# Intro to Data Science - HW 11

##### Copyright 2021, Jeffrey Stanton, Jeffrey Saltz, Christopher Dunham, and Jasmina Tacheva

# Enter your name here: Hrishikesh Telang

### 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

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

#This article is about a snowplowing naming contest organized my Mayor Walsh of Syracuse.   
#The city of Syracuse purchased 10 new snowplows that needed to be named,  
#so, everyone were invited to draft their own creative names for these snowplows.   
#The city received close to ~1910 names and announced 10 winning names in December.

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

library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──

## ✓ ggplot2 3.3.5 ✓ purrr 0.3.4  
## ✓ tibble 3.1.6 ✓ dplyr 1.0.7  
## ✓ tidyr 1.1.4 ✓ stringr 1.4.0  
## ✓ readr 2.1.0 ✓ forcats 0.5.1

## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(readr)  
df <- 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.

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

str(df)

## spec\_tbl\_df [1,907 × 5] (S3: spec\_tbl\_df/tbl\_df/tbl/data.frame)  
## $ submission\_number : num [1:1907] 1 2 3 4 5 6 7 8 9 10 ...  
## $ submitter\_name\_anonymized: chr [1:1907] "kjlt9cua" "KXKaabXN" "kjlt9cua" "Rv9sODqp" ...  
## $ snowplow\_name : chr [1:1907] "rudolph" "salt life" "blizzard" "butter" ...  
## $ meaning : chr [1:1907] "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 [1:1907] FALSE FALSE FALSE FALSE FALSE FALSE ...  
## - attr(\*, "spec")=  
## .. cols(  
## .. submission\_number = col\_double(),  
## .. submitter\_name\_anonymized = col\_character(),  
## .. snowplow\_name = col\_character(),  
## .. meaning = col\_character(),  
## .. winning\_name = col\_logical()  
## .. )  
## - attr(\*, "problems")=<externalptr>

head(df)

## # A tibble: 6 × 5  
## submission\_number submitter\_name\_anonymized snowplow\_name meaning winning\_name  
## <dbl> <chr> <chr> <chr> <lgl>   
## 1 1 kjlt9cua rudolph The re… FALSE   
## 2 2 KXKaabXN salt life We may… FALSE   
## 3 3 kjlt9cua blizzard This p… FALSE   
## 4 4 Rv9sODqp butter It's a… FALSE   
## 5 5 zzcc5FDn santa's 10 r… They c… FALSE   
## 6 6 wOrKO7XI plowy mcplow… It wou… FALSE

tail(df)

## # A tibble: 6 × 5  
## submission\_number submitter\_name\_anonymized snowplow\_name meaning winning\_name  
## <dbl> <chr> <chr> <chr> <lgl>   
## 1 1941 35KBUE6l bubba "It so… FALSE   
## 2 1942 35KBUE6l bart "I pic… FALSE   
## 3 1943 OIhNAvlb optimus "His c… FALSE   
## 4 1944 9N87xMNL jocko "James… TRUE   
## 5 1945 7F1njdoT santa maria "Remem… FALSE   
## 6 1948 BvIgeaPM santa maria "Santa… FALSE

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)

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

## Parallel computing: 8 of 8 threads used.

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

tweetCorpus <- corpus(df$meaning, docnames=df$submission\_number)

## Warning: NA is replaced by empty string

tweetCorpus

## Corpus consisting of 1,907 documents.  
## 1 :  
## "The red nose cuts through any storm."  
##   
## 2 :  
## "We may not be near the ocean like everyone else with the sti..."  
##   
## 3 :  
## "This plow can handle any storm."  
##   
## 4 :  
## "It's amazing how the snow plows through snow like butter!"  
##   
## 5 :  
## "They can deliver through the bad weather and snow."  
##   
## 6 :  
## "It would be a great name"  
##   
## [ reached max\_ndoc ... 1,901 more documents ]

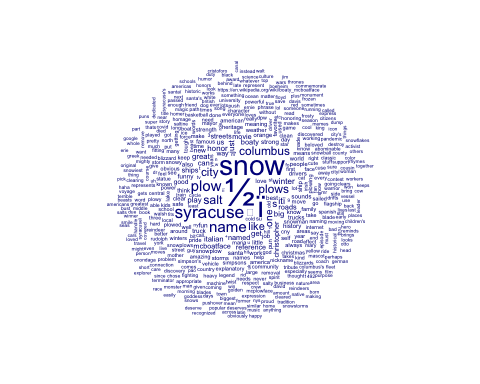
toks <- tokens(tweetCorpus, remove\_punct=TRUE)  
#This code removes punctuation.  
  
toks\_nostop <- tokens\_select(toks, pattern = stopwords("en"), selection = "remove")  
#We take our list of tokens and remove all the stopwords such as 'I", 'a' as we don't need them.  
  
dfDFM <- dfm(toks\_nostop, tolower = TRUE)  
dfDFM

## Document-feature matrix of: 1,907 documents, 2,807 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 ... 2,797 more features ]

#Document-feature matrix of: 1,907 documents, 2,807 features (99.83% sparse).

