Applied Statistical Programming - Spring 2022

Problem Set 4

Due Wednesday, March 23, 10:00 AM (Before Class)

Instructions

1. The following questions should each be answered within an Rmarkdown file. Be sure to provide many comments in your code blocks to facilitate grading. Undocumented code will not be graded.
2. Work on git. Continue to work in the repository you forked from https://github.com/johnsontr/AppliedStatisticalProgramming2022 and add your code for Problem Set 4. Commit and push frequently. Use meaningful commit messages because these will affect your grade.
3. You may work in teams, but each student should develop their own Rmarkdown file. To be clear, there should be no copy and paste. Each keystroke in the assignment should be your own.
4. For students new to programming, this may take a while. Get started.

`tidyverse`

Your task in this problem set is to combine two datasets in order to observe how many endorsements each candidate received using only `dplyr` functions. Use the same Presidential primary polls that were used for the in class worksheets on February 28 and March 2.

```
# Change eval=FALSE in the code block. Install packages as appropriate.
#install.packages("fivethirtyeight")
library(fivethirtyeight)
```

```
## Warning: package 'fivethirtyeight' was built under R version 4.0.5
```

```
## Some larger datasets need to be installed separately, like senators and
## house_district_forecast. To install these, we recommend you install the
## fivethirtyeightdata package by running:
## install.packages('fivethirtyeightdata', repos =
## 'https://fivethirtyeightdata.github.io/drat/', type = 'source')
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.0.5
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1

## Warning: package 'ggplot2' was built under R version 4.0.5

## Warning: package 'tibble' was built under R version 4.0.5

## Warning: package 'tidyr' was built under R version 4.0.5

## Warning: package 'readr' was built under R version 4.0.5

## Warning: package 'dplyr' was built under R version 4.0.5

## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
# URL to the data that you've used.
url <- 'https://jmontgomery.github.io/PDS/Datasets/president_primary_polls_feb2020.csv'
polls <- read_csv(url)
```

```
## Rows: 16661 Columns: 33

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (21): state, pollster, sponsors, display_name, pollster_rating_name, fte...
## dbl  (8): question_id, poll_id, cycle, pollster_id, pollster_rating_id, samp...
## lgl  (3): internal, tracking, nationwide_batch

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
Endorsements <- endorsements_2020 # from the fiverthirtyeight package
```

First, create two new objects `polls` and `Endorsements`. Then complete the following.

- Change the `Endorsements` variable name endorsee to `candidate_name`.
- Change the `Endorsements` dataframe into a `tibble` object.
- Filter the `poll` variable to only include the following 6 candidates: Amy Klobuchar, Bernard Sanders,Elizabeth Warren, Joseph R. Biden Jr., Michael Bloomberg, Pete Buttigieg **and** subset the dataset to the following five variables: `candidate_name, sample_size, start_date, party, pct`
- Compare the candidate names in the two datasets and find instances where the a candidates name is spelled differently i.e. Bernard vs. Bernie. Using only `dplyr` functions, make these the same across datasets.

