Liad Nehemia

The Media Impact On Society

#Twint
#Vader
#TextBlob

Mass Media

The influence of *mass media* has an effect on many aspects of human life, which can include voting a certain way, individual views and beliefs, or skewing a person's knowledge of a specific topic due to being provided **false** information.

How fantasy becomes reality?

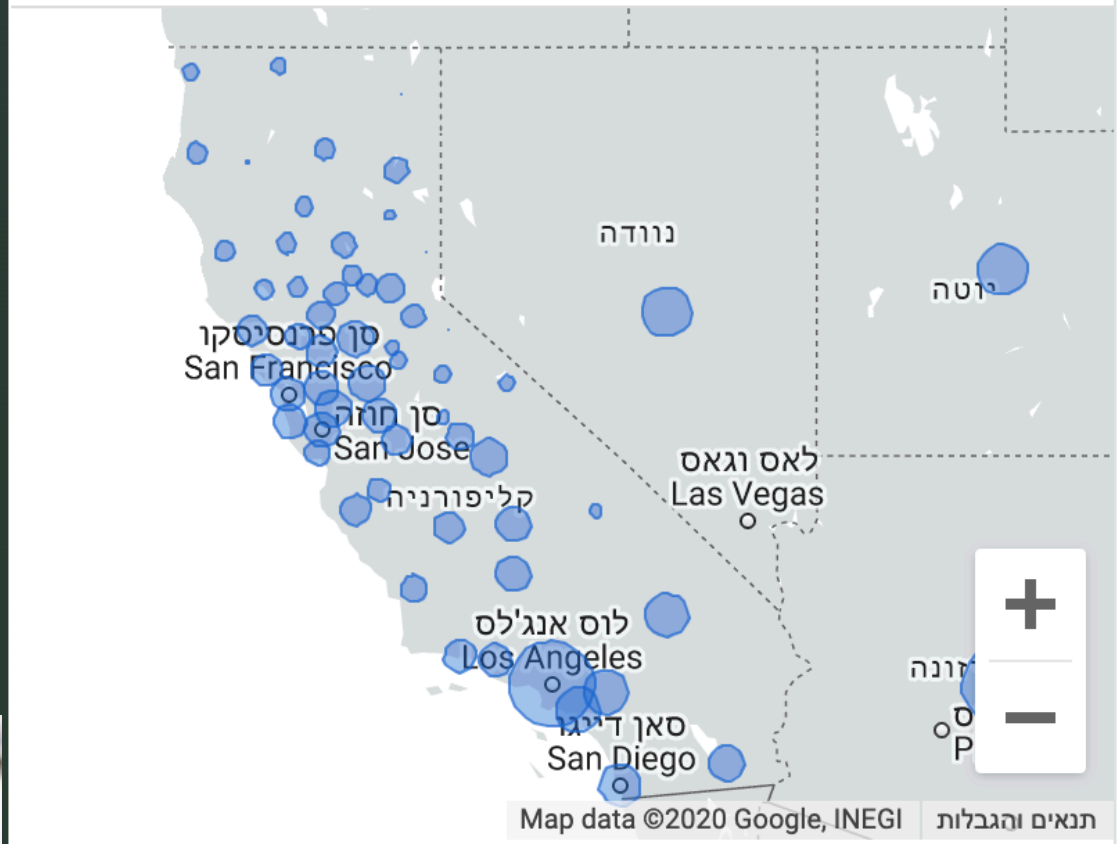
It is a widely held belief that because we understand that mass media stories are fantasies, they cannot affect our realities. Karen E. Dill argues against that premise using research and theory in social psychology. Although mass media scholarship demonstrates important and consistent media effects, many of us are either unaware of those effects or disbelieve that they could be true, thus becoming media apologists who enable our own manipulation.

“How fantasy becomes reality: Seeing Through Media Influence” Book by “Karen E. Dill”

Covid-19 at California:

- 39.5 million residents
- Governor: Gavin Newsom

15/7/2020



מקורות: ויקיפדיה וכן הניו יורק טיימס. · מידע על נתונים אלו

סקירת מקרים

קליפורניה

מקרים מאומתים	חולים שהחלימו	אנשים שנפטרו
347K	-	7,227

Web-Sites Scraped

- Los Angeles Times https://www.latimes.com/
- Time https://time.com/
- Ynet https://www.ynetnews.com/
- N12 https://www.n12.co.il/

Format:

```
data = {
    'title': article.title,
    'date_published': article.publish_date,
    'news_outlet': self.newspaper,
    'authors': article.authors,
    'feature_img': article.top_image,
    'article_link': article.canonical_link,
    'keywords': article.keywords,
    'movies': article.movies,
    'summary': article.summary,
    'text': article.text,
    'html': article.html
}
```

Code: in Python

```
class NewspaperScraper:
    def __init__ (self, newspaper, searchTerm, dateStart, dateEnd):
        self.newspaper = newspaper
        self.searchTerm = searchTerm
        self.dateStart = parse(dateStart)
        self.dateEnd = parse(dateEnd)
        self.links = []

    def get_newspaper_name (self):
        return self.newspaper

    def get_pages (self):
        print('Unimplemented for ' + self.newspaper + ' scraper')
        return

    def check_dates (self, date):
        print("Checking date...")
        page_date = parse(date)
        if page_date >= self.dateStart and page_date <= self.dateEnd:
            print("Date is fine..")
            return True
        return False

    def newspaper_parser (self, sleep_time=1.5):
        print('running newspaper_parser()...')

        results = []
        categories = []
        count = 0
```

```
        if self.newspaper == "LaTimes":
            categories =
["california","entertainment","sports","environment","opinion
","world-nation",

"travel","science","politics","housing","climate","business"]
        for l in self.links:
            print("Try to extract article from : ", l)
            category = "None"
            for cat in categories:
                if cat in l:
                    #print("Yes!" , cat)
                    category = cat
            article = Article(url=l)
            try:
                article.build()
            except Exception as e:
                print("Error loading article")
                print(e)
                time.sleep(10)
            data = {
                'title': article.title,
                'date_published': article.publish_date,
                'news_outlet': self.newspaper,
                'authors': article.authors,
                'feature_img': article.top_image,
                'article_link': article.canonical_link,
                'keywords': article.keywords,
                'movies': article.movies,
                'summary': article.summary,
                'text': article.text,
                'html': article.html,
                'category': category
            }
            if data:
                results.append(data)
                count += 1
                print(count)
            time.sleep(sleep_time)
        return results
```

Code: in Python

```
def write_to_csv (self, data, file_name):
    print('writing to CSV...')

    keys = data[0].keys()
    with open(file_name, 'wb') as output_file:
        dict_writer = csv.DictWriter(output_file, keys)
        dict_writer.writeheader()
        dict_writer.writerows(data)

def write_to_mongo (self, data, collection):
    print('writing to mongoDB...')
    count = 0

    for d in data:
        collection.insert(d)
        count += 1
        print(count)
```

Export

Code: in Python

LaTimes news scraper

```
class LaTimesScraper(NewspaperScraper):
    def get_pages (self, sleep_time=3,num_of_pages = 50):
        print('running get_pages().HELLO..')
        profile = webdriver.FirefoxProfile()
        browser = webdriver.Firefox(profile)
        links = []
        stop = False
        index = 1
        while not stop:
            try:
                browser.get('http://www.latimes.com/search?q='
                    + self.searchTerm
                    + "&p="
                    + str(index)
                 )
            except:
                stop = True

            try:
                soup = BeautifulSoup(browser.page_source,'html.parser')
                print("Working on ",browser.current_url , "index:",index)
            except Exception as e:
                print("[ERROR]:Working on ",browser.current_url , "index:",index)
                stop = True
```

```
#print(soup.prettify())
    if not soup.find('ul', class_='search-results-module-
results-menu'):
            print("Error 304")
            stop = True
        for result in soup.find_all('div', class_='promo-wrapper'):
            pub_date = result.find('p', class_='promo-
timestamp').get('data-date')
            print("pub_date:",pub_date)
            if self.check_dates(pub_date):
                head_line = result.find('p', class_='promo-title')
                head_line = str(head_line)
                if head_line:
                    t = head_line.split('>')
                    #print("headline:", t)
                    link = t[1].split('href=')[1]
                    link = link.strip('\"')
                    if link not in links:
                        print("Adding link", link)
                        links.append(link)
            else:
                print("Date is not in range")
        index += 1
        if index > num_of_pages + 1:
            stop = True
        time.sleep(sleep_time)
    browser.close()
    self.links = links
    return links
```

Code: in Python

Time news scraper

```
class TimeScraper(NewspaperScraper):
    def get_pages (self, sleep_time=3,num_of_pages = 25):
        print('running get_pages().TimeScraper..')
        profile = webdriver.FirefoxProfile()
        browser = webdriver.Firefox(profile)
        links = []
        sub_links = []
        stop = False
        index = 1
        while not stop:
            browser.get('https://time.com/search/?q='
                + self.searchTerm
                + "&page="
                + str(index)
             )
            try:
                soup = BeautifulSoup(browser.page_source,'html.parser')
                print("Working on ",browser.current_url , "index:",index)
            except Exception as e:
                print("[ERROR]:Working on ",browser.current_url , "index:",index)
                print(e)
                stop = True
            if not soup.find('div', class_='search-results-content'):
                print("Error 304")
                stop = True
            data = soup.findAll('div', attrs={'class': 'media-body'})
            for div in data:
                links = div.findAll('a')
                for a in links:
                    print("link:",a['href'])
                    if("https://time.com" in a['href']):
                        print("Adding ", a['href'])
                        sub_links.append(a['href'])
            index += 1
            if index > num_of_pages + 1:
                stop = True
            time.sleep(sleep_time)
        browser.close()
        self.links = sub_links
        return links
```

