Applied Statistical Programming - Spring 2022

Alma Velazquez

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.

```
# Change eval=FALSE in the code block. Install packages as appropriate.
#install.packages("fivethirtyeight")
library(fivethirtyeight)
library(tidyverse)
# URL to the data that you've used.
url <- 'https://jmontgomery.github.io/PDS/Datasets/president_primary_polls_feb2020.csv'
polls <- read_csv(url)
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.

```
# Check whether Endorsements is already tibble - this should already be the case
is_tibble(Endorsements)
```

```
## [1] TRUE
```

```
# Rename endorsee variable
Endorsements <- Endorsements %>%
  rename(candidate_name = endorsee)
```

- 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`

```
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)

# Check that it worked
unique(polls$candidate_name)
```

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

- 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.

```
unique(polls$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
```

```
Endorsements <- Endorsements %>%
  mutate(candidate_name = case_when(
    grepl("biden", candidate_name, ignore.case = TRUE) ~ "Joseph R. Biden Jr.",
    grepl("sanders", candidate_name, ignore.case = TRUE) ~ "Bernard Sanders",
    TRUE ~ candidate_name
  ))


unique(Endorsements$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
```

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

```
length(unique(polls$candidate_name))
```

```
## [1] 6
```

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

```
## [1] 16
```

```
polls <- polls %>%
  inner_join(Endorsements, by="candidate_name")

length(unique(polls$candidate_name))
```

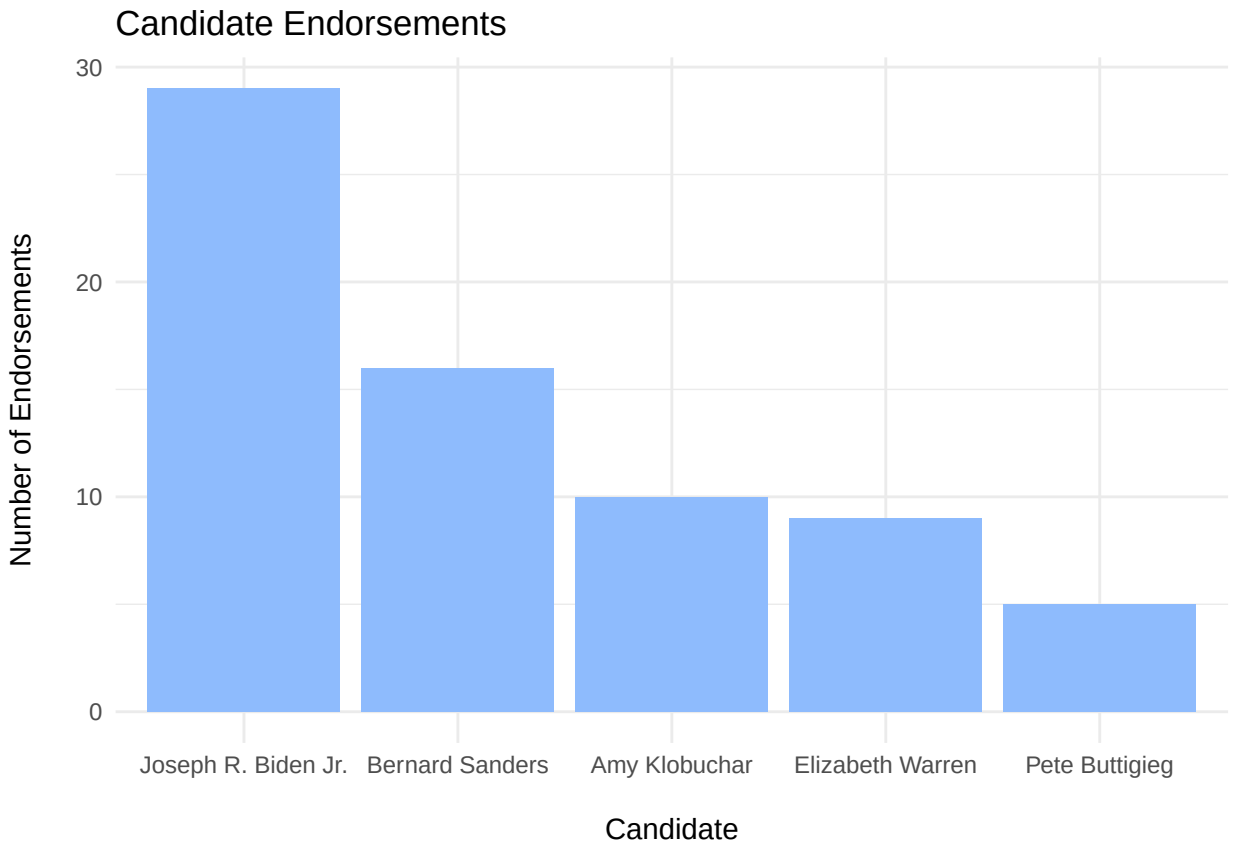
```
## [1] 5
```

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

