# 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.

```
library(fivethirtyeight)
```

```
## Warning: package 'fivethirtyeight' was built under R version 4.1.3
```

```
## 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)
```

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

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
```

```
## -- 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).
- 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.
- 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.

```
Endorsements <- Endorsements %>%
    rename(candidate_name = endorsee)
# FINISHED!

Endorsements <- as_tibble(Endorsements)
class(Endorsements)
```

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

```
# FINISHED!

polls <- 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)
# FINISHED!

# Compare candidate names.
eucn <- summarize(Endorsements, endorsements_unique_candidate_names = unique(candidate_name))
# eucn

pucn <- summarize(polls, polls_unique_candidate_names = unique(candidate_name))
# pucn Same names (FORMAT - candidate_name candidate_name): Bernard Sanders
# Bernie Sanders Pete Buttigieg\t\t\t Pete Buttigieg (same) Joseph R. Biden Jr.
# Joe Biden Amy Klobuchar\t\t\t Amy Klobuchar (same) Elizabeth Warren\t\t
# Elizabeth Warren (same) Michael Bloomberg NONE (not in Endorsements dataset)
rm(eucn, pucn)
# Change candidate names so they are equivalent.
polls <- polls %>%
    mutate(candidate_name = replace(candidate_name, candidate_name == "Bernard Sanders",
        "Bernie Sanders")) %>%
    mutate(candidate_name = replace(candidate_name, candidate_name == "Joseph R. Biden Jr.",
        "Joe Biden"))
# FINISHED!

# No shared variables besides candidate_name.  merge (use inner_join to keep
# only those ones that match)
combined <- inner_join(polls, Endorsements, by = "candidate_name")
dim(combined)
```

```
## [1] 65376    17
```

```
summarize(combined, endorsements_unique_candidate_names = unique(candidate_name))
```

```
## # A tibble: 5 x 1
##   endorsements_unique_candidate_names
##   <chr>
## 1 Bernie Sanders
## 2 Pete Buttigieg
## 3 Joe Biden
## 4 Amy Klobuchar
## 5 Elizabeth Warren
```

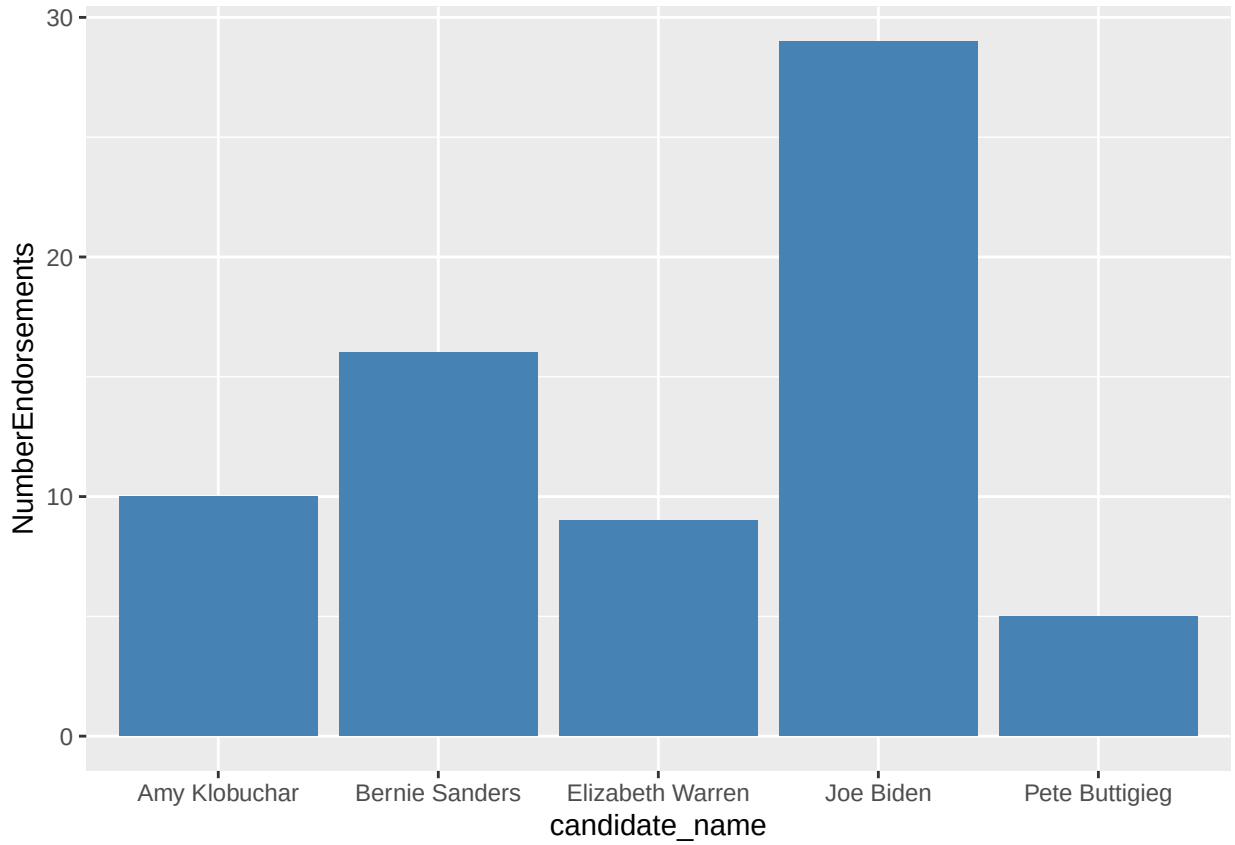
```r
# Only 5 unique candidates left.  FINISHED!

NumberEndorsements <- Endorsements %>%
    group_by(candidate_name) %>%
    mutate(NumberEndorsements = n()) %>%
    select(candidate_name, NumberEndorsements) %>%
    unique() %>%
    filter(candidate_name %in% c("Amy Klobuchar", "Bernie Sanders", "Elizabeth Warren",
        "Joe Biden", "Pete Buttigieg"))
NumberEndorsements
```

```
## # A tibble: 5 x 2
## # Groups:   candidate_name [5]
##   candidate_name   NumberEndorsements
##   <chr>                         <int>
## 1 Joe Biden                        29
## 2 Bernie Sanders                   16
## 3 Amy Klobuchar                    10
## 4 Elizabeth Warren                  9
## 5 Pete Buttigieg                    5
```

