Sentiment Analysis of Kansans' Tweets

K C, Rajiv Jung

10/03/2020

Read me first

Please note following before you try to run this code:

Show/Hide

- Some chunks, marked by (*) in this code will not run by its own. Twitter API access keys and Google Map API access code will be REQUIRED to be fully able to reproduce the results of this project.
- Please check this link for how to get the Twitter API keys: https://developer.twitter.com/en/docs/authentication/oauth-1-0a/obtaining-user-access-tokens

  OR here: https://www.slickremix.com/docs/how-to-get-api-keys-and-tokens-for-twitter/
- Here's how to obtain the Google API key: https://developers.google.com/maps/documentation/javascript/get-api-key
- Once the required keys are supplied a few lines of code should be removed or commented out. Each of codes have a comment supplied with it to do so.
- However, by locating all the files from datasets and list-of-words in your working directory, you should be able to obtain the most of the final results.
- Some familiarity with R - programming language is assumed to be able to reproduce the results. With a strong internet connection and an i-7 computer, all the computation in this code may take somewhere between 3-6 minutes without the results(files listed above) being loaded or around 1-2 minutes with the results being loaded.

Getting the Tweet data

Change your directory

Load required packages

Enter install.packages('package name') to install a package in R.

For help type ?install.packages.

```r
# load twitter library - the rtweet
library(rtweet)

# plotting and pipes - tidyverse!
library(ggplot2)
library(dplyr)

# text mining library
library(tidytext)
#to webscrape
library(rvest)

# to geo-code google api
library(tidyverse)
library(ggmap)

# to create spatial plots with map of KS
library(usmap)

# to visualize word clouds
library(tm)
library(wordcloud)
```

Load personal information to access twitter within R (*)

Please note: You will need to register on twitter to get the following keys so that you can log into Twitter and scrape the Tweets.

After getting the required keys, change eval = TRUE in 'knitr' code chunk below.

```r
appname <- "YOUR APP USERNAME"

## api key
key <- "YOUR API KEY"

## api secret
secret <- "YOUR SECRET KEY"

access_token <- "YOUR ACCESS TOKEN"
access_token_secret <- "YOUR ACCESS TOKEN SECRET"

# create token named "twitter_token"
twitter_token <- create_token(
  app = appname,
  consumer_key = key,
  consumer_secret = secret,
  access_token = access_token,
  access_secret = access_token_secret)
```

Get Kansas county data

Scraping data from Wikipedia

```
## scraping Kansas County data from wiki)
url <- "https://en.wikipedia.org/wiki/List_of_counties_in_Kansas"

my_html <- read_html(url)

my_tables <- html_nodes(my_html,"table")[[2]] # the information we are looking at is at table 2 in the
ks_counties <- html_table(my_tables)

ks_counties <- ks_counties[,c(1,2,3,4,8,9)]
#head(ks_counties)

colnames(ks_counties) <- c("county","fips", "county_seat", "est", "population", "area_sq_mi")
#head(ks_counties)

ks_counties$population <- as.numeric(gsub(",","",ks_counties$population))
ks_counties$area_sq_mi <- gsub("\\s.*","",ks_counties$area_sq_mi)
ks_counties$area_sq_mi <- as.numeric(gsub(",","",ks_counties$area_sq_mi))
ks_counties$county_seat <- paste(ks_counties$county_seat, "KS")
head(ks_counties)
```

```
##            county fips       county_seat  est population area_sq_mi
## 1    Allen County    1           Iola KS 1855      13319        503
## 2 Anderson County    3        Garnett KS 1855       7917        583
## 3 Atchison County    5       Atchison KS 1855      16813        432
## 4   Barber County    7 Medicine Lodge KS 1867       4861       1134
## 5   Barton County    9     Great Bend KS 1867      27557        894
## 6  Bourbon County   11     Fort Scott KS 1855      14897        637
```

```
## Scraping Kansas Household Income Data

url <- "https://en.wikipedia.org/wiki/List_of_Kansas_locations_by_per_capita_income"

my_html <- read_html(url)

my_tables <- html_nodes(my_html,"table")[[2]] # the information we are looking at is at table 2 in the
ks_counties_inc <- html_table(my_tables, fill = T)
#head(ks_counties_inc)
ks_counties_inc <- na.omit(ks_counties_inc)

for (i in 3:ncol(ks_counties_inc)) {
  ks_counties_inc[,i] <- as.numeric(gsub("\\$|,", "", ks_counties_inc[,i]))
}
head(ks_counties_inc)
```

```
##   Rank    County Per capitaincome Medianhouseholdincome Medianfamilyincome
## 1    1   Johnson            37882                 73733              90380
## 2    2     Scott            28872                 58341              65000
## 3    3   Greeley            28698                 55972              63967
## 4    4      Ness              27622                  47639               55875
## 6    5 McPherson              26467                  53026               68016
## 7    6    Butler              26436                  56290               66581
##   Population Number ofhouseholds
## 1     544179              212882
## 2       4936                1983
## 3       1247                 525
## 4       3107                1365
## 6      29180               11748
## 7      65880               23992
```

```
ks_counties_inc <- ks_counties_inc[order(ks_counties_inc$County),]
#head(ks_counties_inc)
ks_counties <- cbind(ks_counties, ks_counties_inc$Medianfamilyincome)
#head(ks_counties)
```

