Text Mining
Q1 (1pts)
Q2 (1pts)
Q3 (1pts)
Q4 (1pts)
Q5 (2pts)
Q6 (1pts)
Q7 (1pts)
Q8 (2pts)

HW 8

SDS322E

Enter your name and EID here

Please submit as a PDF or HTML file on Canvas before the due date.

For all questions, include the R commands/functions that you used to find your answer. Answers without supporting code will not receive credit.

Review of how to submit this assignment

All homework assignments will be completed using R Markdown. These `.Rmd` files consist of >text/syntax (formatted using Markdown) alongside embedded R code. When you have completed the assignment (by adding R code inside codeblocks and supporting text outside of the codeblocks), create your document as follows (assuming you are using the edupod server and submitting HTML):

- Click the arrow next to the “Knit” button (above)
- Choose “Knit to HTML”
- Go to Files pane and put checkmark next to the correct HTML file
- Click on the blue gear icon (“More”) and click Export
- Download the file and then upload to Canvas
- To submit a PDF, open your HTML file and print it to a pdf, then upload the pdf as your submission.

Text Mining

In this homework we will practice our text mining and sentiment analysis skills. We will use the `senators` dataset in the `fivethirtyeightdata` package, which can be installed by running the following code:

```
install.packages(
  'fivethirtyeightdata',
  repos = 'https://fivethirtyeightdata.github.io/drat/',
  type = 'source'
)
```

We can then load the dataset by running

```
library(fivethirtyeightdata)
senators <- senators
```

If this does not work for you: change the previous code chunk to have `{r, eval = FALSE}` and remove the `eval = FALSE` in the following code chunk.

```
senators <- read_csv("https://github.com/fivethirtyeight/data/blob/master/twitter-ratio/se
nators.csv?raw=true") %>%
  mutate(
    created_at = mdy_hm(created_at, tz = "GMT"),
    party = factor(party, levels = c("D", "I", "R")),
    state = as.factor(state)
  ) %>% select(created_at, user, text, url, replies, retweets, everything())
```

Review the documentation by running `?senators` for more information.

Q1 (1pts)

