

# DATA607 - Assign2

Rachel Greenlee

9/4/2020

## Introduction

For this assignment I went through the following steps in order to collect data using Google Forms and ultimately have a small dataframe in R with my friends' movie ratings ready for further analysis.

## Step 1 - Data Collection

I used Google Forms to set up an informal survey with just 1 matrix question listing all movies that I could send to my 5 friends. You can see that survey at this link: [Movie Survey](#)

## Step 2 - Export to CSV and Import into MySQL

Google Forms conveniently provides an export to CSV option, which I downloaded to my local files. In MySQL Workbench I was able to write code to create the empty table and load the data from the CSV file. Finally, I added an ID as a primary key. A screenshot of the code from my MySQL session is below. (I also learned how to put an image in RMarkdown!)


A screenshot of a MySQL code editor window showing SQL commands. The code is numbered 1 through 18. It starts with a 'CREATE TABLE' statement for a table named 'movies' with columns: 'name', 'crazyrichasians', 'moana', 'bladerunner2049', 'blackpanter', 'arrival', and 'lalaland', all of type 'VARCHAR(255)'. This is followed by a 'LOAD DATA INFILE' statement to load data from a CSV file located at 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/MySQL Local File Uploads/moviequizresponses.csv'. The load statement specifies fields terminated by commas, enclosed by double quotes, and lines terminated by newlines. It also includes 'IGNORE 1 ROWS;'. The final command is 'ALTER TABLE movies ADD id INT NOT NULL AUTO\_INCREMENT PRIMARY KEY'.

Figure 1: Image of MySQL Code.

### Step 3 - Connect R to MySQL to create dataframe

First I load the RMySQL package in order to connect with MySQL. I also load the keyring package so I can store my password and hide it from the world. I won't show the code I've used to store them, but it looks like this:

```
key_set_with_value(service = "mysql", username = "root", password = "XXXXXX")
```

```
## Loading required package: DBI
```

Now we must access the dataset from MySQL. And let's preview it quick as well.

```
mysqlpass <- key_get('mysql', 'root')

rachdb = dbConnect(MySQL(), user='root', password=mysqlpass,
  dbname='607assign2_movies', host='localhost')

movies <- dbGetQuery(rachdb, "select * from movies")

movies
```

```
##      name crazyrichasians moana bladerunner2049 blackpanter  arrival  lalaland
## 1    Ben      Not seen      2          5          4          4          3\r
## 2   Suman          4        3      Not seen      5 Not seen      2\r
## 3 Daniela      Not seen      4      Not seen      5 Not seen      3\r
## 4    Kate      Not seen      5      Not seen      5 Not seen Not seen\r
## 5   José          2        2          3      5 Not seen Not seen\r
##   id
## 1  1
## 2  2
## 3  3
## 4  4
## 5  5
```

### Step 4 - Cleaning the Dataframe

I see my movie ratings imported as characters, lets change them to numerics. In doing so, the "Not seen" value will render a NA - which is just fine for our purposes.

```
movies$crazyrichasians <- as.numeric(movies$crazyrichasians)
movies$moana <- as.numeric(movies$moana)
movies$bladerunner2049 <- as.numeric(movies$bladerunner2049)
movies$blackpanter <- as.numeric(movies$blackpanter)
movies$arrival <- as.numeric(movies$arrival)
movies$lalaland <- as.numeric(movies$lalaland)
```

Next lets put the ID field first.

```
movies <- movies[, c(8, 1, 2, 3, 4, 5, 6, 7)]
movies
```

##	id	name	crazyrichasians	moana	bladerunner2049	blackpanter	arrival	lalaland
## 1	1	Ben	NA	2	5	4	4	3
## 2	2	Suman	4	3	NA	5	NA	2
## 3	3	Daniela	NA	4	NA	5	NA	3
## 4	4	Kate	NA	5	NA	5	NA	NA
## 5	5	José	2	2	3	5	NA	NA

## Step 5 - Analysis

*Is there a movie that you would recommend or not recommend to one of the participants? Explain your reasoning.* For my two friends, José and Kate, who have not watched (NA) La La Land - I would recommend they do not watch it in the future as my first three friends all rated it only a 2-3, despite there being some, at first glance, variability in their ratings of the other movies. Also, Black Panther appears to be a universal favorite of my friends with everyone having seen the movie, and 4/5 ratings being a perfect 5!

To manipulate this dataframe a bit further and try to find something meaningful to plot from such a small dataset, I've made a new dataframe that just displays the minimum and maximum values for each of the 6 movies after dropping the first two irrelevant rows (ID and name).

```
maxs <- sapply(movies, max, na.rm = TRUE)
mins <- sapply(movies, min, na.rm = TRUE)

movieminmax <- data.frame(mins, maxs, stringsAsFactors = FALSE)

movieminmax[-c(1, 2),]
```

##		mins	maxs
##	crazyrichasians	2	4
##	moana	2	5
##	bladerunner2049	3	5
##	blackpanter	4	5
##	arrival	4	4
##	lalaland	2	3

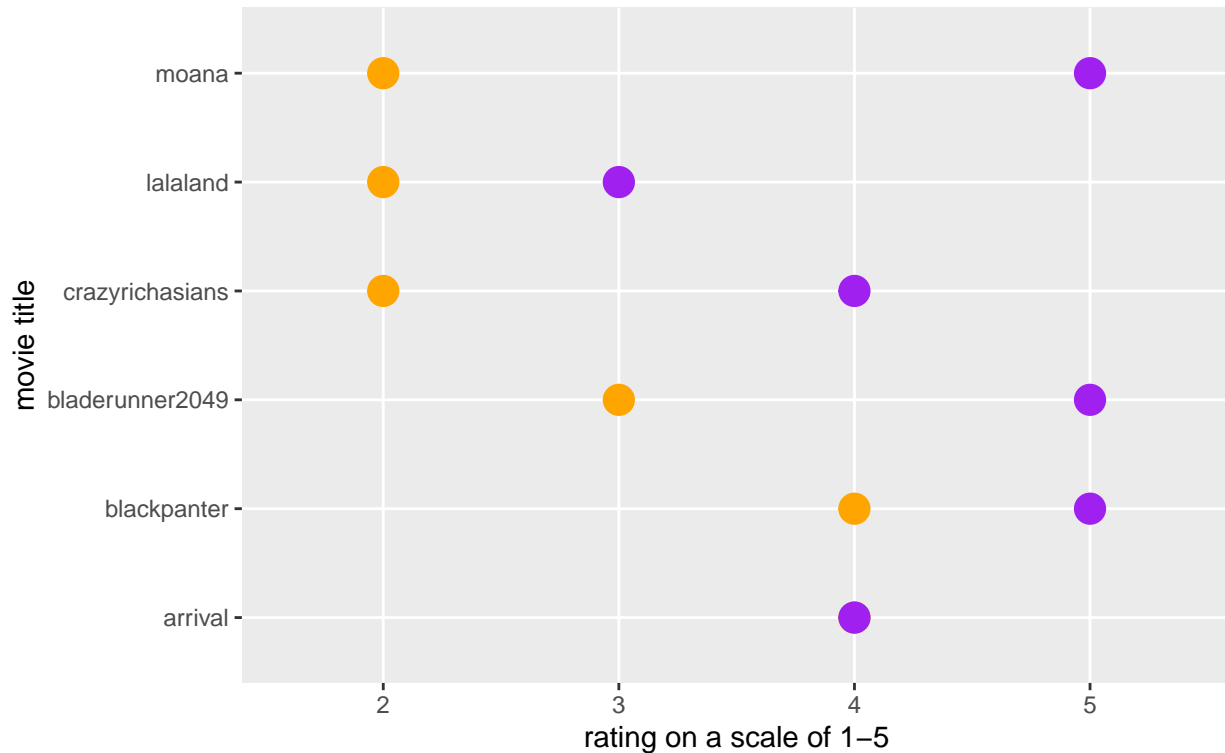
```
movieminmax <- movieminmax[-c(1, 2),]
```

I then used the ggplot2 package to make a simple plot that shows each movie with its minimum rating (orange dot) and maximum rating (purple dot). This isn't a rigorous analysis, but does make it easier to see that Moana has quite a spread of ratings compared to Black Panther. This graph hides the fact that some of these movies only had 1 or 2 raters.

```
library(ggplot2)
library(scales)
ggplot(movieminmax) +
  geom_point(aes(x=rownames(movieminmax), y=mins), color = "orange", size=5) +
  geom_point(aes(x=rownames(movieminmax), y=maxs), color = "purple", size=5) +
  coord_flip() +
  ylab('rating on a scale of 1-5') +
  xlab('movie title') +
  labs(title="Minimum and Maximum Ratings by Movie",
       subtitle="Note: Arrival movie only had one rating, displayed as a max here")
```

## Minimum and Maximum Ratings by Movie

Note: Arrival movie only had one rating, displayed as a max here



## Conclusion

This process shows how, with very little code, a simple survey done on a free platform like Google Forms can be brought into MySQL and then accessed by R Studio. Along the way I learned how to use the keyring package to store my passwords securely, how to display an image in R Markdown, a new way to use ggplot2, and most importantly how to access a MySQL database from R Studio.

Of course, if you wanted to expand this further, you could incorporate variables such as movie genres or main actors (Ryan Gosling is in both Blade Runner 2049 and La La Land, but they are very different genres).

# Assignment Week 1

rachelgreenlee

8/29/2020

## Overview

In June of 2020 FiveThirtyEight published an article discussing how voter registration started out strong in early 2020, but dropped dramatically once COVID hit. This data set compares 2016 and 2020 voter registration for January through April or May (depending on locale) in 11 states and Washington DC. FiveThirtyEight obtained the data from the Center for Innovation and Research.

Link to the article: <https://fivethirtyeight.com/features/voter-registrations-are-way-way-down-during-the-pandemic/>

## The Work

First I import the dataset that I've uploaded to my GitHub repository. These variables are well-named, but I'll rename two for practice.

```
reg_data <- read.csv(url("https://raw.githubusercontent.com/rachel-greenlee/538voter_registration/master/reg_data.csv"))
colnames(reg_data) <- c("State", "Year", "Month", "New Reg Voters")
colClasses = c("factor", "factor", "factor", "numeric")
```

Lets also write out the month names instead of the abbreviations for practice. At first I got some errors using this code, but then realized it was because these imported as factors after using the class function to check, so I switched the class to character and then could rename the columns. Here is a peek at the first few rows with these changes.

```
reg_data$Month <- as.character(reg_data$Month)
class(reg_data$Month)
```

```
## [1] "character"
```

```
reg_data$Month[reg_data$Month == "Jan"] <- "January"
reg_data$Month[reg_data$Month == "Feb"] <- "February"
reg_data$Month[reg_data$Month == "Mar"] <- "March"
reg_data$Month[reg_data$Month == "Apr"] <- "April"

head(reg_data)
```

```
##      State Year      Month New Reg Voters
## 1 Arizona 2016   January      25852
## 2 Arizona 2016  February      51155
## 3 Arizona 2016    March      48614
## 4 Arizona 2016    April      30668
## 5 Arizona 2020   January      33229
## 6 Arizona 2020  February      50853
```

Since there are only 4 variables, I don't need to select a subset of columns as the data would be pointless without all 4, but for practice let's remove the State variable and create the "sillysubset".

```
sillysubset <- subset(reg_data, select = c("Year", "Month", "New Reg Voters"))
head(sillysubset)
```

```
##      Year      Month New Reg Voters
## 1 2016   January      25852
## 2 2016  February      51155
## 3 2016    March      48614
## 4 2016    April      30668
## 5 2020   January      33229
## 6 2020  February      50853
```

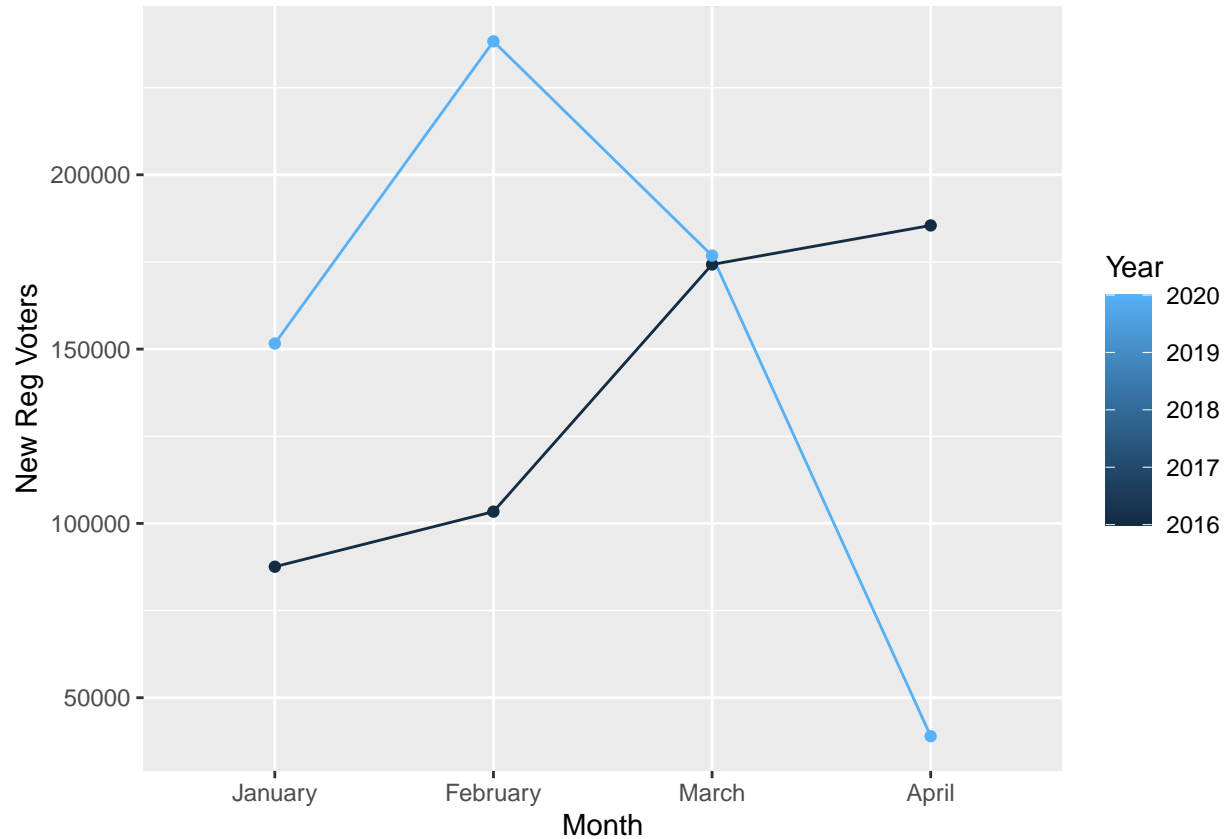
Just to see if I could figure it out, I did a California only subset of the data and graphed it with 2 lines, one representing each year. I used ggplot2. I had to figure out how to put the months in order, which I found I could do by setting the levels of these factors. On the graph, you can see with the light blue line that in 2020 voter registration drops dramatically below 2016 from February to April.

```
reg_dataCA2020 <- subset(reg_data, State == "California")
head(reg_dataCA2020)
```

```
##      State Year      Month New Reg Voters
## 9 California 2016   January      87574
## 10 California 2016  February     103377
## 11 California 2016    March     174278
## 12 California 2016    April     185478
## 13 California 2020   January     151595
## 14 California 2020  February     238281
```

```
reg_dataCA2020$Month <- factor(reg_dataCA2020$Month, levels = c("January", "February", "March", "April"))

ggplot(data=reg_dataCA2020, aes(x=Month, y='New Reg Voters', group=Year)) +
  geom_line(aes(color=Year)) +
  geom_point(aes(color=Year))
```



## Conclusions

When this article was published in June I'm sure the April/May data was the most recent available, but now that we are much closer to the election it would be a natural additions to add on the New Voter Registration numbers for these same states up until July/August if possible. Expanding to include more states could also be beneficial.

Considering that COVID is still limiting many traditional voter registration efforts even as we enter September I'd imagine the registrations are still dampened when compared to 2016, though at the same time there has been many political protests in the past months that may have motivated more people to register. I'd be very vurious to see an updated dataset!

# Assign3

Rachel Greenlee

9/9/2020

## Introduction

### Problem 1

Using the 173 majors listed in [fivethirtyeight.com's College Majors dataset](https://fivethirtyeight.com/features/the-economic-guide-to-picking-a-college-major/) [https://fivethirtyeight.com/features/the-economic-guide-to-picking-a-college-major/], provide code that identifies the majors that contain either "DATA" or "STATISTICS"

First, import the CSV provided by FiveThirtyEight on Github.

```
majors <- read.csv(url("https://raw.githubusercontent.com/fivethirtyeight/data/master/college-majors/majors-list.csv"))
```

Next, run the code to filter and only display majors that contain Data or Statistics.

```
grep(pattern = 'Data|Statistics', majors$Major, value = TRUE, ignore.case = TRUE)
```

```
## [1] "MANAGEMENT INFORMATION SYSTEMS AND STATISTICS"  
## [2] "COMPUTER PROGRAMMING AND DATA PROCESSING"  
## [3] "STATISTICS AND DECISION SCIENCE"
```

### Problem 2

Write code that transforms the data below:

```
[1] "bell pepper" "bilberry" "blackberry" "blood orange"  
[5] "blueberry" "cantaloupe" "chili pepper" "cloudberry"  
[9] "elderberry" "lime" "lychee" "mulberry"  
[13] "olive" "salal berry"
```

Into a format like this: c("bell pepper", "bilberry", "blackberry", "blood orange", "blueberry", "cantaloupe", "chili pepper", "cloudberry", "elderberry", "lime", "lychee", "mulberry", "olive", "salal berry")

This can be achieved by transforming the original dataframe using the concatenate and paste functions.

```
original <- data.frame(c("bell pepper", "bilberry", "blackberry", "blood orange", "blueberry", "cantaloupe", "chili pepper", "cloudberry", "elderberry", "lime", "lychee", "mulberry", "olive", "salal berry"))
```

```
cat(paste(original), collapse=" ")
```

```
## c("bell pepper", "bilberry", "blackberry", "blood orange", "blueberry", "cantaloupe", "chili pepper", "cloudberry", "elderberry", "lime", "lychee", "mulberry", "olive", "salal berry")
```

### Problem 3

Describe, in words, what these expressions will match...

First I'll create a dataset for this and problem 4 to test on. Also get stringr library loaded.

```
testdata <- c("hhh", "iceiceice", "amma", "coco", "riror", "ratextrachacteritar", "kayak", "mom", "blurb", "church", "bangbang", "greenlee", "banana", "777", "data\\1\\1", "anna", "2002", "elle")
```

```
library(stringr)
```

```
(.)\\1\\1
```

Any character followed by the literal string \\1\\1 since only one backslash is used. For example "data/1/1".



```
str_view(testdata, '(.)\\1', match = TRUE)
```

data

`"(.)\\2\\1"`

Two characters repeated once but in the opposite order the second time. For example "amma".

```
str_view(testdata, '(.)\\2\\1', match = TRUE)
```

amma

anna

2002

elle

`"(..)\\1"`

Any two character followed by the literal string \\1\\1 since only one backslash is used. For example "data/1/1".

```
str_view(testdata, '(..)\\1', match = TRUE)
```

data

`"(.)\\1\\.\\1"`

Character 1 then any character at all, then we need to see the exact same Character 1 again, then any character at all, then Character 1 again. Basically positions 1, 3, and 5 must all be the same character, 2 and 4 can be anything. For example, "riror". This can occur at any position within the string.

```
str_view(testdata, '(.)\\1\\.\\1', match = TRUE)
```

riror

banana

`"(.)\\1\\1\\.\\1"`

Any three characters and then "zero or more" of any other character. Next repeat just those first three characters but backwards. For example, "ratextracharactertar". With rat on the front, anything of any length in the middle, and tar (backwards rat) at the end.

```
str_view(testdata, "(.)\\1\\1\\.\\1", match = TRUE)
```

ratextracharactertar

## Problem 4

Construct regular expressions to match words that...

Start and end with the same character.

We can use:

```
str_subset(testdata, "^((.*)\\1)$")
```

```
## [1] "hhh"      "amma"     "riror"
## [4] "ratextracharactertar" "kayak"    "mom"
## [7] "blurb"    "777"      "anna"
## [10] "2002"     "elle"
```

Contain a repeated pair of letters (e.g. "church" contains "ch" repeated twice.)

We can use:

```
str_subset(testdata, "[A-Za-z][A-Za-z].*\\1")
```

```
## [1] "iceiceice"      "coco"           "ratextracharactertar"  
## [4] "church"         "bangbang"       "greenlee"  
## [7] "banana"
```

Contain one letter repeated in at least three places (e.g. “eleven” contains three “e”s.)

We can use:

```
str_subset(testdata, "[a-z].*\\1.*\\1")
```

```
## [1] "hhh"            "iceiceice"      "riror"  
## [4] "ratextracharactertar" "greenlee"       "banana"
```

# Project 1 - DATA607

Rachel Greenlee

9/14/2020

## Step 1 - Import the text file

I access the .txt file on my github repo. I skip the first 4 lines as it's dashes and headers. In order to remove the full lines that comprise of dashes every 3 rows, I can write the pattern True, True, False for it to take the first and second rows, skip the third, and repeat.

Now we read that into a table, with no header as we stripped it in line 1, and set the delimiter to a vertical bar/pipe. We tell it to fill in missing values in case the rows have unequal length for any player.

```
raw.tourney <- read_lines("https://raw.githubusercontent.com/rachel-greenlee/data607_proj1/master/tournament.txt",
                           skip = 4)[c(TRUE, TRUE, FALSE)]
raw.tourney <- read.table(textConnection(raw.tourney), header = FALSE, sep="|", fill = TRUE)
glimpse(raw.tourney)
```

```
## Rows: 128
## Columns: 11
## $ V1 <chr> "      1 ", "   ON ", "      2 ", "   MI ", "      3 ", "   MI ", " ...
## $ V2 <chr> " GARY HUA", "      ", " 15445895 / R: 1794  ->1...
## $ V3 <chr> "6.0  ", "N:2  ", "6.0  ", "N:2  ", "6.0  ", "N:2  ", "5.5  ", ...
## $ V4 <chr> "W 39", "W   ", "W 63", "B   ", "L  8", "W   ", "W 23", ...
## $ V5 <chr> "W 21", "B   ", "W 58", "W   ", "W 61", "B   ", "D 28", ...
## $ V6 <chr> "W 18", "W   ", "L  4", "B   ", "W 25", "W   ", "W  2", ...
## $ V7 <chr> "W 14", "B   ", "W 17", "W   ", "W 21", "B   ", "W 26", ...
## $ V8 <chr> "W  7", "W   ", "W 16", "B   ", "W 11", "W   ", "D  5", ...
## $ V9 <chr> "D 12", "B   ", "W 20", "W   ", "W 13", "B   ", "W 19", ...
## $ V10 <chr> "D  4", "W   ", "W  7", "B   ", "W 12", "W   ", "D  1", ...
## $ V11 <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
```

## Step 2 - Create data frame & clean it

I create two data frames, one called first\_rows and the other second\_rows since this data has one chess players information across two rows. Then I add an ID, merge across on that ID, drop some unnecessary rows, and we have a full dataframe called d.tourney.

```
#create 2 data frames
first_rows <- data.frame(first_rows <- raw.tourney %>%
  filter(row_number() %% 2 == 1))
second_rows <- data.frame(second_rows <- raw.tourney[c(rep(FALSE),TRUE),])

#create IDs for each data frame so we can match on it
first_rows$ID <- seq.int(nrow(first_rows))
second_rows$ID <- seq.int(nrow(first_rows))

#merge these two datasets together so all a player's data is in one row
d.tourney <- merge(first_rows, second_rows, by="ID")
#drop some columns we don't need
d.tourney <- subset(d.tourney, select = -c(V1.x, V11.x, V11.y))
```

Our dataframe columns need some cleaning so I create new columns and select the data I need.

```
#create a pre-rating column and extract characters 15-18 to grab the relevant
#part of the string
d.tourney$PreRating <- str_sub(d.tourney$V2.y, 15, 19)

#create a column for each of the possible 7 opponents, and start taking the
#value from the existing column at character 2 as to avoid the letter
#representing the outcome of the match, we just need the opponent's ID
#there are some blank entries at this point as not all players had 7 opponents
d.tourney$R1opp <- str_sub(d.tourney$V4.x, 2, )
d.tourney$R2opp <- str_sub(d.tourney$V5.x, 2, )
d.tourney$R3opp <- str_sub(d.tourney$V6.x, 2, )
d.tourney$R4opp <- str_sub(d.tourney$V7.x, 2, )
d.tourney$R5opp <- str_sub(d.tourney$V8.x, 2, )
d.tourney$R6opp <- str_sub(d.tourney$V9.x, 2, )
d.tourney$R7opp <- str_sub(d.tourney$V10.x, 2, )

#drop all those columns now that we have what we need from them
d.tourney <- subset(d.tourney, select = -c(V4.x, V5.x, V6.x, V7.x, V8.x, V9.x,
  V10.x, V2.y, V3.y, V4.y, V5.y, V6.y,
  V7.y, V8.y, V9.y, V10.y))

glimpse(d.tourney)
```

```
## Rows: 64
## Columns: 12
## $ ID      <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17...
## $ V2.x    <chr> " GARY HUA", " DAKSHESH DARURI ...
## $ V3.x    <chr> "6.0 ", "6.0 ", "6.0 ", "5.5 ", "5.5 ", "5.0 ", "5....
## $ V1.y    <chr> " ON ", " MI ", " MI ", " MI ", " MI ", " OH ...
## $ PreRating <chr> " 1794", " 1553", " 1384", " 1716", " 1655", " 1686", " 1...
## $ R1opp   <chr> " 39", " 63", " 8", " 23", " 45", " 34", " 57", "...
## $ R2opp   <chr> " 21", " 58", " 61", " 28", " 37", " 29", " 46", "...
```

```
## $ R3opp      <chr> " 18", " 4", " 25", " 2", " 12", " 11", " 13", "...
## $ R4opp      <chr> " 14", " 17", " 21", " 26", " 13", " 35", " 11", "...
## $ R5opp      <chr> " 7", " 16", " 11", " 5", " 4", " 10", " 1", "...
## $ R6opp      <chr> " 12", " 20", " 13", " 19", " 14", " 27", " 9", "...
## $ R7opp      <chr> " 4", " 7", " 12", " 1", " 17", " 21", " 2", "...
```