Code: in Python

Run:

```
client = MongoClient("mongodb+srv://Test1:Test1@cluster0-pwaip.mongodb.net/test?retryWrites=true&w=majority")

db = client.USArticles




def run_scraper (scraper):
    scraper.get_pages()
    data = scraper.newspaper_parser()
    #print(data)
    scraper.write_to_mongo(data, db.articles)



#run_scraper(LaTimesScraper("LaTimes","Covid","20-5-2020","25-5-2020"))

run_scraper(TimeScraper("Time","Covid","20-6-2020","25-7-2020"))

#run_scraper(ChicagoTribuneScraper("LaTimes","Covid","1-6-2020","10-6-2020"))
```

Sentiment analysis for Articles in Python

```
client = MongoClient("mongodb+srv://Test1:Test1@cluster0-pwaip.mongodb.net/test?retryWrites=true&w=majority")
db = client.USArticles
num_of_articles = 0
total_articles = []
all_text = ""
news_outlet = "Time"
with client:
    #db = client.Categories
    articles = db.articles
    for x in articles.find({"news_outlet": news_outlet}, {'text': 1,'keywords':1}):
        num_of_articles +=1
        #print(x)
        total_articles.append(x['text'])
```

Read from MongoDB

```
scores =  {"Positive": 0 , "Negative": 0 , "Neutral": 0}
scores_list = []

analyzer = SentimentIntensityAnalyzer()
pos=neg=neu=compound=compound2=0

for article in total_articles:
    idx = 1
    pos = neg = neu = compound = compound2 = 0
    peregraphs = article.split("Advertisement")
    for per in peregraphs:
        sentences = per.split(".")
        for sentence in sentences:
            if(len(sentence) < 3):
                continue
            sent_len = len(sentence.split(" "))
            if sent_len < 1:
                sent_len = 50
            vs = analyzer.polarity_scores(sentence)
            pos += vs["pos"]
            neg += vs["neg"]
            neu += vs["neu"]
            #compound += vs["compound"] / sent_len
            compound2 += vs["compound"]
            idx += 1
    score = compound2/idx
    if score >= 0.05:
        scores["Positive"] += 1
        scores_list.append("Positive")
    elif score <= - 0.05:
        scores["Negative"] += 1
        scores_list.append("Negative")
    else:
        scores["Neutral"] += 1
        scores_list.append("Neutral")
print(scores)
print(scores_list)
```

Calculate polarity

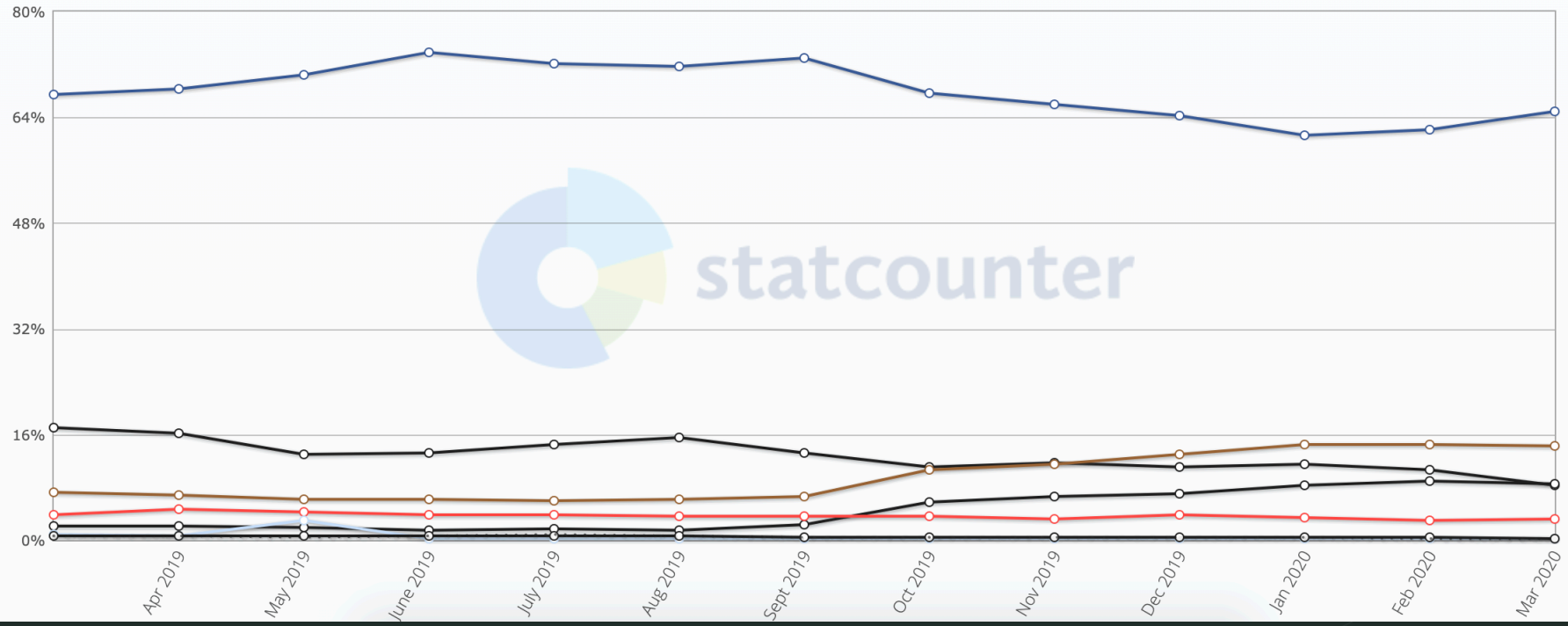
Facebook	Twitter	Instagram	Pinterest	YouTube	Tumblr
64.99%	14.29%	8.54%	8.31%	3.1%	0.28%

Social Media Stats Worldwide - March 2020

Social Media Stats Worldwide

Mar 2019 - Mar 2020

Edit Chart Data



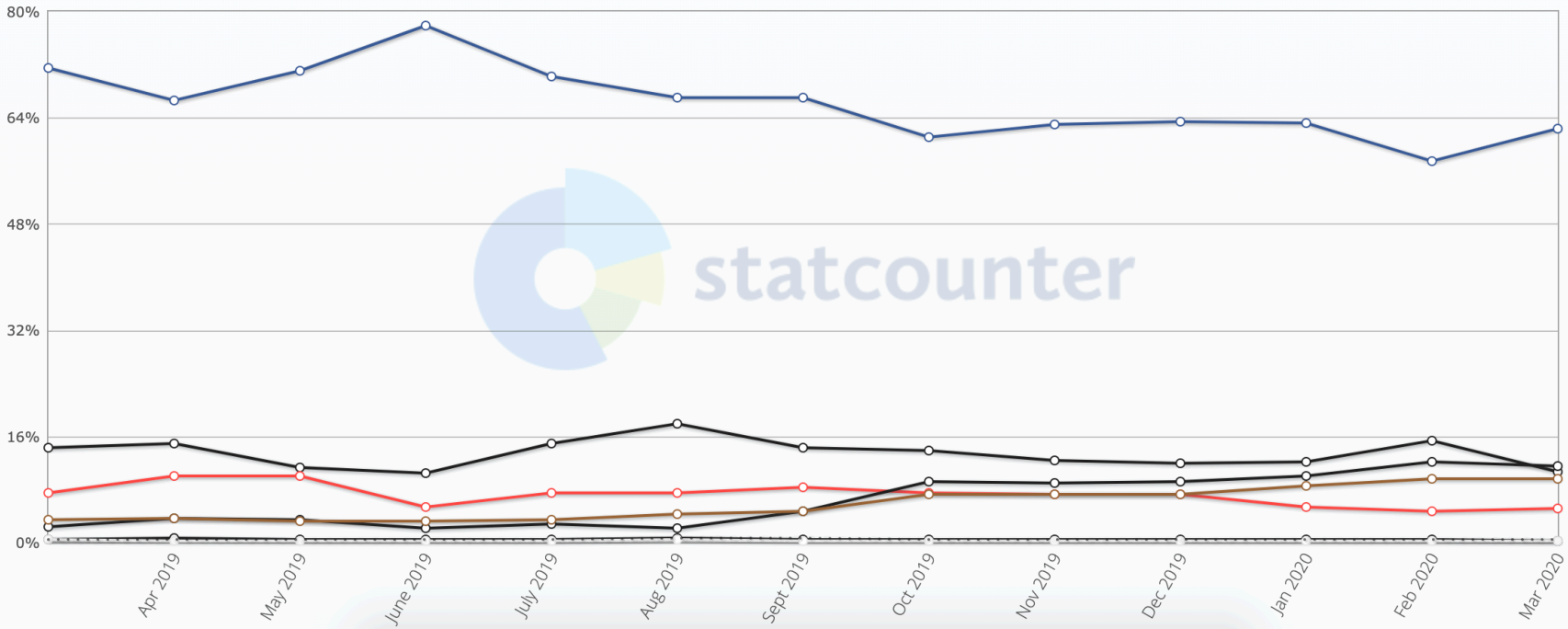
Facebook	Instagram	Pinterest	Twitter	YouTube	Tumblr
62.31%	11.53%	10.67%	9.58%	5.05%	0.3%

