STAT694 Project

Chun Yin Kong

12/10/2020

A Research on the relationship between the seriousness of Natural Disaster and the Sentiment of Twitter Text - A COVID-19 Pandamic Case Study

Introduction

Research Purpose

In this research project, I am trying to analyze the effect of natural disaster to the sentiment level of the text. We know that in 2020, COVID-19 has affected the whole world and we experienced different levels of community, cities, or even country lockdown, limiting our social activities, teaching and working life, sickness and deaths. As we spent more time on internet, Twitter, one of the very popular social media platform in the US, has tons of texts and information update daily. I would like to through learning the tweets and the geo-location of tweets, to study the effect of natural disaster to our feeling.

Methodology

- Obtain real tweets from Twitter daily, from Oct 5 to Oct 17, 2020, a total of 13 days. In the process of extracting tweets, I used Twitter API in Python, which the relevant codes can be found in the last page of the research report as well.
- The downloaded data set will be in csv format which are able to load into R for Data Wrangling. Primarily removing unnecessary characters, like emojis, html tags. After that, I also removed stopwords for more accurate analysis.
- The cleaned data will then put into analysis and visualization.
- In this research, I chose five major US cities, including:
  - San Francisco, CA
  - Los Angeles, CA
  - Chicago, IL
  - New York, NY
  - Miami, FL.

These five cities has a significantly high confirmed COVID-19 rate per 100K population. I believe the studies of these cities will be interesting.

Data Extraction using Twitter API on Python:

In the data extraction stage, I used Python as my programming language because I am more familiar with Python in API. First I connected my codes to my free Twitter developer account using my own credentials and tokens. Hence I set some filtering parameters, (filer locations to the five cities) so that the exported csv data set is the result that I would like to see. For privacy and security issue, I hidden my credentials and tokens manually.

The use of lexicon

In this project I used "afinn" lexicon because it the sentiment score ranges from -5 to 5. The larger the range, it is easier to see the difference when the data set is large. The sample of the "afinn" lexicon is shown below.

```r
library(tidytext)
library(dplyr)
library(kableExtra)
table_0 <- get_sentiments("afinn") %>%
  sample_n(10)
```

Table 1: Sample Sentiment Score from afinn Lexicon

Word	Score
delighting	3
nigger	-5
discard	-1
enslaved	-2
bitch	-5
comfort	2
fascinate	3
fire	-2
apologises	-1
bad	-3

Method to calculate sentiment score

I will use two library, *tidytext* and *syuzhet* to get individual word's sentiment score. Here I use one of the quotes from Martin Luther King to explain the method that I am using.

```r
library(tidytext)
library(syuzhet)
sample <- get_tokens("Darkness cannot drive out darkness;
                     only light can do that. Hate cannot drive out hate;
                     only love can do that.")
sample
```

```
##  [1] "darkness" "cannot"   "drive"    "out"      "darkness" "only"
##  [7] "light"    "can"      "do"       "that"     "hate"     "cannot"
## [13] "drive"    "out"      "hate"     "only"     "love"     "can"
## [19] "do"       "that"
```

And the Score will be taking the average of all words, and becoming:

```r
sample_score1 <- get_sentiment(sample, method="afinn")
sample_score1
```

```
##  [1] -1  0  0  0 -1  0  0  0  0  0 -3  0  0  0 -3  0  3  0  0  0
```

```r
mean(sample_score1)
```

```
## [1] -0.25
```

Summary Report

Twitter Data Set

```r
library(dplyr)
library(kableExtra)
data <- read.csv("data/tweet_cleaned_withstopwords.csv") # A Data Set with Stop Words
tweet_dataset <- read.csv("data/tweet_cleaned.csv") # Data Set without Stop Words
tweet_dataset <- tweet_dataset %>%
  mutate(tweetcreatedts = as.Date(tweetcreatedts)) %>%
  select(location, text, tweetcreatedts)
```

```r
nrow(tweet_dataset)
```

```
## [1] 22867
```

In this data set, I successfully obtained and filtered 22867 tweets, ranged from Oct 5 to Oct 17 2020.

Table 2: Oct 5 Tweet Data Structure

Location	Tweet Text	Tweet Created Time
Los Angeles	trump has been the biggest source of covid misinformation study finds	[]
Miami	i just got off the phone with coach bowden told me hes feeling okay he tested positive for covid while at the hospital last week he doesnt know how he contracted it told me hes getting tested again today	[]
New York	no reports of receiving either hydroxychloroquine or chloroquine since being diagnosed with covid	[]
San Francisco	president trump s treatment regimen for covid suggests he s either being overtreated or is sicker than is being let on based on interviews with physicians over the past days	[]
Chicago	my wife doesn t believe kayleigh mcenany or any of the trump staff have covid she thinks they are just saying it to make it look like trump really must have it when he really doesn t i m inclined to agree especially if he leaves walter reed today	[]
Chicago	if senior government officials do not announce their diagnoses of covid immediately they put other senior officials at risk that s why im calling on congress to establish clear standards for the public disclosure of senior officials becoming infected with covid	[]

Table 1 shows the structure of the data set. It only includes location of the tweets, the main text, and the time of the tweet was created. Since in this research project, I only interested in seeing the relationship between the sentiment and the location, hence I removed hyperlinks, mentions, hashtags etc.

