ALMA MATER STUDIORUM – UNIVERSITY OF BOLOGNA

SCHOOL OF ECONOMICS, MANAGEMENT, AND STATISTICS

Bachelor's degree in Statistical Sciences – Stats&Maths

GAME OF THRONES:

SENTIMENT ANALYSIS OF THE GEORGE R.R. MARTIN'S BOOK SERIES "A SONG OF ICE AND FIRE"

Lab 1: Text Mining in R

Student:
Marika D'Agostini
0000760024

Period: II

Academic Year 2017/2018

Table of Contents

Introduction

A Song of Ice and Fire is a book series written by George R. R. Martin, which has also been adapted into the widely appreciated HBO television series *Game of Thrones*. The series has grown from a planned trilogy to seven volumes, the fifth and most recent of which, *A Dance with Dragons*, took Martin five years to write before its publication while the sixth novel, *The Winds of Winter*, is still being written.

The story of *A Song of Ice and Fire* takes place on the fictional continents Westeros and Essos. The point of view of each chapter in the story is a limited perspective of a range of characters growing from nine, in the first novel, to thirty-one by the fifth. Three main stories interweave a dynastic war among several families for control of Westeros, the rising threat of the supernatural Others beyond Westeros' northern border, and the ambition of Daenerys Targaryen, the deposed king's exiled daughter, to assume the Iron Throne.

This five-book series follows a very distinctive style of writing. Martin uses a limited omniscient view for each chapter: we see everything in the chapter through one person's point of view, but we kind of see it through that character's biases.

Even if, from a narrative point of view, this choice lets the reader get really close to each of the character's experiences, it could be interesting analyzing the evolution of each character singularly to better understand the character development and feelings. The aim of the project is to achieve this goal.

It starts with splitting each book in chapters using the function `split.chap()` and organizing them using the function `book.table()` in order to create a table containing for each chapter text, number, narrator and number of the book in which it is contained (Both functions have been written specifically for this project).

Then the extracted chapters are transformed to build a document-term matrix. Frequent words and associations are found from the matrix and a word cloud is used to present important words in documents. After that, using tidy text format of data, a sentiment analysis on the series is performed in order to understand feelings of readers when they approach the books.

In the end, chapters are reassembled by narrator: frequency of words and sentiments are analyzed in the same way for the four characters who narrated the largest number of chapters.

Unfortunately, the task of performing a sentiment analysis on the splitted chapters to better understand the story of the single characters was not successfully reached because of very negative values of the polarity value. Anyway, the analysis revealed some fascinating peculiarity of the G.R.R. Martin's narrative style and some useful tools that could be used also in future works.

Methodology

Importing Data

To approach the sentiment analysis of the five-book series A Song of Ice and Fire I used the tidy text format to make handling data easier and more effective. Tidy data has a specific structure since each variable is a column, each observation is a row, each type of observational unit is a table and it can be manipulated with a set of consistent tools, including packages such as `dplyr`, `tidyr` and `ggplot2`.

In order to turn the books into tidy text datasets, I needed to put them into a data frame in which each line is a chapter of the book and so, I needed to import the books in .txt format in a proper way.

Firstly, for each book, I read the *.txt* file using the function `readLines()` in the package `base` which reads some or all text lines from a connection. In this way I obtained a text file in which each line is a part of the book, not a chapter. Moreover, the file presented some non-Latin characters due to the importation of the quotation marks, extra whitespace and empty lines.

To solve these problems, I wrote a function called `clean.text()`. It allows to apply the functions `iconv()` (from the package `base`, it uses system facilities to convert a character vector between encodings) to remove all the non-Latin characters from the file, `stripWhitespace()` (from the package `tm`) to collapse all extra whitespaces to a single blank and `which()` (`base` function) to remove all the empty lines.

```
clean.text<-function(text){
  text<-iconv(text, "latin1", "ASCII", sub="")
  text<-stripWhitespace(text)
  text<-text[which(text !="")]
}
```

Secondly, I needed to split the book in chapters. To achieve this goal, I wrote the function `split.chap()` that allows to create a new .txt file for each chapter in the book given a list of book's chapter titles.

```
split.chap<-function(text, chapters){
  for(i in (2:length(chapters))){
    for(j in (1:length(text))){
      if (chapters[i]==text[j]){
        z<-j-1
        new.chapter<-text[1:z]
        write.table(new.chapter[2:length(new.chapter)],
                    file = paste(new.chapter[1], (i-1),".txt", sep=""))
        text<-text[j:length(text)]
        break
      }
    }
  }
  write.table(text[2:length(text)],
              file = paste(text[1],(length(chapters)),".txt", sep=""))}
```

Since, in the cleaned text file, the title of each chapter is written in uppercase letters in its own line, the function `split.chap()` compares each title in the list of book's chapter names with every line of the text. When a match is found, the nested function `write.table()` saves all the lines of the text before the one matched in a new *.txt* file while all the remaining lines become the text to be compared in the following loop. The name of this new file is given by the first line of this new file, so the title of the chapter, and the number of the chapter, given by the number of the compared line in the list of book's chapter names minus one. For this reason, the comparison starts from the name of the second chapter. At the end of the loops nested in the function `split.chap()`, for each chapter of the book, I get a new .txt file saved in the working directory.