Social Media Stats in Israel - March 2020

Social Media Stats Israel

Mar 2019 - Mar 2020

Edit Chart Data



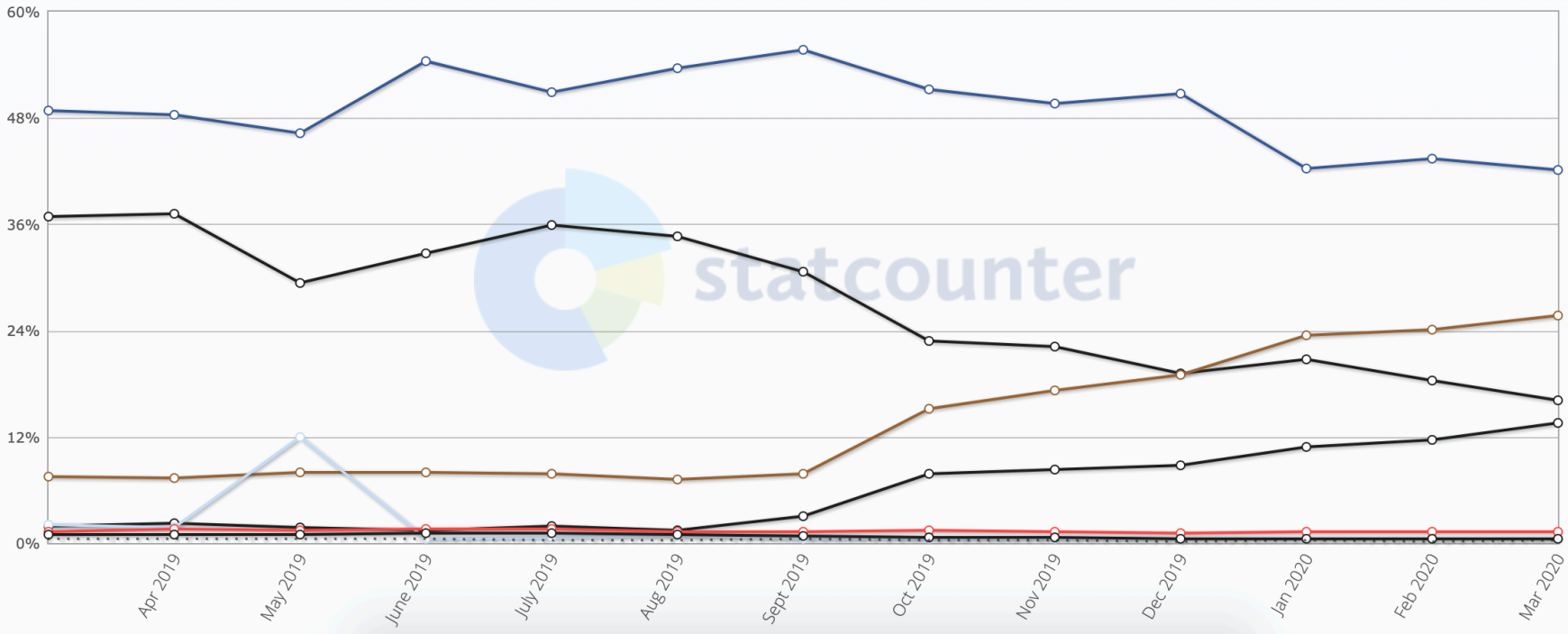
Facebook	Twitter	Pinterest	Instagram	YouTube	Tumblr
42.08%	25.69%	16.11%	13.57%	1.34%	0.5%

Social Media Stats in United States Of America - March 2020

Social Media Stats United States Of America

Mar 2019 - Mar 2020

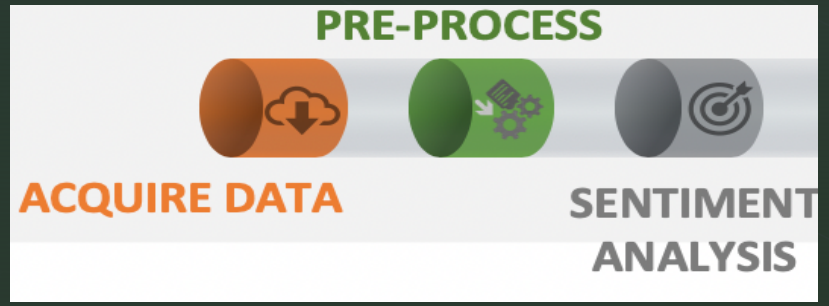
Edit Chart Data



TWINT - Twitter Intelligence Tool

	date	username	tweet	hashtags	likes_count	clean_tweet	sentiment
0	25/05/2020	sraphilly	@GovMurphy your move sir. It's over...News	['#opennjnow']	25	GovMurphy your move sir It s over Newsome jus	Negative
1	25/05/2020	scottholleran	What a terrible shame that we 40 million Ca	[]	10	What a terrible shame that we 40 million Califor	Negative
2	25/05/2020	abc	One California city is grappling with COVID-	[]	78	One California city is grappling with COVID-19 ou	Negative
3	25/05/2020	gregrubini	seems that California is finally RE-	[]	408	seems that California is finally RE-OPENING Cali	Neutral
4	25/05/2020	drewcadell94	I love how California went from a cool 85 de	[]	121	I love how California went from a cool 85 degree	Positive
5	25/05/2020	latimes	California limits church capacity to 25% for r	[]	84	California limits church capacity to 25 for reopen	Negative
6	25/05/2020	newshour	California churches can resume in-person se	[]	73	California churches can resume in-person service	Positive
7	25/05/2020	alanbdavis10	Please send one to Inslee too!	[]	14	Please send one to Inslee too	Neutral
8	25/05/2020	johnbarrowman	Cleaning and organizing the kitchen	['#interference', '#isolation'	270	Cleaning and organizing the kitchen drawers toda	Neutral
9	25/05/2020	relaxiamhere	Translation: here,Äôs your rights back that w	[]	13	Translation here s your rights back that we didn t	Positive
10	25/05/2020	business	NEW: California will limit attendance in chu	[]	61	NEW California will limit attendance in churches	Positive

Scrape tweets by dates and Search:“Covid-19” ,“California”.



Python

Code: in Python

```
class HiddenPrints:
    def __enter__(self):
        self._original_stdout = sys.stdout
        sys.stdout = open(os.devnull, 'w')

    def __exit__(self, exc_type, exc_val, exc_tb):
        sys.stdout.close()
        sys.stdout = self._original_stdout
```

Hide twint output

```
nest_asyncio.apply()
c = twint.Config()
```

```
def available_columns():
    return twint.output.panda.Tweets_df.columns

def twint_to_pandas(columns):
    return twint.output.panda.Tweets_df[columns]

def printmd(string, color=None):
    colorstr = "<span
style='color:{}'>{}</span>".format(color, string)
    display(Markdown(colorstr))
```

Code: in Python

```
def read_tweets(search, limit=1,until = "2020-05-26 00:00:00"):
    c.Search = search
    # Custom output format
    c.Format = "Username: {username} |  Tweet: {tweet}"
    c.Limit = limit
    c.Popular_tweets = True
    c.Lang = "en"
    # c.Translate = True

    #c.Geo(lat=36.778,lng=-119.417)
    #c.Geo("36.778259, -119.417931,10km")
    # location = 'California'
    # c.Near(location)
    c.Store_csv = True
    # c.Since = '2019-05-25'
    c.Until = until
    c.Pandas = True
    c.Output = "TopPopularAfter25.csv"
    twint.run.Search(c)

    with HiddenPrints():
        twint.run.Search(c)
```

```
for x in range(5):
    day = 25
    day = day + x
    day = str(day) + " 00:00:00"
    date = "2020-05-" + day
    read_tweets("covid",
limit=1500,until=date)
```

5 days scraping

Code: in Python

```
def sentiment_analysis():
    all_data = pd.read_csv("TopPopularAfter25.csv", converters={"hashtags": literal_eval})
    all_data = all_data.drop_duplicates(subset='id', keep="first")


    data = pd.DataFrame(all_data[["date", "username", "tweet", "hashtags", "likes_count"]])
    # Regex pattern for only alphanumeric, hyphenated text with 2 or more chars
    data.dropna(inplace=True)
    pattern = re.compile(r"[A-Za-z0-9\-]{1,50}")
    data['clean_tweet'] = data['tweet'].str.findall(pattern).str.join(' ')
    # print(data.head())
    # Transform Pandas DF to Optimus/Spark DF
    # df = op.create.data_frame(pdf=df_pd)
    # #
    # # # Clean tweets
    # clean_tweets = df.cols.remove_accents("tweet") \
    #     .cols.remove_special_chars("tweet")
    #
    # # # Add sentiment to final DF

    return data
```

```
df_result = sentiment_analysis()
```

