Text Analysis with R for Students of Literature

Ignacio Almodovar Cárdenas and Alejandra Estrada Sanz

22/03/2022

Contents

0. Preprocessing 2
1. Analyse and study the occurrence of words related with love or positive feelings in general . . . 2
2. Make frequency plots 2
3. Compare word frequency data of words like “he”, “she”, “him”, “her” and show also relative frequencies 5
4. Make a token distribution analysis 6
5. Identify chapter breaks 7
7. Show some measures of lexical variety 8
8. Calculate the Hapax Richness 10

0. Preprocessing

First of all, we have to upload our text file. In order to do that we searched for a few books in gutenberg.org and decided to analyze the book "Macbeth".

```
data_macbeth <- texts(readtext("https://www.gutenberg.org/files/1533/1533-0.txt"))
names(data_macbeth) <- "Macbeth"
```

Once we have our book ready to analyze into an R object, we can start working on it. This book contains some metadata both at end and start of the book. Therefore, we separate the content from metadata and extract the header and final leftover information.

```
start_v <- stri_locate_first_fixed(data_macbeth, "SCENE I. An open Place.")[1]
end_v <- stri_locate_last_fixed(data_macbeth, "[_Flourish. Exeunt._]")[1]
novel_v <- stri_sub(data_macbeth, start_v, end_v)
novel_v = gsub("€", "", novel_v)
novel_v = gsub("", "", novel_v)
```

Now that we have the text that we have to analyze selected, we can reprocess the content and put everything in lower case.

```
novel_lower_v <- char_tolower(novel_v)
macbeth_word_v <- tokens(novel_lower_v, remove_punct = TRUE) %>% as.character()
total_length <- length(macbeth_word_v)
```

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

First let's start by counting the number of times the word "love" is repeated.

```
length(macbeth_word_v[which(macbeth_word_v == "love")])
```

```
[1] 19
```

We can also do the same thing using kwic() function. This returns a list, therefore we have to count the number of rows to obtain the number of time the word selected is repeated.

```
nrow(kwic(novel_lower_v, pattern = "love"))
```

```
[1] 19
```

We can also count the number of times that words that begin with "love" are repeated, e.g. "lovers", "loved" or "lover". We can see that there is not much more from words form the family "love".

```
nrow(kwic(novel_lower_v, pattern = "love*"))
```

```
[1] 25
```

Finally we can also compute the ratio for the word "love" between all the total number of words.

```
total_love_hits <- nrow(kwic(novel_lower_v, pattern = "^love{0,1}$", valuetype = "regex"))
total_love_hits / ntoken(novel_lower_v, remove_punct = TRUE)
```

```
      text1
0.001048797
```

2. Make frequency plots

We are now going to obtain the ten most frequent words. Using the function dfm() we can create a matrix of counts of each word type. We can also obtain the frequency of the n most repeated words with the function textstat_frequency()

```
macbeth_dfm <- dfm(novel_lower_v, remove_punct = TRUE)
textstat_frequency(macbeth_dfm, n = 5)
```

