Homework 1 DS-GA 1015

Alexandre Vives, N14948495

2/18/2020

Please list everyone you collaborated with on this assignment

---

```
# import libraries
library(quanteda)
library(quanteda.corpora)
library(quanteda.textstats)
library(stringdist)
library(gutenbergr)
library(tidyverse)
library(stylest)
library(sophistication)
library(pbapply)
```

QUESTION 1

a) Write a function in R to calculate the type-token ratio (TTR) of each of these speeches and report your findings.

```
# 1.1 load the corpus ---------------------
speeches <- corpus_subset(data_corpus_inaugural, President == "Reagan")
corpusinfo <- summary(speeches, n = ndoc(speeches)) # note n default is 100
corpusinfo$TTR <- corpusinfo$Types / corpusinfo$Tokens
print(corpusinfo$TTR)
```

```
## [1] 0.3244604 0.3179787
```

The TTR for both speeches are very similar, around 30%. More specifically, the speech of 1981 contains less unique and total words than the 1985 speech.

b) Create a document feature matrix of the two speeches, with no pre-processing other than to remove the punctuation–be sure to check the options on "dfm" in R as appropriate. Calculate the cosine similarity between the two documents with quanteda. Report your findings.

```
df <- dfm(tokens(speeches))
punc_df <- dfm(speeches, remove_punct = TRUE, tolower=FALSE)
textstat_simil(punc_df, margin = "documents", method = "cosine")
```

```
## textstat_simil object; method = "cosine"
##             1981-Reagan 1985-Reagan
## 1981-Reagan       1.000       0.956
## 1985-Reagan       0.956       1.000
```

The cosine similarity between both documents is 0.974, an expectedly high similarity.

QUESTION 2

a) Stemming the words?

```
stem_df <- dfm(speeches, stem = TRUE, remove_punct = TRUE, tolower=FALSE)
textstat_simil(stem_df, margin = "documents", method = "cosine")
```

```
## textstat_simil object; method = "cosine"
##             1981-Reagan 1985-Reagan
## 1981-Reagan       1.000       0.957
## 1985-Reagan       0.957       1.000
```

```
ntype(stem_df) / ntoken(stem_df)
```

```
## 1981-Reagan 1985-Reagan
##   0.3322368   0.3178627
```

Stemming should reduce the number of types significantly and keep tokens constant, therefore the overall TTR should go down. Additionally, it barely increases the cosine similarity.

b) Removing stop words?

```
stopwords_df <- dfm(speeches, remove = stopwords("english"), remove_punct = TRUE, tolower=FALSE)
textstat_simil(stopwords_df, margin = "documents", method = "cosine")
```

```
## textstat_simil object; method = "cosine"
##             1981-Reagan 1985-Reagan
## 1981-Reagan       1.000       0.668
## 1985-Reagan       0.668       1.000
```

```
ntype(stopwords_df) / ntoken(stopwords_df)
```

```
## 1981-Reagan 1985-Reagan
##   0.6608544   0.6059908
```

Removing stop words should reduce the total number of words (tokens) by a large amount because stop words occupy a good percentage of the whole text. Types will also be reduced but only by the number of unique stop words which is far less. Therefore TTR should increase.

c) Converting all words to lowercase?

```
lowercase_df <- dfm(speeches, remove_punct = TRUE, tolower=TRUE)
textstat_simil(lowercase_df, margin = "documents", method = "cosine")
```

```
## textstat_simil object; method = "cosine"
##             1981-Reagan 1985-Reagan
## 1981-Reagan       1.000       0.959
## 1985-Reagan       0.959       1.000
```

```
ntype(lowercase_df) / ntoken(lowercase_df)
```

```
## 1981-Reagan 1985-Reagan
##   0.3466283   0.3377535
```

Applying lower case should reduce the types (as it will combine words in beginning of sentence with that same eord in middle sentence) but should not affect tokens. Therefore, TTR should decrease.

d) Does tf-idf weighting make sense here? Calculate it and explain why or why not

```
weighted_dfm <- dfm_tfidf(punc_df)
topfeatures(weighted_dfm, n = 5, groups = docnames(weighted_dfm))
```

```
## $`1981-Reagan`
## ourselves        To    beyond     means     price
##   1.50515   1.20412   1.20412   1.20412   1.20412
##
## $`1985-Reagan`
## weapons nuclear  reduce     ago  better
## 1.80618 1.80618 1.50515 1.20412 1.20412
```

Using tf-idf is not very useful in this case because we are working with only two documents. This means that the fact that a word is in a document but not in the other it already gives it a lot of importance.

QUESTION 3

a) Write code in R to calculate the Euclidean distance between these sentences

```
sentence1 = "Nasa Mars rover: Perseverance robot all set for big test."
sentence2 = "NASA Lands Its Perseverance Rover on Mars."

dfm1 <- dfm(sentence1, remove_punct = TRUE)
dfm2 <- dfm(sentence2, remove_punct = TRUE)
textstat_dist(x=dfm1, y=dfm2, margin = "documents", method = "euclidean")
```

```
## textstat_dist object; method = "euclidean"
##       text1
## text1     3
```

I removed punctuation because it does not reflect anything regarding the meaning of the sentence and also lower cased because the word "NASA" is expressed differently in the questions.

