Intro to Data Science - Lab 11

Week 11 - Text Mining

```
# Enter your name here: Hendi Kushta
```

Please include nice comments.

Instructions:

Run the necessary code on your own instance of R-Studio.

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

```
# 1. I did this lab assignment 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 exercise, we work with a small number of social media posts on the topic of climate change. Make sure to install and library the **tidiverse**, **quanteda**, **quanteda.textplots**, and **quanteda.textstats** packages before starting the exercise.

```
# install.packages("tidyverse")
# install.packages("quanteda")
# install.packages("quanteda.textplots")
# install.packages("quanteda.textstats")
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v purrr   0.3.5
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
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: 3.2.3
## Unicode version: 13.0
## ICU version: 66.1
## Parallel computing: 16 of 16 threads used.
## See https://quanteda.io for tutorials and examples.
```

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

1. Read in the following data set with **read_csv()**: https://intro-datascience.s3.us-east-2.amazonaws.com/ClimatePosts.csv The name of the file is ** ClimatePosts.csv **. Store the data in a data frame called** tweetDF**. Use** str(tweetDF)** to summarize the data. Add a comment describing what you see. Make sure to explain what each of the three variables contains.

```
tweetDF <- read_csv("https://intro-datascience.s3.us-east-2.amazonaws.com/ClimatePosts.csv")
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## Rows: 18 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): ID, Tweet
## dbl (1): Skeptic
##
## 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.
```

```
str(tweetDF)
```

```
## spc_tbl_ [18 x 3] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ID     : chr [1:18] "climatechange" "billmckibben" "megancollins" "neiltyson" ...
##  $ Skeptic: num [1:18] 0 0 0 0 0 0 0 0 0 1 ...
##  $ Tweet  : chr [1:18] "BREAKING: Iran soars to record of 129 degrees - near hottest ever reliably m
##  - attr(*, "spec")=
##   .. cols(
##   ..   ID = col_character(),
##   ..   Skeptic = col_double(),
##   ..   Tweet = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
# 18 rows 3 variables. data is unstructered.
```

2. Use the **corpus** commands to turn the text variable into a quanteda corpus. You can use the IDs as the document titles with the following command: `tweetCorpus <- corpus(tweetDF$Tweet, docnames=tweetDF$ID)`

```
tweetCorpus <- corpus(tweetDF$Tweet, docnames=tweetDF$ID)
```

3. Next, convert the corpus into a **document-feature matrix (DFM)**. Before you do that you can use ** tokens ** to remove punctuation and stop words. Use this code and comment each line:

```
toks <- tokens(tweetCorpus, remove_punct=TRUE)
toks_nostop <- tokens_select(toks, pattern = stopwords("en"), selection = "remove")
```

Here s a command that will create the DFM: `tweetDFM <- dfm(toks_nostop, tolower = TRUE )`

```
toks <- tokens(tweetCorpus, remove_punct=TRUE)
toks_nostop <- tokens_select(toks, pattern = stopwords("en"), selection = "remove")
tweetDFM <- dfm(toks_nostop, tolower = TRUE )
```

4. Type **tweetDFM** in the code cell to find out the basic characteristics of the DFM (the number of

terms, the number of documents, and the sparsity of the matrix). Write a comment describing what you observe.

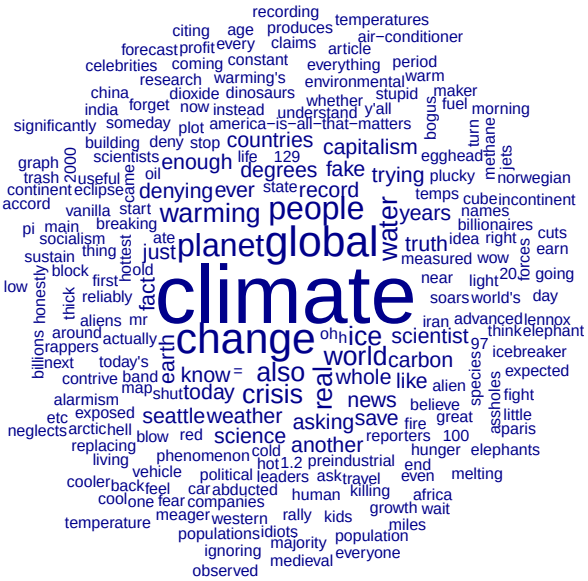
```
tweetDFM
```