Table 3: Top 10 Most Frequent Word Appeared

Word	Count
covid	18886
trump	6023
president	2191
tested	2145
positive	1888
pandemic	1845
amp	1837
americans	1458
health	1422
white	1267

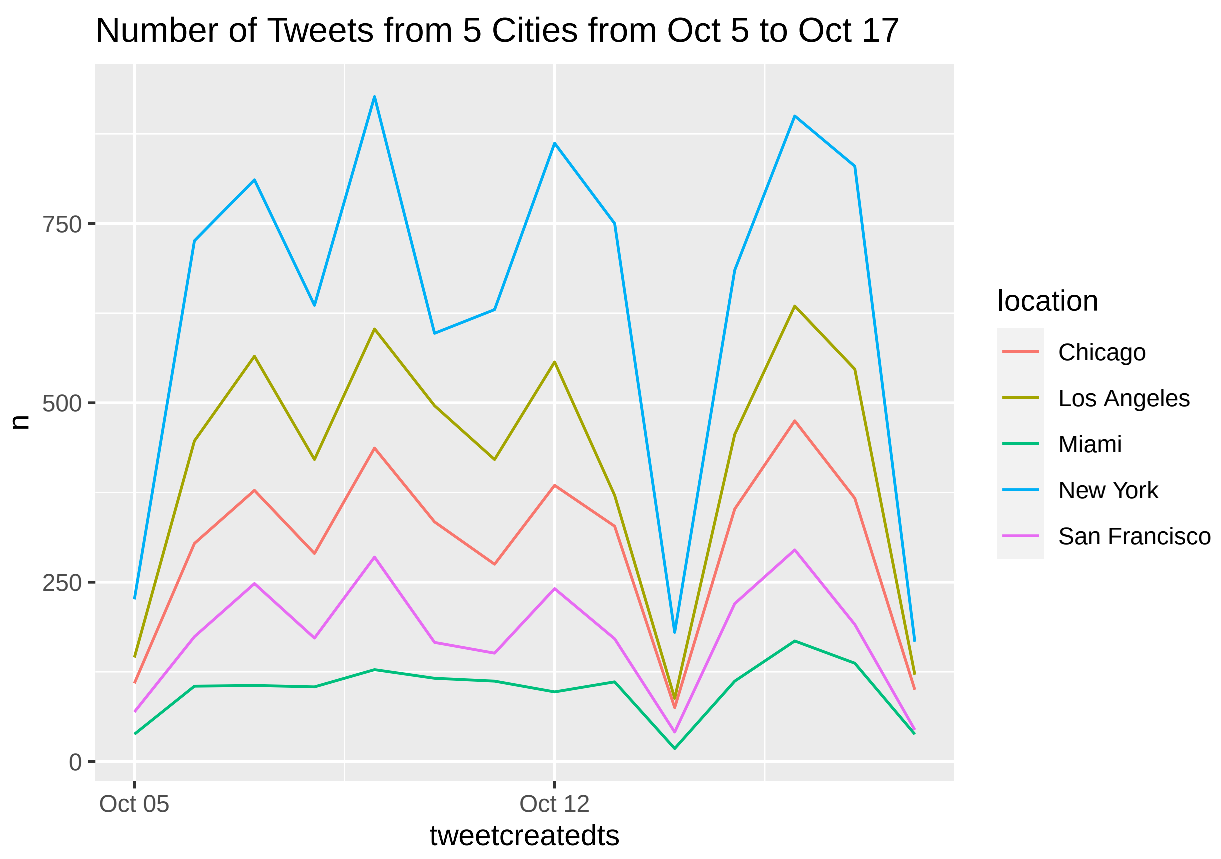
Table 2 Shows the top 10 most frequent word appeared in the Twitter Dataset, after removed stop words. As we can see COVID is the most frequent word appeared in tweets. It is normal because when I choose the search words to be included in Python scripting, I included *"COVID-19"* as one of the keyword to search. Interestingly, we found that two words that are appeared on this table unexpectedly, *"trump"* and *president*. They came as second and third in the table respectively. It seems that a great portion of tweets are related to President Trump, the US president in year 2019 and 2020.

Table 4: Summary Statistics of Tweets with stopwords

Location	Tweet Count	Tweet Ratio	Tweet Mean Character	Tweet Character Usage	Population	Population Ratio
Chicago	4209	0.1840644	158	0.5642857	2830144	0.1771438
Los Angeles	5873	0.2568330	158	0.5642857	3911500	0.2448278
Miami	1390	0.0607863	146	0.5214286	386740	0.0242068
New York	8927	0.3903879	155	0.5535714	8124427	0.5085225
San Francisco	2468	0.1079285	162	0.5785714	723724	0.0452992

Table 3 is the summary statistics of the tweets grouped by locations. The number of tweets, the average length of tweets, and the population of each cities. It is important to know the ratio because the population of the five cities differs a lot and direct comparing the number is meaningless.

```r
library(ggplot2)
tweet_counts <- tweet_dataset %>%
  group_by(location, tweetcreatedts) %>%
  summarise(n=n())
ggplot(tweet_counts, aes(x=tweetcreatedts, y=n, col=location)) +
  geom_line() +
  labs(title="Number of Tweets from 5 Cities from Oct 5 to Oct 17")
```



If we look into the break down of tweets by cities and date, we can see the patterns of total count of tweets over the 13 days periods is similar, giving that it doesn't have much differences.

COVID-19 Figures

Data Collection Method

- For Miami, I will use the Miami-Dade County, Florida data.
- For Los Angeles, I will use the Los Angeles County, California data.
- For San Francisco, I will use the San Francisco County, California data.
- For Chicago, I will use Cook County, Illinois data.
- For New York City, I will use the combined 5 counties data
  - Bronx County with population approximately 1.4 million
  - Kings County with population approximately 2.5 million
  - New York County with population approximately 1.6 millon
  - Queens County with population approximately 2.2 million
  - Richmond County with population approximately 500 thousands.
  - The total population is around 8.2 million which is similar to the result pulled from the package maps and the result above. Hence, to obtain New York City's COVID-19 data, I will consider the sum of these 5 counties.

