Comparative Analysis between Bursty Twitter Events and Real-Time Trends

Amir Malek (am6370)
DSGA 1015 - Text as Data Final Paper

Word Count: 1775

Introduction

In the Summer of 2008, the Trends feature was introduced by Twitter, further transforming the microblogging service into a realtime reflection of current events. This feature also triggered countless questions and criticisms regarding the mechanics of the real-time trending list (which is sorted by popularity), which was designed to highlight current, popular topics based on the volume and burstiness of relevant tweets. As Twitter themselves state,

> *"Trends are determined by an algorithm and, by default, are tailored for you based on who you follow, your interests, and your location. This algorithm identifies topics that are popular now, rather than topics that have been popular for a while or on a daily basis, to help you discover the hottest emerging topics of discussion on Twitter."* [1]

Twitter's description of the trends algorithm, although slightly vague, implies that events or groups of tweets that emerge quickly are more likely to be labeled as a trend than a topic that has been popular over a longer period of time. To verify this claim, this paper will explore these twitter trends and compare them to their statistical metrics such as burstiness, duration, and volume. In other words, we will examine whether Twitter's claimed trending topics are quantifiably justified.

Literature

Due to Twitter's heavy influence on social media, the literature surrounding trends and bursty event detection is quite rich. Some notable studies examine burst

[1] Twitter Trends FAQ. https://help.twitter.com/en/using-twitter/twitter-trending-faqs

detection in random samples of tweets. For example, Pau Perng-Hwa Kung describes in "Detecting and Analyzing Bursty Events on Twitter" [2] a novel system for detecting and characterizing bursts of tweets generated by multiple sources, called BurstMapper. Similarly, a group from the University of Tsukuba in Japan analyzed and categorized five types of bursts and how the number of retweets and replies change during periods of bursty activity.[3]

These studies were an important reference in the design of our experiment, especially in terms of data collection and burst detection. For data collection, it became clear that in order to collect a random sample of twitter data, an appropriate approach would be to use a data crawler which systematically collects tweets that haven't been previously filtered by category or hashtag. For burst detection, relevant literature validated using Kleinberg's algorithm, which is implemented in the 'bursts' package in R.

Theory and Hypotheses

As outline by Twitter, we can expect to see a correlation between the burstiness of the data, the duration of bursts, and the overall trend ranking. As burstiness increases, the trends should move higher in rank. Also, we can expect to see topics with a lower average burst duration to also be higher in rank. This correlation would indicate that Twitter's algorithm looks not only for high volume when deciding on trends, but prioritizes those high volume topics based on their

[2] Pau Perng-Hwa, Kung (2016). "Detecting and Analyzing Bursty Events on Twitter"

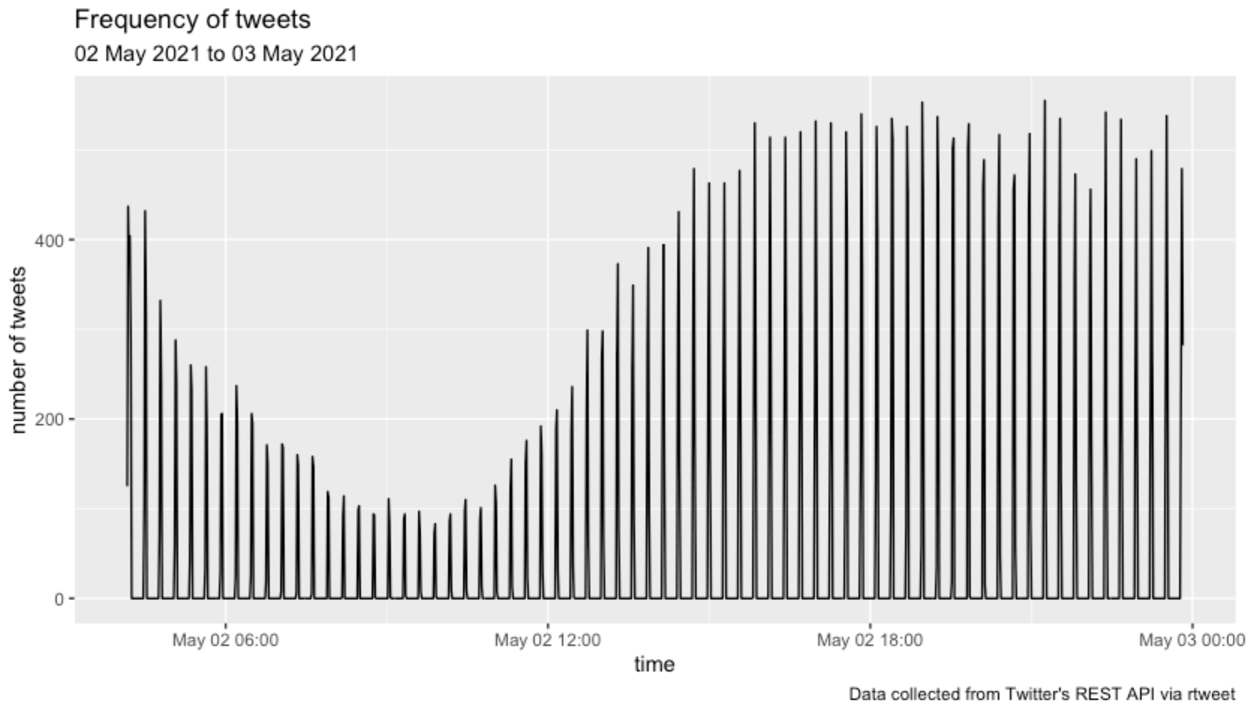
[3] Mizunuma, Yuhiro; Yamamoto, Shuhei; Yamaguchi, Yutaro; Ikeuchi, Atsushi; Satoh, Tetsuji (2014). "Twitter Bursts: Analysis of their Occurrences and Classifications" Tsukuba, Japan

burstiness. In other words, a lower-level bursty topic over five hours would be lower in ranking than a higher-level bursty topic over 30 minutes. Other relationships we can examine would be trend ranking against minimum and maximum duration, as well as the number of mentions of the topic across all tweets in the recent time period.