The distance would be useful if we had to compare distances between sentences, as a reference value. But in this case, where we only have two sentences, we can only deduce that they are very similar (as it is obvious)

b) Write code in R to calculate the Manhattan distance between these sentences. Report your findings.

```
textstat_dist(x=dfm1, y=dfm2, margin = "documents", method = "manhattan")
```

```
## textstat_dist object; method = "manhattan"
##       text1
## text1     9
```

The Manhattan distance between these two sentences is 9.

c) Write code in R to calculate the cosine similarity between these sentences. Report your findings.

```
textstat_simil(x=dfm1, y=dfm2, margin = "documents", method = "cosine")
```

```
## textstat_simil object; method = "cosine"
##       text1
## text1 0.478
```

The cosine similarity between these two sentences is 0.478

d) Manually calculate the Levenshtein distance between robot and rover. Report your findings

robot rover

Procedure: switch b, o and t for v, e and r

```
stringdist('robot', 'rover', method = 'lv') #Sanity check
```

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

The Levenshtein distance between robot and rover is 3.

QUESTION 4

a) First you will need to get the data from Project Gutenberg using their gutenbergr package

```
n <- gutenberg_authors[,]

author_list <- c("Poe, Edgar Allan", "Twain, Mark", "Shelley, Mary Wollstonecraft",
                 "Doyle, Arthur Conan")

#Here a list of the gutenberg_id associated with the books is given below
book_list<-c(932,1064,1065,32037,74,76,86,91,84,6447,15238,18247,108,126,139,244)

#meta <- gutenberg_works(author == "Doyle, Arthur Conan") %>%?slice(1:4)

# Prepare data function
meta <- gutenberg_works(gutenberg_id == book_list)
meta <- meta %>% mutate(author = unlist(str_split(author, ","))[1] %>% tolower(.))
prepare_dt <- function(book_list, num_lines, removePunct = TRUE){
  meta <- gutenberg_works(gutenberg_id == book_list)
  meta <- meta %>% mutate(author = unlist(str_split(author, ","))[1] %>% tolower(.))
  texts <- lapply(book_list, function(x) gutenberg_download(x, mirror="http://mirrors.xmission.com/guten
                    #select(text) %>%
                    sample_n(500, replace=TRUE) %>%
                    unlist() %>%
                    paste(., collapse = " ") %>%
                    str_replace_all(., "^ +| +$|( ) +", "\\1"))

  # remove apostrophes
  texts <- lapply(texts, function(x) gsub("'|'", "", x))
  if(removePunct) texts <- lapply(texts, function(x)
    gsub("[^[:alpha:]]", " ", x))

  # remove all non-alpha characters
  output <- tibble(title = meta$title, author = meta$author, text = unlist(texts, recursive = FALSE))
}

# run function
set.seed(1984L)
texts_dt <- lapply(book_list, prepare_dt, num_lines = 500, removePunct = TRUE)
texts_dt <- do.call(rbind, texts_dt)
```

b) Print the str() of your table

```
str(texts_dt)
```

```
## tibble [16 x 3] (S3: tbl_df/tbl/data.frame)
##  $ title : chr [1:16] "The Fall of the House of Usher" "The Masque of the Red Death" "The Raven" "Eu
##  $ author: chr [1:16] "poe" "poe" "poe" "poe" ...
##  $ text  : chr [1:16] "
```

c) Now use the stylest select vocab function to select the terms you will include in your model. Justify any pre-processing choices you make. What percentile (of term frequency) has the best prediction rate? Also report the mean rate of incorrectly predicted speakers of held-out texts.

```r
filter <- corpus::text_filter(drop_punct = TRUE, drop=stopwords("english"), drop_number = TRUE)
vocab_custom <- stylest_select_vocab(texts_dt$text, texts_dt$author, filter = filter, smooth=1,
                                     nfold = 5, cutoff_pcts = c(25, 50, 75, 80, 90, 99))

print(vocab_custom)
```

```
## $cutoff_pct_best
## [1] 90
##
## $cutoff_pcts
## [1] 25 50 75 80 90 99
##
## $miss_pct
##          [,1]     [,2]     [,3]     [,4] [,5]     [,6]
## [1,] 33.33333 33.33333 33.33333 33.33333    0 33.33333
## [2,]  0.00000  0.00000  0.00000  0.00000    0  0.00000
## [3,] 25.00000 25.00000 25.00000 25.00000   25 25.00000
## [4,] 50.00000 50.00000 50.00000 50.00000   50  0.00000
## [5,] 25.00000 25.00000 25.00000 25.00000   25 50.00000
##
## $nfold
## [1] 5
##
## attr(,"class")
## [1] "stylest_select_vocab"
```

The best prediction rate is using the 99th percentile cutoff (it may vary when ran again) and its mean rate of incorrectly predicted speakers is 44%. Additionally, I removed punctuation and stopwords as preprocessing because they do not provide value to which speaker is the author.