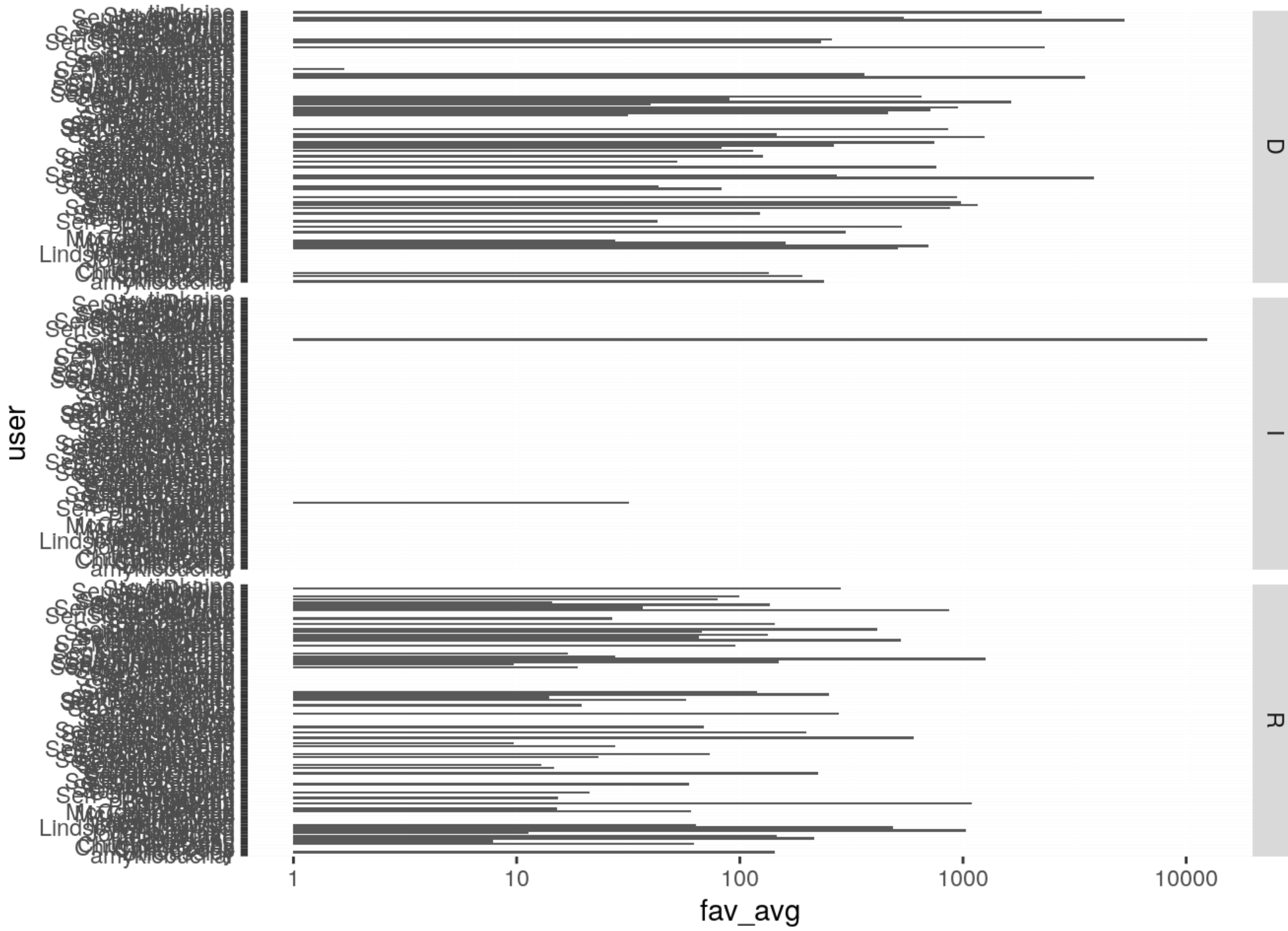
Using `group_by()` and `summarise()`, for each senator compute the average number of favorites for their tweets. Then make a histogram (with `scale_x_log10()`), faceted by party, of the average number of favorites. **Based on this, which party (D or R) tends to have senators that attract the most favorites? And who are the five most popular senators in terms of this metric?**

```
## Your code here
fav_avg_sen <- senators %>% group_by(user) %>% summarise(fav_avg = mean(favorites), party
 = party) %>% unique()
```

```
## `summarise()` has grouped output by 'user'. You can override using the
## `.groups` argument.
```

```
ggplot(fav_avg_sen, aes(x = fav_avg, y = user)) + geom_histogram(stat = "identity") + scal
e_x_log10() + facet_grid(party~.)
```

```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```