Data and Methods

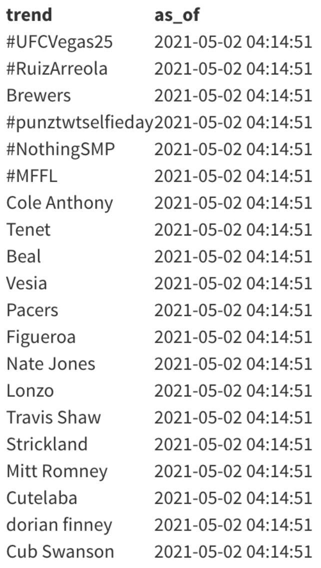
The data for this project was collected with rtweet, an R package that uses Twitter's REST and stream API to collect and organize tweets. The goal of the data collection process was to collect tweets over several hours such that an analysis could be performed over several cycles and updates of Twitter's trend list. As mentioned previously, trends are updated in real time, so it was important to collect data regularly.

Using rtweet, we designed a data crawler which streams live tweets for 2 minutes, then sleep for 15 minutes before running the next iteration. Although the rtweet package does support random tweet selection, the function prioritized recent tweets without the ability to specify a time range with a uniform distribution. Thus, the best option was to simulate a random sampling of data by using the stream_tweets function, which collects a live stream of Twitter data when called. With the stream duration and sleep duration parameters set to 2 minutes and 15 minutes, respectively, we were able to collect on average 1000 tweets every 15 minutes. The data crawler was then set to run for around 24 hours, resulting in a randomly sampled dataset of around 50,000 tweets.


Frequency of tweets
02 May 2021 to 03 May 2021

Data collected from Twitter's REST API via rtweet

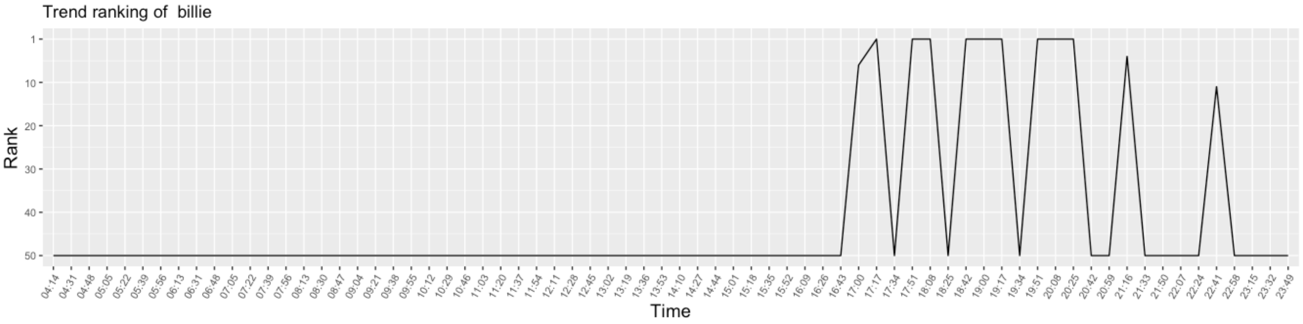
For each iteration of the tweet stream, a snapshot of the top 50 current trends list was also saved with an associated timestamp. This list would allow for a direct mapping of topic burst levels and durations to the topic's ranking in the trends list. For both the trending list and the tweet stream, the location was set to "united states" to avoid issues with differences in language and trend rankings.



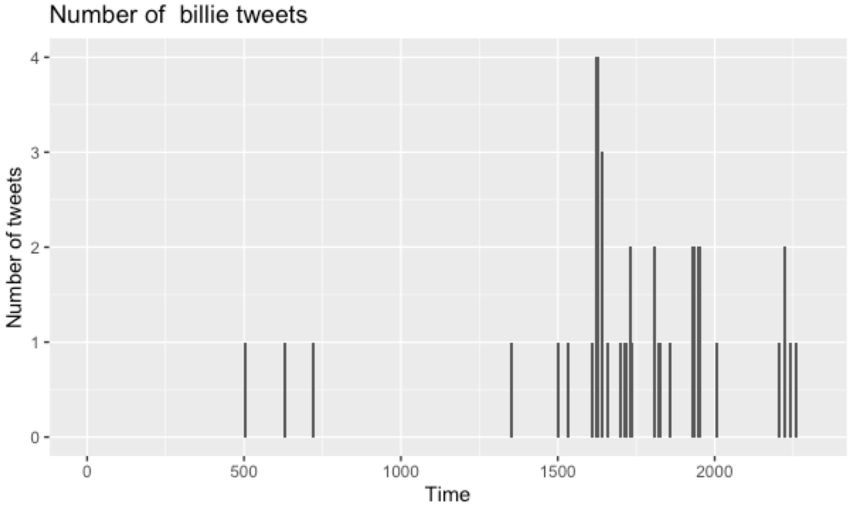
trend	as_of
#UFCVegas25	2021-05-02 04:14:51
#RuizArreola	2021-05-02 04:14:51
Brewers	2021-05-02 04:14:51
#punztwtselfieday	2021-05-02 04:14:51
#NothingSMP	2021-05-02 04:14:51
#MFFL	2021-05-02 04:14:51
Cole Anthony	2021-05-02 04:14:51
Tenet	2021-05-02 04:14:51
Beal	2021-05-02 04:14:51
Vesia	2021-05-02 04:14:51
Pacers	2021-05-02 04:14:51
Figueroa	2021-05-02 04:14:51
Nate Jones	2021-05-02 04:14:51
Lonzo	2021-05-02 04:14:51
Travis Shaw	2021-05-02 04:14:51
Strickland	2021-05-02 04:14:51
Mitt Romney	2021-05-02 04:14:51
Cutelaba	2021-05-02 04:14:51
dorian finney	2021-05-02 04:14:51
Cub Swanson	2021-05-02 04:14:51