d) Use your optimal percentile from above to subset the terms to be included in your model. Now go ahead and fit the model using stylest fit. The output of this function includes information on the rate at which each author uses each term (the value is labeled rate). Report the top 5 terms (in terms of usage rate) for each author. Do these terms make sense?

```r
vocab_subset <- stylest_terms(texts_dt$text, texts_dt$author, vocab_custom$cutoff_pct_best ,
                              filter = filter)

style_model <- stylest_fit(texts_dt$text, texts_dt$author, terms = vocab_subset, filter = filter)
summary(style_model)
```

```
##                     Length Class             Mode
## speakers               4   -none-            character
## filter                16   corpus_text_filter list
## terms               1293   -none-            character
## ntoken                 4   -none-            numeric
## smooth                 1   -none-            numeric
## weights                0   -none-            NULL
## rate                5172   -none-            numeric
## terms_without_weights  0   -none-            NULL
## fill_weight            0   -none-            NULL
```

```r
head(stylest_term_influence(style_model, texts_dt$text, texts_dt$author))  # influential terms
```

```
##    term infl_avg  infl_max
## 1   one 2.236439  4.233441
## 2   now 2.042552  3.434383
## 3   see 6.255129 17.892286
## 4    us 2.285810  6.018727
## 5  long 1.050807  1.419865
## 6 still 2.148114  6.407744
```

```r
authors <- unique(texts_dt$author)
term_usage <- style_model$rate
print(lapply(authors, function(x) head(term_usage[x,][order(-term_usage[x,])])) %>% setNames(authors))
```

```
## $poe
##       upon       door        one    chamber        now    nothing
## 0.01770935 0.01130195 0.01112397 0.01094598 0.01041203 0.00827623
##
## $twain
##         tom         got        said         see         now        well
## 0.013061031 0.011952822 0.011794506 0.011636191 0.009103143 0.009103143
##
## $shelley
##         one         now         may        love        life        must
## 0.009652345 0.007602289 0.007089775 0.006577261 0.006235586 0.005893910
##
## $doyle
##        said        upon         one          us         man         now
## 0.015937263 0.015431318 0.013070242 0.008685387 0.008348090 0.006324311
```

The top 5 terms per author are in the output. I glanced over the texts and they appear to make sense (but it is difficult to tell).

Additionally, terms like 't' and 's' sometimes show up. Which most probably come from words like don't, won't and someone's car, etc. Once we drop the punctuation, the t and s remain alone.

e) Choose any two authors, take the ratio of their rate vectors (make sure dimensions are in the same order) and arrange the resulting vector from largest to smallest values. What are the top 5 terms according to this ratio? How would you interpret this ordering?

```r
doyle_poe <- term_usage['doyle',] / term_usage['poe',]
#poe_doyle <- term_usage['poe',] / term_usage['doyle',]

print(doyle_poe[1:5])
```

```
##       one       now       see        us      long
## 1.1749625 0.6074041 2.3852130 1.9917896 1.0350166
```

Here we can see that the word "now" is used much more by poe, whereas the word "see" is used a lot more by doyle.

f) Load the mystery excerpt provided. According to your fitted model, who is the most likely author?

```
mystery_excerpt <- readRDS("C:/Users/alexx/OneDrive/Escritorio/mystery_excerpt.rds")
pred <- stylest_predict(style_model, mystery_excerpt)
print(pred$predicted)
```

```
## [1] twain
## Levels: doyle poe shelley twain
```

```
print(pred$log_probs)
```

```
## 1 x 4 Matrix of class "dgeMatrix"
##          doyle       poe   shelley         twain
## [1,] -18.64886 -54.78678 -35.64735 -7.959843e-09
```

According to the model, twain is the most likely author.

g) Use textstat collocation to inspect 2-grams with min count = 5 from your DFM of all 16 labeled novels. Report the 10 collocations with the largest value. Report the 10 collocations with the largest count. Discuss which set of n-grams is likely to be multi-word expressions.

```
corpus <- corpus(texts_dt$text)

matrix_2grams <- textstat_collocations(corpus, method = "lambda", size = 2, min_count = 5)
```

Top 10 collocations with largest lambda: edgar allan, denser perfumed, whispering vows, syllable expressing, candelabrum amid, unseen censer, allan poe, arabesque figures, densely crowded, unsuited limbs.

Top 10 collocations with largest count: of the, in the, and the, to the, it was, on the, of a, from the, to be, that the.

We can see that the the most commons n-grams are combinations of prepositions or even some phrasal verbs (which should be removed during preprocessing).

QUESTION 5

a) Using the aforementioned corpus make snippets between 150 to 350 characters in length and clean the snippets (print the top 10).

```
un_data = corpus_subset(data_corpus_ungd2017)
snippetData <- snippets_make(un_data, nsentence = 1, minchar = 150, maxchar = 350)
snippetData <- snippets_clean(snippetData)
head(snippetData,10)
```

```
##          docID snippetID
## 1  Afghanistan    100001
## 2  Afghanistan    100002
## 3  Afghanistan    100003
## 4  Afghanistan    100009
## 5  Afghanistan    100011
## 6  Afghanistan    100012
## 7  Afghanistan    100015
## 8  Afghanistan    100016
## 9  Afghanistan    100017
## 10 Afghanistan    100020
##
## 1                                                      As I stand here before the General Assembly to
## 2                              Shaped by the Great Depression and tempered by the carnage of the Seco
## 3                   The United Nations, the International Monetary Fund, the World Bank and other orga
## 4                                                              There is an emerging consensus that ad
## 5                                                               Sixteen years after the tragedy of 11
## 6      Driven by transnational terrorist networks, criminal organizations, cybercrime and State spons
## 7  Terrorism is not only an attack on human life and basic freedoms, but an attack on the compact of
## 8                                                                        We must confront the threat o
## 9                           Lastly, despite the incorporation of tenets of the Universal Declaration
## 10                                                                         I welcome the chance for Af
```

