Project 3: Text Analytics in R

Ekaterina Galkina & Tejaswi Pala

Big Data & Analytics

The George Washington University

April 2025

Table of contents

Introduction

This report aims to process and analyze a large dataset in the text format. The dataset in question is a part of a book named “The Princess of Mars by Edgar Rice Burroughs” chapters I - XIV (about 60 pages). For these purposes, we are going to use different R libraries, such as tm, wordcloud, quanteda, syuzhet, topicmodels and termineR. The report will first show how to obtain the items asked in question 1. a and then following the rubric will apply new functions to the dataset. For important functions, the following points are going to be discussed: the function’s purpose, the results obtained by applying it to the data set, and what it tells us about the text. Finally, we will conclude by discussion on what this project helped us learn about text analytics.

1. The data set transformation into proper form

First, we have to import the initial text, namely the whole book as it is from Blackboard. Then, we create a VCorpus object and store the whole book’s text in the list of strings called book. Furthermore, we will find all chapters’ (from 1 to 15) indices, to be able further to separate chapters that we need in separate files. Figure 1.1 shows these actions.

```r
library(tm)
library(rstudioapi) # To be able to select a directory
library(wordcloud)
library(quanteda)
library(syuzhet)
library(topicmodels)
library(textmineR)

path <- selectDirectory()

text <- VCorpus(DirSource(path, ignore.case = TRUE, mode = "text"))
book <- as.character(text[[1]])

chapter_titles <- paste("CHAPTER", as.roman(1:15))

chapter_indices <- sapply(chapter_titles, function(title) {
  which(book == title)[1]
})
```

Figure 1.1: Data preparation: step 1: Importing the text

We can see on Figure 1.2 that all chapters that we need to study, start indices were stored correctly.

```
> chapter_indices
   CHAPTER I   CHAPTER II  CHAPTER III   CHAPTER IV    CHAPTER V   CHAPTER VI  CHAPTER VII CHAPTER VIII
           2         270          437          706          928         1081         1247         1452
  CHAPTER IX    CHAPTER X   CHAPTER XI  CHAPTER XII CHAPTER XIII  CHAPTER XIV   CHAPTER XV
        1660         1809         2168         2427         2632         2877         3228
>
```

Figure 1.2: Starting indices per chapter

Then, we will create a new directory, and 14 files for each chapter, these files we populate according to the chapter in the book. The code to perform these actions is depicted in Figure 1.3.

```r
dir.create(file.path(path, "Chapters"))

for (i in 1:14) {
  chapter_lines <- book[(chapter_indices[i] + 1):(chapter_indices[i + 1] - 1)]
  file_name <- sprintf("chapter_%02d.txt", i)
  write.table(chapter_lines, file = file.path(path, "Chapters", file_name),
              sep = "\t", row.names = FALSE, col.names = FALSE, quote = FALSE)
}
```

Figure 1.3: Data preparation: step 2: Creating folder with chapters

In this case, we have created this directory inside the project 3 directory (where our source code is) and we can see by going to this directory after the previously shown code execution, that everything worked well and all chapters are well saved. The results are shown on Figures 1.4.1 and 1.4.2.

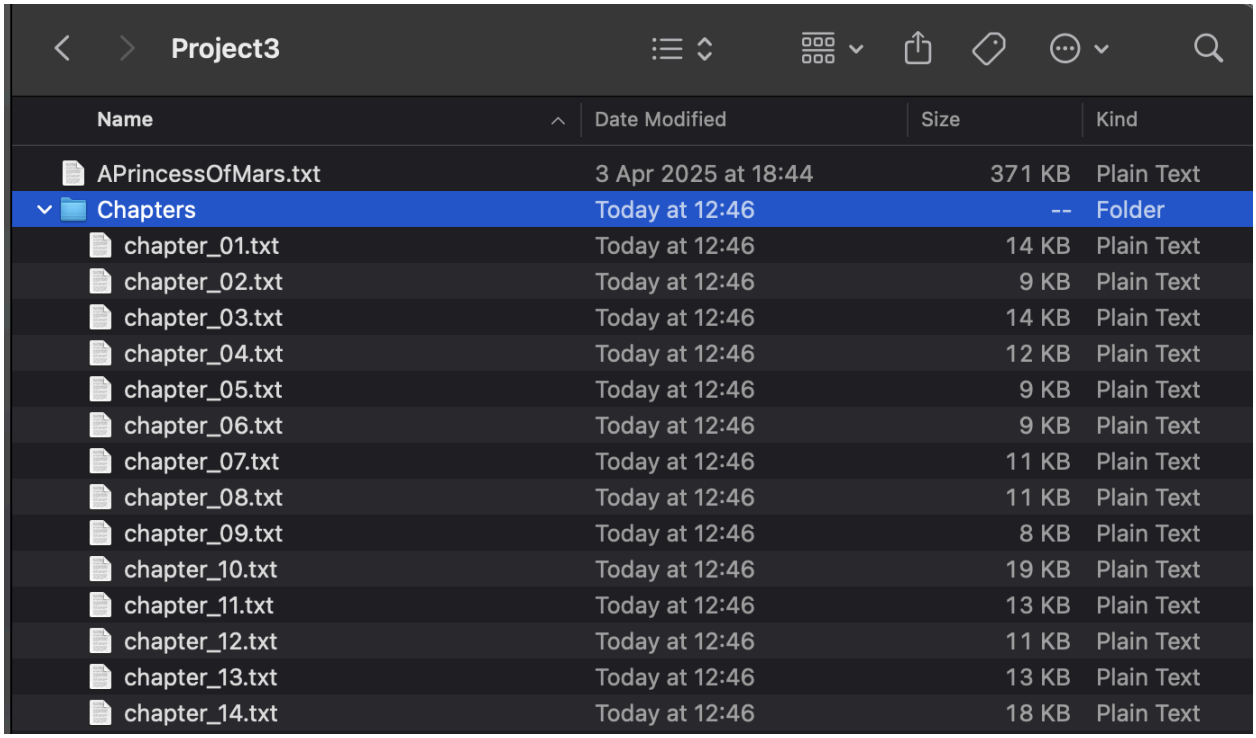


Figure 1.4: Created folder with chapters

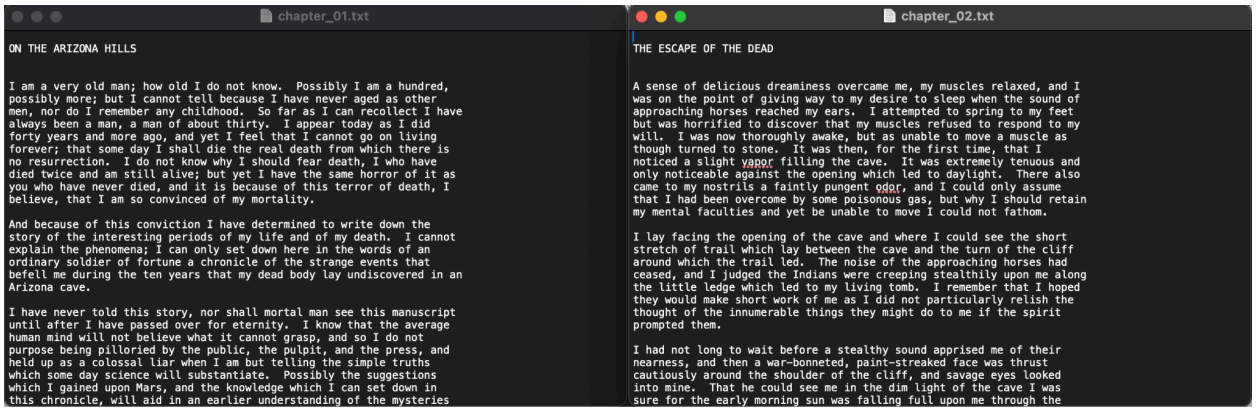


Figure 1.5: Example of contents of files: Chapters 1 and 2

Furthermore, we can finally create a proper VCorpus, where each document is a file with its number chapter of the book, the source code to do this, as well as the summary function launched on the newly created VCorpus object to confirm that everything works well are shown on Figure 1.5.

```
> chapters_dir <- file.path(path, "Chapters")
> corpus_chapters <- VCorpus(DirSource(chapters_dir, ignore.case = T, mode = "text"))
> summary(corpus_chapters)
               Length Class             Mode
chapter_01.txt 2      PlainTextDocument list
chapter_02.txt 2      PlainTextDocument list
chapter_03.txt 2      PlainTextDocument list
chapter_04.txt 2      PlainTextDocument list
chapter_05.txt 2      PlainTextDocument list
chapter_06.txt 2      PlainTextDocument list
chapter_07.txt 2      PlainTextDocument list
chapter_08.txt 2      PlainTextDocument list
chapter_09.txt 2      PlainTextDocument list
chapter_10.txt 2      PlainTextDocument list
chapter_11.txt 2      PlainTextDocument list
chapter_12.txt 2      PlainTextDocument list
chapter_13.txt 2      PlainTextDocument list
chapter_14.txt 2      PlainTextDocument list
>
```

Figure 1.6: Creating the right VCorpus object

2. Longest word and sentence of each chapter

Furthermore, before cleaning the text, we have to find the 10 longest words and 10 longest sentences in each chapter. To do this, we are going to iterate through each file (chapter) in our just created “Chapters” directory and for each we first store the text in pure string format, then we use a pre-build R function get_sentences() from the rubric to get the list of all sentences and store this information in a sentences_chap variable, and then we split this same current chapter into words, by using strsplit() function found on the internet, with the argument “\\W+” (it separates the text based on the non-alphabetic characters), we store this in a list called

words_chap. Then we filter found words, by removing empty strings from words_chap. Then we sort two obtained lists (of sentences and of words) in decreasing order so at the beginning of both lists we have the longest word and sentence accordingly for each chapter. And we print these values. The code below (Figure 2.1), shows these actions and Figure 2.2 depicts the beginning of the output with these values for the first 3 chapters. Then, you can find the entire data (longest word and sentence for each chapters from 1 to 14) on Table 2.3.

```
33  # Question 1. a
34
35  for (i in 1:14){
36    chap <- get_text_as_string(file.path(chapters_dir, sprintf("chapter_%02d.txt", i)))
37    sentences_chap <- get_sentences(chap)
38    # we are splitting once we see a non alphabetic character :
39    words_chap <- unlist(strsplit(chap, "\\W+"))
40    words_chap <- words_chap[nchar(words_chap) > 0] # eliminating empty strings
41    # We sort words and sentences by their number of characters
42    sorted_words <- words_chap[order(nchar(words_chap), decreasing = TRUE)]
43    sorted_sentences <- sentences_chap[order(nchar(sentences_chap), decreasing = TRUE)]
44
45    cat("For chapter", as.roman(i), " -> \n")
46    cat(" Longest word: ", sorted_words[1], "\n")
47    cat(" Longest sentence: ", sorted_sentences[1], "\n\n")
48  }
```

Figure 2.1: Extracting the longest word and sentence per chapter

```
For chapter 1  ->
 Longest word:  subconsciously
 Longest sentence:  However, I am not prone to sensitiveness, and the following of a sense of duty, wherever it
may lead, has always been a kind of fetich with me throughout my life; which may account for the honors bestowed
upon me by three republics and the decorations and friendships of an old and powerful emperor and several lesser
kings, in whose service my sword has been red many a time.

For chapter 2  ->
 Longest word:  contemplation
 Longest sentence:  Few western wonders are more inspiring than the beauties of an Arizona moonlit landscape; th
e silvered mountains in the distance, the strange lights and shadows upon hog back and arroyo, and the grotesque
details of the stiff, yet beautiful cacti form a picture at once enchanting and inspiring; as though one were ca
tching for the first time a glimpse of some dead and forgotten world, so different is it from the aspect of any
other spot upon our earth.

For chapter 3  ->
 Longest word:  characteristics
 Longest sentence:  He sat his mount as we sit a horse, grasping the animal's barrel with his lower limbs, while
the hands of his two right arms held his immense spear low at the side of his mount; his two left arms were outs
tretched laterally to help preserve his balance, the thing he rode having neither bridle or reins of any descrip
tion for guidance.
```

Figure 2.2: Extracting the longest word and sentence per chapter: Output for first 3 chapters

Chapter number	Word	Sentence
Chapter 1	subconsciously	However, I am not prone to sensitiveness, and the following of a sense of duty, wherever it may lead, has always been a kind of fetich with me throughout my life; which may account for the honors bestowed upon me by three republics and the decorations and friendships of an old and powerful emperor and several lesser kings, in whose service my sword has been red many a time.
Chapter 2	contemplation	Few western wonders are more inspiring than the beauties of an Arizona moonlit landscape; the silvered mountains in the distance, the strange lights and shadows upon hog back and arroyo, and the grotesque details of the stiff, yet beautiful cacti form a picture at once enchanting and inspiring; as though one were catching for the first time a glimpse of some dead and forgotten world, so different is it from the aspect of any other spot upon our earth
Chapter 3	characteristics	He sat his mount as we sit a horse, grasping the animal's barrel with his lower limbs, while the hands of his two right arms held his immense spear low at the side of his mount; his two left arms were outstretched laterally to help preserve his balance, the thing he rode having neither bridle or reins of any description for guidance.
Chapter 4	circumstances	I saw no signs of extreme age among them, nor is there any appreciable difference in their appearance from the age of maturity, about forty, until, at about the age of one thousand years, they go voluntarily upon their last strange pilgrimage down the river Iss, which leads no living Martian knows whither and from whose bosom

		no Martian has ever returned, or would be allowed to live did he return after once embarking upon its cold, dark waters.
Chapter 5	characteristics	The nights are either brilliantly illumined or very dark, for if neither of the two moons of Mars happen to be in the sky almost total darkness results, since the lack of atmosphere, or, rather, the very thin atmosphere, fails to diffuse the starlight to any great extent; on the other hand, if both of the moons are in the heavens at night the surface of the ground is brightly illuminated.
Chapter 6	overwhelmingly	My beast had an advantage in his first hold, having sunk his mighty fangs far into the breast of his adversary; but the great arms and paws of the ape, backed by muscles far transcending those of the Martian men I had seen, had locked the throat of my guardian and slowly were choking out his life, and bending back his head and neck upon his body, where I momentarily expected the former to fall limp at the end of a broken neck.
Chapter 7	representative	Between these walls the little Martians scampered, wild as deer; being permitted to run the full length of the aisle, where they were captured one at a time by the women and older children; the last in the line capturing the first little one to reach the end of the gauntlet, her opposite in the line capturing the second, and so on until all the little fellows had left the enclosure and been appropriated by some youth or female.
Chapter 8	reinforcements	For example, a proportion of them, always the best marksmen, direct their fire entirely upon the wireless finding and sighting apparatus of the big guns of an attacking naval force; another detail attends to the smaller guns in the same way; others pick off the gunners; still others the officers; while certain other quotas concentrate their attention upon the other members of the crew, upon the upper works, and upon the steering gear and propellers.
Chapter 9	responsibilities	With this added incentive I nearly drove Sola distracted by my importunities to hasten on my education and within a few more days I had mastered the Martian tongue sufficiently well to enable me to carry on a passable conversation and to fully understand practically all that I heard.
Chapter 10	responsibilities	What words of moment were to have fallen from his lips were never spoken, as just then a young warrior, evidently sensing the trend of thought among the older men, leaped down from the steps of the rostrum, and striking the frail captive a powerful blow across the face, which felled her to the floor, placed his foot upon her prostrate form and turning toward the assembled council broke into peals of horrid, mirthless laughter.
Chapter 11	accouterments	During the ages of hardships and incessant warring between their own various races, as well as with the green men, and before they had fitted themselves to the changed conditions, much of the high civilization and many of the arts of the fair-haired Martians had

		become lost; but the red race of today has reached a point where it feels that it has made up in new discoveries and in a more practical civilization for all that lies irretrievably buried with the ancient Barsoomians, beneath the countless intervening ages.
Chapter 12	personification	While the court was entirely overgrown with the yellow, moss-like vegetation which blankets practically the entire surface of Mars, yet numerous fountains, statuary, benches, and pergola-like contraptions bore witness to the beauty which the court must have presented in bygone times, when graced by the fair-haired, laughing people whom stern and unalterable cosmic laws had driven not only from their homes, but from all except the vague legends of their descendants.
Chapter 13	notwithstanding	Following the battle with the air ships, the community remained within the city for several days, abandoning the homeward march until they could feel reasonably assured that the ships would not return; for to be caught on the open plains with a cavalcade of chariots and children was far from the desire of even so warlike a people as the green Martians.
Chapter 14	disconsolately	Another thing I saw, too, which almost lost my life for me then and there, for it took my mind for the fraction of an instant entirely from my antagonist; for, as Dejah Thoris struck the tiny mirror from her hand, Sarkoja, her face livid with hatred and baffled rage, whipped out her dagger and aimed a terrific blow at Dejah Thoris; and then Sola, our dear and faithful Sola, sprang between them; the last I saw was the great knife descending upon her shielding breast.

Table 2.3: The longest word and sentence per chapter

3. Application of functions from the Rubric

Now let's begin analyzing our text using the functions introduced in the Introduction to Text Analytics rubric.

General analysis

First, we can see the structure of the VCorpus object we previously created, which includes the 14 chapters we want to analyze. Figure 3.1 shows the beginning of the output of the str() function applied to this object (only the first 2 documents of the object to see what this looks like).

We can see that the object belongs to the classes VCorpus and Corpus, and its content is a list of 14 elements, each corresponding to a document (one chapter). Each of these documents has two main components: content, which stores the text lines of the chapter (for example, for chapter 1 we can see that it has 267 lines, but we also can see that some of them are empty), and meta, a list of metadata attributes. For instance, among these attributes, we can find the id of the documents, named like the files, and the language (all marked as "en" - english).

```
> str(corpus_chapters) # The data structure
Classes 'VCorpus', 'Corpus'  hidden list of 3
 $ content:List of 14
  ..$ :List of 2
  .. ..$ content: chr [1:267] "" "ON THE ARIZONA HILLS" "" "" ...
  .. ..$ meta   :List of 7
  .. .. ..$ author       : chr(0)
  .. .. ..$ datetimestamp: POSIXlt[1:1], format: "2025-04-23 19:36:30"
  .. .. ..$ description  : chr(0)
  .. .. ..$ heading      : chr(0)
  .. .. ..$ id           : chr "chapter_01.txt"
  .. .. ..$ language     : chr "en"
  .. .. ..$ origin       : chr(0)
  .. .. ..- attr(*, "class")= chr "TextDocumentMeta"
  .. ..- attr(*, "class")= chr [1:2] "PlainTextDocument" "TextDocument"
  ..$ :List of 2
  .. ..$ content: chr [1:166] "" "THE ESCAPE OF THE DEAD" "" "" ...
  .. ..$ meta   :List of 7
  .. .. ..$ author       : chr(0)
  .. .. ..$ datetimestamp: POSIXlt[1:1], format: "2025-04-23 19:36:30"
  .. .. ..$ description  : chr(0)
  .. .. ..$ heading      : chr(0)
  .. .. ..$ id           : chr "chapter_02.txt"
  .. .. ..$ language     : chr "en"
  .. .. ..$ origin       : chr(0)
  .. .. ..- attr(*, "class")= chr "TextDocumentMeta"
  .. ..- attr(*, "class")= chr [1:2] "PlainTextDocument" "TextDocument"
```

Figure 3.1: str() on the data set in VCorpus object

Next, we explore our corpus further by running the inspect() function on the VCorpus object. As shown on Figure 3.2.1, this function provides some additional information on each document in the corpus. We can see the beginning of this output, which includes the information about the first 3 chapters. Specifically, we can see that the first three chapters contain 13962, 9000, and

14190 characters respectively. The full set of character counts for all chapters is presented on Table 3.2.2.

```
> inspect(corpus_chapters)
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 14

[[1]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 13962

[[2]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 9000

[[3]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 14190
```

Figure 3.2.1: inspect() on the data set in VCorpus object

Chapter number	Number of characters
1	13962
2	9000
3	14190
4	11830
5	8382
6	8857
7	11222
8	11290
9	7768
10	19113

11	13151
12	10485
13	12508
14	17282

Figure 3.2.2: Number of characters per chapter

We can also print the VCorpus object itself (which output is shown on Figure 3.3), but no new information is available here (after the first 2 functions), we can also see one more time its class and that it has 14 documents.

```
> corpus_chapters
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 14
>
```

Figure 3.3: Printing the VCorpus object

We can also extract the text itself contained in the documents of the corpus. For instance, the commands to extract chapter 1 are shown on the next Figure 3.4.

```
> chap1 <- corpus_chapters[[1]]
> chap1
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 13962
> chap1[1]
$content
  [1] ""
  [2] "ON THE ARIZONA HILLS"
  [3] ""
  [4] ""
  [5] "I am a very old man; how old I do not know.  Possibly I am a hundred,"
  [6] "possibly more; but I cannot tell because I have never aged as other"
  [7] "men, nor do I remember any childhood.  So far as I can recollect I have"
  [8] "always been a man, a man of about thirty.  I appear today as I did"
  [9] "forty years and more ago, and yet I feel that I cannot go on living"
 [10] "forever; that some day I shall die the real death from which there is"
 [11] "no resurrection.  I do not know why I should fear death, I who have"
 [12] "died twice and am still alive; but yet I have the same horror of it as"
 [13] "you who have never died, and it is because of this terror of death, I"
 [14] "believe, that I am so convinced of my mortality."
 [15] ""
 [16] "And because of this conviction I have determined to write down the"
```

Figure 3.4: Extracting the text itself from the VCorpus object: Chapter 1

Furthermore, we can construct a Document-Term Matrix (DTM) from our corpus. A Document-Term Matrix is a table where each row corresponds to a document (in our case, a chapter) and each column corresponds to a term (a word). Each cell contains the frequency of that term in the respective document. Figure 3.5 shows the command used to create this matrix, as well as the structure of the resulting object.