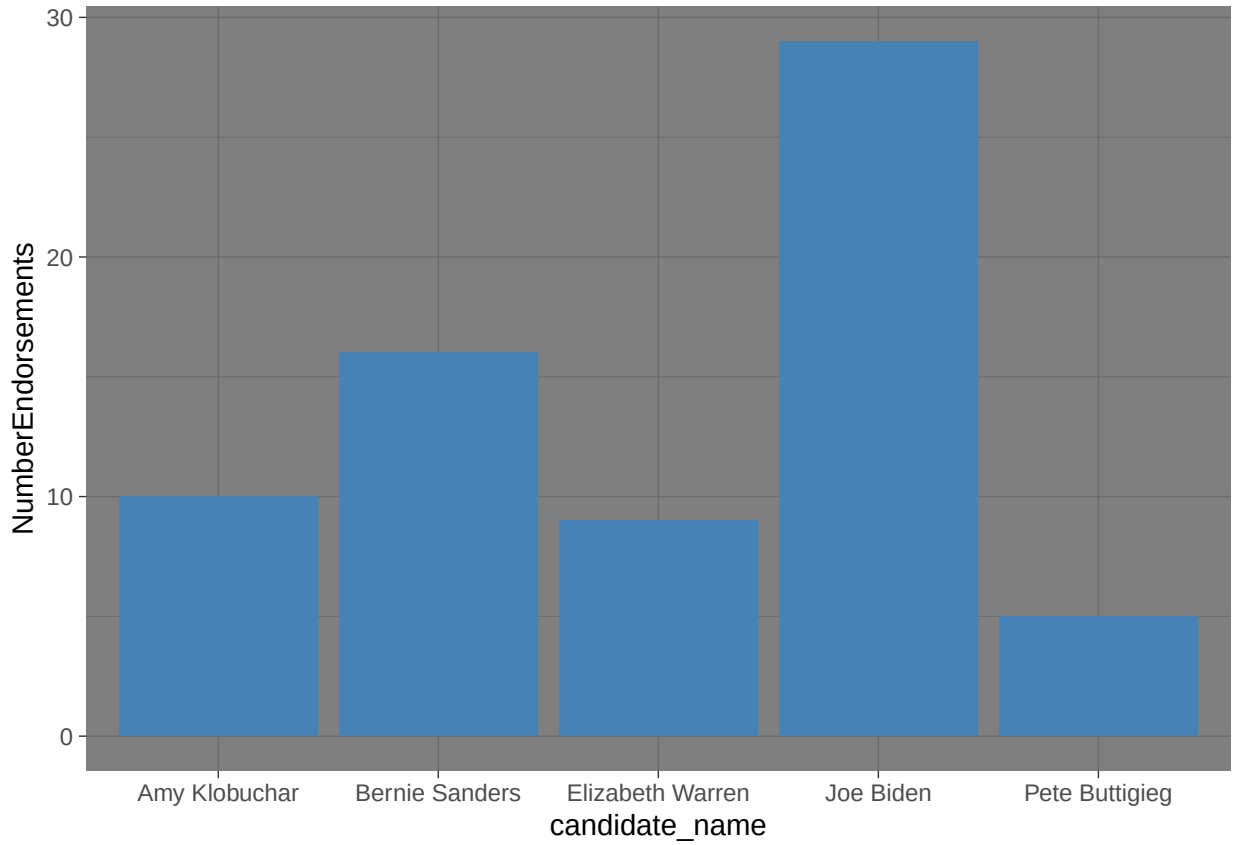
```r
# FINISHED!

library(ggplot2)
p <- ggplot(data = NumberEndorsements, mapping = aes(x = candidate_name, y = NumberEndorsements)) +
    geom_bar(stat = "identity", fill = "steelblue")
p
```