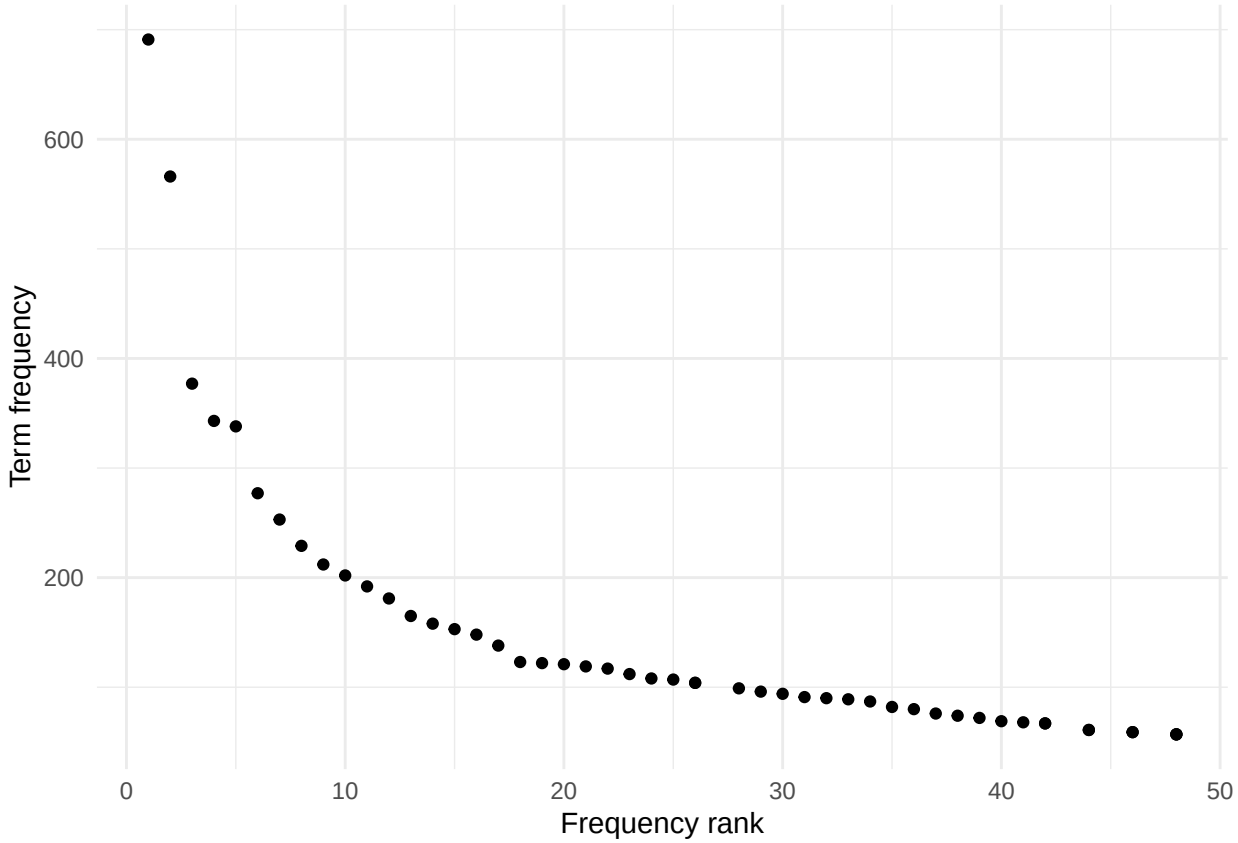
```
  feature frequency rank docfreq group
1     the       691    1       1   all
2     and       566    2       1   all
3      to       377    3       1   all
4      of       343    4       1   all
5       i       338    5       1   all
```

As expected, the most repeated words are the most used prepositions, nouns and conjunctions in the English language.

We can also plot the frequency of the n most frequent terms. Notice that moreorless, the the 20 most repeated over a hundred times.

```
theme_set(theme_minimal())

textstat_frequency(macbeth_dfm, n = 50) %>%
  ggplot(aes(x = rank, y = frequency)) +
  geom_point() +
  labs(x = "Frequency rank", y = "Term frequency")
```