b) Randomly sample 1000 snippets and use these to generate pairs for a minimum spanning tree. From these generate 10 gold pairs. Without looking at the automated classification, read each pair and select. whichever you think is "easiest" to read. Now compare your classification with those made by the package. What proportion of the ten gold pairs were you in agreement with the automated classification? Any reasons why you may have arrived at a different judgment?

```
testData <- sample_n(snippetData, 10)
snippetPairsMST <- pairs_regular_make(testData)
pairs_regular_browse(snippetPairsMST)

snippetPairsAll <- pairs_regular_make(snippetData[sample(1:nrow(snippetData), 1000), ])
gold_questions <- pairs_gold_make(snippetPairsAll, n.pairs = 10)

print(gold_questions)
```

```
##        docID1 snippetID1
## 1       Sudan   15200003
## 2    Paraguay   14500002
## 3    Kiribati    9300084
```

```
## 4  Sierra Leone   15600062
## 5      Colombia    3800068
## 6        Zambia   19500039
## 7      Bulgaria    1700036
## 8      Cameroon    3500057
## 9     Sri Lanka   10300031
## 10      Liberia    9900067
##
## 1
## 2  On behalf of the people and the Government of the Republic of Paraguay, I wish to express to the p
## 3
## 4
## 5                                                                                                   It
## 6
## 7                                                                                     Bulgaria categori
## 8
## 9
## 10
##          docID2 snippetID2
## 1    Montenegro   11900047
## 2        Norway   13100055
## 3       Bahrain    1800012
## 4      Botswana    2800023
## 5        Belize    2200040
## 6  Vatican City   18600076
## 7         India    7900140
## 8      Bulgaria    1700013
## 9         Egypt    5300023
## 10   Seychelles   16700048
##
## 1                                                                                                     Monter
## 2
## 3  Accordingly, this year has witnessed numerous initiatives for fruitful cooperation, notably the la
## 4
## 5
## 6
## 7
## 8
## 9
## 10
##          read1     read2  readdiff _golden easier_gold
## 1   34.455172 -29.21375  63.66892    TRUE           1
## 2    4.758333  63.58500 -58.82667    TRUE           2
## 3   44.811250 -18.13500  62.94625    TRUE           1
## 4  -19.304091  46.66500 -65.96909    TRUE           2
## 5   34.686053 -24.54731  59.23336    TRUE           1
## 6   41.563333 -14.81200  56.37533    TRUE           1
## 7    5.795000  65.33088 -59.53588    TRUE           2
## 8   35.950000 -46.43500  82.38500    TRUE           1
## 9   50.285455 -12.39286  62.67831    TRUE           1
## 10  -5.436667  57.79310 -63.22977    TRUE           2
##
## 1  Text A is "easier" to read because it contains some combination of shorter sentences, more commonl
## 2  Text B is "easier" to read because it contains some combination of shorter sentences, more commonl
```

```
## 3  Text A is "easier" to read because it contains some combination of shorter sentences, more commonl
## 4  Text B is "easier" to read because it contains some combination of shorter sentences, more commonl
## 5  Text A is "easier" to read because it contains some combination of shorter sentences, more commonl
## 6  Text A is "easier" to read because it contains some combination of shorter sentences, more commonl
## 7  Text B is "easier" to read because it contains some combination of shorter sentences, more commonl
## 8  Text A is "easier" to read because it contains some combination of shorter sentences, more commonl
## 9  Text A is "easier" to read because it contains some combination of shorter sentences, more commonl
## 10 Text B is "easier" to read because it contains some combination of shorter sentences, more commonl
```

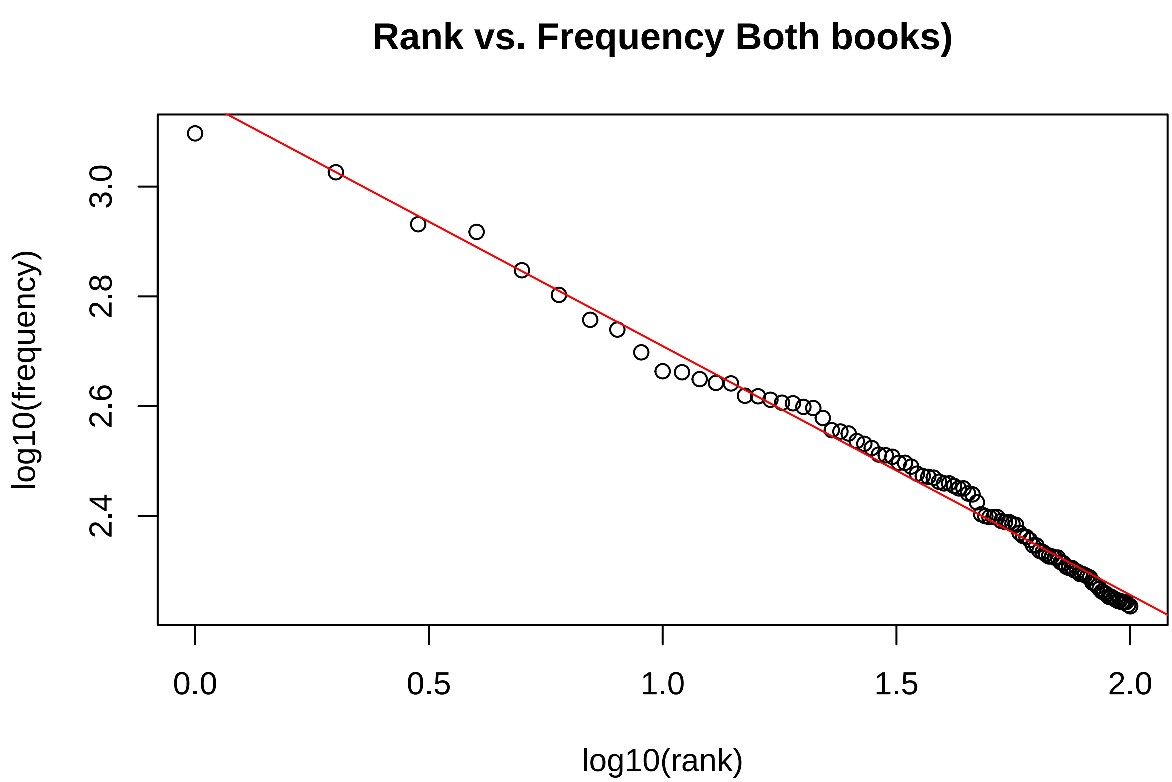
The golden question show up in the browser as an html when the print statement is executed.

