# Intro to Data Science - HW 11



```
# Enter your name here: Prajakta Mane
```

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

```
# 1. I did this homework by myself, with help from the book and the professor.
```

**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.

```
#Syracuse brought the new snow plows and now they are having the new naming contest for the ten
 snow plows.
#In December, the winning names were announced.
#Those names included: Below Zero Hero, Beast of the East, Control Salt Delete, Golden Snowball,
Lake Effect Crusher, Jocko, Abominable, Blizzard Beater, Winged Warrior, and Salt City Express
#There were 2000 submissions. Some are duplicates so total 1948 observations were unique.
#some submissions have the similar game and out of those only 2 people submitted for the winning
game
```

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(quanteda)
```

```
## Warning: package 'quanteda' was built under R version 4.1.2
```

```
## Package version: 3.1.0
## Unicode version: 13.0
## ICU version: 69.1
```

```
## Parallel computing: 4 of 4 threads used.
```

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

```
library(quanteda.textplots)
```

```
## Warning: package 'quanteda.textplots' was built under R version 4.1.2
```

```
library(quanteda.textstats)
```

```
## Warning: package 'quanteda.textstats' was built under R version 4.1.2
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.1.2
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
df <- readr :: 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?

```
glimpse(df)
```

```
## Rows: 1,907
## Columns: 5
## $ submission_number         <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 1~
## $ submitter_name_anonymized <chr> "kjlt9cua", "KXKaabXN", "kjlt9cua", "Rv9sODq~
## $ snowplow_name             <chr> "rudolph", "salt life", "blizzard", "butter"~
## $ meaning                   <chr> "The red nose cuts through any storm.", "We ~
## $ winning_name              <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FA~
```

```
#meaning of each submitted snowplow name can be shown by the column snowplow name.
```

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**.

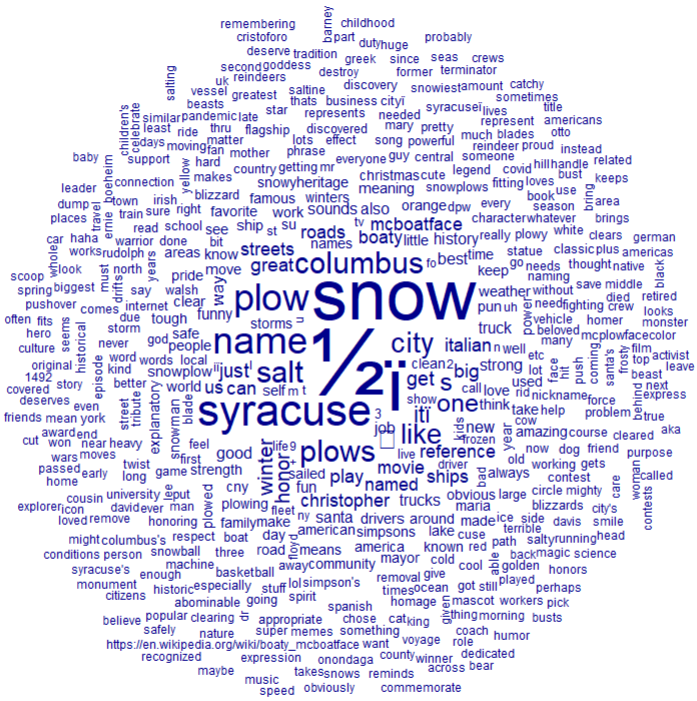
```
#install.packages("quanteda")
library(quanteda)
dfCorpus <- corpus(df$meaning, docnames=df$submission_number)
```

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

```
#from the column "meaning" it creates separate documents as key value pair
toks <- tokens(dfCorpus, remove_punct=TRUE)
#removes punctuation.
toks_nostop <- tokens_select(toks, pattern = stopwords("en"),
selection = "remove")
#removes stop words
df_DFM <- dfm(toks_nostop, tolower = TRUE) #creating the dfm
```