- Now combine the two datasets by candidate name using `dplyr` (there will only be five candidates after joining).
- Create a variable which indicates the number of endorsements for each of the five candidates using `dplyr`.

```
# two new objects are created by running the given code
polls <- read_csv(url)
```

```
## Rows: 16661 Columns: 33

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (21): state, pollster, sponsors, display_name, pollster_rating_name, fte...
## dbl  (8): question_id, poll_id, cycle, pollster_id, pollster_rating_id, samp...
## lgl  (3): internal, tracking, nationwide_batch

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
Endorsements <- endorsements_2020 # from the fiverthirtyeight package

# changing variable name to candidate name
Endorsements <- Endorsements %>%
  rename(candidate_name = endorsee)

# changing the dataframe into a tibble
Endorsements <- as_tibble(Endorsements)

# filtering down to six candidates
polls_six <- polls %>%  filter (candidate_name %in% c("Amy Klobuchar", "Bernard Sanders", "Elizabeth Wa
                                          "Joseph R. Biden Jr.", "Michael Bloomberg", "Pete Buttigieg" ))

# keep only five variables
polls_six <- polls_six %>% select(candidate_name, sample_size, start_date, party, pct)


  # check for any differences in candidate names
unique(polls_six$candidate_name)
```

```
## [1] "Bernard Sanders"     "Pete Buttigieg"      "Joseph R. Biden Jr."
## [4] "Amy Klobuchar"       "Elizabeth Warren"    "Michael Bloomberg"
```

```
unique(Endorsements$candidate_name)
```

```
##  [1] "John Delaney"       "Joe Biden"          "Julian Castro"
##  [4] "Kamala Harris"      "Bernie Sanders"     "Cory Booker"
##  [7] "Amy Klobuchar"      "Elizabeth Warren"   "Jay Inslee"
## [10] "John Hickenlooper"  "Beto O'Rourke"      "Kirsten Gillibrand"
## [13] "Pete Buttigieg"     "Eric Swalwell"      "Steve Bullock"
## [16] NA
```

```
  # so there is a problem with Bernie and Joe. We'll opt for the spellings in polls data
Endorsements_names <- Endorsements %>%
  mutate(candidate_name = ifelse(candidate_name == "Joe Biden", "Joseph R. Biden Jr.",
                                 ifelse(candidate_name == "Bernie Sanders", "Bernard Sanders", candidat

  # checking to see if they were fixed:
unique(Endorsements_names$candidate_name)
```

```
##  [1] "John Delaney"        "Joseph R. Biden Jr." "Julian Castro"
##  [4] "Kamala Harris"       "Bernard Sanders"     "Cory Booker"
##  [7] "Amy Klobuchar"       "Elizabeth Warren"    "Jay Inslee"
## [10] "John Hickenlooper"   "Beto O'Rourke"       "Kirsten Gillibrand"
## [13] "Pete Buttigieg"      "Eric Swalwell"       "Steve Bullock"
## [16] NA
```

```
# combine datasets with only five candidates
combined <- polls_six %>%
  inner_join(Endorsements_names, by="candidate_name")

  # are there only five candidates? yes! because we used inner_join
unique(combined$candidate_name)
```

```
## [1] "Bernard Sanders"     "Pete Buttigieg"      "Joseph R. Biden Jr."
## [4] "Amy Klobuchar"       "Elizabeth Warren"
```

```
# creating numbers-of-endorsement variable: I create a separate dataset of counts first
no_endorsements <- Endorsements_names %>%
  filter(candidate_name %in% c("Amy Klobuchar", "Bernard Sanders","Elizabeth Warren",
                               "Joseph R. Biden Jr.", "Pete Buttigieg")) %>% #limiting to only the give
  group_by(candidate_name) %>% # we will count per candidate
  summarise(no_endorsements = sum(!is.na(endorser))) # what are we counting? number of endorsers

unique(no_endorsements$no_endorsements)
```