Since as I search I can't find a complete dataset for all the counties, I will use separate state's open data source to obtain the COVID-19 Cases and related figures.

- For State of California, I used the data set on https://data.ca.gov and the URL is :
  - https://data.ca.gov/dataset/covid-19-cases/resource/926fd08f-cc91-4828-af38-bd45de97f8c3
- For New York City, I used the data set published by NYC Health department on GitHub, and the URL is: https://github.com/nychealth/coronavirus-data. There is a downside as in the readme file on GitHub stated, *Note that sum of counts in this file may not match values in Citywide tables because of records with missing geographic information. This file does not contain information on probable deaths.*
- For Miami and Chicago, since I can't find an official source with downloadable data, I then use COVID-19 Data Hub as my download source. *The COVID-19 Data Hub provides a function in R "covid" in the package "COVID19" which provides easy and simple way to extract data with specific date range*

Below are the codes to obtain a nice table of COVID-19 figures in the format of total confirmed cases per 100K population.

```
california_covid <- read.csv("https://data.ca.gov/dataset/590188d5-8545-4c93-a9a0-e230f0db7290/resource/
                             encoding="UTF-8")
```

```
SF_covid <- california_covid %>%
  filter(county=="San Francisco") %>%
  mutate(date = as.Date(date)) %>%
  filter(date >= as.Date("2020-10-05")) %>%
  filter(date <= as.Date("2020-10-17")) %>%
  mutate(SF_case_per_100k = totalcountconfirmed/723724*100000) %>%
  select(date, SF_case_per_100k)
```

```r
LA_covid <- california_covid %>%
  filter(county=="Los Angeles") %>%
  mutate(date = as.Date(date)) %>%
  filter(date >= as.Date("2020-10-05"))%>%
  filter(date <= as.Date("2020-10-17"))%>%
  mutate(LA_case_per_100k = totalcountconfirmed/3911500*100000)%>%
  select(date, LA_case_per_100k)
```

```r
NYC_covid_raw <- read.csv("https://raw.github.com/nychealth/coronavirus-data/master/trends/data-by-day.
```

Since the structure of the GitHub version of data set is not the the desired data structure, we have to perform some data cleaning.

```r
NYC_covid <- NYC_covid_raw %>%
  mutate(date = as.Date(date_of_interest, format="%m/%d/%Y")) %>%
  mutate(Brooklyn = cumsum(BK_CASE_COUNT),
         Bronx = cumsum(BX_CASE_COUNT),
         Manhattan = cumsum(MN_CASE_COUNT),
         Queens = cumsum(QN_CASE_COUNT),
         Staten_Island = cumsum(SI_CASE_COUNT)) %>%
  filter(date >= as.Date("2020-10-05")) %>%
  filter(date <= as.Date("2020-10-17")) %>%
  select(date, Brooklyn, Bronx, Manhattan, Queens, Staten_Island) %>%
  rowwise() %>%
  mutate(total = sum(c_across(Brooklyn:Staten_Island))) %>%
  mutate(NYC_case_per_100k = total/8124427*100000) %>%
  select(date, NYC_case_per_100k)
```

```r
#For Miami and Chicago Data:
library(COVID19)
Chicago_covid <- covid19(country="USA", level=3, start="2020-10-05", end="2020-10-17", verbose = FALSE)
  filter(administrative_area_level_2 == c("Illinois")) %>%
  filter(administrative_area_level_3 == c("Cook")) %>%
  select(date, confirmed, deaths, population, administrative_area_level_2,
         administrative_area_level_3, latitude, longitude) %>%
  mutate(Chicago_case_per_100k = confirmed/population*100000)

Chicago_covid <- Chicago_covid %>%
  select(date, Chicago_case_per_100k)

Miami_covid <- covid19(country="USA", level=3, start="2020-10-05", end="2020-10-17", verbose = FALSE) %
  filter(administrative_area_level_2 == c("Florida")) %>%
  filter(administrative_area_level_3 == c("Miami-Dade")) %>%
  select(date, confirmed, deaths, population, administrative_area_level_2,
         administrative_area_level_3, latitude, longitude) %>%
  mutate(Miami_case_per_100k = confirmed/population*100000) %>%
  select(date, Miami_case_per_100k)
```

Below is the summary statistics of the 5 major cities's COVID-19 Cases from Oct 5 to Oct 17 2020.