From the output, we observe that the matrix includes 14 documents (chapters) and 6000 unique terms (words). The matrix contains 84,000 cells in total (14 x 6000), but only 12084 of these cells are non-zero, meaning that the majority of words do not appear in the text (which is quite intuitive), this also suggested by the high value of the sparsity - 86%. Finally, we can also see the maximal term length - 19, so the longest word is composed of 19 characters.

```
> chaptersDTM <- DocumentTermMatrix(corpus_chapters)
> chaptersDTM
<<DocumentTermMatrix (documents: 14, terms: 6000)>>
Non-/sparse entries: 12084/71916
Sparsity           : 86%
Maximal term length: 19
Weighting          : term frequency (tf)
>
```

Figure 3.5: Creating Document Term Matrix

We can get some additional information on our Document Term Matrix by running the inspect() function on it. We can see the output of this command on Figure 3.6. The first part of the output is the same as when we simply print the object, but then we can see some of the actual terms and the corresponding number of their appearances in the chapters. For instance, in chapter 1, the word “the” appears 190 times, “and” - 90, and “that” - 50.

```
> inspect(chaptersDTM)
<<DocumentTermMatrix (documents: 14, terms: 6000)>>
Non-/sparse entries: 12084/71916
Sparsity           : 86%
Maximal term length: 19
Weighting          : term frequency (tf)
Sample             :
               Terms
Docs             and but for had that the upon was which with
  chapter_01.txt  90  13  20  25   50 190   12  36    20   25
  chapter_03.txt  92  15  18  19   26 166   20  36    34   20
  chapter_04.txt  68  21  10  22   19 148   10  26    16   20
  chapter_07.txt  64   9  18  16   12 166    7  17    22   12
  chapter_08.txt  81   7  15  24   11 184   22  32    17   11
  chapter_10.txt 113  18  34  41   50 193   18  54    15   26
  chapter_11.txt  82  14   9  29   37 119   13  18    13   23
  chapter_12.txt  58   9  16  13   19 131    8  16    18   20
  chapter_13.txt  63  11  17  29   39 131   15  21    11   29
  chapter_14.txt 115  17  41  35   43 171   18  49     9   33
```

Figure 3.6: inspect() on Document Term Matrix

Furthermore, we can also run the str() function to see the structure of our DocumentTermMatrix object. The output is shown on Figure 3.7. From this screenshot, we can see a list of 6 attributes. Among the interesting and new information, we can see the first 3 attributes named i, j, and v, which respectively represent the document index (which goes from 1 to 14: our chapters), the term index (which goes from 1 to 6000: our unique words), and the frequency - how many times the word appears in that chapter. For example, the 67th word (j[1] = 67) in chapter 1 (i[1] = 1) appears only once (v[1] = 1).

We can also see the dimnames variable, which contains a list of documents and terms' names. We can notice from the second list that special characters were sometimes counted as part of the words.

```
> str(chaptersDTM)
List of 6
 $ i       : int [1:12084] 1 1 1 1 1 1 1 1 1 1 ...
 $ j       : int [1:12084] 67 69 70 75 77 100 109 116 120 125 ...
 $ v       : num [1:12084] 1 1 1 1 7 1 1 4 1 1 ...
 $ nrow    : int 14
 $ ncol    : int 6000
 $ dimnames:List of 2
  ..$ Docs : chr [1:14] "chapter_01.txt" "chapter_02.txt" "chapter_03.txt" "chapter_04.txt" ...
  ..$ Terms: chr [1:6000] "'gentleman'" "'til" "\"and" "\"as" ...
 - attr(*, "class")= chr [1:2] "DocumentTermMatrix" "simple_triplet_matrix"
 - attr(*, "weighting")= chr [1:2] "term frequency" "tf"
>
```

Figure 3.7: str() on Document Term Matrix

We can also create a Term-Document Matrix. In this case, we have the transpose of the previous matrix, meaning that now the rows represent unique terms, and the columns represent documents. The command used to create the Term-Document Matrix and the printed object are shown on Figure 3.8.

```
> chaptersTDM <- TermDocumentMatrix(corpus_chapters)
> chaptersTDM
<<TermDocumentMatrix (terms: 6000, documents: 14)>>
Non-/sparse entries: 12084/71916
Sparsity           : 86%
Maximal term length: 19
Weighting          : term frequency (tf)
>
```

Figure 3.8: Creating Term Document Matrix

Now, it is time to clean our text for further analysis. The next screenshot (Figure 3.9) shows the function we created to remove punctuation, as well as the command used to apply it to our corpus object containing the text to analyze.

```
> removeNumPunct <- function(x) gsub("[^[:alpha:][:space:]]*","",x)
> corpus_chapters_clean <- tm::tm_map(corpus_chapters, content_transformer(removeNumPunct))
```

Figure 3.9: Removing punctuation

We can run the inspect() function on this updated corpus and see that the number of characters per chapter has indeed decreased. The output for the first 3 chapters is shown below on Figure 3.10.1, and the full updated set of character counts for all chapters is presented on Table 3.10.2.

```
> inspect(corpus_chapters_clean)
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 14

[[1]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 13711

[[2]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 8844

[[3]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 13921
```

Figure 3.10.1: inspect() on a cleaned VCorpus object: first 3 chapters

Chapter number	Number of characters
1	13711
2	8844
3	13921
4	11591
5	8210
6	8679
7	11012
8	11090
9	7617
10	18669
11	12812
12	10221

13	12200
14	16816

Table 3.10.2: The updated set of character counts per chapter

We will proceed further in the cleaning process by converting all text to lowercase. The command used to perform this operation is shown below in Figure 3.11. But, after this step, the number of terms should stay the same, because while creating the VCorpus, we precised in the arguments that we are not taking into account uppercase letters.

```
> corpus_chapters_clean_low <- tm_map(corpus_chapters_clean, tm::content_transformer(tolower))
> corpus_chapters_clean_low
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 14
```

Figure 3.11: Removing capital letters

Now, let's create a Document Term Matrix using our newly cleaned corpus, as shown on Figure 3.12, to observe how our data has changed. From this output, we can see that the number of unique terms has decreased from 6000 to 4484. Additionally, the sparsity value has decreased, and the longest term now contains 17 characters.

```
> chapters_clean_dtm <- DocumentTermMatrix(corpus_chapters_clean_low)
> chapters_clean_dtm
<<DocumentTermMatrix (documents: 14, terms: 4484)>>
Non-/sparse entries: 10675/52101
Sparsity           : 83%
Maximal term length: 17
Weighting          : term frequency (tf)
>
```

Figure 3.12: Creating Document Term Matrix on a cleaned version

We can also run the str() function (Figure 3.13), and this time, compared to the previous output, we can see that all words appear without special characters and are well divided.

```
> str(chapters_clean_DTM)
List of 6
 $ i       : int [1:10675] 1 1 1 1 1 1 1 1 1 1 ...
 $ j       : int [1:10675] 3 5 25 33 39 41 45 74 76 87 ...
 $ v       : num [1:10675] 1 7 1 1 4 1 1 1 1 3 ...
 $ nrow    : int 14
 $ ncol    : int 4484
 $ dimnames:List of 2
  ..$ Docs : chr [1:14] "chapter_01.txt" "chapter_02.txt" "chapter_03.txt" "chapter_04.txt" ...
  ..$ Terms: chr [1:4484] "abandoning" "ability" "able" "ablest" ...
 - attr(*, "class")= chr [1:2] "DocumentTermMatrix" "simple_triplet_matrix"
 - attr(*, "weighting")= chr [1:2] "term frequency" "tf"
>
```

Figure 3.13: str() on a new Document Term Matrix

Next, we can convert our Document Term Matrix to a standard matrix using the as.matrix() function, and print a subset of the result for the first 10 terms, as shown on Figure 3.14 just below. In this output, we can see the frequency of terms for our 14 chapters. Each row represents a chapter, and each column represents a unique term. For example, the word "ability" appears twice in chapter 4, once in chapter 5 and so on. We can see that this time the words are sorted in the alphabetic order.

```
> words_freq <- as.matrix(chapters_clean_DTM)
> words_freq[, 1: 10]
               Terms
Docs            abandoning ability able ablest about above abreast abruptly absence absent
  chapter_01.txt          0       0    1      0     7     0       0        0       0      0
  chapter_02.txt          0       0    0      0     0     0       0        0       0      0
  chapter_03.txt          0       0    0      0     8     3       0        0       0      0
  chapter_04.txt          0       2    0      0    10     1       0        1       0      0
  chapter_05.txt          0       1    1      0     5     0       0        0       0      0
  chapter_06.txt          0       0    0      0     1     0       0        0       1      0
  chapter_07.txt          0       0    1      0     8     0       1        0       0      0
  chapter_08.txt          0       0    0      0     2     3       0        0       0      0
  chapter_09.txt          0       0    0      0     1     0       0        0       0      0
  chapter_10.txt          0       2    0      0     0     3       0        0       0      0
  chapter_11.txt          0       0    0      0     2     0       0        0       2      0
  chapter_12.txt          0       0    0      1     2     1       0        0       0      0
  chapter_13.txt          1       0    0      0     0     0       0        0       1      1
  chapter_14.txt          0       1    1      0     4     1       0        0       0      0
>
```

Figure 3.14: DTM presented as matrix

Going further, we will continue to clean our text by removing stop words. Stop words are common words that do not contribute significantly to the meaning of a text. These words usually include pronouns, prepositions, articles, and auxiliary verbs. They are removed to focus on the more meaningful frequent words in the analysis, which we are planning to do. For example, words like "i," "the," "and," "but," and "is" are stop words because they appear very frequently but do not provide a lot of useful information for text analysis.

In the tm library, we can access a list of common English stop words. Figure 3.15 shows how we store these words in a variable and print them to see the full list.

```
> my_stop_words <- c(tm::stopwords("english"))
> my_stop_words
  [1] "i"          "me"         "my"         "myself"     "we"         "our"        "ours"       "ourselves"  "you"
 [10] "your"       "yours"      "yourself"   "yourselves" "he"         "him"        "his"        "himself"    "she"
 [19] "her"        "hers"       "herself"    "it"         "its"        "itself"     "they"       "them"       "their"
 [28] "theirs"     "themselves" "what"       "which"      "who"        "whom"       "this"       "that"       "these"
 [37] "those"      "am"         "is"         "are"        "was"        "were"       "be"         "been"       "being"
 [46] "have"       "has"        "had"        "having"     "do"         "does"       "did"        "doing"      "would"
 [55] "should"     "could"      "ought"      "i'm"        "you're"     "he's"       "she's"      "it's"       "we're"
 [64] "they're"    "i've"       "you've"     "we've"      "they've"    "i'd"        "you'd"      "he'd"       "she'd"
 [73] "we'd"       "they'd"     "i'll"       "you'll"     "he'll"      "she'll"     "we'll"      "they'll"    "isn't"
 [82] "aren't"     "wasn't"     "weren't"    "hasn't"     "haven't"    "hadn't"     "doesn't"    "don't"      "didn't"
 [91] "won't"      "wouldn't"   "shan't"     "shouldn't"  "can't"      "cannot"     "couldn't"   "mustn't"    "let's"
[100] "that's"     "who's"      "what's"     "here's"     "there's"    "when's"     "where's"    "why's"      "how's"
[109] "a"          "an"         "the"        "and"        "but"        "if"         "or"         "because"    "as"
[118] "until"      "while"      "of"         "at"         "by"         "for"        "with"       "about"      "against"
[127] "between"    "into"       "through"    "during"     "before"     "after"      "above"      "below"      "to"
[136] "from"       "up"         "down"       "in"         "out"        "on"         "off"        "over"       "under"
[145] "again"      "further"    "then"       "once"       "here"       "there"      "when"       "where"      "why"
[154] "how"        "all"        "any"        "both"       "each"       "few"        "more"       "most"       "other"
[163] "some"       "such"       "no"         "nor"        "not"        "only"       "own"        "same"       "so"
[172] "than"       "too"        "very"
>
```

Figure 3.15: Stop words from the tm library

The next screenshot (Figure 3.16) shows the command to remove these stop words from our text. In this screenshot, we also print the beginning of the first chapter to check if the deletion was successful. We can indeed see that these stop words are no longer present in the text.

```
> corpus_chapters_clean_low_stop <- tm::tm_map(corpus_chapters_clean_low,
+                                              tm::removeWords, my_stop_words)
> # We can see the cleaned chapter 1 for example
> tm::inspect(corpus_chapters_clean_low_stop[[1]])
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 9812


  arizona hills


    old man  old    know  possibly    hundred
possibly     tell    never aged
men    remember  childhood   far   can recollect
always   man  man   thirty   appear today
forty years   ago  yet  feel    go  living
forever   day  shall die  real death
 resurrection     know    fear death
died twice   still alive  yet     horror
   never died       terror  death
believe     convinced   mortality

    conviction   determined  write
story   interesting periods   life    death
explain  phenomena  can  set     words
ordinary soldier  fortune  chronicle   strange events
befell    ten years   dead body lay undiscovered
arizona cave
```

Figure 3.16: Stop words removing

However, the stop words from the tm library are too common, and to improve our results, we need to identify the most frequent words in each chapter to create a list of stop words specific to our text. The screenshot (Figure 3.17) illustrates how to do this. For each chapter, we first transform the document into a Term-Document Matrix, then use the findFreqTerms() function to find the most frequent terms (in our case, those which appear at least 4 times), and finally, we print these terms. The beginning of the output for the first two chapters is shown on Figure 3.18, and the entire list per chapter is presented on Table 3.19.

```r
for (i in 1:14) {
  tdm <- TermDocumentMatrix(corpus_chapters_clean_low_stop[[i]])
  cat("Chapter", i, "Top Frequent Terms:\n")
  freq_terms <- findFreqTerms(tdm, lowfreq = 4)
  print(freq_terms)
}
```

Figure 3.17: Finding the most frequent terms per chapter

```
> for (i in 1:14) {
+   tdm <- TermDocumentMatrix(corpus_chapters_clean_low_stop[[i]])
+   cat("Chapter", i, "Top Frequent Terms:\n")
+   freq_terms <- findFreqTerms(tdm, lowfreq = 4)
+   print(freq_terms)
+ }
Chapter 1 Top Frequent Terms:
 [1] "arizona"   "hundred"   "know"      "man"       "old"       "possibly"  "never"     "far"       "years"     "death"
[11] "convinced" "life"      "body"      "dead"      "cave"      "upon"      "way"       "powell"    "many"      "three"
[21] "two"       "horse"     "across"    "valley"    "morning"   "little"    "level"     "plateau"   "trail"     "knew"
[31] "soon"      "tracks"    "however"   "following" "red"       "now"       "suddenly"  "pass"      "fact"      "right"
[41] "face"      "opening"   "left"      "feet"
Chapter 2 Top Frequent Terms:
 [1] "dead"     "sound"    "first"    "time"     "cave"     "opening"  "yet"      "lay"      "short"    "upon"     "ledge"
[12] "thought"  "face"     "eyes"     "stood"    "behind"   "back"     "suddenly" "thing"    "left"     "body"     "seemed"
```

Figure 3.18: The beginning of the output of the code from Figure 3.17

Chapter number	The most frequent words
Chapter 1	arizona, hundred, know, man, old, possibly, never, far, years, death, convinced, life, body, dead, cave, upon, way, powell, many, three, two, horse, across, valley, morning, little, level, plateau, trail, knew, soon, tracks, however, following, red, now, suddenly, pass, fact, right, face, opening, left, feet
Chapter 2	dead, sound, first, time, cave, opening, yet, lay, short, upon, ledge, thought, face, eyes, stood, behind, back, suddenly, thing, left, body, seemed
Chapter 3	mars, eyes, upon, either, earth, seemed, low, hundred, little, enclosure, feet, four, first, martian, carried, without, extreme, must, easily, back, time, side, white, two, head, arms, learned, legs, weapons, later, behind, martians, spear, metal, held, mount, entirely, one, turned, less, evidently, toward
Chapter 4	ten, one, mars, martians, time, enormous, upon, toward, buildings, plaza, hundred, creatures, now, except, men, among, feet, age, martian, death, various, evidently, entrance, building, first, floor, chamber, chairs, desks, room, chieftain, name, tarkas, tars, made
Chapter 5	sola, room, martian, away, hand, animal, brute, mars, almost, one, without, upon, moon
Chapter 6	earthly, held, martians, one, creature, cudgel, feet, standing, arms, eyes, upon, ape, full, window, beast, great, life, floor, whose, blow, seemed, second, turned, sola, tarkas, tars

Chapter 7	mars, day, green, sola, community, entire, animals, chariots, hundred, two, one, female, martian, upon, young, warriors, martians, line, five, side, women, years, little, incubator, tarkas, tars, eggs, later, year, forth, another, almost, incubators
Chapter 8	toward, city, ground, return, though, green, buildings, warriors, building, upon, one, see, upper, craft, slowly, swung, another, strange, works, vessels, decks, martian, little, great, vessel, fire, guns, back, position, followed, seemed, figure, caught, first, entirely, sight, hope, life, just, eyes
Chapter 9	sola, upon, women, war, young, martians, make, time, men, one, first, toward, sarkoja, among
Chapter 10	attempt, leave, long, sola, city, come, like, martian, great, must, woola, nature, back, dead, hills, also, brute, man, upon, never, now, arms, among, full, eyes, may, feet, first, laughter, toward, moment, saw, one, though, mars, tarkas, tars, prisoner, well, audience, chamber, turned, act, know, human, green, woman, done, kindliness, lorquas, ptomel, warriors, men, attitude, dejah, thoris, battle, without, face, will, little, speak, warrior, blow, kill, chieftain, answered
Chapter 11	dejah, thoris, though, little, sola, sarkoja, upon, kill, men, found, quarters, carter, great, john, know, lorquas, ptomel, tarkas, tars, asked, may, new, building, ancient, many, people, much, beautiful, barsoom, tell, lost, eyes, believe, seemed, earth, planet, well, ages, conditions, martians, race
Chapter 12	prisoner, lorquas, ptomel, upon, yet, among, may, one, must, now, escape, either, hajus, tal, red, tarkas, tars, tharks, green, two, without, matter, also, cold, women, quarters, directed, floor, second, chieftains, sleeping, rooms, martians, community
Chapter 13	battle, community, days, even, far, green, tarkas, tars, great, warriors, thoats, time, well, though, upon, first, much, ever, always, every, martian, one, arm, warrior, know, since, moment, lorquas, ptomel, left, little, dejah, thoris, sola, plaza, earth, barsoom, powder, radium, can, night, dead
Chapter 14	love, thought, face, must, upon, return, even, feel, might, dejah, thoris, sola, always, carter, john, barsoom, people, word, low, lovers, finally, women, turned, held, silks, without, see, years, whose, green, little, martian, sought, one, sarkoja, saw, key, tarkas, tars, seemed, escape, will, much, something, sword, zad, chieftains, fact, two, fight, day, time, indeed, polish, hand, seen, great, just, stood, incubator, eggs, battle, struck, thrust

Table 3.19: The list of the most frequent terms per chapter

Next, by skimming these lists, we can identify some stop words like “one”, “two”, “upon”, “even” and so on. We create a list of strings containing these words and then we remove them from our text. The code to execute is shown below, on figure 3.20.

```
my_new_stop_words <- c("one", "two", "first", "second",
        "upon", "now", "yet", "possibly", "also",
        "even", "ever", "among", "much", "well", "just")

# Let's update to delete just found new stop words:
corpus_chapters_clean_low_stop <- tm::tm_map(corpus_chapters_clean_low_stop,
                                             tm::removeWords, my_new_stop_words)
```

Figure 3.20: The list of new stop words and the command to remove them

To see how this worked, we can run the script below (Figure 3.21), where we iterate through each chapter one more time and also try a new function to obtain term frequencies (termFreq()), and finally we print the 10 most frequent words per chapter. Here, we also store these frequency lists per chapter because we are going to use them in the future. On Figure 3.22, you can find the beginning of the output - frequency lists for the first 2 chapters, and on Table 3.23, you can see the table containing all these lists.

```
frequency_lists_per_chap <- list()
for (i in 1:14) {
  cat("Chapter", i, "Top 10 Frequent Terms:\n")
  freq_terms <- tm::termFreq(corpus_chapters_clean_low_stop[[i]])
  frequency_lists_per_chap[[i]] <- freq_terms
  top_terms <- head(sort(freq_terms, decreasing = TRUE), 10)
  print(top_terms)
  cat("\n")
}
```

Figure 3.21: New frequency lists per chapter

```
> frequency_lists_per_chap <- list()
> for (i in 1:14) {
+   cat("Chapter", i, "Top 10 Frequent Terms:\n")
+   freq_terms <- tm::termFreq(corpus_chapters_clean_low_stop[[i]])
+   frequency_lists_per_chap[[i]] <- freq_terms
+   top_terms <- head(sort(freq_terms, decreasing = TRUE), 10)
+   print(top_terms)
+   cat("\n")
+ }
Chapter 1 Top 10 Frequent Terms:
 powell   trail    knew    cave   death     man   horse   three arizona    body
     19      14       9       8       7       7       6       6       5       5

Chapter 2 Top 10 Frequent Terms:
  cave  sound    lay   back behind   dead  ledge  stood   body   eyes
    14      7      6      5      5      5      5      5      4      4
```

Figure 3.22: The beginning of the output of the code on Figure 3.21 (first two chapters)

Chapter number	Top 10 Most Frequent words and their number of occurence
Chapter 1	powell (19), trail (14), knew (9), cave (8), death (7), man (7), horse (6), three (6), arizona (5), body (5)
Chapter 2	cave (14), sound (7), lay (6), back (5), behind (5), dead (5), ledge (5), stood (5), body (4), eyes (4)
Chapter 3	feet (15), earth (11), little (11), mars (11), toward (10), low (8), spear (8), hundred (7), arms (6), enclosure (6)
Chapter 4	feet (9), martian (9), martians (8), tarkas (8), tars (8), plaza (7), creatures (6), hundred (6), mars (6), toward (6)
Chapter 5	mars (9), animal (5), away (5), martian (5), moon (5), room (5), sola (5), almost (4), brute (4), hand (4)
Chapter 6	ape (7), cudgel (7), martians (7), arms (5), eyes (5), floor (5), great (5), held (5), tarkas (5), tars (5)
Chapter 7	incubator (15), martians (11), little (10), martian (10), eggs (9), chariots (7), five (7), sola (7), young (7), warriors (6)
Chapter 8	warriors (14), craft (9), fire (8), green (8), ground (7), martian (7), see (7), seemed (7), swung (7), building (6)
Chapter 9	sola (13), men (6), toward (6), women (6), time (5), make (4), martians (4), sarkoja (4), war (4), young (4)

Chapter 10	martian (13), prisoner (11), woola (10), tarkas (9), tars (9), toward (9), will (9), back (8), know (8), lorquas (8)
Chapter 11	dejah (17), thoris (17), sola (11), people (10), men (8), barsoom (7), beautiful (7), believe (7), found (7), race (7)
Chapter 12	may (8), floor (7), women (7), hajus (6), red (6), tal (6), community (5), escape (5), prisoner (5), tarkas (5)
Chapter 13	dejah (10), thoris (10), great (9), martian (9), know (8), thoats (8), warriors (7), little (6), sola (6), tarkas (6)
Chapter 14	dejah (20), thoris (20), sarkoja (10), sola (10), love (9), might (9), time (9), eggs (8), little (8), will (8)

Table 3.23: Top 10 the most frequent words and their corresponding number of occurrence per chapter

Dendrograms

Furthermore, we can generate dendrograms for each chapter. A dendrogram is a tree-like diagram to illustrate the arrangement of the most frequent words based on their similarities. It is useful in text analysis because it visually groups words that are more similar to each other, and appear in similar context, which helps to understand the thematic structure of the text.

