif(100, 10, 100)
y <- 2 + 0.1 * x + rnorm(n, mean = 0, sd = 4) ## generates
xy <- data.frame(x = x,  y = y) # column "x" gets values in variable x
```

We'll point you to the `ggplot2` library. General strategy, map your columns to names (such as "x" and "y") and then add additional elements to the plot.

```
library(ggplot2)
ggplot(data = xy, aes(x = x, y = y)) + geom_point() + geom_abline(intercept = 2, slope = 0.1, color = "
```



Sometimes tables are more helpful than plots. The `knitr` package provides a nice table formatting function:

```
library(knitr)
kable(xy[1:5, ]) # first 5 rows of the xy table
```

x	y
19.89390	5.480209
90.22203	15.663096
41.25918	1.828875
70.98973	0.028765
77.85141	8.211288

RStudio Tools

Getting Help

You can search the help in the left hand pane or interactively using

```
?t.test
??"Student"
```

Several useful "Cheatsheets" under `Help` menu.

External Packages

We will frequently use the following packages:

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.2      v forcats 0.5.2
## v purrr   0.3.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Other packages in the `Packages` pane.

Sessions and Environment

The `Console` pane contains a live R session. Session menu can

- Restart the session
- Set the working directory (makes it easier to load and save files)

The `Environment` pane shows useful information about the current session.

Example:

```
random_letters <- sample(letters, size = 20, replace = TRUE)
```

Debugging

If you RMarkdown document fails on a given R chunk, it can be useful to `Run all chunks above` then step through the code:

```
xy <- mutate(xy, z = x + y) # this depends on xy, so we need to get that from a previous chunk
mean_x <- mean(xy$x)
```

R has a built in debugger as well, but this only works in pure .R scripts, not RMarkdown documents.