Regarding the output, I was in agreement with 9 out of the 10 gold questions. The one I differed with the model is the second one. The reason why we differed, I believe, is because the model classifies a sentence as not easily readable if it finds many words that are not very common in that language. In this case, even though the sentence had a good amount of uncommon words, it was very clearly expressed, even more so that the other sentence which used simpler words.

QUESTION 6

Using Louisa May Alcott's "Little Women" (gutenberg id = 514) and F. Scott Fitzgerald's "The Great Gatsby" (gutenberg id = 64317), make a graph demonstrating Zipf's law. Include this graph and also discuss any pre-processing decisions you made.

```
little_women <- gutenberg_download(514)
great_gatsby <- gutenberg_download(64317)

women_dfm <- dfm(corpus(little_women), remove_punct=TRUE, remove=stopwords("english"))
gatsby_dfm <- dfm(corpus(great_gatsby), remove_punct=TRUE, remove=stopwords("english"))

full_dfm <- rbind(women_dfm, gatsby_dfm)

plot(log10(1:100), log10(topfeatures(full_dfm, 100)), xlab = "log10(rank)",
     ylab ="log10(frequency)", main = "Rank vs. Frequency Both books)")

regression <- lm(log10(topfeatures(full_dfm, 100)) ~ log10(1:100))
abline(regression, col = "red")
```




```
confint(regression)
```

```
##                    2.5 %     97.5 %
## (Intercept)    3.1474105  3.1781007
## log10(1:100) -0.4627007 -0.4438701
```

```r
summary(regression)
```

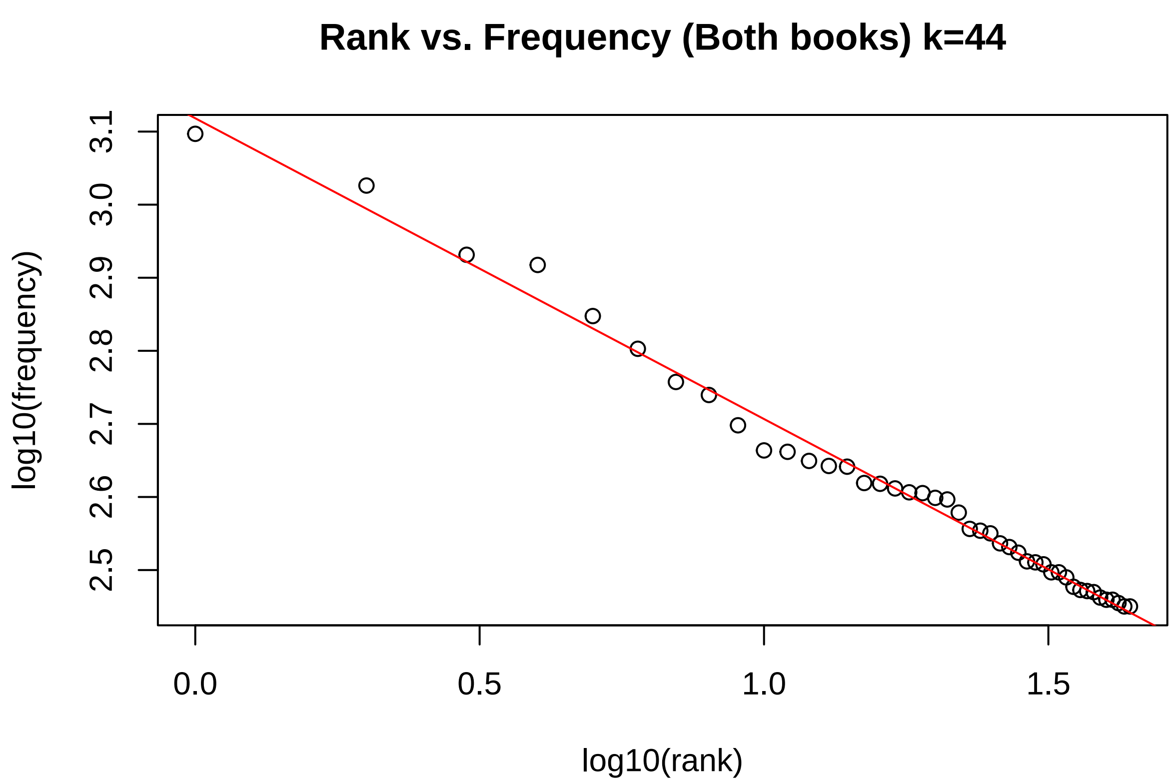
```
##
## Call:
## lm(formula = log10(topfeatures(full_dfm, 100)) ~ log10(1:100))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.065846 -0.014063 -0.001273  0.016517  0.033184
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.162756   0.007733  409.01   <2e-16 ***
## log10(1:100) -0.453285   0.004744  -95.54   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01903 on 98 degrees of freedom
## Multiple R-squared:  0.9894, Adjusted R-squared:  0.9893
## F-statistic:  9128 on 1 and 98 DF,  p-value: < 2.2e-16
```

In the graph we can see that the word frequency is inversely proportional to its rank. Additional?y, I removed punctuation and stopwords. Stopwords could have been left unremoved because even though they change the slope of the regression curve, they still show the inversely proportional relation between rank and frequency.