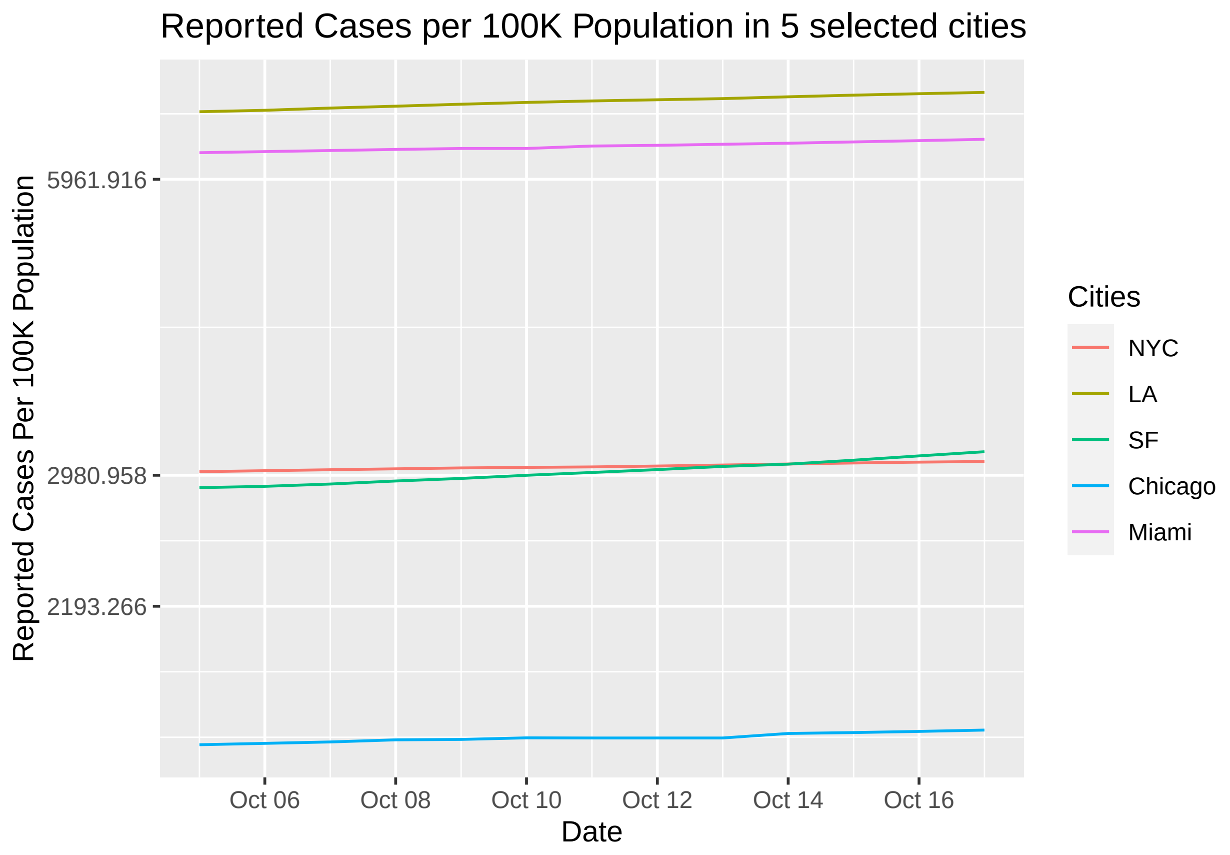
```r
covid_plot <- NYC_covid %>%
  inner_join(SF_covid, by=c("date"="date")) %>%
  inner_join(LA_covid, by=c("date"="date")) %>%
  inner_join(Chicago_covid, by=c("date"="date")) %>%
  inner_join(Miami_covid, by=c("date"="date")) %>%
  select(-c("id.y", "id.x"))
```

Table 5: Summary Statistics of COVID-19 Reported Cases by Cities per 100K Population

Date	New York City	San Francisco	Los Angeles	Chicago	Miami
2020-10-05	3005.960	1585.687	6984.533	2895.539	6345.263
2020-10-06	3012.914	1590.800	7007.925	2904.490	6361.863
2020-10-07	3019.573	1596.050	7044.842	2920.256	6377.726
2020-10-08	3025.998	1603.926	7075.623	2941.110	6393.958
2020-10-09	3032.485	1605.584	7108.245	2958.837	6408.349
2020-10-10	3036.300	1611.526	7138.259	2980.720	6408.349
2020-10-11	3039.734	1610.973	7163.237	3000.233	6445.376
2020-10-12	3046.073	1610.973	7182.948	3020.582	6455.645
2020-10-13	3053.311	1610.973	7202.250	3043.086	6471.840
2020-10-14	3060.437	1628.107	7232.980	3059.531	6487.813
2020-10-15	3067.847	1631.561	7261.127	3087.919	6507.615
2020-10-16	3073.989	1635.983	7285.901	3119.179	6527.122
2020-10-17	3078.469	1641.095	7308.143	3149.760	6547.513

If we visualize it using ggplot:

```r
ggplot(covid_plot, aes(x=date)) +
  geom_line(aes(y=NYC_case_per_100k, color="blue")) +
  geom_line(aes(y=LA_case_per_100k, color="green")) +
  geom_line(aes(y=SF_case_per_100k, color="red")) +
  geom_line(aes(y=Chicago_case_per_100k, color="purple")) +
  geom_line(aes(y=Miami_case_per_100k, color="yellow")) +
  scale_color_discrete(name = "Cities", labels = c("NYC", "LA", "SF", "Chicago", "Miami")) +
  labs(title="Reported Cases per 100K Population in 5 selected cities") +
  xlab("Date") +
  ylab("Reported Cases Per 100K Population") +
  scale_y_continuous(trans = "log")
```



The number of confirmed cases in the five cities during the 13-days period doesn't fluctuate much. But we can see there are significant differences between cities. It may suggest there might be difference in sentiment score across the five cities.

In the next session, I will put the dataset into testing using the package *tidytext* and *syuzhet* to calculate the sentiment score, and trying to see if there are differences in sentiment score between cities.

