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))
```

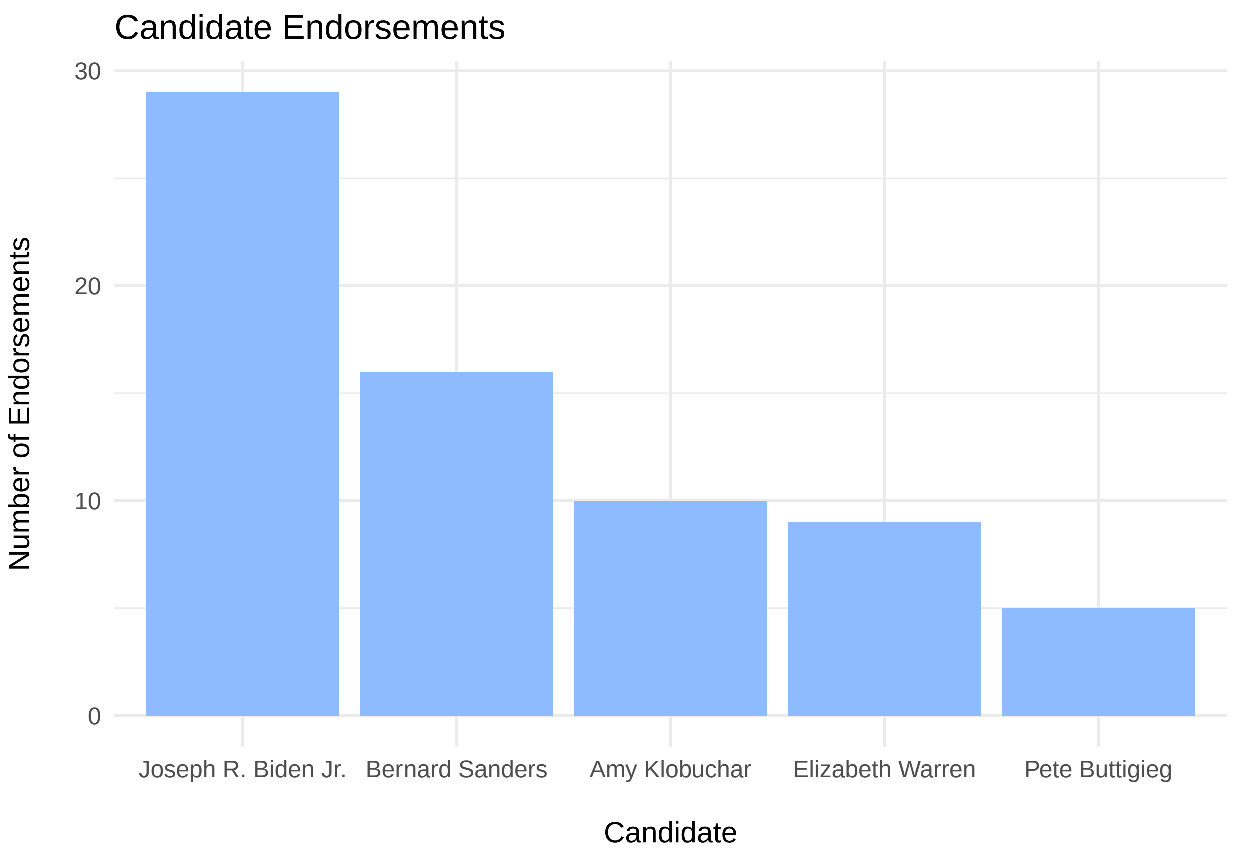
```
## [1] 5
```

- 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()
```



```r
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:

```r
# 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.

```r
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)


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


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

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

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

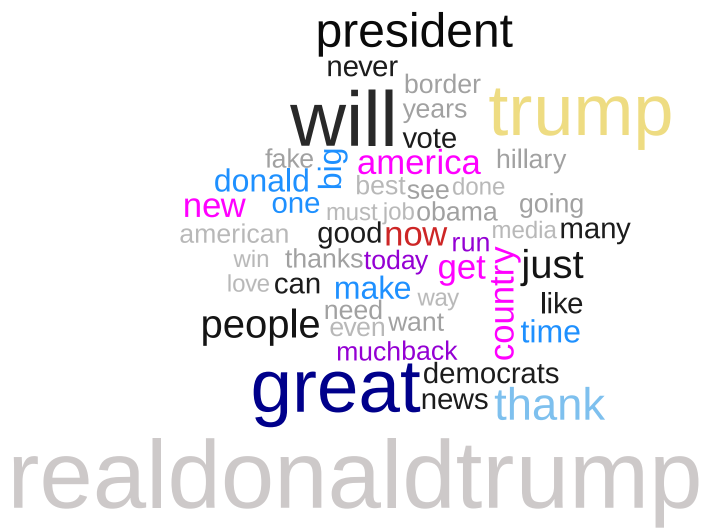
```
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 268
##
## I'm seeing Governor Cuomo today at The White House. He must understand that National Security far ex
```