Geo-Coding with R Using Google API (*)

Likewise for twitter, geo-coding using goggle maps also require you an api key.

After getting the required key, change eval = TRUE in 'knitr' code chunk below.

```
register_google(key = "YOUR GOOGLE API KEY")

coords.df <- as.data.frame(ks_counties)
locations.df <- mutate_geocode(coords.df, county_seat)

geo_code_twitter <- paste(locations.df$lat,",",locations.df$lon,",","10mi")
geo_code_twitter <- gsub(" ", "", geo_code_twitter,
                         fixed = TRUE)
ks_counties.df <- cbind(locations.df, geo_code_twitter)
# ks_counties.df$fips <- ks_counties.df$fips + 20000 # fips code in 5 charc
# head(ks_counties.df)
# tail(ks_counties.df)
```

Saving data up to the last check-point

```
# write.csv(ks_counties.df, "ks_county_data.csv")
# saveRDS(ks_counties.df, file = "ks_county_data.rds") # save as a RDS file
# the file uploaded might have some edit like column names and etc
```

At this point, the "ks_counties_data.csv" file should contain data on KS counties, geo-coordinates of the county seat, and the median family income.

Twitter Data Extraction (*)

After loading the required TWITTER access keys in above part, change eval = TRUE in 'knitr' code chunk below.

```r
## load KS counties data with geo-cords
ks.counties <- read.csv("ks_county_data.csv")
ks.counties$geo_code_twitter <- gsub('\"', "",
                                     ks.counties$geo_code_twitter,
                                     fixed = TRUE) # remove Remove Backslash and Quotations

## scrape and save all the desired tweets from 11/17/2020-11/24/2020
tweets.dataframe <- list()

for (i in 1:nrow(ks.counties)) {
  try(result <- search_tweets("lang:en", include_rts = FALSE,
                          geocode = ks.counties$geo_code_twitter[i],
                          n = 100)[,c(3:6,70,72,74,75,78,79,81,84)])
  # debug with "try" just like try-except in python
  Sys.sleep(1.5)

  result$location <- ks.counties$county[i]
  result$coords_coords <- paste(ks.counties$lat[i],",",ks.counties$lon[i])

  if (nrow(result) > 0) {  # only if result has data
    tweets.dataframe <- c(tweets.dataframe, list(result))
    }
  }

# to combine them into one data frame
tweets.df <- do_call_rbind(tweets.dataframe)
dim(tweets.df)

length(tweets.df$text[duplicated(tweets.df$text)==TRUE]) # check no. duplicate tweets
# remove duplicate tweets
tweets.df <- tweets.df[!duplicated(tweets.df$text), ]
dim(tweets.df)

# tweets without scores
# write.csv(tweets.df, "tweets_ks.csv")
```

Data Cleaning for Filtration

```r
tweets.df <- read.csv("tweets_ks.csv") # comment out this if you are extracted tweets
attr(tweets.df$created_at, "tzone") <- "" # UTC to CST
tweets.df$source <- gsub("Twitter/for", "",tweets.df$source)
tweets.df$description <- plain_tweets(tweets.df$description)

# tryTolower() function changes the texts in the tweets to all lower letters
tryTolower = function(x)
{
  y = NA
  # tryCatch error
  try_error = tryCatch(tolower(x), error = function(e) e)
  # if not an error
  if (!inherits(try_error, "error"))
```

```
    y = tolower(x)
  return(y)
}

# The tweet_clean() function cleans the twitter feeds and splits the strings into a vector of words

tweet_clean <- function(tweets) {
  tweets <- plain_tweets(tweets)
  tweets <- gsub("//.*", "", tweets)
  tweets <- gsub('[[:punct:]]','',tweets)
  tweets <- gsub('[[:cntrl:]]','',tweets)
  tweets <- gsub('\\d+','',tweets)
  tweets <- gsub('[[:digit:]]','',tweets)
  tweets <- gsub('@\\w+','',tweets)
  tweets <- gsub('http\\w+','',tweets)
  tweets <- gsub("^\\s+|\\s+$", "", tweets)
  #tweets <- lapply(tweets,function(x) tryTolower(x))
  tweets <- tryTolower(tweets)
  tweets <- unlist(strsplit(tweets," "))
  return(tweets)
}
```