Result of Analysis

The Code below is how to calculate sentiment score using the *tidytext* and *syuzhet* library.

```r
library(tidytext)
library(syuzhet)
sample_text <- tweet_dataset %>%
  filter(location == "Los Angeles") %>%
  select(text, tweetcreatedts)

sample_text_a <- sample_text %>%
  select(text) %>%
  unlist()

result_score <- data.frame(Score=numeric(0), Location=character(0))

for (i in 1:length(sample_text_a)){
  score <- as.numeric(mean(get_sentiment(get_tokens(sample_text_a[i]), method="afinn")))
  location = "Los Angeles"
  result_row = data.frame(Score=score, Location=location)
  result_score <- rbind(result_score, result_row)
}

result_score <- cbind(sample_text, result_score)
### End of Location 1
sample_text2 <- tweet_dataset %>%
  filter(location == "San Francisco") %>%
  select(text, tweetcreatedts)

sample_text2a <- sample_text2 %>%
  select(text) %>%
  unlist()

result_score2 <- data.frame(Score=numeric(0), Location=character(0))

for (i in 1:length(sample_text2a)){
  score <- as.numeric(mean(get_sentiment(get_tokens(sample_text2a[i]), method="afinn")))
  location = "San Francisco"
  result_row = data.frame(Score=score, Location=location)
  result_score2 <- rbind(result_score2, result_row)
}

result_score2 <- cbind(sample_text2, result_score2)

### End of Location 2

sample_text3 <- tweet_dataset %>%
  filter(location == "Chicago") %>%
  select(text, tweetcreatedts)

sample_text3a <- sample_text3 %>%
  select(text) %>%
  unlist()
```

```r
result_score3 <- data.frame(Score=numeric(0), Location=character(0))

for (i in 1:length(sample_text3a)){
  score <- as.numeric(mean(get_sentiment(get_tokens(sample_text3a[i]), method="afinn")))
  location = "Chicago"
  result_row = data.frame(Score=score, Location=location)
  result_score3 <- rbind(result_score3, result_row)
}

result_score3 <- cbind(sample_text3, result_score3)

### End of Location 3
sample_text4 <- tweet_dataset %>%
  filter(location == "Miami") %>%
  select(text, tweetcreatedts)

sample_text4a <- sample_text4 %>%
  select(text) %>%
  unlist()

result_score4 <- data.frame(Score=numeric(0), Location=character(0))

for (i in 1:length(sample_text4a)){
  score <- as.numeric(mean(get_sentiment(get_tokens(sample_text4a[i]), method="afinn")))
  location = "Miami"
  result_row = data.frame(Score=score, Location=location)
  result_score4 <- rbind(result_score4, result_row)
}

result_score4 <- cbind(sample_text4, result_score4)

### End of Location 4
sample_text5 <- tweet_dataset %>%
  filter(location == "New York") %>%
  select(text, tweetcreatedts)

sample_text5a <- sample_text5 %>%
  select(text) %>%
  unlist()

result_score5 <- data.frame(Score=numeric(0), Location=character(0))

for (i in 1:length(sample_text5a)){
  score <- as.numeric(mean(get_sentiment(get_tokens(sample_text5a[i]), method="afinn")))
  location = "New York"
  result_row = data.frame(Score=score, Location=location)
  result_score5 <- rbind(result_score5, result_row)
}

result_score5 <- cbind(sample_text5, result_score5)
### End of Location 5

result_combine <- rbind(result_score, result_score2, result_score3, result_score4, result_score5)
result_combine <- na.omit(result_combine)
```

```r
table_5 <- result_combine %>%
  group_by(Location) %>%
  summarise(n=n(),
            Mean=mean(Score),
            SD=sd(Score),
            Variance=var(Score),
            Median= median(Score))
```

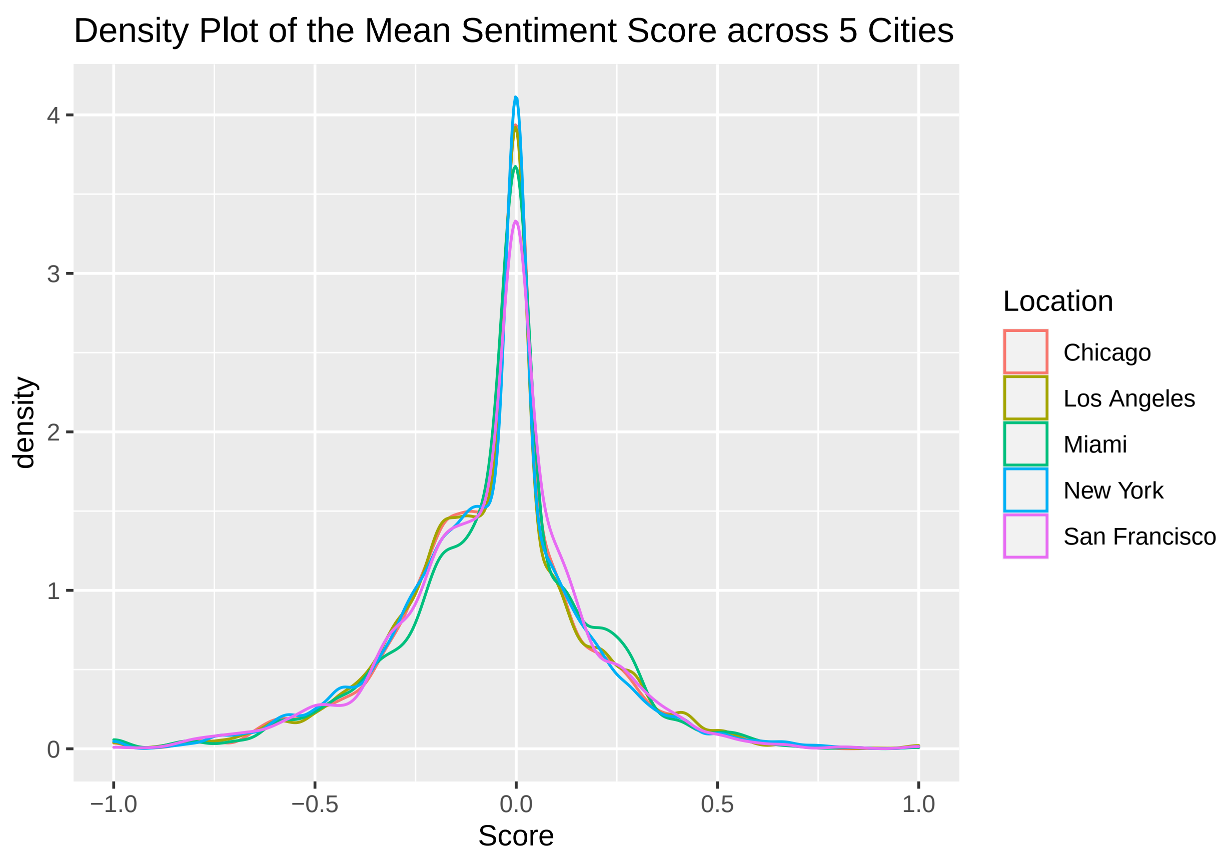
Table 6: Summary Statistics of Sentiment Score in 5 Cities

