Applied Statistical Programming - Spring 2022

Problem Set 3

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.

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

```r
# install.packages('fivethirtyeight')
library(fivethirtyeight)
```

```
## 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')
```

```r
library(tidyverse)
```

```
## -- 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   1.4.0     v forcats 0.5.1

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

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

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_character(),
##   question_id = col_double(),
##   poll_id = col_double(),
##   cycle = col_double(),
##   pollster_id = col_double(),
##   sponsor_ids = col_number(),
##   pollster_rating_id = col_double(),
##   sample_size = col_double(),
##   sponsor_candidate = col_logical(),
##   internal = col_logical(),
##   partisan = col_logical(),
##   tracking = col_logical(),
##   nationwide_batch = col_logical(),
##   candidate_id = col_double(),
##   pct = col_double()
## )
## i Use `spec()` for the full column specifications.
```

```
## Warning: 122 parsing failures.
##  row      col           expected actual
## 1394 partisan 1/0/T/F/TRUE/FALSE    DEM 'https://jmontgomery.github.io/PDS/Datasets/president_primar
## 1395 partisan 1/0/T/F/TRUE/FALSE    DEM 'https://jmontgomery.github.io/PDS/Datasets/president_primar
## 1396 partisan 1/0/T/F/TRUE/FALSE    DEM 'https://jmontgomery.github.io/PDS/Datasets/president_primar
## 1397 partisan 1/0/T/F/TRUE/FALSE    DEM 'https://jmontgomery.github.io/PDS/Datasets/president_primar
## 1398 partisan 1/0/T/F/TRUE/FALSE    DEM 'https://jmontgomery.github.io/PDS/Datasets/president_primar
## .... ........ .................. ...... .............................................................
## See problems(...) for more details.
```

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

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

```r
Endorsements <- Endorsements %>%
    rename(candidate_name = endorsee)
```

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

```r
Endorsements <- as_tibble(Endorsements)
```

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: \texttt{candidate_name, sample_size, start_date, party, pct.

```r
polls <- polls %>%
    filter(candidate_name %in% c("Amy Klobuchar", "Bernard Sanders", "Elizabeth Warren",
        "Joseph R. Biden Jr.", "Michael Bloomberg", "Pete Buttigieg"))
# check it worked
table(polls$candidate_name)
```

```
##
```

```
##       Amy Klobuchar     Bernard Sanders    Elizabeth Warren Joseph R. Biden Jr.
##                 874                 960                 963                 971
##   Michael Bloomberg      Pete Buttigieg
##                 232                 890
```

```r
# subset the dataset
library(dplyr)
polls <- polls %>%
    select(candidate_name, sample_size, start_date, party, pct)
# check it worked
glimpse(polls)
```

```
## Rows: 4,890
## Columns: 5
## $ candidate_name <chr> "Bernard Sanders", "Pete Buttigieg", "Joseph R. Biden J~
## $ sample_size    <dbl> 1295, 1295, 1295, 1295, 1295, 556, 556, 556, 556, 556, ~
## $ start_date     <chr> "2/7/20", "2/7/20", "2/7/20", "2/7/20", "2/7/20", "2/6/~
## $ party          <chr> "DEM", "DEM", "DEM", "DEM", "DEM", "DEM", "DEM", "DEM",~
## $ pct            <dbl> 28.0, 26.0, 9.0, 13.0, 14.0, 20.0, 17.0, 15.0, 11.0, 8.~
```

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
table(Endorsements$candidate_name)
```

```
## 
##      Amy Klobuchar     Bernie Sanders      Beto O'Rourke        Cory Booker
##                 10                 16                  7                 20
##   Elizabeth Warren      Eric Swalwell         Jay Inslee          Joe Biden
##                  9                  1                  4                 29
##       John Delaney  John Hickenlooper      Julian Castro      Kamala Harris
##                  2                  1                  4                 31
## Kirsten Gillibrand     Pete Buttigieg      Steve Bullock
##                  1                  5                  3
```