On Figure 3.24 below, you can find the code to execute to obtain a dendrogram for each chapter. Here, we are using our list of the most frequent words per chapter, selecting the top 20 terms to obtain a clear dendrogram structure. Then, we are transforming this list of terms into a data frame using the data.frame() function. Afterward, we calculate the distance between terms using the dist() function. Finally, we are clustering the terms using the hclust() function with the ward.D2 method to obtain a dendrogram. We are also printing the structure of the obtained object with the str() function and, of course, plotting it.

```r
# We are generating a dendrogram for each chapter's top 20 terms
for (i in 1:14) {
  cat("Dendrogram for Chapter", i, "\n")
  chapter_freq <- frequency_lists_per_chap[[i]]
  top_terms <- head(sort(chapter_freq, decreasing = TRUE), 20)
  freq_df <- data.frame(top_terms)
  dist_matrix <- dist(freq_df)
  hc <- hclust(dist_matrix, method = "ward.D2")
  str(hc)
  plot(hc, main = paste("Dendrogram of Top Terms in Chapter", i))
}
```

Figure 3.24: The code to execute to obtain a dendrogram for each chapter

On the next Figure (3.25), the structures of the first two dendrograms corresponding to the first two chapters are shown. Here we can see that each dendrogram is a list of 7 attributes: merge, height, order, labels, method, call, and dist.method.

- merge is a matrix of 19 columns and 2 rows in our case, where each row shows which two terms were merged at which step (from 1 to 19), if the number is positive then it represents the number of clusters and if it is negative, then it is a word's number. On our screenshot only one line is shown, where we can see which word was merged at which step, but we can see the full matrix by running hc$merge (example is shown on Figure 3.26). But we can see for example that for chapter 1 the word number 5 was merged first and then the 7th.
- height is the distance at which the clusters were merged. In our case, we can see the first values are all 0, which means that first terms were very close or even identical in the context of the frequency data. But by printing the whole list, by running hc$height we can see that final values are not null (example is shown on Figure 3.26).
- order shows the order in which the labels (words) appear in the plotted dendrogram.

- labels represent the actual words that are included in the dendrogram (in our case: the top 20 most frequent words for each chapter).
- method is the clustering method used - in this case, ward.D2.
- call stores the command used to generate the dendrogram with the hclust() function.
- dist.method is the distance function used - in our case, it's the default euclidean.

```
> # We are generating a dendrogram for each chapter's top 20 terms
> for (i in 1:14) {
+   cat("Dendrogram for Chapter", i, "\n")
+   chapter_freq <- frequency_lists_per_chap[[i]]
+   top_terms <- head(sort(chapter_freq, decreasing = TRUE), 20)
+   freq_df <- data.frame(top_terms)
+   dist_matrix <- dist(freq_df)
+   hc <- hclust(dist_matrix, method = "ward.D2")
+   str(hc)
+   plot(hc, main = paste("Dendrogram of Top Terms in Chapter", i))
+ }
Dendrogram for Chapter 1
List of 7
 $ merge      : int [1:19, 1:2] -5 -7 -9 -11 -12 -13 -14 -15 -16 -17 ...
 $ height     : num [1:19] 0 0 0 0 0 0 0 0 0 0 ...
 $ order      : int [1:20] 1 2 20 19 18 17 16 15 14 13 ...
 $ labels     : chr [1:20] "powell" "trail" "knew" "cave" ...
 $ method     : chr "ward.D2"
 $ call       : language hclust(d = dist_matrix, method = "ward.D2")
 $ dist.method: chr "euclidean"
 - attr(*, "class")= chr "hclust"
Dendrogram for Chapter 2
List of 7
 $ merge      : int [1:19, 1:2] -4 -6 -7 -8 -9 -11 -12 -13 -14 -15 ...
 $ height     : num [1:19] 0 0 0 0 0 0 0 0 0 0 ...
 $ order      : int [1:20] 1 20 19 18 17 16 15 14 13 12 ...
 $ labels     : chr [1:20] "cave" "sound" "lay" "back" ...
 $ method     : chr "ward.D2"
 $ call       : language hclust(d = dist_matrix, method = "ward.D2")
 $ dist.method: chr "euclidean"
 - attr(*, "class")= chr "hclust"
```

Figure 3.25: The structure of the first two dendrograms for the first two chapters

```
> chapter_freq <- frequency_lists_per_chap[[1]]
> top_terms <- head(sort(chapter_freq, decreasing = TRUE), 20)
> freq_df <- data.frame(top_terms)
> dist_matrix <- dist(freq_df)
> hc <- hclust(dist_matrix, method = "ward.D2")
> hc$merge
      [,1] [,2]
 [1,]   -5   -6
 [2,]   -7   -8
 [3,]   -9  -10
 [4,]  -11    3
 [5,]  -12    4
 [6,]  -13    5
 [7,]  -14    6
 [8,]  -15    7
 [9,]  -16    8
[10,]  -17    9
[11,]  -18   10
[12,]  -19   11
[13,]   -3   -4
[14,]  -20   12
[15,]    1    2
[16,]   13   15
[17,]   -1   -2
[18,]   14   16
[19,]   17   18
> hc$height
 [1]  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
[11]  0.000000  0.000000  1.000000  1.354006  1.414214  3.265986  5.000000  6.363961 20.554805
>
```

Figure 3.26: Examples of merge and height variables of dendrogram object on chapter 1

Furthermore, on the next 14 figures (Figures 3.27.1 – 3.27.14), you can see the dendrograms generated for the first 14 chapters of the book. Among the interesting observations we can highlight the following points: we know that if words appear on different branches at different heights, it means they occur in different contexts - and the greater the height value, the greater the contextual difference. In our case, we observe that in chapters 1, 3, 4, and especially 6 (even though there are three main branches, all words are at height 0), as well as chapters 11, 13, and 14, there are multiple branch separations, but nearly all the words are clustered at height 0. This suggests that the contextual differences between these words are probably minimal. However, in chapters like 2, 5, 8, and 9, we can clearly see separation of unique terms that are frequently used in those chapters but appear in very different contexts from other common words. Specifically,

these words are: cave(height ~12) in chapter 2; mars (height 6) in chapter 5; warriors (height ~10) in chapter 8; and sola (height 11) in chapter 9.

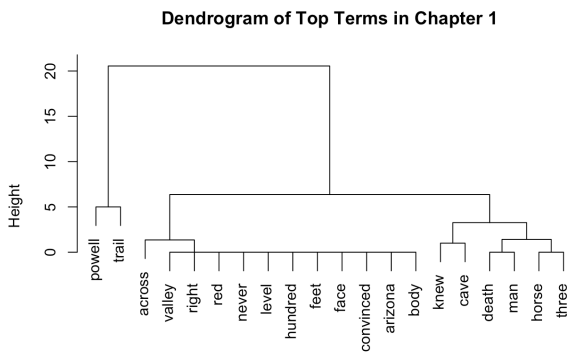


Figure 3.27.1: Dendrogram - Chapter 1

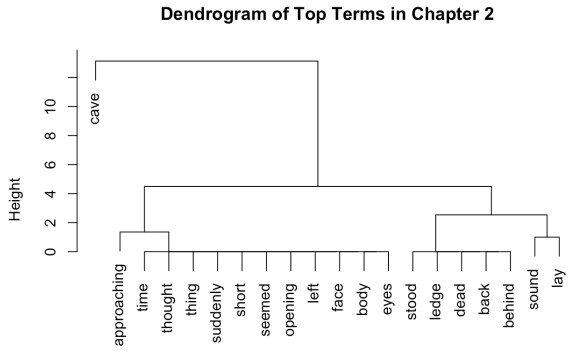


Figure 3.27.2: Dendrogram - Chapter 2

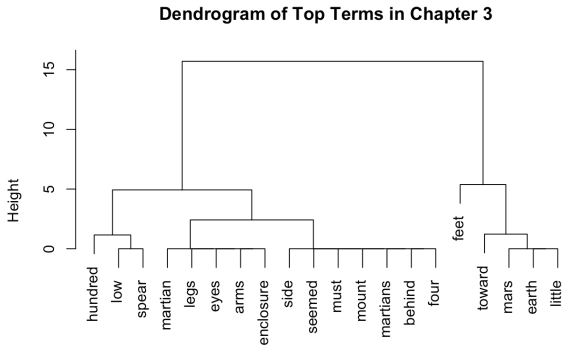


Figure 3.27.3: Dendrogram - Chapter 3

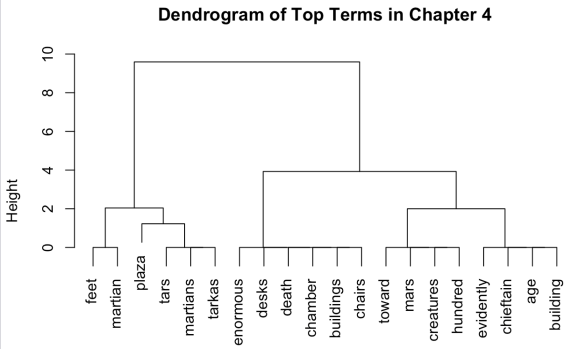


Figure 3.27.4: Dendrogram - Chapter 4

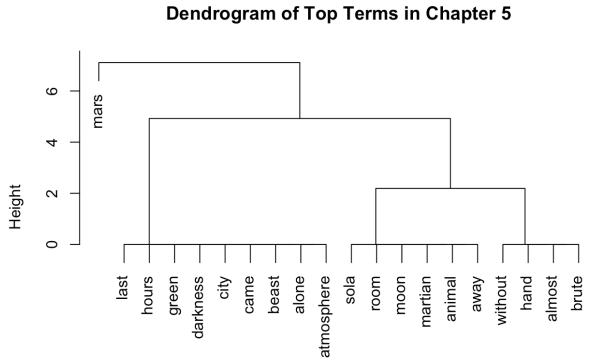


Figure 3.27.5: Dendrogram - Chapter 5

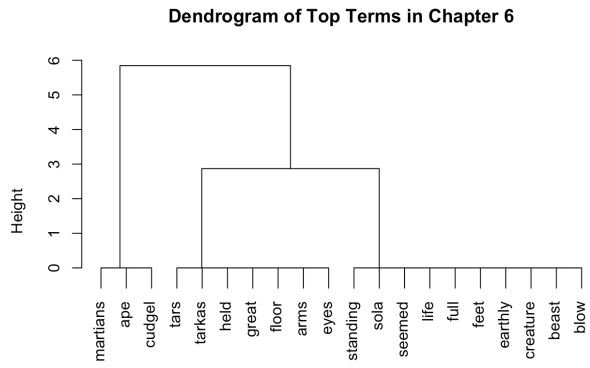


Figure 3.27.6: Dendrogram - Chapter 6

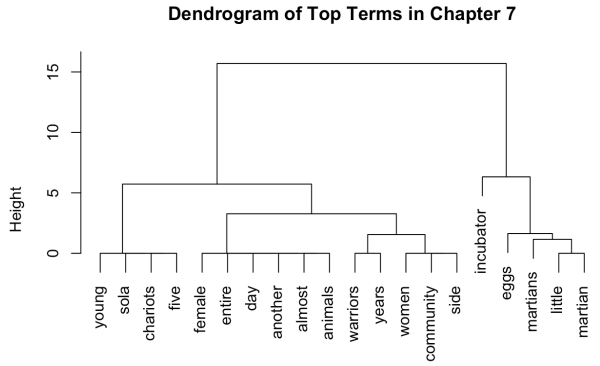


Figure 3.27.7: Dendrogram - Chapter 7

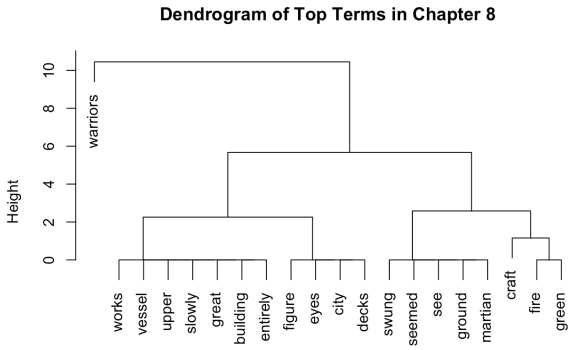


Figure 3.27.8: Dendrogram - Chapter 8

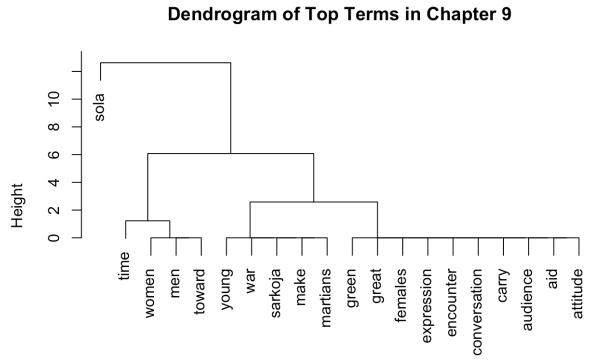


Figure 3.27.9: Dendrogram - Chapter 9

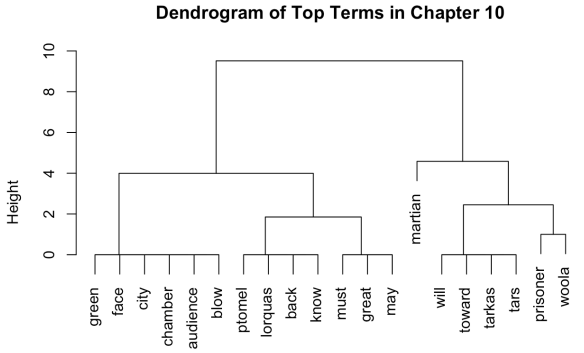


Figure 3.27.10: Dendrogram - Chapter 10

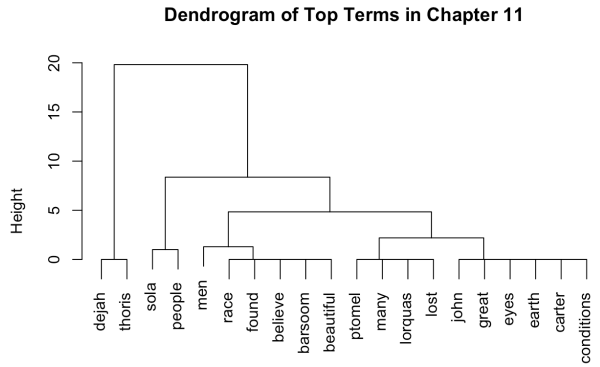


Figure 3.27.11: Dendrogram - Chapter 11

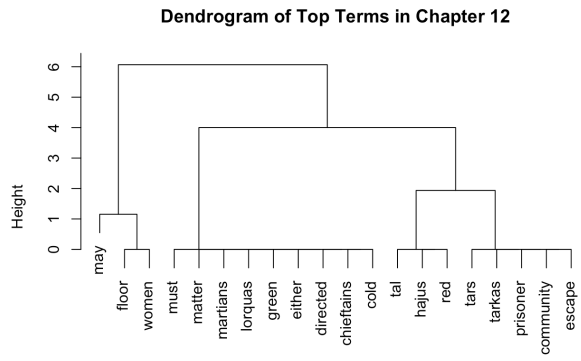


Figure 3.27.12: Dendrogram - Chapter 12

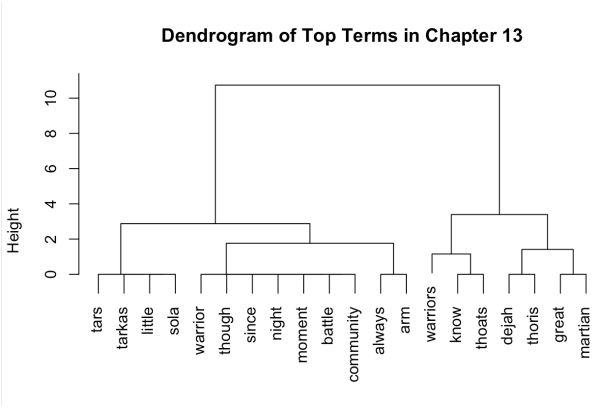


Figure 3.27.13: Dendrogram - Chapter 13

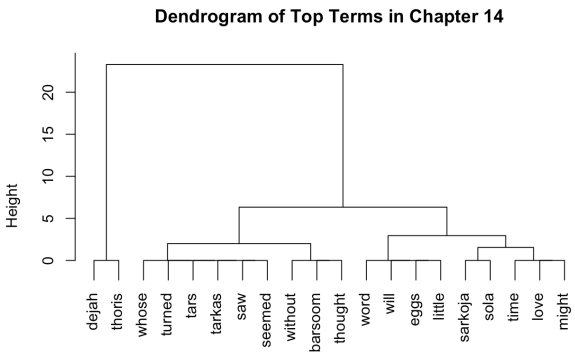


Figure 3.27.14: Dendrogram - Chapter 14

Word Clouds

Furthermore, let's move on in our analysis and construct word clouds for each chapter of our text. A word cloud is a visual representation of word frequency, where the size and the color saturation (in our case from the light green to dark green) of each word reflects how often it appears in the text. This is particularly useful in our analysis because it provides a quick overview of the major frequent terms of each chapter.

Using the same list of the most frequent words per chapter, we iterate through each chapter and select the top 35 most frequent words to ensure that the word cloud remains clear. Then, we apply the wordcloud() function to generate the visualizations. The code used to produce these word clouds is shown on Figure 3.28.

```r
# Word Cloud
pal <- brewer.pal(9, "BuGn")
for (i in 1:14) {
  freq_terms <- frequency_lists_per_chap[[i]]
  top_terms <- head(sort(freq_terms, decreasing = TRUE), 35)
  wordcloud(words = names(top_terms), freq = top_terms, colors = pal[-(1:4)],
            scale = c(2.5, 0.4))
  title(paste("Word Cloud for Chapter", i), line = -1)
}
```

Figure 3.28: Code to execute to get word clouds for each chapter

Furthermore, in Figures 3.29.1 to 3.29.14, you can see the word clouds generated for each chapter. Among the interesting details we can observe: in chapters like 1 (the words power and trail), especially 2 (cave), 5 (mars), 8 (warriors), and 9 (sola), we can see a clear distinction between one or two words that are in bold green and quite large in size, meaning they appear very frequently, and the other words represented, which are light green and small. This indicates a high distinction in word frequency in these chapters, with only one or two words appearing very often in contrast to the rest. On the other hand, in chapters 3, 4, especially 6 and 12 (where the majority of words are approximately the same color and size), as well as 10, word frequency is almost equally distributed among the 35 most frequent words. This contrast shows the differences in repetitions of words across chapters: we can clearly see the keywords of some chapters because they stand out from the rest of the words, while other chapters appear more variant.