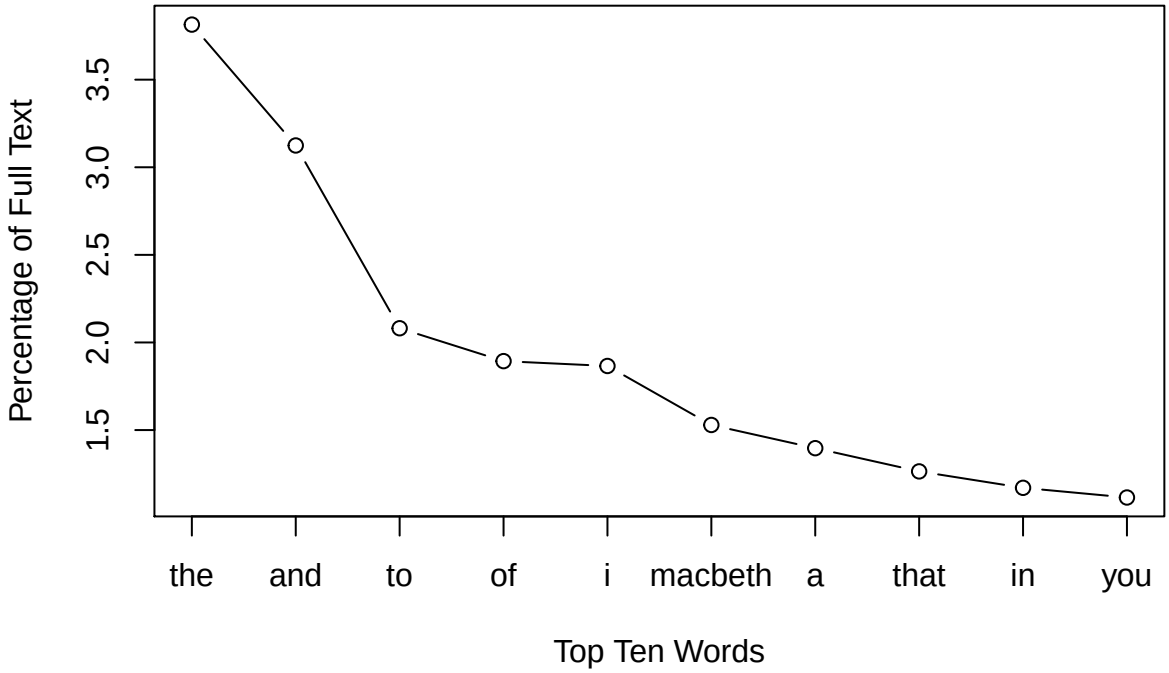



```
sorted_macbeth_freqs_t <- topfeatures(macbeth_dfm, n = nfeat(macbeth_dfm))
```

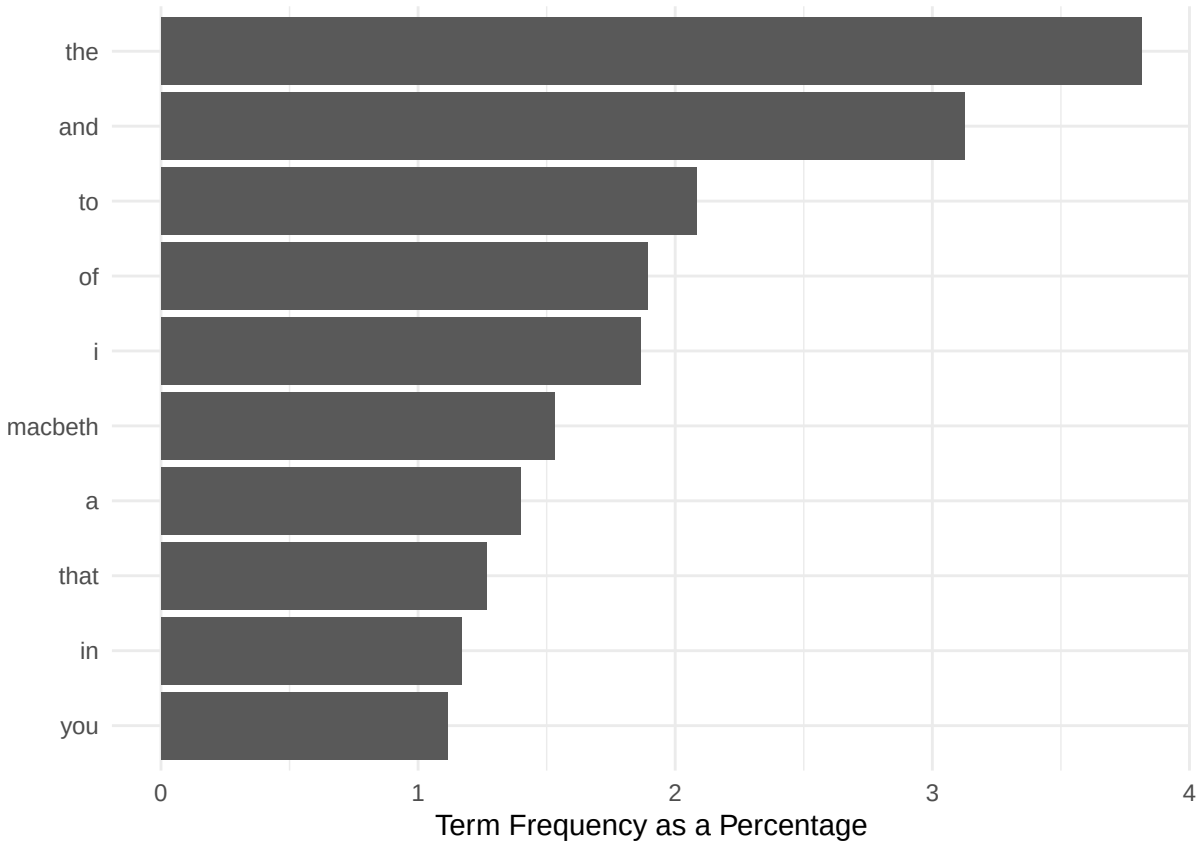
We can also analyze the percentage of full text and the term frequency as a percentage for the ten most frequent words.

```
sorted_macbeth_rel_freqs_t <- sorted_macbeth_freqs_t / sum(sorted_macbeth_freqs_t) * 100
macbeth_dfm_pct <- dfm_weight(macbeth_dfm, scheme = "prop") * 100

plot(sorted_macbeth_rel_freqs_t[1:10], type = "b",
     xlab = "Top Ten Words", ylab = "Percentage of Full Text", xaxt = "n")
axis(1,1:10, labels = names(sorted_macbeth_rel_freqs_t[1:10]))
```




```
textstat_frequency(macbeth_dfm_pct, n = 10) %>%
  ggplot(aes(x = reorder(feature, -rank), y = frequency)) +
  geom_bar(stat = "identity") + coord_flip() +
  labs(x = "", y = "Term Frequency as a Percentage")
```



