Intro to Data Science - HW 11


```
# Enter your name here: Bhavya Shah
```

Attribution statement: (choose only one and delete the rest)

```
# 1. I did this homework with the help from Shruti Rao, but did not cut/paste any code.
```

Text mining plays an important role in many industries because of the prevalence of text in the interactions between customers and company representatives. Even when the customer interaction is by speech, rather than by chat or email, speech to text algorithms have gotten so good that transcriptions of these spoken word interactions are often available. To an increasing extent, a data scientist needs to be able to wield tools that turn a body of text into actionable insights. In this homework, we explore a real **City of Syracuse dataset** using the **quanteda** and **quanteda.textplots** packages. Make sure to install the **quanteda** and **quanteda.textplots** packages before following the steps below:

Part 1: Load and visualize the data file

A. Take a look at this article: https://samedelstein.medium.com/snowplow-naming-contest-data-2dcd38272caf (https://samedelstein.medium.com/snowplow-naming-contest-data-2dcd38272caf) and write a comment in your R script, briefly describing what it is about.

```
#The data discusses the most common names chosen by people and the writer's opinions on the snow plow naming contest data.
```

B. Read the data from the following URL into a dataframe called **df**: https://intro-datascience.s3.us-east-2.amazonaws.com/snowplownames.csv (https://intro-datascience.s3.us-east-2.amazonaws.com/snowplownames.csv)

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.2     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.4.2     ✓ tibble    3.2.1
## ✓ lubridate 1.9.2     ✓ tidyr     1.3.0
## ✓ purrr     1.0.1
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
e errors
```

```
df <- data.frame(read_csv("https://intro-datascience.s3.us-east-2.amazonaws.com/snowplownames.csv"))
```

```
## Rows: 1907 Columns: 5
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (3): submitter_name_anonymized, snowplow_name, meaning
## dbl (1): submission_number
## lgl (1): winning_name
##
## 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.
```

C. Inspect the **df** dataframe which column contains an explanation of the meaning of each submitted snowplow name?

```
str(df)
```

```
## 'data.frame':    1907 obs. of  5 variables:
##  $ submission_number        : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ submitter_name_anonymized: chr  "kjlt9cua" "KXKaabXN" "kjlt9cua" "Rv9sODqp" ...
##  $ snowplow_name            : chr  "rudolph" "salt life" "blizzard" "butter" ...
##  $ meaning                  : chr  "The red nose cuts through any storm." "We may not be near the ocean like everyone else with the stickers that say Salt Life, but we have plenty of salt!" "This plow can handle any storm." "It's amazing how the snow plows through snow like butter!" ...
##  $ winning_name             : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
```

```
#Each submitted snowplow name's significance is explained in the "meaning" column.
```

D. Transform that column into a **document-feature matrix**, using the **corpus()**, **tokens(), tokens_select()**, and **dfm()** functions from the quanteda package. Do not forget to **remove stop words**.

```
#install.packages("quanteda")
library(quanteda)
```

```
## Warning in stringi::stri_info(): Your current locale is not in the list of
## available locales. Some functions may not work properly. Refer to
## stri_locale_list() for more details on known locale specifiers.

## Warning in stringi::stri_info(): Your current locale is not in the list of
## available locales. Some functions may not work properly. Refer to
## stri_locale_list() for more details on known locale specifiers.
```