Filter tweets (-ve/ +ve)

```
ks.counties <- read.csv("ks_county_data.csv") # please comment/remove this line of code out if you are
positives <- readLines("positive-words.txt")
negatives <- readLines("negative-words.txt")
score <- c()
for (j in 1:length(tweets.df$text)) {

  for (i in 1:length(unlist(strsplit(tweets.df$text[j], " ")))) {
    words <- tweet_clean(tweets.df$text[j])
    # compare words to the dictionaries of positive & negative terms
    positive_matches = match(words, positives)
    negative_matches = match(words, negatives)
    # get the position of the matched term or NA
    # we just want a TRUE/FALSE
    positive_matches = !is.na(positive_matches)
    negative_matches = !is.na(negative_matches)
    # final score
    score[j] = sum(positive_matches) - sum(negative_matches)
    #print(score)
  }
}
#print(score) # score for each individual tweets
tweets.df <- cbind(tweets.df, score)

# mean tweet score for each individual county
mean_score_county <- c()
for (k in 1:length(ks.counties$county)) {
  mean_score_county[k] <- try(mean(tweets.df$score[which(tweets.df$location==ks.counties$county[k])]))
```

```r
}
ks.counties.a <- cbind(ks.counties, mean_score_county)
```

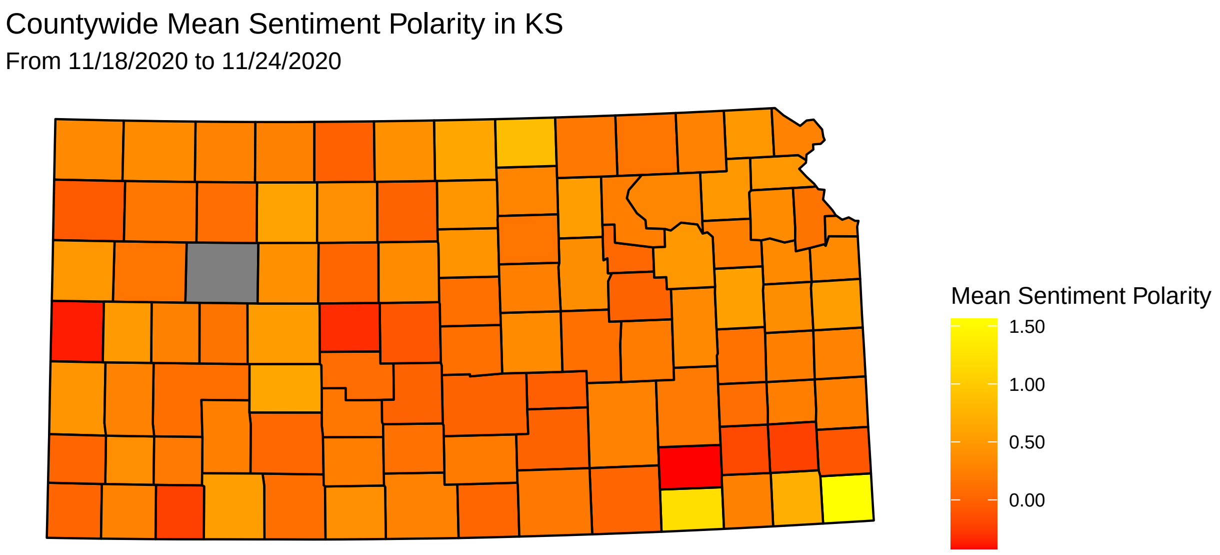
Save the data with tweet score

```r
# write.csv(tweets.df, "tweet_ks_sc.csv")
# write.csv(ks.counties.a, "ks_county_data_sc.csv")
```