```r
# change Bernie Sanders to Bernard Sanders
Endorsements <- Endorsements %>%
    mutate(candidate_name = ifelse(candidate_name == "Bernie Sanders", "Bernard Sanders",
        candidate_name))
# change Joe Biden to Joseph R Biden Jr
Endorsements <- Endorsements %>%
    mutate(candidate_name = ifelse(candidate_name == "Joe Biden", "Joseph R. Biden Jr.",
        candidate_name))
# check it worked
table(Endorsements$candidate_name)
```

```
## 
##       Amy Klobuchar     Bernard Sanders       Beto O'Rourke         Cory Booker
##                  10                  16                   7                  20
##    Elizabeth Warren       Eric Swalwell          Jay Inslee        John Delaney
##                   9                   1                   4                   2
##   John Hickenlooper Joseph R. Biden Jr.       Julian Castro       Kamala Harris
##                   1                  29                   4                  31
##  Kirsten Gillibrand      Pete Buttigieg       Steve Bullock
##                   1                   5                   3
```

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

```
# use inner join since we are supposed to end up with only 5 candidates after
# joining and endorsements doesn't have bloomberg
polls_endorse <- inner_join(polls, Endorsements, by = "candidate_name")
table(polls_endorse$candidate_name)
```

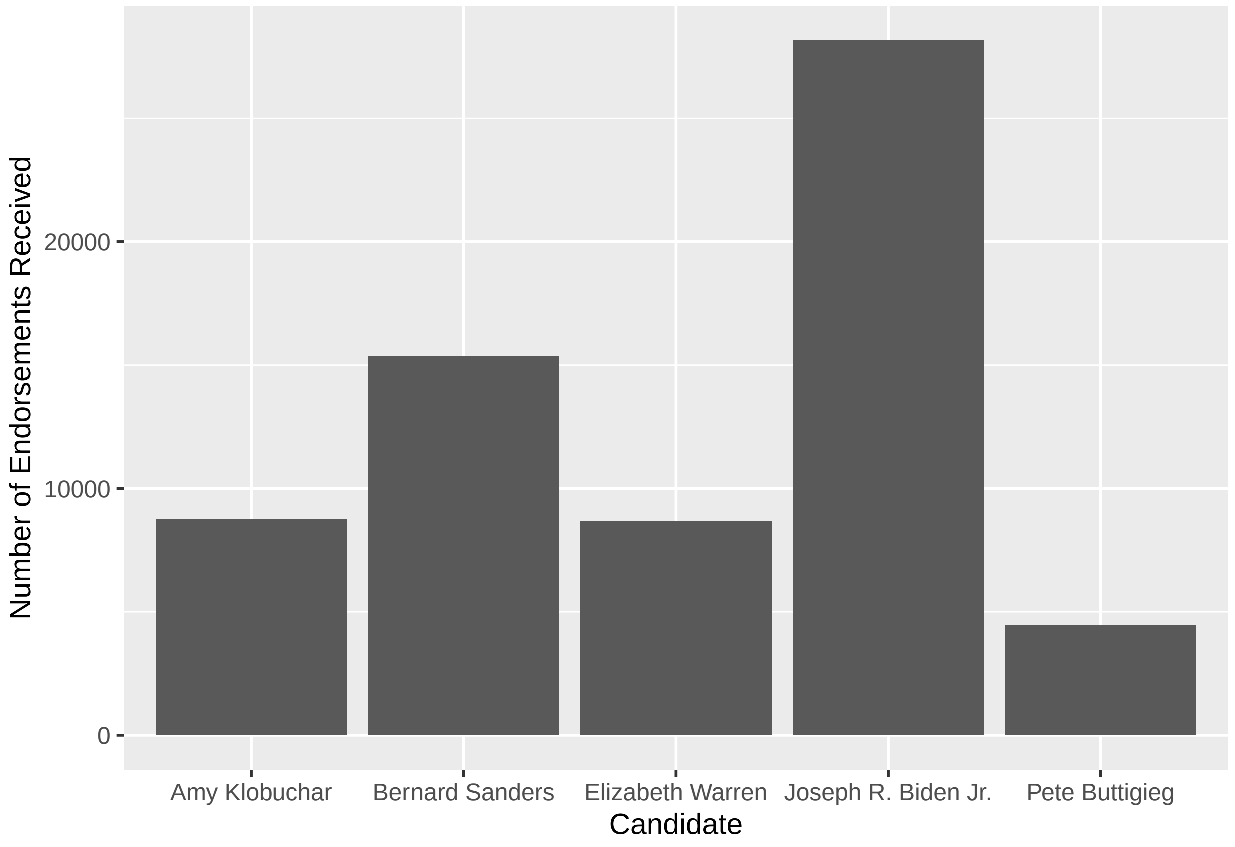
```
##
##       Amy Klobuchar     Bernard Sanders    Elizabeth Warren Joseph R. Biden Jr.
##                8740               15360                8667               28159
##      Pete Buttigieg
##                4450
```

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