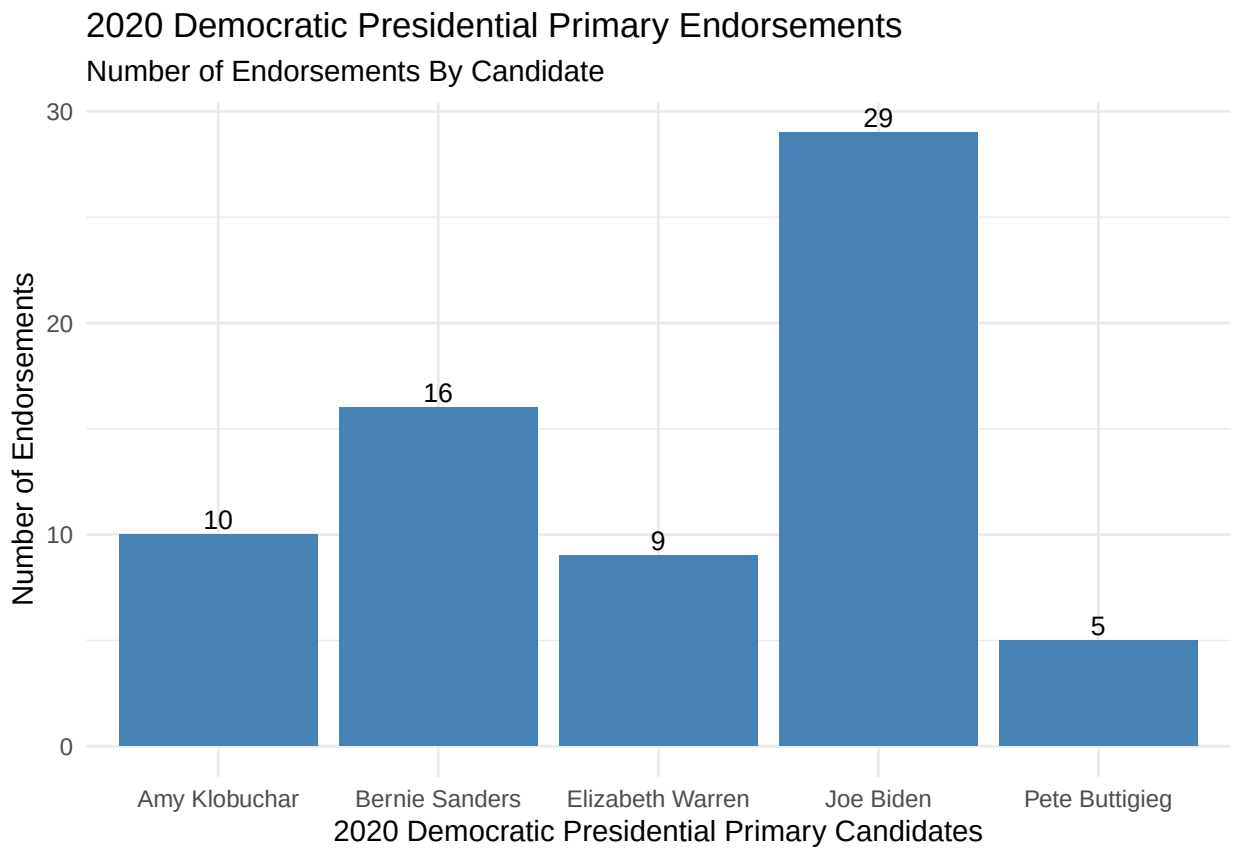

```
# FINISHED!

p + theme_dark()
```




```
# FINISHED!

p <- p + geom_text(aes(label = NumberEndorsements), vjust = -0.3, size = 3.5) + labs(title = "2020 Demo
    subtitle = "Number of Endorsements By Candidate") + ylab("Number of Endorsements") +
    xlab("2020 Democratic Presidential Primary Candidates") + theme_minimal()
p
```




```
ggsave("PS4_p_plot.png")
```