Location	Number of Tweets	Mean	Standard Deviation	Variance	Median
Chicago	4201	-0.0617642	0.2505062	0.0627533	0
Los Angeles	5862	-0.0616610	0.2520286	0.0635184	0
Miami	1388	-0.0471243	0.2395445	0.0573816	0
New York	8904	-0.0652148	0.2540056	0.0645188	0
San Francisco	2464	-0.0575869	0.2419290	0.0585296	0

From Table 5, we can see that for the mean score for all 5 cities is having negative number. It suggest that the average sentiment for all cities are slightly negative. However, since in the “afinn” lexicon library in R, the score ranges between -5 to 5, the mean score here doesn’t reflect that it is very negative. In fact, it is really close to 0, incidicating actually it is close to netural.

```r
# Plotting the Density Plot of Sentiment Score
ggplot(result_combine, aes(x=Score, col=Location)) +
  geom_density() +
  xlim(c(-1, 1)) +
  labs(title="Density Plot of the Mean Sentiment Score across 5 Cities")
```

```
## Warning: Removed 74 rows containing non-finite values (stat_density).
```



From the density plot, it doesn't look like normal distribution. But, I also use Shapiro-Wilk normality test to test if the data is following distribution. Since the *shapiro.test* in R only accept data with a maximum of 5000 rows, I will draw a random sample of 1000 to check the normality. Picking 1000 samples for each cities is because for data in Miami, the number of rows is 1390. 1000 samples will fit for all cities.

```
# the normality of data
result_score1a <- result_score %>%
  sample_n(., 1000)
shapiro.test(result_score1a$Score)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  result_score1a$Score
## W = 0.93209, p-value < 2.2e-16
```

```
result_score2a <- result_score2 %>%
  sample_n(., 1000)
shapiro.test(result_score2a$Score)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  result_score2a$Score
## W = 0.91871, p-value < 2.2e-16
```

```r
result_score3a <- result_score3 %>%
  sample_n(., 1000)
shapiro.test(result_score3a$Score)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  result_score3a$Score
## W = 0.90705, p-value < 2.2e-16
```

```r
result_score4a <- result_score4 %>%
  sample_n(., 1000)
shapiro.test(result_score4a$Score)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  result_score4a$Score
## W = 0.93417, p-value < 2.2e-16
```

```r
result_score5a <- result_score5 %>%
  sample_n(., 1000)
shapiro.test(result_score5a$Score)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  result_score5a$Score
## W = 0.92147, p-value < 2.2e-16
```

In here we know that: For Shapiro-Wilk Normality Test, the hypothesis are as follows.

$$H_0 : X \sim N(\mu, \sigma^2)$$
$$H_1 : X \nsim N(\mu, \sigma^2)$$

The alternative hypothesis suggest that the data is not following Normal Distribution. From the five tests above, the p-value for all 5 cities is less than 0.05, our desired confidence level, suggesting that rejecting the null hypothesis. Hence, concluding that the five cities data is not following normal distribution, we can not use parametric ANOVA to compare their means and distribution.

So, I performed Wilcoxon rank sum test to test pairwisely to see if all 5 cities are come from the same distribution or they have differences.

```r
# Testing the dataset
# Need to use Wilcoxon rank sum test # A Nonparametric Test
wilcox.test(result_score$Score, result_score2$Score)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  result_score$Score and result_score2$Score
## W = 7058760, p-value = 0.1004
## alternative hypothesis: true location shift is not equal to 0
```

```r
wilcox.test(result_score$Score, result_score3$Score)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  result_score$Score and result_score3$Score
## W = 12225343, p-value = 0.5379
## alternative hypothesis: true location shift is not equal to 0
```

```r
wilcox.test(result_score$Score, result_score4$Score)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  result_score$Score and result_score4$Score
## W = 3880320, p-value = 0.006876
## alternative hypothesis: true location shift is not equal to 0
```

```r
wilcox.test(result_score$Score, result_score5$Score)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  result_score$Score and result_score5$Score
## W = 26124333, p-value = 0.9154
## alternative hypothesis: true location shift is not equal to 0
```

```r
wilcox.test(result_score2$Score, result_score3$Score)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  result_score2$Score and result_score3$Score
## W = 5259724, p-value = 0.2635
## alternative hypothesis: true location shift is not equal to 0
```

```r
wilcox.test(result_score2$Score, result_score4$Score)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  result_score2$Score and result_score4$Score
## W = 1670520, p-value = 0.2292
## alternative hypothesis: true location shift is not equal to 0
```

```r
wilcox.test(result_score2$Score, result_score5$Score)
```

```
##
##  Wilcoxon rank sum test with continuity correction
```

```
##
## data:  result_score2$Score and result_score5$Score
## W = 11228727, p-value = 0.07025
## alternative hypothesis: true location shift is not equal to 0
```

```
wilcox.test(result_score3$Score, result_score4$Score)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  result_score3$Score and result_score4$Score
## W = 2799854, p-value = 0.02508
## alternative hypothesis: true location shift is not equal to 0
```

```
wilcox.test(result_score4$Score, result_score5$Score)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  result_score4$Score and result_score5$Score
## W = 6470487, p-value = 0.004361
## alternative hypothesis: true location shift is not equal to 0
```

In Wilcoxon rank sum test, the null hypothesis means the two groups are coming from the same distribution, while the alternative hypothesis suggest that "true location shift is not equal to 0". In other words, it tells that the two groups are not coming from the same distribution.