```
# should be able to do this just by counting how many times the candidate name
# appears in the dataset
n_endorse <- polls_endorse %>%
    count(candidate_name)
```

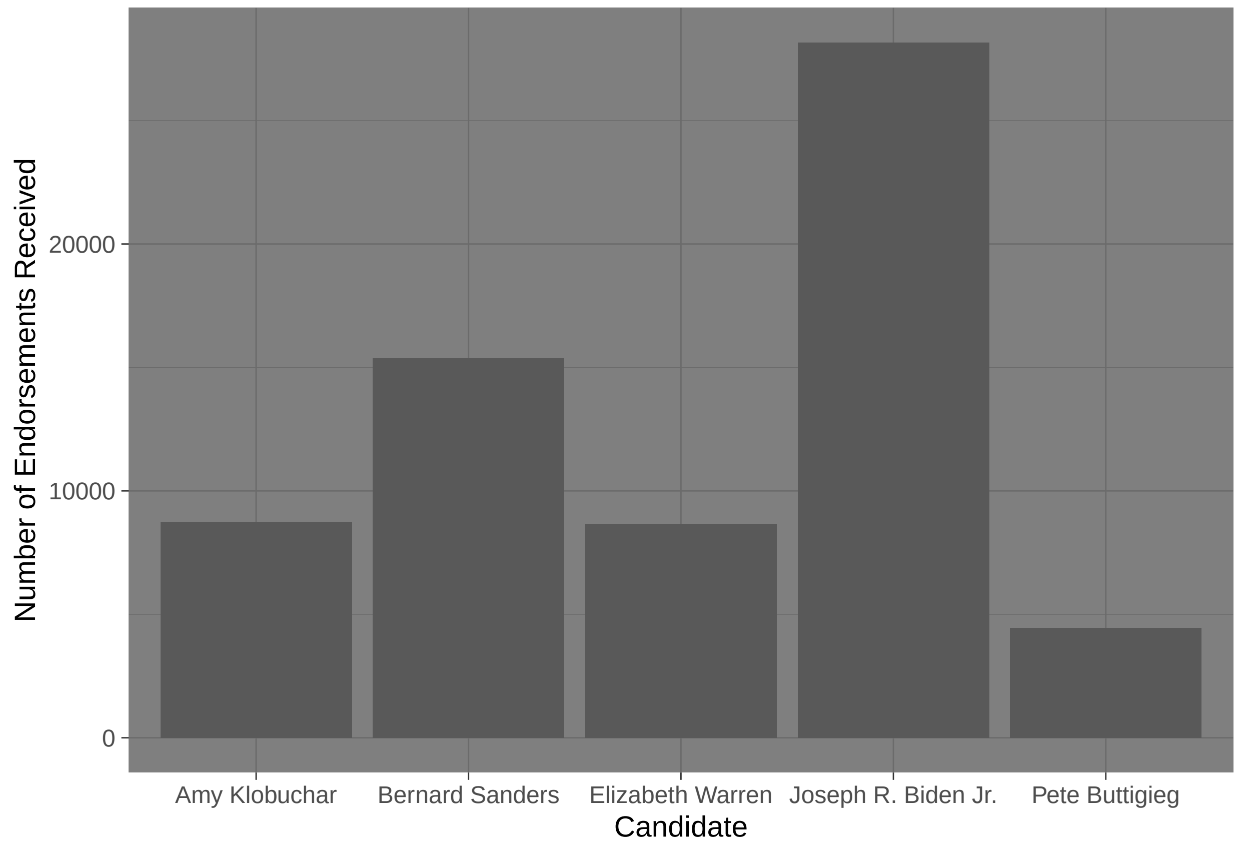
Plot the number of endorsement each of the 5 candidates have using `ggplot()`. Save your plot as an object `p`.