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

```
# FINISHED!

ls()
```

```
## [1] "combined"           "Endorsements"       "NumberEndorsements"
## [4] "p"                  "polls"              "url"
```

```
rm(combined, Endorsements, NumberEndorsements, p, polls, url)
```

## 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 4.1.3
```

```
## 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)
```

```
## Warning: package 'wordcloud' was built under R version 4.1.3
```

```
## 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.

```
# Investigate Data:
colnames(tweets)
```

```
## [1] "source"         "text"           "created_at"     "retweet_count"
## [5] "favorite_count" "is_retweet"
```

```
tail(tweets$created_at)
```

```
## [1] "1/2/2014 6:03"  "1/2/2014 6:02"  "1/2/2014 6:00"  "1/2/2014 5:47"
## [5] "1/2/2014 0:39"  "1/1/2014 12:56"
```

```
class(tweets$created_at)
```

```
## [1] "character"
```

```
# Separate 'created_at' variable
tweets <- tweets %>%
    separate(created_at, c("date", "time"), " ")

# Report the range of dates in dataset:
tweets <- tweets %>%
    mutate(date = as.Date(date, "%m/%d/%Y"))
class(tweets$date)
```

```
## [1] "Date"
```

```
summary(tweets$date)
```

```
##         Min.      1st Qu.       Median         Mean      3rd Qu.         Max.
## "2014-01-01" "2015-04-17" "2016-09-11" "2017-02-09" "2019-03-29" "2020-02-14"
```

```
# Earliest tweet is January 1st, 2014.  Latest tweet is February 14, 2020.

# FINISHED!
```

Using `dplyr` subset the data to only include original tweets (remove retweets) 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 subset of 'tweets' dataset that includes only original tweets.
subtweets <- tweets %>%
    filter(is_retweet == FALSE)

class(subtweets$favorite_count)
```

```
## [1] "numeric"
```

```
class(subtweets$retweet_count)
```

```
## [1] "numeric"
```

```
colnames(subtweets)
```

```
## [1] "source"         "text"           "date"           "time"
## [5] "retweet_count"  "favorite_count" "is_retweet"
```

```
# Find top 5 favorited tweets:
top5fav <- subtweets %>%
    arrange(desc(favorite_count)) %>%
    slice(1:5)
top5fav
```

```
## # A tibble: 5 x 7
##   source          text  date       time  retweet_count favorite_count is_retweet
##   <chr>           <chr> <date>     <chr>         <dbl>          <dbl> <lgl>
## 1 Twitter for iP~ A$AP~ 2019-08-02 17:41        251530         879647 FALSE
## 2 Twitter for iP~ http~ 2020-01-03 2:32         172157         814012 FALSE
## 3 Twitter for iP~ All ~ 2020-01-08 2:45         158004         764333 FALSE
## 4 Twitter for iP~ MERR~ 2019-12-25 12:26        115372         735775 FALSE
## 5 Twitter for iP~ Kobe~ 2020-01-26 23:54         94246         735478 FALSE
```

```
# Find text of top 5 favorited tweets:
top5fav_text <- top5fav %>%
    select(text)