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

We first have to calculate the frequencies of "he", "she", "him" and "her".

```
sorted_macbeth_freqs_t[c("he", "she", "him", "her")]
```

```
 he she him her
117  19  91  43
```

There exist another method to obtain the same solution, now indexing the dfm.

```
macbeth_dfm[, c("he", "she", "him", "her")]
```

```
Document-feature matrix of: 1 document, 4 features (0.00% sparse) and 0 docvars.
       features
docs     he she him her
  text1 117  19  91  43
```

We can see clearly that the most repeated one between these words is "he", whereas the less one is "she". Therefore, without having any idea about the book we can guess that the main character is a boy and the book is written in third person or that it is written by a women which talks a lot about a men.

We also estimate its relative frequencies.

```
sorted_macbeth_rel_freqs_t["he"]
```

```
       he
0.6458379
```

```
sorted_macbeth_rel_freqs_t["she"]
```

```
      she
0.1048797
```

```
sorted_macbeth_rel_freqs_t["him"]
```

```
      him
0.5023184
```

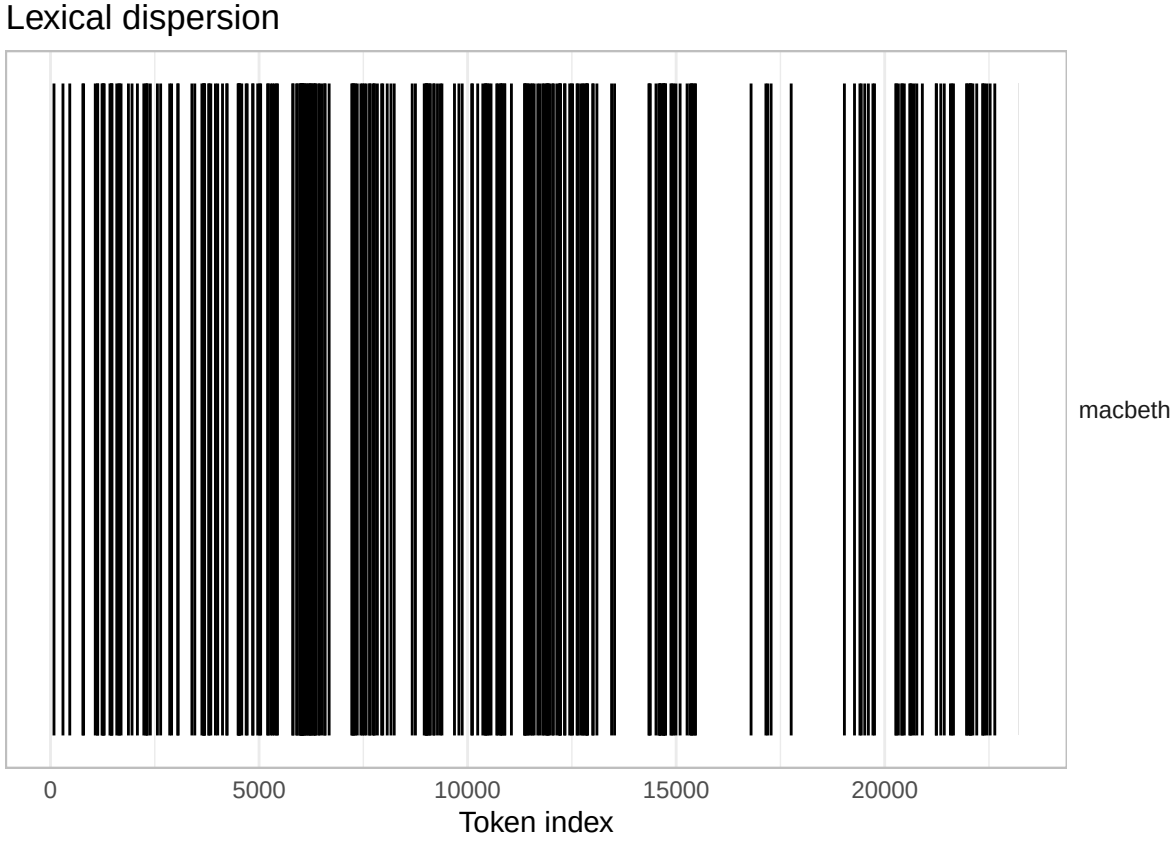
```
sorted_macbeth_rel_freqs_t["her"]
```

```
      her
0.2373592
```

4. Make a token distribution analysis

We can make token distribution analysis by doing dispersion plots. With this dispersion plot we can see the occurrences of particular terms. Let's do them by using words "Macbeth" and "Macduff".

```
textplot_xray(kwic(novel_v, pattern = "macbeth")) +
  ggtitle("Lexical dispersion")
```

