Intro to Data Science - HW 12

Copyright 2024, Jeffrey Stanton, Jeffrey Saltz, Christopher Dunham, and Jasmina Tacheva

USE OF ARTIFICIAL INTELLIGENCE: While you are allowed to use generative AI as a learning tool for this assignment (e.g., to help with brainstorming, to see how to code a certain concept, to help identify flaws in reasoning, or to spot confusing or underdeveloped code segments), it must not be used to input homework questions, problems, or instructions directly, and you may not copy and paste any AI-generated content into your work. By submitting this assignment, you confirm that all the work is your own.

```
# Enter your name here: Aidan Rudick
```

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

```
# 4. I did this homework with help from ChatGPT or another generative
AI tool,
# but I followed the homework guidelines.
```

Text mining is essential in many industries because of the extensive use of text in customer interactions with company representatives. Even when customer interactions are through speech, modern speech-to-text algorithms are so advanced that transcripts of these interactions are often available. Therefore, a data scientist needs to be proficient in tools that can turn text into actionable insights.

In this homework, we will explore a TripAdvisor reviews dataset using the **quanteda** and **quanteda.textplots** packages. Make sure to install these packages before proceeding with the steps below:

Part 1: Load and visualize the data file

A. Read the data from the following URL into a dataframe called **reviews_df**:
https://data-science-intro.s3.us-east-2.amazonaws.com/TA_reviews.csv
(https://data-science-intro.s3.us-east-2.amazonaws.com/TA_reviews.csv)

```
library(readr)
reviews_df <- read_csv("https://data-science-intro.s3.us-east-2.amazonaws.com/TA_reviews.csv")
```

```
## Rows: 650 Columns: 13
## ── Column specification ────────────────────────────────────────────────
## Delimiter: ","
## chr (13): Name, Street Address, Location, Type, Reviews, No of Reviews, Comm...
##
## 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.
```

B. Inspect the dataframe. Which variable contains the text of the reviews?

```
str(reviews_df)
```

```
## spc_tbl_ [650 × 13] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Name           : chr [1:650] "Genesee Brew House" "Diana of Li
ttle Chef, Little Kitchen" "Frank Pepe Pizzeria Napoletana" "Dirt Can
dy" ...
##  $ Street Address  : chr [1:650] "25 Cataract St." "Not Specified"
"1955 Central Ave" "86 Allen St" ...
##  $ Location       : chr [1:650] "Rochester, NY" "New York City, N
Y" "Yonkers, NY" "New York City, NY 10002-3014" ...
##  $ Type           : chr [1:650] "American, Bar, Pub" "Dine With a
Local Chef, Italian, American" "Italian, Pizza, Vegetarian Friendly"
"American, Vegetarian Friendly, Vegan Options" ...
##  $ Reviews        : chr [1:650] "4.5 of 5 bubbles" "5 of 5 bubble
s" "4 of 5 bubbles" "4.5 of 5 bubbles" ...
##  $ No of Reviews   : chr [1:650] "807 reviews" "105 reviews" "185
reviews" "353 reviews" ...
##  $ Comments       : chr [1:650] "In fact the menu is pretty broa
d. I saw several varieties of mac’n cheese being enjoyed. The view of
the High F"| __truncated__ "This restaurant was recommended by our ho
st in NY...and I'm glad we went...Diana is just fantastic and food wa
s"| __truncated__ "A limited menu but a grat tasting white clam pizz
a.  If pizza and salad are what you crave, this is the place" "This w
as truly the worst meal we've ever eaten (4 of us dined this evenin
g). We stopped for a slice of pizza on"| __truncated__ ...
##  $ Contact Number  : chr [1:650] "+1 585-263-9200" "Not Available"
"+1 914-961-8284" "+1 212-228-7732" ...
##  $ Trip_advisor Url: chr [1:650] "https://www.tripadvisor.com//Res
taurant_Review-g48503-d3497245-Reviews-Genesee_Brew_House-Rochester_F
inger_Lakes_New_York.html" "https://www.tripadvisor.com//Restaurant_R
eview-g60763-d11752088-Reviews-Diana_of_Little_Chef_Little_Kitchen-Ne
w"| __truncated__ "https://www.tripadvisor.com//Restaurant_Review-g48
922-d2423597-Reviews-Frank_Pepe_Pizzeria_Napoletana-Yonkers_New_York.
html" "https://www.tripadvisor.com//Restaurant_Review-g60763-d1236468
-Reviews-Dirt_Candy-New_York_City_New_York.html" ...
##  $ Menu           : chr [1:650] "Check The Website for a Menu" "C
heck The Website for a Menu" "https://pepespizzeria.com/yonkers/" "Ch
eck The Website for a Menu" ...
##  $ Price_Range    : chr [1:650] "$$ - $$$" "$$ - $$$" "$$ - $$$"
"$$ - $$$" ...
##  $ State          : chr [1:650] "NY" "NY" "NY" "NY" ...
##  $ City           : chr [1:650] "Rochester" "New York City" "Yonk
ers" "New York City" ...
```

```
##  - attr(*, "spec")=
##   .. cols(
##   ..   Name = col_character(),
##   ..   `Street Address` = col_character(),
##   ..   Location = col_character(),
##   ..   Type = col_character(),
##   ..   Reviews = col_character(),
##   ..   `No of Reviews` = col_character(),
##   ..   Comments = col_character(),
##   ..   `Contact Number` = col_character(),
##   ..   `Trip_advisor Url` = col_character(),
##   ..   Menu = col_character(),
##   ..   Price_Range = col_character(),
##   ..   State = col_character(),
##   ..   City = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
# Comment: Comments
```