```
library(ggplot2)
p <- n_endorse %>%
    ggplot() + geom_bar(aes(x = candidate_name, y = n), stat = "identity") + labs(x = "Candidate",
    y = "Number of Endorsements Received")
p
```



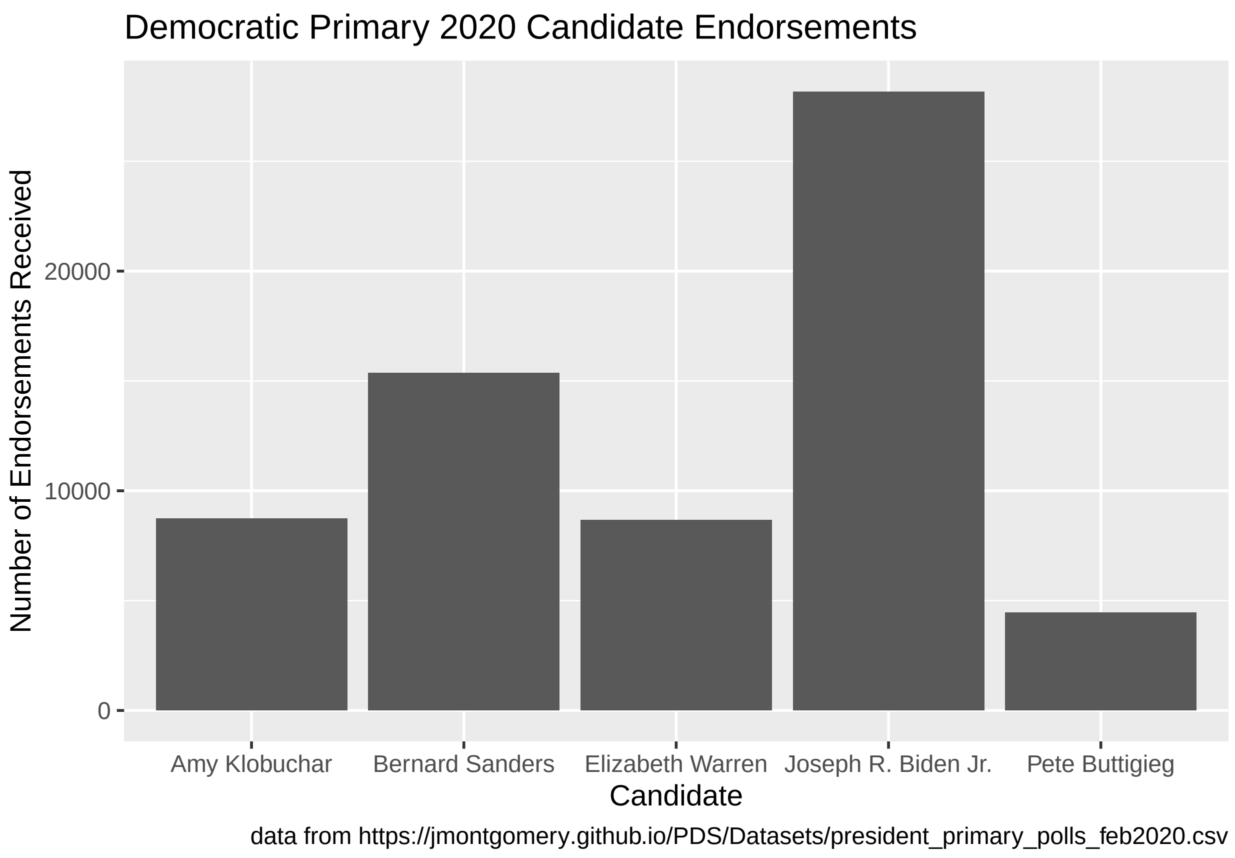
Rerun the previous line as follows: `p + theme_dark()`. Notice how you can still customize your plot without rerunning the plot with new options.