## Step 3 - Create calculated variables

Next I calculate the number of rounds each player had, out of a possible 7.

```
#set the 7 variables of opponent IDs to be numeric and this also creates NAs in
#the blank spaces
cols = c(6:12);
d.tourney[,cols] = apply(d.tourney[,cols], 2, function(x) as.numeric(as.character(x)))

#sum up the number of NAs across each row, subtract for 7 possible games to get
#the number of rounds that player plaid - store in "roundsplayed" variable
d.tourney$roundsplayed <- 7 - (apply(is.na(d.tourney), 1, sum))
```

Now we have to look-up the PreRating for all of a player's opponents, based on their ID so we can calculate the average PreRating of their opponents. I did this via a look-up table.

```
#create a 2-variable look-up table with just the player ID and their PreRating
ratinglookup <- subset(d.tourney, select = c(ID, PreRating))

#we can overwrite the player ID that's in each RXopp variable with their
#PreRating, which we can access by using the lookup table created above
d.tourney$R1opp =
  ratinglookup[match(d.tourney$R1opp,
    ratinglookup$ID), "PreRating"]

d.tourney$R2opp =
  ratinglookup[match(d.tourney$R2opp,
    ratinglookup$ID), "PreRating"]

d.tourney$R3opp =
  ratinglookup[match(d.tourney$R3opp,
    ratinglookup$ID), "PreRating"]

d.tourney$R4opp =
  ratinglookup[match(d.tourney$R4opp,
    ratinglookup$ID), "PreRating"]

d.tourney$R5opp =
  ratinglookup[match(d.tourney$R5opp,
    ratinglookup$ID), "PreRating"]

d.tourney$R6opp =
  ratinglookup[match(d.tourney$R6opp,
    ratinglookup$ID), "PreRating"]

d.tourney$R7opp =
```

```

ratinglookup[match(d.tourney$R7opp,
ratinglookup$ID), "PreRating"]

#make a numeric class so we can do calculations
d.tourney <- d.tourney %>%
  mutate_at(vars(matches('R(.)opp')), list(as.numeric))

```

Finally, we create the variable that holds the average PreRating of the player's opponents.

```

#sum the appropriate rows or the opponents' PreRatings then divide by the rounds
#played
d.tourney$Avg_PreRating_of_Opponents = as.numeric(((rowSums(d.tourney[,c(6:12)]),
  na.rm = TRUE))/d.tourney$roundsplayed))

#round to nearest whole number
d.tourney$Avg_PreRating_of_Opponents <- round(d.tourney$Avg_PreRating_of_Opponents)

glimpse(d.tourney)

```

```

## Rows: 64
## Columns: 14
## $ ID <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 1...
## $ V2.x <chr> " GARY HUA ", " D...
## $ V3.x <chr> "6.0 ", "6.0 ", "6.0 ", "5.5 ", "5.5...
## $ V1.y <chr> " ON ", " MI ", " MI ", " MI ", ...
## $ PreRating <chr> " 1794", " 1553", " 1384", " 1716", " 16...
## $ R1opp <dbl> 1436, 1175, 1641, 1363, 1242, 1399, 1092...
## $ R2opp <dbl> 1563, 917, 955, 1507, 980, 1602, 377, 14...
## $ R3opp <dbl> 1600, 1716, 1745, 1553, 1663, 1712, 1666...
## $ R4opp <dbl> 1610, 1629, 1563, 1579, 1666, 1438, 1712...
## $ R5opp <dbl> 1649, 1604, 1712, 1655, 1716, 1365, 1794...
## $ R6opp <dbl> 1663, 1595, 1666, 1564, 1610, 1552, 1411...
## $ R7opp <dbl> 1716, 1649, 1663, 1794, 1629, 1563, 1553...
## $ roundsplayed <dbl> 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 6, 7, 7...
## $ Avg_PreRating_of_Opponents <dbl> 1605, 1469, 1564, 1574, 1501, 1519, 1372...

```

## Step 4 - Final cleaning & export

Do some cleaning of the final dataframe.

```
#remove unnecessary columns
d.tourney <- subset(d.tourney, select = -c(ID, R1opp, R2opp, R3opp, R4opp,
                                           R5opp, R6opp, R7opp, roundsplayed))

#reorder to match assignment
d.tourney <- d.tourney[, c(1, 3, 2, 4, 5)]

#rename columns
d.tourney <- rename(d.tourney, c("V2.x"="Name", "V1.y"="State",
                                  "V3.x"="TotalNumPoints"))

#capitlize names correctly
d.tourney$Name <- str_to_title(d.tourney$Name)

#trim whitespace around state abbreviations
d.tourney$State <- trimws(d.tourney$State, which = c("both"))

head(d.tourney)
```

```
##              Name State TotalNumPoints PreRating
## 1 Gary Hua          ON             6.0      1794
## 2 Dakshesh Daruri    MI             6.0      1553
## 3 Aditya Bajaj       MI             6.0      1384
## 4 Patrick H Schilling MI             5.5      1716
## 5 Hanshi Zuo         MI             5.5      1655
## 6 Hansen Song       OH             5.0      1686
## Avg_PreRating_of_Opponents
## 1              1605
## 2              1469
## 3              1564
## 4              1574
## 5              1501
## 6              1519
```

Last, we write the dataframe to a CSV for export to the desktop.

```
write.csv(d.tourney, "C:\\Users\\rgreenlee\\Desktop\\\\\\chess_tournament_results.csv",
          row.names = FALSE)
```

# Assignment 5

Rachel Greenlee

9/21/2020

## Introduction

I start with a screenshot of a small dataset and the goal is to put it into a tidy data format and then perform analysis to compare the arrival delays for the two airlines.

## Step 1 - Reproduce data in MySQL and import to R

For the sake of practice, I'll create two separate tables in MySQL, one for each airline. In MySQL I create the empty tables with the variables in such a way that the data will be in a long structure per the grading rubric. I then input the handful of rows of data.

```
1 ● ○ CREATE TABLE alaska_fl (  
2     destination VARCHAR(255),  
3     on_time NUMERIC(10),  
4     delay NUMERIC(10)  
5 );  
6  
7 ● INSERT INTO alaska_fl (destination, on_time, delay)  
8 VALUES ('Los Angeles', 497, 62),  
9         ('Phoenix', 221, 12),  
10        ('San Diego', 212, 20),  
11        ('San Francisco', 503, 102),  
12        ('Seattle', 1841, 305);  
13  
14 ● ○ CREATE TABLE amwest_fl (  
15     destination VARCHAR(255),  
16     on_time NUMERIC(10),  
17     delay NUMERIC(10)  
18 );  
19  
20 ● INSERT INTO amwest_fl (destination, on_time, delay)  
21 VALUES ('Los Angeles', 694, 117),  
22         ('Phoenix', 4840, 415),  
23         ('San Diego', 383, 65),  
24         ('San Francisco', 320, 129),  
25         ('Seattle', 201, 61);
```

Figure 1: Image of MySQL Code.



## Step 2 - Connect R to MySQL to create dataframe

First I load the RMySQL package in order to connect with MySQL. I also load the keyring package so I can access my stored password and keep it hidden from the world. Let's load kable too for tables.

```
library('keyring')
library("RMySQL")
library("kableExtra")
```

Now we must access the datasets from MySQL. And let's preview them quick as well.

```
mysqlpass <- key_get('mysql', 'root')

rachdb = dbConnect(MySQL(), user='root', password=mysqlpass,
  dbname='607assign5_flights', host='localhost')

amwest_fl <- dbGetQuery(rachdb, "select * from amwest_fl")
alaska_fl <- dbGetQuery(rachdb, "select * from alaska_fl")

kable(amwest_fl, format = "markdown")
```

destination	on_time	delay
Los Angeles	694	117
Phoenix	4840	415
San Diego	383	65
San Francisco	320	129
Seattle	201	61

```
kable(alaska_fl, format = "markdown")
```

destination	on_time	delay
Los Angeles	497	62
Phoenix	221	12
San Diego	212	20
San Francisco	503	102
Seattle	1841	305

## Step 3 - Use tidyr and dplyr as needed to tidy and transform data

Using dplyr we need to combine our two datasets into one and add a column to identify which airline the data is from.

```
library(dplyr)
```

```
#add airline variable with correct airline for each dataset
alaska_fl <- mutate(alaska_fl, airline = "ALASKA")
amwest_fl <- mutate(amwest_fl, airline = "AMWEST")
```

```
#bind the two datasets together vertically then arrange by destination
flights <- bind_rows(alaska_fl, amwest_fl)
flights <- arrange(flights, destination)
kable(flights, format = "markdown")
```

destination	on_time	delay	airline
Los Angeles	497	62	ALASKA
Los Angeles	694	117	AMWEST
Phoenix	221	12	ALASKA
Phoenix	4840	415	AMWEST
San Diego	212	20	ALASKA
San Diego	383	65	AMWEST
San Francisco	503	102	ALASKA
San Francisco	320	129	AMWEST
Seattle	1841	305	ALASKA
Seattle	201	61	AMWEST

## Step 4 - Perform analysis to compare arrival delays for the two airlines

First by looking at a summary of the full dataset we see that the median frequency of on time flights is 440 and the median frequency of delayed flights is 83.5. One airline, we don't know which yet, has the max of 415 delayed flights to a certain destination.

```
kable((summary(flights)), format = "markdown")
```

destination	on_time	delay	airline
Length:10	Min. : 201.0	Min. : 12.00	Length:10
Class :character	1st Qu.: 245.8	1st Qu.: 61.25	Class :character
Mode :character	Median : 440.0	Median : 83.50	Mode :character
NA	Mean : 971.2	Mean :128.80	NA
NA	3rd Qu.: 646.2	3rd Qu.:126.00	NA
NA	Max. :4840.0	Max. :415.00	NA

Looking by airline we can see a comparison between the two. ALASKA has a much higher median on-time frequency across the destinations at 597 compared to AMWEST's 383. AMWEST has a larger IQR suggesting more variance in their on-time rates by destination.

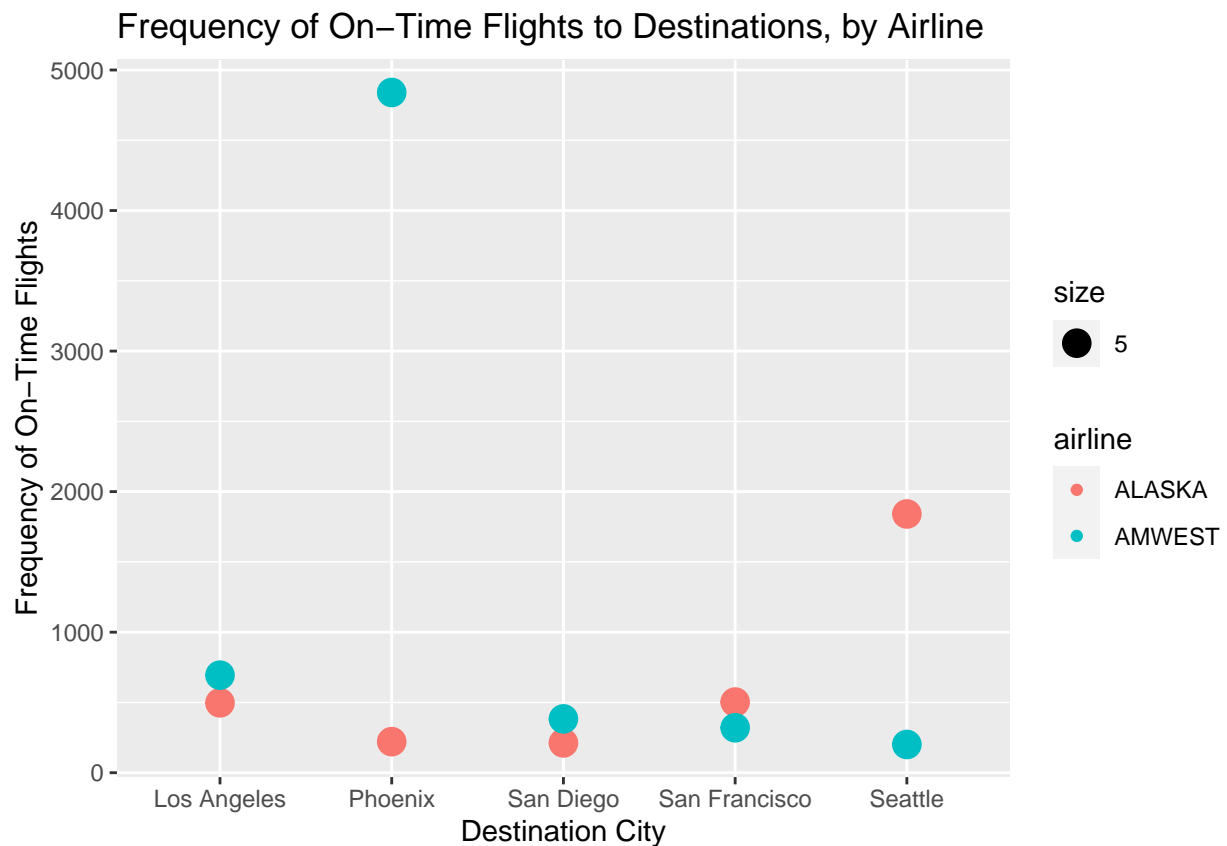
```
kable((flights %>%
  group_by(airline) %>%
  summarise(median_on_time = median(on_time), iqr_on_time = IQR(on_time),
    min_on_time = min(on_time), max_on_time = max(on_time))),
  format = "markdown")
```

airline	median_on_time	iqr_on_time	min_on_time	max_on_time
ALASKA	497	282	212	1841
AMWEST	383	374	201	4840

Since this data is already aggregated, a nice way to visualize it might be this first dot plot below. The number of on-time flights for Los Angeles, San Diego, and San Francisco look similar, but there are large difference in the other two cities. In Phoenix AMWEST has nearly 5,000 on-time flights while ALASKA only has 250. In a flip, ALASKA has the greater amount of on-time flights to Seattle, around 2000, with AMWEST only have around 200.

```
library(ggplot2)
```

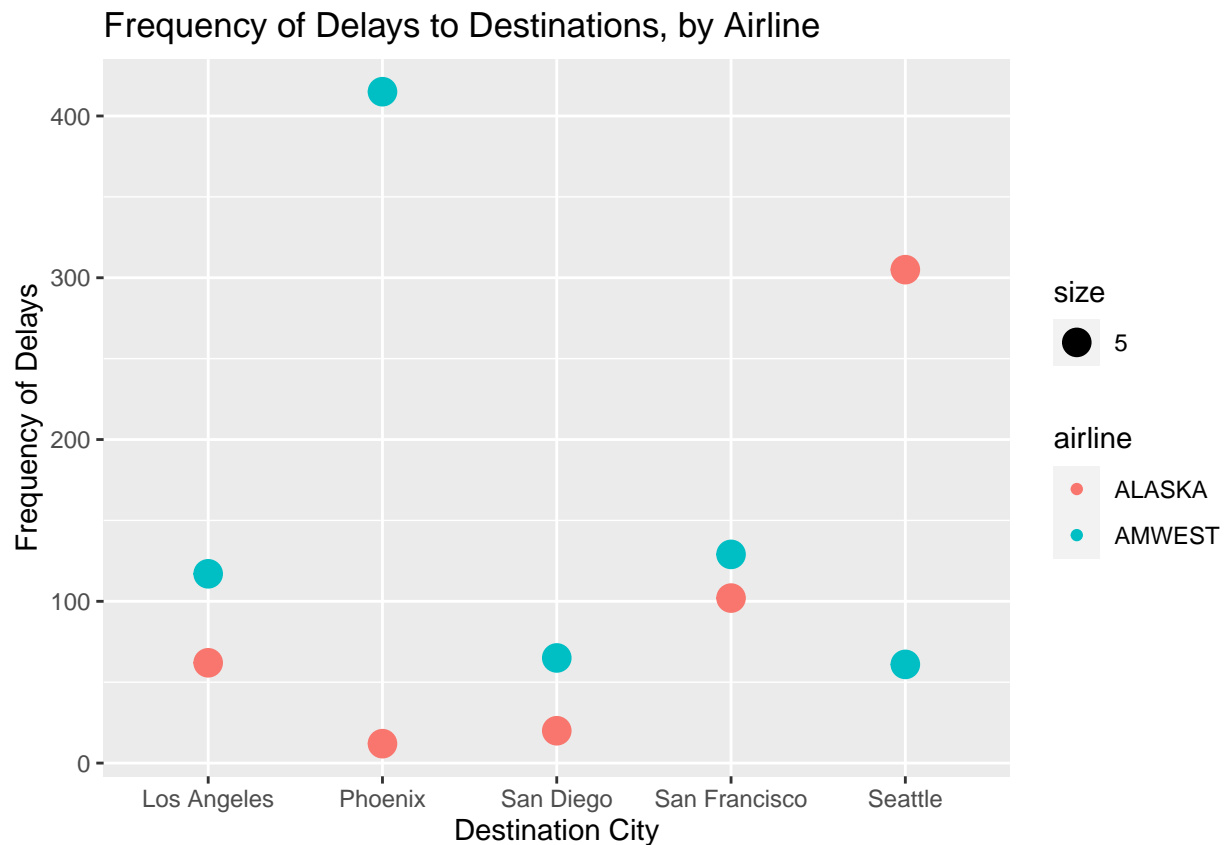
```
ggplot(data = flights, aes(x = on_time, y = destination)) +  
  geom_point(aes(color = airline, size = 5)) +  
  ylab('Destination City') +  
  xlab('Frequency of On-Time Flights') +  
  labs(title="Frequency of On-Time Flights to Destinations, by Airline") +  
  coord_flip()
```



Below, in looking at the delayed flights variable we can see that across all destinations, except for Seattle, AMWEST has more delays. The largest discrepancies are found in Seattle and Phoenix. Seattle is the only destination where ALASKA has more delays, at over 300 compared to AMWEST's around 60. Phoenix is the inverse of that, ALASKA has what looks to be around 10 delays while that is the worst amount of delays in the entire dataset for AMWEST, that 415 maximum delays.

From this view one would conclude if you were flying to these destinations you'd want to fly ALASKA, unless you're going to Seattle, then it's AMWEST. However...

```
ggplot(data = flights, aes(x = delay, y = destination)) +  
  geom_point(aes(color = airline, size = 5)) +  
  ylab('Destination City') +  
  xlab('Frequency of Delays') +  
  labs(title="Frequency of Delays to Destinations, by Airline") +  
  coord_flip()
```

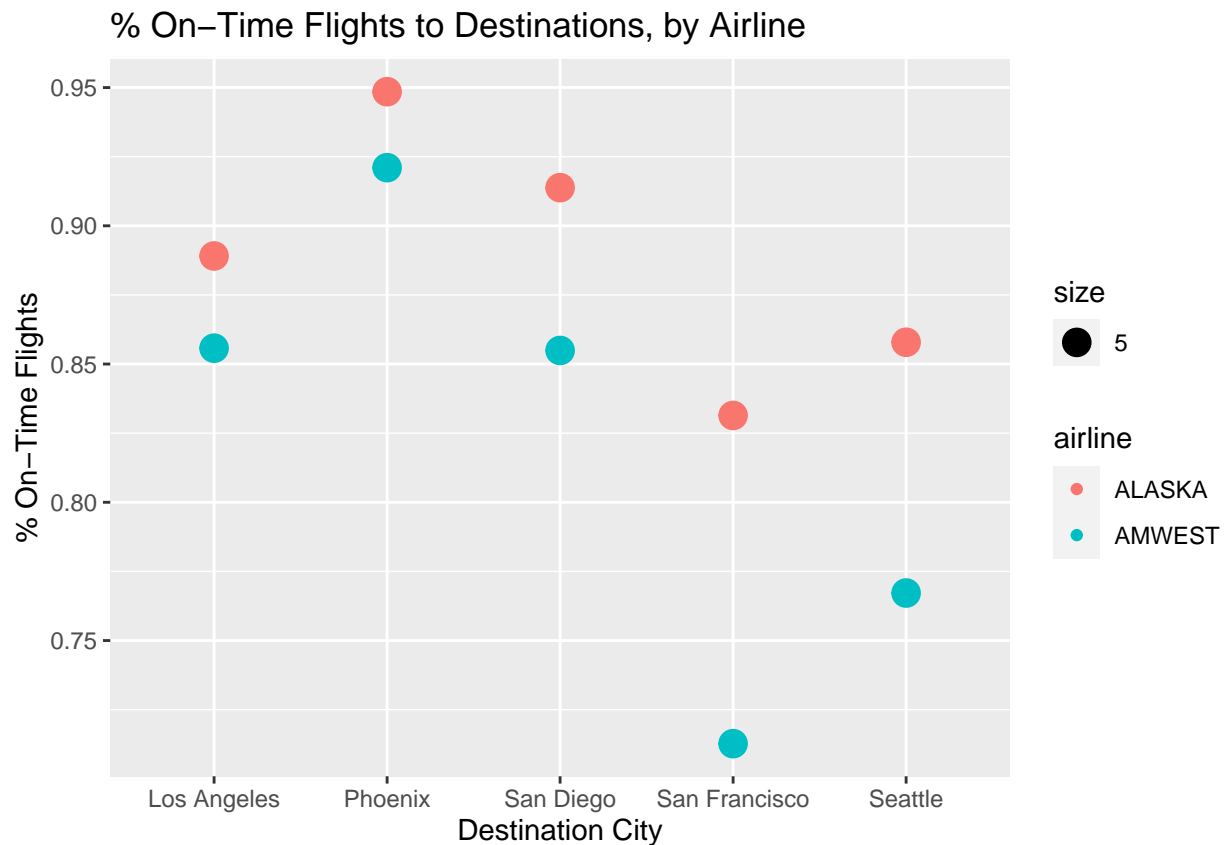


It might be more valuable to look at these not as just counts, but as percents out of the total flights to that destination for that airline. In this final dot plot below, now we see that for every destination, regardless of the raw number of flights, ALASKA has a greater percentage of on-time flights.

This plot also shows us that AMWEST has a particularly hard time being on-time in San Francisco, it's the lowest dot on the chart near 71%.

```
flights <- mutate(flights, on_time_perc = flights$on_time/(flights$on_time + flights$delay))

ggplot(data = flights, aes(x = on_time_perc, y = destination)) +
  geom_point(aes(color = airline, size = 5)) +
  ylab('Destination City') +
  xlab('% On-Time Flights') +
  labs(title="% On-Time Flights to Destinations, by Airline") +
  coord_flip()
```



## Conclusion

This process shows how important it is to get the data in the correct format for analysis and graphing, and the importance of not just plotting the variables you have but taking time to create a more logical measure, such as the percent of on-time flights as opposed to the frequency. While the number of times an airline flies on time to a destination certainly holds merit, the percentage allows us to compare between the two airlines better as in some cases one simply didn't fly to a certain destination as often.

# Project 2 - Global High-Protein Food Production in Light of Carbon Footprint

Rachel Greenlee, Atina Karim, Douglas Barley

9/28/2020

## Introduction

We choose to work with the dataset Rachel posted in last week's discussion forum.

Rachel found a dataset on Kaggle, originally from the Food and Agriculture Organization of the United Nations, that gives food production data for 245 countries. It shows what food items were produced for humans vs animals from 1961-2013. The years in the CSV file are displayed across the columns, making this a very wide dataset!