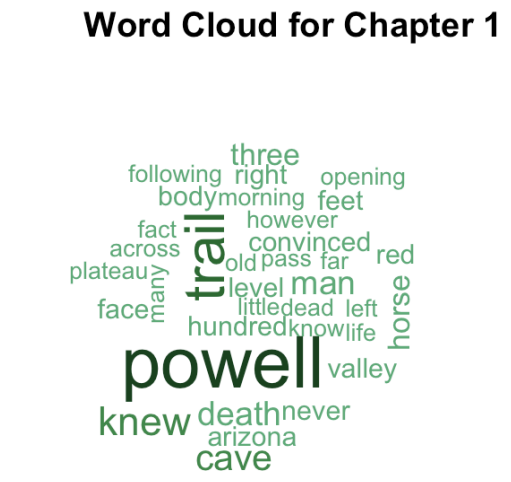


Figure 3.29.1: Word Cloud - Chapter 1

Figure 3.29.2: Word Cloud - Chapter 2

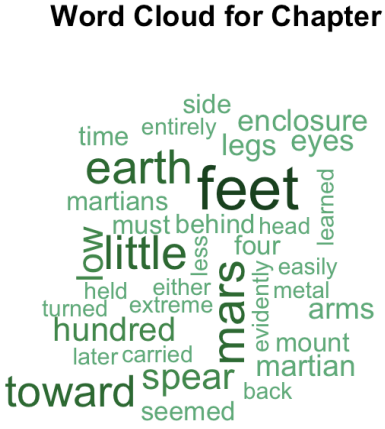


Figure 3.29.3: Word Cloud - Chapter 3

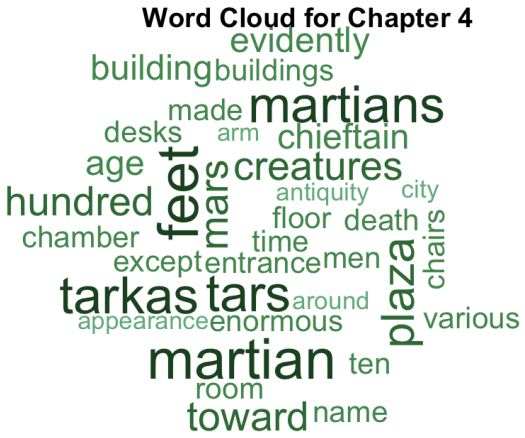


Figure 3.29.4: Word Cloud - Chapter 4

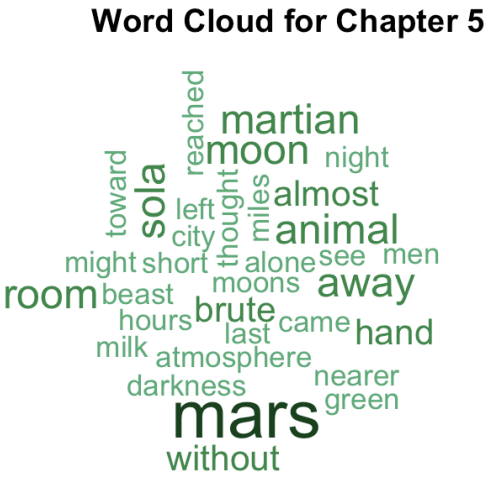


Figure 3.29.5: Word Cloud - Chapter 5

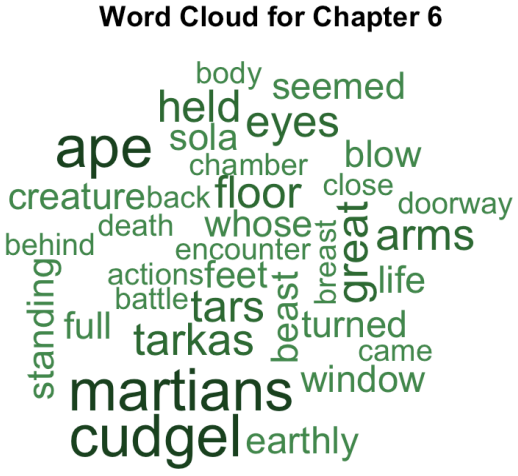


Figure 3.29.6: Word Cloud - Chapter 6

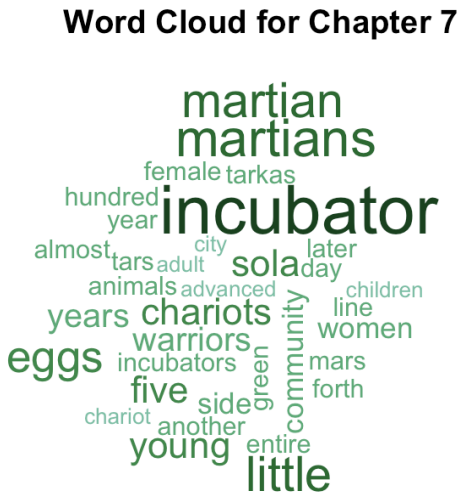


Figure 3.29.7: Word Cloud - Chapter 7

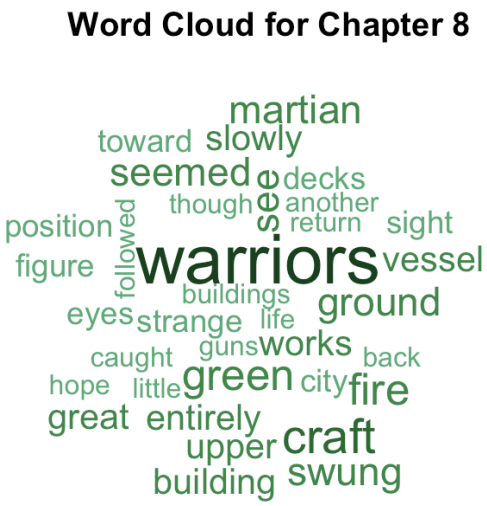


Figure 3.29.8: Word Cloud - Chapter 8

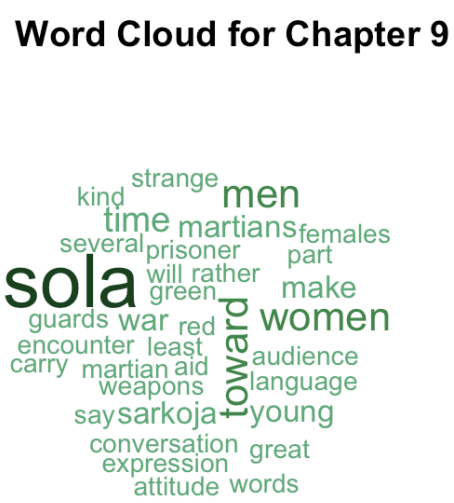


Figure 3.29.9: Word Cloud - Chapter 9

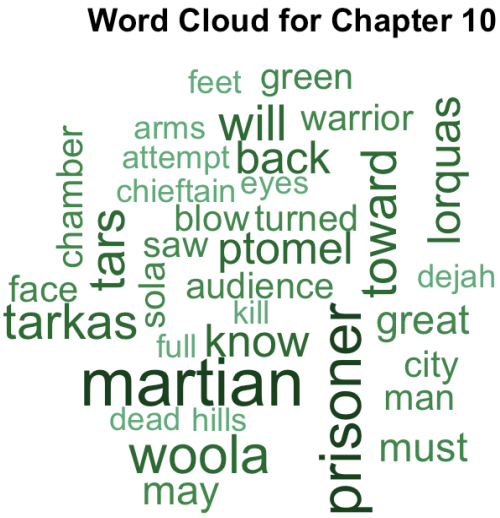


Figure 3.29.10: Word Cloud - Chapter 10

Figure 3.29.11: Word Cloud - Chapter 11

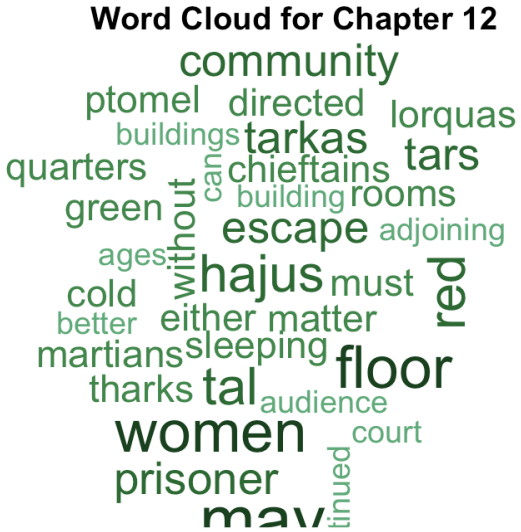


Figure 3.29.12: Word Cloud - Chapter 12

Figure 3.29.13: Word Cloud - Chapter 13

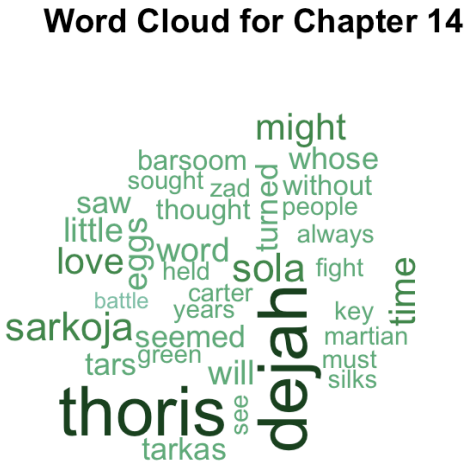


Figure 3.29.14: Word Cloud - Chapter 14

Analysis with tokenization

Next, we proceed to tokenization of the cleaned text. To do this, we first collapse all chapters (from 1 to 14) of the book into single character strings using sapply(), and then apply the tokens() function from the quanteda package. The code used for this step, as well as the str() function applied to examine the structure of the resulting object, are shown on Figure 3.30. This process breaks each chapter into a sequence of individual word tokens, stored in a list with one entry per chapter. We are going to use this in the future analysis. As shown in the output, each chapter now corresponds to a vector of words. We can also observe the number of words per chapter - for example, the 9th chapter contains the fewest tokens (616), while the 10th chapter contains the most (1570).

```
> chapters_char <- sapply(corpus_chapters_clean_low_stop, paste, collapse = " ")
> corpus_chapters_tokens <- tokens(chapters_char)
> str(corpus_chapters_tokens)
List of 14
 $ chapter_01.txt: chr [1:1163] "arizona" "hills" "old" "man" ...
 $ chapter_02.txt: chr [1:740] "escape" "dead" "sense" "delicious" ...
 $ chapter_03.txt: chr [1:1162] "advent" "mars" "opened" "eyes" ...
 $ chapter_04.txt: chr [1:982] "prisoner" "gone" "perhaps" "ten" ...
 $ chapter_05.txt: chr [1:716] "elude" "watch" "dog" "sola" ...
 $ chapter_06.txt: chr [1:738] "fight" "won" "friends" "thing" ...
 $ chapter_07.txt: chr [1:932] "childraising" "mars" "breakfast" "exact" ...
 $ chapter_08.txt: chr [1:958] "fair" "captive" "sky" "third" ...
 $ chapter_09.txt: chr [1:616] "learn" "language" "came" "back" ...
 $ chapter_10.txt: chr [1:1570] "champion" "chief" "early" "next" ...
 $ chapter_11.txt: chr [1:1061] "dejah" "thoris" "reached" "open" ...
 $ chapter_12.txt: chr [1:840] "prisoner" "power" "entered" "saluted" ...
 $ chapter_13.txt: chr [1:1017] "lovemaking" "mars" "following" "battle" ...
 $ chapter_14.txt: chr [1:1427] "duel" "death" "impulse" "tell" ...
 - attr(*, "class")= chr "tokens"
 - attr(*, "types")= chr [1:4377] "arizona" "hills" "old" "man" ...
 - attr(*, "padding")= logi TRUE
 - attr(*, "docvars")='data.frame':     14 obs. of  3 variables:
  ..$ docname_: chr [1:14] "chapter_01.txt" "chapter_02.txt" "chapter_03.txt" "chapter_04.txt" ...
  ..$ docid_  : Factor w/ 14 levels "chapter_01.txt",..: 1 2 3 4 5 6 7 8 9 10 ...
  ..$ segid_  : int [1:14] 1 1 1 1 1 1 1 1 1 1 ...
 - attr(*, "meta")=List of 3
  ..$ system:List of 5
  .. ..$ package-version:Classes 'package_version', 'numeric_version'  hidden list of 1
  .. .. ..$ : int [1:3] 4 2 0
  .. ..$ r-version      :Classes 'R_system_version', 'package_version', 'numeric_version'  hidden list of 1
  .. .. ..$ : int [1:3] 4 3 2
  .. ..$ system         : Named chr [1:3] "Darwin" "arm64" "ekaterinagalkina"
  .. .. ..- attr(*, "names")= chr [1:3] "sysname" "machine" "user"
  .. ..$ directory      : chr "/Users/ekaterinagalkina"
  .. ..$ created        : Date[1:1], format: "2025-04-22"
  ..$ object:List of 7
  .. ..$ unit        : chr "documents"
  .. ..$ what        : chr "word"
  .. ..$ tokenizer   : chr "tokenize_word4"
  .. ..$ ngram       : int 1
  .. ..$ skip        : int 0
  .. ..$ concatenator: chr "_"
  .. ..$ summary     :List of 2
  .. .. ..$ hash: chr(0)
  .. .. ..$ data: NULL
  ..$ user  : list()
```

Figure 3.30: Tokenization of the corpus

Subsequently, let's produce a Document-Feature Matrix (DFM) using the tokens we have just obtained. A DFM is essentially the same as a Document-Term Matrix (just from a different package). Each row corresponds to a document (a chapter), and each column to a unique word, with the cell values indicating the frequency of that word in the document. This object serves as

the starting point for our analysis using the tools of the quanteda package. The code to obtain a Document-Feature Matrix from our tokens is shown on Figure 3.31.

```
corpus_chapters_dfm <- quanteda::dfm(corpus_chapters_tokens)
```

Figure 3.31: The command to produce a Document-Feature Matrix

We can view word frequencies using the docfreq() function applied to the previously obtained Document-Feature Matrix. Figure 3.32 shows the command to execute, the output of the str() function to inspect the structure of the resulting object, and the first 20 lines of this object, each displaying a word with its frequency across all chapters. From the str() output, we learn that the object contains 4377 terms and shows the number of occurrences for each one. And by printing the first part of the object itself we can see for example, among the first 20 terms, the word "men" appears 12 times, while "aged" appears only once.

```
> corpus_chapters_docfreq <- quanteda::docfreq(corpus_chapters_dfm) # frequency of terms in dfm
> str(corpus_chapters_docfreq)
 Named int [1:4377] 3 6 5 11 11 8 5 9 1 12 ...
 - attr(*, "names")= chr [1:4377] "arizona" "hills" "old" "man" ...
> corpus_chapters_docfreq[1:20]
  arizona     hills       old       man      know   hundred      tell     never      aged       men  remember childhood       far
        3         6         5        11        11         8         5         9         1        12         3         2        13
      can recollect    always    thirty    appear     today     forty
       10         1         9         6         2         5         4
>
```

Figure 3.32: Frequency of the terms with Document-Feature Matrix

Next, we apply the dfm_weight() function from the quanteda package to our previously obtained Document-Feature Matrix in order to compute the term frequency weights. The resulting object is a weighted version of our DFM, where each cell represents how much weight a particular word obtained in a given chapter by the application of the dfm_weight function(). Figure 3.33 shows the command used, and the printed output of the first few features (words) across all 14 chapters. As shown, for example, the word "arizona" gets a weight of 5 in chapter 1, 2 in chapter

2, and 1 in chapter 3, but then 0 in all the remaining chapters. Overall, we can see in general information that the matrix is sparse, with 84.51% of entries being equal to zero, meaning most words do not appear in most chapters (we already saw this at the beginning).

```
> corpus_chapters_weights <- quanteda::dfm_weight(corpus_chapters_dfm)
> print(corpus_chapters_weights, max_ndoc = 14)
Document-feature matrix of: 14 documents, 4,377 features (84.51% sparse) and 0 docvars.
                features
docs             arizona hills old man know hundred tell never aged men
  chapter_01.txt       5     1   4   7    4       5    1     5    1   3
  chapter_02.txt       2     0   0   2    1       0    0     0    0   0
  chapter_03.txt       1     2   0   3    0       7    0     0    0   1
  chapter_04.txt       0     0   1   1    2       6    0     0    0   4
  chapter_05.txt       0     0   0   1    0       0    0     1    0   3
  chapter_06.txt       0     0   0   1    1       0    0     0    0   2
  chapter_07.txt       0     1   0   0    2       4    0     1    0   2
  chapter_08.txt       0     2   1   0    0       1    0     2    0   0
  chapter_09.txt       0     0   0   0    2       0    0     2    0   6
  chapter_10.txt       0     5   2   6    8       1    2     5    0   5
  chapter_11.txt       0     2   1   3    5       1    4     2    0   8
  chapter_12.txt       0     0   0   1    1       0    0     0    0   3
  chapter_13.txt       0     0   0   2    8       0    1     1    0   1
  chapter_14.txt       0     0   0   3    3       1    1     3    0   1
[ reached max_nfeat ... 4,367 more features ]
>
```

Figure 3.33: Assigning weights to each term per chapter

Now, we apply the dfm_tfidf() function from the quanteda package to calculate the TF-IDF scores for each word. The TF-IDF (Term Frequency - Inverse Document Frequency) statistic measures how important a word is to a document within a corpus. On Figure 3.34, you can see the command used to generate this matrix of scores (one for each word per document) based on a previously created Document-Feature Matrix. The structure of the resulting TF-IDF matrix is also shown using the str() function.

This structure includes several variables:

- docvars: stores metadata for each document (name, id, segid).

- meta: holds some metadata about the matrix.
- dim: indicates the size of the matrix, in our case: 14 chapters (rows) and 4378 unique terms (columns).
- dimnames: contains the names of the rows (documents) and columns (terms).
- i, x, and p: these three variables contain the main matrix information:
    - i (length 9494): stores the row indices of non-zero values, meaning which document (chapter) each word appears in.
    - x (also length 9494): holds the TF-IDF scores corresponding to each entry in i.
    - p (length 4378): contains the start index in i and x for each term (i.e., column in the matrix). Since terms are sorted, p lets us locate the exact range in i and x that corresponds to each word.

```
> corpus_chapters_tfidf <- quanteda::dfm_tfidf(corpus_chapters_dfm,
+                          scheme_tf = "count", scheme_df = "inverse")
> str(corpus_chapters_tfidf)
Formal class 'dfm' [package "quanteda"] with 8 slots
  ..@ docvars :'data.frame':    14 obs. of  3 variables:
  .. ..$ docname_: chr [1:14] "chapter_01.txt" "chapter_02.txt" "chapter_03.txt" "chapter_04.txt" ...
  .. ..$ docid_  : Factor w/ 14 levels "chapter_01.txt",..: 1 2 3 4 5 6 7 8 9 10 ...
  .. ..$ segid_  : int [1:14] 1 1 1 1 1 1 1 1 1 1 ...
  ..@ meta    :List of 3
  .. ..$ system:List of 5
  .. .. ..$ package-version:Classes 'package_version', 'numeric_version'  hidden list of 1
  .. .. .. ..$ : int [1:3] 4 2 0
  .. .. ..$ r-version      :Classes 'R_system_version', 'package_version', 'numeric_version'  hidden list of 1
  .. .. .. ..$ : int [1:3] 4 3 2
  .. .. ..$ system         : Named chr [1:3] "Darwin" "arm64" "ekaterinagalkina"
  .. .. .. ..- attr(*, "names")= chr [1:3] "sysname" "machine" "user"
  .. .. ..$ directory      : chr "/Users/ekaterinagalkina"
  .. .. ..$ created        : Date[1:1], format: "2025-04-23"
  .. ..$ object:List of 10
  .. .. ..$ unit        : chr "documents"
  .. .. ..$ what        : chr "word"
  .. .. ..$ tokenizer   : chr "tokenize_word4"
  .. .. ..$ ngram       : int 1
  .. .. ..$ skip        : int 0
  .. .. ..$ concatenator: chr "_"
  .. .. ..$ weight_tf   :List of 3
  .. .. .. ..$ scheme: chr "count"
  .. .. .. ..$ base  : NULL
  .. .. .. ..$ k     : NULL
  .. .. ..$ weight_df   :List of 2
  .. .. .. ..$ scheme: chr "inverse"
  .. .. .. ..$ base  : num 10
  .. .. ..$ smooth      : num 0
  .. .. ..$ summary     :List of 2
  .. .. .. ..$ hash: chr(0)
  .. .. .. ..$ data: NULL
  .. ..$ user  : list()
  ..@ i       : int [1:9494] 0 1 2 0 2 6 7 9 10 0 ...
  ..@ p       : int [1:4378] 0 3 9 14 25 36 44 49 58 59 ...
  ..@ Dim     : int [1:2] 14 4377
  ..@ Dimnames:List of 2
  .. ..$ docs    : chr [1:14] "chapter_01.txt" "chapter_02.txt" "chapter_03.txt" "chapter_04.txt" ...
  .. ..$ features: chr [1:4377] "arizona" "hills" "old" "man" ...
  ..@ x       : Named num [1:9494] 3.345 1.338 0.669 0.368 0.736 ...
  .. ..- attr(*, "names")= chr [1:9494] "arizona" "arizona" "arizona" "hills" ...
  ..@ factors : list()
```

Figure 3.34: Calculating TFI - DF and printing the structure of the received object

Furthermore, Figure 3.35 displays the beginning of the resulting TF-IDF matrix. In this output, we are printing the first 6 documents (chapters) and the first 10 terms (words) to see the matrix's content.

For example, we can observe that the word "arizona" has a relatively high TF-IDF score at the beginning of the book - a score of 3.34 in the first chapter, then 1.33 in the second, and

decreasing afterward. In contrast, the other words generally have lower scores, most of them below 1, which indicates that “arizona” is important in the early chapters.

However, it is important to note that this is just a small part of the whole matrix - only 10 terms out of 4377 and 6 chapters out of 14 are shown.

```
> corpus_chapters_tfidf
Document-feature matrix of: 14 documents, 4,377 features (84.51% sparse) and 0 docvars.
                features
docs               arizona     hills      old       man      know  hundred     tell     never     aged
  chapter_01.txt 3.3450339 0.3679768 1.788632 0.7331475 0.4189414 1.215190 0.447158 0.9594276 1.146128
  chapter_02.txt 1.3380136 0         0        0.2094707 0.1047354 0        0        0         0
  chapter_03.txt 0.6690068 0.7359536 0        0.3142061 0         1.701266 0        0         0
  chapter_04.txt 0         0         0.447158 0.1047354 0.2094707 1.458228 0        0         0
  chapter_05.txt 0         0         0        0.1047354 0         0        0        0.1918855 0
  chapter_06.txt 0         0         0        0.1047354 0.1047354 0        0        0         0
                features
docs                    men
  chapter_01.txt 0.20084037
  chapter_02.txt 0
  chapter_03.txt 0.06694679
  chapter_04.txt 0.26778716
  chapter_05.txt 0.20084037
  chapter_06.txt 0.13389358
[ reached max_ndoc ... 8 more documents, reached max_nfeat ... 4,367 more features ]
>
```

Figure 3.35: Printing the beginning of the object returned by the dfm_tfidf() function

Sentimental analysis

Let’s move on to the sentiment analysis of the phrases in each chapter to examine the emotional progression throughout the part of the book we are studying. For this, we are using the syuzhet library, which offers two main methods for evaluating the sentiment (how negative or positive it is) of a phrase based on the words it contains. These two methods are: syuzhet (the default) and Bing. Each method relies on its own dictionary of words, with associated sentiment scores. Figure 3.36 shows the code used to display the first 15 words and their corresponding sentiment scores for both methods. On Figures 3.37.1 and 3.37.2, you can see the output of this code: the

first figure shows the beginning of the syuzhet dictionary, and the second one shows the Bing dictionary.

```
# First we can see the dictionary of different methods
sentiment_dictionary <- get_sentiment_dictionary()
sentiment_dictionary[1:15,,]

# Dictionary of another method (Bing)
sentiment_dictionary_bing <- get_sentiment_dictionary("bing")
sentiment_dictionary_bing[1:15,,]
```

Figure 3.36: Printing the beginning of the object returned by the dfm_tfidf() function

```
> # First we can see the dictionary of diff
> sentiment_dictionary <- get_sentiment_dic
> sentiment_dictionary[1:15,,]
          word value
1      abandon -0.75
2    abandoned -0.50
3    abandoner -0.25
4  abandonment -0.25
5     abandons -1.00
6     abducted -1.00
7    abduction -0.50
8   abductions -1.00
9     aberrant -0.60
10  aberration -0.80
11       abhor -0.50
12    abhorred -1.00
13   abhorrent -0.50
14      abhors -1.00
15   abilities  0.60
```

Figure 3.37.1: Sentimental dictionary : Syuzhet method

```
> # Dictionary of another method (Bing)
> sentiment_dictionary_bing <- get_senti
> sentiment_dictionary_bing[1:15,,]
            word value
1             a+     1
2         abound     1
3        abounds     1
4      abundance     1
5       abundant     1
6     accessable     1
7     accessible     1
8        acclaim     1
9      acclaimed     1
10   acclamation     1
11      accolade     1
12     accolades     1
13 accommodative     1
14  accomodative     1
15    accomplish     1
```

Figure 3.37.2: Sentimental dictionary: Bing method

Furthermore, we are going to obtain different sentiment metrics for each chapter. On Figures 3.38.1, 3.38.2, and 3.38.3, the main loop that iterates through each chapter is depicted. Each function is placed in a separate part of the code, and between each section, the program pauses using the readline() function to avoid overwhelming the user with too much printed and plotted

data at once. This readline() function is also used between chapters, so after each loop iteration, the user must press enter to proceed to the next one.