1. 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:

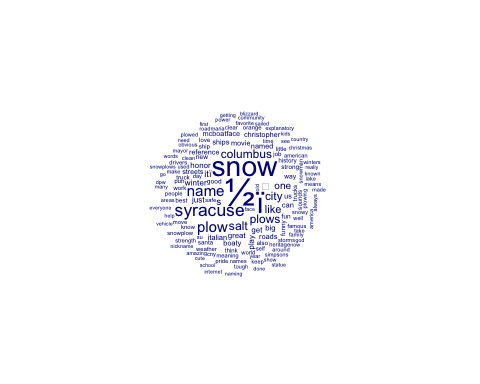
#install.packages("quanteda.textplots")  
library(quanteda.textplots)  
textplot\_wordcloud(dfDFM, min\_count = 1)



#The wordcloud portrays the frequency of a given word, shown by its size in the diagram.   
#Snow and '1/2' is the most frequent strings and this is shown by its size in the wordcloud.   
#Only words which have appeared atleast twice have been included

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

textplot\_wordcloud(dfDFM, min\_count = 10)



1. 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)  
head(textstat\_frequency(dfDFM))

## 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

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

#We see the 10 most freuqntly occuring words/strings. The string '1/2' is the most frequently used subset

## Part 2: Analyze the sentiment of the descriptions

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

1. 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>

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]  
#posWords  
#This does a frequency match between a positive word and all the documents in corpusDFM.  
#Only the positive words from the original corpus and the URL will be included

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>

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

1. 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(dfDFM, posWords)   
#This code matches the positive words with the tweetDFM dataframe  
posFreq <- textstat\_frequency(posDFM)  
#This code 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

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

count(data.frame(textstat\_frequency(posDFM)))[1, 'n']

## [1] 211

1. 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(dfDFM, negWords)  
#This code matches the negative words with the tweetDFM dataframe  
negFreq <- textstat\_frequency(negDFM)  
#This code tells the frequency of negative words in the documents.  
negFreq