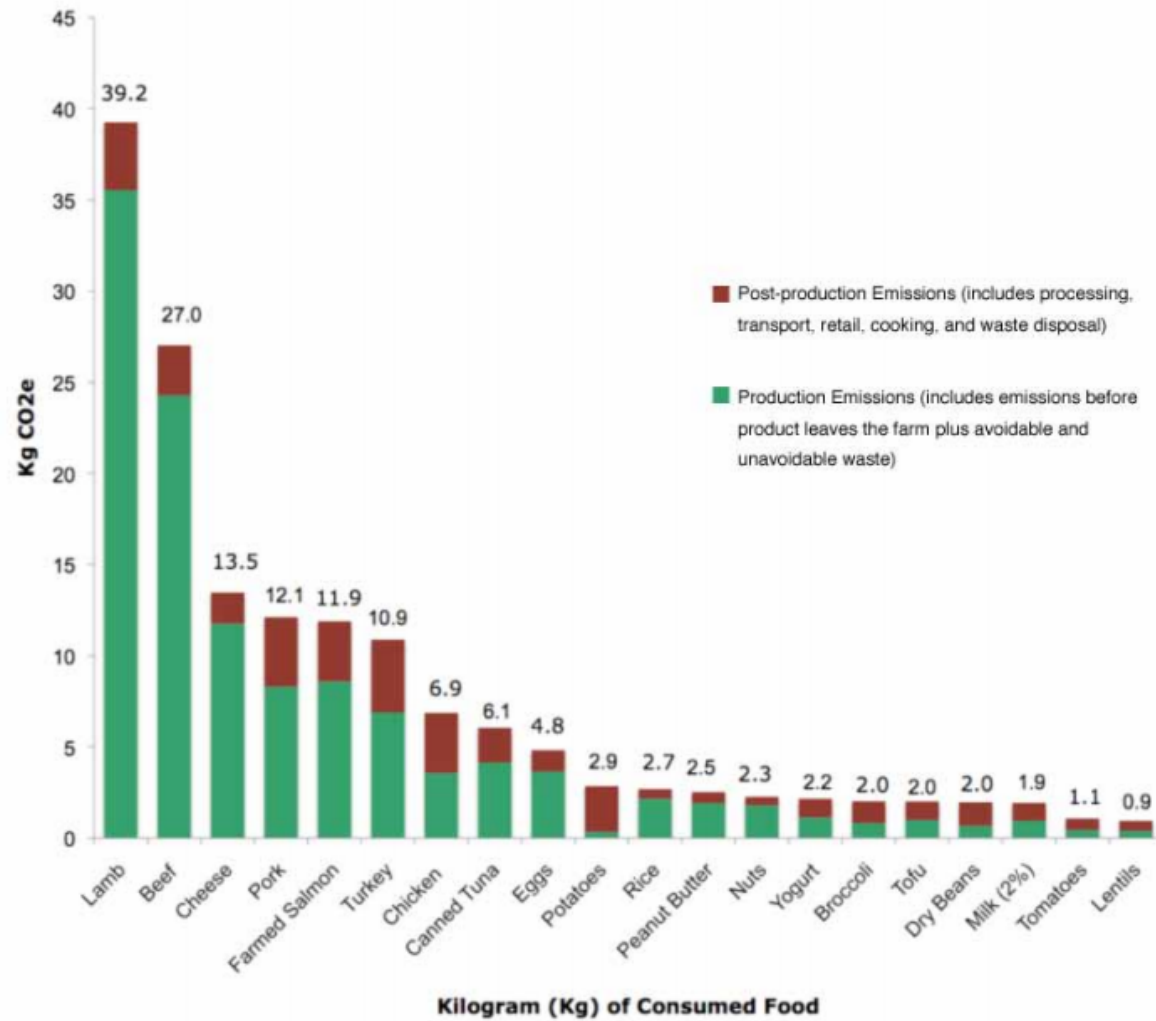
<https://www.kaggle.com/dorbicycle/world-foodfeed-production>

In this assignment we will compare the carbon footprint of high-protein foods. For simplicity's sake we used a separate resource to determine which food items to include. The graph below is taken from the 2011 report called "Meat Eater's Guide to Climate Change + Health" by The Environmental Working Group, a research and advocacy organization that partnered with CleanMetrics Corp, an environmental firm.

[http://static.ewg.org/reports/2011/meateaters/pdf/methodology\\_ewg\\_meat\\_eaters\\_guide\\_to\\_health\\_and\\_climate\\_2011.pdf](http://static.ewg.org/reports/2011/meateaters/pdf/methodology_ewg_meat_eaters_guide_to_health_and_climate_2011.pdf)

Based on this I will compare legume production (legumes and dry beans) vs the top three animal meats with the highest carbon footprint, lamb (mutton), beef (bovine), and pork (pigmeat).

**Figure 1. Full Lifecycle Assessment of Greenhouse Gas Emissions: Most Emissions from Common Proteins and Vegetables Occur During Production**



\*These include production emissions from avoidable (plate waste, spoilage) and unavoidable waste (fat and moisture loss during cooking)

## Step 1 - Import data from the provided CSV file and filter by food item

Let's load the libraries we might need.

```
library("dplyr")
library("tidyr")
library("ggplot2")
library("rcartocolor")
library("kableExtra")
```

We've put the CSV file into my Github repository, we will read it from there, all 21,477 rows and 63 columns.

```
food_rawdt <- read.csv(url("https://raw.githubusercontent.com/rachel-greenlee/data607_proj2/master/FA0.
```

Take a peek at the raw data.

```
glimpse(food_rawdt)
```

```
## Rows: 21,477
## Columns: 63
## $ Area.Abbreviation <chr> "AFG", "AFG", "AFG", "AFG", "AFG", "AFG", "AFG", ...
## $ Area.Code         <int> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2...
## $ Area              <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Afg...
## $ Item.Code         <int> 2511, 2805, 2513, 2513, 2514, 2514, 2517, 2520, 2...
## $ Item              <chr> "Wheat and products", "Rice (Milled Equivalent)",...
## $ Element.Code      <int> 5142, 5142, 5521, 5142, 5521, 5142, 5142, 5142, 5...
## $ Element           <chr> "Food", "Food", "Feed", "Food", "Feed", "Food", "...
## $ Unit              <chr> "1000 tonnes", "1000 tonnes", "1000 tonnes", "100...
## $ latitude          <dbl> 33.94, 33.94, 33.94, 33.94, 33.94, 33.94, 33.94, ...
## $ longitude         <dbl> 67.71, 67.71, 67.71, 67.71, 67.71, 67.71, 67.71, ...
## $ Y1961             <int> 1928, 183, 76, 237, 210, 403, 17, 0, 111, 45, 0, ...
## $ Y1962             <int> 1904, 183, 76, 237, 210, 403, 18, 0, 97, 45, 0, 4...
## $ Y1963             <int> 1666, 182, 76, 237, 214, 410, 19, 0, 103, 45, 0, ...
## $ Y1964             <int> 1950, 220, 76, 238, 216, 415, 20, 0, 110, 45, 0, ...
## $ Y1965             <int> 2001, 220, 76, 238, 216, 415, 21, 0, 113, 31, 0, ...
## $ Y1966             <int> 1808, 195, 75, 237, 216, 413, 22, 0, 117, 14, 16,...
## $ Y1967             <int> 2053, 231, 71, 225, 235, 454, 23, 0, 128, 19, 23,...
## $ Y1968             <int> 2045, 235, 72, 227, 232, 448, 24, 0, 130, 30, 31,...
## $ Y1969             <int> 2154, 238, 73, 230, 236, 455, 25, 0, 134, 34, 28,...
## $ Y1970             <int> 1819, 213, 74, 234, 200, 383, 26, 0, 125, 15, 9, ...
## $ Y1971             <int> 1963, 205, 71, 223, 201, 386, 26, 0, 147, 0, 13, ...
## $ Y1972             <int> 2215, 233, 70, 219, 216, 416, 27, 0, 138, 0, 13, ...
## $ Y1973             <int> 2310, 246, 72, 225, 228, 439, 27, 0, 143, 28, 6, ...
## $ Y1974             <int> 2335, 246, 76, 240, 231, 445, 28, 0, 160, 32, 0, ...
## $ Y1975             <int> 2434, 255, 77, 244, 234, 451, 29, 0, 169, 20, 0, ...
## $ Y1976             <int> 2512, 263, 80, 255, 240, 463, 37, 0, 324, 28, 10,...
## $ Y1977             <int> 2282, 235, 60, 185, 228, 439, 32, 0, 176, 24, 16,...
## $ Y1978             <int> 2454, 254, 65, 203, 234, 451, 33, 0, 225, 24, 13,...
## $ Y1979             <int> 2443, 270, 64, 198, 228, 440, 31, 0, 232, 34, 6, ...
## $ Y1980             <int> 2129, 259, 64, 202, 226, 437, 31, 0, 240, 61, 15,...
## $ Y1981             <int> 2133, 248, 60, 189, 210, 407, 29, 0, 247, 50, 0, ...
## $ Y1982             <int> 2068, 217, 55, 174, 199, 384, 27, 0, 248, 43, 0, ...
```



```
## $ Y1983      <int> 1994, 217, 53, 167, 192, 371, 28, 0, 242, 38, 0, ...
## $ Y1984      <int> 1851, 197, 51, 160, 182, 353, 26, 0, 235, 46, 0, ...
## $ Y1985      <int> 1791, 186, 48, 151, 173, 334, 25, 0, 226, 23, 0, ...
## $ Y1986      <int> 1683, 200, 46, 145, 170, 330, 23, 0, 217, 25, 0, ...
## $ Y1987      <int> 2194, 193, 46, 145, 154, 298, 23, 0, 196, 3, 0, 8...
## $ Y1988      <int> 1801, 202, 47, 148, 148, 287, 23, 0, 198, 45, 0, ...
## $ Y1989      <int> 1754, 191, 46, 145, 137, 265, 23, 0, 184, 54, 0, ...
## $ Y1990      <int> 1640, 199, 43, 135, 144, 279, 24, 0, 205, 47, 0, ...
## $ Y1991      <int> 1539, 197, 43, 132, 126, 245, 24, 0, 203, 29, 0, ...
## $ Y1992      <int> 1582, 249, 40, 120, 90, 170, 18, 0, 210, 29, 0, 5...
## $ Y1993      <int> 1840, 218, 50, 155, 141, 272, 22, 0, 210, 29, 0, ...
## $ Y1994      <int> 1855, 260, 46, 143, 150, 289, 20, 0, 211, 29, 0, ...
## $ Y1995      <int> 1853, 319, 41, 125, 159, 310, 21, 0, 212, 29, 0, ...
## $ Y1996      <int> 2177, 254, 44, 138, 108, 209, 17, 0, 213, 29, 0, ...
## $ Y1997      <int> 2343, 326, 50, 159, 90, 173, 20, 0, 214, 28, 0, 5...
## $ Y1998      <int> 2407, 347, 48, 154, 99, 192, 21, 0, 214, 28, 0, 5...
## $ Y1999      <int> 2463, 270, 43, 141, 72, 141, 17, 0, 217, 28, 0, 6...
## $ Y2000      <int> 2600, 372, 26, 84, 35, 66, 20, 0, 219, 29, 0, 61,...
## $ Y2001      <int> 2668, 411, 29, 83, 48, 93, 20, 0, 215, 29, 0, 61,...
## $ Y2002      <int> 2776, 448, 70, 122, 89, 170, 18, 0, 217, 29, 0, 7...
## $ Y2003      <int> 3095, 460, 48, 144, 63, 117, 16, 1, 347, 51, 0, 9...
## $ Y2004      <int> 3249, 419, 58, 185, 120, 231, 15, 2, 276, 50, 0, ...
## $ Y2005      <int> 3486, 445, 236, 43, 208, 67, 21, 1, 294, 29, 0, 1...
## $ Y2006      <int> 3704, 546, 262, 44, 233, 82, 11, 1, 294, 61, 0, 1...
## $ Y2007      <int> 4164, 455, 263, 48, 249, 67, 19, 0, 260, 65, 0, 1...
## $ Y2008      <int> 4252, 490, 230, 62, 247, 69, 21, 0, 242, 54, 0, 2...
## $ Y2009      <int> 4538, 415, 379, 55, 195, 71, 18, 0, 250, 114, 0, ...
## $ Y2010      <int> 4605, 442, 315, 60, 178, 82, 14, 0, 192, 83, 0, 2...
## $ Y2011      <int> 4711, 476, 203, 72, 191, 73, 14, 0, 169, 83, 0, 2...
## $ Y2012      <int> 4810, 425, 367, 78, 200, 77, 14, 0, 196, 69, 0, 2...
## $ Y2013      <int> 4895, 422, 360, 89, 200, 76, 12, 0, 230, 81, 0, 2...
```

Ok in looking at how this dataset is categorizing the crop food types, we will need to subset just the following items as they are called by this dataset:

*Legumes* {Pulses, Other and products; Pulses; Beans; Soyabeans}

*3\_Meats* {Bovine Meat; Mutton & Goat Meat; Pigmeat}

Now lets filter this dataset down just to the food items we are interested in that were listed above.

```
food_dt <- food_rawdt %>%
  filter(Item %in% c("Bovine Meat", "Mutton & Goat Meat", "Pigmeat", "Pulses,
    Other and products", "Pulses", "Beans", "Soyabeans"))
```

## Step 2 - Tidy the data

First, we need to make this dataset long instead of wide, as currently the years run across columns.

```
food_dt <- food_dt %>%  
  gather(Year, Amount, Y1961:Y2013)
```

Lets make a new variable that shows if the item is part of the “Pulses” or the “3\_Meats” using the mutate function from dplyr and ifelse.

```
food_dt <- food_dt %>%  
  mutate(food_dt, Category = ifelse(Item %in% c("Bovine Meat",  
                                                "Mutton & Goat Meat", "Pigmeat"),  
                                     "3_Meats", "Pulses"))
```

We can also drop some columns we don't need and reorder.

```
food_dt <- select(food_dt, -c(Area.Code, Item.Code, Element.Code, latitude,  
                             longitude))  
  
food_dt <- food_dt[, c(8, 3, 4, 2, 6, 5, 7, 1)]
```

We also want the Year variable to be an integer so we can plot it easily later.

```
food_dt$Year <- gsub("[^0-9.-]", "", food_dt$Year)  
  
food_dt$Year <- as.numeric(food_dt$Year)
```

## Step 3 - Exploratory analysis

Okay first lets take a peek at what the dataset looks like now.

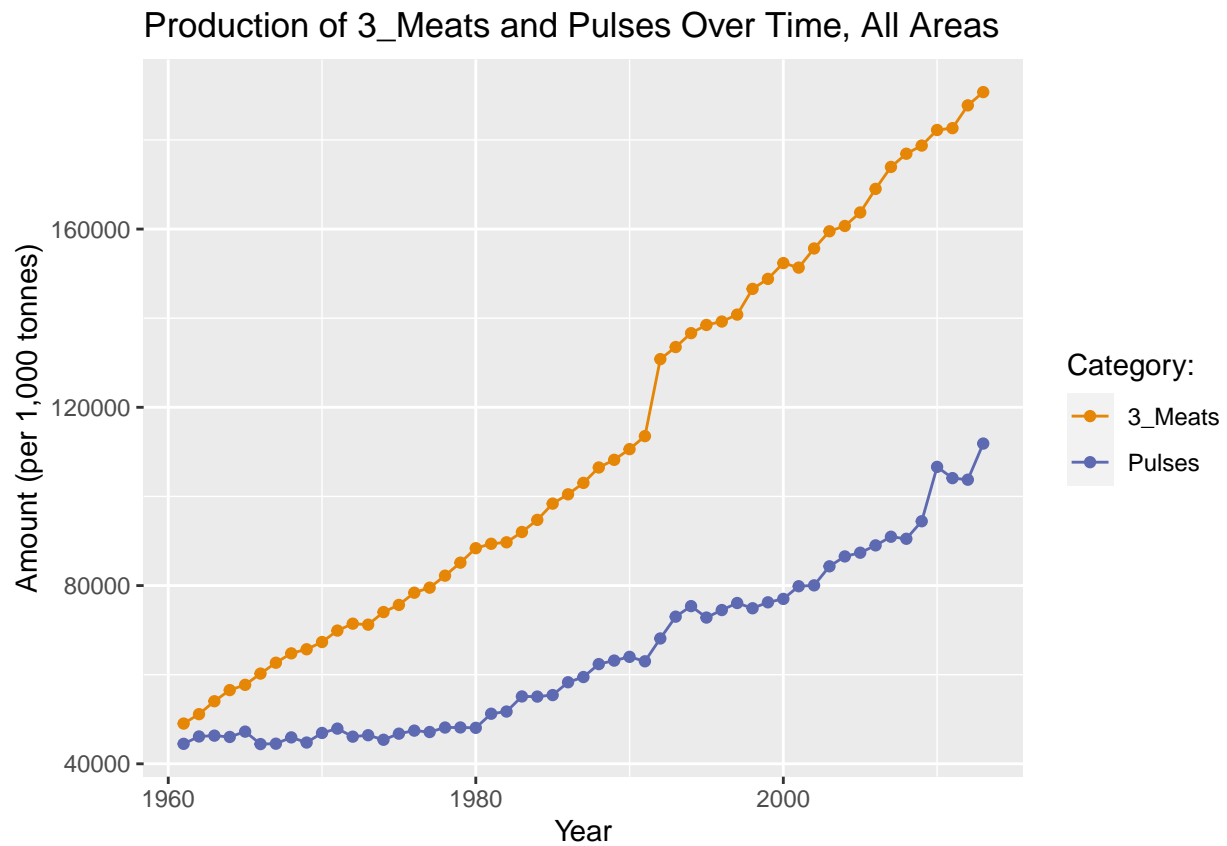
```
glimpse(food_dt)
```

```
## Rows: 63,653  
## Columns: 8  
## $ Category      <chr> "3_Meats", "3_Meats", "Pulses", "Pulses", "Pulses...  
## $ Item           <chr> "Bovine Meat", "Mutton & Goat Meat", "Pulses", "P...  
## $ Element        <chr> "Food", "Food", "Feed", "Food", "Feed", "Food", "...  
## $ Area           <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Afg...  
## $ Year           <dbl> 1961, 1961, 1961, 1961, 1961, 1961, 1961, 1961, 1...  
## $ Unit           <chr> "1000 tonnes", "1000 tonnes", "1000 tonnes", "100...  
## $ Amount         <int> 43, 73, 1, 15, 0, 5, 0, 7, 13, 4, 2, 5, 4, 0, 0, ...  
## $ Area.Abbreviation <chr> "AFG", "AFG", "AFG", "AFG", "ALB", "ALB", "ALB", ...
```

To start, let's combine all of the areas' total tonnes produced and see how the 3\_Meats and Pulses production has changed over time for all areas.

Production of both of these food categories has increased since 1960, with the 3\_Meats surpassing Pulses around 1965 and having a steady climb since then.

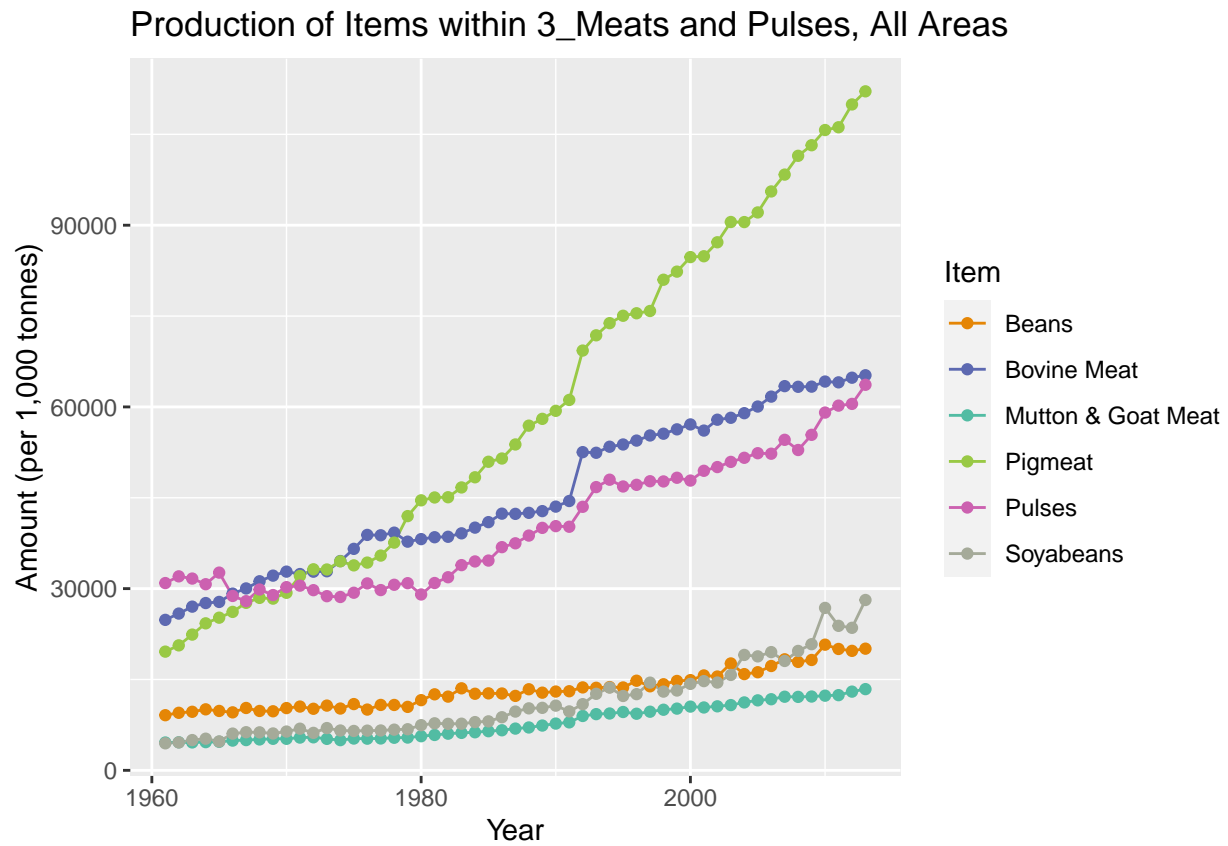
```
ggplot(food_dt, aes(x = Year, y = Amount, color=Category)) +  
  stat_summary(fun=sum, geom="line") +  
  stat_summary(fun=sum, geom="point") +  
  labs(title="Production of 3_Meats and Pulses Over Time, All Areas") +  
  ylab('Amount (per 1,000 tonnes)') +  
  xlab('Year') +  
  scale_color_carto_d(name = "Category: ", palette = "Vivid")
```



We can break this down and look at the types within 3\_Meats and Pulses as well.

This shows that Pigmeat has had the most growth over the years of this dataset.

```
ggplot(food_dt, aes(x = Year, y = Amount, colour=Item)) +  
  stat_summary(fun=sum, geom="line") +  
  stat_summary(fun=sum, geom="point") +  
  labs(title="Production of Items within 3_Meats and Pulses, All Areas") +  
  ylab('Amount (per 1,000 tonnes)') +  
  xlab('Year') +  
  scale_color_carto_d(name = "Item", palette = "Vivid")
```



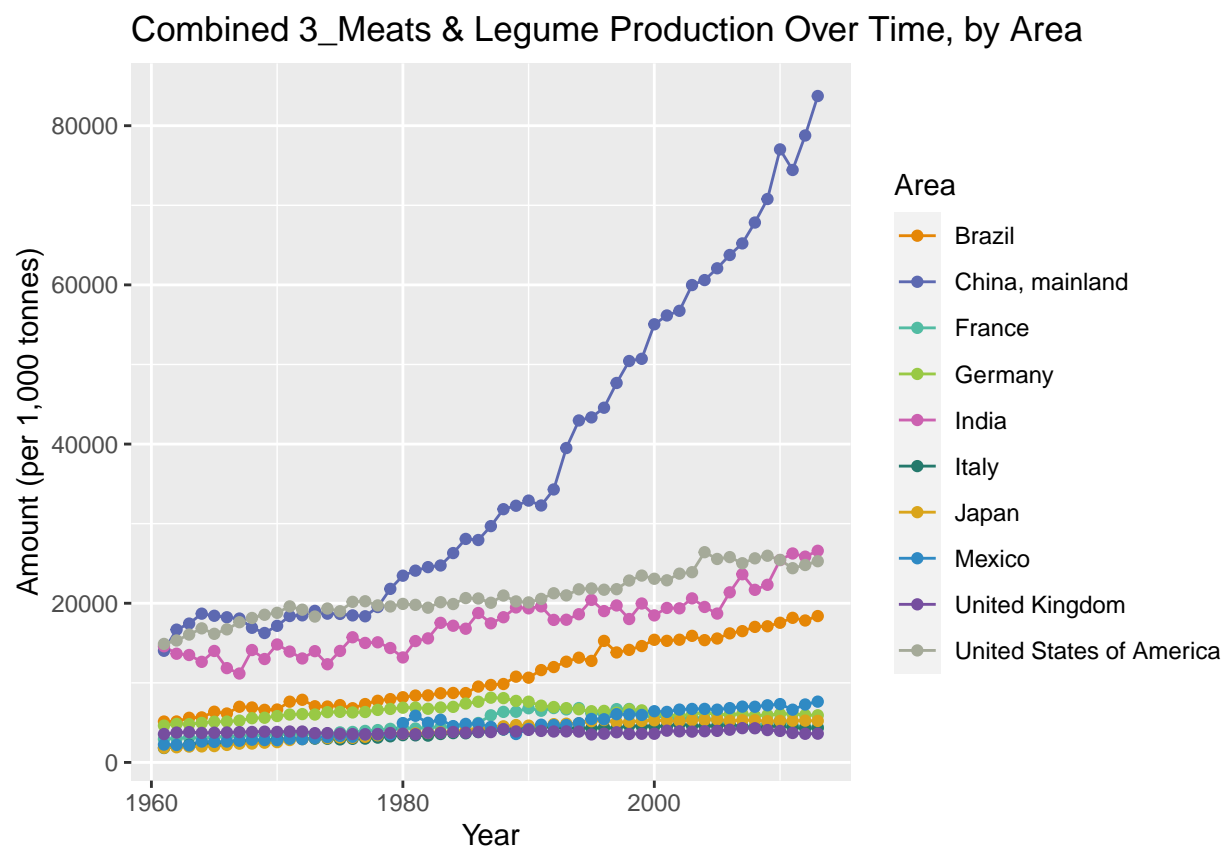
This is a lot of data, let's focus in on some of the largest producers when it comes to Legumes and/or 3\_Meats. Presumably, the Areas with the largest production have the most impact on carbon footprint shifts from one food product to another. First we will find the top 10 producers of all year's combined, and Legumes & 3\_Meats combines.

```
tot_production <- aggregate(food_dt$Amount, by=list(Area=food_dt$Area), FUN=sum)
tot_production <- top_n((tot_production[order(-tot_production$x),]), 10)
kable(tot_production, format = "markdown")
```