Prepare data for sentiment analysis

Code: in Python

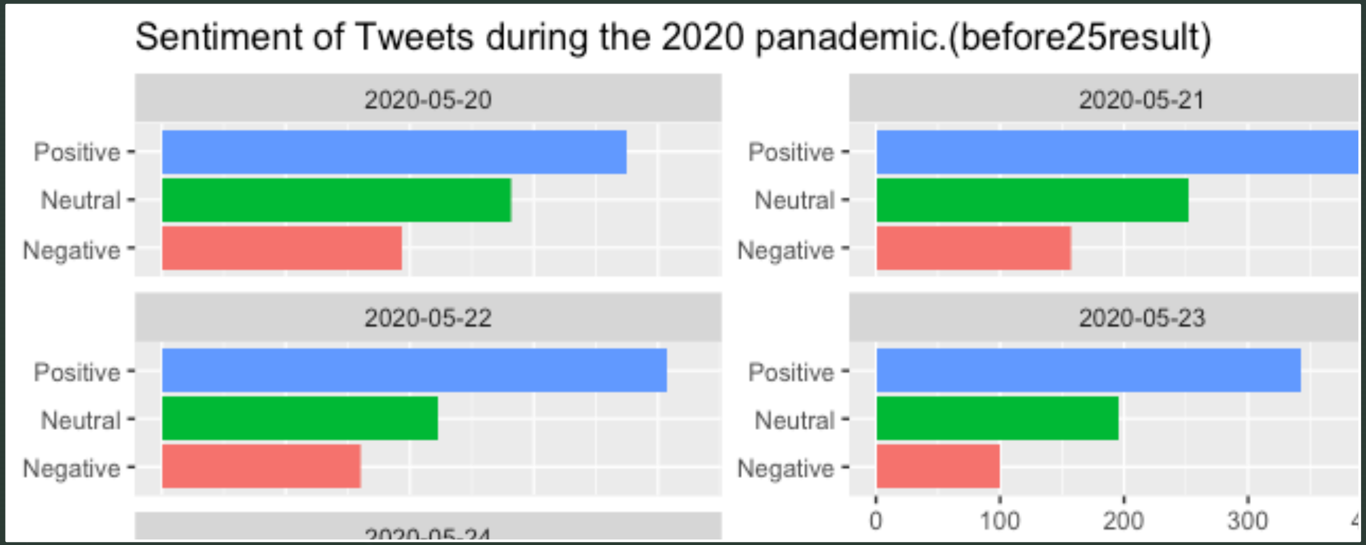
```
def apply_blob(sentence):
    temp = TextBlob(sentence).sentiment[0]
    print("TExtBlob:",temp)
    if temp == 0.0:
        return "Neutral" # Neutral
    elif temp >= 0.0:
        return "Positive" # Positive
    else:
        return "Negative"# Negative
```

```
tweets = df_result['clean_tweet'].tolist()
sent_list = []
for tweet in tweets:
    answer = apply_blob(tweet)
    #print("Tweet:",tweet)
    sent_list.append(answer)
    #print("Senti:", answer)
```

```
df_result["sentiment"] = sent_list
df_result.to_csv("TopPopularAfter25result.csv")
```

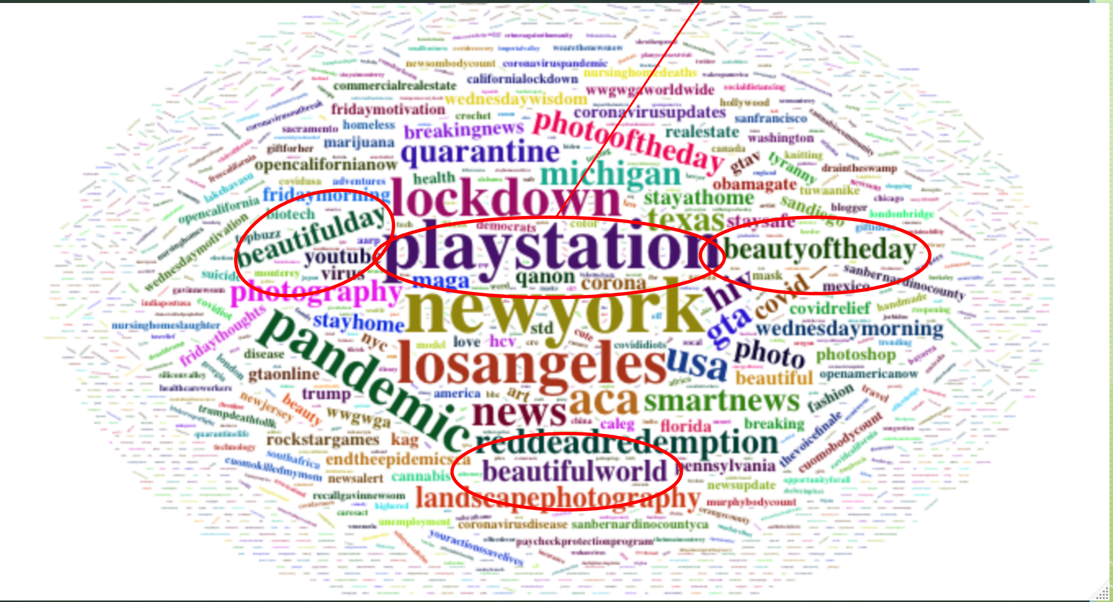
Lots of data, lots of questions.

- George Floyd, 46, died after being arrested by police outside a shop in Minneapolis, Minnesota.(May 25)
- Lets see what are the Media & Social Media changes at California area.



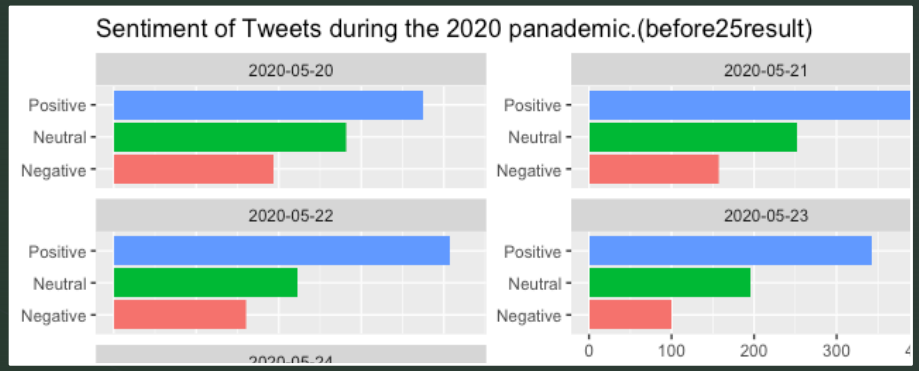
wordcount

#HashTags



Code: in R.

```
data <- read.csv("PycharmProjects/untitled17/before25result.csv")

data["y"] <- 1

#data$by_date <- factor(data$date,levels = unique(data$date))

ggplot(data=data,aes(sentiment,y, fill = sentiment)) +

  geom_col(show.legend = FALSE) +

  facet_wrap(~date, scales = "free_y", ncol = 2) +

  labs(title = "Sentiment of Tweets during the 2020
panademic.(before25result)",

       y = "At May 25 George Floyd killed during an arrest in Minneapolis.",

       x = NULL) +

  coord_flip()
```



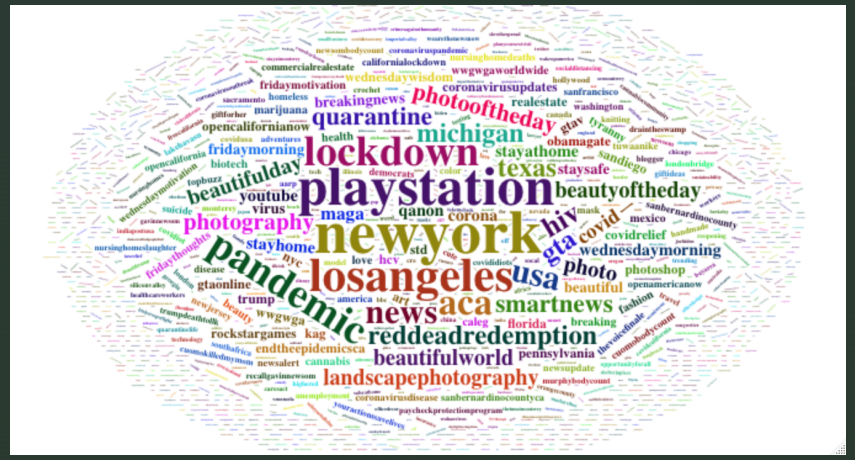
Code: in R.