```r
p + theme_dark()
```



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.

```r
# i already have x and y axes so just add title
p + labs(title = "Democratic Primary 2020 Candidate Endorsements", caption = "data from https://jmontgom
```



```r
# Save the plot in your forked repository.
ggsave("p.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)
```

```
## Loading required package: NLP

##
## Attaching package: 'NLP'

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

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

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

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

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

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

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

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   source = col_character(),
##   text = col_character(),
##   created_at = col_character(),
##   retweet_count = col_double(),
##   favorite_count = col_double(),
##   is_retweet = col_logical()
## )
```

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.

```
library(stringr)
tweets[c("date", "time")] <- str_split_fixed(tweets$created_at, " ", 2)
# make dates useable - Ryan did this in inclass14.rmd
tweets$date <- as.Date(tweets$date, format = "%m/%d/%y")
# report the range of dates that is in this dataset
range(tweets$date)
```

```
## [1] "2020-01-01" "2020-12-31"
```

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.)

```r
library(dplyr)
tweets_og <- tweets %>%
    filter(is_retweet == F)
# this got rid of 2k obs

# top 5 retweets
top_retwts <- tweets_og %>%
    top_n(5, retweet_count)
top_retwts$text
```

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

```r
# 5 most popular
top_fav <- tweets_og %>%
    top_n(5, favorite_count)
top_fav$text
```

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

```r
# (Hint: The \texttt{match} function can help you find the index once you
# identify the largest values.)  select tweets from og tweets dataset that
# match rows from top 5 retweet and favorite vectors
top_tweets <- tweets_og %>%
    filter(retweet_count %in% top_retwts$retweet_count | favorite_count %in% top_fav$favorite_count) %>%
    select(text)
```

Create a *corpus* of the tweet content and put this into the object `Corpus` using the `tm` (text mining) package.

```r
# vignette('tm')
tweets_corpus <- VCorpus(VectorSource(tweets_og$text))
# VectorSource= a vector of characters (treats each component as a document)
tweets_corpus
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 30199
```

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

```r
# Create a function called 'addspace' that finds a user specified pattern and
# substitutes the pattern with a space.
addspace <- content_transformer(function(x, pattern) {
    return(gsub(pattern, " ", x))
})
library(tm)
```

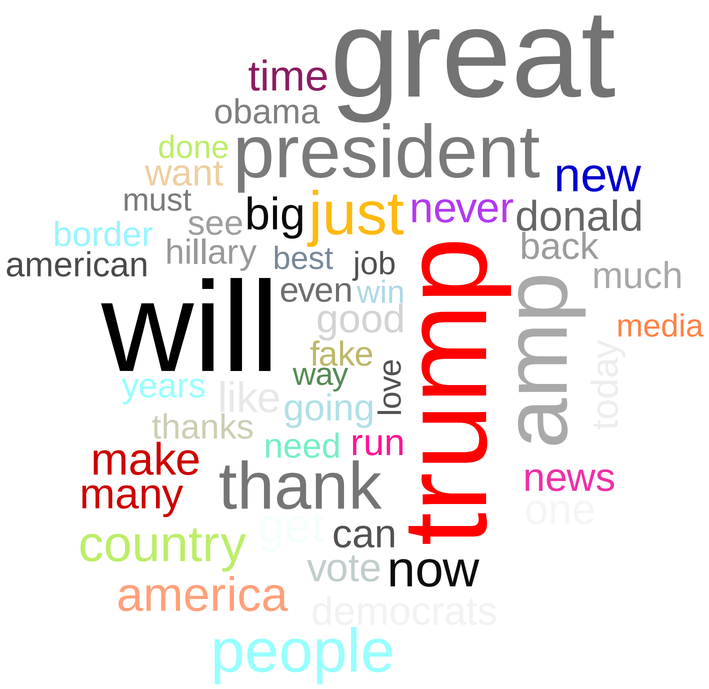
```r
tweets_corpus <- tm_map(tweets_corpus, addspace, "-")