Area	x
China, mainland	1967145
United States of America	1110256
India	930166
Brazil	581264
Germany	333710
France	252633
Mexico	251092
Japan	214020
United Kingdom	201316
Italy	195921

```
#create a top 10 producers only subset
topareas_dt <- subset(food_dt, (Area %in% c("China, mainland", "India",
      "United States of America", "Brazil",
      "Germany", "Mexico", "France",
      "Japan", "Italy", "United Kingdom")))

#graph their total production over time
ggplot(topareas_dt, aes(x = Year, y = Amount, colour=Area)) +
  stat_summary(fun=sum, geom="line") +
  stat_summary(fun=sum, geom="point") +
  labs(title="Combined 3_Meats & Legume Production Over Time, by Area") +
  ylab('Amount (per 1,000 tonnes)') +
  xlab('Year') +
  scale_color_carto_d(name = "Area", palette = "Vivid")
```

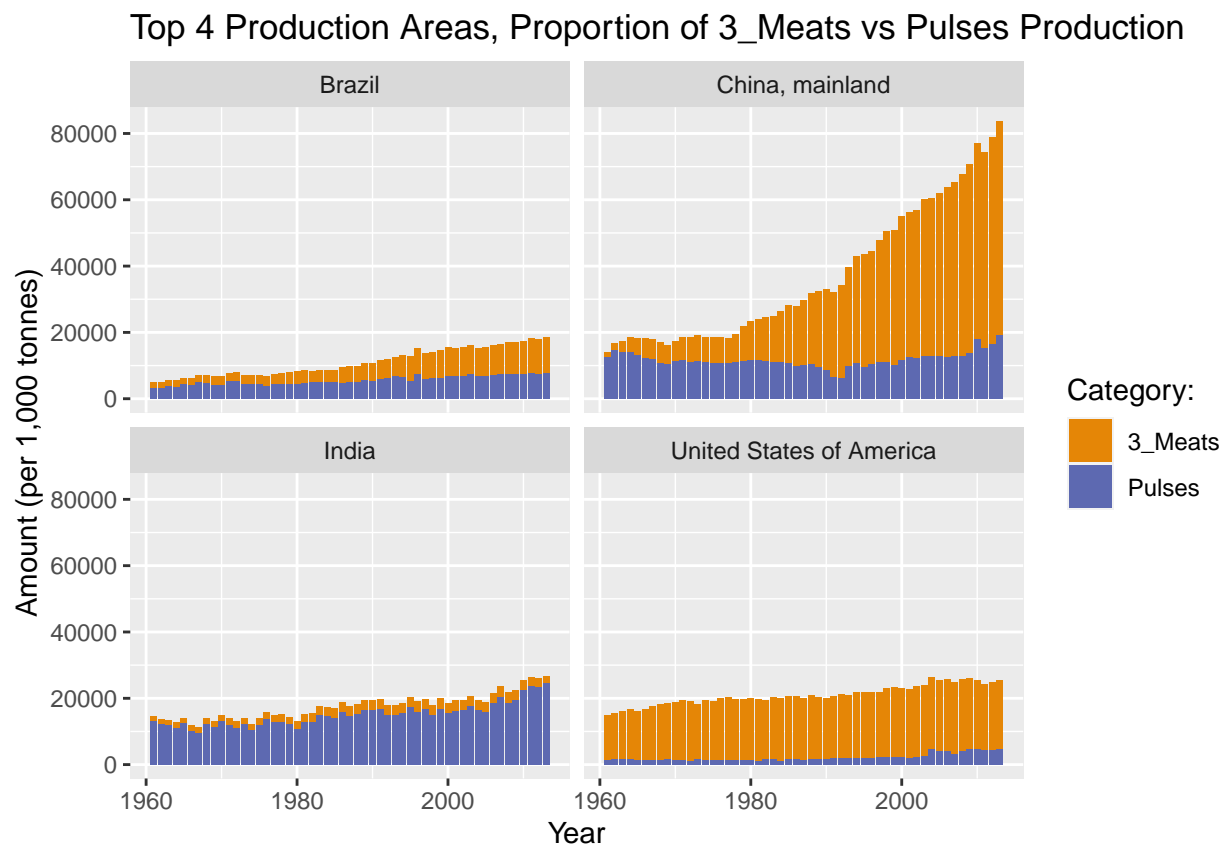


After we graph it, and see clearly there are 4 Areas with much higher production than the rest. We make a separate dataset for those countries, China, India, USA, and Brazil.

```
top4_dt <- subset(topareas_dt, (Area %in% c("China, mainland", "India",
      "United States of America", "Brazil")))
```

Let's look at each of these top 4 countries and their Legume v 3\_Meats production over time. We can see that even by sheer tonnes produced China is hugely ahead of the other countries. China's Pulses production has stayed relatively steady over time, but the 3\_Meats production has increased greatly over this time span. Closer to 1961 China was barely producing any of these 3\_Meats, but as we approach 2013 the 3\_Meats production explodes. We also see clearly that India, one of the country's with the highest vegetarian population, has a 3\_Meats production that is miniscule compared to their Pulses production.

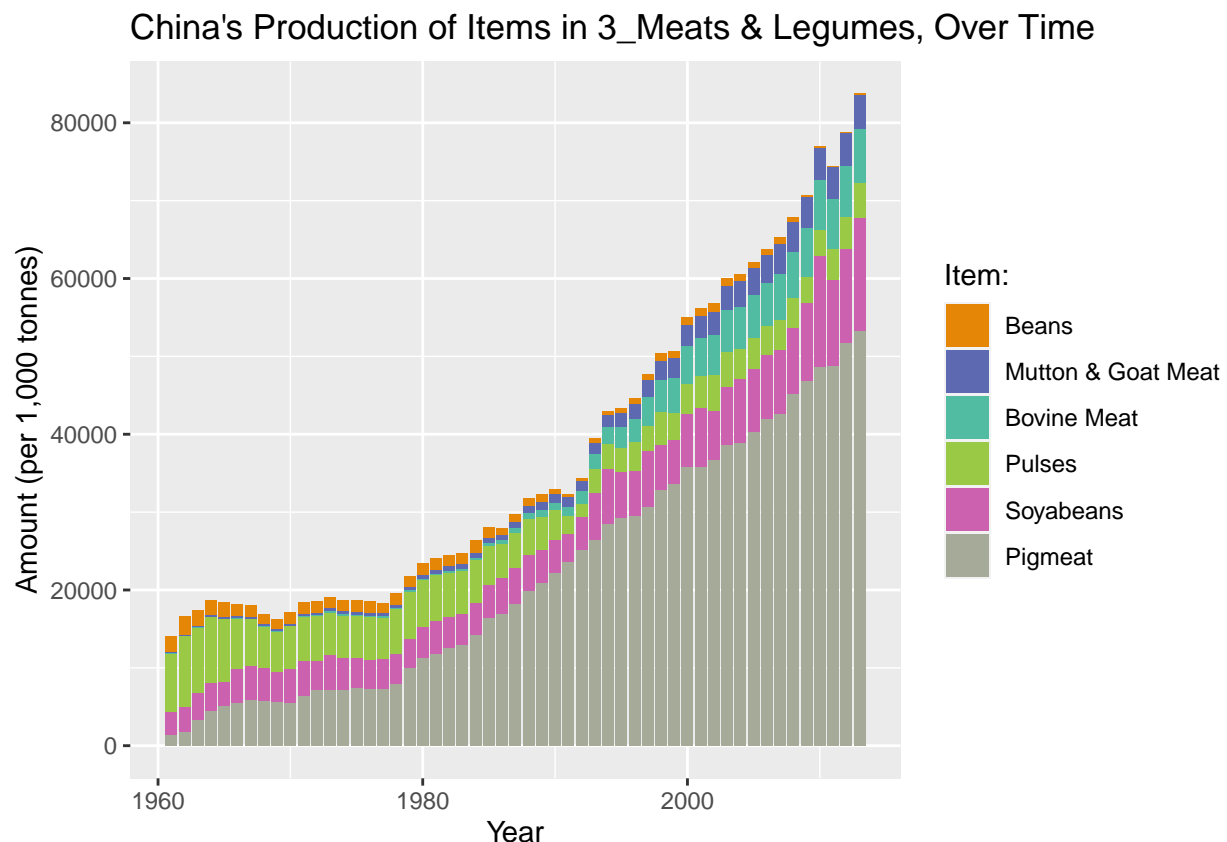
```
ggplot(top4_dt, aes(x = Year, y = Amount, fill = Category)) +
  geom_col() +
  facet_wrap(~Area) + labs(title = "test") +
  labs(title="Top 4 Production Areas, Proportion of 3_Meats vs Pulses Production") +
  ylab('Amount (per 1,000 tonnes)') +
  xlab('Year') +
  scale_fill_carto_d(name = "Category: ", palette = "Vivid")
```



Finally, let's dig in one more time to take a closer look at China, as it's presumably the country which has the most leverage in food production with regards to lowering the overall carbon footprint. We see over time that Pigmeat grows to take up a huge proportion of these high-protein food items.

```
#create China-only subset
china_dt <- subset(top4_dt, (Area == "China, mainland"))

ggplot(china_dt, aes(x = Year, y = Amount, fill = reorder(Item, Amount))) +
  geom_col() +
  labs(title="China's Production of Items in 3_Meats & Legumes, Over Time") +
  ylab('Amount (per 1,000 tonnes)') +
  xlab('Year') +
  scale_fill_carto_d(name = "Item: ", palette = "Vivid")
```



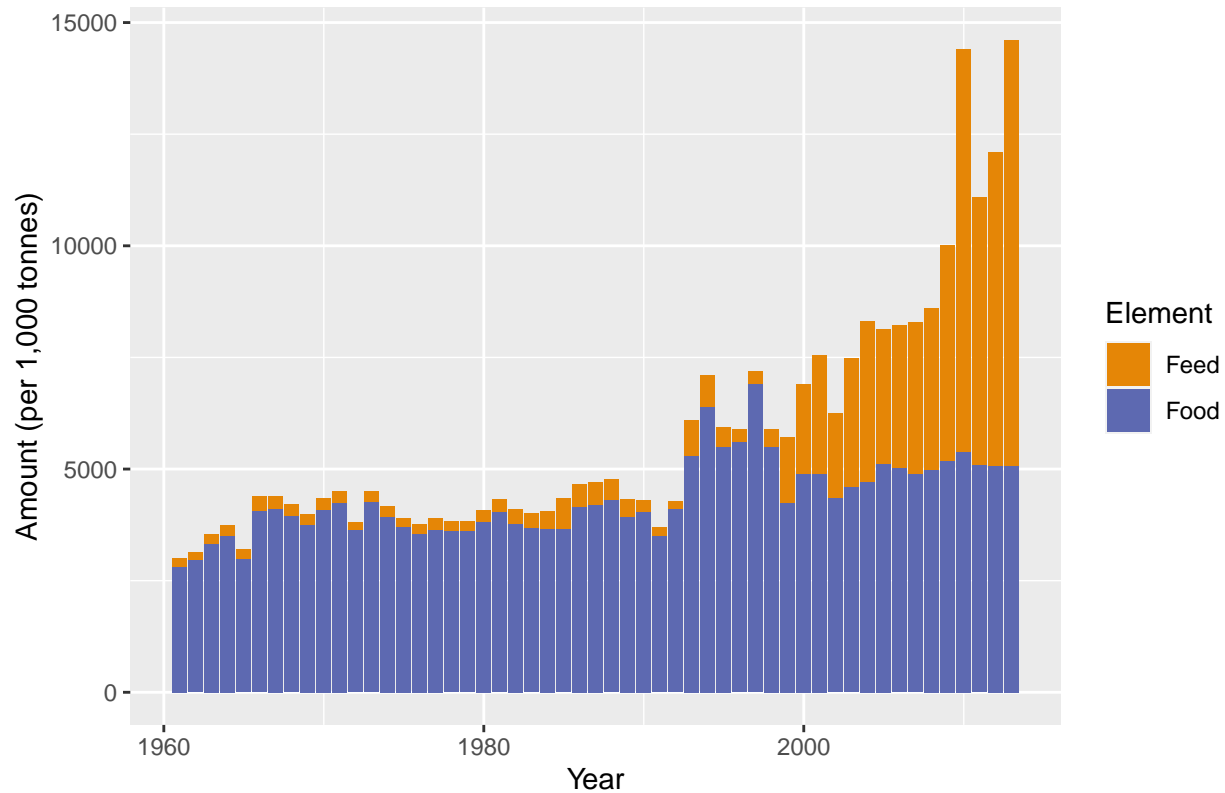
Soyabeans are the next largest category, but in investigating that further we see that growth doesn't appear to be for feeding humans ("Food") but rather for feeding animals ("Feed") - quite possibly the Pigs.

```
#create subset of China data with only Soyabeans
china_soya_dt <- subset(china_dt, (Item == "Soyabeans"))

ggplot(china_soya_dt, aes(x = Year, y = Amount, fill = Element)) +
  geom_col() +
  labs(title="China's Soyaproduction, Proportion of Food v Feed, Over Time") +
  ylab('Amount (per 1,000 tonnes)') +
  xlab('Year') +
  scale_fill_carto_d(name = "Element", palette = "Vivid")
```



## China's Soyaproduction, Proportion of Food v Feed, Over Time



## Conclusion

There is much you could do with this large dataset, and the original poster on Kaggle used it to look more at the Food v Feed aspect of the data. In doing some exploratory analysis and graphing of this dataset, we eventually identified China as a huge producer of these food items we are looking at (Legumes & 3\_Meats), and further that the amount of Pigmeat they've been producing has rapidly increased in the 1961-2013 timeframe.

From a carbon offset footprint and sheer numbers perspective, China could make a major impact in the carbon emissions by altering their food production. This could be done by reigning in their increasing production of Pigmeat and considering shifting to more environmentally-friendly protein sources such as Pulses that are consumed by humans, not necessarily animals.

If we were doing this analysis in an instance in which policy decisions were being made based on it, we'd definitely need some content experts from the food production industry to weigh in. We are positive there is nuance to this dataset that we're not aware of, and it'd be crucial to have someone more familiar with this industry on the team. For example, did domestic demand for Pigmeat drive the increase in Pigmeat production or are those exports?

## Citation

Oppenheim, D. (2017, November). Who eats the food we grow?, Version 7. Retrieved October 2, 2020 from <https://www.kaggle.com/dorbicycle/world-foodfeed-production>.

# Data607 - Assign7

Rachel Greenlee

10/5/2020

## Introduction

In this assignment I used Notepad to write the data for 3 books in 3 different formats: HTML, XML, and JSON. Next, I load each of these files into R and place in separate data frames.

## XML Data

Using the XML and RCurl packages (as xmlParse wouldn't accept my file as XML without RCurl), I access the .xml from my GitHub, parse it, and put it into a data frame.

```
library(XML)
library(RCurl)
library(kableExtra)
```

```
xml_URL <- "https://raw.githubusercontent.com/rachel-greenlee/data607_assign7/master/book_xml.xml"
xml_link <- getURL(xml_URL)
xml_data <- xmlParse(file = xml_link)
xml_df <- xmlToDataFrame(nodes = getNodeSet(xml_data, "//Book"))
kable(xml_df, format = "simple")
```

Title	Author	Genre	Story_Location	Goodreads_Rating	Year_Pu
The Lowland	Jhumpa Lahiri	Fiction	Calcutta, India	3.85	2013
An American Marriage	Tayari Jones	Fiction	Georgia, USA	3.96	2018
Leviathan Wakes	Daniel Abraham, Ty Franck	Science Fiction	The Milky Way	4.25	2011

## HTML Data

Conveniently also from the XML package, there is a readHTMLTable function that allows us to read straight into a dataframe.

```
html_URL <- "https://raw.githubusercontent.com/rachel-greenlee/data607_assign7/master/books_html.html"
html_link <- getURL(html_URL)

html_df <- readHTMLTable(html_link)
kable(html_df, format = "simple")
```

Title	Author	Genre	Story_Location	Goodreads_Rating	Year_Pu
The Lowland	Jhumpa Lahiri	Fiction	Calcutta, India	3.85	2013
An American Marriage	Tayari Jones	Fiction	Georgia, USA	3.96	2018
Leviathan Wakes	Daniel Abraham, Ty Franck	Science Fiction	The Milky Way	4.25	2011

## JSON Data

The jsonlite package made quick work of reading my JSON file into a dataframe, and I didn't need to use RCurl this time.

```
library("jsonlite")
```

```
json_df <-fromJSON("https://raw.githubusercontent.com/rachel-greenlee/data607_assign7/master/book_json.json")
kable(json_df, format = "simple")
```

Title	Author	Genre	Story_Location	Goodreads_Rating	Year_Pu
The Lowland	Jhumpa Lahiri	Fiction	Calcutta, India	3.85	2013
An American Marriage	Tayari Jones	Fiction	Georgia, USA	3.96	2018
Leviathan Wakes	Daniel Abraham, Ty Franck	Science Fiction	The Milky Way	4.25	2011

## Data Frame Comparisons

By sight these data frames all seem identical, but when we use the all.equal function we see a list of some of the differences between the three data frames.

```
all.equal(html_df, xml_df, json_df)
```

```
## [1] "Names: 1 string mismatch"
## [2] "Attributes: < names for current but not for target >"
## [3] "Attributes: < Length mismatch: comparison on first 0 components >"
## [4] "Length mismatch: comparison on first 1 components"
## [5] "Component 1: Modes: list, character"
## [6] "Component 1: names for target but not for current"
## [7] "Component 1: Attributes: < Modes: list, NULL >"
## [8] "Component 1: Attributes: < names for target but not for current >"
## [9] "Component 1: Attributes: < current is not list-like >"
## [10] "Component 1: Length mismatch: comparison on first 3 components"
## [11] "Component 1: Component 1: Lengths (3, 1) differ (string compare on first 1)"
## [12] "Component 1: Component 2: Lengths (3, 1) differ (string compare on first 1)"
## [13] "Component 1: Component 2: 1 string mismatch"
## [14] "Component 1: Component 3: Lengths (3, 1) differ (string compare on first 1)"
## [15] "Component 1: Component 3: 1 string mismatch"
```

## Conclusion

Loading these three formats into data frames was not as difficult as I had expected it would be, though I see from some online resources that as the nodes increase, the code needs to increase as well. In my case, even

before using `kable`, each data frame looks identical to the next which is pretty amazing! However, when we dig deeper using the `all.equal` function there are differences between them that could make further analysis easier/harder depending on the task.

# Assignment 9

Rachel Greenlee

10/19/2020

## Purpose

For this assignment I'll be learning how to read data in from an API with the New York Times that requires a access key in order to look at data from their bestseller book lists the week of my birthday this past year.

## Reading in the Data

First I'll need to install some packages to access, manipulate, and display the data

```
library(httr)
library(jsonlite)
library(dplyr)
library(stringr)
library(kableExtra)
```

Next I've put my API key for NYT in "nytkey" in a code chunk that is hidden, but I reference it below while using the GET function to access the API for NY Times Books.

```
#path provided by NY Times
path <- "https://api.nytimes.com/svc/books/v3/lists/overview.json"

#setting the date the bestseller list as my birthday, and a just-before the pandemic time
query <- "?published_date=2020-03-01"

#string it all together
url <- str_c(path, query, "&api-key=", nytkey, sep = "", collapse = NULL)

nyt_api <- GET(url, verbose())
```

## View List Types

Transform what we pulled in to a dataframe where we can look at the lists available to us.

```
#grab the raw import of the API and set is text with proper encoding
bestseller_lists <- content(nyt_api, "text", encoding='UTF-8')

#convert from JSON
json_list <- fromJSON(bestseller_lists, flatten = TRUE)
```

```
#drill down to the lsits we want to see and display them
bestseller_lists <- json_list$results$list
kable(bestseller_lists$list_name, format = 'markdown')
```

---

x

---

Combined Print and E-Book Fiction  
 Combined Print and E-Book Nonfiction  
 Hardcover Fiction  
 Hardcover Nonfiction  
 Trade Fiction Paperback  
 Paperback Nonfiction  
 Advice How-To and Miscellaneous  
 Childrens Middle Grade Hardcover  
 Picture Books  
 Series Books  
 Young Adult Hardcover  
 Audio Fiction  
 Audio Nonfiction  
 Business Books  
 Mass Market Monthly  
 Middle Grade Paperback Monthly  
 Young Adult Paperback Monthly

---

## Accessing the Book Lists

Finally, lets look at two bestseller lists from my birthday week this year. First, I'll pull paperback nonfiction books and a description.

```
#the 6 as it's the 6th list in the entry above
paperback_nf <- bestseller_lists[[6, "books"]]
paperback_nf <- subset(paperback_nf, select = c("title", "author",
                                                "description", "weeks_on_list"))
kable(paperback_nf, format = 'pipe')
```

title	author	description	weeks_on_list
JUST MERCY	Bryan Stevenson	A civil rights lawyer and MacArthur grant recipient's memoir of his decades of work to free innocent people condemned to death.	192
ON TYRANNY	Timothy Snyder	Twenty lessons from the 20th century about the course of tyranny.	62
THE BODY KEEPS THE SCORE	Bessel van der Kolk	How trauma affects the body and mind, and innovative treatments for recovery.	69
SAPIENS	Yuval Noah Harari	How Homo sapiens became Earth's dominant species.	92
WHITE FRAGILITY	Robin DiAngelo	Historical and cultural analyses on what causes defensive moves by white people and how this inhibits cross-racial dialogue.	80

Next, let's look at the hardcover fiction books from my birthday week and pull slightly different variables.

```
#the 1 as it's the 1st list in the entry above
hardcover_fic <- bestseller_lists[[1, "books"]]
hardcover_fic <- subset(hardcover_fic, select = c("title", "author",
                                                "publisher",
                                                "book_review_link"))

kable(hardcover_fic, format = 'pipe')
```

title	author	publisher	book_review_link
AMERICAN DIRT	Jeanine Cummins	Flatiron	<a href="https://www.nytimes.com/2020/01/17/books/review-american-dirt-jeanine-cummins.html">https://www.nytimes.com/2020/01/17/books/review-american-dirt-jeanine-cummins.html</a>
WHERE THE CRAWDADS SING	Delia Owens	Putnam	
GOLDEN IN DEATH	JD Robb	St. Martin's	
LITTLE FIRES EVERYWHERE	Celeste Ng	Penguin Press	<a href="https://www.nytimes.com/2017/09/25/books/review/little-fires-everywhere-celeste-ng.html">https://www.nytimes.com/2017/09/25/books/review/little-fires-everywhere-celeste-ng.html</a>
THE SILENT PATIENT	Alex Michaelides	Celadon	

## Conclusion

While getting the URL correct to access the API took me some time, it was fairly intuitive to move through the file and pick out the best seller lists I wanted to see after that.

For this week's assignment I had a hard time accessing the NY Times Books API, and found this vignette helpful in getting it to work: <https://cran.r-project.org/web/packages/jsonlite/vignettes/json-apis.html>

# Assignment 10

Rachel Greenlee

10/26/2020

## Purpose

For this assignment I'll be using the base code from Chapter 2 of Text Mining with R (<https://www.tidytextmining.com/sentiment.html>) to learn how to do some basic sentiment analysis with provided lexicons.

## Base code

```
#gather the needed libraries, lexicons, and Jane Austen books as our corpus
```

```
library(tidytext)
library(textdata)
library(dplyr)
library(stringr)
library(janeaustenr)
library(bigreadr)
get_sentiments("afinn")
```

**word**

<chr>

abandon

abandoned

abandons

abducted

abduction

abductions

abhor

abhorred

abhorrent

abhors

1-10 of 2,477 rows | 1-1 of 2 columns

Previous 1 2 Next

```
get_sentiments("bing")
```

**word**

<chr>

2-faces

abnormal

abolish

abominable

abominably

abominate

abomination

abort

aborted

aborts

1-10 of 6,786 rows | 1-1 of 2 columns

Previous 1 2 Next



```
get_sentiments("nrc")
```

**word**

<chr>

abacus

abandon

abandon

abandon

abandoned

abandoned

abandoned

abandoned

abandonment

abandonment

1-10 of 10,000 rows | 1-1 of 2 columns

Previous **1** [2](#) [Next](#)

```
#group by books, set variables for line number and chapter, then list out each word as it's own row
```

```
tidy_books <- austen_books() %>%  
  group_by(book) %>%  
  mutate(linenum = row_number(),  
         chapter = cumsum(str_detect(text, regex("^chapter [\\divxlc]",  
                                              ignore_case = TRUE)))) %>%  
  ungroup() %>%  
  unnest_tokens(word, text)
```

```
# using the NRC lexicon filter for the 'joy' words
```

```
nrc_joy <- get_sentiments("nrc") %>%  
  filter(sentiment == "joy")
```

```
#then join with the words (rows) from Emma that have the joy sentiment
```

```
tidy_books %>%  
  filter(book == "Emma") %>%  
  inner_join(nrc_joy) %>%  
  count(word, sort = TRUE)
```

```
## Joining, by = "word"
```

**word**

<chr>

good

young

friend

hope

happy

love

deal

found

present

kind

1-10 of 303 rows | 1-1 of 2 columns

Previous **1** [2](#) [Next](#)

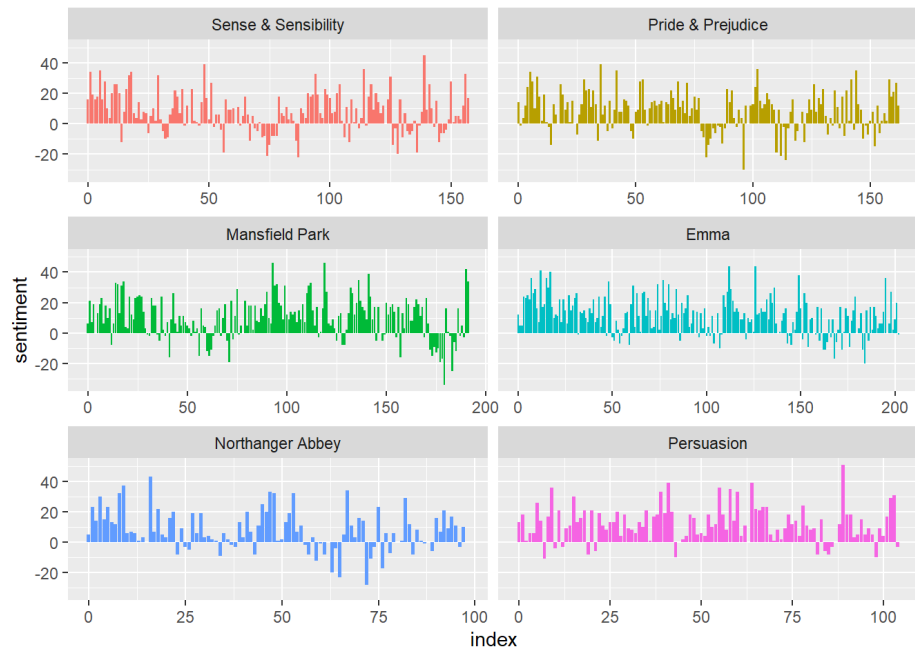
```
library(tidyr)
```

```
jane_austen_sentiment <- tidy_books %>%  
  inner_join(get_sentiments("bing")) %>%  
  count(book, index = linenumbers %/% 80, sentiment) %>%  
  spread(sentiment, n, fill = 0) %>%  
  mutate(sentiment = positive - negative)
```

```
## Joining, by = "word"
```

```
library(ggplot2)
```

```
ggplot(jane_austen_sentiment, aes(index, sentiment, fill = book)) +  
  geom_col(show.legend = FALSE) +  
  facet_wrap(~book, ncol = 2, scales = "free_x")
```



## Extending the base code

Now I will use that code above to apply this method to a new corpus and using a new lexicon.