```
fav_avg_sen %>% arrange(-fav_avg) %>% head()
```

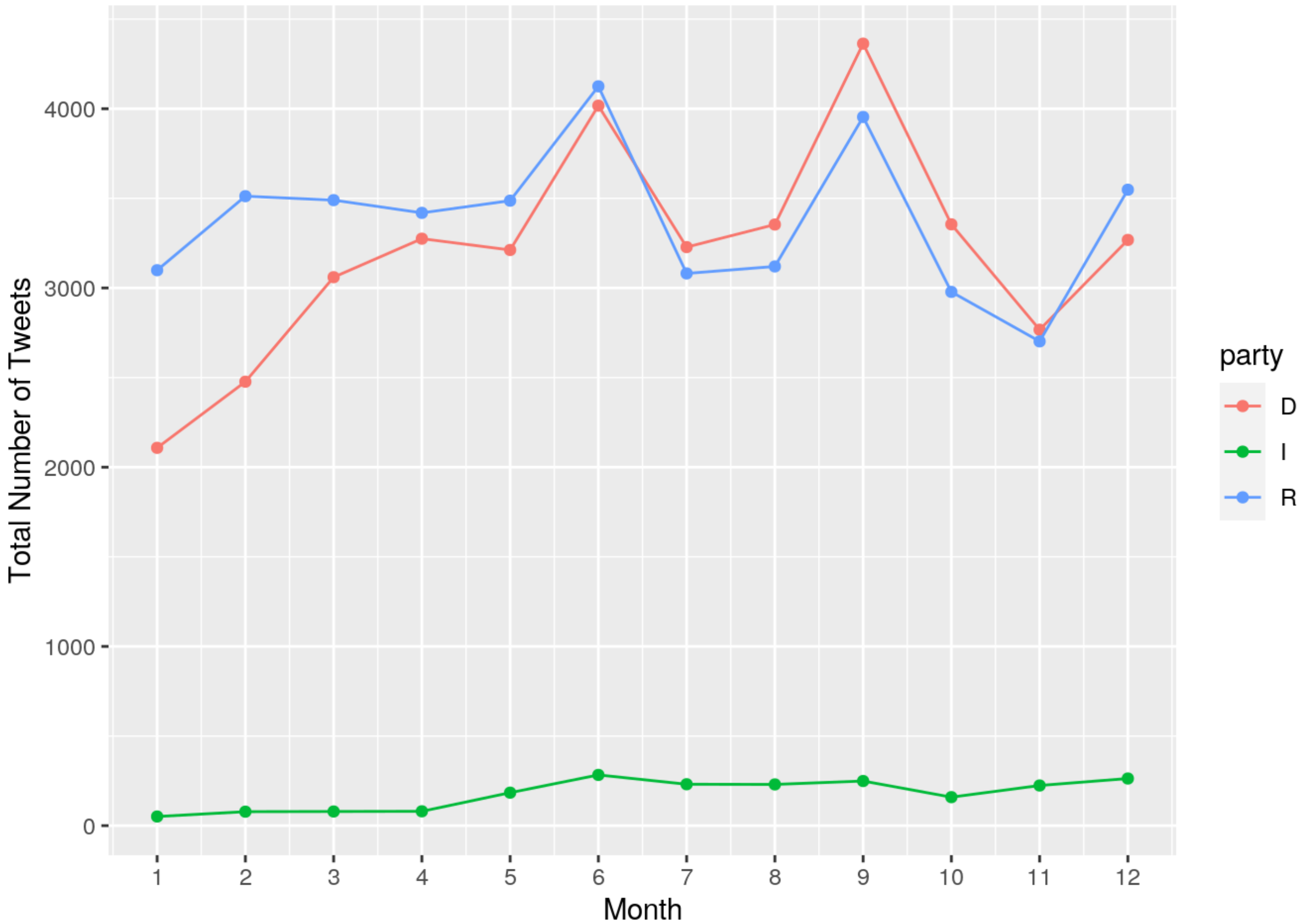
```
## # A tibble: 6 × 3
## # Groups:   user [6]
##   user            fav_avg party
##   <chr>             <dbl> <fct>
## 1 SenSanders       12387. I
## 2 SenWarren         5281. D
## 3 SenatorLeahy      3866. D
## 4 SenKamalaHarris   3518. D
## 5 SenSchumer        2316. D
## 6 timkaine          2245. D
```

Answer: Based on these graphs, between D and R, it can be seen that D tends to have senators that attract the most favorites. The five most popular senators in terms of this metric are: SenSanders, SenWarren, SenatorLeahy, SenKamalaHarris, and SenSchumer. If the I party is not considered, then the five most popular senators in terms of this metric are: SenWarren, SenatorLeahy, SenKamalaHarris, SenSchumer, and timkaine.

Q2 (1pts)

The `created_at` column contains the date and time at which a given tweet was posted. Using the `year()` and `month()` functions from `lubridate` (along with any useful functions from `dplyr`), use `ggplot()` to make a figure with the month on the x-axis and the total number of tweets for each party during 2016 on the y-axis; use `geom_point()` and `geom_line()` , coloring by party. **What trends are present in the data regarding the use of twitter by senators across parties during 2016? No “right” answer here, just speak to what you see.**