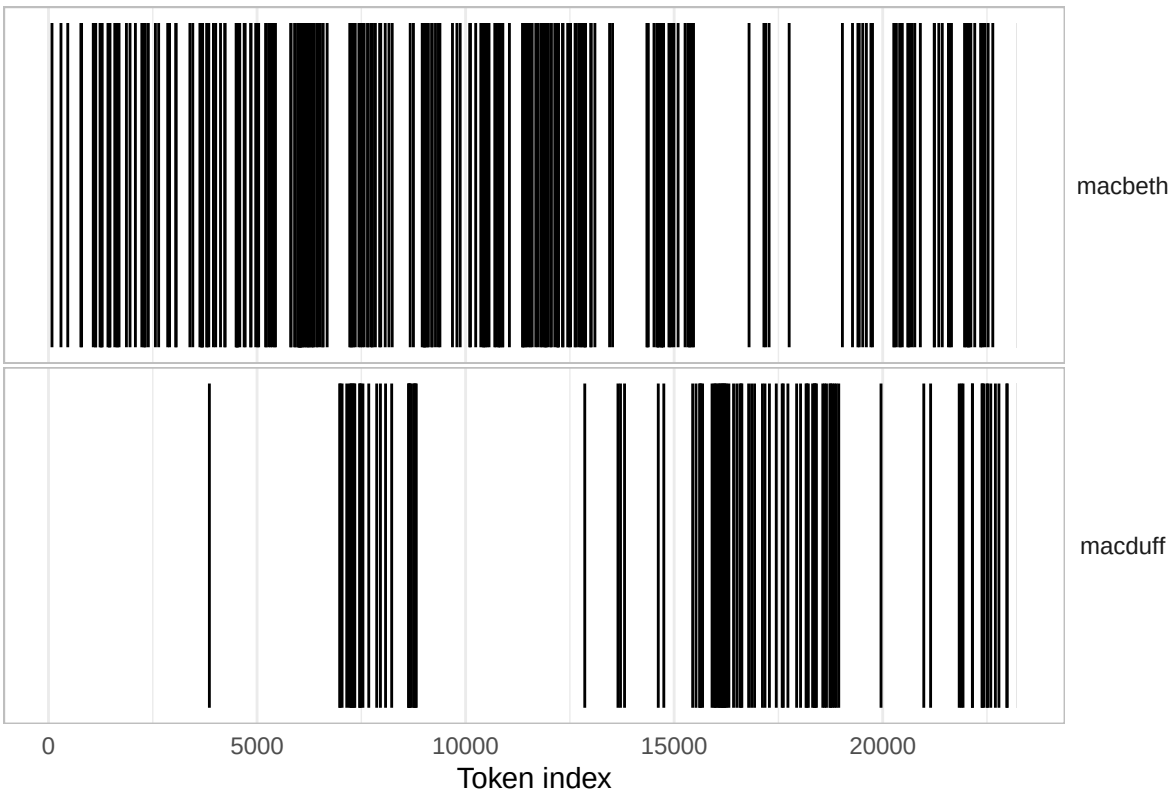


We can see that the word "macbeth" is repeated many times through the book. Looks like this might be the main character of the book.

We are now going to compare it with another character.

```
textplot_xray(
  kwic(novel_v, pattern = "macbeth"),
```

```
  kwic(novel_v, pattern = "macduff")) +
  ggtitle("Lexical dispersion")
```



We can see that "macduff" has to be a secondary character in the novel. Also we can see that it appears more at the middle of the book and end rather than at the beggining.

5. Identify chapter breaks

Now we are going to identify the chapter break locations. For that we are going to use the function kwic() that we have been using through all the analysis. As we know that all the chapters beging with "SCENE", we can obtain the chapter breaks straightforward.

```
chap_positions_v <- kwic(novel_v, phrase(c("SCENE")), valuetype = "regex")$from
chap_positions_v
```

```
 [1]     1   128   802  2410  3039  3829  4195  5083  5802  6716  8400  8878
[13] 10408 11012 11349 13046 13402 13931 15599 16591 19072 19916 20257 20959
[25] 21220 21810 21952 22340
```

Then we can save our chapter breaks in the variable "chapters_corp".

```
chapters_corp <-
  corpus(novel_v) %>%
  corpus_segment(pattern = "SCENE\\s*.*\\n", valuetype = "regex")
summary(chapters_corp, 10)
```

```
Corpus consisting of 28 documents, showing 10 documents:

     Text Types Tokens Sentences
```

```
  text1.1    67    120        25
  text1.2   362    666        52
  text1.3   587   1602       142
  text1.4   316    618        44
  text1.5   372    779        56
  text1.6   203    356        25
  text1.7   414    876        53
  text1.8   336    709        52
  text1.9   363    908       101
 text1.10   617   1678       162
                                        pattern
                        SCENE I. An open Place.\n
                   SCENE II. A Camp near Forres.\n
                                SCENE III. A heath.\n
          SCENE IV. Forres. A Room in the Palace.\n
 SCENE V. Inverness. A Room in Macbeth's Castle.\n
           SCENE VI. The same. Before the Castle.\n
      SCENE VII. The same. A Lobby in the Castle.\n
    SCENE I. Inverness. Court within the Castle.\n
                               SCENE II. The same.\n
                              SCENE III. The same.\n
```

```
docvars(chapters_corp, "pattern") <- stringi::stri_trim_right(docvars(chapters_corp, "pattern"))
docnames(chapters_corp) <- docvars(chapters_corp, "pattern")
```

7. Show some measures of lexical variety

With ndoc(), we obtain the number of chapters included in the book.

```
ndoc(chapters_corp)
```

```
[1] 28
```

With docnames we can obtain chapter names.

```
docnames(chapters_corp) %>% head(4)
```

```
[1] "SCENE I. An open Place."
[2] "SCENE II. A Camp near Forres."
[3] "SCENE III. A heath."
[4] "SCENE IV. Forres. A Room in the Palace."
```

Using the function ntoken() we can calculate the size of the vocabulary for each chapter. However, using ntype() the count is more precise as it counts unique words and excludes numbers and symbols.

```
ntoken(chapters_corp) %>% head(2)
```

```
      SCENE I. An open Place. SCENE II. A Camp near Forres.
                          120                            666
```

```
ntype(chapters_corp,remove_punct = TRUE) %>% head(2)
```

```
##       SCENE I. An open Place. SCENE II. A Camp near Forres.
##                            59                             349
```

