Text Practice

Sara Dovalo and Javier Muñoz

18/3/2022

Preprocessing

First of all, we load the dataset with `readtext`function, which treats the text file as `data.frame`object.

```
#Load the data
data_text <- texts(readtext("https://www.gutenberg.org/cache/epub/67584/pg67584.txt"))
names(data_text) <- "Archibald Gracie"
```

In order to separate the text from the metadata which can include the file, we find the start and the end of the book looking directly the text. It is recommendable for the next steps of the practice.

```
start_text <- stri_locate_first_fixed(data_text, "CHAPTER I\n\nTHE LAST DAY ABOARD SHIP")[1]
end_text <- stri_locate_last_fixed(data_text, "deeds.")[1]
# Check the end of the text
kwic(tokens(data_text), "deeds")
```

```
## Keyword-in-context with 2 matches.
##  [Archibald Gracie, 87851]    destiny. Good and bad | deeds |
##  [Archibald Gracie, 88570] one more legend of brave | deeds |
##
##  were done that night and
##  . Transcriberâ \200 \231 s
```

```
novel <- stri_sub(data_text, start_text, end_text)
```

With the objective of an easier analysis, it is recomendable to convert the text to lower case and split the document into words. It can do easily using `char_tolower()`(as we handle with `character` objects) and then `tokens()` functions to the word partition.

```
novel_lower <- char_tolower(novel)
titanic_word <- tokens(novel_lower, remove_punct = TRUE) %>% as.character()
titanic_word[1:10]
```

```
##  [1] "chapter" "i"       "the"     "last"    "day"     "aboard"  "ship"
##  [8] "â"       "\200"       "œthere"
```

Tasks

1. Analyse and study the occurrence of words related with love or positive feelings in general.

Fot carrying out this task we have used `which()` function to match the words we have indicated previously in a vector. We keep into a list the number of times that some words related with positive feelings appear in the novel.

```
Words <- c("love","feeling", "smile", "desire", "wonderful","happy")
list_pwords <- list()
for (i in 1:length(Words)){
  list_pwords = c(list_pwords, length(titanic_word[which(titanic_word == Words[i])]))
}
Frequency <- list_pwords %>% unlist()
matrix_count <- data.frame(Words, Frequency)
knitr::kable(matrix_count,"latex")
```