QUESTION 7

Find the value of b that best fit the two works from the previous question to Heap's law, fixing k = 44. Report the value of b as well as any pre-processing decisions you made.

```
k=44
plot(log10(1:k), log10(topfeatures(full_dfm, k)), xlab = "log10(rank)", ylab = "log10(frequency)",
     main = "Rank vs. Frequency (Both books) k=44")

regression <- lm(log10(topfeatures(full_dfm, k)) ~ log10(1:k))
abline(regression, col = "red")
```




```
# Returns the 95% confidence intervals for the regression coefficients
confint(regression)
```

```
##                  2.5 %     97.5 %
## (Intercept)  3.1019066  3.1345463
## log10(1:k)  -0.4241482 -0.3989123
```

```
# Provides R-squared, F-test, and cofficient estimates from regression
summary(regression)
```

```
##
## Call:
```

```
## lm(formula = log10(topfeatures(full_dfm, k)) ~ log10(1:k))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.042995 -0.005025  0.000853  0.004804  0.047045
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.118226   0.008087  385.59   <2e-16 ***
## log10(1:k)  -0.411530   0.006252  -65.82   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01568 on 42 degrees of freedom
## Multiple R-squared:  0.9904, Adjusted R-squared:  0.9902
## F-statistic:  4332 on 1 and 42 DF,  p-value: < 2.2e-16
```

```
M = nfeat(full_dfm)
T = sum(ntoken(full_dfm))

b = log(M/k) / log(T)
print(b)
```

```
## [1] 0.4922091
```

As seen in the output, the value of b is 0.4922. I decided to remove punctuation and stopwords because as seen in the lab, they have a large effect on the graph (even though it still shows the negative proportion)

QUESTION 8

Both "Little Women" and "The Great Gatsby" broach the topic of class, but in very different ways. Choose a few Key Words in Context and discuss the different context in which those words are used by each author. Give a brief discussion of how the two works treat this theme differently.

```
quanteda::kwic(corpus(little_women), pattern = "class", valuetype="glob", window=8)
```

```
## Keyword-in-context with 7 matches.
##    [text3891, 8] about fine clothes which attracts a certain | class |
##  [text10337, 13]          vanquished enemies. The' men of my | class |
##   [text11072, 4]                                " Our drawing | class |
##   [text11105, 7]                  " Twelve or fourteen in the | class |
##  [text11456, 12]      murder, for the story belonged to that | class |
##   [text19478, 9]           " Yes, indeed, and there's another | class |
##  [text19626, 12]    join in Grandma's laugh, and dismiss the | class |
##
##  of people and secures
##  ',
##  breaks up next week, and before the
##  , but I dare say they won't all
##  of light
##  who can't ask, and who suffer
##  in
```

```
quanteda::kwic(corpus(great_gatsby), pattern = "class", valuetype="glob", window=8)
```

```
## Keyword-in-context with 1 match.
##  [text4476, 2] your | class | at Yale."
```

```
#quanteda::kwic(corpus(little_women), pattern = "rich", valuetype="glob", window=8)
#quanteda::kwic(corpus(great_gatsby), pattern = "rich", valuetype="glob", window=8)

quanteda::kwic(corpus(little_women), pattern = "wealthy", valuetype="glob", window=8)
```

```
## Keyword-in-context with 0 matches.
```

```
quanteda::kwic(corpus(great_gatsby), pattern = "wealthy", valuetype="glob", window=8)
```

```
## Keyword-in-context with 3 matches.
##    [text183, 7] anticlimax. His family were enormously | wealthy |
##    [text188, 1]                                        | wealthy |
##  [text2163, 13]                ." I am the son of some | wealthy |
##
##  - even in college his
##  enough to do that.
##  people in the
```

```
#quanteda::kwic(corpus(little_women), pattern = "money", valuetype="glob", window=8)
#quanteda::kwic(corpus(great_gatsby), pattern = "money", valuetype="glob", window=8)
```

Little women narrates the story of a wealthy family that goes into poverty. We can see that when class or richness is mentioned, it is referred as something negative and with negative adjectives. On the other hand, in The Great Gatsby class is treated as a social position and comparable to honor, so seen positively.