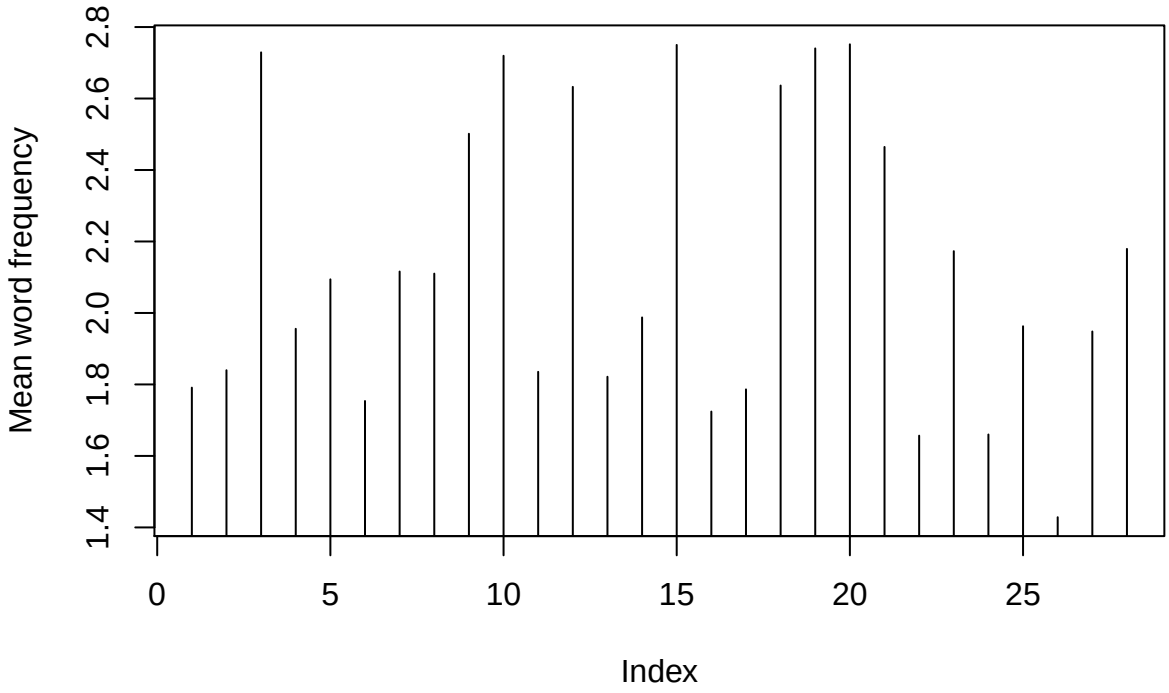
As expected, the number of words using ntype() is lower than the one obtained with ntoken. We can check the difference in proportions:

```
(ntoken(chapters_corp) / ntype(chapters_corp)) %>% head()
```

```
                        SCENE I. An open Place.
                                       1.791045
                   SCENE II. A Camp near Forres.
                                       1.839779
                              SCENE III. A heath.
                                       2.729131
          SCENE IV. Forres. A Room in the Palace.
                                       1.955696
SCENE V. Inverness. A Room in Macbeth's Castle.
                                       2.094086
           SCENE VI. The same. Before the Castle.
                                       1.753695
```

Then, we can also plot the proportion.

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



As we can see each chapter have a very different proportion than the other. therefore there must be very different in terms of length and vocabulary used.

We can also rank the chapters to see which ones are bigger and the Type-Token Ratio (TTR), which is a measure of lexical diversity calculated using textstat_lexdiv().

```
mean_word_use_m <- (ntoken(chapters_corp) / ntype(chapters_corp))
sort(mean_word_use_m, decreasing = TRUE) %>% head()
```

```
          SCENE III. England. Before the King's Palace.
                                                2.751670
     SCENE IV. The same. A Room of state in the Palace.
                                                2.750000
            SCENE II. Fife. A Room in Macduff's Castle.
```

```
                                          2.740223
                                SCENE III. A heath.
                                          2.729131
                               SCENE III. The same.
                                          2.719611
SCENE I. A dark Cave. In the middle, a Cauldron Boiling.
                                          2.636364
```

