Applied Statistical Programming - Spring 2022

Cecilia Sui

Problem Set 4

Due Wednesday, March 16, 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.

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

```
## 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.
```

```r
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`.
- 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.

1. Change the `Endorsements` variable name endorsee to `candidate_name`

```r
# rename() to change variable name
Endorsements <- rename(Endorsements, candidate_name = endorsee)
# check to see whether the change took place
names(Endorsements)
```

```
##  [1] "date"           "position"       "city"           "state"
##  [5] "endorser"       "candidate_name" "endorser_party" "source"
##  [9] "order"          "category"       "body"           "district"
## [13] "points"
```

2. Change the `Endorsements` dataframe into a `tibble` object.

```r
Endorsements <- as_tibble(Endorsements)
# check if the class includes tibble
class(Endorsements)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

3. Filter the `polls` dataset 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`

```r
# create a new dataset to store the changes
polls_filtered <- polls %>% filter(candidate_name %in% c("Amy Klobuchar",
                                       "Bernard Sanders",
                                       "Elizabeth Warren",
                                       "Joseph R. Biden Jr.",
                                       "Michael Bloomberg",
                                       "Pete Buttigieg")) %>%
                            select(candidate_name,
                                   sample_size,
                                   start_date,
                                   party,
                                   pct)
```

4. 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.

```r
# compare the names of candidates of the two datasets
# identify differences
sort(unique(polls_filtered$candidate_name))
```

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

```r
sort(unique(Endorsements$candidate_name))
```

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

```r
# use dplyr to make them the same
polls_filtered <- polls_filtered %>%
  # "Bernard Sanders" --> "Bernie Sanders"
  mutate(candidate_name = replace(candidate_name,
                          candidate_name == "Bernard Sanders",
                          "Bernie Sanders" )) %>%
  # "Joseph R. Biden Jr." --> "Joe Biden"
  mutate(candidate_name = replace(candidate_name,
                          candidate_name == "Joseph R. Biden Jr.",
                          "Joe Biden"))

# Check to see if changes are made
sort(unique(polls_filtered$candidate_name))
```

```
## [1] "Amy Klobuchar"     "Bernie Sanders"    "Elizabeth Warren"
## [4] "Joe Biden"         "Michael Bloomberg" "Pete Buttigieg"
```

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

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

5. Now combine the two datasets by candidate name using `dplyr` (there will only be five candidates after joining).

```
joined_df <- merge(polls_filtered, Endorsements,
                       by = "candidate_name")
unique(joined_df$candidate_name)
```

```
## [1] "Amy Klobuchar"    "Bernie Sanders"   "Elizabeth Warren" "Joe Biden"
## [5] "Pete Buttigieg"
```