Words	Frequency
love	2
feeling	5
smile	1
desire	2
wonderful	1
happy	6

2. Make frequency plots.

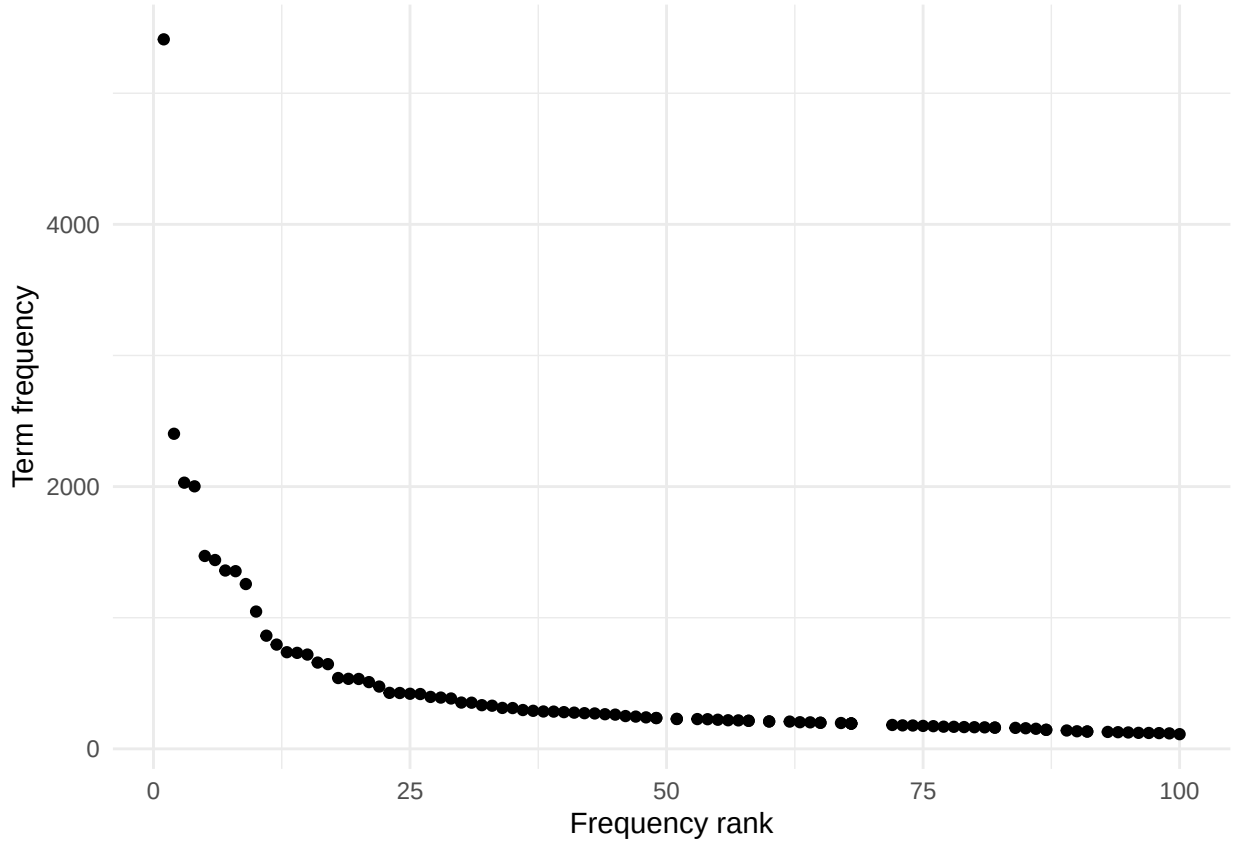
It can be interesting to make a plot of the 100 words most frequent in the text. The `dfm()` command allows to us to create a document-frequency matrix. Once this matrix has been computed, we only have to use the matrix and to indicate the first 100 most frequent words in the parameter `n`of the function `textstat_frequency()`.

```
titanic_dfm <- dfm(novel_lower, remove_punct = TRUE)
```

```
## Warning: 'dfm.character()' is deprecated. Use 'tokens()' first.
```

```
## Warning: '...' should not be used for tokens() arguments; use 'tokens()' first.
```

```
theme_set(theme_minimal())
textstat_frequency(titanic_dfm, n = 100) %>%
  ggplot(aes(x = rank, y = frequency)) +
  geom_point() +
  labs(x = "Frequency rank", y = "Term frequency")
```




```
textstat_frequency(titanic_dfm, n = 10)
```

```
##    feature frequency rank docfreq group
## 1      the      5411    1       1   all
## 2      and      2403    2       1   all
## 3       to      2030    3       1   all
## 4       of      2002    4       1   all
## 5      was      1471    5       1   all
## 6        \200      1440    6       1   all
## 7        i      1360    7       1   all
## 8       in      1355    8       1   all
## 9        a      1257    9       1   all
## 10    boat      1047   10       1   all
```

As expected, the most repeated words are the most common words in any text document, i.e *the*, *and* or *to*.

3. Compare word frequency data of words like "he", "she", "him", "her" and show also relative frequencies.

First, we show frequencies of each word in the text to identify the most repeated ones.

```
sorted_titanic_freqs_t <- topfeatures(titanic_dfm, n = nfeat(titanic_dfm))
sorted_titanic_freqs_t[c("he", "she", "him", "her")]
```

```
##  he she him her
## 646 228 162 227
```

It seems that *he* is much more repeated than the other items. However, the number of the rest of the words in the text is similar. We compute the ratio between *he* and *she* to check the first fact and the ratio between *she* and *her* to check the second one.

```
sorted_titanic_freqs_t["he"] / sorted_titanic_freqs_t["she"]
```

```
##       he
## 2.833333
```

```
sorted_titanic_freqs_t["she"] / sorted_titanic_freqs_t["her"]
```

```
##      she
## 1.004405
```