In the first part (Figure 3.38.1), you can see the different metrics obtained using the syuzhet method. Here, we begin by preparing the current chapter: for each chapter, we read the entire text from the corresponding file using the get_text_as_string() function. Then, we extract the sentences and store them in a list of strings using the get_sentences() function.

Next, we start our analysis by calling the get_sentiment() function, which returns a sentiment score for each sentence. If the score is greater than 0, the sentence is considered positive; otherwise, it is negative. We then print the sentiment score for each sentence.

We also calculate the overall emotional valence of the text by summing (using the sum() function) all the previously obtained sentiment values for each sentence and printing this result.

Next, we compute the average sentiment using the mean() function and print this as well. We then run the summary() function to see the distribution of sentiment across the chapter.

While these overall sentiment measures are informative, they reveal little about how the narrative is structured and how positive and negative sentiments are distributed throughout the text.

Therefore, it is useful to visualize these values in a graph, where the x-axis represents the passage of time from the beginning to the end of the text, and the y-axis represents the degree of positive or negative sentiment.

For this reason, we also plot the sentiment values per sentence using the plot() function. To get a clearer picture of how sentiment evolves throughout the chapter, we divide the text into 10 equal parts and compute sentiment values for each part using the get_percentage_values() function. These are then plotted using the same plot() function.

```
for (i in 1:14) {
  chapter_path <- file.path(path, "Chapters", sprintf("chapter_%02d.txt", i))
  chapter_text <- get_text_as_string(chapter_path)
  chapter_sentences <- get_sentences(chapter_text)

  cat("Chapter", i, "information:\n")

  # syuzhet method
  chapter_sentences_sentiment <- get_sentiment(chapter_sentences, method = "syuzhet")
  cat("\nSentiment for each sentence obtained with syuzhet method:\n")
  print(chapter_sentences_sentiment)

  chapter_sentiment_sum <- sum(chapter_sentences_sentiment)
  cat("\nThe overall emotional valence in the text with syuzhet method:\n")
  print(chapter_sentiment_sum)

  chapter_sentiment_mean <- mean(chapter_sentences_sentiment)
  cat("\nThe mean of sentiment in the text with syuzhet method:\n")
  print(chapter_sentiment_mean)

  cat("\nThe distribution with syuzhet method:\n")
  print(summary(chapter_sentences_sentiment))

  plot(chapter_sentences_sentiment, main = "Book Plot Trajectory: syuzhet",
       xlab = "Narrative", ylab = "Emotional Valence")

  sentiment_pct_syuzhet <- get_percentage_values(chapter_sentences_sentiment, bins = 10)
  cat("\nPercentage values with syuzhet method:\n")
  print(sentiment_pct_syuzhet)

  plot(sentiment_pct_syuzhet, main = "Book Plot PCTValue 10 Bins (syuzhet)",
       xlab = "Narrative", ylab = "Emotional Valence")

  readline(prompt = "\nPress [Enter] to see with Bing method")
```

Figure 3.38.1: Sentimental analysis : Syuzhet method

In the next screenshot (Figure 3.38.2), you can see the second part of the program, which calculates the exact same metrics as before, but using a different method: Bing, to observe how the sentiment values differ from those calculated with the syuzhet method.

Finally, in the last screenshot of this program (Figure 3.38.3), the final part is shown: the NRC method. This method differs from the previous two because we are not just assigning a positive or negative score to each sentence, but we are attempting to identify the specific emotion expressed. These emotions are: anger, anticipation, disgust, fear, joy, sadness, surprise, trust, negative, and positive.

This time, we use the get_nrc_sentiment() function, which provides, for each sentence in the chapter, a score for each of the above emotions ranging from 0 to 5. Then, as the final step in the analysis of the current chapter, we calculate the top 3 dominant emotions (excluding negative and positive) by summing the total score for each emotion across all sentences. We then print these top emotions along with their corresponding scores.

```
# bing method
chapter_sentences_sentiment_bing <- get_sentiment(chapter_sentences, method = "bing")
cat("\nSentiment for each sentence obtained with bing method:\n")
print(chapter_sentences_sentiment_bing)

chapter_sentiment_bing_sum <- sum(chapter_sentences_sentiment_bing)
cat("\nThe overall emotional valence in the text with bing method:\n")
print(chapter_sentiment_bing_sum)

chapter_sentiment_bing_mean <- mean(chapter_sentences_sentiment_bing)
cat("\nThe mean of sentiment in the text with bing method:\n")
print(chapter_sentiment_bing_mean)

cat("\nThe distribution with bing method:\n")
print(summary(chapter_sentences_sentiment_bing))

plot(chapter_sentences_sentiment_bing, main = "Book Plot Trajectory: bing",
     xlab = "Narrative", ylab = "Emotional Valence")

sentiment_pct_bing <- get_percentage_values(chapter_sentences_sentiment_bing, bins = 10)
cat("\nPercentage values with bing method:\n")
print(sentiment_pct_bing)

plot(sentiment_pct_bing, main = "Book Plot PCTValue 10 Bins (bing)",
     xlab = "Narrative", ylab = "Emotional Valence")

readline(prompt = "\nPress [Enter] to see with NRC method")
```

Figure 3.38.2: Sentimental analysis : Bing method

```
  # nrc method
  chapter_nrc_sentiment <- get_nrc_sentiment(chapter_sentences)
  cat("\nSentiment for each sentence obtained with nrc method:\n")
  print(chapter_nrc_sentiment)

  # We are calculating top 3 emotions for this chapter (without positive or negative)
  emotion_sums <- colSums(chapter_nrc_sentiment[, 1:8])
  sorted_emotions <- sort(emotion_sums, decreasing = TRUE)
  top_3_emotions <- head(sorted_emotions, 3)
  cat("Top 3 emotions for this chapter:\n")
  for (i in 1:length(top_3_emotions)) {
    emotion_name <- names(top_3_emotions)[i]
    emotion_score <- top_3_emotions[i]
    cat(emotion_name, ":", emotion_score, "; ")
  }

  if (i != 14){
    readline(prompt = "\nPress [Enter] to continue to the next chapter's sentimental analysis")
  }
}
```

Figure 3.38.3: Sentimental analysis : NRC method

Furthermore, Figures 3.39.1 to 3.39.6, as well as Figures 3.39.7.1 and 3.39.7.2, show the output of the first iteration, meaning all the metrics described above for the first chapter.

Then, you can find all the main information about the 14 chapters in Tables 3.40.1, 3.40.2, and 3.40.3. Namely, the first table stores the main metrics for all three methods: the valence and mean values for the syuzhet and Bing methods, and the top three emotions based on the NRC method. In the second table, you can find the main statistics for the syuzhet and Bing methods: minimum, first quartile, median, mean, third quartile, and maximum. Finally, the last table stores the percentage value plots for the syuzhet and Bing methods per chapter.

Among the interesting findings, we can note the following: from the Sentiment Analysis Summary (Table 3.40.1), we can glean that the mean values of both the Syuzhet and Bing methods remain close to 0, indicating that overall, there is an approximately equal number of positive and negative sentences.

Regarding sentiment valence, we observe high variability with the Syuzhet method - for example, between chapters 5 and 8: in chapter 5, the value is 11.65, then drops to -18.55 in chapter 6, rises sharply to 25.9 in chapter 7, and falls again to -22 in chapter 8. As for the Bing method, it produces generally low negative values (-60 in chapter 2, -33 in chapter 10), while the positive values remain modest (the highest being 12 in chapter 13, with others not exceeding 5). According to the NRC method, the three most common emotions across the chapters are trust, fear, and anticipation. These emotions appear among the top three in almost every chapter. The highest overall values for these emotions are in chapter 10, where trust reaches 114, fear - 99, and anticipation - 88 - therefore, this chapter is likely to convey emotions very clearly.

From the Sentiment Distribution Summary (Table 3.40.2), we can observe that the Syuzhet method generally produces higher maximum values for positive sentiment, while Bing method: lower minimum values for negative sentiment. This suggests that Syuzhet tends to identify more positive sentiment than Bing, which, by contrast, focuses on negative sentiment more.

Finally, in Table 3.40.3, which shows the Plot of Percentage Values for Syuzhet and Bing methods, we see that under Syuzhet, chapter 2 appears to be the most “vivid” chapter - its sentiment scores range from -2.0 to +1.5 (a spread of 3.5), while most other chapters show a range of only about 1.5 to 2. Chapter 8 is the most negative, as its points fall entirely within the -1.5 to 0 range, while other chapters appear more evenly distributed.

With the Bing method, chapters 6 and 2 are the most vivid, showing a range from just below -3 to +1 (a spread of around 4). In chapter 2, there is a smooth transition between negative and positive sentiment, with no drastic changes between points - unlike in other chapters, where sharp jumps between highly negative and highly positive values occur. This suggests that chapter

2 may feature a more gradual emotional shift, while other chapters experience more abrupt transitions in sentiment.

```
+   if (i != 14){
+     readline(prompt = "\nPress [Enter] to continue to the next chapter's sentimental analysis")
+   }
+ }
Chapter 1 information:

Sentiment for each sentence obtained with syuzhet method:
 [1]  0.00  0.60  0.00 -1.00 -3.40  0.00 -0.90 -0.25 -0.30  3.20  0.80 -0.10 -1.05 -0.65  0.95  0.65  2.00  2.60  0.50  0.80  1.40
[22]  0.00  0.80  0.50 -4.95  1.65  0.65  1.00  0.00 -0.75 -0.25  2.85  1.50 -0.85  0.70 -1.50 -0.50 -0.75  0.80 -1.50  1.05  1.20
[43]  1.80  0.05 -1.00 -1.25  0.25  0.70 -2.00  0.25 -1.35 -0.65 -1.25 -1.25 -0.25  1.30  1.00 -0.35  0.20 -0.60  0.30  1.00  1.20
[64]  0.80  0.80  0.00 -1.00 -0.40 -0.45  3.70 -0.60  1.20 -0.75 -0.95  1.25  0.05 -0.85

The overall emotional valence in the text with syuzhet method:
[1] 8.4

The mean of sentiment in the text with syuzhet method:
[1] 0.1090909

The distribution with syuzhet method:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-4.9500 -0.7500  0.0000  0.1091  0.8000  3.7000

Percentage values with syuzhet method:
         1          2          3          4          5          6          7          8          9         10
-0.6187500  0.4375000  1.1571429 -0.2687500  0.2812500  0.0500000 -0.6625000  0.2285714  0.2437500  0.3812500

Press [Enter] to see with Bing method
```

Figure 3.39.1: Sentimental analysis: Syuzhet method: Results of execution

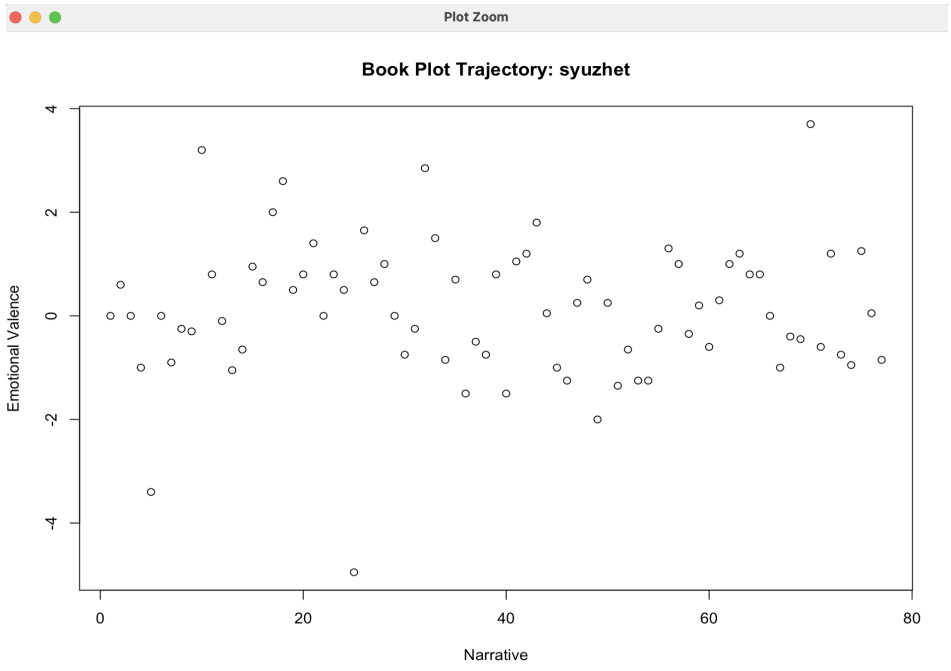


Figure 3.39.2: Sentimental analysis: Syuzhet method: Plot Trajectory

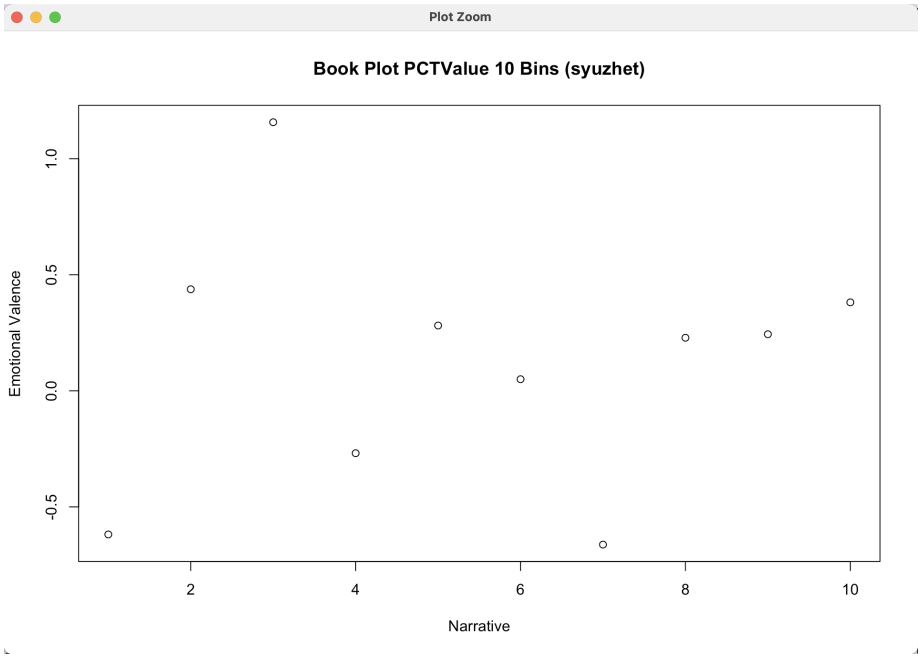


Figure 3.39.3: Sentimental analysis: Syuzhet method: Plot percentage values

```
Press [Enter] to see with Bing method

Sentiment for each sentence obtained with bing method:
 [1]  0  0  0 -2 -4  0 -2  0 -1  1  1  0  1  0  2  1  2  2  0  2  3  0  0 -1 -8  1  0  0  0  0 -1  2  3 -1  0 -2 -1  0 -1 -2 -1  0
[43]  1 -1 -1 -1  2  1 -2  1 -2  0 -3  0 -3  0  1 -2  0 -1  1  1  1  0  1  0 -2 -1  0  2 -2  0 -3 -1  0 -2 -1

The overall emotional valence in the text with bing method:
[1] -22

The mean of sentiment in the text with bing method:
[1] -0.2857143

The distribution with bing method:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-8.0000 -1.0000  0.0000 -0.2857  1.0000  3.0000

Percentage values with bing method:
         1          2         3          4         5          6          7         8         9         10
-1.0000000  0.6250000  1.2857143 -1.1250000  0.0000000 -0.7142857 -0.3750000 -0.5714286  0.0000000 -0.8750000

Press [Enter] to see with NRC method
```

Figure 3.39.4: Sentimental analysis: Bing method: Results of execution

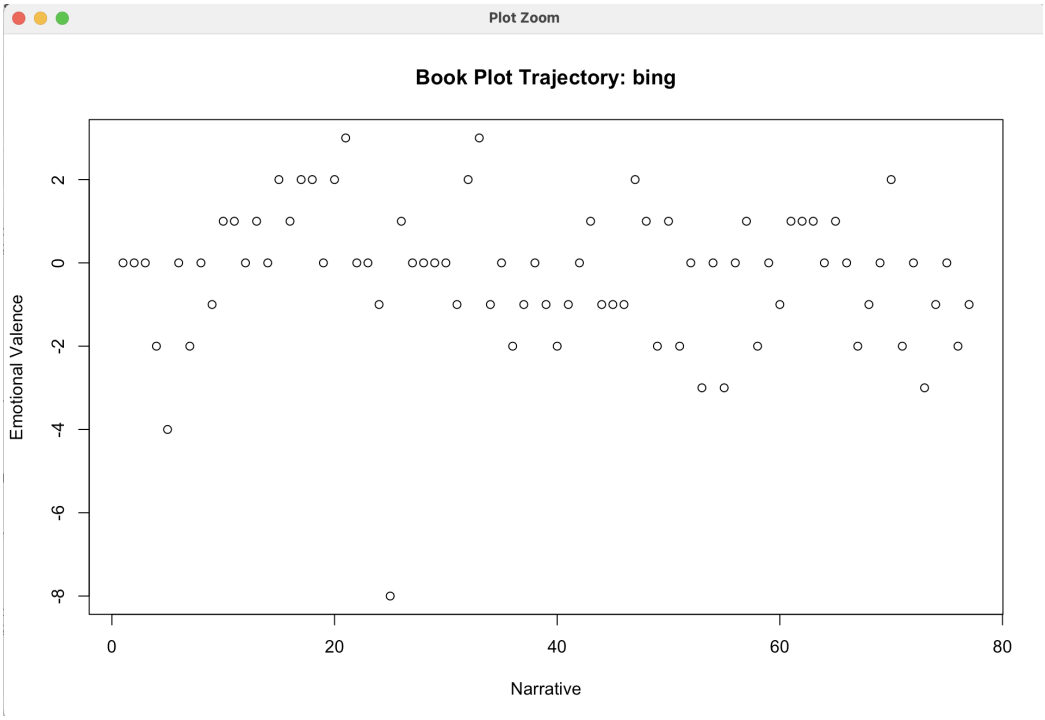


Figure 3.39.5: Sentimental analysis: Bing method: Plot Trajectory

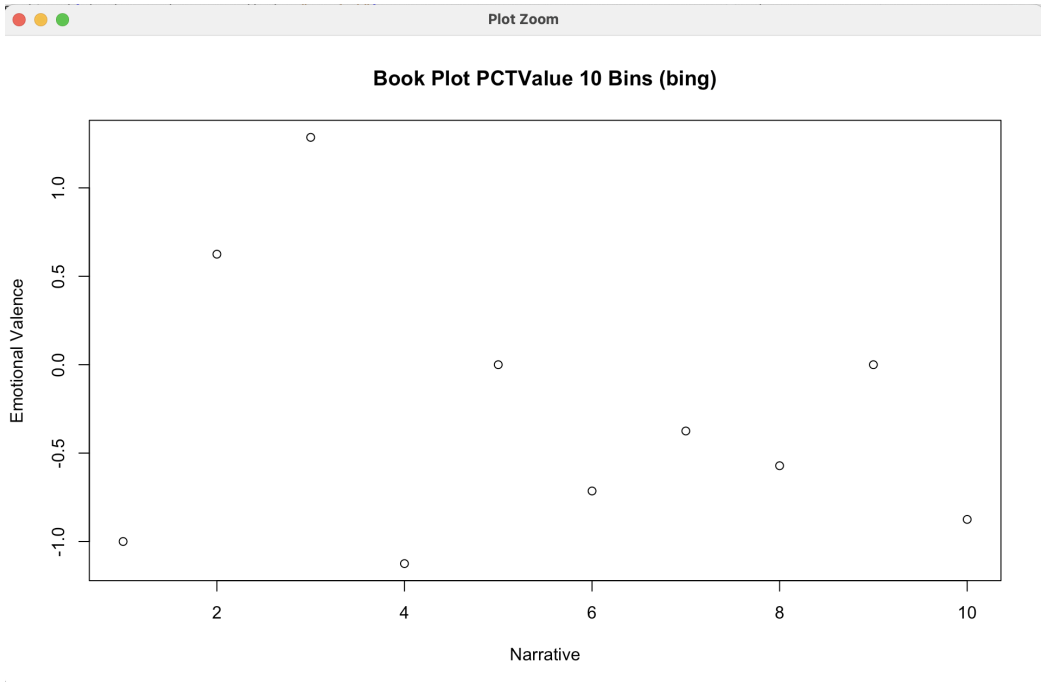


Figure 3.39.6: Sentimental analysis: Bing method: Plot percentage values

```
Press [Enter] to see with NRC method

Sentiment for each sentence obtained with nrc method:
   anger anticipation disgust fear joy sadness surprise trust negative positive
1      0            0       0    0   0       0        0     0        0        0
2      0            0       0    0   1       0        0     0        0        1
3      0            0       0    0   0       0        0     0        0        0
4      1            1       1    2   0       2        1     1        2        1
5      4            2       2    5   1       3        2     2        5        1
```