# Find top 5 retweeted tweets:
top5retweets <- subtweets %>%
    arrange(desc(retweet_count)) %>%
    slice(1:5)

# Find text of top 5 retweeted tweets:
top5retweets_text <- top5retweets %>%
    select(text)

# Create one item out of both:
t5favretweets_text <- rbind(top5fav_text, top5retweets_text)

# Turn into character vector:
t5favretweets_text <- t5favretweets_text %>%
    pull(text)
t5favretweets_text
```

```
##  [1] "A$AP Rocky released from prison and on his way home to the United States from Sweden. It was a
##  [2] "https://t.co/VXeKiVzpTf"
##  [3] "All is well! Missiles launched from Iran at two military bases located in Iraq. Assessment of
##  [4] "MERRY CHRISTMAS!"
##  [5] "Kobe Bryant despite being one of the truly great basketball players of all time was just gettin
##  [6] "#FraudNewsCNN #FNN https://t.co/WYUnHjjUjg"
##  [7] "TODAY WE MAKE AMERICA GREAT AGAIN!"
##  [8] "Why would Kim Jong-un insult me by calling me \"old\" when I would NEVER call him \"short and
##  [9] "A$AP Rocky released from prison and on his way home to the United States from Sweden. It was a
## [10] "Such a beautiful and important evening! The forgotten man and woman will never be forgotten aga
```

```
class(t5favretweets_text)
```

```
## [1] "character"
```

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.)

```r
# vignette('tm')

# Create Corpus object:
Corpus <- VCorpus(VectorSource(t5favretweets_text))
inspect(Corpus[[1]])
```

```
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 125
##
## A$AP Rocky released from prison and on his way home to the United States from Sweden. It was a Rocky
```

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
# Remove extraneous whitespace:
Corpus <- tm_map(Corpus, stripWhitespace)

# Remove numbers:
Corpus <- tm_map(Corpus, content_transformer(removeNumbers))

# Remove punctuation:
Corpus <- tm_map(Corpus, content_transformer(removePunctuation))

# Convert everything to lower case:
Corpus <- tm_map(Corpus, content_transformer(tolower))

# Remove 'stop words' with little substantive meaning:
Corpus <- tm_map(Corpus, removeWords, stopwords("english"))

inspect(Corpus[[2]])
```

```
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 18
##
## httpstcovxekivzptf
```

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
# To create a word matrix, I need to create a document term matrix (as is
# requested in the next section).
doctermat <- TermDocumentMatrix(Corpus)
doctermat <- as.matrix(doctermat)
doctermat <- sort(rowSums(doctermat), decreasing = TRUE)
doctermat <- data.frame(word = names(doctermat), freq = doctermat)
doctermat <- doctermat %>%
    filter(freq >= 3)
doctermat
```

```
##        word freq
## aap     aap    4
## home   home    4
## rocky rocky    4
## will   will    4
## never never    3
## well   well    3
```

```
# Huh? There are only 6 words with a frequency greater than three. I think we
# need more.  I'm going to use the top 100 most-favorited tweets instead.

# Redoing all the above
subtweets <- tweets %>%
    filter(is_retweet == FALSE)
# Find top 100 favorited tweets:
top100fav <- subtweets %>%
    arrange(desc(favorite_count)) %>%
    slice(1:100)
# Find text of top 100 favorited tweets:
top100fav_text <- top100fav %>%
    select(text)
# Turn into character vector:
top100fav_text <- top100fav_text %>%
    pull(text)
# Create Corpus object:
Corpus <- VCorpus(VectorSource(top100fav_text))
# Remove extraneous whitespace:
Corpus <- tm_map(Corpus, stripWhitespace)
# Remove numbers:
Corpus <- tm_map(Corpus, content_transformer(removeNumbers))
# Remove punctuation:
Corpus <- tm_map(Corpus, content_transformer(removePunctuation))
# Convert everything to lower case:
Corpus <- tm_map(Corpus, content_transformer(tolower))
# Remove 'stop words' with little substantive meaning:
Corpus <- tm_map(Corpus, removeWords, stopwords("english"))
# Create a document term matrix:
doctermat <- TermDocumentMatrix(Corpus)
doctermat <- as.matrix(doctermat)
doctermat <- sort(rowSums(doctermat), decreasing = TRUE)
doctermat <- data.frame(word = names(doctermat), freq = doctermat)
doctermat <- doctermat %>%
    filter(freq >= 3)

# Some last fixes!
doctermat <- doctermat %>%
    filter(word != "amp") %>%
    mutate(word = replace(word, word == "aap", "a$ap"))
doctermat <- head(doctermat, 50)
doctermat
```

```
##                          word freq
## great                   great   28
## will                     will   24
```