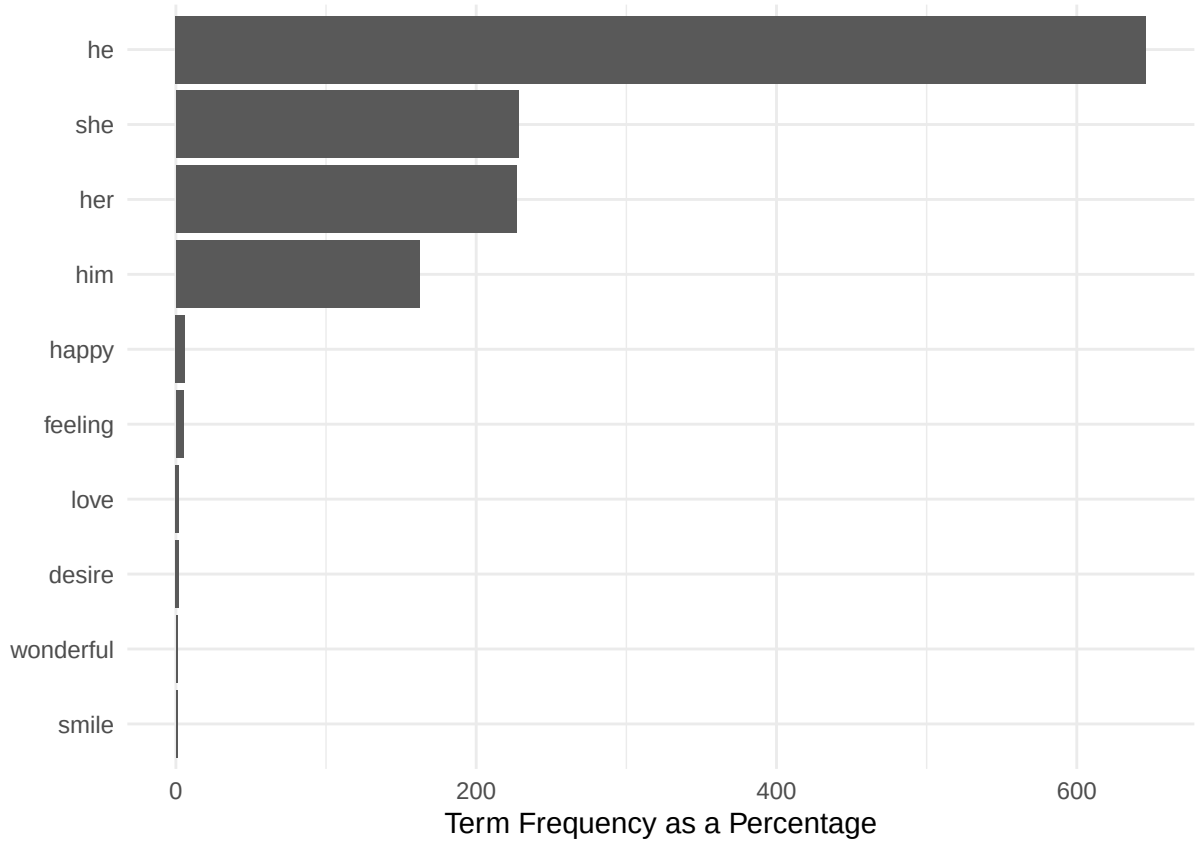
While the ratio in the first case is almost three, i.e the word *he* is almost three times more in the text than word *she*, the second ratio shows that both words appear in a similar way in terms of number of times.

The relative frequencies can be extract weighting directly the matrix created previously using `dfm_weight()` function. In addition, we plot these frequencies and the positive words of task 1.

```
titanic_dfm_pct <- dfm_weight(titanic_dfm, scheme = "prop") * 100
dfm_select(titanic_dfm_pct, pattern = c("he", "she", "him", "her", Words))
```

```
## Document-feature matrix of: 1 document, 10 features (0.00% sparse) and 0 docvars.
##        features
## docs         her       she        he       him       happy       smile
##   text1 0.296864 0.2981717 0.8448199 0.2118589 0.007846625 0.001307771
##        features
## docs        feeling      desire        love   wonderful
##   text1 0.006538854 0.002615542 0.002615542 0.001307771
```

```
textstat_frequency(titanic_dfm[, c("he", "she", "him", "her", Words)]) %>%
  ggplot(aes(x = reorder(feature, -rank), y = frequency)) +
  geom_bar(stat = "identity") + coord_flip() +
  labs(x = "", y = "Term Frequency as a Percentage")
```