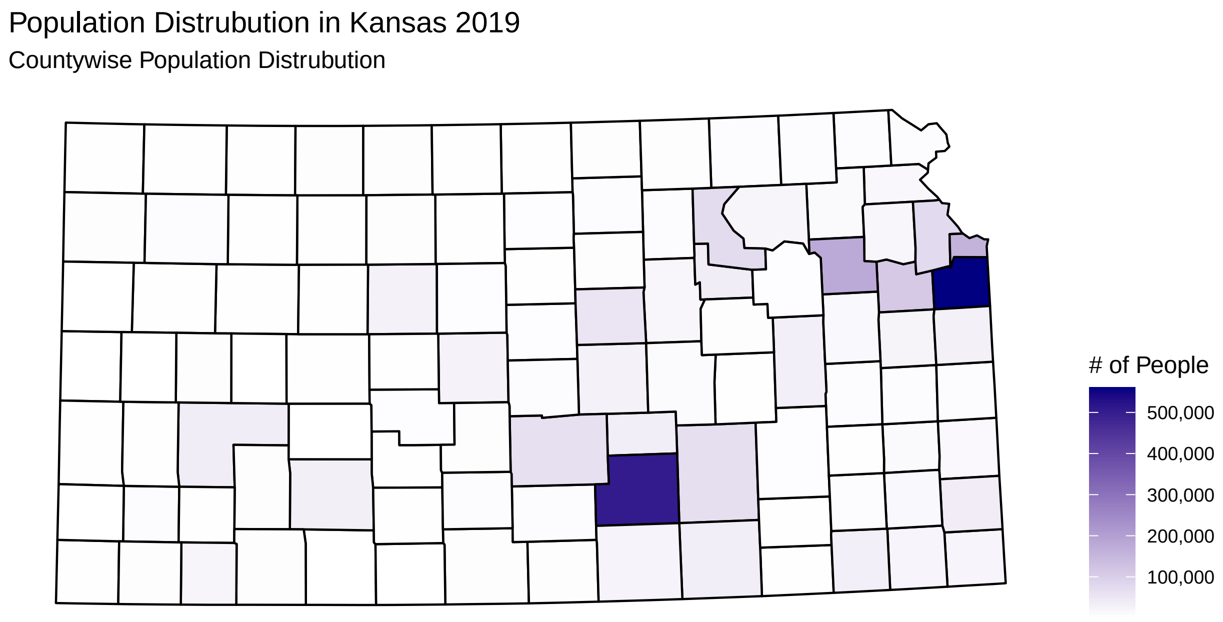
Data Visualization

Spatial Plots

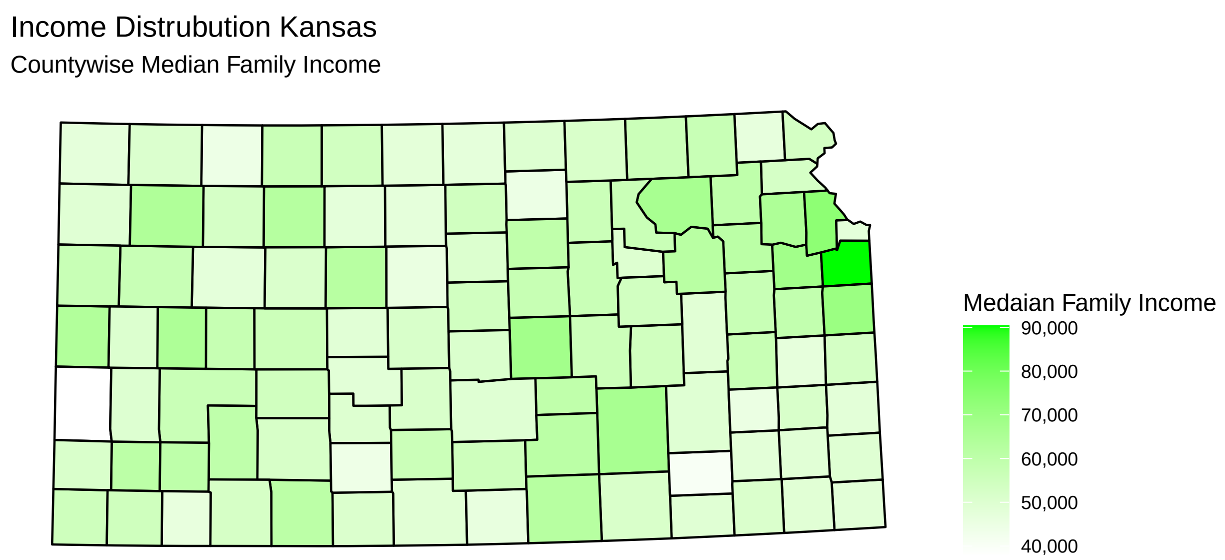
```r
ks.counties.b <- read.csv("ks_county_data_sc.csv")

# plot mean sentiment score of each county in Kansas
plot_usmap(data = ks.counties.b, values = "mean_score_county", include = "KS",
           regions = "counties", labels = F) +
  labs(title = "Countywide Mean Sentiment Polarity in KS",
       subtitle = "From 11/18/2020 to 11/24/2020") +
  scale_fill_continuous(
    low = "red", high = "yellow", name = "Mean Sentiment Polarity", label = scales::comma
  ) + theme(legend.position = "right")
```