WordCount

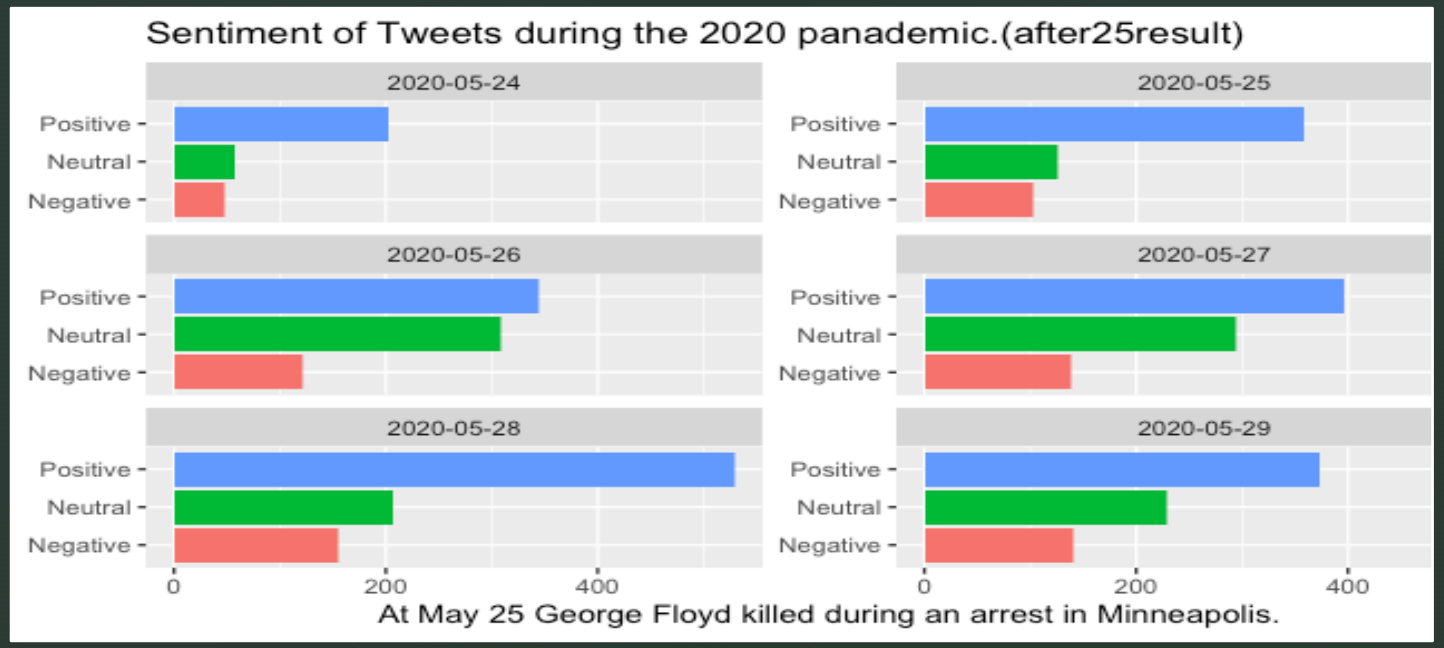

```
data <- read.csv('PycharmProjects/untitled17/TopPopularAfter25result.csv')

text <- data$clean_tweet

docs <- Corpus(VectorSource(text))

docs <- docs %>%tm_map(removeNumbers) %>% tm_map(removePunctuation) %>%tm_map(stripWhitespace)

docs <- tm_map(docs, content_transformer(tolower))

docs <- tm_map(docs, removeWords, stopwords("english"))

dtm <- TermDocumentMatrix(docs)

matrix <- as.matrix(dtm)

words <- sort(rowSums(matrix),decreasing=TRUE)

df <- data.frame(word = names(words),freq=words)

set.seed(1234)

#wordcloud(words = df$word, freq = df$freq, min.freq = 1,max.words=200, random.order=FALSE,
rot.per=0.35,colors=brewer.pal(8, "Dark2"))

write.csv(df,'TopPopularAfter25resultWordCount.csv')

dftest <- read.csv('TopPopularAfter25resultWordCount.csv')

wordcloud2(data=dftest, size=0.6, color='random-dark')
```

Code:in R.

HashTags

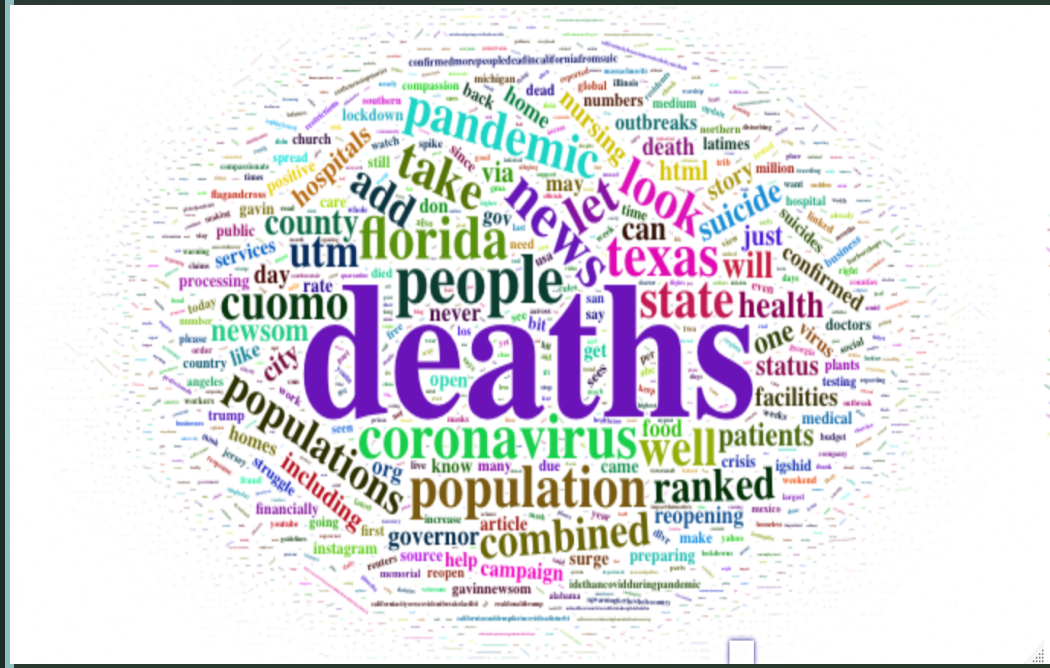


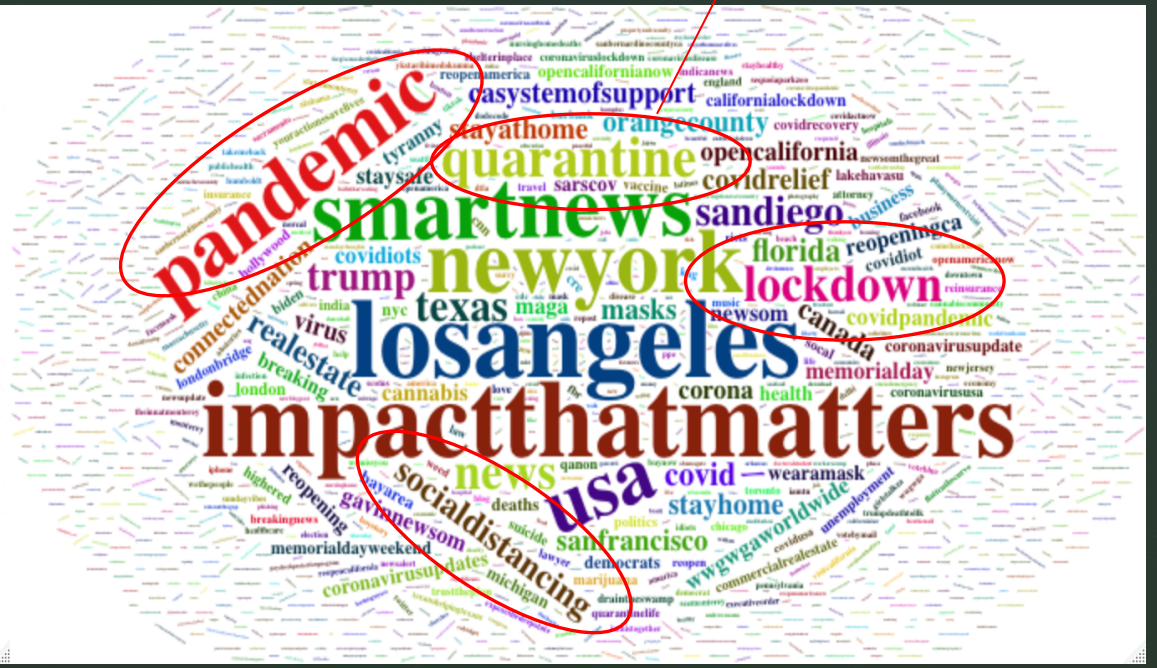

```
data <- read.csv('PycharmProjects/untitled17/PopularBefore25result.csv')

text <- data$hashtags

docs <- Corpus(VectorSource(text))

docs <- docs %>%tm_map(removeNumbers) %>% tm_map(removePunctuation) %>%tm_map(stripWhitesp

docs <- tm_map(docs, content_transformer(tolower))

docs <- tm_map(docs, removeWords, stopwords("english"))

dtm <- TermDocumentMatrix(docs)

matrix <- as.matrix(dtm)

words <- sort(rowSums(matrix),decreasing=TRUE)

df <- data.frame(word = names(words),freq=words)

set.seed(1234)

#wordcloud(words = df$word, freq = df$freq, min.freq = 1,max.words=200, random.order=FALSE,
rot.per=0.35,colors=brewer.pal(8, "Dark2"))

write.csv(df,'PopularBefore25resultHashtags.csv')

#dftest <- read.csv('PycharmProjects/untitled17/TimeKeyWords.csv')

dftest <- read.csv('TopPopularAfter25resultHashtags.csv')

wordcloud2(data=dftest, size=0.6, color='random-dark')
```