4. Make a token distribution analysis

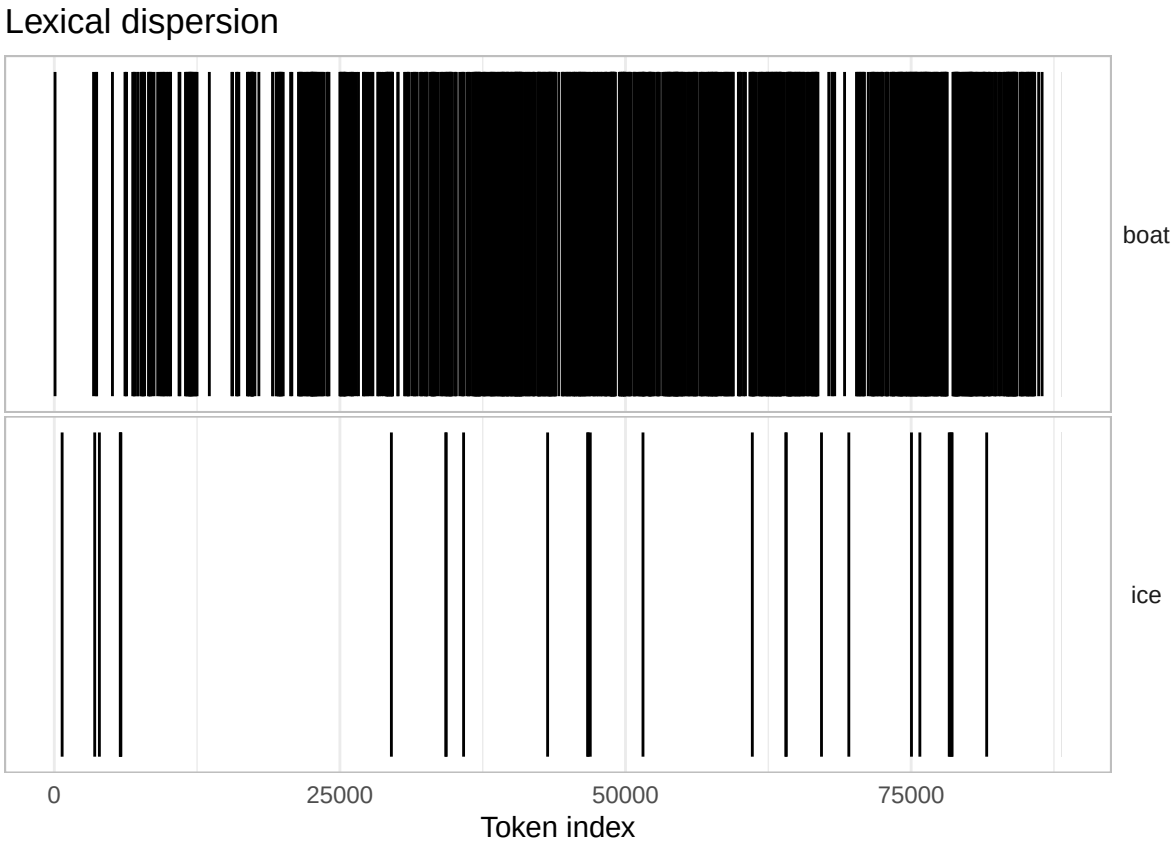
Lexical dispersion plots allows to measure the frequency at which a word appears along the different parts of a text. We will use them to compare the frequency of words like *boat* and *ice*.

```
textplot_xray(
    kwic(novel, pattern = "boat"),
    kwic(novel, pattern = "ice")) +
    ggtitle("Lexical dispersion")
```

```
## Warning: 'kwic.character()' is deprecated. Use 'tokens()' first.

## Warning: 'kwic.character()' is deprecated. Use 'tokens()' first.

## Warning: Use of `x$ntokens` is discouraged. Use `ntokens` instead.
```



This plot suggests that the word *boat* appears during all the text piece, since obviously almost the entire work occurs in a boat. However, *ice* appears with more frequency in the last part.