Countywide Mean Sentiment Polarity in KS
From 11/18/2020 to 11/24/2020
Mean Sentiment Polarity
1.50
1.00
0.50
0.00

```
# plot population distribution in each county in Kasnas
plot_usmap(data = ks.counties.b, values = "population", include = "KS",
           regions = "counties", labels = F) +
  labs(title = "Population Distrubution in Kansas 2019",
       subtitle = "Countywise Population Distrubution") +
  scale_fill_continuous(
    low = "white", high = "navy", name = "# of People", label = scales::comma
  ) + theme(legend.position = "right")
```


Population Distrubution in Kansas 2019
Countywise Population Distrubution
# of People
500,000
400,000
300,000
200,000
100,000

```r
# plot median family income of each county in Kansas
plot_usmap(data = ks.counties.b, values = "median_fam_inc", include = "KS",
           regions = "counties", labels = F) +
  labs(title = "Income Distrubution Kansas",
       subtitle = "Countywise Median Family Income") +
  scale_fill_continuous(
    low = "white", high = "green", name = "Medaian Family Income", label = scales::comma
  ) + theme(legend.position = "right")
```



some descriptive analysis of tweets

```
ks.counties.c <- na.omit(ks.counties.b)[, c(1, 5:7, 11)] # coz we have a NA in our data
# county with the highest positive tweets average
ks.counties.c[ks.counties.c$mean_score_county == max(ks.counties.c$mean_score_county),]
```

```
##              county population area_sq_mi median_fam_inc mean_score_county
## 11 Cherokee County      21226        587          48319          1.561224
```

```
## county with the lowest positive tweets average
ks.counties.c[ks.counties.c$mean_score_county == min(ks.counties.c$mean_score_county),]
```

```
##         county population area_sq_mi median_fam_inc mean_score_county
## 25 Elk County       2720        648          40463        -0.4210526
```

```
## counties with positive tweets average
nrow(ks.counties.c[ks.counties.c$mean_score_county > 0, ])
```

```
## [1] 91
```

```r
## counties with negative tweets average
nrow(ks.counties.c[ks.counties.c$mean_score_county < 0, ])
```

```
## [1] 11
```

```r
## counties with neutral tweets average
nrow(ks.counties.c[ks.counties.c$mean_score_county == 0, ])
```

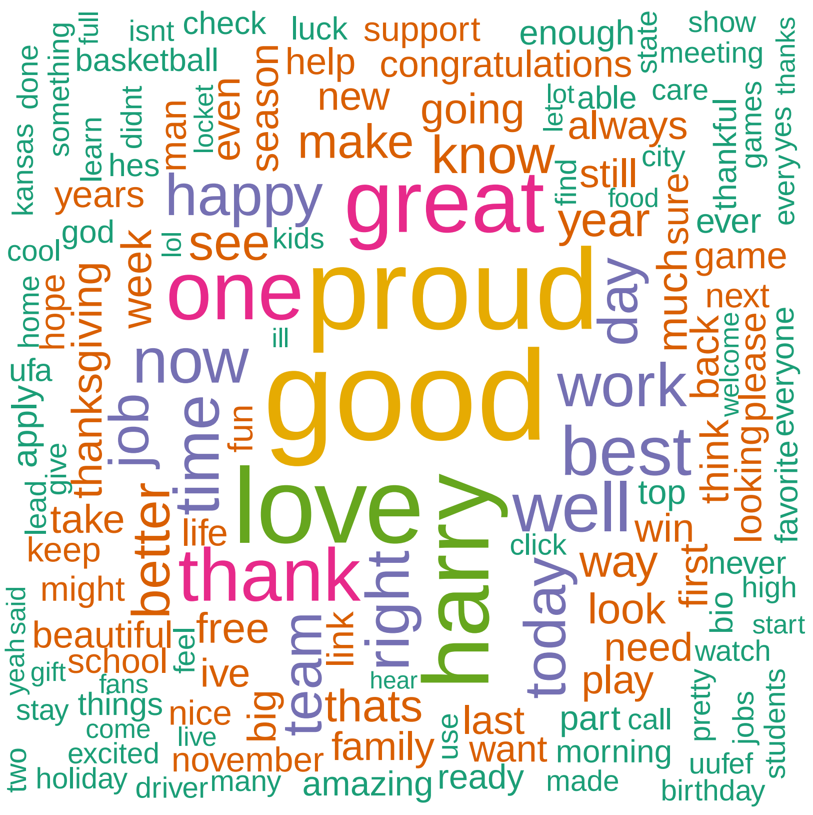
```
## [1] 2
```

```r
# summary of average tweets polarity
summary(ks.counties.c$mean_score_county)
```