## feature frequency rank docfreq group  
## 1 funny 25 1 25 all  
## 2 cold 8 2 8 all  
## 3 twist 8 2 8 all  
## 4 hard 7 4 7 all  
## 5 abominable 6 5 6 all  
## 6 problem 6 5 6 all  
## 7 bad 5 7 5 all  
## 8 destroy 5 7 5 all  
## 9 died 5 7 5 all  
## 10 bust 4 10 4 all  
## 11 dump 4 10 4 all  
## 12 frozen 4 10 4 all  
## 13 monster 4 10 4 all  
## 14 terrible 4 10 4 all  
## 15 blow 3 15 3 all  
## 16 busts 3 15 3 all  
## 17 crush 3 15 3 all  
## 18 silly 3 15 3 all  
## 19 bash 2 19 2 all  
## 20 challenging 2 19 2 all  
## 21 chilly 2 19 2 all  
## 22 crazy 2 19 2 all  
## 23 difficult 2 19 2 all  
## 24 dirt 2 19 2 all  
## 25 erase 2 19 2 all  
## 26 evil 2 19 2 all  
## 27 fear 2 19 2 all  
## 28 joke 2 19 2 all  
## 29 killed 2 19 2 all  
## 30 loud 2 19 2 all  
## 31 messes 2 19 2 all  
## 32 miser 2 19 2 all  
## 33 protest 2 19 2 all  
## 34 rough 2 19 2 all  
## 35 slow 2 19 2 all  
## 36 stuck 2 19 2 all  
## 37 abused 1 37 1 all  
## 38 adversary 1 37 1 all  
## 39 alarm 1 37 1 all  
## 40 apocalypse 1 37 1 all  
## 41 assault 1 37 1 all  
## 42 avalanche 1 37 1 all  
## 43 badly 1 37 1 all  
## 44 bleeds 1 37 1 all  
## 45 cancer 1 37 1 all  
## 46 cloud 1 37 1 all  
## 47 cloudy 1 37 1 all  
## 48 comical 1 37 1 all  
## 49 complaining 1 37 1 all  
## 50 crisis 1 37 1 all  
## 51 critical 1 37 1 all  
## 52 damage 1 37 1 all  
## 53 damn 1 37 1 all  
## 54 dark 1 37 1 all  
## 55 dead 1 37 1 all  
## 56 deadly 1 37 1 all  
## 57 death 1 37 1 all  
## 58 defiance 1 37 1 all  
## 59 delay 1 37 1 all  
## 60 despise 1 37 1 all  
## 61 destroyer 1 37 1 all  
## 62 die 1 37 1 all  
## 63 difficulty 1 37 1 all  
## 64 disabled 1 37 1 all  
## 65 disdain 1 37 1 all  
## 66 disorder 1 37 1 all  
## 67 disrespect 1 37 1 all  
## 68 dope 1 37 1 all  
## 69 doubt 1 37 1 all  
## 70 dreary 1 37 1 all  
## 71 drunk 1 37 1 all  
## 72 dumb 1 37 1 all  
## 73 dumps 1 37 1 all  
## 74 enemies 1 37 1 all  
## 75 evasion 1 37 1 all  
## 76 excuse 1 37 1 all  
## 77 fall 1 37 1 all  
## 78 fallen 1 37 1 all  
## 79 fatally 1 37 1 all  
## 80 fierce 1 37 1 all  
## 81 forged 1 37 1 all  
## 82 freezing 1 37 1 all  
## 83 frost 1 37 1 all  
## 84 hardships 1 37 1 all  
## 85 harsh 1 37 1 all  
## 86 hates 1 37 1 all  
## 87 horrible 1 37 1 all  
## 88 ignorance 1 37 1 all  
## 89 inadequacy 1 37 1 all  
## 90 inclement 1 37 1 all  
## 91 inexorable 1 37 1 all  
## 92 infamous 1 37 1 all  
## 93 intimidating 1 37 1 all  
## 94 ironic 1 37 1 all  
## 95 kills 1 37 1 all  
## 96 limit 1 37 1 all  
## 97 limited 1 37 1 all  
## 98 lone 1 37 1 all  
## 99 loot 1 37 1 all  
## 100 manipulate 1 37 1 all  
## 101 mar 1 37 1 all  
## 102 misery 1 37 1 all  
## 103 misfit 1 37 1 all  
## 104 miss 1 37 1 all  
## 105 mobster 1 37 1 all  
## 106 monstrous 1 37 1 all  
## 107 moody 1 37 1 all  
## 108 myth 1 37 1 all  
## 109 naive 1 37 1 all  
## 110 nasty 1 37 1 all  
## 111 object 1 37 1 all  
## 112 outbreak 1 37 1 all  
## 113 pander 1 37 1 all  
## 114 pig 1 37 1 all  
## 115 pity 1 37 1 all  
## 116 poorly 1 37 1 all  
## 117 puppet 1 37 1 all  
## 118 restriction 1 37 1 all  
## 119 rhetoric 1 37 1 all  
## 120 rival 1 37 1 all  
## 121 rivalry 1 37 1 all  
## 122 ruining 1 37 1 all  
## 123 ruins 1 37 1 all  
## 124 rust 1 37 1 all  
## 125 sarcasm 1 37 1 all  
## 126 scare 1 37 1 all  
## 127 scrap 1 37 1 all  
## 128 self-interest 1 37 1 all  
## 129 sour 1 37 1 all  
## 130 standstill 1 37 1 all  
## 131 suffer 1 37 1 all  
## 132 suffering 1 37 1 all  
## 133 tense 1 37 1 all  
## 134 toughness 1 37 1 all  
## 135 treacherous 1 37 1 all  
## 136 trickery 1 37 1 all  
## 137 undesirable 1 37 1 all  
## 138 unfortunately 1 37 1 all  
## 139 unfriendly 1 37 1 all  
## 140 unknown 1 37 1 all  
## 141 unpleasant 1 37 1 all  
## 142 unusually 1 37 1 all  
## 143 upset 1 37 1 all  
## 144 virus 1 37 1 all  
## 145 worry 1 37 1 all  
## 146 worse 1 37 1 all  
## 147 worst 1 37 1 all  
## 148 zombie 1 37 1 all

#Words such as funny, cold, twist, abomiable are in the negative words list in the name variable.  
#Funny is the most negative word used.

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

count(data.frame(textstat\_frequency(negDFM)))[1, 'n']

## [1] 148

#Positive words are more common as we have more numbers with higher frequency for each word.

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

library(tidyverse)  
library(quanteda.textstats)  
library(quanteda)  
  
doMySentiment <- function(posWords, negWords, stringToAnalyze ) {  
   
 toks <- tokens(stringToAnalyze, remove\_punct=TRUE)  
 toks\_nostop <- tokens\_select(toks, pattern = stopwords("en"),   
 selection = "remove")  
 meaningDFM <- dfm(toks\_nostop)  
   
 out<-tryCatch(  
   
 {  
 posDFM <- dfm\_match(meaningDFM, posWords)  
 posFreq <- textstat\_frequency(posDFM)  
 sum\_pos<- sum(posFreq$frequency)  
 },  
 error=function(err){  
 return (0)  
 }  
 )  
   
 out1<-tryCatch(  
   
 {  
 negDFM <- dfm\_match(meaningDFM, negWords)  
 negFreq <- textstat\_frequency(negDFM)  
 sum\_neg<-(sum(negFreq$frequency))  
 },  
 error=function(err){  
 return (0)  
 }  
 )  
 sentimentScore =ifelse(out==0,ifelse(out1==0,0,-out1),  
 ifelse(out1==0,out,out-out1))  
  
 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)  
  
get\_sentiment("This book is horrible", method="afinn")

## [1] -3