5. Identify chapter breaks.

The chapters will be broken using the parameter `pattern` of the function `corpus_segment`. It allows to specify certain regular expression to search the beginning of each chapter. In particular, in our case all chapters start with: *CHAPTER (CHARACTERS IN ROMAN LETTERS) (character)*. Thus, we are able to identify the seven chapters which contain the text and split it in seven different documents.

```
# Identify the location of the chapter breaks
chapters_corp <-
    corpus(data_text) %>%
    corpus_segment(pattern = "CHAPTER\\s[A-Z]*\\n", valuetype = "regex")
summary(chapters_corp, 7)
```

```
## Corpus consisting of 7 documents, showing 7 documents:
##
##                Text Types Tokens Sentences       pattern
##  Archibald Gracie.1  1064   3132        82   CHAPTER I\n
##  Archibald Gracie.2  1990  10126       325  CHAPTER II\n
##  Archibald Gracie.3   977   3534       102 CHAPTER III\n
##  Archibald Gracie.4  1396   6223       198  CHAPTER IV\n
##  Archibald Gracie.5  1606   7079       237   CHAPTER V\n
##  Archibald Gracie.6  3178  30062      1576  CHAPTER VI\n
##  Archibald Gracie.7  3836  32422      1743 CHAPTER VII\n
```

In order to tidy it up, the final character of the chapters title can be removed using `stri_trim_right()`. Finally, the titles of the chapters are renamed.

```
# Tidy up \n
docvars(chapters_corp, "pattern") <- stringi::stri_trim_right(docvars(chapters_corp, "pattern"))
summary(chapters_corp, n = 7)
```

```
## Corpus consisting of 7 documents, showing 7 documents:
##
##                Text Types Tokens Sentences     pattern
##  Archibald Gracie.1  1064   3132        82   CHAPTER I
##  Archibald Gracie.2  1990  10126       325  CHAPTER II
##  Archibald Gracie.3   977   3534       102 CHAPTER III
##  Archibald Gracie.4  1396   6223       198  CHAPTER IV
##  Archibald Gracie.5  1606   7079       237   CHAPTER V
##  Archibald Gracie.6  3178  30062      1576  CHAPTER VI
##  Archibald Gracie.7  3836  32422      1743 CHAPTER VII
```

```
# Rename chapters
docnames(chapters_corp) <- docvars(chapters_corp, "pattern")
```

6. Only if you have some knowledge about the novel: Make a correlation analysis between words related with love or positive feelings and some particular characters or people of the novel.

We have not read *The Truth about the Titanic*, but it is known that it is based on Gracie's detailed account of his experience the night in which passengers of *Titanic* ship suffered lot, becoming a popular tragedy known around the entire word. Hence, Gracie is a character which appears in the document, as he tells his own testimony. We analyse the *Gracie* word with happy to do the correlation analysis.

```
chap_dfm <- dfm(chapters_corp)
```

```
## Warning: 'dfm.corpus()' is deprecated. Use 'tokens()' first.
```

```
dfm_weight(chap_dfm, scheme = "prop") %>%
    textstat_simil(selection = c("gracie", "happy"), method = "correlation", margin = "features") %>%
    as.matrix() %>%
    head(2)
```

```
## Warning: 'selection' is deprecated. Use 'y' instead.
```