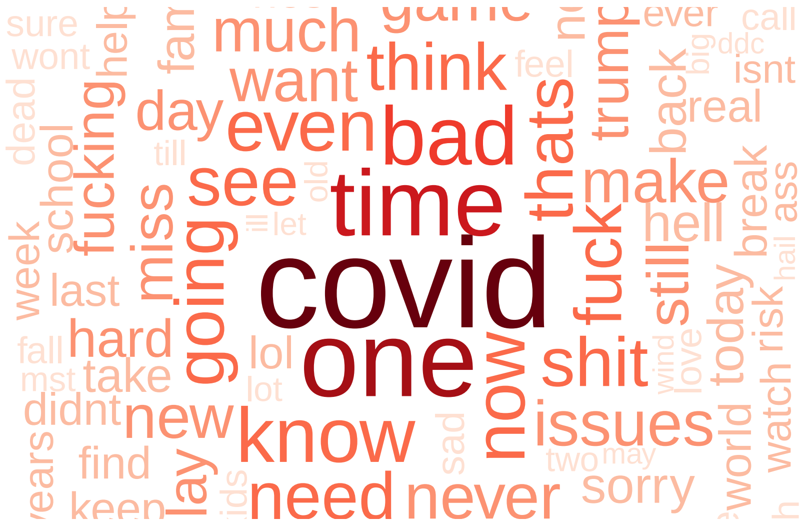
```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.42105  0.09951  0.25000  0.24832  0.37166  1.56122
```

Word Clouds

```r
set.seed(8613)
tweets.df <- read.csv("tweet_ks_sc.csv")
custom_stop_words = c("will", "dont", "youre", "else", "also", "say",
                      "get", "gets", "doesnt", "though", "theres",
                      "gonna", "like", "can", "cant", "theyre",
                      "got", "really", "just", "people")

# Positive Word Cloud
# For those tweets that have overall positive score
tweetscorpus=Corpus(VectorSource(tweet_clean(tweets.df$text[tweets.df$score > 0])))
tweetscorpus=tm_map(tweetscorpus,removeWords,c(stopwords("english"), custom_stop_words))

wordcloud(tweetscorpus,random.order = FALSE,rot.per = 0.40,
          use.r.layout = FALSE,colors = brewer.pal(6,"Dark2"),
          max.words = 200)
```



```
# Negative Word Cloud
# For those tweets that have overall negative score
tweetscorpus=Corpus(VectorSource(tweet_clean(tweets.df$text[tweets.df$score < 0])))
tweetscorpus=tm_map(tweetscorpus,removeWords,c(stopwords("english"), custom_stop_words))

wordcloud(tweetscorpus,random.order = FALSE,rot.per = 0.40,
          use.r.layout = FALSE,colors = brewer.pal(9,"Reds"),
          max.words = 200)
```


covid
one
time
bad
know
now
thats
think
even
see
going
fuck
shit
issues
much
want
make
need
never
new
hard
take
miss
fucking
school
still
today
break
hell
back
trump
game
day
last
lol
sorry
find
didnt
real
isnt
sure
wont
dead
week
watch
risk
world
love
wind
two
may
sad
lot
let
old
till
feel
ever
call
big
ddc
ass
hail
fall
mst
help
keep
play

All Done!!