```
## Your code here
senator2016 <- senators %>% filter(year(created_at) == 16) %>% mutate(month = month(create
d_at)) %>% group_by(month, party) %>% count()
ggplot(senator2016, aes(x = month, y = n, color = party)) + geom_point() + geom_line() + x
lab('Month') + ylab('Total Number of Tweets') + scale_x_continuous(breaks = 1:12)
```

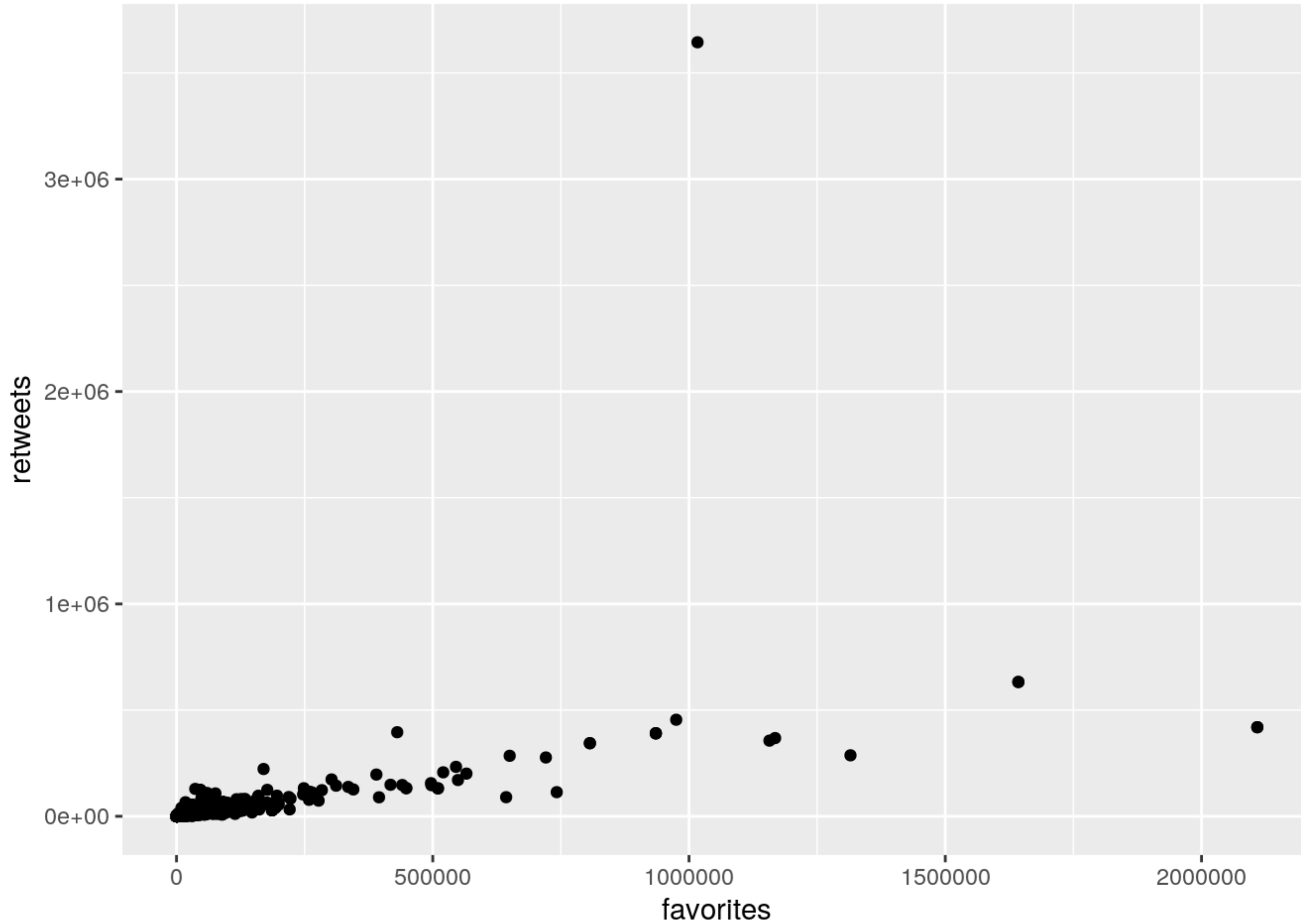


Answer: Based on the graph, it can be seen that number of tweets for parties generally tended to peak the most during the months of July and September as the number of tweets appeared to go up and down the most between these months. Overall there was a general increase of number of tweets throughout the year, with D and R parties making drastically more than I. R generally tended to make more tweets than D overall during the first half of the year while D appears to make more towards the second half.

Q3 (1pts)

Make a scatterplot of `retweets` (y-axis) and `favorites` (x-axis) using `ggplot()` . You will notice a data point that is not following the trend of the others (an outlier). **Use the information in this plot to help you isolate the row/tweet in question using `dplyr` functions. Then, display the user and text of the tweet.**