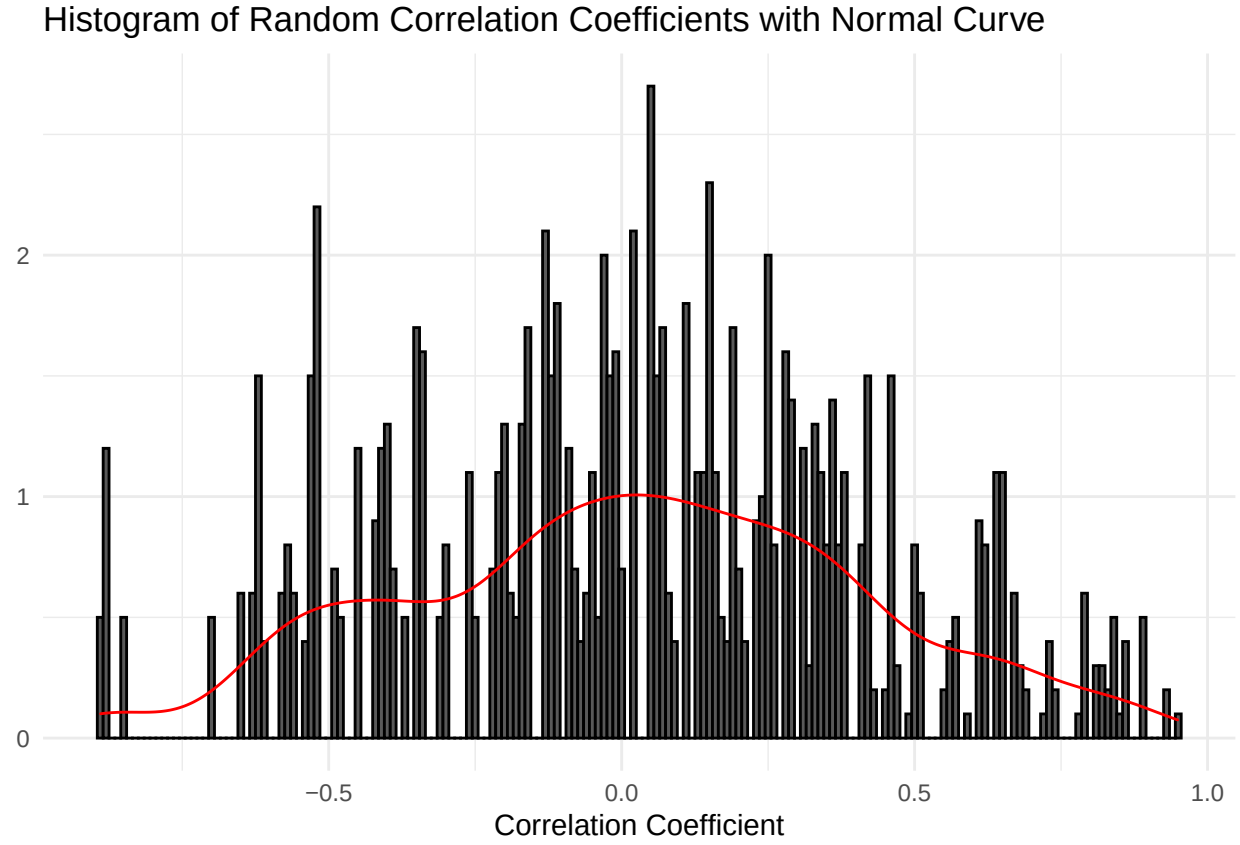
```
##          gracie      happy
## the  -0.1187102 -0.0714125
## last -0.4717194  0.3895891
```

```
cor_data_df <- dfm_weight(chap_dfm, scheme = "prop") %>%
    dfm_keep(pattern = c("gracie", "happy")) %>%
    convert(to = "data.frame")

# sample 1000 replicates and create data frame
```

```r
n <- 1000
samples <- data.frame(
    cor_sample = replicate(n, cor(sample(cor_data_df$gracie), cor_data_df$happy)),
    id_sample = 1:n
)

# plot distribution of resampled correlations
ggplot(data = samples, aes(x = cor_sample, y = ..density..)) +
    geom_histogram(colour = "black", binwidth = 0.01) +
    geom_density(colour = "red") +
    labs(x = "Correlation Coefficient", y = NULL,
         title = "Histogram of Random Correlation Coefficients with Normal Curve")
```



7. Show some measures of lexical variety.

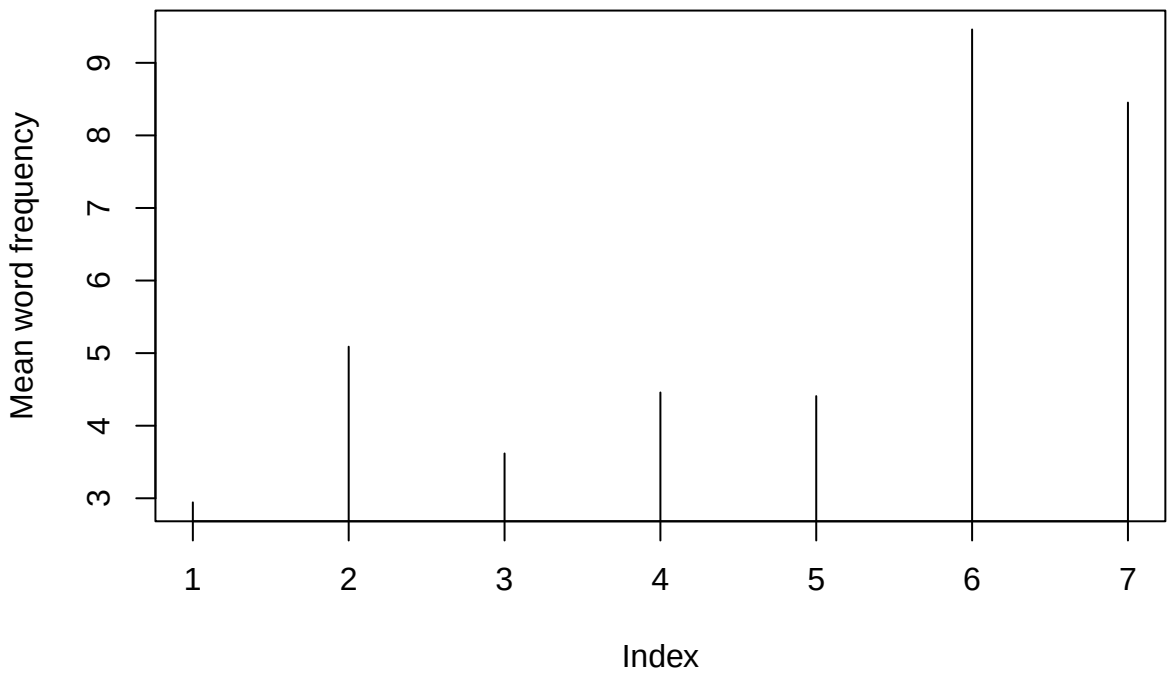
We will calculate the mean word frequency for each chapter, i.e, the number of words in the chapter by size of vocabulary of it. We sort it by the value of the mean calculated, in decreasing order.

```r
mean_words <- (ntoken(chapters_corp) / ntype(chapters_corp))
sort(mean_words, decreasing = TRUE)
```