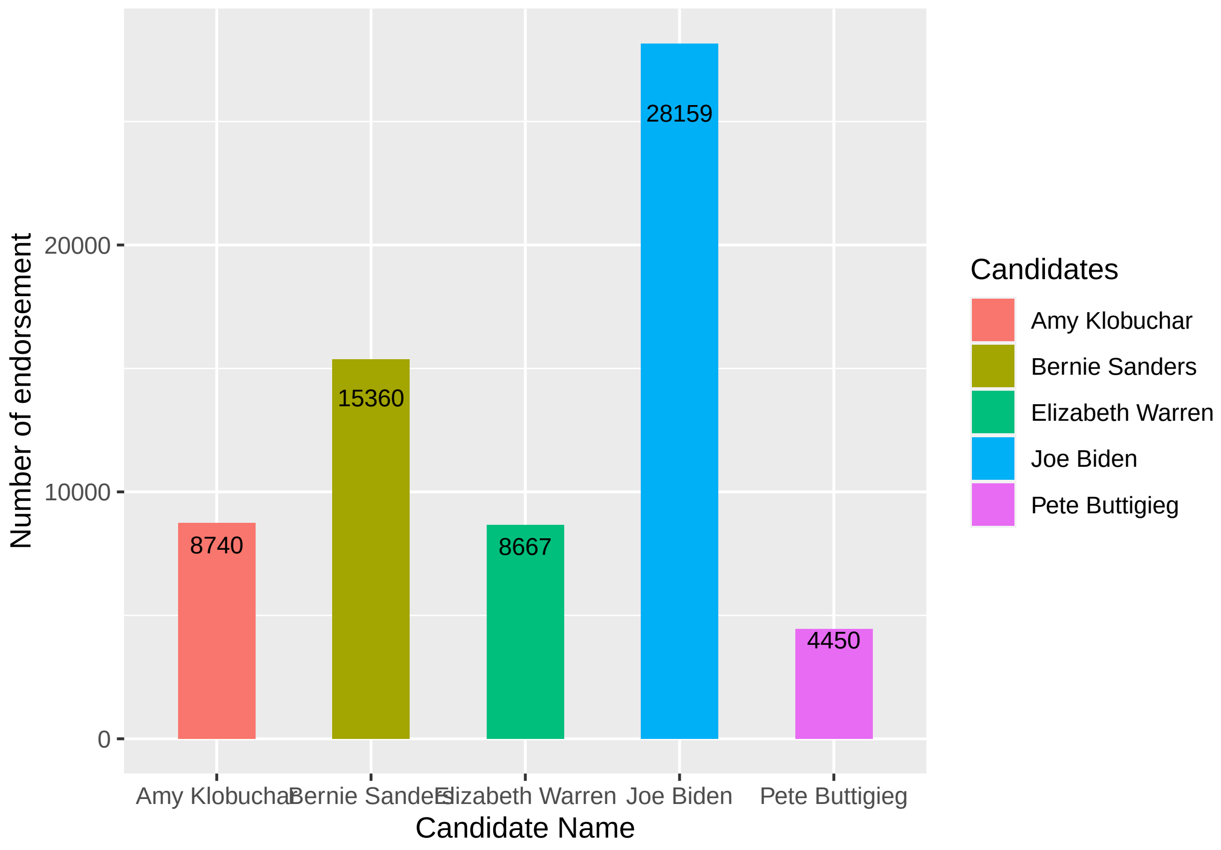
6. Create a variable which indicates the number of endorsements for each of the five candidates using `dplyr`.

```
cnt_endorsement <- joined_df %>%
                   group_by(candidate_name) %>%
                   count() %>%
                   rename(endorsement_count = n)
cnt_endorsement
```

```
## # A tibble: 5 x 2
## # Groups:   candidate_name [5]
##   candidate_name   endorsement_count
##   <chr>                        <int>
## 1 Amy Klobuchar                 8740
## 2 Bernie Sanders               15360
## 3 Elizabeth Warren              8667
## 4 Joe Biden                    28159
## 5 Pete Buttigieg                4450
```