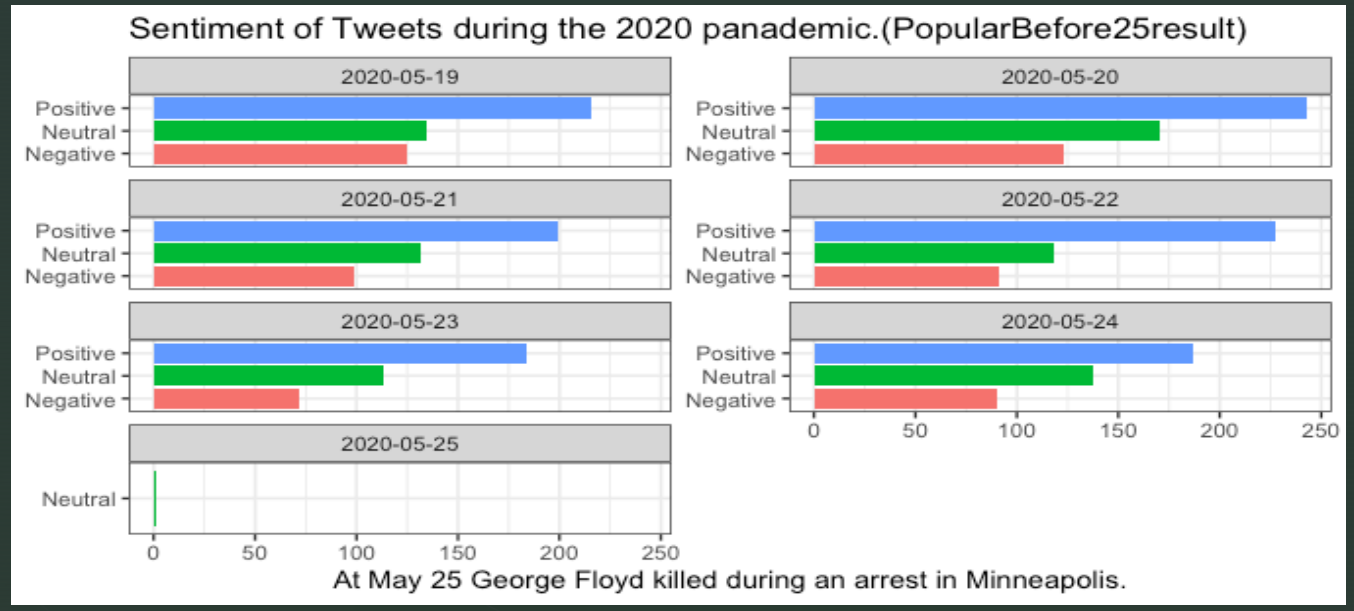


wordcount

#HashTags

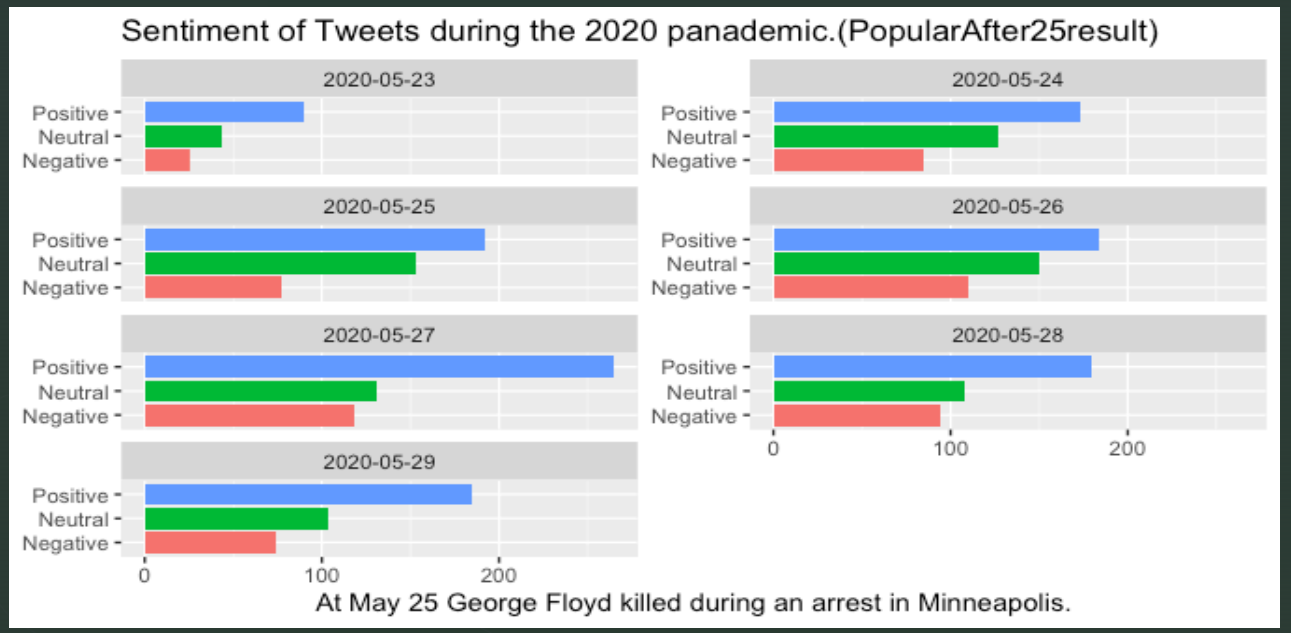






wordcount

#HashTags



wordcount

#HashTags

TextBlob Vs Vader

- Lets try vader sentiment analyzer.

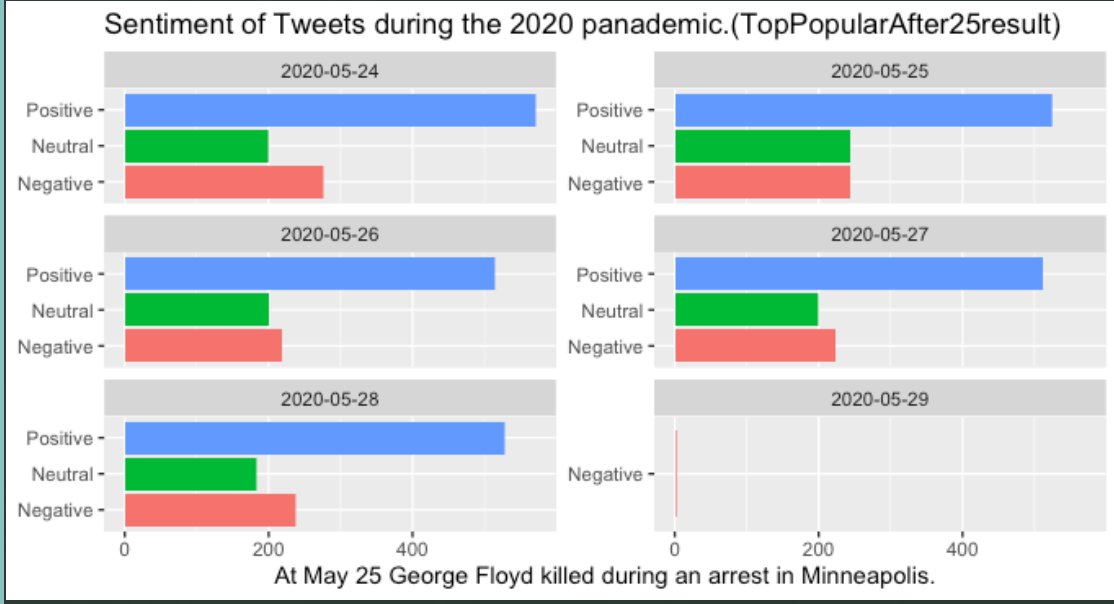
Code: in R.

```
def apply_vader(sentence):
    vs = analyzer.polarity_scores(sentence)
    #temp = TextBlob(sentence).sentiment[0]
    vs["compound"]
    if vs["compound"] <=  -0.05:
        return "Negative" # Negative
    elif vs["compound"] >= 0.05:
        return "Positive" # Positive
    else:
        return "Neutral"# Neutral
```

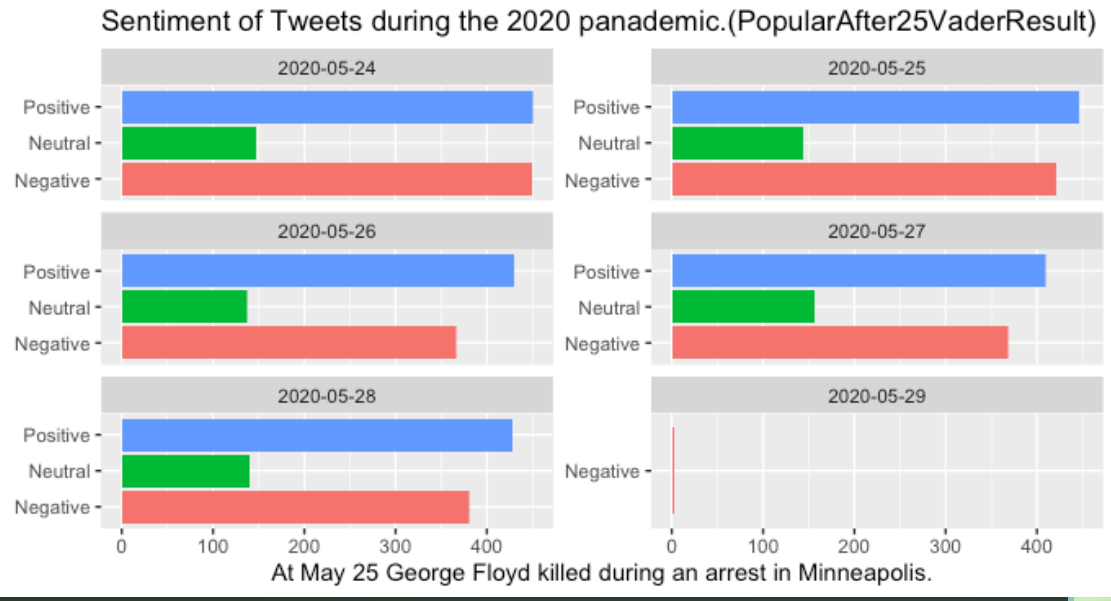
```
def apply_blob(sentence):
    temp = TextBlob(sentence).sentiment[0]
    print("TExtBlob:",temp)
    if temp == 0.0:
        return "Neutral" # Neutral
    elif temp >= 0.0:
        return "Positive" # Positive
    else:
        return "Negative"# Negative
```