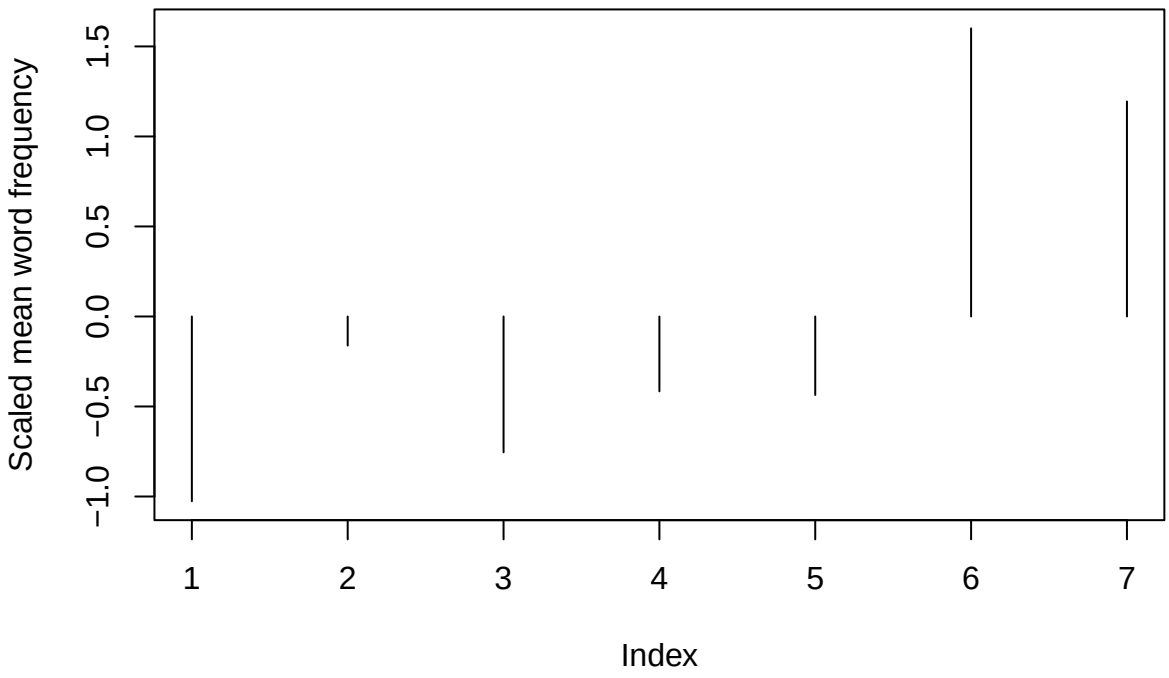
```
##  CHAPTER VI CHAPTER VII  CHAPTER II  CHAPTER IV   CHAPTER V CHAPTER III
##    9.459408    8.452033    5.088442    4.457736    4.407846    3.617195
##   CHAPTER I
##    2.943609
```

We can plot the mean along the number of chapters.

```
# Normal plot
(ntoken(chapters_corp) / ntype(chapters_corp)) %>%
    plot(type = "h", ylab = "Mean word frequency")
```




```
# Scaled plot
(ntoken(chapters_corp) / ntype(chapters_corp)) %>%
    scale() %>%
    plot(type = "h", ylab = "Scaled mean word frequency")
```



The plot suggests that the last two chapters contain a large number of tokens in comparison with the vocabulary size. It means that in the last two chapters there are more repeated words as the ratio is higher.