Figure 3.39.7.1: Sentimental analysis: NRC method: Results of execution (Part 1)

```
76     1            1       1    1   0       2        1     0        3        1
77     0            0       0    0   0       0        1     0        1        1
Top 3 emotions for this chapter:
trust : 79 ; fear : 69 ; anticipation : 57 ;

Press [Enter] to continue to the next chapter's sentimental analysis
```

Figure 3.39.7.2: Sentimental analysis: NRC method: Results of execution (Part 2)

Chapter	Syuzhet Valence	Syuzhet Mean	Bing Valence	Bing Mean	NRC Top 3 Emotions (with counts)
1	8.4	0.1091	-22	-0.2857	Trust (79), Fear (69), Anticipation (57)
2	-26.75	-0.5047	-60	-1.1321	Fear (71), Anticipation (50), Trust (37)
3	25.15	0.2891	3	0.0345	Anticipation (66), Fear (63), Trust (57)
4	18	0.2609	-10	-0.1449	Trust (57), Anticipation (49), Fear (42)
5	11.65	0.2589	2	0.0444	Trust (40), Fear (39), Anticipation (35)
6	-18.55	-0.3710	-32	-0.6400	Fear (66), Anger (48), Trust (40)
7	25.9	0.4246	4	0.0656	Anticipation (47), Trust (44), Joy (40)
8	-22	-0.3793	-26	-0.4483	Fear (63), Trust (47), Anger (41)

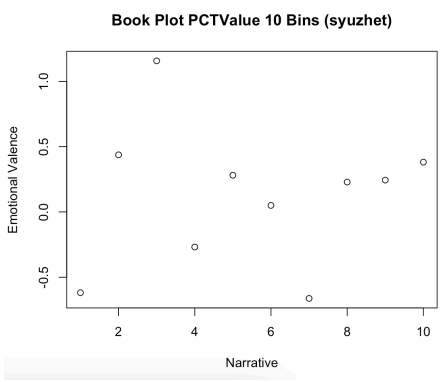
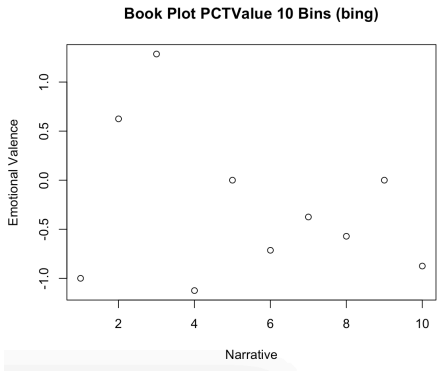
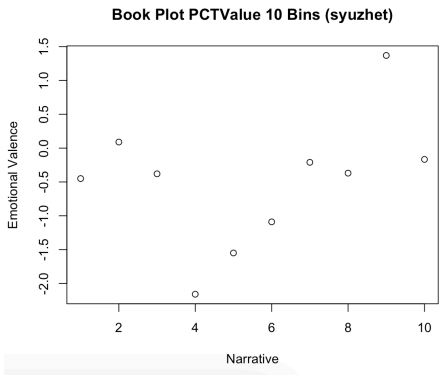
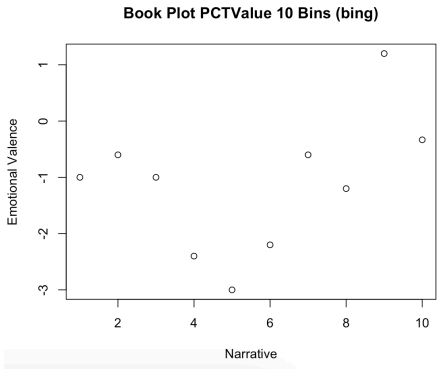
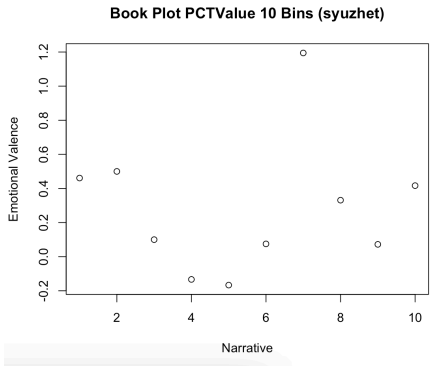
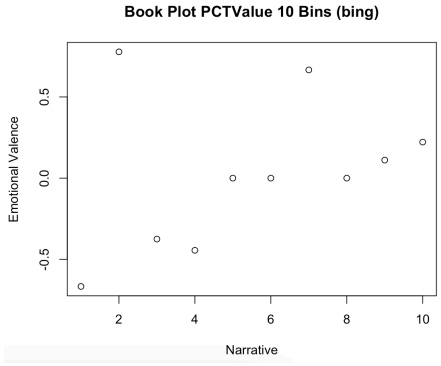
9	-0.35	-0.0071	-16	-0.3265	Fear (48), Trust (40), Anticipation (36)
10	9.85	0.0801	-33	-0.2683	Trust (114), Fear (99), Anticipation (88)
11	25.7	0.2920	-5	-0.0568	Trust (80), Joy (54), Anticipation (52)
12	14.9	0.2292	-4	-0.0615	Fear (55), Trust (55), Anticipation (37)
13	20.5	0.2440	12	0.1429	Fear (64), Anger (57), Trust (55)
14	-8.25	-0.0676	-13	-0.1066	Trust (80), Anticipation (74), Fear (74)

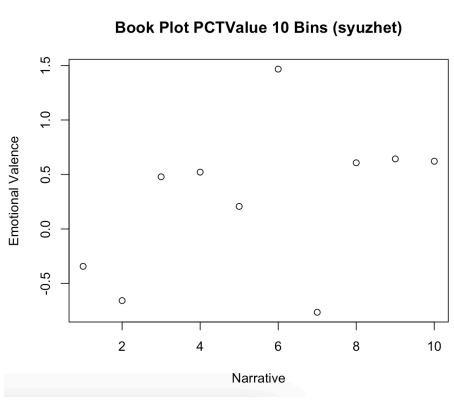
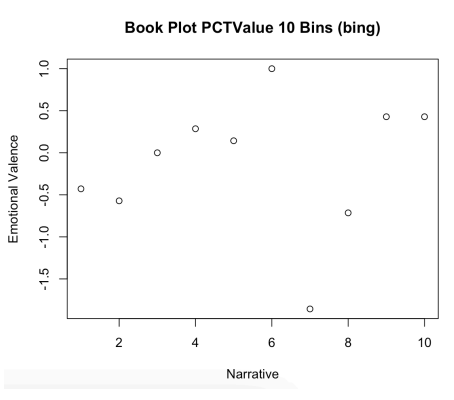
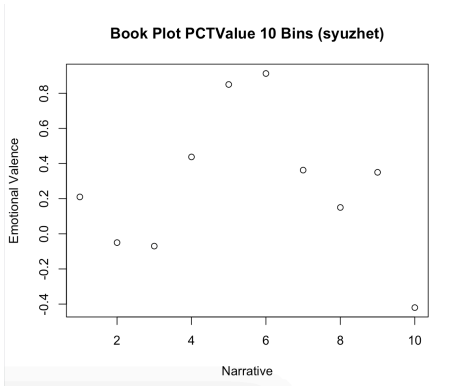
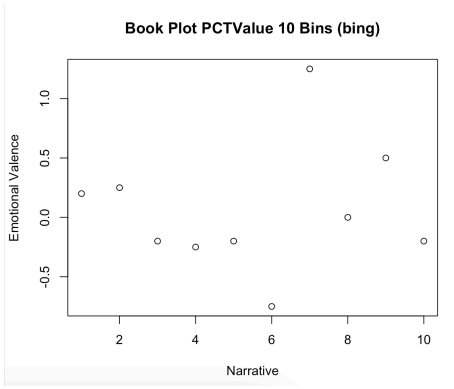
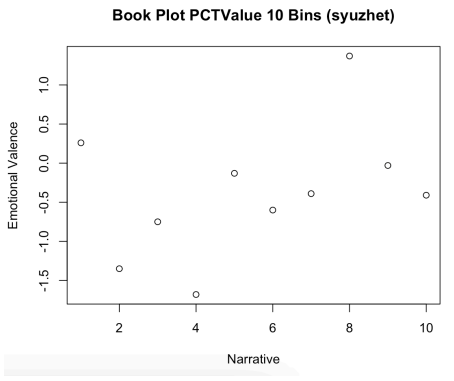
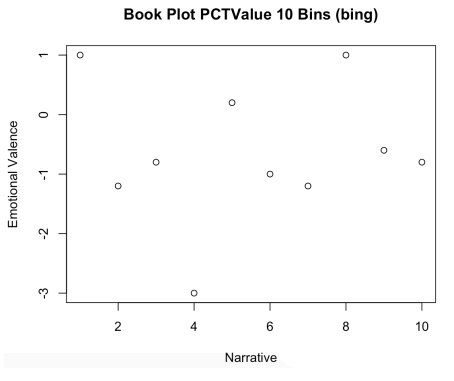
Table 3.40.1: Sentimental analysis summary: Syuzhet, Bing and NRC methods

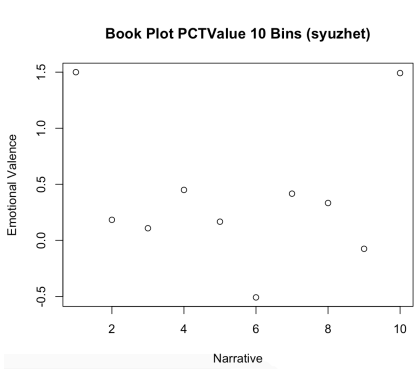
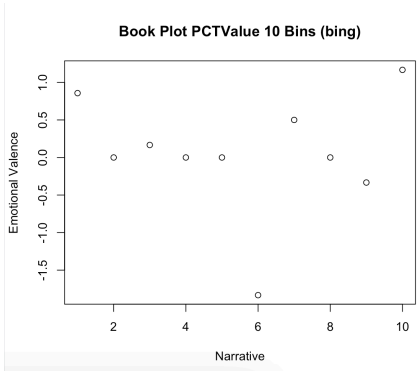
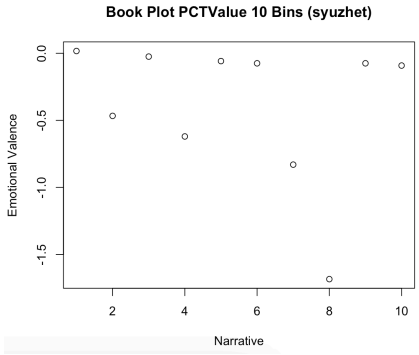
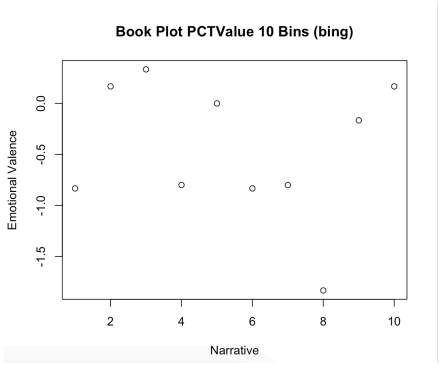
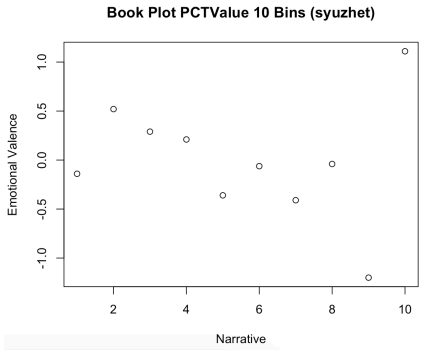
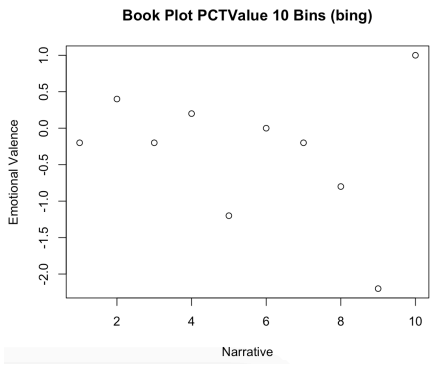
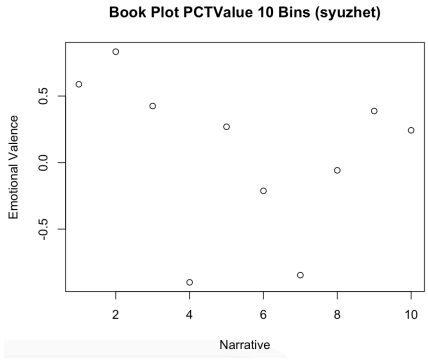
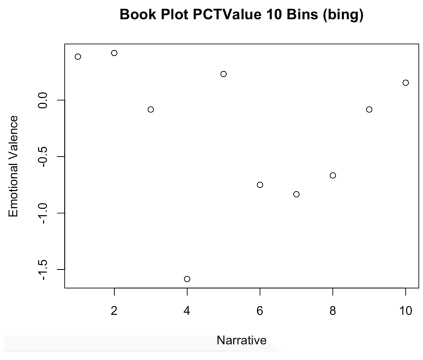
Chapter	Method	Min	1st Qu.	Median	Mean	3rd Qu.	Max
1	Syuzhet	-4.95	-0.75	0.00	0.1091	0.80	3.70
1	Bing	-8.00	-1.00	0.00	-0.2857	1.00	3.00
2	Syuzhet	-4.90	-1.25	-0.50	-0.5047	0.00	3.85
2	Bing	-8.00	-2.00	-1.00	-1.1321	0.00	3.00
3	Syuzhet	-2.05	-0.40	0.25	0.2891	0.90	2.85
3	Bing	-3.00	-1.00	0.00	0.0345	1.00	4.00
4	Syuzhet	-2.75	-0.50	0.25	0.2609	1.00	3.25
4	Bing	-6.00	-1.00	0.00	-0.1449	1.00	3.00
5	Syuzhet	-1.95	-0.50	0.00	0.2589	0.95	4.40
5	Bing	-2.00	-1.00	0.00	0.0444	1.00	5.00
6	Syuzhet	-3.40	-1.50	-0.025	-0.371	0.30	8.00
6	Bing	-6.00	-2.00	-1.00	-0.64	1.00	7.00
7	Syuzhet	-1.60	-0.20	0.25	0.4246	0.75	4.05
7	Bing	-4.00	0.00	0.00	0.0656	1.00	3.00
8	Syuzhet	-4.35	-1.10	-0.25	-0.3793	0.40	1.80
8	Bing	-4.00	-1.00	-0.50	-0.4483	0.00	4.00
9	Syuzhet	-3.25	-0.60	0.00	-0.0071	0.25	3.85
9	Bing	-7.00	-1.00	0.00	-0.3265	0.00	4.00
10	Syuzhet	-5.25	-0.75	0.00	0.0801	0.875	3.05
10	Bing	-7.00	-1.00	0.00	-0.2683	1.00	5.00
11	Syuzhet	-2.25	-0.25	0.075	0.292	1.025	3.80
11	Bing	-4.00	-1.00	0.00	-0.0568	1.00	4.00
12	Syuzhet	-3.25	-0.70	0.00	0.2292	0.95	6.35
12	Bing	-5.00	-1.00	0.00	-0.0615	1.00	8.00
13	Syuzhet	-3.25	-0.5125	0.075	0.2440	0.7625	4.85
13	Bing	-5.00	-1.00	0.00	0.1429	1.00	6.00

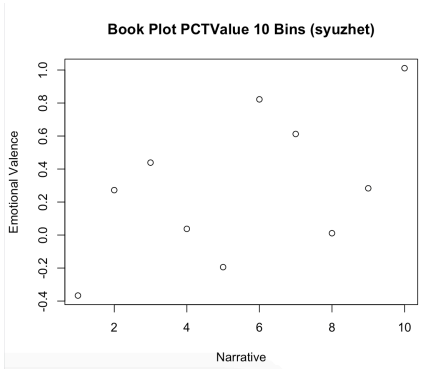
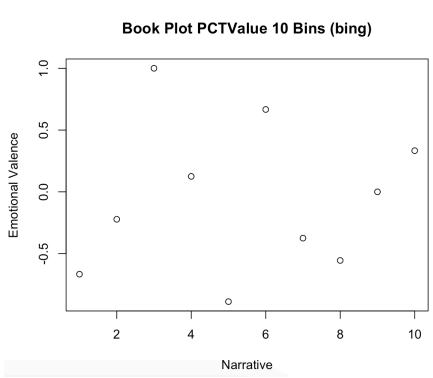
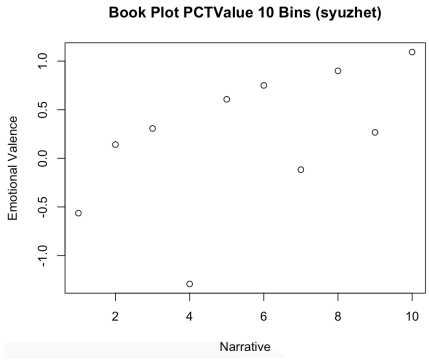
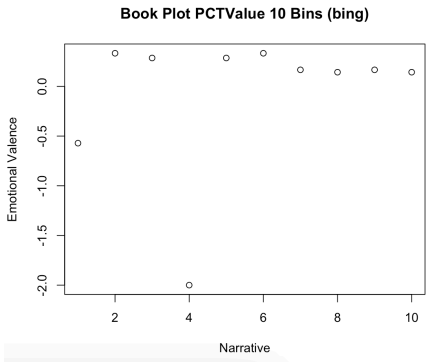
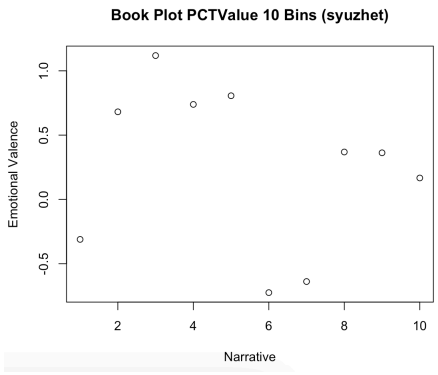
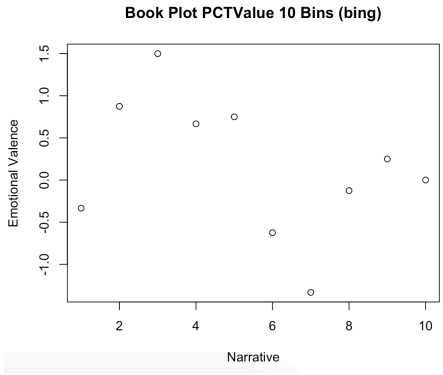
14	Syuzhet	-5.25	-0.7125	0.00	-0.0676	0.50	3.75
14	Bing	-6.00	-1.00	0.00	-0.1066	1.00	5.00

Table 3.40.2: Sentimental Distribution: Syuzhet and Bing methods

Chapter number	Syuzhet	Bing
Chapter 1		
Chapter 2		
Chapter 3		

Chapter 4		
Chapter 5		
Chapter 6		

Chapter 7		
Chapter 8		
Chapter 9		
Chapter 10		

Chapter 11		
Chapter 12		
Chapter 13		

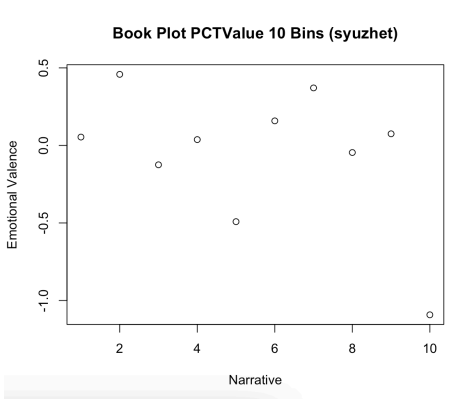
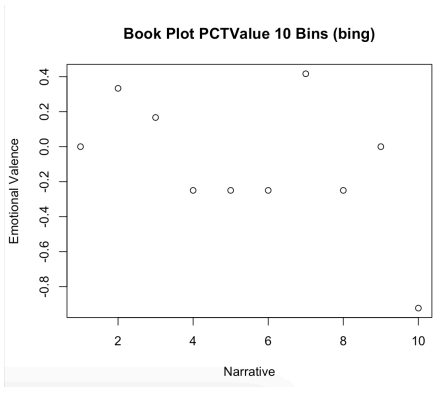
Chapter 14		

Table 3.40.3: Plot Percentage Values: Syuzhet and Bing methods

4. Unsupervised Learning

Topicmodels

Furthermore, let's move on to theme prediction. In this part, we are going to use two R libraries for this purpose: topicmodels and textmineR.

First, we load the chapter texts from files and store them all into one variable called chapters. Figure 4.1 shows these steps.

```
# Unsupervised Machine Learning
chapter_files <- list.files(file.path(path, "Chapters"), full.names = TRUE)
chapters <- lapply(chapter_files, function(f) paste(readLines(f), collapse = " "))
```

Figure 4.1: Data preparation

Then, we set a random seed and proceed to iteration through each chapter of the book to find out 5 topics per document. Figure 4.2 illustrates the code to run to get 5 topics expressed by 5 words for each chapter. In this code, first we are storing current chapter's sentences and then splitting this list of sentences per 5 documents of approximately equal number of sentences. This step is important because very short documents (like a single sentence in our case) may not provide

enough context for effective topic modeling. By creating five pseudo-documents from each chapter, we help the algorithm detect more coherent and consistent topics.

