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.

- 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.

```
Endorsements <- Endorsements %>% #Using dplyr to rename the endorsee column
  rename(candidate_name = endorsee)

#This line will transform our dataframe into a tibble
Endorsements <- as_tibble(Endorsements)

#Next we're going to use the "filter()" function to get 6 candidates and then
#use the "select()# function to chose our 5 variables. We can do both with the
#piping function
polls <- polls %>%
  filter(candidate_name == "Amy Klobuchar" | candidate_name == "Bernard Sanders" |
           candidate_name == "Elizabeth Warren" | candidate_name == "Joseph R. Biden Jr."|
           candidate_name == "Michael Bloomberg" |candidate_name == "Pete Buttigieg") %>%
  select(candidate_name, sample_size, start_date, party, pct)

#Here I'm going to check for the unique candidate names in each dataset
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
```

```
#I'll need to change Bernie and Joe's name in the Endorements dataset to make
#sure they match
Endorsements <- Endorsements %>%
  mutate(candidate_name = replace(candidate_name, candidate_name ==
                                    "Bernie Sanders", "Bernard Sanders"),
         candidate_name = replace(candidate_name, candidate_name ==
                                    "Joe Biden", "Joseph R. Biden Jr."))

#We're going to join the two datasets on candidate_name
new_data <- inner_join(polls, Endorsements, by = "candidate_name")
```

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.
- 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.