The TTR (Type Token Ratio), the total number of unique words (types) divided by the total number of words (tokens) can be computed transforming into a `dfm`object each chapter. Then, the function `textstat_lexdiv()` allows to get the ratio for each chapter.

```
dfm(chapters_corp) %>%
    textstat_lexdiv(measure = "TTR")
```

```
## Warning: 'dfm.corpus()' is deprecated. Use 'tokens()' first.
```

```
##      document       TTR
## 1   CHAPTER I 0.3590494
## 2  CHAPTER II 0.2112275
## 3 CHAPTER III 0.2980263
## 4  CHAPTER IV 0.2367652
## 5   CHAPTER V 0.2407978
## 6  CHAPTER VI 0.1146212
## 7 CHAPTER VII 0.1262283
```

As expected, the chapters with lowest RTT are precisely the ones that have the highest mean word. Thus, the last two chapters are the segments with lowest lexical richness, since the closer the TTR ratio is to 1, the greater the lexical richness of the chapter.

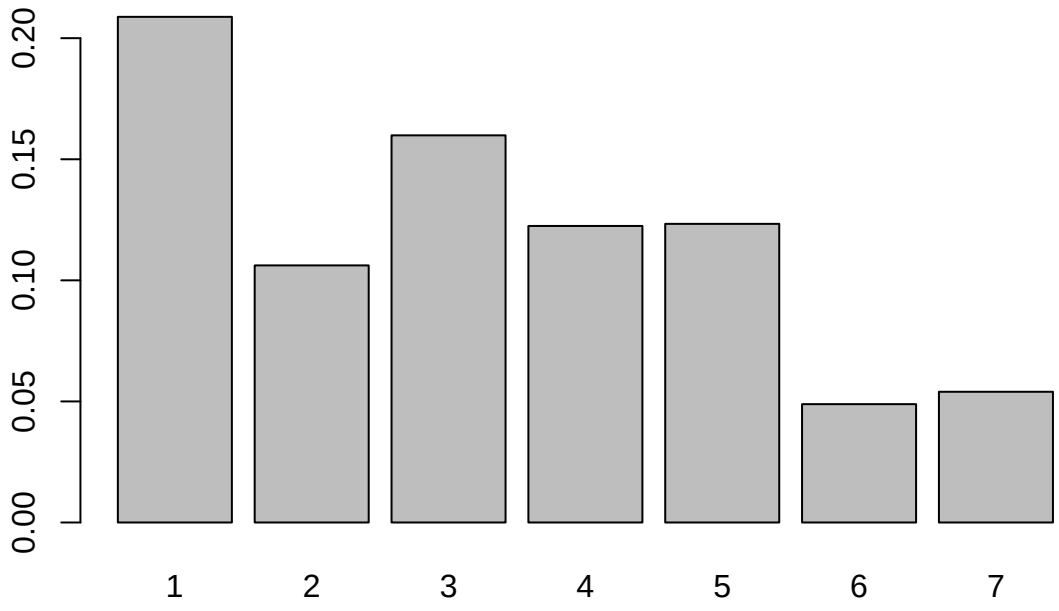
8. Calculate the Hapax Richness.

Hapax Richness measure is defined as the number of words that occur only once divided by the total number of words.

```
# Calculate proportion
hapax_measure <- rowSums(chap_dfm == 1) / ntoken(chap_dfm)
head(hapax_measure)
```

```
##   CHAPTER I  CHAPTER II CHAPTER III  CHAPTER IV   CHAPTER V  CHAPTER VI
##  0.20881226  0.10616235  0.15987550  0.12244898  0.12332250  0.04886568
```

```
# Plot ratios
barplot(hapax_measure, beside = TRUE, col = "grey", names.arg = seq_len(ndoc(chap_dfm)))
```



The barplot suggest the same as before, the last two chapters contain few unique words in comparison with the total number of words in the chapter.