Thirdly, I used the function `readtext()` to read all these files from the working directory getting a table with the name of the file in the column `doc_id` and the text of the chapter in the column `text`. Since the name of each file is composed by the title of the chapter, which is also the name of the character who narrates it, and by the number of the chapter, I used the function `book.table()` to create a data frame containing for each chapter text, number, narrator and number of the book in which it is contained.

```
book.table<-function(text,n){
  text$num<-gsub('.* ([0-9]+).txt','\\1',text$doc_id)
  text$num <- as.numeric(as.character( text$num ))
  text$narrator<-gsub("^([[:alpha:]]*).([0-9]+).txt","\\1",text$doc_id)
  text$book_n<-as.numeric(n)
  text$book_n <- as.numeric(as.character( text$book_n ))
  text<-text[,2:5]
  text<-text[order(text$num, decreasing=F),]}
```

Applying `book.table()` to the table created by `readtext()`, I got a data frame useful for further analysis thanks to the nested function `gsub()` from the package `base` which searches for matches within each element of a character vector. `gsub()` is used twice: first, it extracts the numeric part of the file name and stores it in the column `num` in order to get a column with the number of each chapter, then it extracts the alphabetic part of the file name

to create the column `narrator` with the name of the character who narrates the chapter. The argument n of `book.table()` instead, is used to create the column `book_n` in which is stored the number of the book in the series.

In this way, I got a four-column with a chapter in each row and I could use it to turn the books into tidy text dataset.

Term Frequency and Word Analysis

Regarding the overall analysis of the series, I created a data frame containing the text of the five books using the function `rbind()` to merge the five data frames obtained with `book.table()`. Then I used the function `Corpus` from the package `tm` to transform the column `text` in a corpus, which is a collection of documents containing (natural language) text, and then I cleaned this corpus using the function `clean.corpus()`.

```
clean.corpus <- function(corpus){
  corpus <- tm_map(corpus, removeNumbers)
  corpus <- tm_map(corpus, tolower)
  corpus <- tm_map(corpus, removeWords, c("x","dont"))
  corpus <- tm_map(corpus, removeWords, c(stopwords("en")))
  corpus <- tm_map(corpus, removeWords, stop_words$word)
  corpus <- tm_map(corpus, removePunctuation,
                           preserve_intra_word_dashes=T)
  corpus <- tm_map(corpus, stripWhitespace)
  return(corpus)}
```

`clean.corpus()` is made by several transformation functions applied to the corpus by the interface `tm_map()` presented in the package `tm`. These nested functions return a new lowercase corpus without numbers, "stop words" (words in a document that occur many times but may not be important; in English, these are probably words like "the", "is", "of", and so forth) from the packages `tokenizers` and `tm`, punctuation and extra whitespaces.

In order to analyze term frequency of the series and create a bar chart of the twenty most used words using `ggplot()`, the cleaned corpus was transformed in a term-document matrix using the function `TermDocumentMatrix()` from the package `tm`. A term-document matrix (TDM) is a sparse matrix describing the number of appearances of a term in that document: each row of the matrix represents a term while each column represents a chapter.

The word analysis by book was performed creating a new corpus with documents the single books instead of the single chapters. The chapters of each book were collapsed together using the function `as.vector()`. `VCorpus(VectorSource())` were then applied to these five vectors to obtain a new corpus that can be transformed in a TDM in order to visualize the results in a word cloud using the function `comparison.cloud()` from the package `wordcloud`.

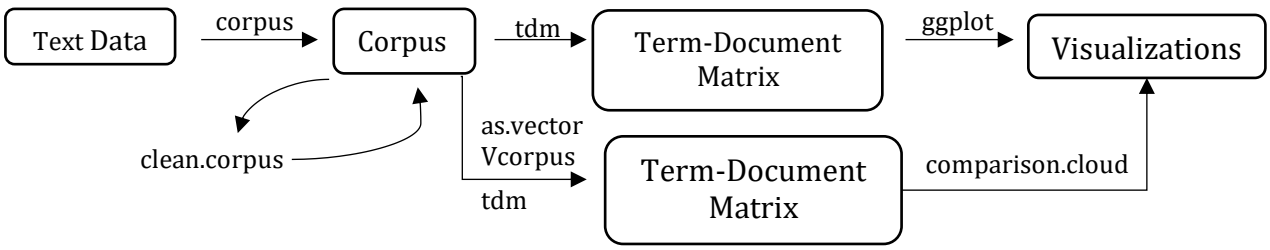

Figure 2.1: A flowchart of the term frequency and word analysis

Sentiment Analysis

The way I chose to perform a sentiment analysis of the series , was to consider the text as a combination of its individual words and the sentiment content of the whole text as the sum of the sentiment content of the individual words, in order to take advantage of the tidy tool ecosystem. The `tidytext` package, in the `sentiments` dataset, contains several sentiment lexicons for evaluating the emotion of a text.