```
# Create standalone dataset with counts
candidate_endorsements <- Endorsements %>%
  count(candidate_name) %>%
  rename(n_endorsements = n) %>%
  semi_join(polls, by="candidate_name")

# Add counts to the merged dataset
polls <- polls %>%
  left_join(candidate_endorsements, by="candidate_name")
```

- 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.

```
p <- ggplot(candidate_endorsements, aes(x=reorder(candidate_name, -n_endorsements), y=n_endorsements))+
  geom_col(fill="#8ebbfd")


# Didn't like the way this looked, used minimal theme instead
# p + theme_dark()


p +
  labs(x="\nCandidate", y="Number of Endorsements\n", title="Candidate Endorsements") +
  theme_minimal()
```




```
ggsave("CandidateEndorsements.pdf", width = 7, height = 3)
```

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)
#install.packages('lubridate')
library(lubridate)
#install.packages('wordcloud')
library(wordcloud)
trump_tweets_url <- 'https://politicaldatascience.com/PDS/Datasets/trump_tweets.csv'
tweets <- read_csv(trump_tweets_url)
```

- 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.

```
# Use the separate() function to split on the space between date and time.
# Save created_date as date.
tweets <- tweets %>%
  separate(created_at, c("created_date", "created_time"), sep=" ") %>%
  mutate(created_date = as.Date(created_date, format="%m/%d/%Y"))

# Use summarise() to view date range.
```

```
tweets %>%
  summarise(min=min(created_date), max=max(created_date))
```

```
## # A tibble: 1 x 2
##   min        max
##   <date>     <date>
## 1 2014-01-01 2020-02-14
```

- 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.)

```
# Remove retweets.
tweets <- tweets %>%
  filter(!is_retweet)

# Use slice_max() to get a vector of the top 5 favorited numbers
top_fav <- tweets %>%
  slice_max(favorite_count, n=5) %>%
  select(favorite_count)

# Use slice_max() to get a vector of the top 5 retweeted numbers
top_rt <- tweets %>%
  slice_max(retweet_count, n=5) %>%
  select(retweet_count)


# Use the match function to identify rows belonging to a top favorited or top retweeted tweet.
# Select the text of these tweets.
tweets %>%
  filter(retweet_count %in% top_rt$retweet_count | favorite_count %in% top_fav$favorite_count) %>%
  select(text)
```

```
## # A tibble: 9 x 1
##   text
##   <chr>
## 1 "Kobe Bryant despite being one of the truly great basketball players of all t~
## 2 "All is well! Missiles launched from Iran at two military bases located in Ir~
## 3 "https://t.co/VXeKiVzpTf"
## 4 "MERRY CHRISTMAS!"
## 5 "A$AP Rocky released from prison and on his way home to the United States fro~
## 6 "Why would Kim Jong-un insult me by calling me \"old\" when I would NEVER cal~
## 7 "#FraudNewsCNN #FNN https://t.co/WYUnHjjUjg"
## 8 "Such a beautiful and important evening! The forgotten man and woman will nev~
## 9 "TODAY WE MAKE AMERICA GREAT AGAIN!"
```