C. Transform that column into a **document-feature matrix**, using the **corpus()**, **tokens(), tokens_select()**, and **dfm()** functions from the quanteda package. Make sure to remove stop words.

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

```
## Package version: 4.1.0
## Unicode version: 14.0
## ICU version: 71.1
```

```
## Parallel computing: disabled
```

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

```
corpus_reviews <- corpus(reviews_df$Comments)

tokens_reviews <- tokens(corpus_reviews)

tokens_reviews_clean <- tokens_select(tokens_reviews, pattern = stopw
ords("en"),
    selection = "remove")

dfm_reviews <- dfm(tokens_reviews_clean)

dfm_reviews
```

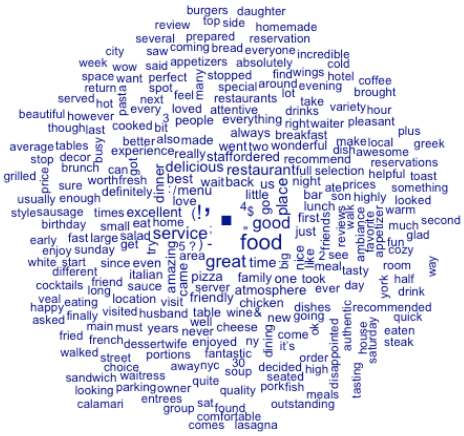
```
## Document-feature matrix of: 650 documents, 3,092 features (99.31%
sparse) and 0 docvars.
##        features
## docs    fact menu pretty broad . saw several varieties mac’n chees
e
##   text1    1    1      1     1 4   1       1         1     1
1
##   text2    0    0      0     0 9   0       0         0     0
0
##   text3    0    1      0     0 1   0       0         0     0
0
##   text4    0    0      0     0 5   0       0         0     0
0
##   text5    0    0      0     0 2   0       0         0     0
0
##   text6    0    0      0     0 1   0       0         0     0
0
## [ reached max_ndoc ... 644 more documents, reached max_nfeat ...
3,082 more features ]
```

D. Plot a word cloud where a word is only represented if it appears at least 10 times in the corpus. Use **textplot_wordcloud()** from the quanteda.textplots package:

```
# install.packages('quanteda.textplots')
library(quanteda.textplots)

dfm_reviews_trimmed <- dfm_trim(dfm_reviews, min_termfreq = 10)

textplot_wordcloud(dfm_reviews_trimmed)
```



E. Next, increase the minimum count to 50. What happens to the word cloud? **Explain in a comment**.

```
library(quanteda.textplots)
dfm_reviews_trimmed_50 <- dfm_trim(dfm_reviews, min_termfreq = 50)

textplot_wordcloud(dfm_reviews_trimmed_50)
```


```
# Comment: Increasing the minimum count to 50 means that only the words that
# appear at least 50 times in the corpus will be displayed.As a result, the
# word cloud will show fewer words compared to when the minimum count was 10.
```

F. What are the top 5 words in the word cloud?

Hint: use textstat_frequency() in the quanteda.textstats package

```
# install.packages('quanteda.textstats')
library(quanteda.textstats)

word_frequencies <- textstat_frequency(dfm_reviews_trimmed_50)

top_5_words <- head(word_frequencies, 5)

top_5_words
```

	feature <chr>	frequency <dbl>	rank <dbl>	docfreq <dbl>	group <chr>
1	.	2959	1	624	all
2	,	857	2	400	all
3	food	343	3	302	all
4	!	277	4	153	all
5	great	207	5	172	all