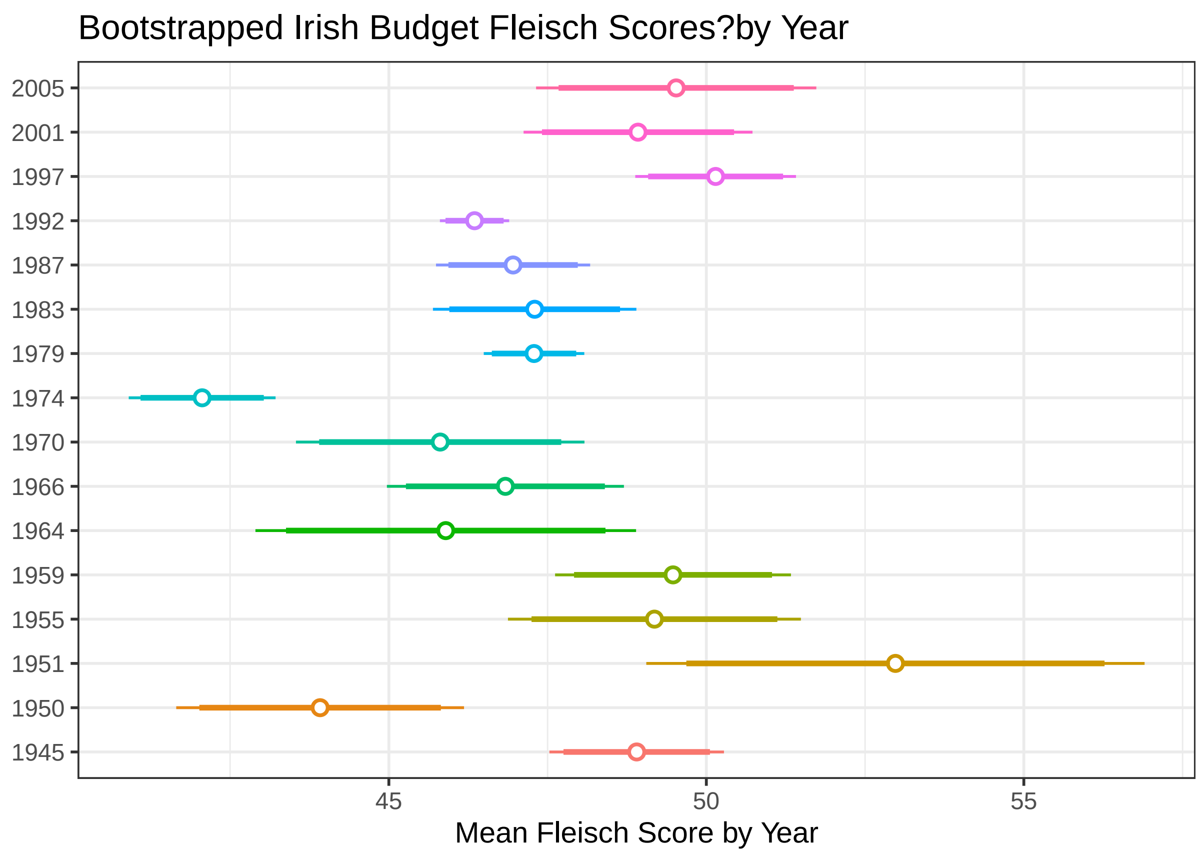
(I commented out two keywords because the output was very messy in the markdown output)

QUESTION 9

a) Obtain the UK Conservative Party's manifestos from quanteda. Generate estimates of the FRE scores of these manifestos over time (i.e. per year), using sentence-level bootstraps instead of the speech-level bootstraps used in Recitation 4. Report the bootstrapped estimates and standard errors in a table

```
data("data_corpus_ukmanifestos")
manifestos <- corpus_subset(data_corpus_ukmanifestos, Party == "Con")

# tokenize by sentences
sent_tokens <- unlist(tokens(manifestos, what = "sentence", include_docvars = TRUE))
# extract year metadata
yearnames <- list(unlist(names(sent_tokens)))
yearnames <- lapply(yearnames[[1]], function(x){strsplit(x, "_")[[1]][3]})
yearslist <- unlist(yearnames)
# create tibble
sentences_df <- tibble(text = sent_tokens, year = yearslist)
# create quanteda corpus object
sent_corp <- corpus(sentences_df$text)
docvars(sent_corp, field = "Year") <- sentences_df$year

#Bootstrap
boot_flesch <- function(sentences_df){
    N <- nrow(sentences_df)
    bootstrap_sample <- corpus_sample(corpus(c(sentences_df$text)), size = N, replace = TRUE)
    bootstrap_sample<- as.data.frame(as.matrix(bootstrap_sample))
    readability_results <- textstat_readability(bootstrap_sample$V1, measure = "Flesch")
    return(mean(readability_results$Flesch))}

boot_flesch_by_year <- pblapply(unique(yearslist), function(x){
    sub_data <- sentences_df %>% filter(year == x)
    output_flesch <- lapply(1:10, function(i) boot_flesch(sub_data))
    return(unlist(output_flesch))
})

# compute mean and std.errors
year_means <- lapply(boot_flesch_by_year, mean) %>% unname() %>% unlist()
year_ses <- lapply(boot_flesch_by_year, sd) %>% unname() %>% unlist()

# Plot results--party
plot_dt <- tibble(year = unique(yearslist), mean = year_means, ses = year_ses)

interval1 <- -qnorm((1-0.9)/2)   # 90% multiplier
interval2 <- -qnorm((1-0.95)/2)  # 95% multiplier

# ggplot point estimate + variance
ggplot(plot_dt, aes(colour = year)) +
  geom_linerange(aes(x = year, ymin = mean - ses*interval1, ymax = mean + ses*interval1),
                 lwd = 1, position = position_dodge(width = 1/2)
  ) +
  geom_pointrange(aes(x = year, y = mean, ymin = mean - ses*interval2, ymax = mean + ses*interval2),
                  lwd = 1/2, position = position_dodge(width = 1/2),
                  shape = 21, fill = "WHITE"
  ) +
```

```
  coord_flip() + theme_bw() +
  xlab("") + ylab("Mean Fleisch Score by Year") +
  ggtitle("Bootstrapped Irish Budget Fleisch Scores?by Year") +
  theme(legend.position = "none")
```