The result above suggest that the cities doesn't makes a difference in terms of sentiment score, given that the confirmed cases per 100K population is different, since most of the p-value is greater than 0.05, suggesting that we can not reject the null hypothesis.

Comments and Future Work

It is fun to understand what people thinks or believes based on locations. In this research project, although the result suggest that there might be no association between the sentiment and the level of natural disaster (In this case I used the number of confirmed COVID-19 cases), it is worth looking further to see how location affects the way people think or receive the same piece of message.

In the mean time, I believe there are more that I can do to further investigate this kind of relationship, by a few ways:
- I can study the change in sentiment and the confirmed COVID-19 cases for a longer period. (i.e. 3 to 6 months) - The limitation currently facing is because I am using a free Twitter Developer Account. There is limit on the number of tweets that I can get in a given time period (15 Minute Now). Also, for free Developer Account, they are not allowing me to search pass tweets. I can only obtain same day's tweet as I run the API. This two limitation greatly reduce the ability to obtain large data set for sentiment analysis.

- Breaking the tweet texts into parts of sentence, perform tokenization, and avoiding word by word sentiment analysis.
  - In this research project, when I apply the get_sentiment() function, R will give a sentiment score word by word. Usually, in our sentence, most of the words are netural and do not carry any sentiment score. If the weighting for each words are the same, when calculating the sentiment score for a tweet (In this case I take the average of each words), a lot of words will shift the sentiment score tends to 0. There are more advanced skills like transforming the tweets into sections of phrases. Consider this sentence: "The brown fox is quick and he is jumping over the lazy dog", if we transform it into parts of sentence, or shallow Parsing or Chunking, the sentence will become "The brown fox | is | quick | and | he | is jumping | over | the lazy dog", which the sentment score are less affected by netural words, even we removed stop words.
- Apply Machine Learning Algorithm
  - Machine Learning algorithms are more modern choice of doing sentiment analysis as it can train models to perform classification, regression on prediction of sentiment score. As we feed more training set and test set to the model, adjust, and getting feedback, we can update with new English words frequently, especially with newly created words from the internet.

References and Appendix

Reference

Text Mining with R Gathering and Cleaning data
https://towardsdatascience.com/text-mining-with-r-gathering-and-cleaning-data-8f8b0d65e67c

An Introduction to Cleaning of Text
https://cran.r-project.org/doc/contrib/de_Jonge+van_der_Loo-Introduction_to_data_cleaning_with_R.pdf

Tidy Text Mining Method in R
https://www.tidytextmining.com/twitter.html

Obtaining Population Details from US Census
https://www.census.gov/data/academy/courses/ranking-project.html

KDnuggets Understand Language Syntax
https://www.kdnuggets.com/2018/08/understanding-language-syntax-and-structure-practitioners-guide-nlp-3.html

Guidotti, E., Ardia, D., (2020), "COVID-19 Data Hub", Journal of Open Source Software 5(51):2376
https://covid19datahub.io/

Appendix

Data Wrangling of Twitter Text in R For data cleaning, I use functions to make the codes more tidy.

```r
#Load required library
library(dplyr)
```

```r
#### Function to filter Location Text
location_cleaning_function <- function(df1){
  count <- 1
  for (i in df1$location){
    if (grepl("San Francisco", i, ignore.case=TRUE) == TRUE)
    {
      location_text <- "San Francisco"
    }
    else if (grepl("New York", i, ignore.case=TRUE) == TRUE)
    {
      location_text <- "New York"
    }
    else if (grepl("Los Angeles", i, ignore.case=TRUE) == TRUE)
    {
      location_text <- "Los Angeles"
    }
    else if (grepl("Miami", i, ignore.case=TRUE) == TRUE)
    {
      location_text <- "Miami"
    }
    else if (grepl("Chicago", i, ignore.case=TRUE) == TRUE)
    {
      location_text <- "Chicago"
    }
    else
    {
      location_text <- i
    }
    df1$location[count] <- location_text
    count = count + 1
  }
  return(df1)
}
```

```r
### Remove Special Characters:
main_text_clean <- function(df2){
  count <- 1
  for (text in df2$text) {
    # Set the text to lowercase
    text <- tolower(text)
    # Remove mentions, urls, emojis, numbers, punctuations, etc.
    text <- gsub("@\\w+", "", text)
    text <- gsub("https?://.+", "", text)
    text <- gsub("\\d+\\w*\\d*", "", text)
    text <- gsub("#\\w+", "", text)
    text <- gsub("[^\x01-\x7F]", "", text)
    text <- gsub("[[:punct:]]", " ", text)
    # Remove spaces and newlines
    text <- gsub("\n", " ", text)
    text <- gsub("^\\s+", "", text)
    text <- gsub("\\s+$", "", text)
    text <- gsub("[ |\t]+", " ", text)
    df2$text[count] <- text
    count <- count + 1
  }
  return(df2)
}
```

```
library(qdap)
## Remove Stop Words
main_text_clean_2 <- function(df3){
  count <- 1
  for (text in df3$text){
    text <- rm_stopwords(text, stopwords = Top200Words, separate=FALSE)
    df3$text[count] <- text
    count <- count + 1
  }
  return(df3)
}
```

```
data <- read.csv("complete_covid_tweet.csv") # Import Downloaded Data
data <- data[c(2:5)] #Drop First Index Column

data_1 <- location_cleaning_function(data) # Location Text Clean

data_2 <- data_1 %>%
  mutate(tweetcreatedts = as.Date(tweetcreatedts)) #### Filter Date, (as.Date())

data_3 <- main_text_clean(data_2) # Remove Punctation
write.csv(data_3, "tweet_cleaned_withstopwords.csv") # For Tweet Stat Analysis

data_4 <- main_text_clean_2(data_3) #Remove Stopwords
write.csv(data_4, "tweet_cleaned.csv") # For Sentiment Analysis
```

Python Code connecting to Twitter API Tweepy Library is one of the Python API that is easy to use to connect to Twitter and Scrap the required tweets, location, time of tweet creations, username.