And finally, we calculate the TTR.

```
dfm(chapters_corp) %>%
  textstat_lexdiv(measure = "TTR") %>%
  head(n = 10)
```

```
                                       document       TTR
1                          SCENE I. An open Place. 0.6321839
2                     SCENE II. A Camp near Forres. 0.6057143
3                               SCENE III. A heath. 0.4000000
4          SCENE IV. Forres. A Room in the Palace. 0.5463710
5  SCENE V. Inverness. A Room in Macbeth's Castle. 0.5039620
6           SCENE VI. The same. Before the Castle. 0.6276596
7      SCENE VII. The same. A Lobby in the Castle. 0.4883402
8     SCENE I. Inverness. Court within the Castle. 0.5151515
9                              SCENE II. The same. 0.4544118
10                            SCENE III. The same. 0.4222573
```

8. Calculate the Hapax Richness

The last analysis that we will do is de Hapax Richness. It is defined as the number of unique words divided by the total number of words. To calculate it we first split it by chapters.

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

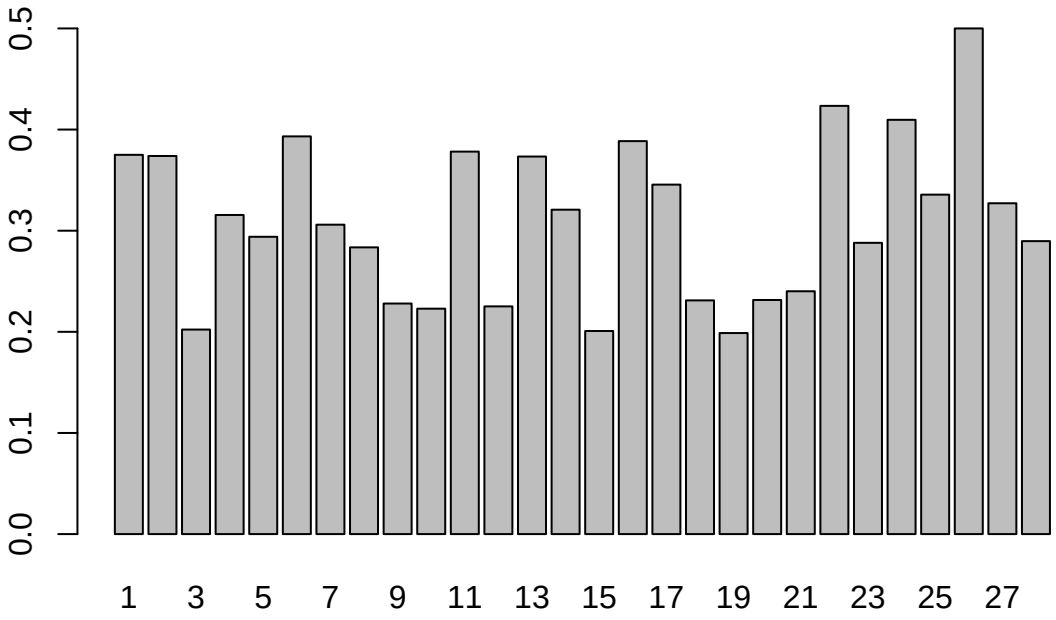
We calculate hapaxes per chapter.

```
rowSums(chap_dfm == 1) %>% head()
```

```
                        SCENE I. An open Place.
                                             45
                   SCENE II. A Camp near Forres.
                                            249
                              SCENE III. A heath.
                                            324
          SCENE IV. Forres. A Room in the Palace.
                                            195
SCENE V. Inverness. A Room in Macbeth's Castle.
                                            229
           SCENE VI. The same. Before the Castle.
                                            140
```

And we calculate them again as a proportion.

```
hapax_proportion <- rowSums(chap_dfm == 1) / ntoken(chap_dfm)
barplot(hapax_proportion, beside = TRUE, col = "grey", names.arg = seq_len(ndoc(chap_dfm)))
```


0.0
0.1
0.2
0.3
0.4
0.5
1
3
5
7
9
11
13
15
17
19
21
23
25
27