5 rows

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

```
# Comment: The sorted list of word counts reveals that common terms r
elated to
# restaurants, food, and service appear most frequently in the review
s. By
# setting a minimum frequency of 50, less common words are filtered o
ut,
# leaving the most significant terms that consistently occur across r
eviews.
```

Part 2: Analyze Review Sentiment

###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 10 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.

```
pos_words <- scan("https://intro-datascience.s3.us-east-2.amazonaws.com/positive-words.txt",
    what = "character", comment.char = ";")
head(pos_words, 10)
```

```
##  [1] "a+"          "abound"      "abounds"     "abundance"   "abundant"
##  [6] "accessable"  "accessible"  "acclaim"     "acclaimed"   "acclamation"
```

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)

```
neg_words <- scan("https://intro-datascience.s3.us-east-2.amazonaws.com/negative-words.txt",
    what = "character", comment.char = ";")
head(neg_words, 10)
```

```
##  [1] "2-faced"     "2-faces"     "abnormal"    "abolish"     "abominable"
##  [6] "abominably"  "abominate"   "abomination" "abort"       "aborted"
```

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

```
positive_matches <- dfm_match(dfm_reviews_trimmed_50, pos_words)

textstat_frequency(positive_matches)[1:10, ]
```

	feature <chr>	frequency <dbl>	rank <dbl>	docfreq <dbl>	group <chr>
1	great	207	1	172	all
2	good	183	2	146	all

	feature <chr>	frequency <dbl>	rank <dbl>	docfreq <dbl>	group <chr>
3	delicious	107	3	101	all
4	excellent	73	4	70	all
5	amazing	66	5	63	all
6	best	61	6	59	all
7	friendly	58	7	58	all
8	nice	53	8	52	all
NA	*NA*	*NA*	*NA*	*NA*	*NA*
NA.1	*NA*	*NA*	*NA*	*NA*	*NA*

1-10 of 10 rows

D. Print out the total number of positive words in the comments.

```
positive_count <- sum(positive_matches)
positive_count
```

```
## [1] 808
```

E. Repeat that process for the negative words you matched. What are the top 10 negative words in the comments, and what is the total number of negative words in the dataset?

```
# negative_matches <- dfm_match(dfm_reviews_trimmed_50, neg_words)

# textstat_frequency(negative_matches)[1:10, ]
```

```
# negative_count <- sum(negative_matches) negative_count
```

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

```
# Comment: After exploring the positive and negative matches in the d
ataset its
# clear that positive words are more common.
```

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

```
calculate_my_sentiment <- function(pos_words, neg_words, string_to_an
alyze) {
    tokens <- tokens(string_to_analyze, remove_punct = TRUE)

    dfm_reviews <- dfm(tokens)

    dfm_pos <- dfm_match(dfm_reviews, pos_words)

    dfm_neg <- dfm_match(dfm_reviews, neg_words)

    sentimentScore <- sum(dfm_pos) - sum(dfm_neg)
    return(sentimentScore)
}
```

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

```
example_string <- "This restaurant is horrible"

sentiment_score <- calculate_my_sentiment(pos_words, neg_words, examp
le_string)

print(sentiment_score)
```

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

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

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

example_phrase <- "This restaurant is horrible"

sentiment_afinn <- get_sentiment(example_phrase, method = "afinn")

print(sentiment_afinn)
```

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

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

```
# Comment: The calculate_my_sentiment() function calculates sentiment
by
# counting positive and negative words, while get_sentiment() (AFINN)
assigns
# specific scores to words based on their intensity. calculate_my_sen
timent()
# provides a basic sentiment score, while get_sentiment() offers a mo
re
# detailed score.
```

J. Examine the sentiment differences between reviews in Syracuse, Buffalo, and Ithaca. You can use subsetting and run the calculate_my_sentiment function for each city separately. Explain what you observe in a comment.

```
get_city_sentiment <- function(city) {
    reviews <- reviews_df[grepl(city, reviews_df$Location), "Comment
s"]
    mean(get_sentiment(reviews, method = "afinn"))
}

syracuse_sentiment <- get_city_sentiment("Syracuse")
buffalo_sentiment <- get_city_sentiment("Buffalo")
ithaca_sentiment <- get_city_sentiment("Ithaca")

print(syracuse_sentiment)
```

```
## [1] 98
```

```
print(buffalo_sentiment)
```

```
## [1] 60
```

```
print(ithaca_sentiment)
```

```
## [1] 81
```

```
# Comment: These values represent the average sentiment of the review
s for each
# city based on the AFINN method. A higher value indicates a more pos
itive
# overall sentiment, while a lower value suggests a more negative sen
timent.
# Based on the scores, Syracuse appears to have the most positive sen
timent,
# followed by Ithaca and then Buffalo.
```