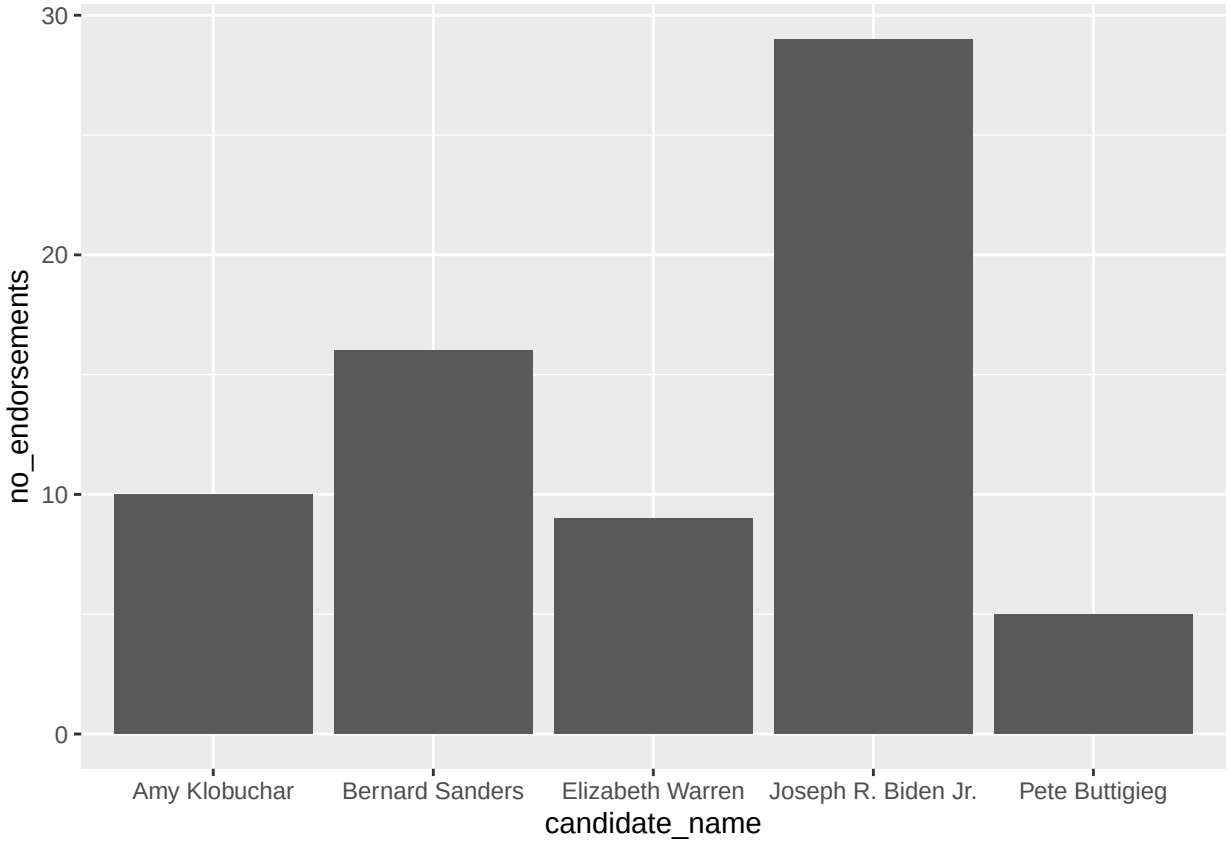
```
## [1] 10 16  9 29  5
```

```
# as a variable in the merged dataset
combined <- combined %>%
  left_join(no_endorsements, by="candidate_name")
  # I check to see if they match the number of endorsement as in the count dataset
unique(combined$no_endorsements)
```

