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
```

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

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

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

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

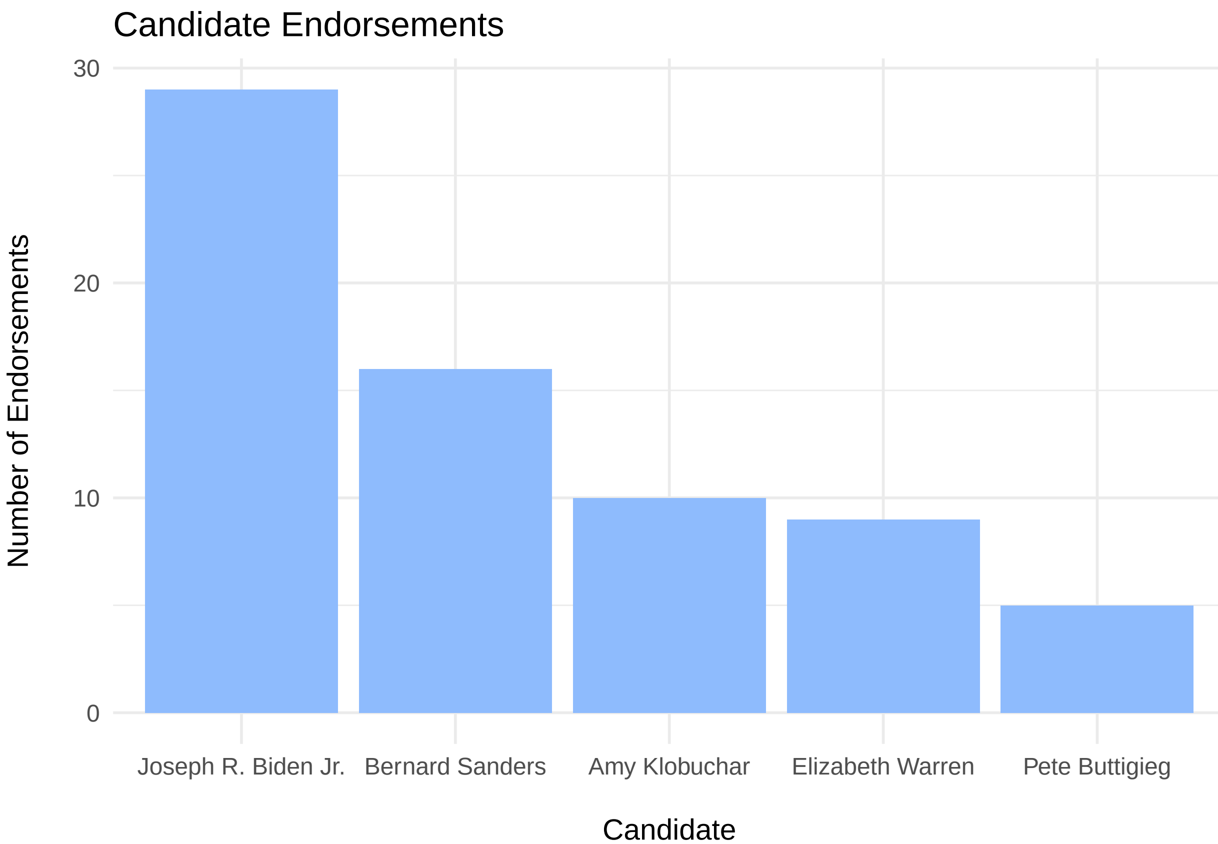
```
## [1] 5
```

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

```r
# 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.

```r
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.

```
tweets <- tweets %>%
  separate(created_at, c("created_date", "created_time"), sep=" ") %>%
  mutate(created_date = as.Date(created_date, format="%m/%d/%Y"))

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.)

```
tweets <- tweets %>%
  filter(!is_retweet)

top_fav <- tweets %>%
  slice_max(favorite_count, n=5) %>%
  select(favorite_count)


top_rt <- tweets %>%
  slice_max(retweet_count, n=5) %>%
  select(retweet_count)




ten_tweets <- tweets %>%
  filter(retweet_count %in% top_rt$retweet_count | favorite_count %in% top_fav$favorite_count) %>%
  select(text)
```