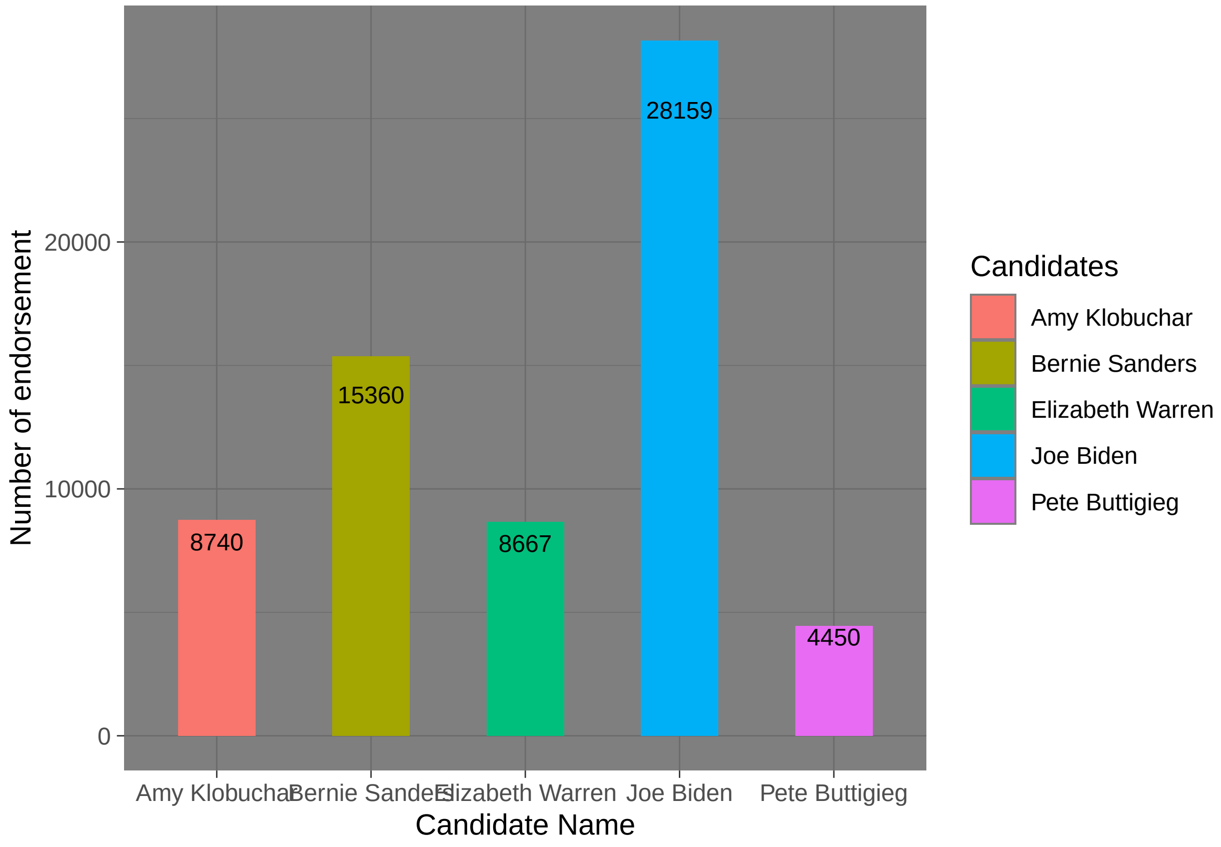
7. Plot the number of endorsement each of the 5 candidates have using `ggplot()`. Save your plot as an object `p`.

```
p <- ggplot(cnt_endorsement, aes(x = factor(candidate_name),
                                 y = endorsement_count,
                                 fill = cnt_endorsement$candidate_name,
                                 label = cnt_endorsement$endorsement_count)) +
  geom_bar(stat = "identity", width = 0.5) +
  xlab("Candidate Name") +
  ylab("Number of endorsement") +
  scale_fill_discrete(name = "Candidates") +
  geom_text(size = 3, position = position_stack(vjust = 0.9))
p
```



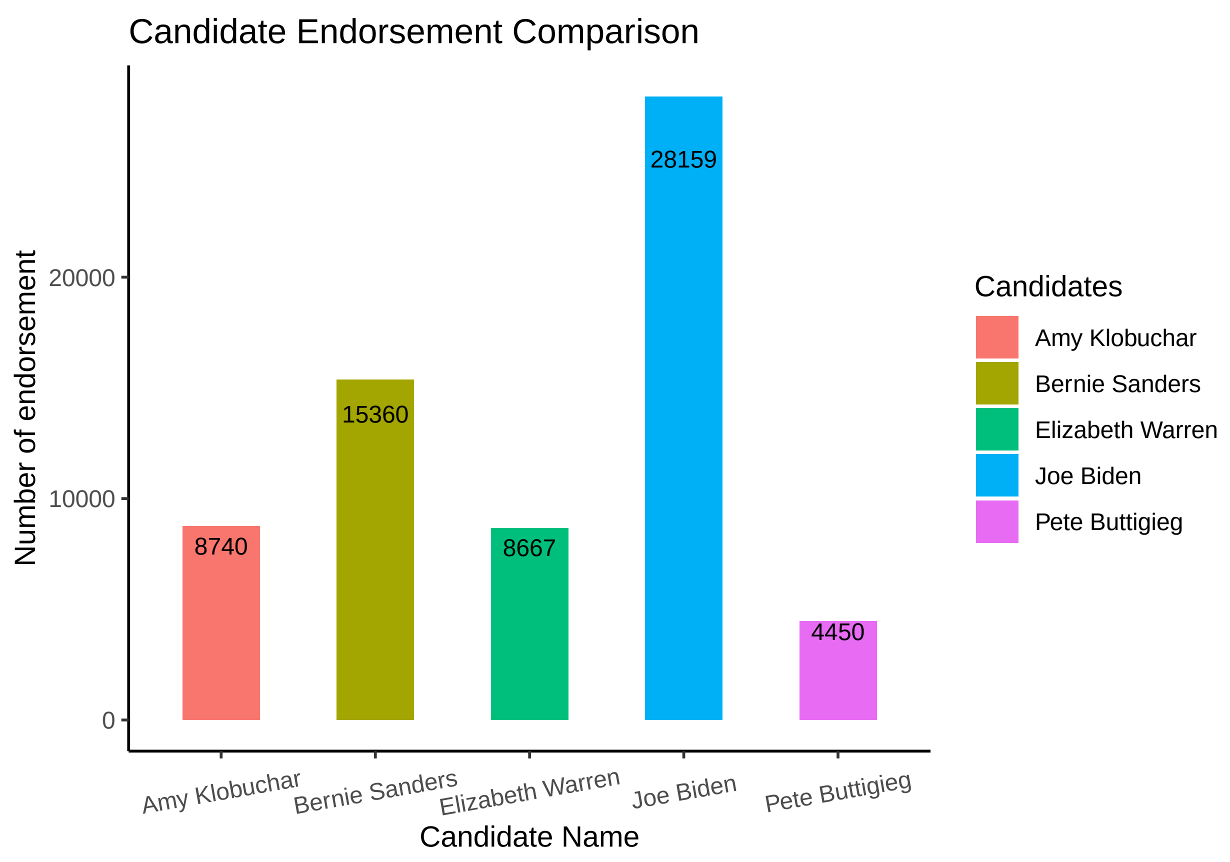
8. Rerun the previous line as follows: `p + theme_dark()`. Notice how you can still customize your plot without rerunning the plot with new options.

```
p + theme_dark()
```