tweets_corpus <- tweets_corpus %>%
    # Remove extraneous whitespace
tm_map(stripWhitespace) %>%
    # remove numbers and punctuation
tm_map(removePunctuation) %>%
    tm_map(removeNumbers) %>%
    # convert everything to lower case
tm_map(content_transformer(tolower)) %>%
    # remove 'stop words' that have little substantive meaning (the, a, it).
tm_map(removeWords, stopwords("english")) %>%
    # remove self mention
tm_map(removeWords, "realdonaldtrump")
# check it worked
writeLines(head(strwrap(tweets_corpus[[2]]), 15))
```

```
## actually military legal obligation incredible wife karen lot respect
## pulled aside amp said strongly "john respects greatly longer will speak
## well " wrong
```

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
cloud <- wordcloud(tweets_corpus, min.freq = 3, random.order = T, random.color = T,
    max.words = 50, colors = colours())
```



```r
# Save the plot into your forked repository.
ggsave("cloud.png")
```

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

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

```
weightTfIdf)
```

```r
# do it for top tweets
toptweets_corpus <- VCorpus(VectorSource(top_tweets$text))
toptweets_corpus <- tm_map(toptweets_corpus, addspace, "-")
toptweets_corpus <- toptweets_corpus %>%
    # Remove extraneous whitespace
tm_map(stripWhitespace) %>%
    # remove numbers and punctuation
tm_map(removePunctuation) %>%
    tm_map(removeNumbers) %>%
    # convert everything to lower case
tm_map(content_transformer(tolower)) %>%
    # remove 'stop words' that have little substantive meaning (the, a, it).
tm_map(removeWords, stopwords("english"))

# create DTM
DTM <- DocumentTermMatrix(toptweets_corpus, control = list(weighting = weightTfIdf))
inspect(DTM)
```

```
## <<DocumentTermMatrix (documents: 9, terms: 97)>>
## Non-/sparse entries: 103/770
## Sparsity           : 88%
## Maximal term length: 18
## Weighting          : term frequency - inverse document frequency (normalized) (tf-idf)
## Sample             :
##     Terms
## Docs   america christmas      fnn fraudnewscnn      great httpstcovxekivzptf
##    1 0.0000000  0.000000 0.000000     0.000000 0.08036759           0.000000
##    2 0.0000000  0.000000 0.000000     0.000000 0.00000000           0.000000
##    3 0.0000000  0.000000 0.000000     0.000000 0.00000000           3.169925
##    4 0.0000000  1.584963 0.000000     0.000000 0.00000000           0.000000
##    5 0.0000000  0.000000 0.000000     0.000000 0.00000000           0.000000
##    6 0.0000000  0.000000 0.000000     0.000000 0.00000000           0.000000
##    7 0.0000000  0.000000 1.056642     1.056642 0.00000000           0.000000
##    8 0.0000000  0.000000 0.000000     0.000000 0.00000000           0.000000
##    9 0.7924813  0.000000 0.000000     0.000000 0.54248125           0.000000
##     Terms
## Docs httpstcowyunhjjujg      make    merry     today
##    1           0.000000 0.0000000 0.000000 0.0000000
##    2           0.000000 0.0000000 0.000000 0.0000000
##    3           0.000000 0.0000000 0.000000 0.0000000
##    4           0.000000 0.0000000 1.584963 0.0000000
##    5           0.000000 0.0000000 0.000000 0.0000000
##    6           0.000000 0.0000000 0.000000 0.0000000
##    7           1.056642 0.0000000 0.000000 0.0000000
##    8           0.000000 0.0000000 0.000000 0.0000000
##    9           0.000000 0.7924813 0.000000 0.7924813
```