I found a scifi stories txt file from a writer/programmer I follow, Robin Sloan. He made the file primarily from archives of the Pulp Magazine which ran from 1896 into the 1950s. Sloan used this to create a science fiction autocomplete program, which is quite fun!

dataset: <https://www.kaggle.com/jannesklaas/scifi-stories-text-corpus>

fun program: <https://www.robinsloan.com/notes/writing-with-the-machine/>

The file is incredibly large and I was not able to load up to my Github account, and further I wasn't able to run code on it without taking a subset. I had to use a bigreadr package just to read in the file, and then needed to take just a subset of the first 1,000 lines of text in order for my code to run.

```

#read in the lines
scifi_raw <- big_fread1('C:/Users/rgreenlee/drive/ds/data607/supports/scifi_text.txt', every_nlines
= 100)

#put in a dataframe
scifi_df <- as.data.frame(t(scifi_raw))

#flip so text is across rows, not columns
scifi_df <- mutate(scifi_df, V1 = scifi_df$V1)

#subset first 1,000 lines of text
scifi_sub <- scifi_df %>% slice(1:1000)

# of those first 1000 rows we cook, put each word in it's own row
scifi_sub <- scifi_sub %>%
  unnest_tokens(word, V1)

```

I found the Loughran lexicon that had a sentiment for “uncertainty” which I thought was quite appropriate for science fiction writing. I pull that in and grab the words associated with that sentiment and then join with my subset of scifi writing I made above.

```

# using the loughran lexicon filter for the sentiment "uncertainty"
loughran_uncertainty <- get_sentiments("loughran") %>%
  filter(sentiment == "uncertainty")

#then join with the words (rows) from scifi sample that have the uncertainty sentiment
scifi_sub %>%
  inner_join(loughran_uncertainty) %>%
  count(word, sort = TRUE)

```

word
<chr>
could
might
maybe
almost
may
suddenly
perhaps
probably
believe
appeared

1-10 of 155 rows | 1-1 of 2 columns

Previous 1 2 Next

The more interesting words that pop out to me from this list are “suddenly”, “believe”, and “appeared” as specific to science fiction worlds, the suspense and perception of the main character.

And for fun, let’s try the “fear” sentiment from the nrc lexicon used above.

```

# using the nrc lexicon filter for the sentiment "fear"
nrc_fear <- get_sentiments("nrc") %>%
  filter(sentiment == "fear")

#then join with the words (rows) from scifi sample that have the uncertainty sentiment
scifi_sub %>%
  inner_join(nrc_fear) %>%
  count(word, sort = TRUE)

```

word
<chr>
gun
kill
death
hell
fear
die
change
police
fire
war

1-10 of 920 rows | 1-1 of 2 columns

Previous **1** [2](#) [Next](#)

Lots of 'gun', 'kill', and 'death' in this subset of scifi stories it appears!

## Conclusion

This is an incredibly interesting way to analyze text, and it was great to see the basic code for such complex analysis. It took me the most time to get the unruly text file loaded and in the format of one word per row, the base code was fairly adaptable after that initial hurdle In my case, I feel the loughran lexicon was a bit more interesting as I associate the sentiment of uncertainty to scifi writing more than fear. It comes down to what sentiment you are expecting/looking for in a project I would guess.

## Citation for base code

Silge, J., & Robinson, D. (2017). Text mining with R: A tidy approach. Sebastopol, CA: O'Reilly.

# Project 3

Team DAREZ

10/17/2020

## Introduction

As recent as March of 2019 we are still hearing repeatedly that the demand of data scientists is not being met. The University of Pennsylvania states “Data analytics is becoming mission-critical to more and more businesses.” They quote LinkedIn co-founder Allen Blue saying, “There are very few data scientists out here passing out their resumes. Data scientists are almost all already employed, because they’re so much in demand.”

(<https://knowledge.wharton.upenn.edu/article/whats-driving-demand-data-scientist/>)

Data scientists come in many different flavors and from many different backgrounds as well, so often it comes down to having the right set of skills for the job. So what are those specific skills, other than it saying “data scientist” on your business card, that are in demand?

We set out to use our growing data science skills to begin looking at the data on what skills are in demand for data science. We chose to use 3 datasets, each looking at a slightly different angle of the question, “Which are the most valued data science skills?”

First, we looked at the 2019 Kaggle Survey on Data Science and Machine Learning. This massive, world-wide survey asks some pertinent questions of what specific tools and skills current practitioners are using. Next, we accessed a scrape of Glassdoor job postings that bring in the perspective of what hiring companies are asking for. Finally, a scrape of Indeed job postings was used to look at trends such as job locations and salaries. In a sense, these tell us what the job market expects of data scientists on a larger scale in terms of where we are willing to live and what salary is acceptable.

## Setup

Packages needed for this project include:

```
#some packages repeated, but nice to know which sections they were used in
```

```
# For Web-scrap of Indeed
```

```
library(rvest)
library(readr)
library(tidyverse)
library(DT)
library(xml2)
```

```
# For graphing and nice tables
```

```
library(ggplot2)
library(rcartocolor)
library(kableExtra)
```

```
# SQL database connection
```

```
library(odbc)
library(DBI)
```

```
#packages for glassdoor
```

```
library(tidyverse)
library(openintro)
library(readr)
library(tidyr)
library(dplyr)
library(stringr)
library(wordcloud)
library(tidytext)
library(igraph)
library(ggraph)
```

```
#package for Kaggle Survey
```

```
library(tidyverse)
library(stringr)
```

## Database Connection: Azure

We wanted to store our data in an Azure online database. We created an account and initialized the database. We set up permissions and granted each user access to the database based on their IP addresses.

We created tables for each of the data sources and imported the data to each table, saving the import packages as SSIS (.dtsx) files. The initial tables included one table for the survey information from both Indeed and Glassdoor, and three tables to store the Kaggle survey data. We also created a “read only” user account that was shared with each of the team members so that they could read the data. The “read only” connection is in the code below:

```
library(odbc) # create connection to SQL database
library(DBI) # query SQL database tables

# connect to the server
my_connection <- dbConnect(drv = odbc::odbc(),
  Driver = "SQL Server",
  server = "data607.database.windows.net",
  database = "Project3",
  uid = "Professor607",
  pwd = "TeamDAREZ#1")
```

What follows is our step-by-step method of importing the data from the database into R. The flow began by reading the data from our Indeed web-scrape into its dataframe. This process is repeated until each source is extracted from the database and in their own dataframes. It was completed using the DBI package's `dbGetQuery` function. Snippets of information containing the number of observations and the number of variables were written above the block of code that was used to read the data. Small examples of the imported data are also shown beneath the chunks.

### Importing Indeed

Reading the Indeed web-scrape data into a dataframe. It contains 1524 observations of 7 variables.

```
# read indeed data into a data frame named "df_indeed"
df_indeed <- dbGetQuery(my_connection, '
  SELECT "SurveyID", "source", "title", "company", "location", "summary", "link"
  FROM "indeed_scrape_1"
  WHERE "source" = \'indeed\'
')

head(df_indeed)
```

1
2
3
4
5
6

6 rows | 1-1 of 8 columns

### Importing Glassdoor

Reading the Glassdoor text data into a dataframe. It contains 1710 observations of 7 variables.

```
# read Glassdoor data into a data frame named "df_Glassdoor"
df_Glassdoor <- dbGetQuery(my_connection,
  SELECT "SurveyID", "source", "title", "company", "location", "summary", "link"
  FROM "indeed_scrape_1"
  WHERE "source" = \'Glassdoor\'
)

head(df_Glassdoor)
```

1
2
3
4
5
6

6 rows | 1-1 of 8 columns

## Importing Kaggle Survey Schema

Reading the Kaggle industry survey schema into a dataframe. It contains 10 observations of 35 variables.

```
# read survey schema data into a data frame named "df_survey_schema"
df_survey_schema <- dbGetQuery(my_connection,
  SELECT "2019 Kaggle Machine Learning and Data Science Survey", "Q1", "Q2", "Q3", "Q4"
  , "Q5", "Q6", "Q7", "Q8", "Q9", "Q10", "Q11", "Q12", "Q13", "Q14", "Q15", "Q16", "Q17"
  , "Q18", "Q19", "Q20", "Q21", "Q22", "Q23", "Q24", "Q25", "Q26", "Q27", "Q28", "Q29"
  , "Q30", "Q31", "Q32", "Q33", "Q34"
  FROM "survey_schema"
)

head(df_survey_schema)
```

1
2
3
4
5
6

6 rows | 1-1 of 36 columns

## Importing Kaggle Survey's MC Responses

Reading the Kaggle industry survey multiple choice responses into a dataframe. It contains 19,718 observations of 246 variables.

```
# read survey multiple choice data into a data frame named "df_survey_multi_choice"
```

```
df_survey_multi_choice <- dbGetQuery(my_connection, '
SELECT "Time from Start to Finish (seconds)"
, "Q1", "Q2", "Q2_OTHER_TEXT", "Q3", "Q4", "Q5", "Q5_OTHER_TEXT"
, "Q6", "Q7", "Q8", "Q9_Part_1", "Q9_Part_2", "Q9_Part_3", "Q9_Part_4"
, "Q9_Part_5", "Q9_Part_6", "Q9_Part_7", "Q9_Part_8", "Q9_OTHER_TEXT"
, "Q10", "Q11", "Q12_Part_1", "Q12_Part_2", "Q12_Part_3", "Q12_Part_4", "Q12_Part_5"
, "Q12_Part_6", "Q12_Part_7", "Q12_Part_8", "Q12_Part_9", "Q12_Part_10", "Q12_Part_11"
, "Q12_Part_12", "Q12_OTHER_TEXT", "Q13_Part_1", "Q13_Part_2", "Q13_Part_3", "Q13_Part_4"
, "Q13_Part_5", "Q13_Part_6", "Q13_Part_7", "Q13_Part_8", "Q13_Part_9", "Q13_Part_10"
, "Q13_Part_11", "Q13_Part_12", "Q13_OTHER_TEXT", "Q14", "Q14_Part_1_TEXT", "Q14_Part_2_TEXT"
, "Q14_Part_3_TEXT", "Q14_Part_4_TEXT", "Q14_Part_5_TEXT", "Q14_OTHER_TEXT", "Q15"
"
, "Q16_Part_1", "Q16_Part_2", "Q16_Part_3", "Q16_Part_4", "Q16_Part_5", "Q16_Part_6"
, "Q16_Part_7", "Q16_Part_8", "Q16_Part_9", "Q16_Part_10", "Q16_Part_11", "Q16_Part_12", "Q16_OTHER_TEXT"
, "Q17_Part_1", "Q17_Part_2", "Q17_Part_3", "Q17_Part_4", "Q17_Part_5", "Q17_Part_6", "Q17_Part_7", "Q17_Part_8"
, "Q17_Part_9", "Q17_Part_10", "Q17_Part_11", "Q17_Part_12", "Q17_OTHER_TEXT", "Q18_Part_1", "Q18_Part_2"
, "Q18_Part_3", "Q18_Part_4", "Q18_Part_5", "Q18_Part_6", "Q18_Part_7", "Q18_Part_8", "Q18_Part_9", "Q18_Part_10"
, "Q18_Part_11", "Q18_Part_12", "Q18_OTHER_TEXT", "Q19", "Q19_OTHER_TEXT", "Q20_Part_1", "Q20_Part_2"
, "Q20_Part_3", "Q20_Part_4", "Q20_Part_5", "Q20_Part_6", "Q20_Part_7"
, "Q20_Part_8", "Q20_Part_9", "Q20_Part_10", "Q20_Part_11", "Q20_Part_12", "Q20_OTHER_TEXT", "Q21_Part_1"
, "Q21_Part_2", "Q21_Part_3", "Q21_Part_4", "Q21_Part_5", "Q21_OTHER_TEXT", "Q22", "Q23", "Q24_Part_1"
, "Q24_Part_2", "Q24_Part_3", "Q24_Part_4", "Q24_Part_5", "Q24_Part_6", "Q24_Part_7", "Q24_Part_8"
, "Q24_Part_9", "Q24_Part_10", "Q24_Part_11", "Q24_Part_12", "Q24_OTHER_TEXT", "Q25_Part_1", "Q25_Part_2"
, "Q25_Part_3", "Q25_Part_4", "Q25_Part_5", "Q25_Part_6", "Q25_Part_7", "Q25_Part_8", "Q25_OTHER_TEXT"
, "Q26_Part_1", "Q26_Part_2", "Q26_Part_3", "Q26_Part_4", "Q26_Part_5", "Q26_Part_6", "Q26_Part_7"
, "Q26_OTHER_TEXT", "Q27_Part_1", "Q27_Part_2", "Q27_Part_3", "Q27_Part_4", "Q27_Part_5", "Q27_Part_6"
, "Q27_OTHER_TEXT", "Q28_Part_1", "Q28_Part_2", "Q28_Part_3", "Q28_Part_4", "Q28_Part_5", "Q28_Part_6"
, "Q28_Part_7", "Q28_Part_8", "Q28_Part_9", "Q28_Part_10", "Q28_Part_11", "Q28_Part_12", "Q28_OTHER_TEXT"
, "Q29_Part_1", "Q29_Part_2", "Q29_Part_3", "Q29_Part_4", "Q29_Part_5", "Q29_Part_6", "Q29_Part_7", "Q29_Part_8"
, "Q29_Part_9", "Q29_Part_10", "Q29_Part_11", "Q29_Part_12", "Q29_OTHER_TEXT"
, "Q30_Part_1", "Q30_Part_2", "Q30_Part_3", "Q30_Part_4", "Q30_Part_5", "Q30_Part_6", "Q30_Part_7"
, "Q30_Part_8", "Q30_Part_9", "Q30_Part_10", "Q30_Part_11", "Q30_Part_12", "Q30_OTHER_TEXT"
, "Q31_Part_1", "Q31_Part_2", "Q31_Part_3", "Q31_Part_4", "Q31_Part_5", "Q31_Part_6", "Q31_Part_7"
, "Q31_Part_8", "Q31_Part_9", "Q31_Part_10", "Q31_Part_11", "Q31_Part_12", "Q31_OTHER_TEXT"
, "Q32_Part_1", "Q32_Part_2", "Q32_Part_3", "Q32_Part_4", "Q32_Part_5", "Q32_Part_6", "Q32_Part_7"
, "Q32_Part_8", "Q32_Part_9", "Q32_Part_10", "Q32_Part_11", "Q32_Part_12", "Q32_OTHER_TEXT"
, "Q33_Part_1", "Q33_Part_2", "Q33_Part_3", "Q33_Part_4", "Q33_Part_5", "Q33_Part_6"
, "Q33_Part_7", "Q33_Part_8", "Q33_Part_9", "Q33_Part_10", "Q33_Part_11", "Q33_Part_12", "Q33_OTHER_TEXT"
, "Q34_Part_1", "Q34_Part_2", "Q34_Part_3", "Q34_Part_4", "Q34_Part_5", "Q34_Part_6"
, "Q34_Part_7", "Q34_Part_8", "Q34_Part_9", "Q34_Part_10", "Q34_Part_11", "Q34_Part_12", "Q34_OTHER_TEXT"
FROM "multiple_choice_responses"
')
```

```
head(df_survey_multi_choice)
```



1
2
3
4
5
6

6 rows | 1-1 of 247 columns

## Importing Other Kaggle Responses

Reading the Kaggle industry survey responses listed as "other" into a dataframe. It contains 19,718 observations of 28 variables.

```
# read survey other text response (free form) data into a data frame named "df_survey_other_text"

df_survey_other_text <- dbGetQuery(my_connection,
  SELECT "Q12_OTHER_TEXT", "Q13_OTHER_TEXT", "Q14_OTHER_TEXT", "Q14_Part_1_T
EXT", "Q14_Part_2_TEXT"
  , "Q14_Part_3_TEXT", "Q14_Part_4_TEXT", "Q14_Part_5_TEXT", "Q16_OTHER_TEXT", "Q17
_OTHER_TEXT", "Q18_OTHER_TEXT"
  , "Q19_OTHER_TEXT", "Q20_OTHER_TEXT", "Q21_OTHER_TEXT"
  , "Q24_OTHER_TEXT", "Q25_OTHER_TEXT", "Q26_OTHER_TEXT", "Q27_OTHER_TEXT", "
Q28_OTHER_TEXT", "Q29_OTHER_TEXT"
  , "Q2_OTHER_TEXT", "Q30_OTHER_TEXT", "Q31_OTHER_TEXT", "Q32_OTHER_TEXT", "Q
33_OTHER_TEXT", "Q34_OTHER_TEXT"
  , "Q5_OTHER_TEXT", "Q9_OTHER_TEXT"
  FROM "other_text_responses"
)

head(df_survey_other_text)
```

1
2
3
4
5
6

6 rows | 1-1 of 29 columns

## Analysis

First, we look at the 2019 Kaggle Survey on Data Science and Machine Learning. This massive, world-wide survey asks some pertinent questions of what specific tools and skills current practitioners are using.

Next, we access a scrape of Glassdoor job postings that bring in the perspective of what hiring companies are asking for.

Finally, a scrape of Indeed job postings is used to look at trends such as job locations and salaries as in a sense, these tell us what the job market expects of data scientists on a larger scale in terms of where we are willing to live and what salary is acceptable

## Kaggle 2019 Survey Data

For three weeks from October 8 to October 28 of 2019 Kaggle conducted an industry-wide survey with 19,717 responses from individuals in the data science and machine learning fields. Responses were aggregated into a file with data from 171 countries and any country that had less than 50 responses was placed in an “other” category to preserve anonymity of the respondents. The data collected gives a comprehensive look at the state of the industry by asking questions about their education, workplace habits, tools in use and more. It was formatted as a multiple-choice survey wherein responses could be categorized into one of several options for each question.

We selected this data set to give us the most comprehensive look at the skills and general expectations of the data science and machine learning job market. We decided to select questions that were most relevant to our interests. Here are a few examples:

- \* What programming languages do you use on a regular basis?
- \* What programming language would you recommend an aspiring data scientist to learn first?
- \* Which of the following relational database products do you use on a regular basis?

These were used in the analysis but prior to performing any computations, the data required cleaning. Due to the way the survey was shuffled to preserve anonymity, the responses needed to be organized into a format that was usable to analyze. With the multiple-choice responses uploaded to our database and GitHub repository as a spreadsheet (.csv) we began to organize and clean the data.

## Tidying and Munging the Kaggle Data

The 2019 Kaggle survey contains thirty-four questions and the corresponding recorded responses. We decided to make a dataframe for each question and response. The code to accomplish this task is below for each dataframe. It is formatted for ease of viewing to see how each question was cleaned and summarized.

Q1 - What is your age (# years)?

```
# create dataframe just for question 1

q_01 <- dbGetQuery(my_connection,'
SELECT "Q1"
FROM "multiple_choice_responses"
WHERE "Q1" != \'\'
AND "Q1" NOT LIKE \'%?%%\' -- remove the record with the question
')

q_01 <- data.frame(q_01) %>%
  gather(key="question", value="response") %>%
  group_by(response) %>%
  summarize(count_reponse = n())

q_01
```

response

<chr>

18-21

22-24

25-29

30-34

35-39

40-44

45-49

50-54

55-59

60-69

1-10 of 11 rows | 1-1 of 2 columns

Previous 1 2 Next

Q2 - What is your gender?

*# create dataframe just for question 2*

```
q_02 <- dbGetQuery(my_connection, '
SELECT "Q2"
FROM "multiple_choice_responses"
WHERE "Q2" != \"\"
AND "Q2" NOT LIKE \"%?%\" -- remove the record with the question
')

q_02 <- data.frame(q_02) %>%
  gather(key="question", value="response") %>%
  group_by(response) %>%
  summarize(count_reponse = n())

q_02
```

response

<chr>

Female

Male

Prefer not to say

Prefer to self-describe

4 rows | 1-1 of 2 columns

Q3 - In which country do you currently reside?

*# create dataframe just for question 3*

```
q_03 <- dbGetQuery(my_connection, '
SELECT "Q3"
FROM "multiple_choice_responses"
WHERE "Q3" != \"\"
AND "Q3" NOT LIKE \"%?%\" -- remove the record with the question
')

q_03 <- data.frame(q_03) %>%
  gather(key="question", value="response") %>%
  group_by(response) %>%
  summarize(count_reponse = n())

q_03
```

response

<chr>

Algeria

Argentina

Australia

Austria

Bangladesh

Belarus

Belgium

Brazil

Canada

Chile

1-10 of 59 rows | 1-1 of 2 columns

Previous 1 2 Next

Q4 - What is the highest level of formal education that you have attained or plan to attain within the next 2 years?

# create dataframe just for question 4

```
q_04 <- dbGetQuery(my_connection, '
SELECT "Q4"
FROM "multiple_choice_responses"
WHERE "Q4" != \"\"
AND "Q4" NOT LIKE \"%?%\" -- remove the record with the question
')
```

```
q_04 <- data.frame(q_04) %>%
gather(key="question", value="response") %>%
group_by(response) %>%
summarize(count_reponse = n())
```

q\_04

response

<chr>

Bachelor's degree

Doctoral degree

I prefer not to answer

Master's degree

No formal education past high school

Professional degree

Some college/university study without earning a bachelor's degree

7 rows | 1-1 of 2 columns

Q5 - Select the title most similar to your current role (or most recent title if retired):

```
# create dataframe just for question 5
```

```
q_05 <- dbGetQuery(my_connection, '  
  SELECT "Q5"  
    FROM "multiple_choice_responses"  
   WHERE "Q5" != \"\"  
     AND "Q5" NOT LIKE \"%?%\" -- remove the record with the question  
' )  
  
q_05 <- data.frame(q_05) %>%  
  gather(key="question", value="response") %>%  
  group_by(response) %>%  
  summarize(count_reponse = n())  
  
q_05
```

**response**

<chr>

Business Analyst

Data Analyst

Data Engineer

Data Scientist

DBA/Database Engineer

Not employed

Other

Product/Project Manager

Research Scientist

Select the title most similar to your current role (or most recent title if retired): -

Selected Choice

1-10 of 13 rows | 1-1 of 2 columns

Previous **1** 2 Next

Q6 - What is the size of the company where you are employed?

```
# create dataframe just for question 6
```

```
q_06 <- dbGetQuery(my_connection, '  
  SELECT "Q6"  
    FROM "multiple_choice_responses"  
   WHERE "Q6" != \"\"  
     AND "Q6" NOT LIKE \"%?%\" -- remove the record with the question  
' )  
  
q_06 <- data.frame(q_06) %>%  
  gather(key="question", value="response") %>%  
  group_by(response) %>%  
  summarize(count_reponse = n())  
  
q_06
```

**response**

&lt;chr&gt;

&gt; 10,000 employees

0-49 employees

1000-9,999 employees

250-999 employees

50-249 employees

5 rows | 1-1 of 2 columns

Q7 - Approximately how many individuals are responsible for data science workloads at your place of business?

*# create dataframe just for question 7*

```
q_07 <- dbGetQuery(my_connection, '
SELECT "Q7"
FROM "multiple_choice_responses"
WHERE "Q7" != \'\'
AND "Q7" NOT LIKE \'%?%%\' -- remove the record with the question
')
```

```
q_07 <- data.frame(q_07) %>%
gather(key="question", value="response") %>%
group_by(response) %>%
summarize(count_reponse = n())
```

q\_07

**response**

&lt;chr&gt;

0

1-2

10-14

15-19

20+

3-4

5-9

7 rows | 1-1 of 2 columns

Q8 - Does your current employer incorporate machine learning methods into their business?

