Intro to Data Science - HW 9


```
# Enter your name here: Benjamin Tisinger
```

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

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

```
library(quanteda)
```

```
## Package version: 3.2.3
## Unicode version: 13.0
## ICU version: 69.1
```

```
## Parallel computing: 8 of 8 threads used.
```

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

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

```
## ── Attaching packages
## ─────────────────────────────────────
## tidyverse 1.3.2 ──
```

```
## ✓ ggplot2 3.3.6      ✓ purrr   0.3.5
## ✓ tibble  3.1.8      ✓ dplyr   1.0.10
## ✓ tidyr   1.2.1      ✓ stringr 1.4.1
## ✓ readr   2.1.3      ✓ forcats 0.5.2
## ── Conflicts ──────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

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.

```
#Snowplow Naming Contest from the city of Syracuse. 10 New Snowplows named. Total of 1948 Entrie
s in the CSV File.
```

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)

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

```
##   submission_number submitter_name_anonymized       snowplow_name
## 1                 1                   kjlt9cua             rudolph
## 2                 2                   KXKaabXN           salt life
## 3                 3                   kjlt9cua            blizzard
## 4                 4                   Rv9sODqp              butter
## 5                 5                   zzcc5FDn santa's 10 reindeer
## 6                 6                   wOrKO7XI    plowy mcplowface
##
meaning
## 1                                                                                The red nose c
uts through any storm.
## 2 We may not be near the ocean like everyone else with the stickers that say Salt Life, but w
e have plenty of salt!
## 3                                                                                    This plow
can handle any storm.
## 4                                                      It's amazing how the snow plows thr
ough snow like butter!
## 5                                                             They can deliver through the
bad weather and snow.
## 6                                                                                       It
would be a great name
##   winning_name
## 1        FALSE
## 2        FALSE
## 3        FALSE
## 4        FALSE
## 5        FALSE
## 6        FALSE
```

C. Inspect the **df** dataframe – which column contains an explanation of the meaning of each submitted snowplow name? Transform that column into a **document-feature matrix**, using the **corpus()**, **tokens()**, **tokens_select(), and** dfm()** functions. Do not forget to **remove stop words**.

Hint: Make sure you have librariedd *quanteda*

```
str(df)
```

```
## 'data.frame':    1907 obs. of  5 variables:
##  $ submission_number        : int  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 ...
```

```
#Meaning Column holds Explanation of Snowplow Name
```

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

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

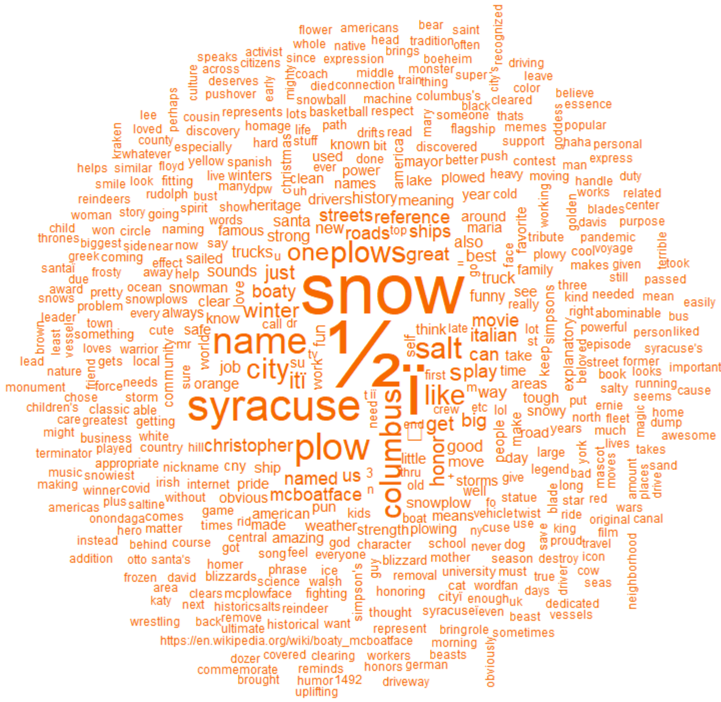
```
token <- tokens(df_corp, remove_punct = TRUE)
rmv_stop_word <- tokens_select(token, pattern = stopwords("en"), selection = "remove")