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

```r
# create dictionary
top_words <- findFreqTerms(DTM, lowfreq = 0.8)
# restrict dimension
top_dtm <- DocumentTermMatrix(toptweets_corpus, list(dictionary = top_words))
# top 50
```

```r
top_dtm <- as.matrix(top_dtm)
colnames(top_dtm)
```

```
## [1] "christmas"          "fnn"                "fraudnewscnn"
## [4] "httpstcovxekivzptf" "httpstcowyunhjjujg" "merry"
```

```r
# there are only six terms left so these are the top words. Some of these are
# obviously the parts to links so I'm not sure what happened there

# try without cleaning
top_tweets_corpus <- VCorpus(VectorSource(top_tweets$text))
DTM2 <- DocumentTermMatrix(top_tweets_corpus, control = list(weighting = weightTfIdf))
inspect(DTM2)
```

```
## <<DocumentTermMatrix (documents: 9, terms: 122)>>
## Non-/sparse entries: 142/956
## Sparsity           : 87%
## Maximal term length: 23
## Weighting          : term frequency - inverse document frequency (normalized) (tf-idf)
## Sample             :
##     Terms
## Docs     #fnn #fraudnewscnn   again!  america christmas!
##    1 0.000000      0.000000 0.000000 0.000000   0.000000
##    2 0.000000      0.000000 0.000000 0.000000   0.000000
##    3 0.000000      0.000000 0.000000 0.000000   0.000000
##    4 0.000000      0.000000 0.000000 0.000000   1.584963
##    5 0.000000      0.000000 0.000000 0.000000   0.000000
##    6 0.000000      0.000000 0.000000 0.000000   0.000000
##    7 1.056642      1.056642 0.000000 0.000000   0.000000
##    8 0.000000      0.000000 0.000000 0.000000   0.000000
##    9 0.000000      0.000000 0.633985 0.633985   0.000000
##     Terms
## Docs https://t.co/vxekivzptf https://t.co/wyunhjjujg     make    merry    today
##    1                0.000000                0.000000 0.000000 0.000000 0.000000
##    2                0.000000                0.000000 0.000000 0.000000 0.000000
##    3                3.169925                0.000000 0.000000 0.000000 0.000000
##    4                0.000000                0.000000 0.000000 1.584963 0.000000
##    5                0.000000                0.000000 0.000000 0.000000 0.000000
##    6                0.000000                0.000000 0.000000 0.000000 0.000000
##    7                0.000000                1.056642 0.000000 0.000000 0.000000
##    8                0.000000                0.000000 0.000000 0.000000 0.000000
##    9                0.000000                0.000000 0.633985 0.000000 0.633985
```

```r
# create dictionary
top_words2 <- findFreqTerms(DTM2, lowfreq = 0.8)
# restrict dimension
top_dtm2 <- DocumentTermMatrix(top_tweets_corpus, list(dictionary = top_words2))
# top 50
top_dtm2 <- as.matrix(top_dtm2)
colnames(top_dtm2)
```

```
## [1] "#fnn"                    "#fraudnewscnn"
## [3] "christmas!"              "https://t.co/vxekivzptf"
## [5] "https://t.co/wyunhjjujg" "merry"
```

```
# still six terms, but now the websites are more recognizable
```