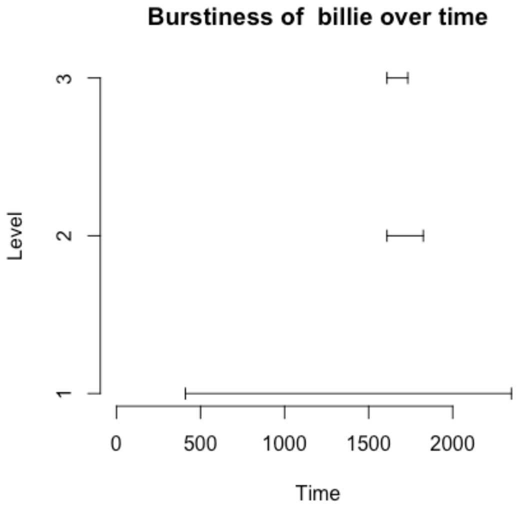
Table 1: List of trends in descending order

Several preprocessing steps were also performed, including string matching and replacement to consolidate different date and time values, text conversion to lower case, and removal of stopwords and links. Next, the text data was tokenized using n-grams where n = (1,2,3), so that trends of more than one word could be filtered in the dataframe.



Using the preprocessed data, a dfm was constructed so that word frequency and burstiness could be computed. We can also perform various visualizations on individual trends in order to gain some insight as to the driving factors behind them. For example, a time series plot of a single trend allows us to visualize a topic's emergence and persistence in the rankings. The burstiness plot can also be used to observe the topic's level at different times.





For example, we can observe that for the trend "billie", there was a constant stream of tweets between 4AM and 12AM, with a level 3 surge of activity between around 3PM and 6PM. This corresponds closely with the ranking of the "billie" trend, which emerged at around 4:43PM, and persisted for several hours thereafter. We can assume that with adequate time and resources allocated for streaming tweets, one could stream continuously to achieve an even closer resemblance between these two plots. For this project, the streaming parameters were determined so that the dataset could be manageable in terms of volume and complexity.

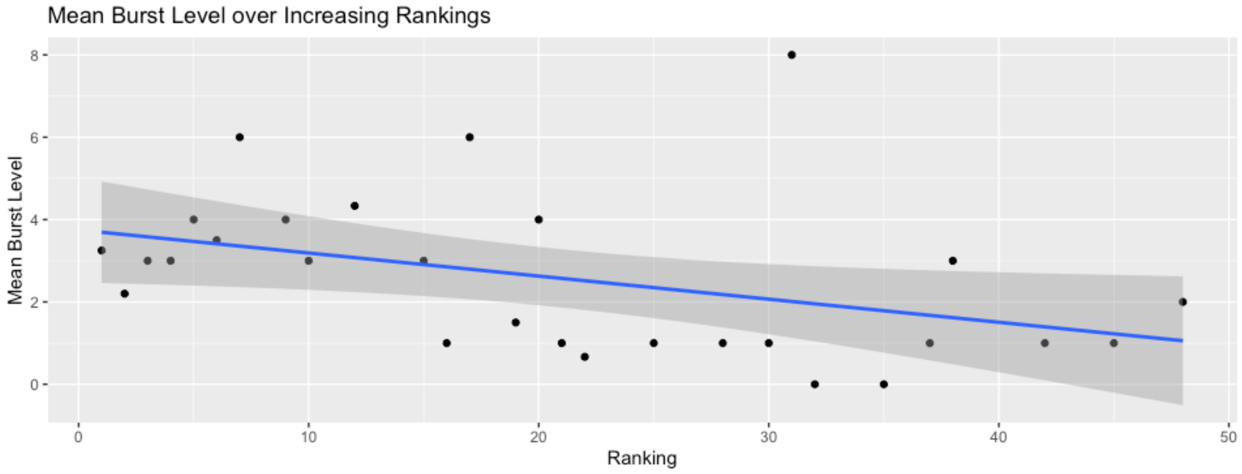
In order to discover latent structure or correlation among the data, we performed several data manipulations to consolidate all of the data and metrics in one dataframe. First, we iterate through all instances of the trends list from various times of day, appending each instance as one column. We can then compute the mean, minimum, and best ranking for each trend in the dataframe (called all_trends_ranked). Next, we iterate through each row of this dataframe, and compute the mean, minimum and maximum of the burstiness level and duration for each trend, and also append the number of mentions of the trend, computed by filtering from the top features of the dfm.

trend	mean_ranking	best_ranking	max_burst_level	max_burst_duration	min_burst_duration	avg_burst_duration	number_of_mentions
#ufcvegas25	2.92	1	6	1,848	2	339.0000	68
old_trafford	7.53846153846154	1	4	1,233	0	290.2000	31
billie	8.07692307692308	1	3	1,939	124	759.6667	34
taemin	13.6818181818182	1	3	908	58	388.6667	6
taemin	1	1	3	908	58	388.6667	6
billie	2.75	1	3	1,939	124	759.6667	34
messi	9.66666666666667	1	2	1,905	365	1,135.0000	8
#sundayvibes	4.05405405405405	1	2	1,922	672	1,297.0000	40
kyrie	2	2	4	1,921	74	621.0000	16
bucks	3.22222222222222	2	3	1,796	57	684.3333	18
big_sean	8.42307692307692	2	2	1,813	348	1,080.5000	9
raisin_bran	13.8823529411765	2	1	1,939	1,939	1,939.0000	18
susan_collins	15.3243243243243	2	1	1,830	1,830	1,830.0000	6