df_fin <- dfm(rmv_stop_word)
```

D. Plot a **word cloud**, where a word is only represented if it appears **at least 2 times** . **Hint:** use **textplot_wordcloud()**:

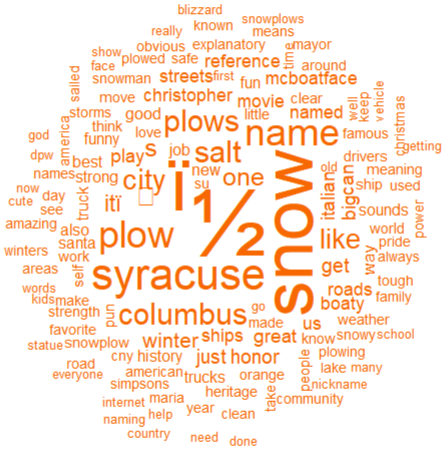
Hint: Make sure you have librariedd (and installed if needed) *quanteda.textplots*

```
textplot_wordcloud(min_count = 2, df_fin, color="#F76900")
```



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

```
textplot_wordcloud(min_count = 10, df_fin, color="#F76900")
```




```
# The word cloud gets smaller because these are the words using 10+ times. Will keep getting sma
ller with min_count increase.
```

F. What are the top words in the word cloud? Explain in a brief comment.

```
# Top Words -> Snow, 1/2 , Plow, Syracuse, Salt, Name, Columbus ,Great, One, Honor
```

Part 2: Analyze the sentiment of the descriptions

A. Create a **named list of word counts by frequency**.

output the 10 most frequent words (their word count and the word).

Hint: use **textstat_frequency()** from the *quanteda.textstats* package.

```
library(quanteda.textstats)

textstat_frequency(n=10,df_fin)
```

```
##     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
```

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

```
#For the Most Part, the words look very good. I am a little confused by the "I.." with the 2 dot
s and what the role its plays as. I am also unsure of the 1/2 and its frequency of 432. I think
if we ruled those weird words outs we are left with snow as the top answer which makes the most
sense.
```

Part 3: Match the 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. Do the same for the the negative words 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)
   https://intro-datascience.s3.us-east-2.amazonaws.com/negative-words.txt (https://intro-datascience.s3.us-east-2.amazonaws.com/negative-words.txt)

There should be 2006 positive words and 4783 negative words, so you may need to clean up these lists a bit.

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

positive_clean <- positive[0:-34]
positive_5 <- positive_clean[1:5]

show(positive_5)
```

```
## [1] "a+"        "abound"    "abounds"   "abundance" "abundant"
```

```
length(positive_clean)
```

```
## [1] 2006
```

```
negative_link <- "https://intro-datascience.s3.us-east-2.amazonaws.com/negative-words.txt"
negative <- scan(negative_link,sep="\n",character(0))

negative_clean <- negative[0:-34]
negative_5 <- negative_clean[1:5]

show(negative_5)
```

```
## [1] "2-faced"    "2-faces"    "abnormal"   "abolish"    "abominable"
```

```
length(negative_clean)
```

```
## [1] 4783
```

B. Use **dfm_match()** to match the words in the dfm with the words in posWords). Note that **dfm_match()** creates a new dfm.

Then pass this new dfm to the **textstat_frequency()** function to see the positive words in our corpus, and how many times each word was mentioned.

```
posi_dfm <- dfm_match(df_fin,positive_clean)
posi_freq <- textstat_frequency(posi_dfm)
head(posi_freq,2)
```

```
##   feature frequency rank docfreq group
## 1    like        88    1      85   all
## 2   honor        47    2      47   all
```

C. Sum all the positive words

```
sum(posi_freq$frequency)
```

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

D. Do a similar analysis for the negative words - show the 10 most requent negative words and then sum the negative words in the document.

```
negative_dfm <- dfm_match(df_fin,negative_clean)
negative_freq <- textstat_frequency(negative_dfm)
head(negative_freq,2)
```

```
##   feature frequency rank docfreq group
## 1   funny        25    1      25   all
## 2    cold         8    2       8   all
```

```
sum(negative_freq$frequency)
```

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

E. Write a comment describing what you found after matching positive and negative words. Which group is more common in this dataset? Might some of the negative words not actually be used in a negative way? What about the positive words?

```
#From My Conclusion, it shows that 866 Pos, and 255 Neg words were used. I think that there is a chance that some neg words may not actually be conceived as Negative. I think this of course is purely up to user discretion, Especially when naming something like a snowplow. I do think this was a really nice exercise to scrub some data and compare it against another set that matches keywords (good or bad).
```