```
## just                       just   15
## america                  america   12
## country                  country   12
## iran                        iran   10
## never                      never   10
## states                    states   10
## make                        make    9
## united                    united    9
## usa                          usa    9
## american                american    8
## happy                      happy    8
## people                    people    8
## thank                      thank    8
## back                        back    6
## new                          new    6
## president              president    6
## time                        time    6
## wonderful              wonderful    6
## aap                          a$ap    5
## iranian                  iranian    5
## military                military    5
## world                      world    5
## year                        year    5
## beautiful              beautiful    4
## call                        call    4
## can                          can    4
## christmas              christmas    4
## even                        even    4
## good                        good    4
## hard                        hard    4
## hit                          hit    4
## killed                    killed    4
## making                    making    4
## many                        many    4
## now                          now    4
## quickly                  quickly    4
## rocky                      rocky    4
## sweden                    sweden    4
## total                      total    4
## way                          way    4
## well                        well    4
## work                        work    4
## basketball           basketball    3
## big                          big    3
## bryant                    bryant    3
## congratulations congratulations    3
## democrats             democrats    3
## done                        done    3
```

```
## add in color: library(RColorBrewer)
color_options <- colors() %>%
    sample(size = 50, replace = FALSE)
# colors
```

```
# Save:
png("./probset4_wordcloud.png")

# Create wordcloud: ?wordcloud()
wordcloud(doctermat$word, doctermat$freq, random.color = TRUE, colors = color_options,
    random.order = FALSE)
```

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

```
doctermat <- TermDocumentMatrix(Corpus, control = list(weighting = weightTfIdf))
```

```
## Warning in weighting(x): empty document(s): 27 75
```

```
doctermat
```

```
## <<TermDocumentMatrix (terms: 641, documents: 100)>>
## Non-/sparse entries: 1033/63067
## Sparsity           : 98%
## Maximal term length: 18
## Weighting          : term frequency - inverse document frequency (normalized) (tf-idf)
```

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

```
# This is largely a repeat of the section before the wordcloud:
doctermat2 <- as.matrix(doctermat)
doctermat2 <- sort(rowSums(doctermat2), decreasing = TRUE)
doctermat2 <- data.frame(word = names(doctermat2), weightTfIdf = doctermat2)
doctermat2 <- doctermat2 %>%
    filter(weightTfIdf >= 0.8)
doctermat2 <- head(doctermat2, 50)
doctermat2
```

```
##                                 word weightTfIdf
## httpstcojdszuxxjg   httpstcojdszuxxjg   11.287712
## great                         great    6.930663
## boring                       boring    6.643856
## httpstcoacyhhstm     httpstcoacyhhstm    6.643856
## httpstcoatpwub         httpstcoatpwub    6.643856
## httpstcodutxclyzw   httpstcodutxclyzw    6.643856
## httpstcoisfaokoip   httpstcoisfaokoip    6.643856
## httpstconzkwoctu     httpstconzkwoctu    6.643856
## httpstcovxekivzptf httpstcovxekivzptf    6.643856
## make                           make    6.543385
## happy                         happy    6.517126
## america                     america    6.510013
## usa                             usa    5.605133
## thank                         thank    5.493389
## christmas                 christmas    5.420243
## merry                         merry    5.239568
## never                         never    4.174653
```

```
## will                              will    4.087601
## year                              year    3.840112
## just                              just    3.729302
## new                                new    3.482562
## building                      building    3.321928
## july                              july    3.321928
## country                        country    3.016619
## wall                              wall    3.010057
## american                      american    2.478235
## morning                        morning    2.445132
## iran                              iran    2.307708
## draining                      draining    2.214619
## fnn                                fnn    2.214619
## fraudnewscnn              fraudnewscnn    2.214619
## greenland                    greenland    2.214619
## hero                              hero    2.214619
## httpstcoddyvuha       httpstcoddyvuha    2.214619
## httpstcoqqytqgktt   httpstcoqqytqgktt    2.214619
## httpstcosgznhindw   httpstcosgznhindw    2.214619
## httpstcouwxkrokx     httpstcouwxkrokx    2.214619
## httpstcowyunhjjujg httpstcowyunhjjujg    2.214619
## httpstcoxccasgfsz   httpstcoxccasgfsz    2.214619
## httpstcozvzeliqsi   httpstcozvzeliqsi    2.214619
## japan                            japan    2.214619
## look                              look    2.214619
## photograph                  photograph    2.214619
## promise                        promise    2.214619
## swamp                            swamp    2.214619
## thanksgiving              thanksgiving    2.214619
## today                            today    2.046775
## back                              back    2.027905
## states                          states    1.859814
## joe                                joe    1.814097
```