Next, we combine each group of sentences into a single string, creating a list of five pseudo-documents. We then convert this list into a quanteda corpus, which serves as the foundation for further text processing.

After creating the corpus, we tokenize it - breaking it down into individual terms - and perform several preprocessing steps: removing punctuation, eliminating stop words, and stemming words to their root forms.

We then convert the tokens into a document-feature matrix (DFM). We also remove words that appear less than two times across all documents.

Note: While the course rubric suggested removing punctuation, stop words, and rare terms directly during the creation of the DFM, we saw warnings and errors due to deprecated functionality in newer versions of the quanteda package. So we have separated the process of filtering from converting into a document-term matrix and then applying the dfm() function.

Next, we convert this Document-Feature matrix into a topicmodels format using convert() function. Then we finally apply the LDA() function with our previously obtained Document-Feature matrix, setting parameter k = 5 to have 5 topics and using the Gibbs method.

Then, we call the terms() function with a previously obtained object presicing that we want to have 5 words per topic and printing these terms. Finally, we found an interesting function called perplexity() to measure how well the model predicted topics. Lower perplexity indicates a better fit.

In total, we use three functions from the topicmodels package: LDA(), terms(), and perplexity().

Figure 4.3 shows an example of the output for the first three chapters, where each chapter includes five topics described by five terms, along with their corresponding perplexity score. Table 4.4 shows the entire data per chapter 1 to 14. However, we still observe a significant number of stop words in the output. Therefore, in the next step, we will create a custom list of stop words tailored to our specific context, remove them from the text, and then re-run the analysis.

```
# Topic Models
set.seed(1)
cat("Here are topics given by the Topic Models library : ")

for (i in 1:14) {
  cat("Chapter", i, ":\n")
  # Split chapter into chunks of phrases (because it didn't worked with the whole text in one doc)
  sentences <- get_sentences(chapters[[i]])
  # 5 groups of approximate equal number of sentences
  chunks <- split(sentences, ceiling(seq_along(sentences) / 5))
  pseudo_docs <- sapply(chunks, function(group) paste(group, collapse = " "))

  # Converting our chapter's chunks into a quanteda corpus
  current_corpus <- corpus(pseudo_docs)
  # Cleaning everything useless
  current_tokens <- tokens(current_corpus, remove_punct = TRUE)
  current_tokens <- tokens_remove(current_tokens, stopwords("english"))
  current_tokens <- tokens_wordstem(current_tokens)

  # Transforming into document-feature matrix
  current_dfm <- dfm(current_tokens)
  current_dfm <- dfm_trim(current_dfm, min_termfreq = 2)

  current_dfm <- convert(current_dfm, to = "topicmodels")
  current_lda <- topicmodels::LDA(current_dfm, k = 5, method = "Gibbs")

  current_terms <- terms(current_lda, 5)

  cat("Top 5 topics and corresponding terms for this chapter:\n")
  print(current_terms)

  # Measuring how well the model predicts new data
  cat("Measurement of how the prediction is good:")
  print(perplexity(current_lda, newdata = current_dfm))

  cat("\n")
}
```

Figure 4.2: Topic prediction with topicmodels

It is interesting to note that some words in the results miss their final letters. This is due to the application of word stemming, which reduces words to their root forms to group similar terms together.

```
+   cat("\n")
+ }
Chapter 1 :
Top 5 topics and corresponding terms for this chapter:
     Topic 1   Topic 2  Topic 3 Topic 4 Topic 5
[1,] "cave"    "man"    "trail" "two"   "knew"
[2,] "three"   "valley" "powel" "feet"  "possibl"
[3,] "arizona" "open"   "upon"  "old"   "death"
[4,] "never"   "know"   "near"  "dead"  "pass"
[5,] "powel"   "far"    "morn"  "now"   "upon"
Measurement of how the prediction is good:[1] 198.332

Chapter 2 :
Top 5 topics and corresponding terms for this chapter:
     Topic 1 Topic 2   Topic 3 Topic 4    Topic 5
[1,] "open"  "sound"   "first" "upon"     "cave"
[2,] "thing" "behind"  "face"  "lay"      "ledg"
[3,] "hors"  "dead"    "pass"  "move"     "time"
[4,] "yet"   "thought" "unabl" "stood"    "upon"
[5,] "now"   "back"    "cave"  "approach" "reason"
Measurement of how the prediction is good:[1] 114.5111

Chapter 3 :
Top 5 topics and corresponding terms for this chapter:
     Topic 1  Topic 2 Topic 3  Topic 4    Topic 5
[1,] "toward" "upon"  "mount"  "martian"  "earth"
[2,] "littl"  "feet"  "two"    "side"     "hundr"
[3,] "one"    "mar"   "behind" "later"    "direct"
[4,] "time"   "seem"  "spear"  "low"      "first"
[5,] "note"   "four"  "hand"   "enclosur" "anim"
Measurement of how the prediction is good:[1] 190.0974
```

Figure 4.3: Topic prediction with topicmodels: the beginning of the output: first 3 chapters

Chapter number	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Score
Chapter 1	cave, three, arizona, never, powel	man, valley, open, know, far	trail, powel, upon, near, morn	two, feet, old, dead, now	knew, possibl, death, pass, upon	198.332

Chapter 2	open, thing, hors, yet, now	sound, behind, dead, thought, back	first, face, pass, unabl, cave	upon, lay, move, stood, approach	cave, ledg, time, upon, reason	114.5111
Chapter 3	toward, littl, one, time, note	upon, feet, mar, seem, four	mount, two, behind, spear, hand	martian, side, later, low, enclosur	earth, hundr, direct, first, anim	190.0974
Chapter 4	tarka, except, ten, now, men	upon, chieftain, hundr, light, warrior	martian, age, plaza, one, mar	feet, time, great, entranc, room	build, death, open, appear, near	163.3405
Chapter 5	martian, mar, night, hand, atmospher	upon, away, short, one, left	almost, without, two, alon, care	moon, sola, brute, might, last	room, anim, learn, watch, time	116.5773
Chapter 6	upon, held, one, creatur, back	turn, arm, beast, mate, close	martian, cudgel, warrior, whose, sola	ape, stand, upon, battl, fight	seem, follow, eye, bodi, rage	120.5045
Chapter 7	martian, littl, side, two, femal	one, sola, chieftain, tarka, mother	incub, egg, martian, five, year	chariot, day, young, warrior, sola	entir, communiti, year, martian, upon	127.614
Chapter 8	warrior, upon, martian, see, upper	warrior, strang, deck, follow, eye	build, craft, great, figur, open	vessel, seem, swung, sight, work	upon, fire, green, one, gun	138.8348
Chapter 9	men, make, us, note, express	carri, will, part, hand, first	sola, one, upon, women, word	martian, great, war, weapon, green	upon, toward, learn, young, sarkoja	92.25799
Chapter 10	one, now, will, brute, speak	martian, chieftain, toward, kill, turn	upon, warrior, may, ptomel, hill	prison, lorqua, act, saw, dead	woola, woman, face, long, must	255.7122
Chapter 11	know, tell, barsoom, tar, perfect	dejah, thori, upon, sola, peopl	much, earth, lorqua, though, eye	found, kill, us, quarter, carter	men, race, beauti, martian, mani	172.916
Chapter 12	women, now, communiti, ptomel, girl	floor, warrior, thark, one, except	us, may, one, red, yet	upon, hajus, adjoin, dejah, age	martian, escap, tar, matter, prison	137.3142

Chapter 13	dejah, tar, tarka, thoat, kind	even, great, communiti, well, arm	thori, martian, time, much, far	upon, thark, night, day, though	warrior, know, one, sinc, alway	168.7783
Chapter 14	upon, will, fight, sola, without	thori, sarkoja, dejah, word, thought	egg, might, barsoom, day, incub	love, carter, whose, much, peopl	one, dejah, tar, time, escap	226.8114

Figure 4.4: Topic prediction with topicmodels: 5 topics per chapter

From this table 4.4, we can see that these stop words are the following: three, upon, two, now, yet, first, one, four, ten, except, near, almost, without, whose, seem, five, us, will, toward, must, much, may, always, even, well, kind, without, might, whose and much. Therefore, let's update our code, so that at the same time when we clean the text we also remove these words to be able to better analyse the text. Below, on Figure 4.5, you can find the updated source code, with new added lines highlighted in blue.

And right after it, on Table 4.6, you can find the entire updated data per chapter.

```r
# Topic Models
set.seed(1)
cat("Here are topics given by the Topic Models library : ")

for (i in 1:14) {
  cat("Chapter", i, ":\n")
  # Split chapter into chunks of phrases (because it didn't worked with the whole text in one doc)
  sentences <- get_sentences(chapters[[i]])
  # 5 groups of approximate equal number of sentences
  chunks <- split(sentences, ceiling(seq_along(sentences) / 5))
  pseudo_docs <- sapply(chunks, function(group) paste(group, collapse = " "))

  # Converting our chapter's chunks into a quanteda corpus
  current_corpus <- corpus(pseudo_docs)
  # Cleaning everything useless
  current_tokens <- tokens(current_corpus, remove_punct = TRUE)
  current_tokens <- tokens_remove(current_tokens, stopwords("english"))

  # Removing specific to our context stop words
  my_stopwords <- c("three", "upon", "two", "now", "yet", "first", "one", "four", "ten", "except",
                    "near", "almost", "without", "whose", "seem", "five", "us", "will", "toward",
                    "must", "much", "may", "always", "even", "well", "kind", "might")
  current_tokens <- tokens_remove(current_tokens, my_stopwords)

  current_tokens <- tokens_wordstem(current_tokens)

  # Transforming into document-feature matrix
  current_dfm <- dfm(current_tokens)
  current_dfm <- dfm_trim(current_dfm, min_termfreq = 2)

  current_dfm <- convert(current_dfm, to = "topicmodels")
  current_lda <- topicmodels::LDA(current_dfm, k = 5, method = "Gibbs")

  current_terms <- terms(current_lda, 5)

  cat("Top 5 topics and corresponding terms for this chapter:\n")
  print(current_terms)

  # Measuring how well the model predicts new data
  cat("Measurement of how the prediction is good:")
  print(perplexity(current_lda, newdata = current_dfm))

  cat("\n")
}
```

Figure 4.5: Topic prediction with topicmodels: updated version which remove more stop words

Chapter number	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Score
Chapter 1	pass, hors, bodi, pursu, level	powel, sudden, dead, arizona, far	death, convinc, follow, never, right	trail, possibl, indian, knew, way	man, cave, powel, continu, know	182.5224

Chapter 2	sound, sudden, seem, turn, open	face, thing, behind, thought, stood	pass, dead, cave, bodi, night	time, hors, eye, short, stretch	cave, lay, move, ledg, left	107.4582
Chapter 3	spear, mount, leg, note, extrem	appear, learn, entir, hand, wall	direct, evid, martian, attempt, turn	littl, feet, martian, low, white	earth, eye, head, arm, seem	182.4975
Chapter 4	martian, age, word, room, later	hundr, appear, various, floor, learn	build, mar, creatur, evid, among	tar, plaza, tarka, chieftain, desk	feet, death, great, men, littl	152.1097
Chapter 5	mar, left, atmospher, fur, sola	learn, alon, hand, moon, thought	moon, beast, way, great, hour	martian, away, short, night, mar	brute, room, anim, wonder, last	106.8254
Chapter 6	turn, warrior, stand, beast, life	martian, great, blow, follow, look	eye, cudgel, sola, behind, doorway	back, seem, floor, held, creatur	arm, ape, second, full, breast	109.9943
Chapter 7	larg, tar, wall, everi, languag	littl, martian, entir, side, egg	chariot, incub, sola, young, women	martian, year, incub, day, femal	communiti, point, hundr, develop, captur	125.3884
Chapter 8	seem, build, warrior, strang, figur	see, sight, upper, dead, flame	martian, entir, swung, ground, fire	vessel, craft, deck, slowli, fire	warrior, green, though, window, life	139.0859
Chapter 9	time, carri, sever, convers, martian	women, learn, among, strang, weapon	sola, war, aid, make, rather	men, individu, say, thought, day	sola, martian, young, note, express	87.8896
Chapter 10	hill, speak, sola, like, tar	ptomel, attempt, citi, among, full	woola, prison, know, kill, act	tarka, respect, back, feet, turn	martian, warrior, chieftain, long, warn	250.5412
Chapter 11	believ, kill, guard, thori, found	thori, barsoom, lost, sola, build	dejah, race, tell, martian, thori	men, know, lorqua, carter, beauti	ask, peopl, quarter, john, eye	168.5906
Chapter 12	race, lorqua, ptomel, tal, made	hajus, red, martian, continu, chieftain	escap, tar, either, custom, time	women, communiti, thark, girl, matter	floor, warrior, tarka, direct, sleep	132.3473

Chapter 13	dejah, tarka, moment, follow, warrior	martian, thori, day, littl, far	thoat, sola, sinc, beast, march	warrior, night, communiti, green, everi	great, time, tar, know, ever	157.8401
Chapter 14	egg, feel, incub, ask, sola	thori, word, final, sola, turn	time, love, seem, carter, peopl	dejah, sarkoja, fight, thori, lover	littl, tarka, held, direct, day	213.3271

Figure 4.6: Topic prediction with topicmodels: 5 topics per chapter: updated version without some stop words

Interesting to notice that this time, after we removed stop words, the score measuring how well the prediction performs has considerably improved almost everywhere. And even though we may still see some stop words from time to time, their quantity has significantly decreased. Considering the topics per chapter themselves and what we can glean from these:

- In the first two chapters, words like horse, death, Arizona, pass, pursue, continue, follow, way, man, powel, and cave appear. We can suggest that in this part, there is a character named Powel (or Powell, judging by the words generated by functions from the textmineR package later) (we didn't recognise a word like that, so it is probably a name or a title of something) who is probably a man traveling on a horse in Arizona (or coming from there). There is probably a cave, which appears frequently at this point and might be a transition to a new mysterious world - probably to Mars, considering the title of the book.
- Then, in chapters 3 and 4, we can see words like extreme, appear, learn, turn, Martian, seem, various, and creature. So, we can immediately think that our character is now on Mars and is discovering this new world.

- Furthermore, in chapters 5, 6, and 7, we see words like Mar (probably Mars but stemmed), atmosphere, learn, Martian, wonder, life, look, seem, language, and community. This may describe in more detail the life on Mars: the society, rules, and life in general.
- In the next part (chapters 8, 9, and 10), there are more words like warrior, dead, fire, life, weapon, war, kill, and warn, which might be due to crucial military and strategic events happening on Mars.
- Finally, in the last chapters to analyze (11 through 14), we see more words like lost, beautiful, people, escape, love, and lover. So, the events described in this part might be more related to deeper emotional or sentimental developments, like love, concerns, and relationships.

TextmineR

Now, let's use another package for text analysis called textmineR. From this package, we will use three different models for topic modeling: LDA, LSA, and CTM.

These models, with the same goal, differ in their approach:

- LDA (Latent Dirichlet Allocation) assumes that documents are mixtures of topics and that topics are distributions over words, using probabilistic inference.
- LSA (Latent Semantic Analysis) is based on matrix decomposition (SVD) and uncovers topics by reducing dimensionality of the term-document matrix without assuming a probabilistic model.
- CTM (Correlated Topic Model) extends LDA by allowing topics to be correlated with one another, providing a more flexible representation of topic relationships.

Figures 4.7.1 to 4.7.3 show a loop that iterates through each chapter and applies all three models to identify and compare the topics' prediction.

On Figure 4.7.1, you can see the same process of extracting sentences and dividing the text into five pseudo-documents of approximately equal length. However, in this case, we did not apply any manual text cleaning (stop word removal, stemming, or punctuation filtering). Interestingly, the output appeared clean and meaningful regardless. We have a hypothesis to explain this: because the CreateDtm() automatically performs preprocessing steps internally, such as tokenization, lowercasing, and filtering of common stop words and punctuation.

Next, we construct a Document-Term Matrix from our pseudo-documents, which is in our case an input for the three topic models.

We start with the FitLdaModel() function to apply LDA and extract the top 10 terms per topic. Furthermore, we display additional model information, including topic proportions for each pseudo-document and the most likely topic assigned to each chunk.

Figures 4.7.2 and 4.7.3 show similar structure as the previous code, but using FitLsaModel() and FitCtmModel() to apply the LSA and CTM algorithms, respectively. For each model, we extract the top terms, topic proportions, and dominant topics per pseudo-document in the same consistent format.

The outputs for the first chapter using each of the three models (LDA, LSA, and CTM) are shown on Figures 4.8.1, 4.8.2, and 4.8.3, respectively. The most important results (top 10 topics, 10 main words per topic, and the most likely topic per chapter) for all chapters are presented in Table 4.9.

Note: According to the rubric, we were supposed to use a function called FittmModel(). However, we could not find this function in the textmineR documentation. Thus, we decided to use FitCtmModel(), which we found on the internet and seems to be interesting to test.

In total, we used the following functions from the textmineR package: CreateDtm(), FitLdaModel(), FitLsaModel(), FitCtmModel(), and GetTopTerms().

```r
# Text Mine R
for (i in 1:14) {
  cat("Chapter", i, ":\n")
  # We are splitting chapter into chunks of phrases (because it didn't worked with the whole text in one doc)
  sentences <- get_sentences(chapters[[i]])
  # 5 groups of approximate equal number of sentences
  chunks <- split(sentences, ceiling(seq_along(sentences) / 5))
  pseudo_docs <- sapply(chunks, function(group) paste(group, collapse = " "))
  names(pseudo_docs) = c("doc1", "doc2", "doc3", "doc4", "doc5")
  dtm <- CreateDtm(doc_vec = pseudo_docs, doc_names = names(pseudo_docs))

  cat("LDA model : \n")
  # LDA model
  lda_model <- FitLdaModel(dtm = dtm, k = 5, iterations = 250)
  # Top terms per topic
  top_terms <- GetTopTerms(phi = lda_model$phi, M = 10)
  cat("Top terms per topic :\n")
  print(top_terms)
  cat("\n")
  # Topic proportions for each chapter
  topic_proportions <- lda_model$theta
  cat("Topic proportions for each chunk :\n")
  print(topic_proportions[1:5,])
  cat("\n")
  # Most likely topic per document
  cat("Most likely topic per chunk :\n")
  print(apply(lda_model$theta, 1, which.max)[1:5])
  cat("\n")

  readline(prompt = "\nPress [Enter] to continue to see the results of the LSA model.")
```

Figure 4.7.1: Topic prediction with textmineR: LDA method

```
  cat("LSA model : \n")
  # LSA model
  lsa_model <- FitLsaModel(dtm = dtm, k = 5)
  # Top terms per topic
  top_terms <- GetTopTerms(phi = lsa_model$phi, M = 10)
  cat("Top terms per topic :\n")
  print(top_terms)
  cat("\n")
  # Topic proportions for each chapter
  topic_proportions <- lsa_model$theta
  cat("Topic proportions for each chunk :\n")
  print(topic_proportions[1:5,])
  cat("\n")
  # Most likely topic per document
  cat("Most likely topic per chunk :\n")
  print(apply(lsa_model$theta, 1, which.max)[1:5])
  cat("\n")

  readline(prompt = "\nPress [Enter] to continue to see the results of the CTM model.")
```

Figure 4.7.2: Topic prediction with textmineR: LSA method

```
  cat("CTM model : \n")
  # CTM model
  ctm_model <- FitCtmModel(dtm = dtm, k = 5)
  # Top terms per topic
  top_terms <- GetTopTerms(phi = ctm_model$phi, M = 10)
  cat("Top terms per topic :\n")
  print(top_terms)
  cat("\n")
  # Topic proportions for each chapter
  topic_proportions <- ctm_model$theta
  cat("Topic proportions for each chunk :\n")
  print(topic_proportions[1:5,])
  cat("\n")
  # Most likely topic per document
  cat("Most likely topic per chunk :\n")
  print(apply(ctm_model$theta, 1, which.max)[1:5])
  cat("\n")

  if (i != 14){
    readline(prompt = "\nPress [Enter] to continue to the next chapter's topic analysis.")
  }
}
```

Figure 4.7.3: Topic prediction with textmineR: CTM method

```
+   if (i != 14){
+     readline(prompt = "\nPress [Enter] to continue to the next chapter's topic analysis.")
+   }
+ }
Chapter 1 :
LDA model :
Top terms per topic :
      t_1        t_2         t_3       t_4         t_5
 [1,] "trail"    "convinced" "powell"  "cave"      "hundred"
 [2,] "horse"    "camp"      "morning" "face"      "man"
 [3,] "knew"     "catch"     "valley"  "feet"      "left"
 [4,] "powell"   "made"      "arrows"  "fact"      "body"
 [5,] "pass"     "arizona"   "forced"  "life"      "possibly"
 [6,] "plateau"  "red"       "moon"    "opening"   "death"
 [7,] "suddenly" "ahead"     "rapidly" "knew"      "army"
 [8,] "apaches"  "center"    "reached" "trail"     "captain"
 [9,] "entered"  "clear"     "dead"    "floor"     "determined"
[10,] "indian"   "duty"      "tracks"  "mysteries" "direction"

Topic proportions for each chunk :
             t_1         t_2         t_3         t_4         t_5
doc1 0.030136986 0.221917808 0.002739726 0.002739726 0.742465753
doc2 0.059854015 0.001459854 0.001459854 0.643795620 0.293430657
doc3 0.001503759 0.001503759 0.001503759 0.001503759 0.993984962
doc4 0.299173554 0.001652893 0.547107438 0.034710744 0.117355372
doc5 0.454088050 0.265408805 0.277987421 0.001257862 0.001257862

Most likely topic per chunk :
doc1 doc2 doc3 doc4 doc5
   5    4    5    3    1


Press [Enter] to continue to see the results of the LSA model.
```