9. 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.

```
p <- p + theme_classic() +
  ggtitle("Candidate Endorsement Comparison") +
  theme(axis.text.x = element_text(angle = 10, vjust = 0.5, hjust=0.5))
p
```



```
# save to file
# please see github repo
# ggsave("endorsement_plot.pdf", p)
```

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:

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

- 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.)
- 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.

1. 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.

```r
# modify in place
tweets <- tweets %>%
    # create a new col date to store the dates
    mutate(date = str_split_fixed(tweets$created_at, " ", 2)[,1]) %>%
    # create a new col time to store the timestamps
    mutate(time = str_split_fixed(tweets$created_at, " ", 2)[,2])

# convert to date
tweets$date <- as.Date(tweets$date, "%m/%d/%Y")

# find the range of dates:
# from "2014-01-01" to "2020-02-14"
max(tweets$date)
```

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

```r
min(tweets$date)
```

```
## [1] "2014-01-01"
```

2. 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.)

```
original_tweets <- tweets %>%
    # remove retweets
filter(is_retweet == FALSE)

original_tweets %>%
    # top 5 most popular
top_n(n = 5, wt = favorite_count) %>%
    arrange(desc(favorite_count)) %>%
    select(text)
```

```
## # A tibble: 5 x 1
##   text
##   <chr>
## 1 A$AP Rocky released from prison and on his way home to the United States from~
## 2 https://t.co/VXeKiVzpTf
## 3 All is well! Missiles launched from Iran at two military bases located in Ira~
## 4 MERRY CHRISTMAS!
## 5 Kobe Bryant despite being one of the truly great basketball players of all ti~
```

```
original_tweets %>%
    # top 5 most retweeted
top_n(n = 5, wt = retweet_count) %>%
    arrange(desc(retweet_count)) %>%
    select(text)
```

```
## # A tibble: 5 x 1
##   text
##   <chr>
## 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 cal~
## 4 "A$AP Rocky released from prison and on his way home to the United States fro~
## 5 "Such a beautiful and important evening! The forgotten man and woman will nev~
```

3. 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.)

```
Corpus <- VCorpus(VectorSource(original_tweets$text))
```