- 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).

```
# This string wasn't getting caught by tm functions; fix in original dataset first
tweets$text = gsub("&amp", "", tweets$text)

# This regex is to try to get rid of URLs; it gets most of them but some are still not caught
tweets$text = gsub(regex("(http:\\/\\/www\\.|https:\\/\\/www\\.|http:\\/\\/|https:\\/\\/)?[a-z0-9]+([\\
```

```r
# Delete empty tweets to avoid warnings later
tweets <- tweets %>%
  filter(text != "")

# This will be the number of "documents" in the corpus below
nrow(tweets)
```

```
## [1] 29893
```

```r
# Create corpus, look at one of the tweets
trump_tweets <- VCorpus(VectorSource(tweets$text))

inspect(trump_tweets[[1]])
```

```
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 268
##
## I'm seeing Governor Cuomo today at The White House. He must understand that National Security far ex
```

```r
# Clean up the corpus, strip of unnecessary characters
trump_tweets <- trump_tweets %>%
  tm_map(content_transformer(tolower)) %>%
  tm_map(stripWhitespace) %>%
  tm_map(removeWords, stopwords("english")) %>%
  tm_map(removePunctuation) %>%
  tm_map(removeNumbers)

# Now view the same tweet after cleaning
inspect(trump_tweets[[1]])
```

```
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 229
##
## 'm seeing governor cuomo today   white house  must understand  national security far exceeds politic
```

- 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
# Set a seed to control randomness of colors.
set.seed(333444346)

# Define the filepath to output PDF.
pdf("trump_wordcloud.pdf")

# Create wordcloud with desired characteristics.
# Choose the 50 random colors by sampling built-in R colors()
wordcloud(trump_tweets,
          min.freq = 3,
          scale=c(3,.5),
          max.words = 50,
          random.order = TRUE,
          random.color = FALSE,
```

```
          colors=sample(colors(), 50))
dev.off()
```

```
## pdf
##   2
```

- 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.

```
# Create the DTM with tf.idf weights - note that there will be empty documents.
DTM <- DocumentTermMatrix(trump_tweets, control = list(weighting = weightTfIdf, global=c(0.8,Inf)))
```

```
## Warning in weighting(x): empty document(s): 109 211 214 216 275 402 431 436 452
## 454 649 651 731 1087 1301 1302 1314 1741 1758 1767 1824 1955 2076 2688 2775 2918
## 2990 4010 6435 6811 7964 8485 8492 8651 8751 11839 17081 20441 26026
```

```
# Inspect the DTM, note that it is 100% sparse.
# Attempting to turn this into a matrix exhausts memory.
inspect(DTM)
```

```
## <<DocumentTermMatrix (documents: 29893, terms: 33942)>>
## Non-/sparse entries: 345933/1014282273
## Sparsity           : 100%
## Maximal term length: 47
## Weighting          : term frequency - inverse document frequency (normalized) (tf-idf)
## Sample             :
##       Terms
## Docs   america great just people president realdonaldtrump thank thanks trump
##   1013       0     0    0      0         0                0     0      0     0
##   236        0     0    0      0         0                0     0      0     0
##   428        0     0    0      0         0                0     0      0     0
##   491        0     0    0      0         0                0     0      0     0
##   500        0     0    0      0         0                0     0      0     0
##   546        0     0    0      0         0                0     0      0     0
##   548        0     0    0      0         0                0     0      0     0
##   755        0     0    0      0         0                0     0      0     0
##   933        0     0    0      0         0                0     0      0     0
##   96         0     0    0      0         0                0     0      0     0
##       Terms
## Docs   will
##   1013    0
##   236     0
##   428     0
##   491     0
##   500     0
##   546     0
##   548     0
##   755     0
##   933     0
##   96      0
```