Switch the function

TextBlob



Vader



wordcount

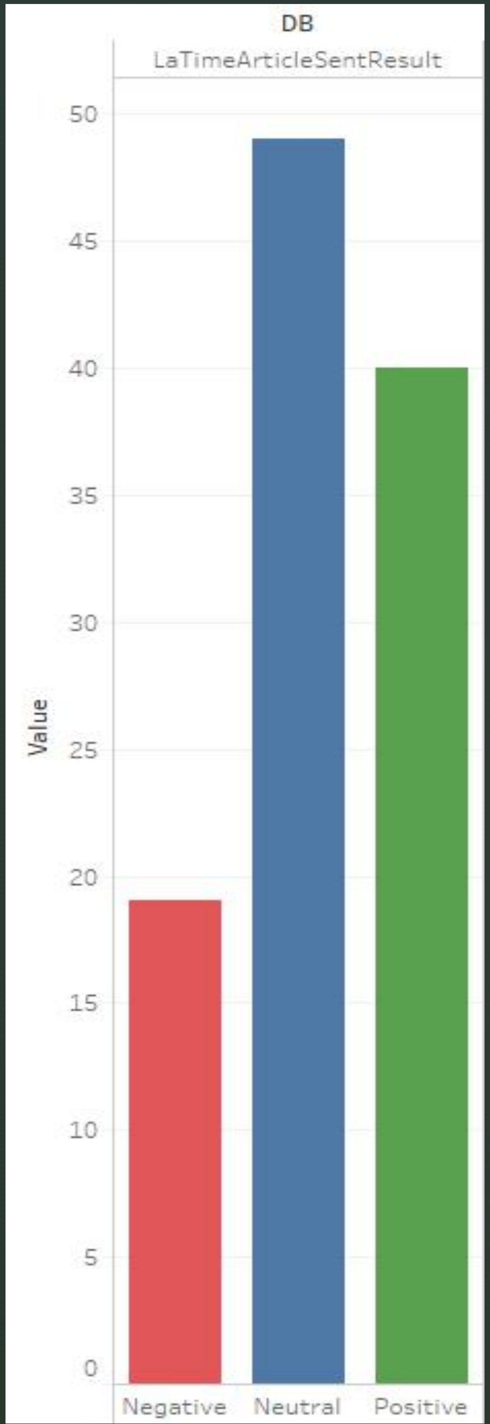
#HashTags

Code: in Python

HashTags by number of likes

```
all_data = pd.read_csv("TopPopularAfter25.csv", converters={"hashtags": literal_eval})
likes_count = all_data["likes_count"].tolist()
# print("Type:",type(likes_count))
# print(likes_count)
hashTags_data = pd.read_csv("TopPopularAfter25resultHashtags.csv")

hashTags = hashTags_data['word'].tolist()

temp_dict = {}

for x,value in all_data['hashtags'].items():
    #print("Type:",value)
    for word in value:
        print(type(word))
        new_word = word.strip(" #")
        print("Word:",new_word)
        if new_word in hashTags:
            if new_word in temp_dict:
                temp_dict[new_word] += likes_count[x]
            else:
                temp_dict[new_word] = likes_count[x]


likes_count_hashtag = []
for hashtag in hashTags:
    if hashtag in temp_dict:
        likes_count_hashtag.append(temp_dict[hashtag])
    else:
        likes_count_hashtag.append(1)


hashTags_data['likes_count'] = likes_count_hashtag

hashTags_data.to_csv("TopPopularAfter25resultHashtags2.csv")
```

LaTime Sentiment:



LaTimes KeyWords:

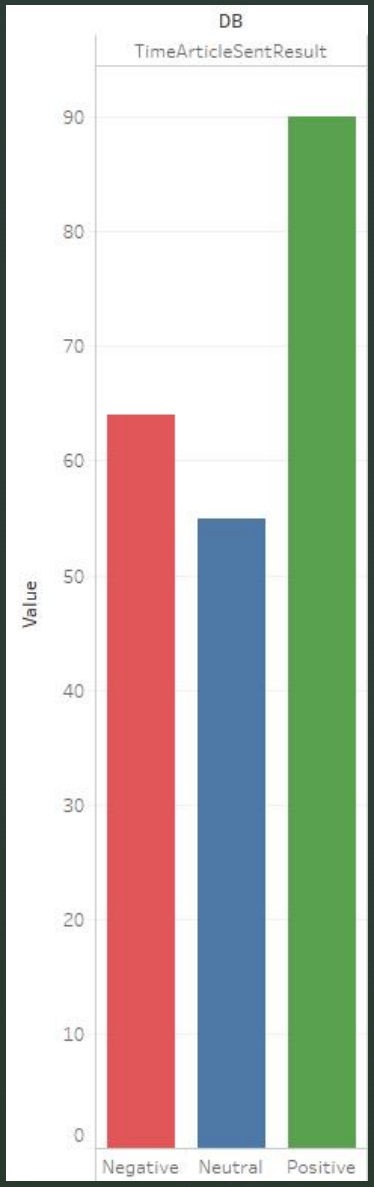
Find keywords in articles

Code: in Python

```
client = MongoClient("mongodb+srv://Test1:Test1@cluster0-pwaip.mongodb.net/test?retryWrites=true&w=majority")
db = client.USArticles
num_of_articles = 0
total_articles = []
keywords_list = []
keywords_dict = {}
all_text = ""
news_outlet = "Time"
with client:
    #db = client.Categories
    articles = db.articles
    for x in articles.find({"news_outlet": news_outlet}, {'text': 1,'keywords':1}):
        num_of_articles +=1
        #print(x)
        total_articles.append(x['text'])
        keywords = x['keywords']
        for word in keywords:
            all_text += word + " "
            keywords_list.append(word)
            if word in keywords_dict:
                keywords_dict[word] += 1
            else:
                keywords_dict[word] = 1
file_name = news_outlet+"KeyWords.csv"
with open(file_name, 'w') as f:
    for key in keywords_dict.keys():
        f.write("%s,%s\n"%(key,keywords_dict[key]))
```

TIME

Time Sentiment:



Time KeyWords:

What's next?

- **Data Correlation:** Is a way to understand the relationship between multiple variables and attributes in out dataset. Using Correlation, we can get some insights such as:
- One or multiple attributes depend on another attribute or a cause for another attribute.
- One or multiple attributes are associated with other attributes.

Spearman's rank correlation coefficient

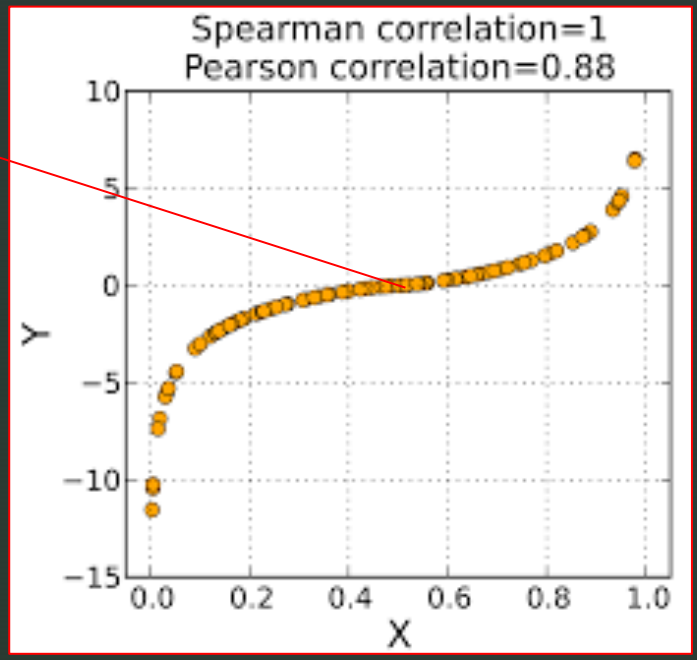
```
df['x'] = [39, 16, 20,
           31, 15, 25,
           16, 17, 22,
           24, 10, 21,
           20, 16, 25,
           ]
df['y'] = [448,155,452,
           425, 151, 392,
           427, 122, 390,
           402, 162, 382,
           420, 145, 393,
           ]
```

Date/Sent	Positive	Netural	Negative
24/5	39	16	20
25/5	31	15	25
26/5	16	17	22
27/5	24	10	21
28/5	20	16	25

Date/Sent	Positive	Netural	Negative
24/5	448	155	452
25/5	425	151	392
26/5	427	122	390
27/5	402	162	382
28/5	420	145	393

Result:

0.54



Python Script:

```
Spearman's correlation
spearmans_rank_correlation is:  0.5475842440973836
Scipy spearmans_rank_correlation is:  0.5475842440973835
```