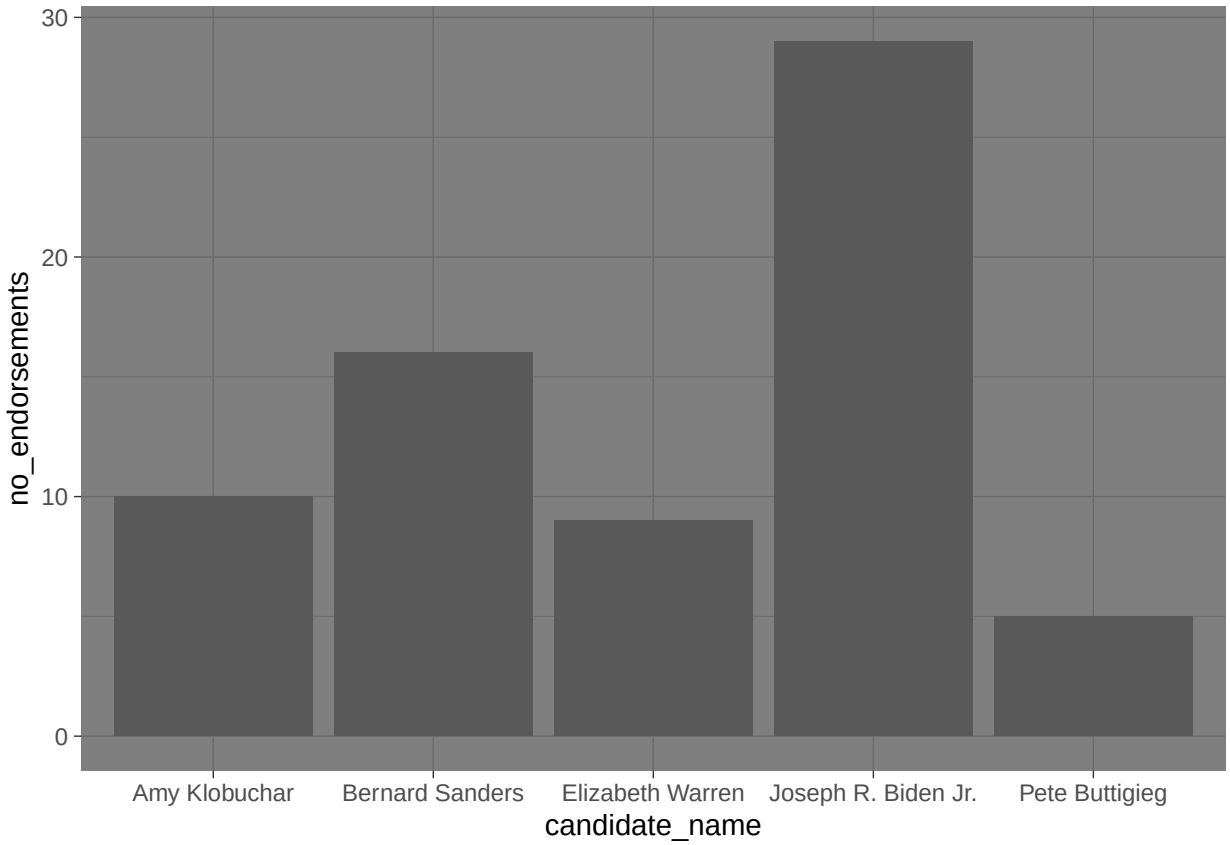
```
## [1] 16  5 29 10  9
```

- Plot the number of endorsement each of the 5 candidates have using `ggplot()`. Save your plot as an object `p`.
- Rerun the previous line as follows: `p + theme_dark()`. Notice how you can still customize your plot without rerunning the plot with new options.
- Now, using the knowledge from the last step change the label of the X and Y axes to be more informative, add a title. Save the plot in your forked repository.