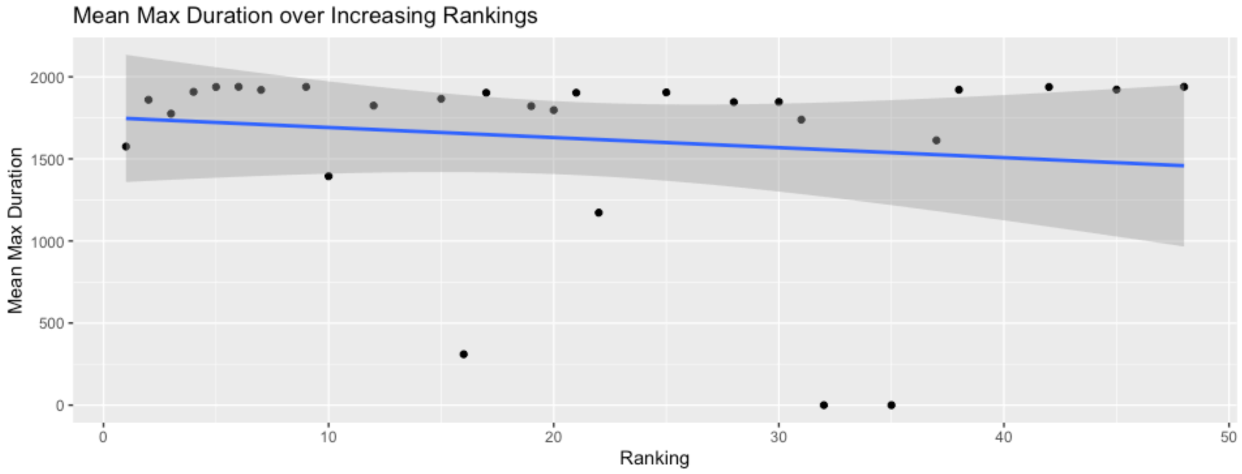
Table 2: First few rows of the all_trends_ranked dataframe

Results

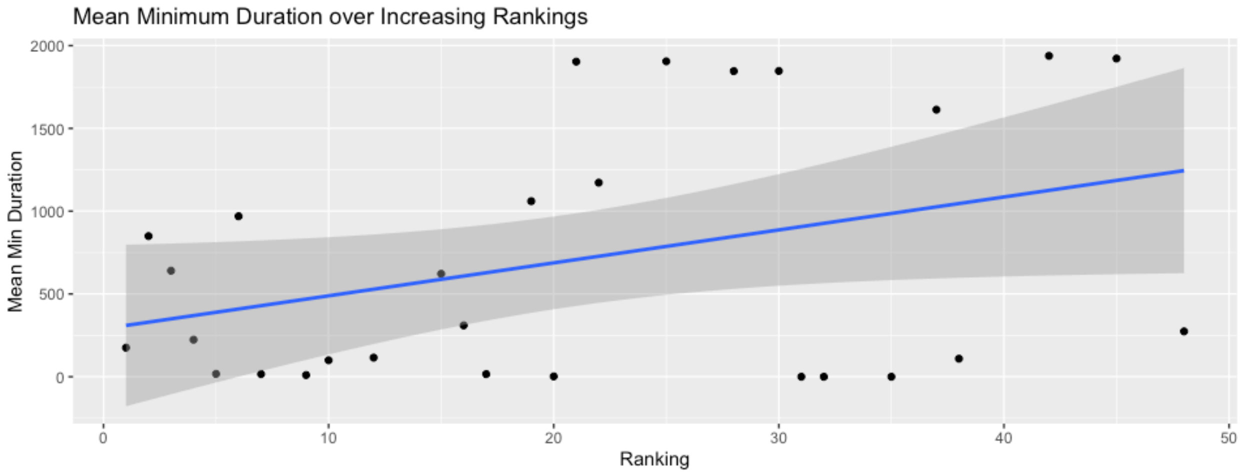
Using the dataframe shown previously, we can plot various relationships that provide some insight into how Twitter trends are determined.



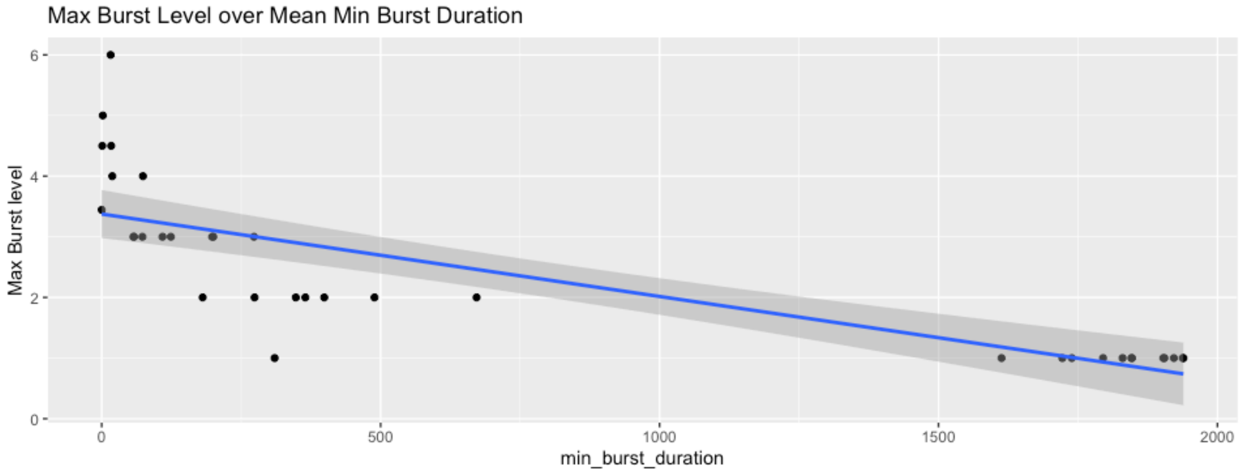
This first plot shows the mean burst level over increasing rankings (where 1 is the best ranking, and 50 is the lowest). As shown by the regression line, there is a negative correlation between burst level and increasing rankings. Such a result is expected since higher burst levels indicate more popular, emerging topics.



