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.

```
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')
```

```
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
```

```
Endorsements_filtered <- Endorsements %>%
  # Change variable name endorsee in Endorsements df to candidate_name
  rename(candidate_name = endorsee) %>%
  # Change endorsements df to tibble
  as.tibble()
```

```
## 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.
```

```
polls_filtered <- polls %>%
  # Filter poll variable to include 6 candidates
  filter(candidate_name %in% c(
    "Amy Klobuchar",
    "Bernard Sanders",
    "Elizabeth Warren",
    "Joseph R. Biden Jr.",
    "Michael Bloomberg",
```

```
    "Pete Buttigieg"
  )) %>%
  # Subset the data to include the five variables
  select(candidate_name, sample_size, start_date, party, pct)

# Find differently spelled names and unify them (use only dplyr)
# The 'polls' names are used as the standard here
Endorsements_filtered <- Endorsements_filtered %>%
  mutate(candidate_name = case_when(
    candidate_name == "Bernie Sanders" ~ "Bernard Sanders",
    candidate_name == "Joe Biden" ~ "Joseph R. Biden Jr.",
    TRUE ~ as.character(candidate_name)
  ))

# Merge the two datasets by candidate_name
# Currently not sure what kind of merge to do here. Will think about it more.
test <- inner_join(polls_filtered, Endorsements_filtered, by = "candidate_name")

# Variable where number of endorsements is counted


# Plot each of the 6(?) candidates


# add theme_dark()


# Add titles, and save plot
```

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:

```
library(tidyverse)
library(tm)
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

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

```
library(lubridate)
```

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

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

```
library(wordcloud)
```

```
## 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.
- 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 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.

```
# PART 1 ---------------------------------------------------------------------
## Separate day and time
tweets_filtered <- tweets %>%
  separate(created_at, into = c("day", "time"), sep = " ", remove = FALSE) %>%
  mutate(day = mdy(day))

## Report range of dates
range(tweets_filtered$day)
```

```
## [1] "2014-01-01" "2020-02-14"
```

```
# PART 2 ---------------------------------------------------------------------
## Filtering out retweets
tweets_filtered <- tweets_filtered %>%
  filter(is_retweet == FALSE)

## Top 5 tweets based on retweet count
slice_max(tweets_filtered, order_by = retweet_count, n = 5)$text
```

```
## [1] "#FraudNewsCNN #FNN https://t.co/WYUnHjjUjg"
```

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

```
## Top 5 retweets based on favorite count
slice_max(tweets_filtered, order_by = favorite_count, n = 5)$text
```

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

```
# PART 3 ----------------------------------------------------------------------
Corpus <- VCorpus(VectorSource(tweets_filtered$text))

# PART 4 ----------------------------------------------------------------------
removeSymbols <- content_transformer(function(x, pattern) gsub(pattern, "", x))

Corpus <- Corpus %>%
  ## Remove Punctuation
  tm_map(removePunctuation) %>%
  ## Various symbols not removed with any function in the tm package
  tm_map(removeSymbols, "[""‘’...&@():$%.\\/\\\\]") %>%
  ## Remove Numbers
  tm_map(removeNumbers) %>%
  ## Make all lowercase
  tm_map(content_transformer(tolower)) %>%
  ## Remove Stopwords
  tm_map(removeWords, c("im", stopwords("english"))) %>%
  ## Stripping whitespace
  tm_map(stripWhitespace)

# PART 5 ----------------------------------------------------------------------
## Create DTM object
DTM <- DocumentTermMatrix(Corpus, control = list(weighting = weightTfIdf))
```

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