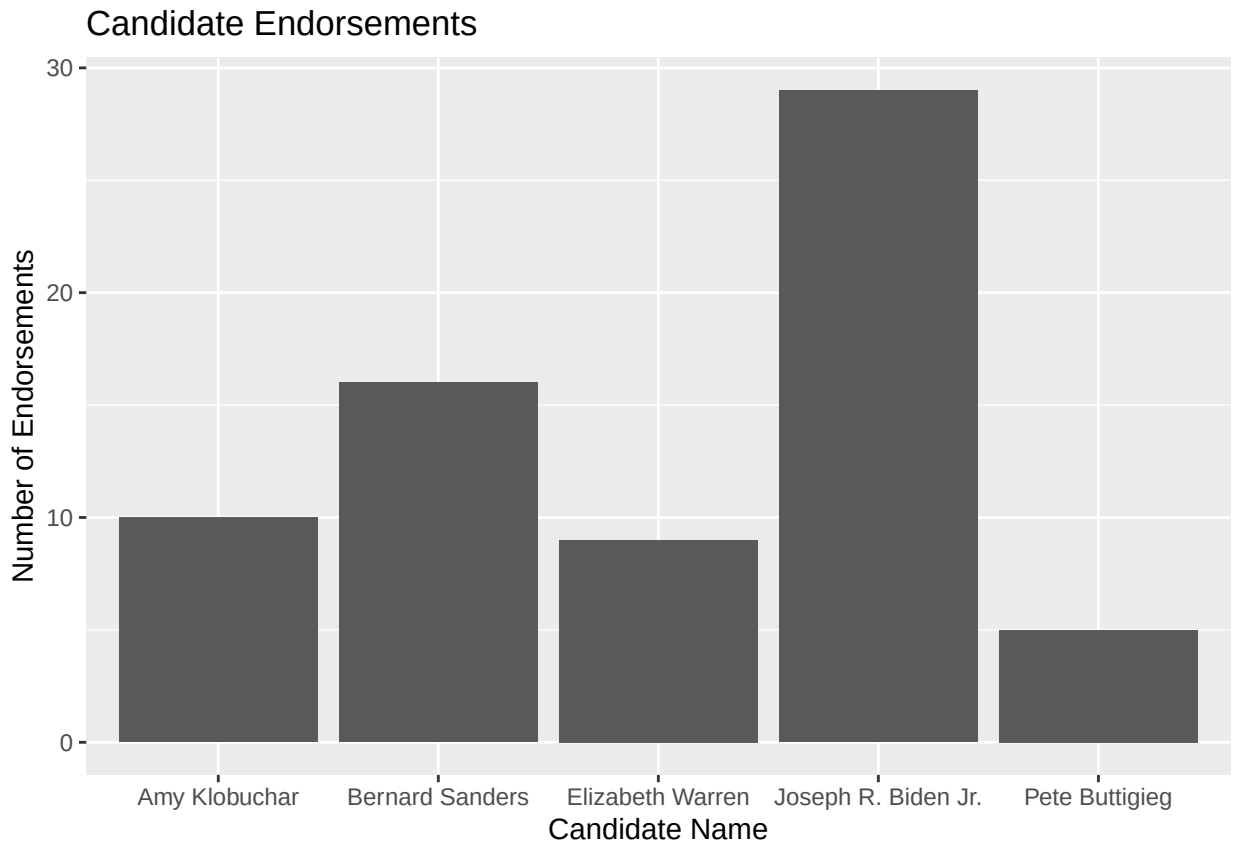
```
# plot the number of endorsements and save as an object
p <- ggplot(data=no_endorsements, mapping = aes(x = candidate_name, y = no_endorsements)) +
  geom_bar(stat = "identity")
print(p)
```




```
# theme_dark
p + theme_dark()
```




```
# adding titles and labels
p + labs(title="Candidate Endorsements", x="Candidate Name", y="Number of Endorsements")
```




```
# save the plot
ggsave("2016Candidate.endorsements.png")
```

```
## Saving 6.5 x 4.5 in image
```

Text-as-Data with `tidyverse`

For this question you will be analyzing Tweets from President Trump for various characteristics. Load in the following packages and data:

```
# Change eval=FALSE in the code block. Install packages as appropriate.
library(tidyverse)
#install.packages('tm')
library(tm)
```

```
## Warning: package 'tm' was built under R version 4.0.5
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

```
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
#install.packages('lubridate')
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 4.0.5
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
#install.packages('wordcloud')
library(wordcloud)
```

```
## Warning: package 'wordcloud' was built under R version 4.0.5
```

```
## Loading required package: RColorBrewer
```

```
trump_tweets_url <- 'https://politicaldatascience.com/PDS/Datasets/trump_tweets.csv'
tweets <- read_csv(trump_tweets_url)
```

```
## Rows: 32974 Columns: 6
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): source, text, created_at
## dbl (2): retweet_count, favorite_count
## lgl (1): is_retweet
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