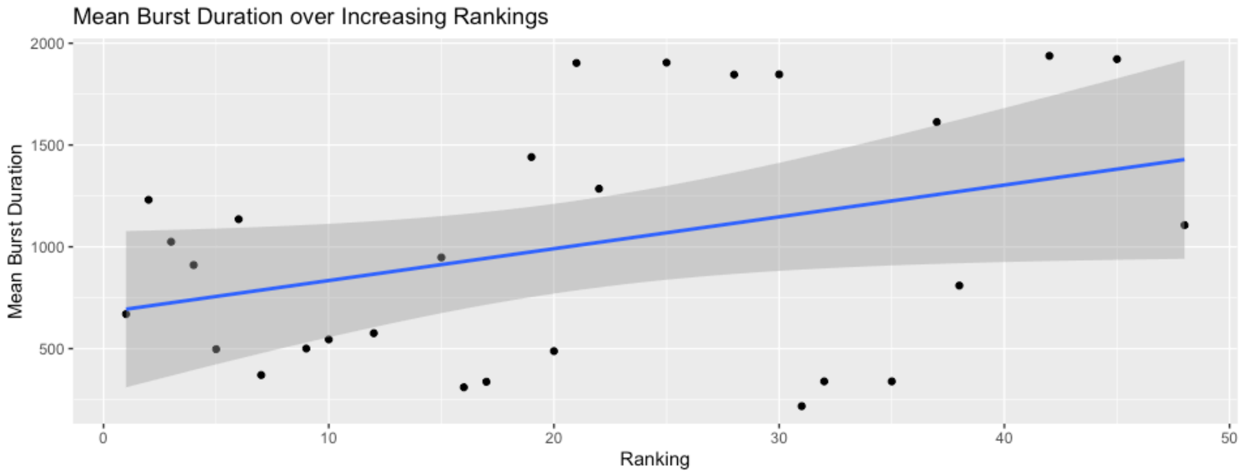
The next relationship measures mean maximum burst duration over increasing rankings. Although a slight negative correlation is present, one could argue that the maximum duration of all bursts of a topic has little influence on the ranking of the topic in the trends list.



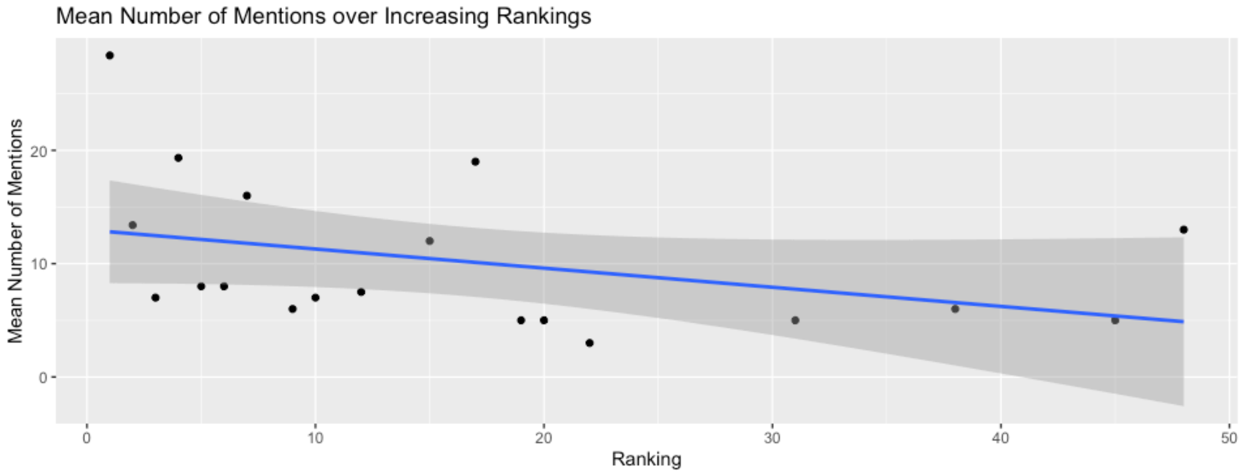
Unlike the mean maximum duration plot, the mean minimum duration does seem to demonstrate a positive correlation. This is also expected since bursts of shorter time periods would, again, be recognized as a popular emerging topic. However, it is also important to note that there is likely a correlation between burst level and mean minimum duration, since by nature of Kleinberg’s algorithm, shorter periods of activity are classified as more bursty. The following plot displays this correlation:



Nevertheless, if we were to expand this project into a supervised learning regression problem, we could use the duration of the bursts as a feature to predict 'trendiness'. The plot below shows the mean burst duration over increasing rankings, for which there is a clear positive correlation:



Finally, another feature that could be examined in the classification of trending topics is the number of mentions of the topic or its derivatives.



We can observe that the number of mentions of a topic decreases as we travel to the lower rankings of the trends list.