```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.seed(33333)

wordcloud(trump_tweets,
          min.freq = 3,
          scale=c(3,.5),
          max.words = 50,
          random.order = TRUE,
          random.color = FALSE,
          colors=sample(colors(), 50))
```



- 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): 20 107 111 118 122 126 128 129 134
## 143 192 193 194 195 198 199 204 227 230 232 233 244 245 275 295 312 313 315 368
## 384 427 456 457 458 463 479 480 482 529 542 555 565 660 661 683 684 685 687 692
## 741 769 784 785 829 832 833 897 945 946 1006 1041 1049 1051 1054 1062 1089 1090
## 1091 1105 1114 1116 1145 1173 1181 1189 1192 1193 1195 1199 1207 1209 1211 1234
## 1235 1236 1258 1259 1260 1261 1262 1328 1378 1379 1382 1384 1393 1477 1579 1599
## 1812 1824 1841 1850 1907 1924 2000 2002 2019 2020 2027 2032 2045 2092 2094 2138
## 2153 2170 2201 2204 2205 2206 2327 2385 2386 2403 2447 2490 2513 2515 2528 2539
## 2541 2568 2578 2584 2611 2676 2693 2696 2718 2720 2734 2735 2761 2809 2818 2826
## 2828 2831 2832 2833 2888 2889 2904 2905 2907 2910 2926 2928 2945 2955 2959 2963
## 2973 2984 2985 2987 2989 2990 3020 3021 3064 3136 3207 3283 3324 3372 3429 3566
## 3627 3633 3794 3902 3903 3917 4077 4164 4169 4171 4263 4270 4287 4289 4330 4334
## 4342 4343 4400 4402 4427 4468 4530 4701 4789 4837 4863 4871 4897 4912 4973 4974
## 4979 4984 4998 5058 5078 5191 5192 5199 5200 5263 5264 5298 5381 5417 5421 5464
## 5508 5697 5703 5970 6214 6224 6227 6255 6374 6506 6507 6515 6516 6521 6524 6525
## 6544 6545 6560 6561 6563 6575 6590 6596 6597 6632 6634 6657 6659 6661 6664 6675
## 6692 6767 6859 6883 6884 6971 7003 7004 7019 7020 7022 7034 7035 7054 7056 7108
## 7138 7197 7271 7282 7296 7308 7353 7380 7430 7449 7475 7850 8037 8162 8223 8224
## 8225 8275 8284 8294 8296 8308 8320 8321 8378 8389 8394 8410 8429 8430 8436 8451
## 8453 8464 8474 8488 8542 8560 8671 8718 8758 8770 8771 8772 8778 8779 8781 8782
## 8804 8822 8826 8835 8899 8946 9046 9063 9270 9424 9511 9577 9619 10002 10098
## 12142 16492 17385 17532 20122 20747 26332
```

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

```
## <<DocumentTermMatrix (documents: 30199, terms: 59)>>
## Non-/sparse entries: 74029/1707712
## Sparsity           : 96%
## Maximal term length: 15
## Weighting          : term frequency - inverse document frequency (normalized) (tf-idf)
## Sample             :
##        Terms
## Docs    america great just people president realdonaldtrump thank   thanks
##   1155        0     0    0      0         0               0     0 0.000000
##   13083       0     0    0      0         0               0     0 0.000000
##   15265       0     0    0      0         0               0     0 5.243777
##   18641       0     0    0      0         0               0     0 5.243777
##   18659       0     0    0      0         0               0     0 5.243777
##   18824       0     0    0      0         0               0     0 5.243777
##   18860       0     0    0      0         0               0     0 5.243777
##   19294       0     0    0      0         0               0     0 5.243777
##   19527       0     0    0      0         0               0     0 5.243777
##   2540        0     0    0      0         0               0     0 0.000000
##        Terms
## Docs    trump will
##   1155      0    0
##   13083     0    0
##   15265     0    0
##   18641     0    0
##   18659     0    0
##   18824     0    0
##   18860     0    0
##   19294     0    0
##   19527     0    0
##   2540      0    0
```

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

nrow(term_mat)
```

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

```r
ncol(term_mat)
```

```
## [1] 1760
```

```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))


print(trump_tweets_tfidf[1:50,], n=50)
```

```
## # A tibble: 50 x 2
##    term                   tf_idf
##    <chr>                   <dbl>
##  1 bigleaguetruth           9.84
##  2 crookedhillary           9.71
##  3 gopdebate                9.56
##  4 disgraceful              9.49
##  5 unbelievable             9.30
##  6 mikepence                9.21
##  7 imwithyou                9.15
##  8 lindseygrahamsc          9.13
##  9 danscavino               8.98
## 10 june                     8.98
## 11 excellent                8.93
## 12 boring                   8.88
## 13 jimjordan                8.77
## 14 agreed                   8.71
## 15 draintheswamp            8.65
## 16 whistleblower            8.63
## 17 americafirst             8.49
## 18 fun                      8.47
## 19 matter                   8.33
## 20 correct                  8.21
## 21 beginning                8.04
## 22 interesting              7.95
## 23 yes                      7.87
## 24 name                     7.78
## 25 game                     7.73
## 26 statement                7.62
## 27 celebapprentice          7.62
## 28 sad                      7.20
## 29 seanhannity              7.20
## 30 terrible                 7.02
## 31 agree                    6.87
## 32 florida                  6.86
## 33 maga                    6.79
## 34 whitehouse              6.67
## 35 enjoy                   6.65
## 36 wow                     6.53
## 37 honor                   6.52
## 38 usa                     6.44
## 39 congratulations         6.41
## 40 totally                 6.38
## 41 crime                   6.35
## 42 nice                    6.32
## 43 amazing                 6.24
## 44 far                     6.20
## 45 via                     6.15
## 46 makeamericagreatagain   6.10
## 47 watch                   6.04
## 48 jobs                    5.96
## 49 foxandfriends           5.93
## 50 true                    5.77
```