```
## Your code here
ggplot(senators, aes(x = favorites, y = retweets)) + geom_point()
```




```
senators %>% filter(retweets > 3e+06) %>% select(user, text)
```

```
## # A tibble: 1 × 2
##   user           text
##   <chr>          <chr>
## 1 SenCortezMasto RT @carterjwm: HELP ME PLEASE. A MAN NEEDS HIS NUGGS https://t…
```

Q4 (1pts)

Recall that `str_detect()` will look for a pattern and render `TRUE` or `FALSE` if the pattern occurs or not, respectively.

Create a new column (using `mutate` with `str_detect`) called `retweet` that is `TRUE` if the tweet (i.e., the variable *text*) begins with the text “RT” and FALSE otherwise. *Note: you can do so using regular expressions and the symbol ^ will be useful!*

Store the new data frame as `senators_rt`. Then using `dplyr` functions **find what proportion of Ted Cruz’s (`user == "SenTedCruz"`) tweets are retweets.**

```
## Your code here
senators_rt <- senators %>% mutate(retweet = str_detect(text, "^(RT)", negate = FALSE))
senators_rt %>% filter(user == "SenTedCruz") %>% summarize(rtwt_prop = sum(retweet)/n())
```

```
## # A tibble: 1 × 1
##   rtwt_prop
##       <dbl>
## 1     0.388
```

Answer: 38.76% of Ted Cruz's tweets are retweets.

Q5 (2pts)

Getting "ratio'd" on twitter often refers to having a high *replies*-to-*favorites* ratio, such that the tweet is generating more discussion/controversy than it is garnering support through likes.

Let's investigate which senator gets ratio'd the most on average:

1. Only consider tweets that are NOT retweets, so filter those out using the retweet column you created in the previous question before doing the calculations.
2. Compute the *replies*-to-*favorites* ratio for each tweet. However, lots of tweets get zero favorites and we can't divide by zero. To fix this, let's add 1 to `favorites` before taking the ratio:

   ratio = replies/(favorites+1)
3. Compute the mean of this ratio for each senator (i.e., `user`).
4. Find the senator whose average mean ratio is greatest.

Which senator has the highest average ratio?

```
## Your code here
senators_rt %>% filter(retweet == FALSE) %>% mutate(ratio = replies/(favorites+1)) %>% gro
up_by(user) %>% summarize(mean_ratio = mean(ratio)) %>% slice_max(mean_ratio, n=1)
```

```
## # A tibble: 1 × 2
##   user       mean_ratio
##   <chr>           <dbl>
## 1 JohnCornyn       1.73
```

Answer: JohnCornyn has the highest average ratio of 1.725035

Q6 (1pts)

Now let's do some analysis of the actual tweets using the `tidytext` package. Using `dplyr` on the `senators_rt` dataset:

1. Remove all of the retweet rows.
2. Mutate/overwrite the *text* column to delete any pattern of the form `"&.*;"` by using `text = str_remove_all(text, "&.*;")` .
3. Use the `tidytext` function `%>% unnest_tokens(word, text)` to split the tweets into individual words and tokens.
4. Save the resulting data frame as `senator_words` .

The expanded dataset should have a row for each word/token in each tweet.

Now, use `dplyr` functions on this new data frame to determine the most commonly used word across all tweets. What is this word?