Discussion

Based on the results, we can claim that the burst level and average burst duration are viable features for determining a topic’s rank on the Twitter trends list. For this study, there are many areas of improvement and further research, most notably examining outliers in the data and exploring more features. For example, we could identify several trends that were higher in burst level, but lower in ranking than other trends. This behavior may suggest that there some threshold that Twitter’s algorithm follows to decide on the topics, or perhaps it looks at other features like hashtags and number of retweets (which were not explored in this paper). Another aspect that could be further researched is how various derivatives of topics affect their trendiness. According to Twitter,

> *“Algorithmically, Trends and hashtags are grouped together if they are related to the same topic.”* [4]

Furthermore, according to a 2018 study, more than 68% of trends were hashtags,[5] so it would be important to understand whether these kinds of features are more likely to detected by Twitter, or if they are simply more viral by nature.

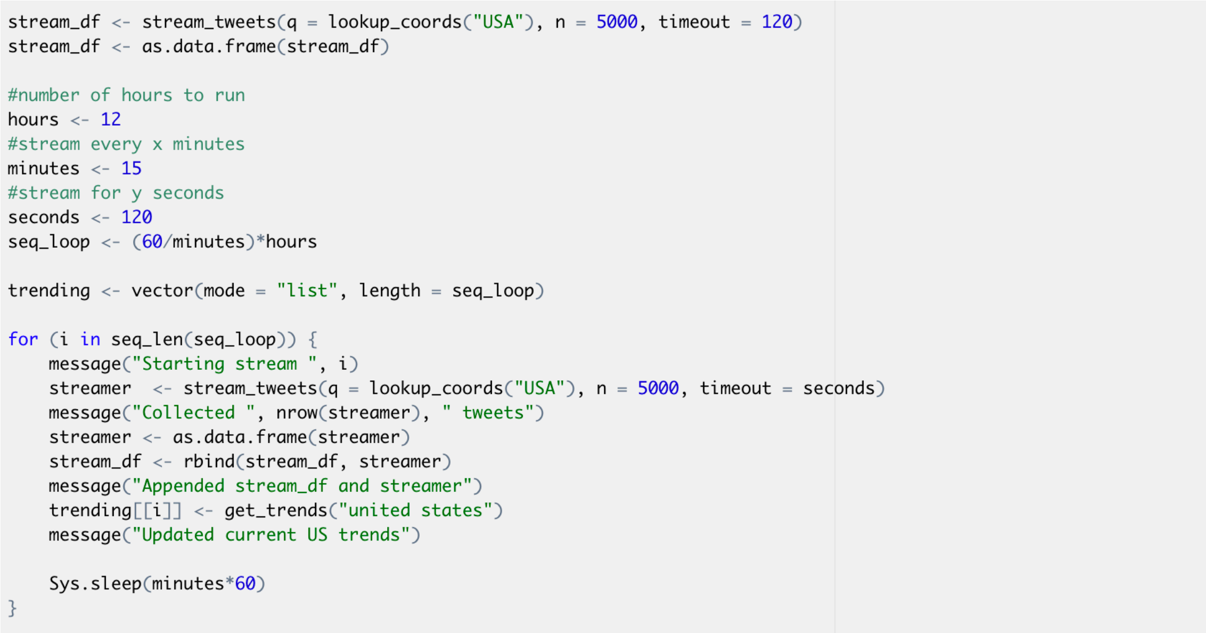
Finally, one common accusation of Twitter trends is that they are curated, or sold for promotional purposes. This, too, could be explored by looking at outliers in the data and seeking the context (or lack thereof) behind the rise and fall of a trend.

[4] Twitter Trends FAQ. https://help.twitter.com/en/using-twitter/twitter-trending-faqs

[5] Annamoradnejad, Issa; Habibi, Jafar (2019). "A Comprehensive Analysis of Twitter Trending Topics". *2019 5th International Conference on Web Research (ICWR)*. Tehran, Iran