- First separate the `created_at` variable into two new variables where the date and the time are in separate columns. After you do that, then report the range of dates that is in this dataset.
- Using `dplyr` subset the data to only include original tweets (remove retweents) and show the text of the President's **top 5** most popular and most retweeted tweets. (Hint: The `match` function can help you find the index once you identify the largest values.)

```
# two new variables
tweets.sep <- tweets %>%
    separate(created_at, c("date_created", "time_created"), sep = " ")

# to get the range of the dates, i first create a separate variable 'date' that
# codes the dates differently
tweets.sep$date <- as.Date(tweets.sep$date_created, format = "%m/%d/%Y")
range(tweets.sep$date)  # '2014-01-01' '2020-02-14'
```

```
## [1] "2014-01-01" "2020-02-14"
```

```
# subset the data to include only the original tweets, using is_retweet
# variable
originaltweets <- tweets %>%
    filter(!is_retweet)

# top five most 'liked' tweets
topfivelikes <- originaltweets %>%
    slice_max(favorite_count, n = 5)

# text of them
topfivelikes$text
```

```
## [1] "A$AP Rocky released from prison and on his way home to the United States from Sweden. It was a l
## [2] "https://t.co/VXeKiVzpTf"
## [3] "All is well! Missiles launched from Iran at two military bases located in Iraq. Assessment of ca
## [4] "MERRY CHRISTMAS!"
## [5] "Kobe Bryant despite being one of the truly great basketball players of all time was just gettin
```

```
# top five most 'retweeted' tweets

topfivert <- originaltweets %>%
    slice_max(retweet_count, n = 5)

# text of them
topfivert$text
```

```
## [1] "#FraudNewsCNN #FNN https://t.co/WYUnHjjUjg"
## [2] "TODAY WE MAKE AMERICA GREAT AGAIN!"
## [3] "Why would Kim Jong-un insult me by calling me \"old\" when I would NEVER call him \"short and fa
## [4] "A$AP Rocky released from prison and on his way home to the United States from Sweden. It was a l
## [5] "Such a beautiful and important evening! The forgotten man and woman will never be forgotten agai
```

```
# find index using AND condition to show the text of top 5 most popular and
# most retweeted tweets.
common <- originaltweets %>%
    filter(favorite_count %in% topfivelikes$favorite_count | retweet_count %in% topfivert$retweet_count)
    select(text)

# for we are creating a corpus of Tweet content, I use texts of Trump's
# original tweets sans Retweets
head(originaltweets$text)
```

```
## [1] "I'm seeing Governor Cuomo today at The White House. He must understand that National Security fa
## [2] "....which he actually has a military and legal obligation to do. His incredible wife Karen who l
## [3] "When I terminated John Kelly which I couldn't do fast enough he knew full well that he was way o
## [4] "DRAIN THE SWAMP! We want bad people out of our government!"
## [5] "Mini Mike is a 5'4" mass of dead energy who does not want to be on the debate stage with these p
## [6] "Mini Mike Bloomberg is a LOSER who has money but can't debate and has zero presence you will see
```

- Create a *corpus* of the tweet content and put this into the object `Corpus` using the `tm` (text mining) package. (Hint: Do the assigned readings.)