Since I only analyze the tweets with location and perform sentiment analysis, I do not need to include usernames, userID, and various other information that will reveal individual user's identification. In the code stage I already do not include the extraction of user information.

```python
from tweepy import OAuthHandler
from tweepy.streaming import StreamListener
import tweepy
import json
import pandas as pd
import csv
import re
from textblob import TextBlob
import string
import preprocessor as p
import os
import time
```

```python
# Twitter credentials
# Obtain them from your twitter developer account
consumer_key = "aaaaaaaaaaaaaaaaaaaaaaaaaa"
consumer_secret = "bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb"
access_key = "cccccccccccccccccccccccccccccccccc"
access_secret = "dddddddddddddddddddddddddddddddd"
# Pass your twitter credentials to tweepy via its OAuthHandler

auth = OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_key, access_secret)
api = tweepy.API(auth)
```

```python
def scraptweets(search_words, date_since, numTweets, numRuns):

    # Define a for-loop to generate tweets at regular intervals
    # We cannot make large API call in one go. Hence, let's try T times

    # Define a pandas dataframe to store the date:
    db_tweets = pd.DataFrame(columns = ['location', 'text', 'hashtags', 'tweetcreatedts'])
    program_start = time.time()
    for i in range(0, numRuns):
        # We will time how long it takes to scrape tweets for each run:
        start_run = time.time()

        # Collect tweets using the Cursor object
        # .Cursor() returns an object that you can iterate or
        # loop over to access the data collected.
        # Each item in the iterator has various attributes that
        # you can access to get information about each tweet
        tweets = tweepy.Cursor(api.search, q=search_words, lang="en",
                               since=date_since, tweet_mode='extended').items(numTweets)
        # Store these tweets into a python list
        tweet_list = [tweet for tweet in tweets]
        # Obtain the following info (methods to call them out):
```

```
        # user.location - where is he tweeting from
        # created_at - when the tweet was created
        # retweeted_status.full_text - full text of the tweet
        # tweet.entities['hashtags'] - hashtags in the tweet# Begin scraping the tweets individually:
        noTweets = 0

        for tweet in tweet_list:# Pull the values
            location = tweet.user.location
            hashtags = tweet.entities['hashtags']
            tweetcreatedts = tweet.created_at
            try:
                text = tweet.retweeted_status.full_text
            except AttributeError:  # Not a Retweet
                text = tweet.full_text
                # Add the 11 variables to the empty list - ith_tweet:
            if 'san francisco' in location.lower():
                ith_tweet = [location, text, hashtags, tweetcreatedts]
                db_tweets.loc[len(db_tweets)] = ith_tweet
                # increase counter - noTweets
                noTweets += 1
            elif 'new york' in location.lower():
                ith_tweet = [location, text, hashtags, tweetcreatedts]
                db_tweets.loc[len(db_tweets)] = ith_tweet
                # increase counter - noTweets
                noTweets += 1
            elif 'los angeles' in location.lower():
                ith_tweet = [location, text, hashtags, tweetcreatedts]
                db_tweets.loc[len(db_tweets)] = ith_tweet
                # increase counter - noTweets
                noTweets += 1
            elif 'miami' in location.lower():
                ith_tweet = [location, text, hashtags, tweetcreatedts]
                db_tweets.loc[len(db_tweets)] = ith_tweet
                # increase counter - noTweets
                noTweets += 1
            elif 'chicago' in location.lower():
                ith_tweet = [location, text, hashtags, tweetcreatedts]
                db_tweets.loc[len(db_tweets)] = ith_tweet
                # increase counter - noTweets
                noTweets += 1
            else:
                pass
            # Append to dataframe - db_tweets

        # Run ended:
        end_run = time.time()
        duration_run = round((end_run-start_run)/60, 2)

        print('no. of tweets scraped for run {} is {}'.format(i + 1, noTweets))
        print('time take for {} run to complete is {} mins'.format(i+1, duration_run))
        if numRuns == 1:
            pass
        else:
            for i in range(15):
                time.sleep(61) #15 minute sleep time
                print(str(i+1) + " minutes of wait time passed.")

    # Once all runs have completed, save them to a single csv file:
    from datetime import datetime
    # Obtain timestamp in a readable format
    to_csv_timestamp = datetime.today().strftime('%Y%m%d')# Define working path and filename
    path = os.getcwd()
    filename = path + '/data/' + to_csv_timestamp + '_selective_cities_covid_tweets.csv'
    # Store dataframe in csv with creation date timestamp
    db_tweets.to_csv(filename, index = False)

    program_end = time.time()
    print('Scraping has completed!')
    print('Total time taken to scrap is {} minutes.'.format(round(program_end - program_start)/60, 2))
```

```
# Keywords:
search_words = "#covid-19 OR #coronavirus OR #virus OR #COVID-19 OR #health OR #CDC OR #facemask OR #fa
```

```
## Setting the range of tweets and number of total tweets to be scraped
date_since = "2020-03-17"
numTweets = 2500
numRuns = 20# Call the function scraptweets
scraptweets(search_words, date_since, numTweets, numRuns)
```