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

```
# vignette("tm")

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

# A bunch of attempts to get rid of URLs in tweets which end up with the highest tf.idf scores in the D
# Attempts unsuccessful so far

# tweets$text = gsub(regex("(https?(:\\/\\/)?[^\\s]+)."), "", tweets$text)

# tweets$text = gsub(regex("(http:\\/\\/www\\.|https:\\/\\/www\\.|http:\\/\\/|https:\\/\\/)?[a-z0-9]+([


# tweets %>%
#   filter(grepl(regex("(http:\\/\\/www\\.|https:\\/\\/www\\.|http:\\/\\/|https:\\/\\/)?[a-z0-9]+([\\-\
#   mutate(new_text = gsub(regex("(http:\\/\\/www\\.|https:\\/\\/www\\.|http:\\/\\/|https:\\/\\/)?[a-z0
#   select(text, new_text)
#
# tweets %>%
#   filter(grepl(regex("((www\\.)?(https?)(:\\/\\/)?[^\\s]+)."),text)) %>%
#   mutate(new_text = gsub(regex("((www\\.)?(https?)(:\\/\\/)?[^\\s]+)."), "", text)) %>%
#   select(text, new_text)
#
#
```

```r
# regex("((www\\.)?(https?)(:\\/\\/)?[^\\s]+).")

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 exc
```

```r
trump_tweets <- trump_tweets %>%
  tm_map(content_transformer(tolower)) %>%
  tm_map(stripWhitespace) %>%
  tm_map(removeWords, stopwords("english")) %>%
  tm_map(removePunctuation) %>%
  tm_map(removeNumbers)
```

- 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.seed(1616161)
wordcloud(trump_tweets,
          min.freq = 3,
          max.words = 50,
          random.order = TRUE,
          random.color = FALSE,
          colors=sample(colors(), 50))
```

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

- 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.

```r
DTM <- DocumentTermMatrix(trump_tweets, control = list(weighting = weightTfIdf, global=c(0.8,Inf)))
```

```
## Warning in weighting(x): empty document(s): 480 482 1824 8946 12142
```

```r
inspect(removeSparseTerms(DTM, 0.98))
```

```
## <<DocumentTermMatrix (documents: 30199, terms: 59)>>
## Non-/sparse entries: 74209/1707532
## Sparsity           : 96%
## Maximal term length: 15
## Weighting          : term frequency - inverse document frequency (normalized) (tf-idf)
## Sample             :
##        Terms
## Docs     america donald     great just people president realdonaldtrump thank
##   11327 0.000000      0 0.0000000    0      0         0       0.0000000     0
##   13071 1.105439      0 0.6812955    0      0         0       0.0000000     0
##   146   1.473919      0 0.9083940    0      0         0       0.0000000     0
##   19657 0.000000      0 0.0000000    0      0         0       0.0000000     0
##   20106 0.000000      0 0.0000000    0      0         0       0.5535537     0
##   21008 0.000000      0 0.0000000    0      0         0       0.3690358     0
##   2798  1.473919      0 0.9083940    0      0         0       0.0000000     0
##   3142  1.473919      0 0.9083940    0      0         0       0.0000000     0
##   3855  1.473919      0 0.9083940    0      0         0       0.0000000     0
##   6952  0.000000      0 0.0000000    0      0         0       0.0000000     0
##        Terms
## Docs    trump will
##   11327     0    0
##   13071     0    0
##   146       0    0
##   19657     0    0
##   20106     0    0
##   21008     0    0
##   2798      0    0
##   3142      0    0
##   3855      0    0
##   6952      0    0
```

```r
term_mat <- as.matrix(removeSparseTerms(DTM, 0.999))

nrow(term_mat)
```

```
## [1] 30199
```

```r
ncol(term_mat)
```

```
## [1] 1758
```

```r
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))
```

```
## Warning: `as.tibble()` was deprecated in tibble 2.0.0.
## Please use `as_tibble()` instead.
## The signature and semantics have changed, see `?as_tibble`.
```

```
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

```r
trump_tweets_tfidf[1:50,]
```

```
## # A tibble: 50 x 2
##    term            tf_idf
##    <chr>            <dbl>
##  1 boring            8.88
##  2 whistleblower     8.63
##  3 name              7.78
##  4 celebapprentice   7.61
##  5 seanhannity       7.20
##  6 enjoy             6.64
##  7 usa               6.43
##  8 foxandfriends     5.93
##  9 cute              4.88
## 10 july              4.86
## # ... with 40 more rows
```