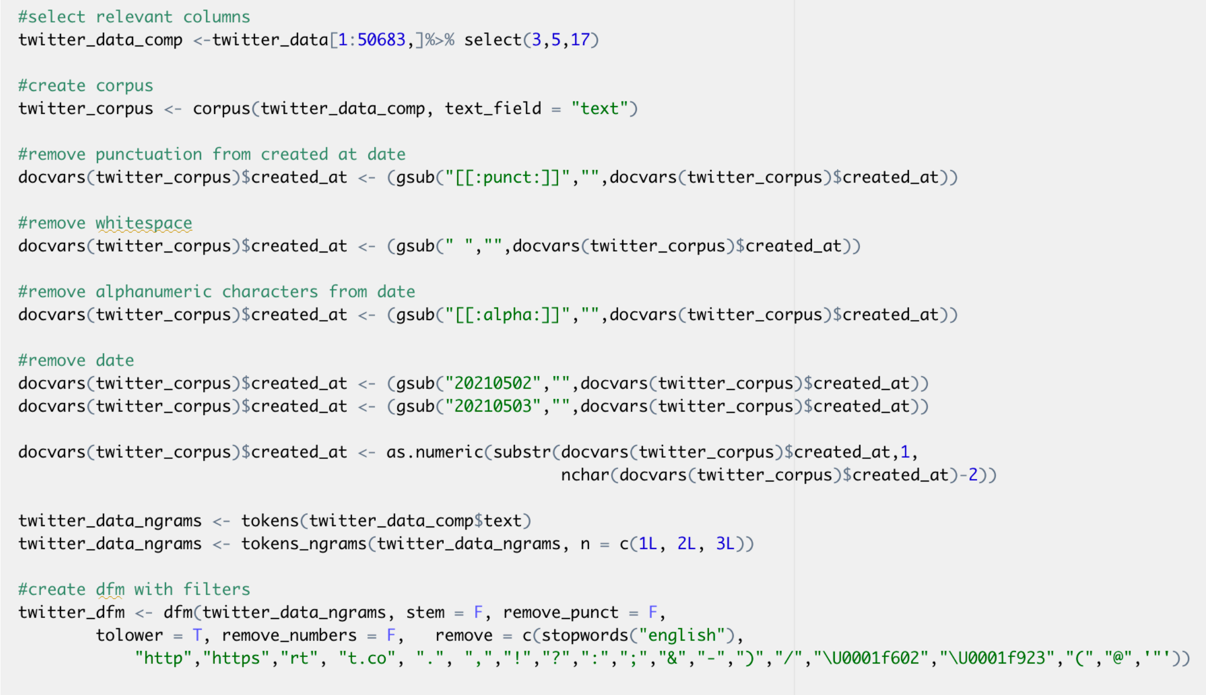
Appendix

A. Twitter Streaming Function



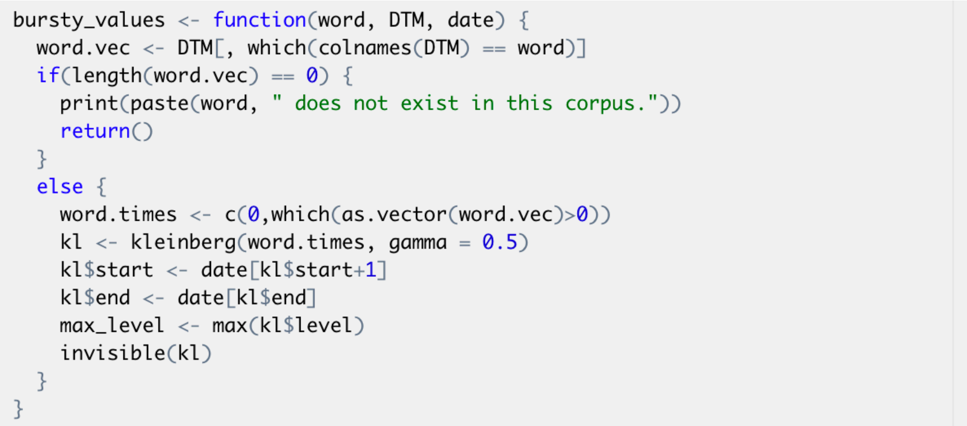
```r
stream_df <- stream_tweets(q = lookup_coords("USA"), n = 5000, timeout = 120)
stream_df <- as.data.frame(stream_df)

#number of hours to run
hours <- 12
#stream every x minutes
minutes <- 15
#stream for y seconds
seconds <- 120
seq_loop <- (60/minutes)*hours

trending <- vector(mode = "list", length = seq_loop)

for (i in seq_len(seq_loop)) {
    message("Starting stream ", i)
    streamer  <- stream_tweets(q = lookup_coords("USA"), n = 5000, timeout = seconds)
    message("Collected ", nrow(streamer), " tweets")
    streamer <- as.data.frame(streamer)
    stream_df <- rbind(stream_df, streamer)
    message("Appended stream_df and streamer")
    trending[[i]] <- get_trends("united states")
    message("Updated current US trends")

    Sys.sleep(minutes*60)
}
```

B. Data Preprocessing and DFM



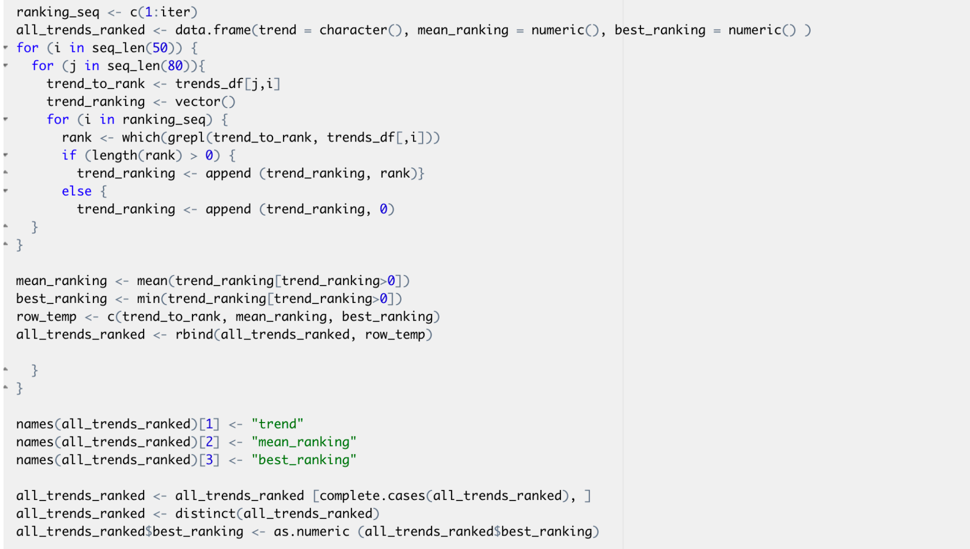
```r
#select relevant columns
twitter_data_comp <-twitter_data[1:50683,]%>% select(3,5,17)

#create corpus
twitter_corpus <- corpus(twitter_data_comp, text_field = "text")

#remove punctuation from created at date
docvars(twitter_corpus)$created_at <- (gsub("[[:punct:]]","",docvars(twitter_corpus)$created_at))

#remove whitespace
docvars(twitter_corpus)$created_at <- (gsub(" ","",docvars(twitter_corpus)$created_at))

#remove alphanumeric characters from date
docvars(twitter_corpus)$created_at <- (gsub("[[:alpha:]]","",docvars(twitter_corpus)$created_at))

#remove date
docvars(twitter_corpus)$created_at <- (gsub("20210502","",docvars(twitter_corpus)$created_at))
docvars(twitter_corpus)$created_at <- (gsub("20210503","",docvars(twitter_corpus)$created_at))

docvars(twitter_corpus)$created_at <- as.numeric(substr(docvars(twitter_corpus)$created_at,1,
                                                        nchar(docvars(twitter_corpus)$created_at)-2))

twitter_data_ngrams <- tokens(twitter_data_comp$text)
twitter_data_ngrams <- tokens_ngrams(twitter_data_ngrams, n = c(1L, 2L, 3L))

#create dfm with filters
twitter_dfm <- dfm(twitter_data_ngrams, stem = F, remove_punct = F,
        tolower = T, remove_numbers = F,    remove = c(stopwords("english"),
            "http","https","rt", "t.co", ".", ",","!","?",":",";","&","-",")","/","\U0001f602","\U0001f923","(","@",'""'))
```