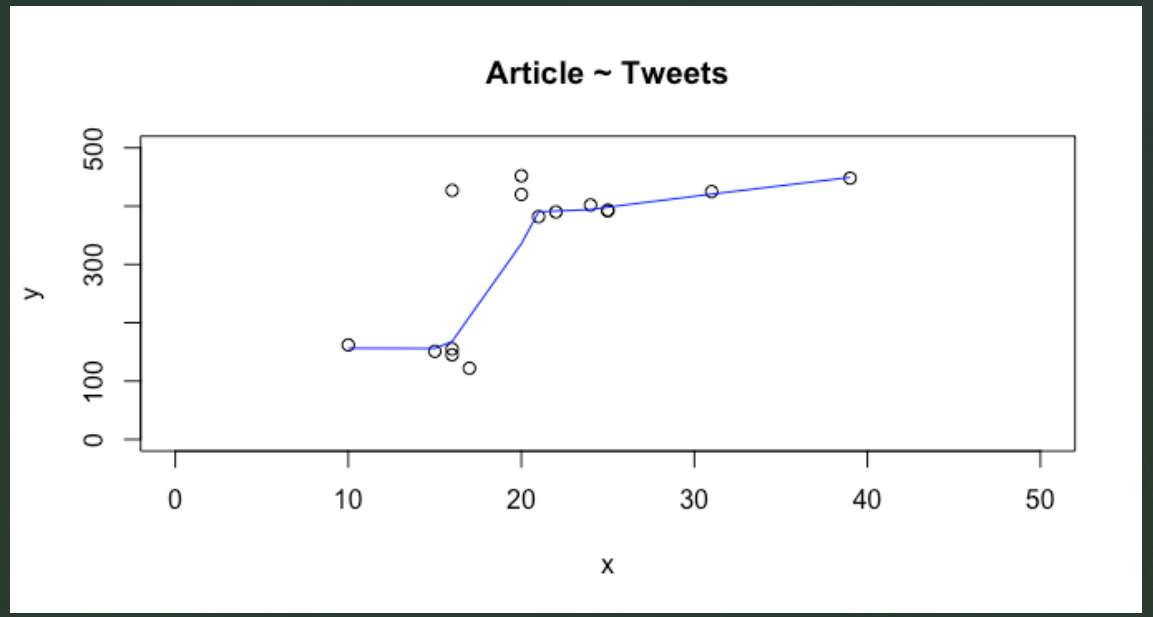
R script:

```
x <- c(39, 16, 20,
       31, 15, 25,
       16, 17, 22,
       24, 10, 21,
       20, 16, 25)
y <- c(448,155,452,
       425, 151, 392,
       427, 122, 390,
       402, 162, 382,
       420, 145, 393)
cor(x, y, method = "spearman")
```

17:1 (Top Level)

Console Terminal Jobs

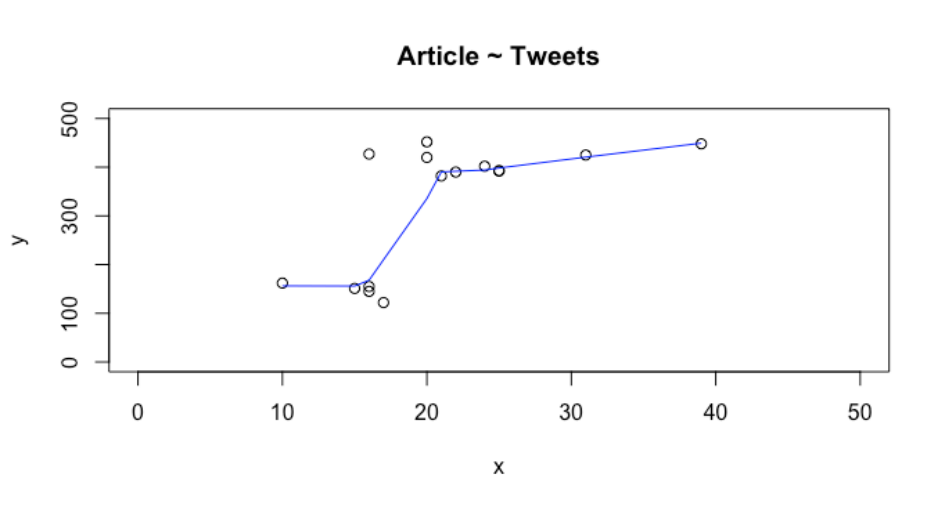
```
> x <- c(39, 16, 20,
+       31, 15, 25,
+       16, 17, 22,
+       24, 10, 21,
+       20, 16, 25)
> y <- c(448,155,452,
+       425, 151, 392,
+       427, 122, 390,
+       402, 162, 382,
+       420, 145, 393)
> cor(x, y, method = "spearman")
[1] 0.5475842
```



Code:

R

```
x <- c(39, 16, 20,
    31, 15, 25,
    16, 17, 22,
    24, 10, 21,
    20, 16, 25)
y <- c(448,155,452,
    425, 151, 392,
    427, 122, 390,
    402, 162, 382,
    420, 145, 393)
cor(x, y, method = "spearman")

plot(x = x, y = y, type = "p", main = "Article ~ Tweets",
    xlim = c(0, 50), ylim = c(0,500))
lines(lowess(x = x, y = y), col = "blue")
```

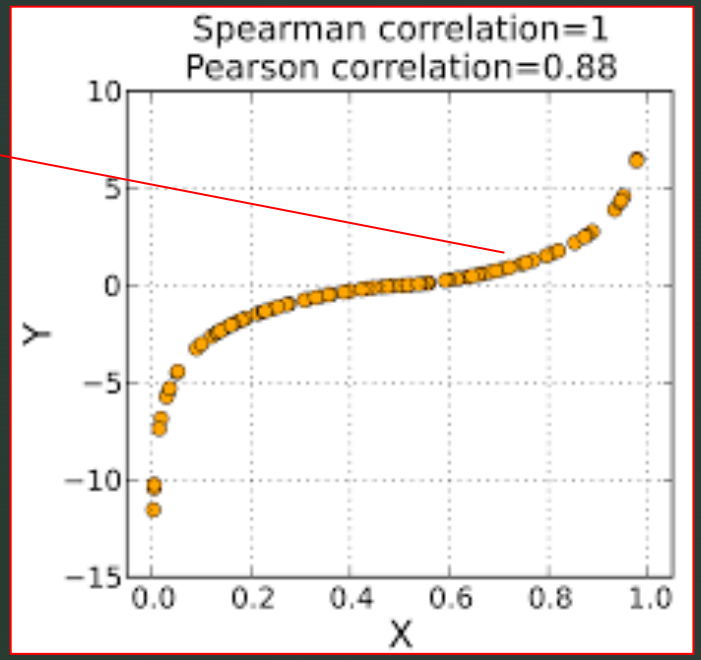


Python

```
def Spearman():
    # Create empty dataframe
    df = pd.DataFrame()
    # Add columns
    #df['x'] = random.sample(range(1, 100), 75)
    df['x'] = [39, 16, 20,
               31, 15, 25,
               16, 17, 22,
               24, 10, 21,
               20, 16, 25,
               ]
    df['y'] = [448,155,452,
               425, 151, 392,
               427, 122, 390,
               402, 162, 382,
               420, 145, 393,
               ]
    def spearmans_rank_correlation(xs, ys):
        # Calculate the rank of x's
        xranks = pd.Series(xs).rank()
        # Caclulate the ranking of the y's
        yranks = pd.Series(ys).rank()
        # Calculate Pearson's correlation coefficient on the ranked versions of the data
        return scipy.stats.pearsonr(xranks, yranks)
    # Show Pearson's Correlation Coefficient
    result = spearmans_rank_correlation(df.x, df.y)[0]
    print("spearmans_rank_correlation is: ", result)
    # Calculate Spearman's Correlation Using SciPy
    print("Scipy spearmans_rank_correlation is: ", scipy.stats.spearmanr(df.x, df.y)[0])
    # reg plot
    # sns.lmplot('x', 'y', data=df, fit_reg=True)
    # plt.show()
Spearman ()
```

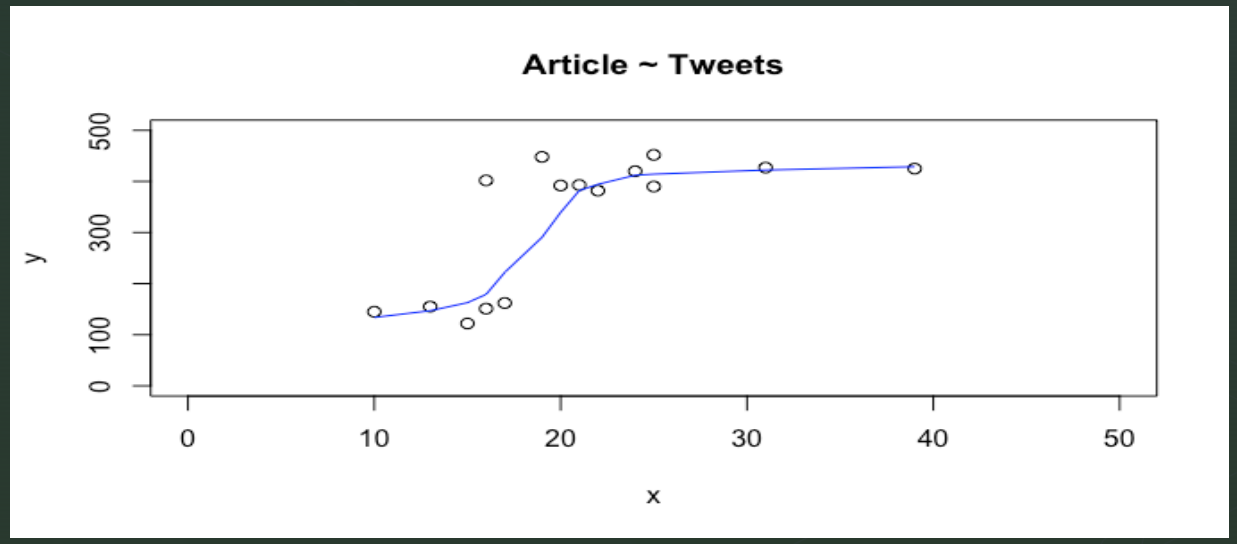
Result One day leg:

0.72



R script:

```
> x <- c(19, 13, 25,
+        39, 16, 20,
+        31, 15, 25,
+        16, 17, 22,
+        24, 10, 21
+        )
> y <- c(448,155,452,
+        425, 151, 392,
+        427, 122, 390,
+        402, 162, 382,
+        420, 145, 393)
> cor(x, y, method = "spearman")
[1] 0.7245092
```



Future development

Try to find correlation via KeyWords.

Scrape more data!!

Add more features to find correlation

The End

Liad Nehemia,2020