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:

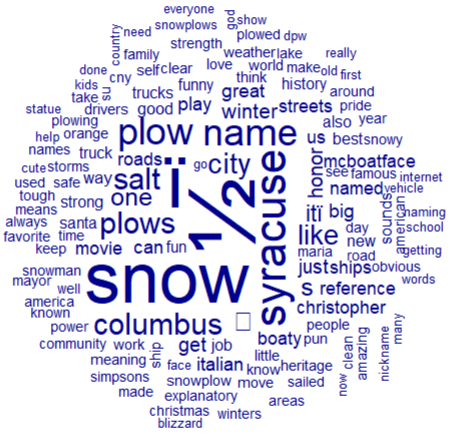
```
#install.packages("quanteda.textplots")
library(quanteda.textplots)
textplot_wordcloud(df_DFM, min_count = 2) #plotting the word cloud using the package quanteda.te
xtplots
```




```
#The terms are repeated in the corpus more than twice or at least twice
```

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

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




```
#here we are plotting the words who are repeating minimum of 10 times
```

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(df_DFM,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       143    5     137   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.

```
#It displays the most often repeated terms, with 1/2 and d I being the most commonly repeated. I
t also provides a list of all the repeated terms that appear in the c orpus's list.
```

# Part 2: Analyze the sentiment of the descriptions

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

I. 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"
posWords <- scan(URL, character(0), sep = "\n")
posWords <- posWords[-1:-34]# loading the positive words
```

J. 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"
negWords <- scan(URL1,character(0),sep= "\n")
negWords <- negWords[-1:-34] #getting the negative words
#negWords
```

J. 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

```
posDFM <- dfm_match(df_DFM, posWords)
#it matches the positive words with the DFM dataframe
posFreq <- textstat_frequency(posDFM)
#it tells the frequency of positive words in the documents
posFreq[1:10,]
```

```
##    feature frequency rank docfreq group
## 1     like        88    1      85   all
## 2    honor        47    2      47   all
## 3    great        43    3      43   all
## 4     good        28    4      28   all
## 5      fun        27    5      24   all
## 6   strong        25    6      25   all
## 7     best        23    7      22   all
## 8     love        21    8      21   all
## 9     work        21    8      21   all
## 10   clear        19   10      19   all
```

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

```
 sum(posFreq$frequency)
```

```
## [1] 866
```

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

```
negDFM <- dfm_match(df_DFM,negWords)
#it matches the negative words with the DFM dataframe
negFreq <- textstat_frequency(negDFM)
#it tells the frequency of negative words in the documents.
sum(negFreq$frequency)
```

```
## [1] 255
```

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

```
#There are 866 good words and just 255 negative ones in the dictionary. The positive word catego
ry is more prevalent
```

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

```
doMySentiment <- function(posWords, negWords, stringToAnalyze ) {
  stringToAnalyze_corpus <- corpus(stringToAnalyze)
toks <- tokens(stringToAnalyze_corpus, remove_punct=TRUE)
#removes punctuation.
toks_nostop <- tokens_select(toks, pattern = stopwords("en"),
selection = "remove")
#removes stop words
df_sa_DFM <- dfm(toks_nostop, tolower = TRUE) #creating the dfm
pos_sa_dfm <- dfm_match(df_sa_DFM,posWords)
# pos_sa_freq<- textstat_frequency(pos_sa_dfm)
neg_sa_DFM <- dfm_match(df_sa_DFM,negWords)
#it matches the negative words with the tweetDFM dataframe
# neg_sa_Freq <- textstat_frequency(neg_sa_DFM)
sentimentScore <- sum(pos_sa_dfm) - sum(neg_sa_DFM)

  return(sentimentScore)
}
```

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

```
doMySentiment(posWords, negWords, "This book is horrible")
```

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

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)
```

```
## Warning: package 'syuzhet' was built under R version 4.1.2
```

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

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