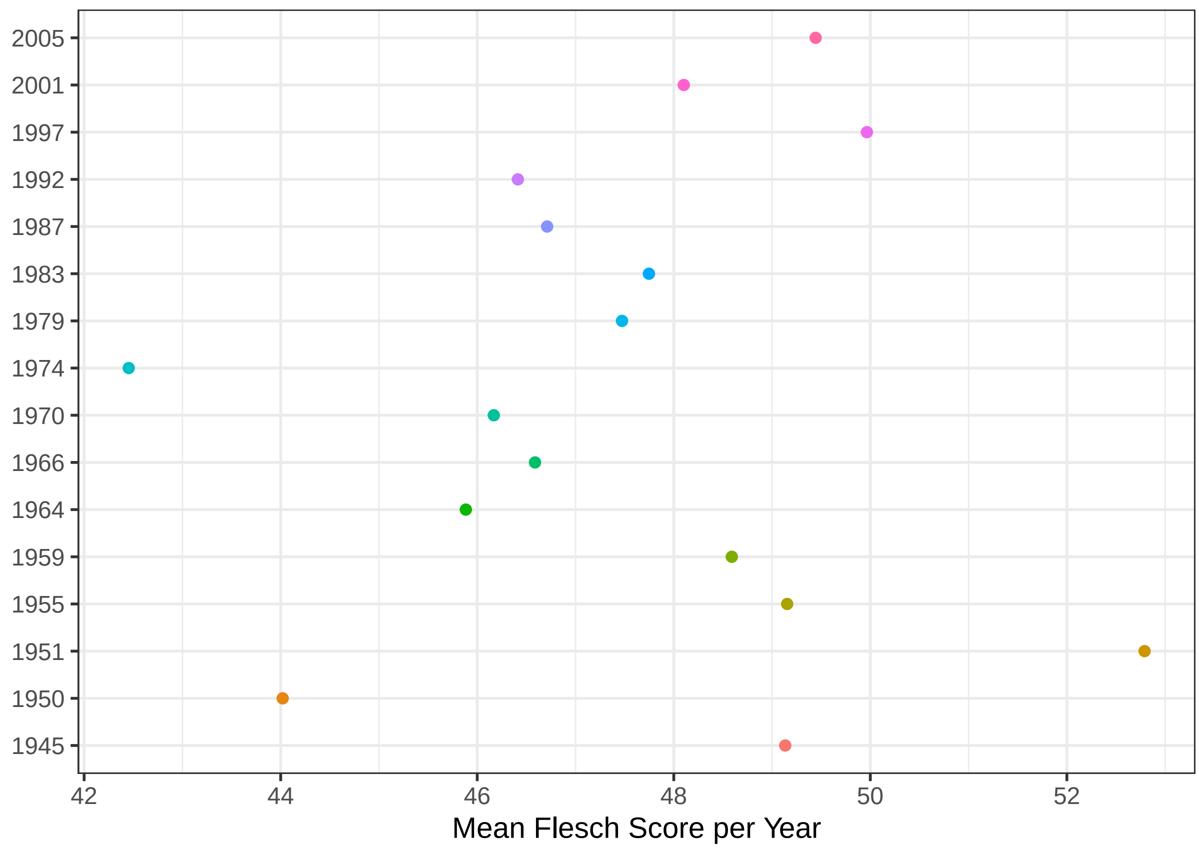



```
df <- data.frame(unique(yearslist), year_means, year_ses)
print(df)
```

```
##    unique.yearslist. year_means  year_ses
## 1               1945   48.90335 0.7013046
## 2               1950   43.91788 1.1562240
## 3               1951   52.97808 2.0018762
## 4               1955   49.18269 1.1769275
## 5               1959   49.47550 0.9476188
## 6               1964   45.89631 1.5291572
## 7               1966   46.83567 0.9521954
## 8               1970   45.80899 1.1588847
## 9               1974   42.06018 0.5898620
## 10              1979   47.28633 0.4039334
## 11              1983   47.29650 0.8171336
## 12              1987   46.95660 0.6193326
## 13              1992   46.34981 0.2782039
## 14              1997   50.14604 0.6457917
## 15              2001   48.92423 0.9197082
## 16              2005   49.52528 1.1258856
```

b) Compute the (non-bootstrapped) mean FRE score over time and report the results in a table. Discuss the contrast with the bootstrapped estimates from the previous section.

```
flesch_point <- sentences_df$text %>% textstat_readability(measure = "Flesch") %>%
  group_by(sentences_df$year) %>%
  summarise(mean_flesch = mean(Flesch)) %>%
  setNames(c("year", "mean")) %>% arrange(year)

ggplot(flesch_point, aes(x = year, y = mean, colour = year)) +
  geom_point() +
  coord_flip() + theme_bw() +
  scale_y_continuous(breaks=seq(floor(min(flesch_point$mean)),
                                ceiling(max(flesch_point$mean)), by = 2)
  ) +
  xlab("") + ylab("Mean Flesch Score per Year") + theme(legend.position = "none")
```




```
print(flesch_point)
```

```
## # A tibble: 16 x 2
##    year   mean
##    <chr> <dbl>
##  1 1945   49.1
##  2 1950   44.0
##  3 1951   52.8
##  4 1955   49.2
```

```
## 5 1959   48.6
## 6 1964   45.9
## 7 1966   46.6
## 8 1970   46.2
## 9 1974   42.5
## 10 1979   47.5
## 11 1983   47.7
## 12 1987   46.7
## 13 1992   46.4
## 14 1997   50.0
## 15 2001   48.1
## 16 2005   49.4
```

As we can see in the tables, there is not much of a difference between the bootstrapped results and the non-boostrapped ones.