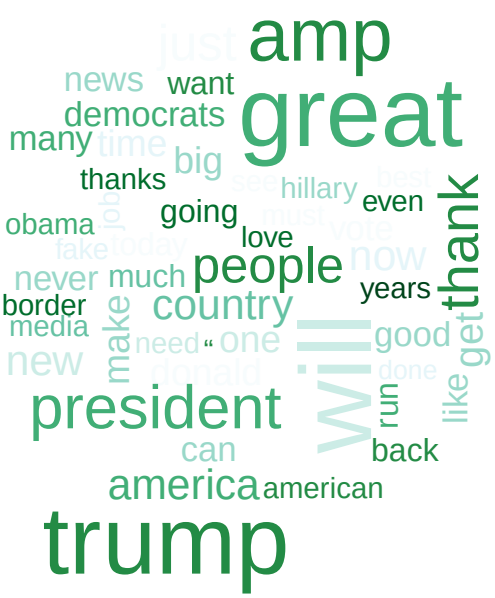
- Remove extraneous whitespace, remove numbers and punctuation, convert everything to lower case and remove 'stop words' that have little substantive meaning (the, a, it).
- Now create a `wordcloud` to visualize the top 50 words the President uses in his tweets. Use only words that occur at least three times. Display the plot with words in random order and use 50 random colors. Save the plot into your forked repository.
- Create a *document term matrix* called `DTM` that includes the argument `control = list(weighting = weightTfIdf)`
- Finally, report the 50 words with the the highest tf.idf scores using a lower frequency bound of .8.

```
## creating corpus
library(tm)
Corpus <- Corpus(VectorSource(as.vector(originaltweets$text)))

## removal using various commands of tm package: resulting new corpus is smaller in size
Corpus.clean <- Corpus %>%
  tm_map(stripWhitespace) %>% #remove extraneous whitespace
  tm_map(removeNumbers) %>% #remove numbers
  tm_map(removePunctuation) %>% #remove punctuations
  tm_map(content_transformer(tolower)) %>% #convert to lowercase
  tm_map(removeWords, stopwords(kind = "en")) #remove stop words
```

```
## Warning in tm_map.SimpleCorpus(., stripWhitespace): transformation drops
## documents

## Warning in tm_map.SimpleCorpus(., removeNumbers): transformation drops documents

## Warning in tm_map.SimpleCorpus(., removePunctuation): transformation drops
## documents

## Warning in tm_map.SimpleCorpus(., content_transformer(tolower)): transformation
## drops documents

## Warning in tm_map.SimpleCorpus(., removeWords, stopwords(kind = "en")):
## transformation drops documents
```

```
#creating wordcloud
library(wordcloud)


  #there were no variations in my random colors,
      #so I referenced RDocumentation Wordcloud page to learn how to set a color palette
  #designating a palette
pal <- brewer.pal(9,"BuGn")

##creating a plot using Wordcloud package
plot <- wordcloud(Corpus.clean, #body of words
          min.freq = 3, # should appear at least  three times
          max.words = 50, # top 50 words
          random.order = TRUE, #"in random order
          random.color = TRUE, colors = pal) #"in random colors"
```

```
## Warning in wordcloud(Corpus.clean, min.freq = 3, max.words = 50, random.order =
## TRUE, : realdonaldtrump could not be fit on page. It will not be plotted.
```




```
##save the plot
pdf("top50wordcloud.pdf")
```

```
## create a document-term matrix with the given argument included
DTM <- DocumentTermMatrix(Corpus, control = list(weighting = weightTfIdf))
```

```
## Warning in TermDocumentMatrix.SimpleCorpus(x, control): custom functions are
## ignored
```

```
## Warning in weighting(x): empty document(s): 12142
```

```
## 50 words with the the highest tf.idf scores using a lower frequency bound of .8.

  #to make the matrix function more efficiently, we remove sparse terms. Classmate Cecilia taught me ho
dtm.smaller <- removeSparseTerms(DTM, 0.9)

library(tidytext)
```

```
## Warning: package 'tidytext' was built under R version 4.0.5
```

```
dtm.df<- tidy(dtm.smaller) # to make operation easier, we make a tiny

  #the function immediately below was what I wanted to do but does not work
dtm.alternate <- dtm.df %>%
  arrange(desc(count))  %>%
  distinct(term, .keep_all = TRUE) %>%
          #set tf.idf lower frequency bound; tf.idf scores takes into account how the given words are u
  filter(count > 0.8) %>%
           #get highest ranking 50 words
  slice_max(count, n = 50)

 # this function does work but does not produce meaningful list of words
dtm.df %>%
  filter(count > 0.8) %>%
  slice_max(count, n = 50)
```

```
## # A tibble: 34 x 3
##    document term             count
##    <chr>    <chr>            <dbl>
##  1 8946     for              2.17
##  2 4756     great            1.14
##  3 10222    great            1.14
##  4 24908    @realdonaldtrump 0.957
##  5 152      great            0.952
##  6 561      great            0.952
##  7 974      great            0.952
##  8 1666     great            0.952
##  9 1852     great            0.952
## 10 2426     great            0.952
## # ... with 24 more rows
```