```
## Your code here
senator_words <- senators_rt %>% filter(retweet == FALSE) %>% mutate(text = str_remove_all
(text, "&.*;")) %>% unnest_tokens(word, text)
senator_words %>% group_by(word) %>% summarize(n=n()) %>% slice_max(n, n=1)
```

```
## # A tibble: 1 × 2
##   word       n
##   <chr>  <int>
## 1 t.co  179768
```

Answer: The most commonly used word across all tweets is t.co

Q7 (1pts)

Words like these are commonly referred to as stopwords (structural English words that do not add much meaning to a sentence). They are routinely filtered out prior to text analysis, since they are rarely informative.

The `tidytext` package contains a dataframe called `stop_words` containing three commonly used stopword lexicons. Filter it down to `lexicon == "snowball"` and save it as a new dataframe called `mystops`. Then `anti_join()` your `senator_words` dataset to `mystops` to remove any of those stopwords. Save the resulting dataset as `senator_words_clean`.

How many distinct words/tokens is that (i.e., how many distinct things are in the word column)?

```
## Your code here
mystops <- stop_words %>% filter(lexicon == "snowball")
senator_words_clean <- anti_join(senator_words, mystops)
```

```
## Joining, by = "word"
```

```
senator_words_clean %>% distinct(word) %>% count()
```

```
## # A tibble: 1 × 1
##        n
##    <int>
## 1 282166
```

Answer: There are 282166 distinct words

Q8 (2pts)

Let’s try a quick sentiment analysis!

Sentiment analysis involves using a scored lexicon of words, with emotion scores or valence labels (negative vs. positive) indicating each word's emotional content.

We will use a standard lexicon stored in a dataset called `sentiments` which, in brief, categorizes each word as either having negative or positive vibes.

Left_join your `senator_words_clean` dataset to the `sentiments` dataset. Note that any word not labelled with a sentiment will have an `NA` for sentiment after joining. Let's change those to "neutral" by piping the following code:

```
%>% mutate(sentiment = replace_na(sentiment, "neutral"))
```

This way these words with "no" sentiment won't get dropped during computations. Save the resulting dataset as `senator_sentiment`.

Which senator has the greatest proportion of negative words in their tweets?

```
## Your code here
senator_sentiment <- left_join(senator_words_clean, sentiments) %>% mutate(sentiment = rep
lace_na(sentiment, "neutral"))
```

```
## Joining, by = "word"
```

```
senator_sentiment %>% select(user, sentiment) %>% group_by(user, sentiment) %>% summarise
(n = n()) %>% pivot_wider(names_from = sentiment, values_from = n) %>% mutate(neg_prop = n
egative/(negative+neutral+positive)) %>% arrange(-neg_prop) %>% head()
```

```
## `summarise()` has grouped output by 'user'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 6 × 5
## # Groups:   user [6]
##   user           negative neutral positive neg_prop
##   <chr>             <int>   <int>    <int>    <dbl>
## 1 SenSanders         2224   31064     2152   0.0628
## 2 SenBobCasey        1624   27911     1477   0.0524
## 3 SenBlumenthal      2115   35686     3303   0.0515
## 4 SenWarren          1558   27511     1728   0.0506
## 5 ChrisVanHollen     1796   33486     2354   0.0477
## 6 SenJeffMerkley     1804   34409     2321   0.0468
```

Answer: SenSanders has the greatest proportion of negative words in their tweets with a proportion of 6.28%