*# create dataframe just for question 8*

```
q_08 <- dbGetQuery(my_connection, '
SELECT "Q8"
FROM "multiple_choice_responses"
WHERE "Q8" != \'\'
AND "Q8" NOT LIKE \'%?%%\' -- remove the record with the question
')
```

```
q_08 <- data.frame(q_08) %>%
gather(key="question", value="response") %>%
group_by(response) %>%
summarize(count_reponse = n())
```

q\_08

**response**

&lt;chr&gt;

I do not know

No (we do not use ML methods)

We are exploring ML methods (and may one day put a model into production)

We have well established ML methods (i.e., models in production for more than 2 years)

We recently started using ML methods (i.e., models in production for less than 2 years)

We use ML methods for generating insights (but do not put working models into production)

6 rows | 1-1 of 2 columns

Q9 - Select any activities that make up an important part of your role at work:  
(Select all that apply) - Selected Choice

# create dataframe just for question 9

```
q_09 <- dbGetQuery(my_connection, '
SELECT "Q9_Part_1", "Q9_Part_2", "Q9_Part_3", "Q9_Part_4"
, "Q9_Part_5", "Q9_Part_6", "Q9_Part_7", "Q9_Part_8"
FROM "multiple_choice_responses"
WHERE "Q9_Part_1" NOT LIKE \'%selected Choice%\' -- remove the record with the question
s
')
```

```
q_09 <- data.frame(q_09) %>%
gather(key="question", value="response") %>%
group_by(response) %>%
summarize(count_reponse = n())
```

q\_09

**response**

&lt;chr&gt;

Analyze and understand data to influence product or business decisions

Build and/or run a machine learning service that operationally improves my product or workflows

Build and/or run the data infrastructure that my business uses for storing, analyzing, and operationalizing data

Build prototypes to explore applying machine learning to new areas

Do research that advances the state of the art of machine learning

Experimentation and iteration to improve existing ML models

None of these activities are an important part of my role at work

Other

9 rows | 1-1 of 2 columns

Q10 - What is your current yearly compensation (approximate \$USD)?

```
# create dataframe just for question 10
```

```
q_10 <- dbGetQuery(my_connection, '  
  SELECT "Q10"  
    FROM "multiple_choice_responses"  
   WHERE "Q10" != \"\\\"  
     AND "Q10" NOT LIKE \"%?%\" -- remove the record with the question  
' )  
  
q_10 <- data.frame(q_10) %>%  
  gather(key="question", value="response") %>%  
  group_by(response) %>%  
  summarize(count_reponse = n())  
  
q_10
```

**response**

<chr>

\$0-999

> \$500,000

1,000-1,999

10,000-14,999

100,000-124,999

125,000-149,999

15,000-19,999

150,000-199,999

2,000-2,999

20,000-24,999

1-10 of 25 rows | 1-1 of 2 columns

Previous **1** 2 Next

Q11 - Approximately how much money have you spent on machine learning and/or cloud computing products at your work in the past 5 years?

```
# create dataframe just for question 11
```

```
q_11 <- dbGetQuery(my_connection, '  
  SELECT "Q11"  
    FROM "multiple_choice_responses"  
   WHERE "Q11" != \"\\\"  
     AND "Q11" NOT LIKE \"%?%\" -- remove the record with the question  
' )  
  
q_11 <- data.frame(q_11) %>%  
  gather(key="question", value="response") %>%  
  group_by(response) %>%  
  summarize(count_reponse = n())  
  
q_11
```



response

<chr>

\$0 (USD)

\$1-\$99

\$10,000-\$99,999

\$100-\$999

\$1000-\$9,999

> \$100,000 (\$USD)

6 rows | 1-1 of 2 columns

Q12 - Who/what are your favorite media sources that report on data science topics? (Select all that apply) - Selected Choice

# create dataframe just for question 12

```
q_12 <- dbGetQuery(my_connection, '
SELECT "Q12_Part_1", "Q12_Part_2", "Q12_Part_3", "Q12_Part_4"
      , "Q12_Part_5", "Q12_Part_6", "Q12_Part_7", "Q12_Part_8"
      , "Q12_Part_9", "Q12_Part_10", "Q12_Part_11"
      , "Q12_Part_12"
FROM "multiple_choice_responses"
WHERE "Q12_Part_1" NOT LIKE \'%selected Choice%\'' -- remove the record with the question
')

q_12 <- data.frame(q_12) %>%
  gather(key="question", value="response") %>%
  group_by(response) %>%
  summarize(count_reponse = n())

q_12 <- q_12[-1,]
q_12
```

response

<chr>

Blogs (Towards Data Science, Medium, Analytics Vidhya, KDnuggets etc)

Course Forums (forums.fast.ai, etc)

Hacker News (<https://news.ycombinator.com/>)

Journal Publications (traditional publications, preprint journals, etc)

Kaggle (forums, blog, social media, etc)

None

Other

Podcasts (Chai Time Data Science, Linear Digressions, etc)

Reddit (r/machinelearning, r/datascience, etc)

Slack Communities (ods.ai, kagglenoobs, etc)

1-10 of 12 rows | 1-1 of 2 columns

Previous 1 2 Next

Q13 - On which platforms have you begun or completed data science courses? (Select all that apply) - Selected Choice

```
# create dataframe just for question 13
```

```
q_13 <- dbGetQuery(my_connection,'
SELECT "Q13_Part_1", "Q13_Part_2", "Q13_Part_3", "Q13_Part_4", "Q13_Part_5"
, "Q13_Part_6", "Q13_Part_7", "Q13_Part_8", "Q13_Part_9", "Q13_Part_10"
, "Q13_Part_11", "Q13_Part_12"
FROM "multiple_choice_responses"
wHERE "Q13_Part_1" NOT LIKE \'%selected Choice%\' -- remove the record with the questio
ns
')

q_13 <- data.frame(q_13) %>%
gather(key="question", value="response") %>%
group_by(response) %>%
summarize(count_reponse = n())

q_13 <- q_13[-1,]
q_13
```

#### response

<chr>

Coursera

DataCamp

DataQuest

edX

Fast.ai

Kaggle Courses (i.e. Kaggle Learn)

LinkedIn Learning

None

Other

Udacity

1-10 of 12 rows | 1-1 of 2 columns

Previous **1** **2** Next

Q14 - What is the primary tool that you use at work or school to analyze data?  
(Include text response) - Selected Choice

```
# create dataframe just for question 14
```

```
q_14 <- dbGetQuery(my_connection,'
SELECT "Q14"
FROM "multiple_choice_responses"
wHERE "Q14" != \'\'
AND "Q14" NOT LIKE \'%?%%\' -- remove the record with the question
')

q_14 <- data.frame(q_14) %>%
gather(key="question", value="response") %>%
group_by(response) %>%
summarize(count_reponse = n())

q_14
```

response

<chr>

Advanced statistical software (SPSS, SAS, etc.)

Basic statistical software (Microsoft Excel, Google Sheets, etc.)

Business intelligence software (Salesforce, Tableau, Spotfire, etc.)

Cloud-based data software & APIs (AWS, GCP, Azure, etc.)

Local development environments (RStudio, JupyterLab, etc.)

Other

6 rows | 1-1 of 2 columns

Q15 - How long have you been writing code to analyze data (at work or at school)?

# create dataframe just for question 15

```
q_15 <- dbGetQuery(my_connection, '
SELECT "Q15"
FROM "multiple_choice_responses"
WHERE "Q15" != \"\"
AND "Q15" NOT LIKE \"%?%\" -- remove the record with the question
')
```

```
q_15 <- data.frame(q_15) %>%
gather(key="question", value="response") %>%
group_by(response) %>%
summarize(count_reponse = n())
```

q\_15

response

<chr>

< 1 years

1-2 years

10-20 years

20+ years

3-5 years

5-10 years

I have never written code

7 rows | 1-1 of 2 columns

Q16 - Which of the following integrated development environments (IDE's) do you use on a regular basis? (Select all that apply) - Selected Choice

```
# create dataframe just for question 16
```

```
q_16 <- dbGetQuery(my_connection,'
SELECT "Q16_Part_1", "Q16_Part_2", "Q16_Part_3", "Q16_Part_4", "Q16_Part_5"
      , "Q16_Part_6", "Q16_Part_7", "Q16_Part_8", "Q16_Part_9", "Q16_Part_10"
      , "Q16_Part_11", "Q16_Part_12"
FROM "multiple_choice_responses"
WHERE "Q16_Part_1" NOT LIKE \'%selected Choice%\'' -- remove the record with the question
')

q_16 <- data.frame(q_16) %>%
  gather(key="question", value="response") %>%
  group_by(response) %>%
  summarize(count_reponse = n())

q_16 <- q_16[-1,]
q_16
```

### response

<chr>

Notepad++

Spyder

Sublime Text

Vim / Emacs

Atom

MATLAB

PyCharm

RStudio

Visual Studio / Visual Studio Code

Jupyter (JupyterLab, Jupyter Notebooks, etc)

1-10 of 12 rows | 1-1 of 2 columns

Previous 1 2 Next

Q17 - Which of the following hosted notebook products do you use on a regular basis? (Select all that apply) - Selected Choice

```
# create dataframe just for question 17
```

```
q_17 <- dbGetQuery(my_connection,'
SELECT "Q17_Part_1", "Q17_Part_2", "Q17_Part_3", "Q17_Part_4", "Q17_Part_5"
      , "Q17_Part_6", "Q17_Part_7", "Q17_Part_8", "Q17_Part_9", "Q17_Part_10"
      , "Q17_Part_11", "Q17_Part_12"
FROM "multiple_choice_responses"
WHERE "Q17_Part_1" NOT LIKE \'%selected Choice%\'' -- remove the record with the question
')

q_17 <- data.frame(q_17) %>%
  gather(key="question", value="response") %>%
  group_by(response) %>%
  summarize(count_reponse = n())

q_17 <- q_17[-1,]
q_17
```

response
<chr>
Binder / JupyterHub
FloydHub
Google Cloud Notebook Products (AI Platform, Datalab, etc)
Google Colab
IBM Watson Studio
Kaggle Notebooks (Kernels)
Microsoft Azure Notebooks
Paperspace / Gradient
AWS Notebook Products (EMR Notebooks, Sagemaker Notebooks, etc)
Code Ocean

1-10 of 12 rows | 1-1 of 2 columns

Previous 1 2 Next

In this review of the Kaggle survey, we were able to summarize the selected multiple-choice responses. This allows us to see what respondents in the industry expect from new data scientists and it provided evidence of what to expect in the future.

## Kaggle Analysis

Aspiring data scientist and current data science professionals may be interested in knowing what the industry expectations are as they are constantly changing. As new software and analysis methods appear, we make changes to our processes and create new programs, develop new models, and improve our data systems. To understand these trends, we took a well-rounded approach.

The data we selected provides a snapshot in time of those industry skills, tools, and expectations. We have three sources to gather information from; a large, global survey called the Kaggle Survey, a web-scrape of current jobs offered on Indeed, and a textual analysis of another web-scrape compiled from Glassdoor. We begin with the Kaggle survey.

### Dataset:Multiple Choice Responses

```
Data <- dbGetQuery(my_connection,"Select * FROM dbo.multiple_choice_responses")
```

Converting to a tibble

```
Data_Tibble <-as_tibble(Data)
```

Filtering out non data roles Since our research question is “What are the most sought after skills for data scientists?” we are only interested in looking at the information for data roles.

```
DataScientists <- filter (Data_Tibble,Q5 == "Data Scientist" | Q5 == "Data Analyst" | Q5 == "Data Engineer")
```

### Responses

We are going to store the responses for **questions 18-34** in separate data frames (q\_18,q\_19,q\_20 and so on till q\_34).However, we have excluded questions that are not relevant for our analysis (non skills based questions)

Question 18:What programming languages do you use on a regular basis?

```
q_18 <- DataScientists %>%
  select(starts_with("Q18")& !"Q18_OTHER_TEXT")
q_18 <-q_18 %>%
  gather(Q,Response) %>%
  group_by(Response) %>%
  summarize(count_response = n())
```

Question 19: What programming language would you recommend an aspiring data scientist to learn first?

```
q_19 <- DataScientists %>%
  select(starts_with("Q19")& !"Q19_OTHER_TEXT")
q_19 <- q_19 %>%
  gather(Q,Response) %>%
  group_by(Response) %>%
  summarize(count_response = n())
```

Question 20: What data visualization libraries or tools do you use on a regular basis?

```
q_20 <- DataScientists %>%
  select(starts_with("Q20")& !"Q20_OTHER_TEXT")

q_20 <-q_20 %>%
  gather(Q20,response) %>%
  group_by(response) %>%
  summarize(count_response = n())
```

Question 24: Which of the following ML algorithms do you use on a regular basis?

```
q_24 <- DataScientists %>%
  select(starts_with("Q24")& !"Q24_OTHER_TEXT")

q_24 <-q_24 %>%
  gather(Q,response) %>%
  group_by(response) %>%
  summarize(count_response = n())
```

Question 25: Which categories of ML tools do you use on a regular basis?

```
q_25 <- DataScientists %>%
  select(starts_with("Q25")& !"Q25_OTHER_TEXT")
q_25 <-q_25 %>%
  gather(Q,response) %>%
  group_by(response) %>%
  summarize(count_response = n())
```

Question 26: Which categories of computer vision methods do you use on a regular basis?

```
q_26 <- DataScientists %>%
  select(starts_with("Q26")& !"Q26_OTHER_TEXT")
q_26 %>%
  gather(Q26,response) %>%
  group_by(response) %>%
  summarize(count_response = n())
```

response
<chr>
General purpose image/video tools (PIL, cv2, skimage, etc)
Generative Networks (GAN, VAE, etc)
Image classification and other general purpose networks (VGG, Inception, ResNet, ResNeXt, NASNet, EfficientNet, etc)
Image segmentation methods (U-Net, Mask R-CNN, etc)
None
Object detection methods (YOLOv3, RetinaNet, etc)
Other

8 rows | 1-1 of 2 columns

Question 27: Which of the following natural language processing (NLP) methods do you use on a regular basis?

```
q_27 <- DataScientists %>%
  select(starts_with("Q27")& !"Q27_OTHER_TEXT")
q_27 %>%
  gather(Q27,response) %>%
  group_by(response) %>%
  summarize(count_response = n())
```

**response**

<chr>

Contextualized embeddings (ELMo, CoVe)
Encoder-decoder models (seq2seq, vanilla transformers)
None
Other
Transformer language models (GPT-2, BERT, XLnet, etc)
Word embeddings/vectors (GLoVe, fastText, word2vec)

7 rows | 1-1 of 2 columns

Question 28: Which of the following machine learning frameworks do you use on a regular basis?

```
q_28 <- DataScientists %>%
  select(starts_with("Q28")& !"Q28_OTHER_TEXT")
q_28 <-q_28 %>%
  gather(Q,response) %>%
  group_by(response) %>%
  summarize(count_response = n())
```

Question 29: Which of the following cloud computing platforms do you use on a regular basis?

```
q_29 <- DataScientists %>%
  select(starts_with("Q29")& !"Q29_OTHER_TEXT")

q_29 <-q_29 %>%
  gather(Q29,response) %>%
  group_by(response) %>%
  summarize(count_response = n())
```

Question 30: Which specific cloud computing products do you use on a regular basis?

```
q_30 <- DataScientists %>%
  select(starts_with("Q30")& !"Q30_OTHER_TEXT")
q_30 <- q_30 %>%
  gather(Q30,response) %>%
  group_by(response) %>%
  summarize(count_response = n())
```

Question 31: Which specific big data / analytics products do you use on a regular basis?

```
q_31 <- DataScientists %>%
  select(starts_with("Q31")& !"Q31_OTHER_TEXT")

q_31 <-
q_31 %>%
  gather(Q31,response) %>%
  group_by(response) %>%
  summarize(count_response = n())
```

Question 32: Which of the following machine learning products do you use on a regular basis?

```
q_32 <- DataScientists %>%
  select(starts_with("Q32") & !"Q32_OTHER_TEXT")
q_32 <- q_32 %>%
  gather(Q, response) %>%
  group_by(response) %>%
  summarize(count_response = n())
```

Question 33: Which automated machine learning tools (or partial AutoML tools) do you use on a regular basis?

```
q_33 <- DataScientists %>%
  select(starts_with("Q33") & !"Q33_OTHER_TEXT")
q_33 <- q_33 %>%
  gather(Q33, response) %>%
  group_by(response) %>%
  summarize(count_response = n())
```

Question 34: Which of the following relational database products do you use on a regular basis?

```
q_34 <- DataScientists %>%
  select(starts_with("Q34") & !"Q34_OTHER_TEXT")
q_34 <- q_34 %>%
  gather(Q, response) %>%
  group_by(response) %>%
  summarize(count_response = n())
```

In this review of the Kaggle survey, we were able to summarize the selected multiple-choice responses. This allows us to see what respondents in the industry expect from aspiring and existing data scientists. It also provided evidence of what to expect in the future.

## Most cited skills

It looks like the majority of the skills-based questions can be categorized into the following areas- 1. **Programming Languages**: Questions 18 and 19 2. **Data Visualization**: Question 20 3. **ML Tools and Products**: Questions 25, 32 and 33 4. **ML Framework**: Question 28 5. **Big Data**: Question 31 6. **Cloud Platforms**: Questions 29 and 30 7. **Relational Databases**: Question 34 8. **Computer Vision**: Question 26 9. **NLP**: Question 27

We are particularly interested in understanding what the most cited tools or skills were for #1-6 above.

### 1. Programming Languages

We will combine the individual data frame for each question first before plotting the data.

```
PL <- rbind(q_18, q_19)
PL <- PL %>%
  group_by(Response) %>%
  summarize(sum_response = sum(count_response))
knitr::kable(PL)
```

#### Responsesum\_response

	64030
Bash	866
C	312
C++	453
Java	512
Javascript	534
MATLAB	361
None	30
Other	390
Python	8357
R	2579
SQL	3496
TypeScript	71

We don't need to see the blank responses



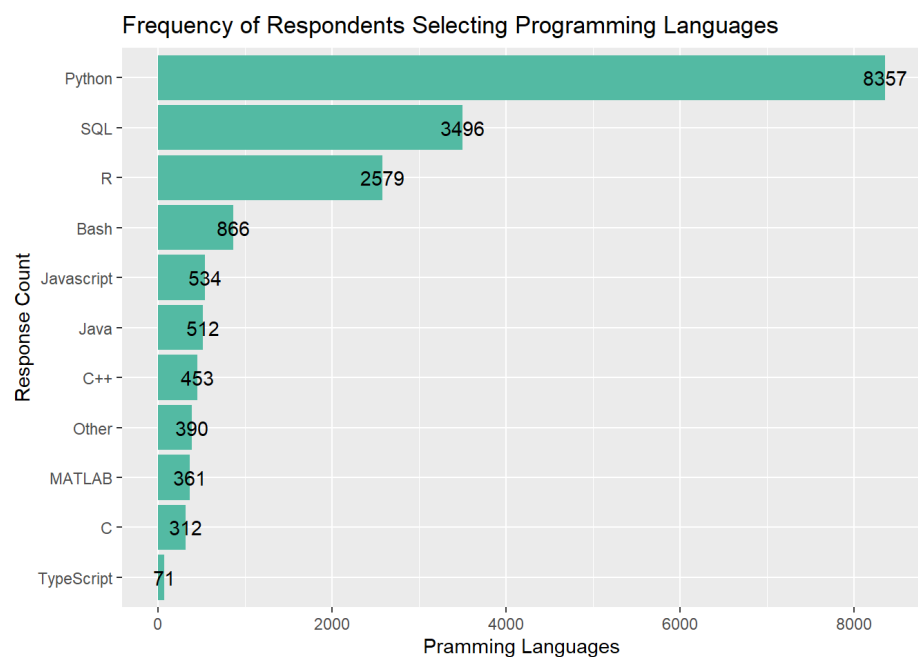
```
PL$Response[PL$Response == ""|PL$Response == "None"] <- NA
PL <- PL %>%
  group_by(Response) %>%
  summarize(sum_response = sum(sum_response))
PL <- na.omit(PL)
knitr::kable(PL)
```

#### Responsesum\_response

Bash	866
C	312
C++	453
Java	512
Javascript	534
MATLAB	361
Other	390
Python	8357
R	2579
SQL	3496
TypeScript	71

```
PL$Response <- factor(PL$Response, levels = PL$Response[order(PL$sum_response)])
```

```
ggplot(data=PL, aes(x=Response,y=sum_response, fill=Response))+
  geom_col(fill="#53baa3") +
  geom_text(label=(PL$sum_response)) +
  ggtitle("Frequency of Respondents Selecting Programming Languages") +
  coord_flip() +
  xlab('Response Count') +
  ylab('Pramming Languages')
```

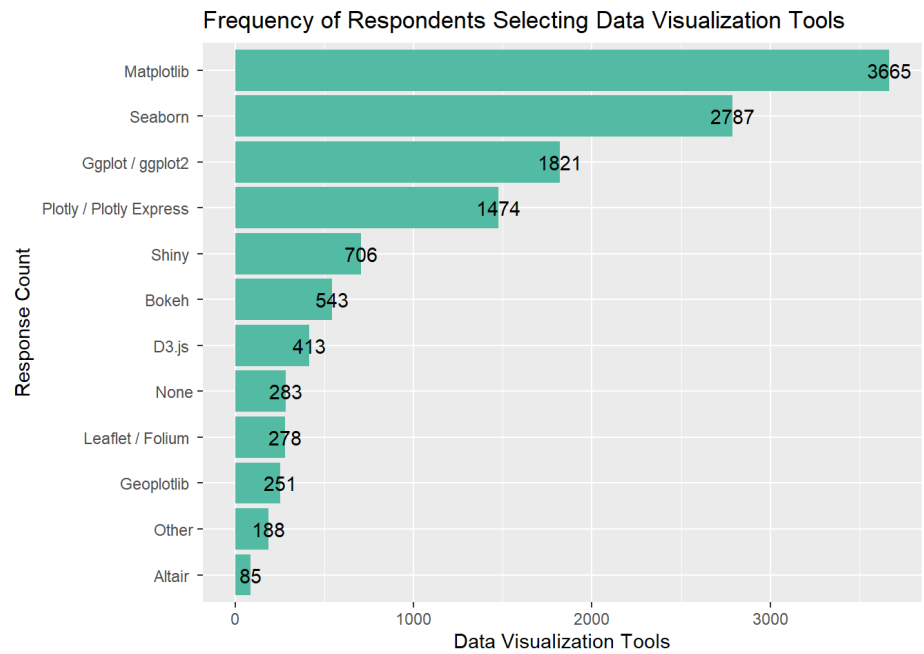


## 2. Data Visualization

```
q_20$response <- factor(q_20$response, levels =
q_20$response[order(q_20$count_response)])
```

```
q_20graph <- q_20[-c(1), ]
```

```
ggplot(data=q_20graph, aes(x=response,y=count_response, fill=response))+
  geom_col(fill="#53baa3")+
  geom_text(label=(q_20graph$count_response))+
  ggtitle("Frequency of Respondents Selecting Data Visualization Tools") +
  coord_flip() +
  xlab('Response Count') +
  ylab('Data Visualization Tools')
```

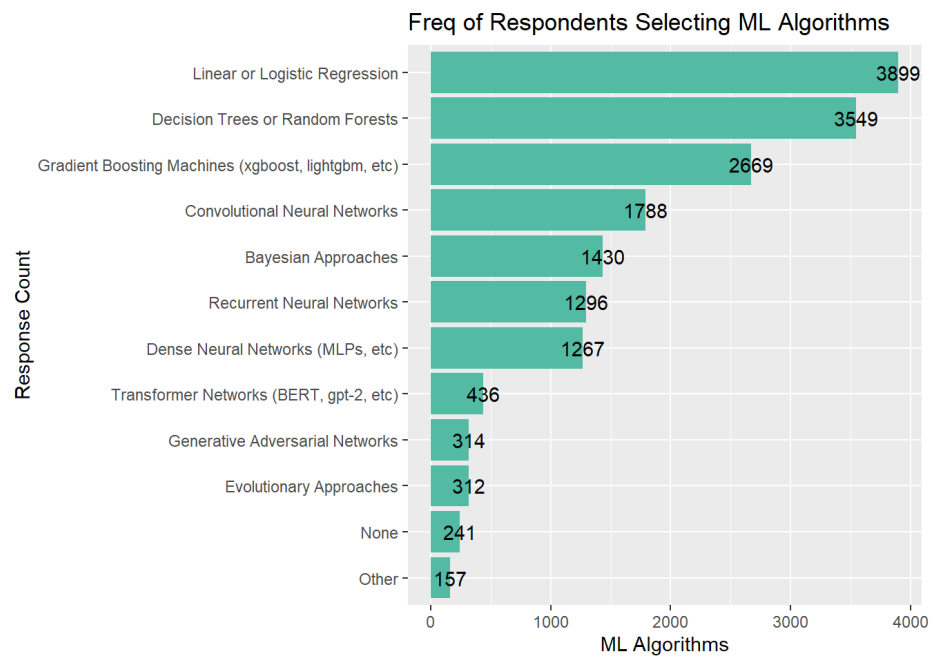


### 3. ML Algorithm

```
q_24$response <- factor(q_24$response, levels =
q_24$response[order(q_24$count_response)])

q_24graph <- q_24[-c(1), ]

ggplot(data=q_24graph, aes(x=response,y=count_response, fill=response)) +
  geom_col(fill="#53baa3") +
  geom_text(label=(q_24graph$count_response)) +
  ggtitle("Freq of Respondents Selecting ML Algorithms") +
  coord_flip() +
  xlab('Response Count') +
  ylab('ML Algorithms')
```