Figure 4.8.1: Topic prediction with textmineR: LDA method: output for 1st chapter

```
Press [Enter] to continue to see the results of the LSA model.
LSA model :
Top terms per topic :
      t_1         t_2            t_3         t_4            t_5
 [1,] "desire"    "powell"       "trail"     "saved"        "powell"
 [2,] "drowsy"    "valley"       "left"      "convinced"    "captain"
 [3,] "drunkenly" "animals"      "horse"     "apaches"      "cave"
 [4,] "effort"    "dots"         "suddenly"  "suddenly"     "carter"
 [5,] "friends"   "pack"         "knew"      "reached"      "confederate"
 [6,] "moments"   "entered"      "pass"      "imprecations" "gold"
 [7,] "reel"      "morning"      "fainter"   "urged"        "hundred"
 [8,] "resist"    "catch"        "savages"   "land"         "tracks"
 [9,] "rest"      "extreme"      "knowledge" "difficult"    "water"
[10,] "scarcely"  "mountainside" "pursuing"  "table"        "army"

Topic proportions for each chunk :
             t_1         t_2         t_3         t_4         t_5
doc1 -0.07985057 -0.28388423 -0.12787786 -0.07027830 -0.06306776
doc2 -0.11730028 -0.49696665 -0.05096974 -0.35943214 -0.13658599
doc3 -0.09800773 -0.04073457 -0.11619685 -0.14658646  0.51854528
doc4 -0.21311818  0.17814888 -0.19473459 -0.04769192  0.16795794
doc5 -0.39909745  0.50593113 -0.26426594 -0.49743054 -0.40272995

Most likely topic per chunk :
doc1 doc2 doc3 doc4 doc5
   5    3    5    2    2

Press [Enter] to continue to see the results of the CTM model.
```

Figure 4.8.2: Topic prediction with textmineR: LSA method: output for 1st chapter

```
Press [Enter] to continue to see the results of the CTM model.
CTM model :
Top terms per topic :
      t_1         t_2       t_3         t_4             t_5
 [1,] "trail"     "powell"  "trail"     "death"         "powell"
 [2,] "cave"      "level"   "powell"    "man"           "horse"
 [3,] "hundred"   "morning" "cave"      "convinced"     "suddenly"
 [4,] "dead"      "indian"  "left"      "saved"         "trail"
 [5,] "opening"   "valley"  "pursuing"  "possibly"      "urged"
 [6,] "feet"      "entered" "knew"      "died"          "return"
 [7,] "captain"   "point"   "body"      "imprecations"  "ahead"
 [8,] "mysteries" "party"   "possibly"  "side"          "mining"
 [9,] "face"      "back"    "camp"      "fact"          "narrow"
[10,] "knew"      "moon"    "water"     "close"         "red"

Topic proportions for each chunk :
              t_1          t_2          t_3          t_4         t_5
doc1 2.258017e-04 1.065593e-04 1.698824e-05 9.916116e-01 0.008039028
doc2 9.973385e-01 2.846590e-07 1.282165e-05 5.069302e-05 0.002597683
doc3 9.971465e-01 3.153218e-07 1.369522e-05 5.286093e-05 0.002786611
doc4 9.986489e-04 1.214210e-03 9.235183e-04 7.330183e-04 0.996130604
doc5 2.259571e-07 9.961741e-01 2.689056e-05 2.906243e-05 0.003769677

Most likely topic per chunk :
doc1 doc2 doc3 doc4 doc5
   4    1    1    5    2


Press [Enter] to continue to the next chapter's topic analysis.
```

Figure 4.8.3: Topic prediction with textmineR: CTM method: output for 1st chapter

As mentioned previously, in the upcoming Tables 4.9.1 - 4.9.3, the complete results (topics and 10 main terms per topic) for all chapters per method (LDA, LSA and CTM) are shown. From this table, we can suggest similar predictions about the context of different chapters as with the predictions with topicmodels package previously.

Chapter number	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
Chapter 1	death, convinced, red, floor, made, moon, rapidly, reached, alive,	powell, valley, fact, morning, face, camp, entered, indian, party, water	trail, horse, level, knew, left, suddenly, tracks, catch, led, point	captain, close, determined, lay, mysteries, return, chronicle,	cave, arizona, feet, man, trail, dead, body, possibly, knew

	center			confederate, dollars, gold	
Chapter 2	felt, life, night, air, arizona, beautiful, cacti, clear, darkness, enchantment	cave, opening, approaching, heard, led, move, facing, horrible, hours, savage	fear, god, held, passed, braves, cold, efforts, moving, power, strange	lay, body, left, thought, stood, morning, short, ledge, cautiously, creeping	sound, dead, suddenly, thing, time, cave, cliff, ears, ceased, danger
Chapter 3	earth, low, eggs, hand, present, strange, yards, weapons, mars, armlet	arms, eyes, legs, mount, animal, head, martian, bodies, color, direction	feet, side, enclosure, easily, determined, ground, height, naked, sat, similar	mars, martians, time, appearance, dark, discovery, fact, learn, lower, man	spear, earth, metal, evidently, held, turned, hundred, barrel, caused, death
Chapter 4	long, buildings, room, ten, height, chamber, bore, center, edge, fifty	age, martian, maturity, thousand, voluntarily, years, land, leading, light, low	feet, martians, green, sak, struck, accord, amusement, applause, consideration, gave	plaza, creatures, building, mars, chairs, desks, entrance, hundred, antiquity, arm	tarkas, tars, chieftain, time, martian, exchanged, motioned, repeated, evidently, ability
Chapter 5	brute, sola, atmosphere, darkness, left, milk, apparently, carefully, cold, dark	moon, mars, hours, miles, nearer, night, brilliant, heavens, makes, nights	hand, reached, beheld, building, cautiously, feet, gained, great, huge, learn	animal, short, mars, city, air, evidently, food, plant, sprang, strange	room, beast, men, martian, captive, dog, eyes, ferocious, ground, moment
Chapter 6	martians, feet, set, actions, body, evidently, thing, witnessed, creatures, finished	turned, death, encounter, strength, swinging, day, end, fellow, ferocity, guardian	arms, eyes, floor, great, held, beast, full, window, green, struck	ape, cudgel, earthly, breast, doorway, pain, approached, building, bull, fall	tars, sola, chamber, warrior, tarkas, arm, brute, deserted, follow, good
Chapter 7	sola, martian, animals, hundred, mars, developed, loaded, telepathic, entire, chariots	eggs, years, incubators, year, adult, hatched, incubation, period, return, twenty	incubator, tarkas, tars, side, warriors, chariots, advanced, journey, quickly, bottom	martians, green, community, incubator, large, arid, cruel, days, formed, knew	young, martians, women, line, chariot, fell, language, men, returned, children
Chapter 8	eyes, caught, chariots, made, martians, sola,	green, martian, building, vessels, face,	city, buildings, life, sight, craft, deserted,	fire, swung, upper, works, decks, position,	ground, warriors, strange, slowly,

	day, detail, dragged, entered	return, carried, cover, dead, drifting	moving, neared, rushed, figure	great, craft, back, guns	vessel, creatures, sign, extreme, flames, guy
Chapter 9	men, war, fight, justice, law, peace, sentiments, tarkas, thing, red	sola, women, sarkoja, time, day, death, manner, progressed, ransom, weakness	attitude, expression, kind, language, strange, words, bearing, existence, fortunate, girl	prisoner, audience, carry, conversation, encounter, sola, back, chamber, learned, night	martians, aid, green, weapons, young, conducted, decided, earthly, education, escape
Chapter 10	warrior, blow, face, martian, speak, attitude, battle, brutal, short, turned	great, audience, chamber, sola, martian, leave, dead, city, full, affection	woola, hills, morning, feet, found, limits, sword, tusks, arms, brute	back, green, prisoner, attempt, long, men, kindliness, laughter, woman, captive	lorquas, ptomel, tarkas, tars, chieftain, dejah, eyes, kill, thoris, prisoner
Chapter 11	dejah, thoris, sola, sarkoja, men, building, guard, hands, metal, women	lorquas, ptomel, barsoom, carter, john, kill, tars, good, man, tarkas	fair, race, conditions, great, quarters, ages, martians, highly, found, men	beautiful, lost, people, ancient, found, city, clad, dor, iss, korus	earth, planet, earthly, fully, logic, perfect, strange, told, asked, people
Chapter 12	lorquas, ptomel, prisoner, race, escape, center, commands, left, mighty, peculiar	red, hajus, quarters, tal, continued, floors, girl, greatest, ways, matter	cold, dejah, thoris, women, cruelty, demands, departed, difficult, fall, martian	community, chieftains, green, martians, women, furs, men, silks, balance, children	floor, rooms, adjoining, building, buildings, court, warriors, directed, apartment, back
Chapter 13	sola, thoris, left, powder, radium, knew, born, buildings, city, fear	martian, night, barsoom, carter, dead, earth, dejah, day, listened, thark	dejah, thoris, time, asked, heart, john, red, words, green, answered	warriors, thoats, great, community, tarkas, tars, days, lorquas, ptomel, march	continued, life, live, riders, moment, animals, beast, body, brutes, ears
Chapter 14	dejah, turned, barsoom, thoris, silks, people, word, great, hand, low	thoris, dejah, sarkoja, key, sought, sola, face, feel, john, return	sword, fight, zad, struck, thrust, long, time, back, brute, felt	love, cross, fought, half, poor, shoulder, strange, woman, held, beautiful	eggs, day, time, incubator, green, evidently, hatching, light, march, short

Table 4.9.1: Topic prediction with textmineR: summary of the results: LDA

Chapter number	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
Chapter 1	desire, drowsy, drunkenly, effort, friends, moments, reel, resist, rest, scarcely	powell, valley, animals, dots, pack, entered, morning, catch, extreme, mountainside	trail, left, horse, suddenly, knew, pass, fainter, savages, knowledge, pursuing	saved, convinced, apaches, suddenly, reached, imprecations, urged, land, difficult, table	powell, captain, cave, carter, confederate, gold, hundred, tracks, water, army
Chapter 2	arms, attracts, call, closed, draw, drawn, immensity, instant, iron, lodestone	cave, lay, sound, trail, left, ledge, unknown, suddenly, morning, faint	dead, faint, moving, silence, left, morning, suddenly, unknown, held, fighting	short, cliff, led, sound, trail, lay, indians, face, time, fear	time, sound, fear, escape, reached, turned, ears, unseen, thing, braves
Chapter 3	belts, cavalcade, designated, distance, fellow, galloped, lifted, range, straps, answering	legs, arms, eyes, head, bodies, ears, center, animal, limbs, color	mars, earth, discovery, extreme, played, walk, manner, weapons, eyes, appearance	great, metal, steel, barrel, miles, death, rifle, spear, warned, attempt	spear, huge, metal, looked, watching, mars, caused, noted, weapons, held
Chapter 4	martian, age, thousand, hundred, martians, death, mars, years, maturity, tarkas	feet, tarkas, tars, creatures, plaza, building, chairs, desks, evidently, entrance	tarkas, tars, chieftain, ability, laugh, smile, men, good, made, humor	mars, evidently, creatures, tarkas, tars, fact, plaza, chieftain, chairs, desks	proper, stones, floor, chamber, entrance, warrior, building, beautifully, assembled, form
Chapter 5	ability, advanced, backed, balance, begun, belief, deserted, felt, finding, gait	room, sola, left, captive, dog, moment, scenes, watch, brute, beast	hand, atmosphere, darkness, animal, nights, short, moons, mars, reached, ground	nights, darkness, moons, green, adding, affection, bodily, brightly, brilliantly	sill, building, feet, gained, secure, ground, pulled, room, beast, huge
Chapter 6	accompanied, adoration, affection, apparent, barely, battled, bravery, carefully, clew, contrary	ape, breast, full, floor, neck, death, swinging, fear, head, momentarily	encounter, perceived, safety, fight, death, standing, fellow, outcome, building, frothing	tarkas, tars, turned, warriors, life, chamber, struck, beast, point, arm	floor, great, pain, close, gratitude, mars, held, cudgel, left, arms
Chapter 7	advantage, glad,	women, line,	incubator,	eggs, years,	cruel, live, love,

	undoubted, vexed, added, additional, carried, counts, creatures, defective	martians, capturing, opening, permitted, walls, fell, returned, young	tarkas, tars, eggs, journey, bottom, chieftain, jed, sea, vaults	year, martians, incubators, adult, incubation, period, perfect, twenty	martians, community, green, point, common, mother, unknown
Chapter 8	answered, answering, appeal, contempt, courage, customs, dejection, depths, edifice, faded	fire, guns, works, upper, swung, volley, vessels, deadly, aim, apparatus	fire, caught, eyes, guns, met, deadly, fear, plaza, detail, owing	ground, city, swung, low, mounted, sudden, majestically, scarcely, slowly, flames	distance, swung, carried, scene, strange, creatures, mighty, unfriendly, city, deserted
Chapter 9	actual, administration, ages, arises, art, ascendency, bodies, branches, culprit, custom	weapons, women, young, war, martians, aid, individual, earthly, day, progressed	men, time, existence, fortunate, part, girl, life, thing, hands, tarkas	existence, fortunate, sola, women, death, surprised, accomplished, ancient, asleep, assure	young, night, weapons, time, beautiful, females, adults, affect, couple, customary
Chapter 10	accoutrements, america, approach, aware, bearing, claim, cleared, earth, flash, interrupted	warrior, speak, tarkas, tars, face, blow, green, countenance, thought, lorquas	back, laughter, head, face, feet, hills, lips, passed, left, tarkas	prisoner, city, audience, chamber, nature, attempt, lorquas, ptomel, proceedings, sola	city, found, woola, hills, walk, prisoner, brute, masters, ravines, limits
Chapter 11	antecedents, behest, confidence, craved, difficult, feminine, flaws, head, hope, perceive	dejah, thoris, sola, sarkoja, people, women, found, quarters, hands, building	dejah, thoris, women, sarkoja, men, made, resulted, hands, reached, arm	earth, men, strange, carter, john, planet, barsoom, sola, fully, young	found, lorquas, ptomel, metal, building, quarters, highly, sola, audience, chamber
Chapter 12	floor, rooms, adjoining, buildings, court, sleeping, apartment, building, warriors, dejah	hajus, tal, tarkas, tars, red, escape, women, prisoner, girl, greatest	women, chieftains, community, martians, furs, silks, ages, children, balance, constituted	red, hajus, tal, green, girl, greatest, directed, matter, dead, martians	fall, dejah, thoris, cold, absolute, abysmal, atavism, awaited, brave, braves
Chapter 13	accidentally, anger, broke,	warriors, thoats, community,	powder, radium, night, treatment,	riders, beast, ears, pistol,	barsoom, green, life, martian,

	conclusion, contrasted, gay, happy, health, home, joking	days, martians, ships, tarkas, tars, beasts, method	exploding, battle, thoats, riders, beast, ears	thoats, treatment, sola, celerity, woola, life	time, beast, ears, pistol, treatment, dead
Chapter 14	dejah, thoris, sarkoja, sola, hand, turned, struck, love, day, thrust	eggs, sword, long, green, day, martian, wished, choice, drawn, weapon	sarkoja, struck, hand, sword, sola, fight, thrust, long, day, mind	eggs, egg, hatching, grow, smaller, green, sarkoja, love, incubator, tarkas	side, evidently, time, sought, effective, escape, pain, parry, rushed, sharp

Table 4.9.2: Topic prediction with textmineR: summary of the results: LSA

Chapter number	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
Chapter 1	powell, trail, knew, tracks, left, captain, pursuing, horse, water, moon	powell, suddenly, horse, red, urged, made, land, men, arrows, return	trail, dead, face, saved, feet, opening, convinced, fact, direction, cliff	man, cave, possibly, death, mysteries, body, hundred, years, lay, day	powell, valley, morning, cave, entered, floor, side, animals, dots, pack
Chapter 2	cave, opening, led, unable, night, cliff, escape, approaching, heard, trail	cave, lay, back, dead, ledge, morning, unknown, stood, eyes, face	felt, god, beautiful, cacti, enchantment, gazed, inspiring, landscape, power, wonders	sound, thing, cave, fear, time, suddenly, strapped, unseen, ears, braves	thought, passed, left, body, life, stood, held, short, cold, fact
Chapter 3	spear, metal, huge, earth, mars, warriors, hand, words, evidently, low	feet, martian, hundred, spear, mount, arms, side, height, naked, yards	mars, feet, low, eyes, earth, white, wall, enclosure, similar, strange	earth, extreme, legs, weight, martians, arms, learned, great, bodies, young	time, easily, weapons, appearance, manner, discovery, entire, noted, present, watching
Chapter 4	feet, tars, tarkas, word, fellow, floor, good, struck, applause, learned	enormous, entrance, building, edge, greatly, land, low, male, proper, steps	martian, age, ten, thousand, hundred, years, appearance, room, maturity, color	chieftain, tars, tarkas, mars, desks, chairs, men, martian, evidently, creatures	motioned, feet, plaza, made, time, martians, creatures, repeated, hundred, large
Chapter 5	moon, mars,	brute, milk,	hand, moons,	room, sola, left,	city, thought,

	miles, nearer, night, hours, makes, oil, quarter, thousand	animal, mars, cautiously, food, jumper, plant, ten, short	darkness, atmosphere, ground, nights, pulled, sill, cold, dark	men, beast, captive, dog, eyes, lay, moment	sprang, martian, mars, reached, short, animal, beheld, air
Chapter 6	ape, full, pain, floor, swinging, earthly, arms, cudgel, accomplishing, ado	life, breast, battle, back, turned, body, witnessed, strength, ape, end	martians, set, evidently, green, arms, earthly, mate, high, creature, creatures	cudgel, eyes, blow, standing, fight, chamber, warriors, encounter, death, doorway	martians, gratitude, mars, held, feet, great, actions, tarkas, tars, sola
Chapter 7	martians, women, line, young, children, eggs, time, incubator, fell, returned	incubator, sola, martian, side, tarkas, tars, advanced, language, quickly, warriors	eggs, incubator, years, vaults, martian, period, year, perfect, twenty, return	martians, common, cruel, live, love, mother, mothers, unknown, year, incubator	chariots, animals, martian, mars, entire, loaded, drawn, meal, warriors, young
Chapter 8	warriors, guns, fire, deadly, volley, direction, aim, apparatus, crew, manner	ground, eyes, slowly, city, martians, sign, flames, guy, head, higher	craft, building, sight, martian, warriors, neared, caught, figure, great, met	fire, craft, warriors, vessel, decks, position, swung, works, vessels, moving	warriors, green, strange, creatures, mighty, unfriendly, sight, return, hope, building
Chapter 9	young, aid, weapons, sola, green, martians, earthly, escape, mars, men	guards, audience, carry, great, days, lorquas, ptomel, prisoner, sarkoja, martians	sola, conversation, sarkoja, prisoner, beautiful, night, weakness, time, captive, question	men, war, kind, expression, strange, make, red, bearing, fight, justice	sola, existence, fortunate, women, time, part, females, words, girl, thing
Chapter 10	blow, face, brute, martian, great, tusks, warrior, moment, eyes, sword	prisoner, lorquas, ptomel, sola, audience, chamber, ages, air, brutal, helium	city, woola, dead, martian, leave, audience, chamber, attempt, resulted, adventure	man, arms, chieftain, martian, common, tharkian, ways, prisoner, men, mighty	back, laughter, woola, turned, long, feet, head, green, full, tars
Chapter 11	planet, asked, told, perfect, thoris, dejah, barsoom, people, eyes, good	sola, thoris, earth, strange, barsoom, escape, wore, room	people, beautiful, found, kill, city, clad, speak, valley, thoris, dejah	race, men, dejah, lost, martians, conditions, reached, women, ages	sola, quarters, great, rank, wished, tars, tarkas, found, dejah, thoris

Chapter 12	hajus, tal, red, girl, greatest, made, green, directed, tarkas, tars	prisoner, cold, dejah, thoris, ways, chieftain, commands, fall, give, position	community, floor, lorquas, ptomel, martians, tarkas, tars, women, building, warriors	adjoining, buildings, court, floor, rooms, sleeping, apartment, people, animals, apartments	race, women, silks, furs, customs, chieftains, ages, tharks, departed, females
Chapter 13	dead, martian, enemy, knew, thark, heart, listened, left, moment, live	powder, radium, treatment, night, battle, thoats, riders, continued, beast	sola, lorquas, ptomel, great, plaza, thoris, dejah, community, days, poor	thoris, dejah, martian, great, good, barsoom, green, times, night, breath	thoats, tarkas, tars, warrior, great, asked, march, battle, arm, fighting
Chapter 14	eggs, thoris, dejah, time, sarkoja, battle, incubator, side, thrust, day	thoris, dejah, tarkas, tars, key, escape, short, lived, night, ten	love, sword, word, time, long, day, held, wished, woman, barsoom	thoris, dejah, sola, word, turned, great, uncle, thought, sarkoja, people	sola, dejah, thoris, polish, teeth, grandmother, low, brothers, friendship, parents

Table 4.9.3: Topic prediction with textmineR: summary of the results: CTM

Finally, we will briefly compare the two packages used (topicmodels and textmineR):

First, textmineR is much more convenient, as the entire text cleaning process is handled internally, whereas with topicmodels, we had to run several functions manually. Even after that, a considerable number of stop words remained, and we had to manually populate a variable to remove them. In contrast, with textmineR, almost no stop words appeared. More generally, we found that textmineR offers more tools for topic prediction and analysis of the results compared to topicmodels. However, it was interesting to explore both packages.

Discussion

During this project we have learned many useful skills: how to work with a large dataset in the text format and how to use functions to extract as much data as possible in the most efficient way. We have discovered interesting and useful functions and how these help to analyze the text. Namely, what are the steps to take prior text analysis: data cleaning and formatting. What packages and tools exist to predict topics and how they work. We also improved our ability to analyze these tools thanks to the internet searches, and the information these tools provide by looking and judging critically the output.

We have also practiced how to handle errors and bugs. For example, following the rubric in the last part we had to use topicmodels library, where first we had some issues because some functions were deprecated in earlier versions. Thus, we had to deal with it by looking on the internet for other ways to solve the task. It was also challenging to understand some functions' outputs, and how to interpret it in the context of our text. But with help of google and some R forums, we have managed to do this and acquired important knowledge and practice.

This is a valuable experience because we acquired some fundamental knowledge about how text analytics occur and tools that are often used for this purpose, and we are confident that it is going to be useful for our future careers.