```
## Document-feature matrix of: 18 documents, 223 features (93.20% sparse) and 0 docvars.
##                  features
## docs              breaking iran soars record 129 degrees near hottest ever
##   climatechange          1    1     1      1   1       1    1       1    1
##   billmckibben           0    0     0      0   0       1    0       0    0
##   megancollins           0    0     0      0   0       0    0       0    0
##   neiltyson              0    0     0      0   0       0    0       0    1
##   johnbiehl              0    0     0      0   0       0    0       0    0
##   ScottWesterfeld        0    0     0      0   0       0    0       0    0
##                  features
## docs              reliably
##   climatechange          1
##   billmckibben           0
##   megancollins           0
##   neiltyson              0
##   johnbiehl              0
##   ScottWesterfeld        0
## [ reached max_ndoc ... 12 more documents, reached max_nfeat ... 213 more features ]
```

```
# we see how many times each of the features is reapeated in each of the
# documents.
```

5. Create a **wordcloud** from the DFM using the following command. Write a comment describing notable features of the wordcloud: `textplot_wordcloud(tweetDFM, min_count = 1)`

```
textplot_wordcloud(tweetDFM, min_count = 1)
```



6. Using **textstat_frequency()** from the **quanteda.textstats** package, show the 10 most frequent words, and how many times each was used/mentioned.

```
textstat_frequency(tweetDFM, 10)
```

```
##    feature frequency rank docfreq group
## 1  climate        13    1       8   all
```

```
## 2   global         6    2       5   all
## 3   change         6    2       6   all
## 4   planet         4    4       3   all
## 5   people         4    4       3   all
## 6     also         3    6       3   all
## 7   crisis         3    6       2   all
## 8  warming         3    6       3   all
## 9     real         3    6       3   all
## 10   world         3    6       3   all
```

7. Next, we will read in **dictionaries** of positive and negative words to see what we can match up to the text in our DFM. Here s a line of code for reading in the list of positive words:

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

Create a similar line of code to read in the negative words, with the following URL: https://intro-datascience.s3.us-east-2.amazonaws.com/negative-words.txt There should be 2006 positive words and 4783 negative words.

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

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

8. Explain what the following lines of code does and comment each line. Then add similar code for the negative words.

```
posDFM <- dfm_match(tweetDFM, posWords)
posFreq <- textstat_frequency(posDFM)
```

```
#finds if positive words are used in these tweets
#and displays their ferquencies
posDFM <- dfm_match(tweetDFM, posWords)
posFreq <- textstat_frequency(posDFM)
#finds if negative words are used in these tweets
#and displays their ferquencies
negDFM <- dfm_match(tweetDFM, negWords)
negFreq <- textstat_frequency(negDFM)
```

9. Explore **posFreq** and **negFreq** using **str()** or **glimpse()**. Explain the fields in these data frames

```
glimpse(posFreq)
```

```
## Rows: 12
## Columns: 5
## $ feature   <chr> "enough", "like", "advanced", "cool", "great", "hot", "hotte~
## $ frequency <dbl> 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
## $ rank      <dbl> 1, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3
## $ docfreq   <dbl> 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
## $ group     <chr> "all", "all", "all", "all", "all", "all", "all", "all", "all~
```

```
glimpse(negFreq)
```

```
## Rows: 12
## Columns: 5
## $ feature   <chr> "enough", "like", "advanced", "cool", "great", "hot", "hotte~
## $ frequency <dbl> 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
## $ rank      <dbl> 1, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3
## $ docfreq   <dbl> 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
## $ group     <chr> "all", "all", "all", "all", "all", "all", "all", "all", "all~
```

10. Output the 10 most frequently occurring positive and negative words including how often each occurred.

```
posFreq <- posFreq %>% arrange(frequency)
head(posFreq,10)
```

```
##     feature frequency rank docfreq group
## 3  advanced         1    3       1   all
## 4      cool         1    3       1   all
## 5     great         1    3       1   all
## 6       hot         1    3       1   all
## 7   hottest         1    3       1   all
## 8  reliably         1    3       1   all
## 9     right         1    3       1   all
## 10   useful         1    3       1   all
## 11     warm         1    3       1   all
## 12      wow         1    3       1   all
```

```
negFreq <- negFreq %>% arrange(frequency)
head(negFreq,10)
```

```
##     feature frequency rank docfreq group
## 3  advanced         1    3       1   all
## 4      cool         1    3       1   all
## 5     great         1    3       1   all
## 6       hot         1    3       1   all
## 7   hottest         1    3       1   all
## 8  reliably         1    3       1   all
## 9     right         1    3       1   all
## 10   useful         1    3       1   all
## 11     warm         1    3       1   all
## 12      wow         1    3       1   all
```