C. Burst Detection Function

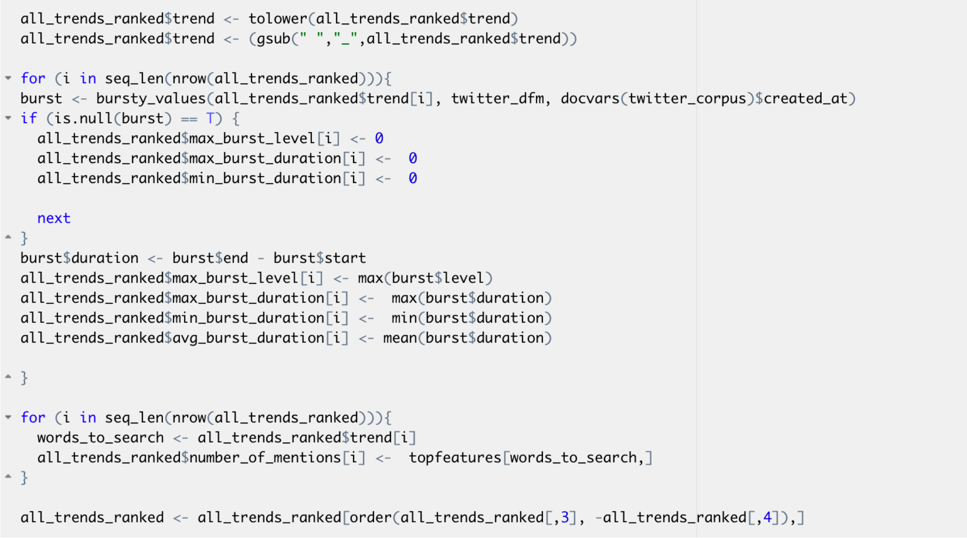


```r
bursty_values <- function(word, DTM, date) {
  word.vec <- DTM[, which(colnames(DTM) == word)]
  if(length(word.vec) == 0) {
    print(paste(word, " does not exist in this corpus."))
    return()
  }
  else {
    word.times <- c(0,which(as.vector(word.vec)>0))
    kl <- kleinberg(word.times, gamma = 0.5)
    kl$start <- date[kl$start+1]
    kl$end <- date[kl$end]
    max_level <- max(kl$level)
    invisible(kl)
  }
}
```

D. All_Trends_Ranked Dataframe



```r
ranking_seq <- c(1:iter)
all_trends_ranked <- data.frame(trend = character(), mean_ranking = numeric(), best_ranking = numeric() )
for (i in seq_len(50)) {
  for (j in seq_len(80)){
    trend_to_rank <- trends_df[j,i]
    trend_ranking <- vector()
    for (i in ranking_seq) {
      rank <- which(grepl(trend_to_rank, trends_df[,i]))
      if (length(rank) > 0) {
        trend_ranking <- append (trend_ranking, rank)}
      else {
        trend_ranking <- append (trend_ranking, 0)
  }
}

mean_ranking <- mean(trend_ranking[trend_ranking>0])
best_ranking <- min(trend_ranking[trend_ranking>0])
row_temp <- c(trend_to_rank, mean_ranking, best_ranking)
all_trends_ranked <- rbind(all_trends_ranked, row_temp)

  }
}

names(all_trends_ranked)[1] <- "trend"
names(all_trends_ranked)[2] <- "mean_ranking"
names(all_trends_ranked)[3] <- "best_ranking"

all_trends_ranked <- all_trends_ranked [complete.cases(all_trends_ranked), ]
all_trends_ranked <- distinct(all_trends_ranked)
all_trends_ranked$best_ranking <- as.numeric (all_trends_ranked$best_ranking)
```



```r
all_trends_ranked$trend <- tolower(all_trends_ranked$trend)
all_trends_ranked$trend <- (gsub(" ","_",all_trends_ranked$trend))

for (i in seq_len(nrow(all_trends_ranked))){
burst <- bursty_values(all_trends_ranked$trend[i], twitter_dfm, docvars(twitter_corpus)$created_at)
if (is.null(burst) == T) {
  all_trends_ranked$max_burst_level[i] <- 0
  all_trends_ranked$max_burst_duration[i] <-   0
  all_trends_ranked$min_burst_duration[i] <-   0

  next
}
burst$duration <- burst$end - burst$start
all_trends_ranked$max_burst_level[i] <- max(burst$level)
all_trends_ranked$max_burst_duration[i] <-  max(burst$duration)
all_trends_ranked$min_burst_duration[i] <-  min(burst$duration)
all_trends_ranked$avg_burst_duration[i] <- mean(burst$duration)

}

for (i in seq_len(nrow(all_trends_ranked))){
  words_to_search <- all_trends_ranked$trend[i]
  all_trends_ranked$number_of_mentions[i] <-  topfeatures[words_to_search,]
}

all_trends_ranked <- all_trends_ranked[order(all_trends_ranked[,3], -all_trends_ranked[,4]),]
```