```
## Package version: 3.3.0
## Unicode version: 13.0
## ICU version: 66.1
```

```
## Parallel computing: 16 of 16 threads used.
```

```
## See https://quanteda.io for tutorials and examples.
```

```
snowcorp <- corpus(df$meaning, docnames=df$submission_number)
```

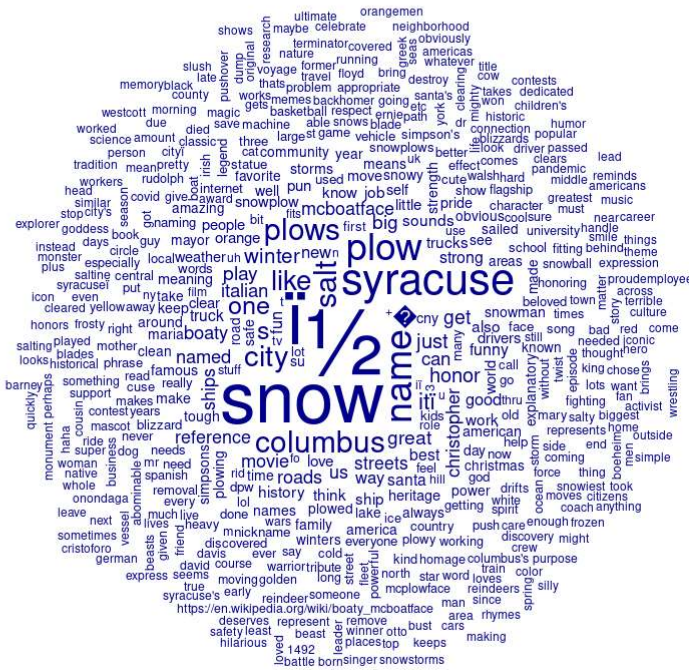
```
## Warning: NA is replaced by empty string
```

```
Token <- tokens(snowcorp, remove_punct=TRUE)
Token_nostop <- tokens_select(Token, pattern = stopwords("en"), selection = "remove")
SnoDFM <- dfm(Token_nostop, tolower = TRUE)
SnoDFM
```

```
## Document-feature matrix of: 1,907 documents, 2,810 features (99.83% sparse) and 0 docvars.
##     features
## docs red nose cuts storm may near ocean like everyone else
##    1   1    1    1     1   0    0     0    0        0    0
##    2   0    0    0     0   1    1     1    1        1    1
##    3   0    0    0     1   0    0     0    0        0    0
##    4   0    0    0     0   0    0     0    1        0    0
##    5   0    0    0     0   0    0     0    0        0    0
##    6   0    0    0     0   0    0     0    0        0    0
## [ reached max_ndoc ... 1,901 more documents, reached max_nfeat ... 2,800 more features ]
```

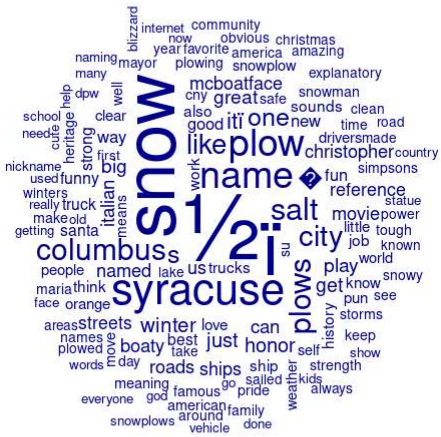
E. Plot a **word cloud** where a word is only represented if it appears **at least 2 times** in the corpus. **Hint:** use **textplot_wordcloud()** from the quanteda.textplots package:

```
#install.packages("quanteda.textplots")
library(quanteda.textplots)
textplot_wordcloud( SnoDFM,min_count = 2)
```



F. Next, **increase the minimum count to 10**. What happens to the word cloud? **Explain in a comment**.

```
textplot_wordcloud( SnoDFM,min_count = 10)
```




```
#The new word cloud is less complex than the previous word cloud. It is comparatively easy to re
ad.
```

G. What are the top 10 words in the word cloud?

Hint: use textstat_frequency in the quanteda.textstats package

```
#install.packages("quanteda.textstats")
library(quanteda.textstats)
library(quanteda)
textstat_frequency(SnoDFM,10)
```

```
##     feature frequency rank docfreq group
## 1         ½       432    1     143   all
## 2         ï       336    2     147   all
## 3      snow       321    3     292   all
## 4  syracuse       174    4     164   all
## 5      name       142    5     136   all
## 6      plow       140    6     130   all
## 7      salt       104    7      83   all
## 8     plows       100    8      98   all
## 9  columbus       100    8      96   all
## 10     city        96   10      94   all
```

H. Explain in a comment what you observed in the sorted list of word counts.