```
# Reducing sparsity by just a bit makes the DTM a lot more manageable.
inspect(removeSparseTerms(DTM, 0.999))
```

```
## <<DocumentTermMatrix (documents: 29893, terms: 1803)>>
## Non-/sparse entries: 253551/53643528
```

```
## Sparsity           : 100%
## Maximal term length: 21
## Weighting          : term frequency - inverse document frequency (normalized) (tf-idf)
## Sample             :
##        Terms
## Docs    america great just people president realdonaldtrump thank thanks trump
##   13046       0     0    0      0         0                 0     0      0     0
##   13623       0     0    0      0         0                 0     0      0     0
##   13889       0     0    0      0         0                 0     0      0     0
##   13916       0     0    0      0         0                 0     0      0     0
##   13938       0     0    0      0         0                 0     0      0     0
##   13985       0     0    0      0         0                 0     0      0     0
##   14000       0     0    0      0         0                 0     0      0     0
##   16195       0     0    0      0         0                 0     0      0     0
##   46          0     0    0      0         0                 0     0      0     0
##   8788        0     0    0      0         0                 0     0      0     0
##        Terms
## Docs    will
##   13046    0
##   13623    0
##   13889    0
##   13916    0
##   13938    0
##   13985    0
##   14000    0
##   16195    0
##   46       0
##   8788     0
```

```
# Turn into matrix
term_mat <- as.matrix(removeSparseTerms(DTM, 0.999))

# This is the number of documents in the corpus.
nrow(term_mat)
```

```
## [1] 29893
```

```
# This is the number of terms.
ncol(term_mat)
```

```
## [1] 1803
```

```
# Get the maximum tf.idf score for each term across all documents.
# Make a "long" tibble.
# Filter out low tf.idf scores and arrange from high to low.
trump_tweets_tfidf <- as_tibble(term_mat) %>%
  summarise(across(everything(), max)) %>%
  pivot_longer(everything()) %>%
  rename(term=name, tf_idf=value) %>%
  filter(tf_idf > 0.8) %>%
  arrange(desc(tf_idf))

# Print the entire list.
print(trump_tweets_tfidf[1:50,], n=50)
```

```
## # A tibble: 50 x 2
```

```
##    term                  tf_idf
##    <chr>                  <dbl>
##  1 bigleaguetruth          9.82
##  2 crookedhillary          9.70
##  3 gopdebate               9.55
##  4 disgraceful             9.48
##  5 unbelievable            9.28
##  6 mikepence               9.20
##  7 imwithyou               9.14
##  8 lindseygrahamsc         9.11
##  9 danscavino              8.96
## 10 june                    8.96
## 11 excellent               8.91
## 12 boring                  8.87
## 13 jimjordan               8.76
## 14 agreed                  8.70
## 15 draintheswamp           8.64
## 16 whistleblower           8.62
## 17 americafirst            8.48
## 18 fun                     8.46
## 19 matter                  8.31
## 20 correct                 8.20
## 21 beginning               8.02
## 22 interesting             7.94
## 23 yes                     7.86
## 24 name                    7.77
## 25 game                    7.72
## 26 statement               7.61
## 27 celebapprentice         7.60
## 28 sad                     7.19
## 29 seanhannity             7.19
## 30 terrible                7.01
## 31 agree                   6.86
## 32 florida                 6.85
## 33 maga                    6.77
## 34 whitehouse              6.66
## 35 enjoy                   6.63
## 36 wow                     6.52
## 37 honor                   6.51
## 38 usa                     6.42
## 39 congratulations         6.40
## 40 totally                 6.36
## 41 crime                   6.34
## 42 nice                    6.31
## 43 amazing                 6.23
## 44 far                     6.19
## 45 via                     6.14
## 46 makeamericagreatagain   6.09
## 47 watch                   6.02
## 48 jobs                    5.95
## 49 foxandfriends           5.91
## 50 true                    5.75
```