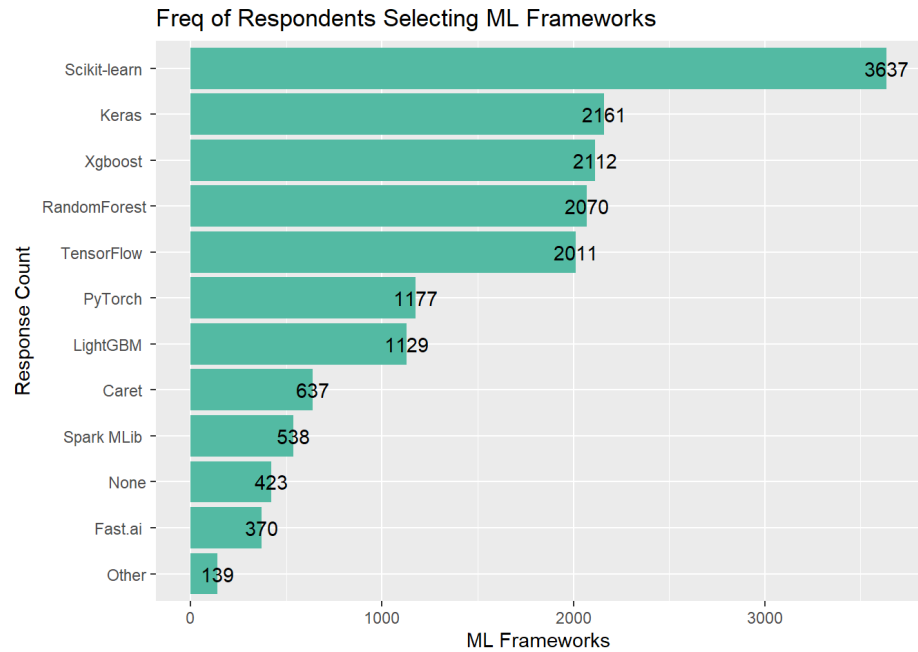


### 4. ML Framework

```
q_28$response <- factor(q_28$response, levels =
q_28$response[order(q_28$count_response)])

q_28graph <- q_28[-c(1), ]

ggplot(data=q_28graph, aes(x=response,y=count_response, fill=response)) +
  geom_col(fill="#53baa3") +
  geom_text(label=(q_28graph$count_response)) +
  ggtitle("Freq of Respondents Selecting ML Frameworks") +
  xlab('Response Count') +
  ylab('ML Frameworks') +
  coord_flip()
```



##### 5. Cloud Platforms

```
CloudPlatform <- rbind(q_29,q_30)
CloudPlatform <-CloudPlatform %>%
  group_by(response) %>%
  summarize(sum_response= sum(count_response))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
knitr::kable(CloudPlatform)
```

response	sum_response
	141085
Alibaba Cloud	47
Amazon Web Services (AWS)	1502
Google Cloud Platform (GCP)	1080
IBM Cloud	192
Microsoft Azure	711
Oracle Cloud	76
Red Hat Cloud	36
Salesforce Cloud	55
SAP Cloud	36
VMware Cloud	64
AWS Batch	149
AWS Elastic Beanstalk	144
AWS Elastic Compute Cloud (EC2)	1005
AWS Lambda	459
Azure Container Service	210
Azure Virtual Machines	442
Google App Engine	262
Google Cloud Functions	329
Google Compute Engine (GCE)	593

response	sum_response
Google Kubernetes Engine	256
None	2425
Other	210

```
CloudPlatform$response[CloudPlatform$response == ""|CloudPlatform$response=="None"] <- NA
```

For cloud platforms, we are only interested in seeing the combined results for the service provider. Therefore, we will combine the responses from Question 30 to only reflect the provider for the tool. For instance, for AWS we are currently seeing 5 naming variations. We would like to reconcile the naming variations under AWS to get the combined results. This applies to Azure and Google Cloud as well.

```
x <- str_detect(CloudPlatform$response,"AWS")
y <- str_detect(CloudPlatform$response,"Azure")
z <- str_detect(CloudPlatform$response,"Google")
```

```
CloudPlatform$response <- ifelse(x,"Amazon Web Services (AWS)",CloudPlatform$response)
CloudPlatform$response <- ifelse(y,"Microsoft Azure",CloudPlatform$response)
CloudPlatform$response <- ifelse(z,"Google Cloud Platform",CloudPlatform$response)
CloudPlatform <- na.omit(CloudPlatform)
CloudPlatform <- CloudPlatform %>%
  group_by(response) %>%
  summarize(sum_response= sum(sum_response))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
knitr::kable(CloudPlatform)
```

response	sum_response
Alibaba Cloud	47
IBM Cloud	192
Oracle Cloud	76
Red Hat Cloud	36
Salesforce Cloud	55
SAP Cloud	36
VMware Cloud	64
Amazon Web Services (AWS)	3259
Google Cloud Platform	2520
Microsoft Azure	1363
Other	210

```
CloudPlatform <- rbind(q_29,q_30)
CloudPlatform <- CloudPlatform %>%
  group_by(response) %>%
  summarize(sum_response= sum(count_response))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
knitr::kable(CloudPlatform)
```

response	sum_response
	141085
Alibaba Cloud	47
Amazon Web Services (AWS)	1502
Google Cloud Platform (GCP)	1080
IBM Cloud	192
Microsoft Azure	711
Oracle Cloud	76
Red Hat Cloud	36
Salesforce Cloud	55
SAP Cloud	36
VMware Cloud	64
AWS Batch	149

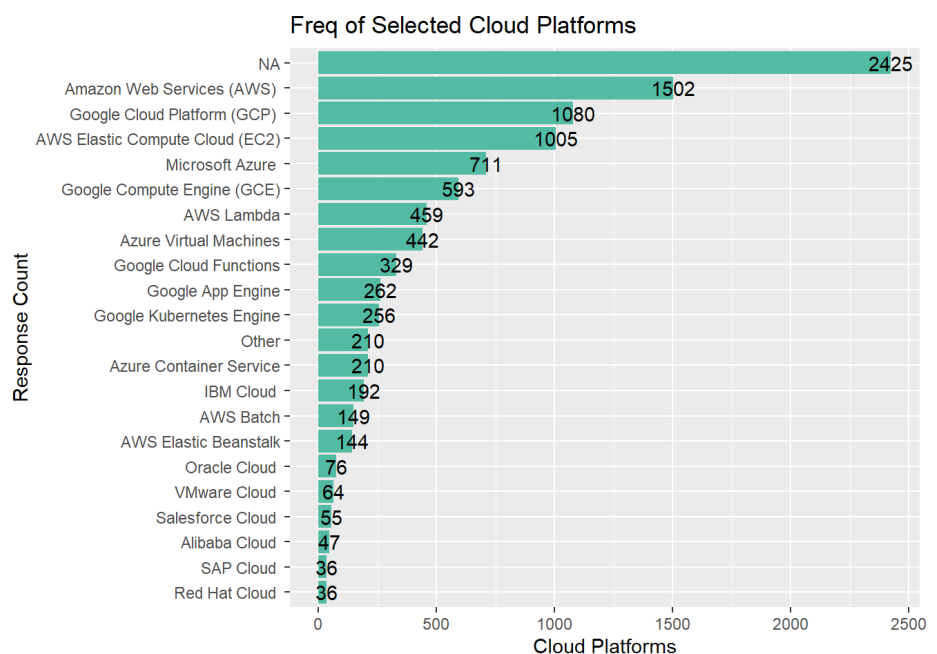
response	sum_response
AWS Elastic Beanstalk	144
AWS Elastic Compute Cloud (EC2)	1005
AWS Lambda	459
Azure Container Service	210
Azure Virtual Machines	442
Google App Engine	262
Google Cloud Functions	329
Google Compute Engine (GCE)	593
Google Kubernetes Engine	256
None	2425
Other	210

```
CloudPlatform$response[CloudPlatform$response == "" | CloudPlatform$response == "None"] <- NA
```

```
CloudPlatform$response <- factor(CloudPlatform$response, levels = CloudPlatform$response[order(CloudPlatform$sum_response)])
```

```
CloudPlatformgraph <- CloudPlatform[-c(1), ]
```

```
ggplot(data=CloudPlatformgraph, aes(x=response,y=sum_response, fill=response)) +
  geom_col(fill="#53baa3") +
  geom_text(label=(CloudPlatformgraph$sum_response)) +
  ggtitle("Freq of Selected Cloud Platforms") +
  xlab('Response Count') +
  ylab('Cloud Platforms') +
  coord_flip()
```



## Dataset 2 (Other Text responses)

Free form responses for those who selected the other option in the main multiple choice survey.

Data

```
Data2 <- dbGetQuery(my_connection, "Select * FROM dbo.other_text_responses")
```

Filtering out non-data roles We are only going to look at respondents primary responsibilities presumably entail working with data on a daily basis, given their titles.

```
DataRoles <- dbGetQuery(my_connection,"Select *
FROM dbo.other_text_responses
WHERE
[Q5_OTHER_TEXT] like '%Data%' OR
[Q5_OTHER_TEXT] like '%Machine Learning%'
OR [Q5_OTHER_TEXT] like '%Artificial Intelligence%' OR [Q5_OTHER_TEXT] like '%Business I
ntelligence%'
OR [Q5_OTHER_TEXT] like '%Analytics%'
OR [Q5_OTHER_TEXT] like '%Algorithm%'
OR [Q5_OTHER_TEXT] like '%Analyst%'
OR [Q5_OTHER_TEXT] like '%Developer%'
OR [Q5_OTHER_TEXT] like '%AI%'
OR [Q5_OTHER_TEXT] like '%Programmer%'
OR [Q5_OTHER_TEXT] like '%Math%")
DataRoles
```

#### Q12\_OTHER\_TEXT

<chr>

Books

1-10 of 223 rows | 1-1 of 28 columns

Previous 1 2 Next

## Primary Tool

The survey question where a significant amount of respondents picked the other option is Question 14 (What is the primary tool that you use at work or school to analyze data? (Include text response) - Selected Choice). Therefore, we would like to see if there are any other skillsets or tools here that are worth looking into besides the ones mentioned in the main dataset.

```
PrimaryTool <- DataRoles %>%
select("Q14_OTHER_TEXT")

PrimaryTool <-PrimaryTool %>%
gather(Q14other,response) %>%
group_by(response) %>%
summarize(count_reponse = n())
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

PrimaryTool

#### response

<chr>

All of the above -- depending on the business problem/dataset/solution

Both Local and Cloud-based

C#

Excel, adobe analytics, Google sheets

Jupyter note

Local developments & Cloud-based data software & APIs

Not sure

Not using at work or school

Octave, python notebooks etc.

1-10 of 18 rows | 1-1 of 2 columns

Previous 1 2 Next

Respondents were allowed to input more than one answer in this field,i.e. one respondent can list multiple tools in their response, usually separated by a comma. We will therefore split the responses into separate columns first and then convert the dataset from wide to long.

```
PrimaryTool <- separate(PrimaryTool,response,
  into=c("T1","T2","T3"),
  sep=",")
```

```
## Warning: Expected 3 pieces. Additional pieces discarded in 2 rows [14, 17].
```

```
## Warning: Expected 3 pieces. Missing pieces filled with `NA` in 15 rows [1, 2, 3,
## 4, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 18].
```

```
PrimaryTool %>% gather(Tool,Response,
  T1,T2,T3)%>% group_by(Response)%>%
  summarize(Sum_response=sum(count_reponse))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

Response
<chr>
adobe analytics
Google sheets
python
Python
python notebooks etc.
Pytorch
R - not a production user of deep learing
All of the above -- depending on the business problem/dataset/solution
Both Local and Cloud-based

1-10 of 27 rows | 1-1 of 2 columns

Previous 1 2 Next

Looks like Python might be the winner here.

## Indeed Scrape

We performed multiple web-scrapes of Indeed.com to collect data on real-world examples of jobs with the title “data scientist.” These data were intended to give further insight into the expectations of the market such as where data scientists’ jobs are located and the salaries. Thus, we looked for those markers of job title, company, salary ranges and location of each posting.

Our initial scrape was compiled to create a sample for analysis. In this, we only wanted to create a data frame with the job title from the first page of results in the Indeed job search. A regular expression in conjunction with functions from the rvest package allowed the information to be extracted from the html of the webpage. The first scraping example is displayed below.

```
url <- 'https://www.indeed.com/jobs?q=data%20scientist&l&vjk=00ba1a22ba67ffd2'
webpage <- read_html(url)
job_data_html <- html_nodes(webpage,'.jobtitle , #sja0 b')
job_data <- html_text(job_data_html)
head(job_data)
```

```
## [1] "\nIntern - Data Scientist - Summer 2021"
## [2] "Data"
## [3] "Scientist"
## [4] "\nSenior Data analyst (REMOTE)"
## [5] "\nData Scientist"
## [6] "\nData Scientist"
```

In each case, the data was extracted as a text file and was rife with errors. It was not usable at this stage. Another parameter was necessary to clean the data and transform it into something functional. In this case, we decided a vector of characters would suffice. The example of the regular expression used to extract the job title of each posting on the first webpage of Indeed for this singular sample is shown.

```
str_extract(job_data, "(\\w+.+)")
```

We found this string to be the most effective at parsing the job title of all listing on the first page into usable form. We repeated this process until we had a clear understanding of each variable of interest to us available on the webpage.

Some information presented challenges that were not transforming data types or parsing strings but rather developing a standard quantifiable value from a range of numbers in different units and categories. A good example of this is the monetary information where the employers listed the information as a range of values per year, per hour, and contractual amounts. None of which would be remotely close when performing calculations.

To solve this, we separated the information we had into categories and selected the category with the most results. Those with a salary listed as a range per year had the highest proportion of jobs with any monetary information on its posting. The process of cleaning can be seen here:

```
# Performing the 1-page scrape and producing a text file from the html of indeed.com
url <- 'https://www.indeed.com/jobs?q=data%20scientist&l&vjk=00ba1a22ba67ffd2'
webpage <- read_html(url)
sal_data_html <- html_nodes(webpage, '.salaryText')
sal_data <- html_text
# Cleaning the subset begins with evaluation of the data
head(sal_data)
# Extracting the salary ranges provided as characters
salary_data <- str_extract(job_data, "(\\w+.+)")
# Removing the hourly rates - a smaller group than salary range
salary_data <- str_remove_all(salary_data, "\\d+ an hour")
# Remove the label "a year" since all should be at this stage
salary_data <- str_remove_all(salary_data, " a year")
# Remove the dollar sign - they are all dollars
salary_data <- str_remove_all(salary_data, "\\$")
# Anticipating duplicates - testing a solution
salary_data <- as.data.frame(salary_data)
salary_data[2,1] <- ("70,000 - $90,000")
salary_data %>%
  mutate(salary = as.factor(job_data)) %>%
  count(salary)
```

As with all samples, there were missing values inherent to the postings on Indeed. Salary information was no exception. These missing values were included in the dataset but were not included in the analysis. To determine how well these chunks of code ran with the full data from the indeed search, we ran it in for-loops to collect pages of job information.

In total, there were at least 18 scrapes performed, two for each variable in the data frame (a sample and a full run). With the html to text data from each sample being pulled from a different section on the webpage, they often required a unique regular expression to produce functional data types. To ensure we could manipulate the data for analysis, the resultant text strings were all converted a data frame as characters. That complete progression of extraction and cleaning was run with the following code. Each variable was stored in its own column and each observation has its own row.

This creates a data frame of the publicly posted jobs with the title “data scientist” on Indeed on October 13, 2020 called *Indeed*. It contains several attributes of the posted jobs, namely, the job title, the company name or employer, location, job summary, and a link to the original posting webpage. Data is then exported as a spreadsheet (.csv) and uploaded to our GitHub



repository to make it more accessible and reproducible.

## Indeed Analysis

The code above results in the CSV stored in our Github repository, which we'll read into a dataframe.

```
indeed_df <- read.csv(url("https://raw.githubusercontent.com/Lnkiim/DATA607_project3/main/Indeed.csv"), stringsAsFactors=FALSE)
indeed_df$minsal <- str_remove(indeed_df$minsal, "[.]")
indeed_df$minsal <- as.numeric(as.character(indeed_df$minsal))
indeed_df$maxsal <- str_remove(indeed_df$maxsal, "[.]")
indeed_df$maxsal <- as.numeric(as.character(indeed_df$maxsal))
glimpse(indeed_df)
```

```
## Rows: 1,412
## Columns: 10
## $ X      <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,...
## $ title  <chr> "Senior Data Scientist", "Data Scientist", "Clinical Data ...
## $ date   <chr> "Today", "Just posted", "30", "8", "30", "Just posted", "3...
## $ company <chr> "Headcheckit.com", "Alto Neuroscience, Inc", "Nucleus", "C...
## $ salary <chr> "112,853 - 204,864", "70,000 - 90,000", "81,078 - 129,709"...
## $ maxsal <dbl> 204864, 90000, 129709, NA, 177616, NA, NA, 204864, 90000, ...
## $ minsal <dbl> 112853, 70000, 81078, NA, 70000, NA, NA, 112853, 70000, 81...
## $ location <chr> NA, "San Francisco Bay Area, CA", "Glen Allen, VA", "Hernd...
## $ summary <chr> "Being part of a team of data scientists.", "This person w...
## $ link    <chr> "https://www.indeed.com/company/Headcheckit.com/jobs/Senio..."
```

Our Indeed scrape provides information on job posting locations, companies, and salaries. First, we can see that by far the greatest number of job posting at the time of this scrape were in Washington, DC and Lexington Park, MD (and NA) with over 100 in each location. Given the current state of global affairs, these "NA" locations could represent a higher number of remote jobs due to the COVID-19 pandemic. It may also be the case that the employers have multiple locations listed as some cases show. This presents room for further analysis.

```
indeed_toploc <- count(indeed_df, location, sort=TRUE)
indeed_toploc <- (top_n(indeed_toploc, 20))
indeed_toploc <- indeed_toploc %>%
  filter(rank(desc(n))<=50)

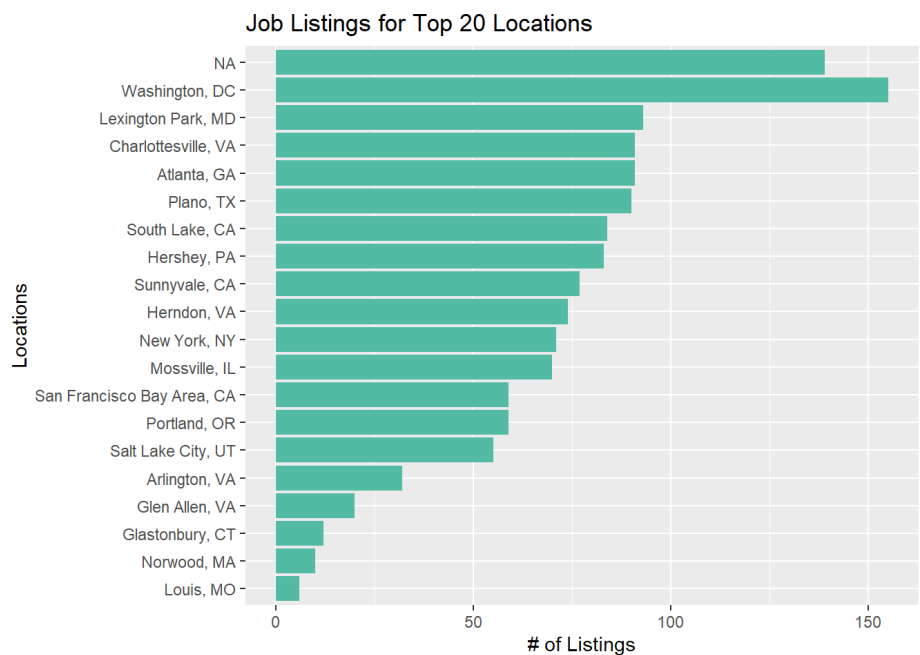
kable(indeed_toploc, format = "markdown")
```

location	n
Washington, DC	155
NA	139
Lexington Park, MD	93
Atlanta, GA	91
Charlottesville, VA	91
Plano, TX	90
South Lake, CA	84
Hershey, PA	83
Sunnyvale, CA	77
Herndon, VA	74
New York, NY	71
Mossville, IL	70
Portland, OR	59
San Francisco Bay Area, CA	59
Salt Lake City, UT	55

location	n
Arlington, VA	32
Glen Allen, VA	20
Glastonbury, CT	12
Norwood, MA	10
Louis, MO	6

```
indeed_toploc$location <- factor(indeed_toploc$location, levels = indeed_toploc$location[order(indeed_toploc$n)])

ggplot(indeed_toploc, aes(x = n, y = location, fill = n)) +
  geom_bar(stat="identity", fill="#53baa3") +
  labs(title="Job Listings for Top 20 Locations") +
  xlab('# of Listings') +
  ylab('Locations')
```



Taking a closer look, we see that for DC the postings primarily come from the CIA and the US Dept of the Treasury. All of the Lexington Park, MD postings are from the KBR. This is just a point-in-time snapshot of the job listings at the time of the scrape (during a strange economic time as well) - but it seems fair to conclude that in general the largest quantity of 'Data Science' jobs are available on the East Coast - one could say that proximity to a urban center is a desirable attribute for a data scientist, as that is where the work seems to be!

```
indeed_dc <- subset(indeed_df, (location == "Washington, DC"))
indeed_topdc <- count(indeed_dc, company, sort=TRUE)
kable(top_n(indeed_topdc, 15), format = "markdown")
```

```
## Selecting by n
```

company	n
Central Intelligence Agency	91
US Department of the Treasury	50
Clarity Campaign Labs	11
Wallethub	2
Perspecta	1

```
indeed_lex <- subset(indeed_df, (location == "Lexington Park, MD"))
indeed_toplex <- count(indeed_lex, company, sort=TRUE)
kable(top_n(indeed_toplex, 15), format = "markdown")
```

```
## Selecting by n
```

company	n
KBR	93

Finally, looking at the salary ranges, where they were available, we see that the mean minimum on the salary range is \$83,578 and the mean maximum is \$139,151. Further, the mean range of the salary is \$55,623 - which is also plotted. While there are certainly many missing values, it is interesting to see the variation in the salary range offered from this Indeed scrape.

```
mean(indeed_df$minsal, na.rm=TRUE)
```

```
## [1] 83578.25
```

```
mean(indeed_df$maxsal, na.rm=TRUE)
```

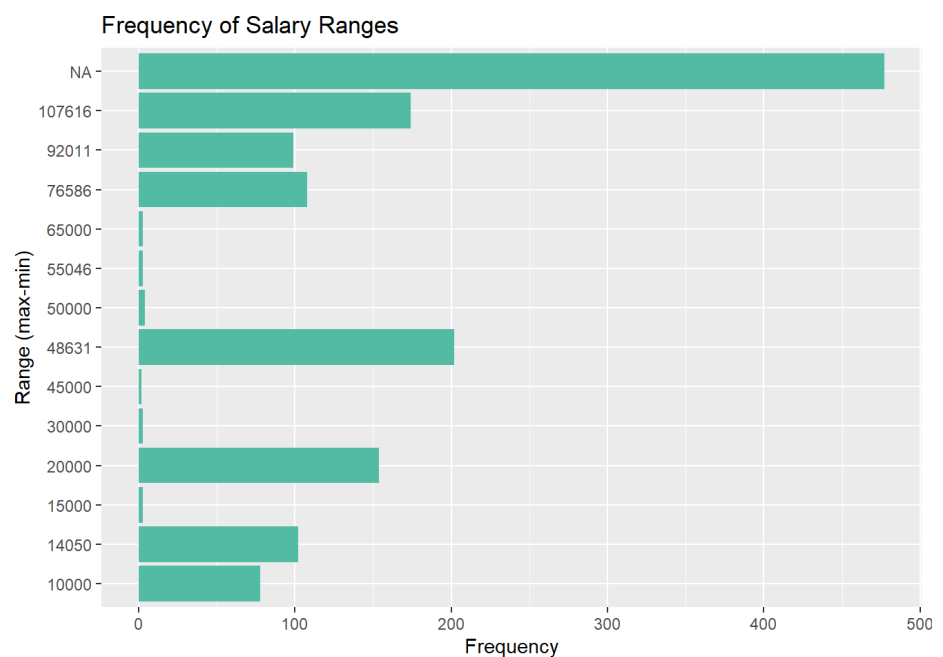
```
## [1] 139151.1
```

```
indeed_df <- indeed_df %>%
  mutate(indeed_df, sal_range = maxsal - minsal)

mean(indeed_df$sal_range, na.rm=TRUE)
```

```
## [1] 55622.74
```

```
ggplot(indeed_df, aes(x = factor(sal_range), fill = sal_range)) +
  geom_bar(fill="#53baa3") +
  coord_flip() +
  labs(title="Frequency of Salary Ranges") +
  xlab("Range (max-min)") +
  ylab("Frequency")
```



## Glassdoor Text Analysis

In this analysis we used several functions to compute the most frequently occurring words in

the data set. The data came from jobs posted on Glassdoor that were scraped into a text file on August 1, 2020. Our goal was to determine if there were any traits or trends that a data scientist would need. To do so, we narrowed our search down to those skills generally thought to be desirable traits of a data scientist based on what we understood as existing norms. This led to an extraction process that gave us the tools, technical skills, and soft skills present in the posting of data science positions for that point in time.