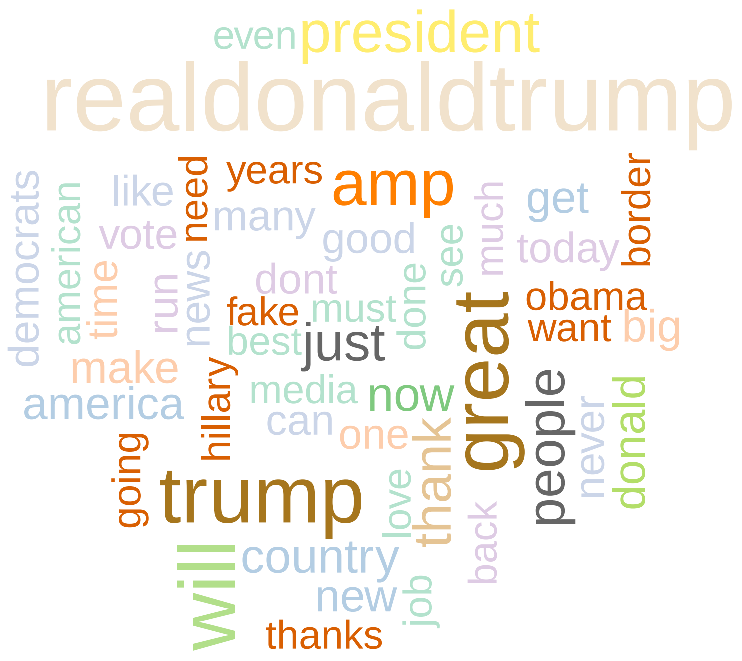
4. Remove extraneous whitespace, remove numbers and punctuation, convert everything to lower case and remove 'stop words' that have little substantive meaning (the, a, it).

```
Corpus <- Corpus %>%
  # remove numbers
  tm_map(removeNumbers) %>%
  # remove punctuation
  tm_map(removePunctuation, ucp = TRUE) %>%
```

```r
  # remove whitespace
  tm_map(stripWhitespace)

# convert to lower case
Corpus <- tm_map(Corpus, content_transformer(tolower))

# remove stop words
Corpus <- tm_map(Corpus, removeWords, stopwords("english"))

# remove urls
Corpus <- tm_map(Corpus, content_transformer(function(x, pattern) gsub("?(f|ht)tp(s?).*", "", x)))
```

5. 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.

```r
# generate term document matrix
tdm <- TermDocumentMatrix(Corpus)
matrix <- as.matrix(tdm)
words <- sort(rowSums(matrix), decreasing = T)
df <- data.frame(word = names(words), freq = words)


# for reproducibility
set.seed(12345)
# Extract color info
palette3_info <- brewer.pal.info[brewer.pal.info$category == "qual", ]
palette3_all <- unlist(mapply(brewer.pal,
                              palette3_info$maxcolors,
                              rownames(palette3_info)))
# Sample colors
palette3 <- sample(palette3_all, 50)

# generate wordcloud
wordcloud(words = df$word[1:50], # top 50 words
          freq = df$freq,
          min.freq = 3, # occur at least 3 times
          max.words = 50,
          random.order = TRUE,
          rot.per = 0.35,
          colors = palette3,
          scale = c(3,1))
```



6. Create a *document term matrix* called DTM that includes the argument `control = list(weighting = weightTfIdf)`

```
# Create document term matrix
# Doc as rows and Term as cols
DTM <- DocumentTermMatrix(Corpus,
                          control = list(weighting = weightTfIdf))
# inspect(DTM)
```

7. Finally, report the 50 words with the the highest tf.idf scores using a lower frequency bound of .8.

```
# findMostFreqTerms(DTM)
# DTM <- removeSparseTerms(DTM, .8)
df_DTM <- tidy(DTM)
df_DTM %>%
  # frequency bound of 0.8
  filter(count > 0.8) %>%
  # get top 50 scores
  slice_max(count , n = 50)
```

```
## # A tibble: 50 x 3
##    document term                 count
##    <chr>    <chr>                <dbl>
##  1 519      winred                14.9
##  2 578      antibenghazi          14.9
##  3 580      antibengahzi          14.9
##  4 1249     newhoaxsameswamp      14.9
##  5 1555     iranintlar            14.9
##  6 2493     donothingdemocrats    14.9
##  7 2706     fakewhistleblower     14.9
##  8 2750     rafbo                 14.9
##  9 7176     holocaustmemorialday  14.9
## 10 9787     usembassyjerusalem    14.9
## # ... with 40 more rows
```