```
#The sorted list contains terms that are in descending order and are ordered in descending orde
r. The first two row contains symbols.
```

Part 2: Analyze the sentiment of the descriptions

###Match the review words with positive and negative words

A. Read in the list of positive words (using the scan() function), and output the first 5 words in the list.

https://intro-datascience.s3.us-east-2.amazonaws.com/positive-words.txt (https://intro-datascience.s3.us-east-2.amazonaws.com/positive-words.txt)

There should be 2006 positive words words, so you may need to clean up these lists a bit.

```
URL <- ("https://intro-datascience.s3.us-east-2.amazonaws.com/positive-words.txt")
positiveword <- scan(URL, character(0), sep = "\n")
positiveword <- positiveword[-1:-34]
```

B. Do the same for the the negative words list (there are 4783 negative words):

https://intro-datascience.s3.us-east-2.amazonaws.com/negative-words.txt (https://intro-datascience.s3.us-east-2.amazonaws.com/negative-words.txt)

```
URL1 <- ("https://intro-datascience.s3.us-east-2.amazonaws.com/negative-words.txt")
negativeword <- scan(URL, character(0), sep = "\n")
negativeword <- negativeword[-1:-34]
```

C. Using **dfm_match()** with the dfm and the positive word file you read in, and then **textstat_frequency()**, output the 10 most frequent positive words

```
positiveDFM <- dfm_match(SnoDFM, positiveword)
positiveFrequency <- textstat_frequency(positiveDFM)
positiveFrequency[1:10, c("feature","frequency")]
```

```
##    feature frequency
## 1     like        88
## 2    honor        47
## 3    great        43
## 4     good        28
## 5      fun        27
## 6   strong        25
## 7     best        23
## 8     love        21
## 9     work        21
## 10   clear        19
```

D. Use R to print out the total number of positive words in the name explanation.

```
print(nrow(positiveFrequency))
```

```
## [1] 211
```

E. Repeat that process for the negative words you matched. Which negative words were in the name explanation variable, and what is their total number?

```
negativeDFM <- dfm_match(SnoDFM, negativeword)
negativeFrequency <- textstat_frequency(negativeDFM)
negativeFrequency[1:10, c("feature","frequency")]
```

```
##    feature frequency
## 1     like        88
## 2    honor        47
## 3    great        43
## 4     good        28
## 5      fun        27
## 6   strong        25
## 7     best        23
## 8     love        21
## 9     work        21
## 10   clear        19
```

```
print(nrow(negativeFrequency))
```

```
## [1] 211
```

F. Write a comment describing what you found after exploring the positive and negative word lists. Which group is more common in this dataset?

```
# The negative frequency  list contains the most frequent negative terms, which are listed in de
scending order.The positive frequency list contains the most frequent positive terms, which are
listed in descending order.
```

G. Complete the function below, so that it returns a sentiment score (number of positive words - number of negative words)

```
doMySentiment <- function(posWords, negWords, stringToAnalyze ) {
sentimentScore = match(stringToAnalyze, positiveword,nomatch=0 )-match(stringToAnalyze,
negativeword,nomatch=0)
  return(sentimentScore)
}
```

H. Test your function with the string “This book is horrible”

```
doMySentiment(positiveword, negativeword, "This book is horrible")
```

```
## [1] 0
```

I. Use the syuzhet package, to calculate the sentiment of the same phrase (“This book is horrible”), using syuzhet’s **get_sentiment()** function, using the afinn method. In AFINN, words are scored as integers from -5 to +5:

```
#install.packages("syuzhet")
library(syuzhet)

get_sentiment("This book is horrible", method="afinn")
```

```
## [1] -3
```

In a block comment, compare the results of your function with the get_sentiment function

```
#The sentiment score of "This book is horrible" is -3 according to the AFINN approach, and I rec
eived a sentiment score of 0 from doMySentiment function.
```