Ideally, the extracted information would be able to tell us what recruiters are looking for and what applicants should have already mastered. Technical skills could be more relevant to express skills related to machine learning and math. Soft skills are relevant to interpersonal behavior.

```
# Reading in data
gd_url="https://raw.githubusercontent.com/Lnkiim/DATA607_project3/main/glassdoor_ds_jobs.csv"
gd_df <- read_csv(url(gd_url))
```

## Word Counts

As a first iteration of the analysis, we looked at the most frequently occurring words across all job descriptions. Notice in the code, that we anti\_join with 'stop words'. 'Stop words' are lexicons that are commonly occurring words that are usually functional in syntax but provide little content (eg: in, the, who, was). This will make our dataset smaller and more compliant for further manipulation.

```
gd_df <-
  gd_df %>%
  mutate(job_id = row_number())

tidy_gd_df <- gd_df %>%
  unnest_tokens(word, Summary)

# remove words that don't mean much
tidy_gd_df <- tidy_gd_df %>%
  anti_join(stop_words, by="word")

# tells you most frequently occurring words across all job descriptions

global_word_count <-
  tidy_gd_df %>%
  count(word, sort = TRUE)

global_word_count %>%
  head(15)
```

### word

<chr>

data

experience

business

team

science

learning

skills

analytics

analysis

machine

1-10 of 15 rows | 1-1 of 2 columns

Previous 1 2 Next

## Wordcloud

The wordcloud below nicely visualizes the most frequently occurring words. The most frequently used words are relevant to describe the data science industry as a whole.

However, there isn't much insight we can glean about the most desirable traits of a data scientist just from a simple count.

# Words must have occurred in the dataset at least 100 times.

```
tidy_gd_df %>%  
  anti_join(stop_words) %>%  
  count(word) %>%  
  with(wordcloud(word, n, max.words = 100))
```

```
## Joining, by = "word"
```



## In-demand Skills

Our methodology was to observe the most frequently occurring words in the job descriptions of each posting. This is appropriate because tools are usually named as single words which indicate hard skills demanded by the potential employer. We start by curating a list of technical skills and see how frequently these words appear in our Glassdoor job description dataset. Instead of scraping a list of technical skills/tools from scratch, we started with some tools that appeared in the Kaggle Survey dataset. After analyzing the results with just the tools that appeared in the Kaggle survey, we found that the data was sparse and hypothesized that the list of technical skills was not inclusive enough to capture the story behind the Glassdoor dataset. Hence, we manually entered more frequently used engineering tools and got better results.

```
# reading raw csv but probably more ideal to bring data from database
url_mult_response="https://raw.githubusercontent.com/Lnkiim/DATA607_project3/main/multiple_
choice_responses_2.csv"
mult_response_df<- read_csv(url(url_mult_response))
```

```
## Parsed with column specification:
## cols(
##   .default = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
# create dataframe just for question 18
q_18 <- mult_response_df %>%
  select(starts_with("Q18"))
# get rid of row 1 bc it has the question
q_18 <- q_18[-1,]

prog_languages <-
q_18 %>%
  gather(key="question", value="response") %>%
  group_by(response) %>%
  summarize(count_reponse = n()) %>%
  arrange(desc(response)) %>%
  head(12)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
prog_languages_list <- tolower(prog_languages$response)

q_16 <- mult_response_df %>%
  select(starts_with("Q16"))

# get rid of row 1 bc it has the question
q_16 <- q_16[-1,]

dev_env <-
q_16 %>%
  gather(key="question", value="response") %>%
  group_by(response) %>%
  summarize(count_reponse = n()) %>%
  arrange(desc(response)) %>%
  head(12)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
dev_env_list <- tolower(dev_env$response)

clean_devEnvList <- c()

# clean list of development environment tools
for (text in dev_env_list) {
  devList <- strsplit(text, "[/(),.]")
  for (item in devList) {
    clean_devEnvList <- c(clean_devEnvList,item)
  }
}

# adding more tools that dataset might not include
my_list <- c("git", "github", "sublime", "sublime text", "docker", "command line", "php", "intellij", "intellij idea", "slack", "gitlab", "item", "item2", "pycharm", "unity", "jetbrains", "linux", "postman", "api", "sas", "apache", "spark", "apache spark", "bigml", "excel", "ggplot", "ggplot2", "tableau", "had oop", "scikit", "scikitlearn", "scikit learn", "tensorflow", "sas", "json", "xml", "")

technical_traits <- c(prog_languages_list,clean_devEnvList,my_list )
technical_traits <- trimws(technical_traits, which = c("both"))

# Global Words that intersect with list of scraped technical traits
technical_skills <- tidy_gd_df %>%
  filter(word %in% technical_traits)

technical_word_count <-
technical_skills %>%
  count(word, sort = TRUE)

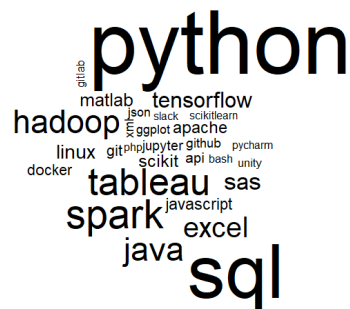
technical_word_count
```

word
<chr>
python
sql
spark
tableau
hadoop
java
excel
sas
tensorflow
linux

1-10 of 30 rows | 1-1 of 2 columns

[Previous](#) **1** [2](#) [Next](#)

We can see from the Wordcloud and the data frame, Python and SQL are the most sought after technical skills by a significant margin. The second and third sought more frequently sought after tools include Spark and Tableau but the large gap between the top two and the next two, suggests that there are technical skills which may not be single word tools. We investigated further in more advanced natural language processing frameworks.



## TF and TF-IDF

```
# grouped so that each company is a "document" and each word is a "token"
```

```
term_freq <- tidy_gd_df %>%  
  anti_join( stop_words, by="word") %>%  
  group_by(`Company Name`) %>%  
  count(word, sort = TRUE)
```

```
term_freq <-  
  term_freq %>%  
  mutate(tot_word_count = sum(n))
```

```
term_freq_by_rank <- term_freq %>%  
  mutate(rank = row_number(),  
         term_freq = n/tot_word_count)
```

```
# bind_tf_idf
```

```
term_freq_by_rank <-  
  term_freq_by_rank %>%  
  bind_tf_idf(word, `Company Name`, n) %>%  
  arrange(desc(tf_idf))
```

```
term_freq_by_rank
```

### Company Name

<chr>

MILLENNIUM CAPITAL MANAGEMENT (SINGAPORE) PTE. LTD.

Samsung Electronics\r3.6

CultureFit Technology Staffing\r5.0

JobsLab

DSW\r3.3

Novacom Technologies

StraitSys\r4.4

Farmer's Business Network, Inc.\r3.6

Idealab\r4.0

Zoba

1-10 of 10,000 rows | 1-1 of 9 columns

Previous 1 2 Next

Unfortunately, the tf-idf doesn't give us much helpful additional insight in discovering which traits make a most desirable data scientist. However, it does show us that the top words are jargon specific to a specialized industry. (eg: arbitrage, banking, scm).

## N-gram Analysis

As mentioned, it is reasonable to search for tools with a global count because tools are often single words. However, soft skills and technical skills may be expressed more frequently with two words, or perhaps even three words. In this section we look at n\_grams, which is an investigation as to how often n number of words occur together.

```
subset_gd_df <- gd_df[,c(3,5)]
```

```
# common pairs of words
```

```
gd_bigrams <- subset_gd_df %>%  
  unnest_tokens(bigram, Summary, token = "ngrams", n = 2)
```

```
bigrams_separated <- gd_bigrams %>%  
  separate(bigram, c("word1", "word2"), sep = " ")
```

```
bigrams_separated <- bigrams_separated %>%  
  filter(!word1 %in% stop_words$word) %>%  
  filter(!word2 %in% stop_words$word)
```

```
bigram_counts <- bigrams_separated %>%  
  count(word1, word2, sort = TRUE)
```



Here we can see that the most frequently occurring couplings of words do illustrate the most sought after technical and soft skills. Let us use the visual below to investigate further.

bigram\_counts

word1

<chr>

machine

data

data

computer

data

communication

data

data

equal

data

1-10 of 10,000 rows | 1-1 of 3 columns

Previous 1 2 Next

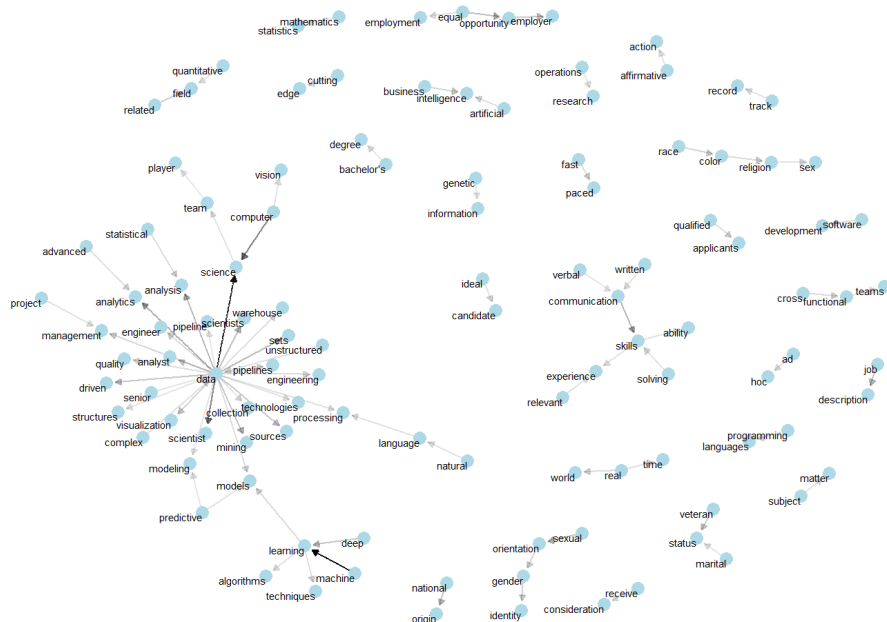
Here we're able to visualize the relationship between couplings of words. The darkness of the arrow expresses how strong the relationship between two words is, and the arrow indicates the direction of the relationship. Using n-grams when n equals 2 lists a lot of technical skills, some of which include: 'machine learning', 'data analytics', 'communication skills', 'data mining', 'deep learning'.

```
set.seed(2022)
```

```
bigram_graph <- bigram_counts %>%  
  filter(n > 120) %>%  
  graph_from_data_frame()
```

```
a <- grid::arrow(type = "closed", length = unit(.05, "inches"))
```

```
ggraph(bigram_graph, layout = "fr") +  
  geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,  
    arrow = a, end_cap = circle(.07, "inches")) +  
  geom_node_point(color = "lightblue", size = 3) +  
  geom_node_text(aes(label = name), vjust = 1, hjust = 1, size=2) +  
  theme_void()
```



Looking at n-grams where n equals three shows us many more soft skills that are sought after. Some of the most sought after soft skills include: written communication skills, fast paced environment, cross functional teams, verbal communication skills, agile software development, and strong analytical skills.

# common 3 words

```
gd_trigrams <- subset_gd_df %>%
  unnest_tokens(trigram, Summary, token = "ngrams", n = 3)

gd_trigrams_separated <- gd_trigrams %>%
  separate(trigram, c("word1", "word2", "word3"), sep = " ")

trigrams_separated <- gd_trigrams_separated %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word) %>%
  filter(!word3 %in% stop_words$word)

trigrams_counts <- trigrams_separated %>%
  count(word1, word2, word3, sort = TRUE)

trigrams_counts
```

word1

<chr>

equal

sexual

race

orientation

machine

natural

machine

data

color

machine

1-10 of 10,000 rows | 1-1 of 4 columns

Previous 1 2 Next

## Conclusion

It is interesting to see that as we write this report in R, the most in-demand language for data science and machine learning appears to be Python from both the Glassdoor and Kaggle analyses. This was closely followed by SQL with both languages maintaining a strong lead. With Python being the most frequently used language, it is a natural progression to see Matplotlib as the most frequently used visualization tool. It comes as no surprise that the other tools for machine learning frameworks, linear regressions, and modeling are also based on working with the Python programming language. This is easily the most notable hard skill and tool demanded from the programming portion of what data scientists do; but it is not the only skill.

Soft skills, elucidated by the tables and clusters of words displayed in the Glassdoor analysis, gives another look into what these positions require. We found that having experience, especially in data, was another high priority. Employers are interested in individuals who can work well in teams and above all, possess some business knowledge. These word clouds added to more hard skills for an aspiring data scientist to learn Spark and Tableau.

There is no clear choice in the data science community when it comes to selecting cloud platforms. The top 3 in use, according to the Kaggle survey analysis were: Amazon's AWS, Google's Cloud, and Microsoft Azure. Although it is important to note that many people did not respond to this at all, which could mean those respondents are using multiple or no cloud platforms. Regardless, selecting one of those main three would be a good place to start for aspiring data scientists.

From Indeed, we can also see that the salary information available at the time of the scrape is inconsistent with prior literature. Although most employers did not list salary information, there appears to be a slightly lower salary for data scientists. However, this may be due to several factors including the numerous government jobs listed at that particular time and the conditions of the economy. Scraping the site with more frequency over time might give us a better sample. It may also be possible that salaries show wider discrepancies when factored by country. For example, salaries in India may be much lower than those in the United States.

Importantly, we are living in an unprecedented time. A pandemic has rearranged and continues rearrange company dynamics as we adapt to a new lifestyle across the world. We have no doubt that these changes have influenced the data.

For us, this project was meant to teach us how to collaborate on a project of this size, work with messy data, and learn a bit more about the field we all are entering. It felt a bit like peeling back the layers of an onion, seeing each other's code and findings made us pose more questions and think of new directions to pursue. So much data, so little time. We hope the analysis above gives others some preliminary insight into the demands and expectations of the data science field, and prompts readers to peel back another layer of the onion.

## Sources

Gan, James. "Various Data Science Titled Jobs in Singapore 2020." Kaggle, 16 Aug. 2020, [www.kaggle.com/jamesgsw/various-data-science-titled-jobs-in-singapore-2020/metadata](https://www.kaggle.com/jamesgsw/various-data-science-titled-jobs-in-singapore-2020/metadata).

Human Resources, University of Pennsylvania. "What's Driving the Demand for Data Scientists?" [Knowledge@Wharton](https://knowledge.wharton.upenn.edu/article/whats-driving-demand-data-scientist), 8 Mar. 2019, [knowledge.wharton.upenn.edu/article/whats-driving-demand-data-scientist](https://knowledge.wharton.upenn.edu/article/whats-driving-demand-data-scientist)

Milan. "GlassDoor(Data Scientist)." Kaggle, 1 Aug. 2020, [www.kaggle.com/milan400/glassdoordata-scientist](https://www.kaggle.com/milan400/glassdoordata-scientist).

Palmore, Zach. "Indeed.csv" Data Scientist Jobs, Employment | Indeed.com, 13 Oct. 2020, [www.indeed.com/jobs?q=data+scientist%5C](https://www.indeed.com/jobs?q=data+scientist%5C).

## Contributors

Douglas Barley: database storage, Kaggle Survey tidying, presenter

Atina Karim: database storage, Kaggle Survey analysis, lead presenter

Rachel Greenlee: project manager, Indeed analysis, graphing support, writing support

Ellen Kim: github master, Glassdoor tidying and analysis, presenter

Zachary Palmore: Indeed scrape, Indeed tidying, lead project writer

# Shareholder Activism for the Climate Crisis

[Code ▾](#)

Rachel Greenlee

11/21/2020

## Introduction

This project is an exploration of how the public and/or shareholders can pressure companies to not turn a blind eye to the affects their operations have on the environment and the climate crisis. Many governments have been slow to impose sweeping regulations on business and industry with regards to harmful emissions, so looking to how the companies themselves can be influenced may be a useful route.

I'm choosing to focus on shareholder activism, what it's done so far and where else it could be applied. Shareholder activism is when a person or entity attempts to use their rights as a shareholder of a publicly-traded corporation to bring about change within or for the corporation. This can happen by a shareholder putting forth a resolution, and if it meets certain requirements, it will be brought to the full group of shareholders to vote on. There are nonprofits who organize shareholders with the explicit purpose of shareholder activism with regards to the climate crisis, such as [The Ceres Investor Network](#), which includes, "over 175 institutional investors, managing more than \$29 trillion in assets, advancing leading investment practices, corporate engagement strategies, and key policy and regulatory solutions...Through powerful networks and advocacy, Ceres tackles the world's biggest sustainability challenges, including climate change, water scarcity and pollution, and inequitable workplaces."

For this project, I imagine I'm working with an organization like The Ceres Investor Network, one that is possible trying to influence corporations even beyond the United States. Maybe they are considering trying to reach out to existing shareholders of these high-emissions companies in order to see if they'll join these shareholder activism movement for the climate crises. What data would an organization with these goals be interested in? What companies (and shareholders) should be targeted? What human beings are driving companies in these harmful directions?

## Background and Outline

My motivation in exploring this topic stems from wanting to find a tangible way to hold those responsible for climate change accountable, but more importantly to get everyone cooperating to act now so that in my last decades on this Earth are not full of (more) humanitarian crises and loss of life like we've never seen before. While individual choices certainly matter (recycling, travel choices, food choices), certain corporations have a grossly disproportionate impact on our environment - making them an efficient use of a climate activist's resources to target with the hopes to create change.

This project explores three datasets.

### CDP Carbon Majors Report, 2017

This 2017 report was [all over the news](#) when it came out with the headline that 71% of the world's greenhouse gas emissions from 1988-2015 (27 years) was from just 100 companies. A single coal company in China was responsible for 14.32% of global greenhouse gas emissions. For this project, I scrape a table from Wikipedia that has the top 20 companies from this list.

*CDP is a not-for-profit charity that runs the global disclosure system for investors, companies, cities, states and regions to manage their environmental impacts. Over the past 20 years we have created a system that has resulted in unparalleled engagement on environmental issues worldwide.*

### Proxy Monitor's Shareholder Resolutions Database

Proxy Monitor has a database of all shareholder resolutions that reached a vote for the largest 250 public companies in the United States (using Forbes rankings). For this project, I filtered to look at the 2 public companies in the United States that made the top 20 companies list from the CDP Carbon Majors Report to see if shareholder activism is at work within these companies.

*ProxyMonitor.org, a website managed by the Manhattan Institute's legal policy team, sheds light on the influence of outside shareholder proposals on publicly traded corporations.*

### The World Wide Web Foundation - Company Data Openness by Country

In the wake of [the Panama Paper's revelations](#) in 2016 many governments, under pressure, promised to make publicly available the data they have on companies registered within their country. In the vast majority of cases governments require companies to provide them information about their accounts, shareholders, owners, directors, beneficiaries, etc. Fighting corruption, particularly between hidden government-corporate relationships, begins by having available data about who benefits from a company's profits.

The World Wide Web Foundation did an analysis across 92 countries to see if governments have been living up to those 2016 promises of making company data available. Taking it a step further, the analysis looks at measures of openness such as if the data is machine readable, available as a bulk download, if a fee is required, etc. Only one country, Australia, met all of their data openness measures.

I will join this by-country data with the top 20 emitters country data to see if the countries where some of this disproportionately large greenhouse gas emitting companies operate, are countries where that company's shareholder and beneficiary data is openly available to the public.

*The World Wide Web Foundation was established in 2009 by web inventor Sir Tim Berners-Lee and Rosemary Leith to advance the open web as a public good and a basic right. We are an independent, international organization fighting for digital equality — a world where everyone can access the web and*

use it to improve their lives.

Code

## Top 20 Emitters

First I will scrape the table from Wikipedia that has the Top 20 Emitters from the 2017 CDP Carbon Majors Report so we have a dataframe of each company, it’s country, and it’s percent share of emissions in the 1988-2015 time frame.

Code

### Exploratory Analysis

Top 20 Table      % Share by Country

Here is the table reproduced as a dataframe. Note the huge percent share China has, at the top of the list with 14.32% of the world’s greenhouse gas emissions from 1988-2015. Also, a few companies appear to span across 2 countries.

Code

Rank	Company	Country	Percentage
1	China (Coal)	China	14.32
2	Saudi Arabian Oil Company (Aramco)	Saudi Arabia	4.50
3	Gazprom OAO	Russia	3.91
4	National Iranian Oil Co	Iran	2.28
5	ExxonMobil Corp	United States	1.98
6	Coal India	India	1.87
7	Petroleos Mexicanos (Pemex)	Mexico	1.87
8	Russia (Coal)	Russia	1.86
9	Royal Dutch Shell PLC	Netherlands, United Kingdom	1.67
10	China National Petroleum Corp (CNPC)	China	1.56
11	BP	United Kingdom	1.53
12	Chevron Corp	United States	1.31
13	Petroleos de Venezuela SA (PDVSA)	Venezuela	1.23
14	Abu Dhabi National Oil Co	United Arab Emirates	1.20
15	Poland (Coal)	Poland	1.16
16	Peabody Energy Corp	United States	1.15
17	Sonatrach SPA	Algeria	1.00
18	Kuwait Petroleum Corp	Kuwait	1.00
19	Total SA	France	0.95
20	BHP Billiton Ltd	Australia, United Kingdom	0.91

## Shareholder Resolutions

The Proxy Monitor website has an interactive feature that allows you to filter shareholder proposals by a variety of filters. I set the filters to all years [2006-2020]] a proposal type of [Environmental], and then the US companies on the top 20 list of [Chevron] and [Exxon Mobil]. The third US company on the list, Peabody Energy, is a private corporation and thus not listed on this website.

Code

### Exploratory Analysis

Cleaned Dataset      Environmental Proposals Over Trime      Sharedholder Support

This table shows a sample of the proposals from shareholders for Chevron and Exxon Mobil with the Environmental tag from 2006-2020. On the website you can click on the table to read the official proposal (and often the company’s response). Here we can see who proposed the resolution (“Proponent”) and the percent of votes the resoulution eventually recived from shareholders.

Code

Company	Year	Title	Proponent	Proponent_Type	Vote
---------	------	-------	-----------	----------------	------

Company	Year	Title	Proponent	Proponent_Type	Vote
Chevron Corp CVX	2006	Report on Oil & Gas Drilling in Protected Areas	NA	Undisclosed	8.66
Chevron Corp CVX	2006	Report on Ecuador	Susan Goldman	Individual	8.37
Chevron Corp CVX	2007	GREENHOUSE GAS EMISSIONS	NA	Undisclosed	8.47
Chevron Corp CVX	2007	HOST COUNTRY ENVIRONMENTAL LAWS	NA	Undisclosed	8.57
Chevron Corp CVX	2008	Report on Canadian Oil Sands	NA	Undisclosed	28.61
Chevron Corp CVX	2008	Greenhouse Gas Emissions Goals	Dominican Sisters of Sparkill, NY	Religious Institutions	10.42
Chevron Corp CVX	2010	Appoint Indep. Dir. with Environmental Expertise	NA	Undisclosed	26.80
Chevron Corp CVX	2010	Report on Financial Risks from Climate Change	NA	Undisclosed	8.60
Chevron Corp CVX	2011	Nominate an Independent Environmental Expert as Director	NA	Undisclosed	24.80
Chevron Corp CVX	2011	Use Sustainability as a Performance Measure for Exec. Comp.	NA	Undisclosed	5.60

## Openness of Company Data

The World Wide Web Foundation’s dataset was available on a Google Sheet and I used the ‘gsheet’ package to read it into a dataframe directly from the public share link. I select just a few of their openness measures to look at for this project. This dataset measures whether governments are making available the company data they collect to the public, with the purpose of being more transparent so corrupt arrangements like the Panama Papers incident are less likely.

Code

### Exploratory Analysis

Cleaned Dataset	Is Company Data Open in These Countries?	Open Data & High Emissions
-----------------	--	----------------------------

Here is the first few rows of the cleaned dataset. Australia was the only country of the 92 they reviewed that met all of their openness criteria. This is only a subset of the criteria, the variable I kept here show if the data was publicly available, machine readable, available as a bulk download, and if there was a fee required.

Code

Country	CalculatedScore	fullyopen	gov_has	public_avail	machine_readable	bulk_download	n
Australia	95	1	1	1	1	1	1
United Kingdom	85	0	1	1	1	1	1
Norway	75	0	1	1	1	1	1
Belgium	65	0	1	1	1	1	0
Switzerland	65	0	1	1	1	0	1
Israel	65	0	1	1	1	1	1
Mexico	65	0	1	1	1	0	1
Macedonia	65	0	1	1	1	1	0
Benin	60	0	1	1	1	1	1
Finland	60	0	1	1	1	0	0

## Conclusion & Recommendations

In this exploratory data project I was able to:

- 1) identify what countries house some of the biggest greenhouse gas emitters in the world,
- 2) for the US see the efforts of shareholder activism in two companies generally receive less than 50% of the shareholder vote, and
- 3) see that most governments, including those which allow huge greenhouse gas emitters to operate in their borders, are not providing open data about those companies.

One obvious early conclusion is that a shareholder activist organization might focus on China regarding more openness with company data so that it can be scrutinized for any corrupt activities between policy-makers and the high-emitting countries on this top 20 list. Further, if shareholders and beneficiaries of these companies in China were public, a shareholder activist organization could engage with these people with the hopes of getting China's high-emitting companies to reduce their global share of emissions.

I would recommend further work bring in content experts on government open data policies, emissions, and shareholder activism. Further analysis at the effectiveness of shareholder activism would be telling and inform those working in this realm of how to use their resources for the most change. Finally, if open data becomes available in some of these countries that are housing high-emitting companies and not putting environmental policies in place - an analysis using Neo4j, [as was done with the Panama Papers](#), could be very telling of any potentially corrupt relationships between government policy makers and company profits.