The three general-purpose lexicons based on unigrams (i.e., single words) are

- `AFINN`: from Finn Årup Nielsen, assigns words with a score that runs between -5 and 5, with negative scores indicating negative sentiment and positive scores indicating positive sentiment.
- `bing:` from Bing Liu and collaborators, categorizes words in a binary fashion into positive and negative categories.
- `nrc:` from Saif Mohammad and Peter Turney, categorizes words in a binary fashion into categories of positive, negative, anger, anticipation, disgust, fear, joy, sadness, surprise, and trust.

First of all, in order to use tidy text format of data, the cleaned courpus was transformed in a document-term matrix using the function `DocumentTermMatrix()` from the package `tm.` A document-term matrix is a sparse matrix describing a collection (i.e., a corpus) of documents with one row for each document (in this case for each chapter) and one column for each term. Each value of the matrix contains the number of appearances of that term in that document.

DTM objects cannot be used directly with tidy tools but the `broom` package provides a verb that converts between the two formats`. tidy()` turns a non-tidy document-term matrix into a tidy three-column data frame with disposable row names and with variables `document`, `term`, and `count`.

With data in a tidy format with one word per row, sentiment analysis can be done as an inner join. Foremost, I created a subset of joy words using the NRC lexicon and then, I used

`inner_join()` to perform the sentiment analysis: in this way I could see how many joy words there are in the whole series.

Regarding the changes of the sentiment throughout the series, I used some `dplyr` functions to analyze it. Firstly, I found a sentiment score for each word using the `bing` lexicon and `inner_join()` and then counted up how many positive and negative words there are in each chapter of each book. Secondly, I used `spread()` so that we have negative and positive sentiment in separate columns, lastly I calculated a net sentiment (positive - negative) and plotted it using `ggplot()`.

Another advantage of having the data frame with both sentiment and word is that we can analyse word counts that contribute to each sentiment. By implementing `count()` with arguments of both word and sentiment, I found out how much each word contributed to each sentiment.

At the end, I analyzed and plotted the emotions in each chapter of the series using `classify_emotion()` from the package `sentiment` which is a function that classifies the emotion (e.g. anger, disgust, fear, joy, sadness, surprise) of a set of texts using a naive Bayes classifier trained on Carlo Strapparava and Alessandro Valitutti's `emotions` lexicon.

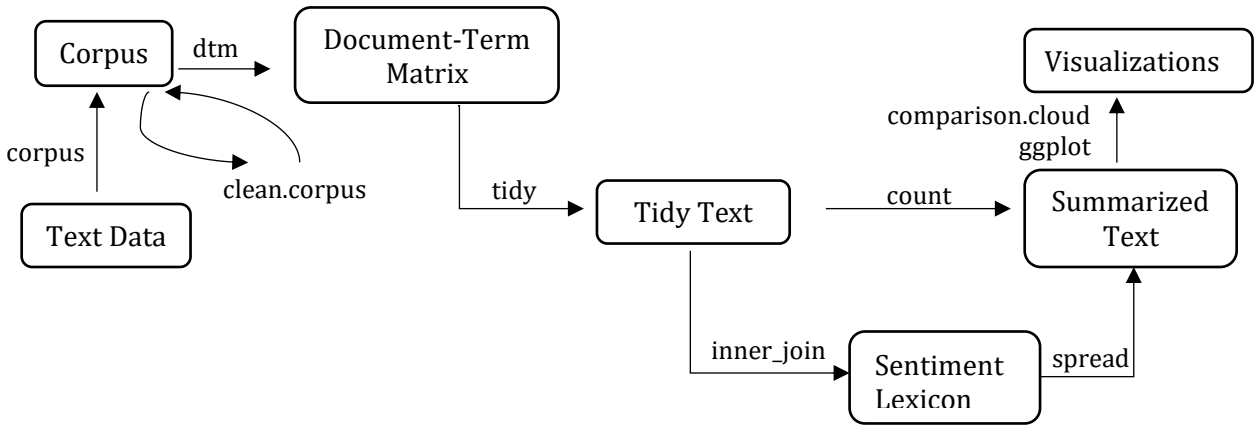

Figure 2.2: A flowchart of the sentiment analysis

For the analysis of the four main characters' story, the corpus for each character was created merging together the chapters in the outputs of the function `book.table()` having as narrator the name of the character under analysis. After that, I followed the same steps I followed for the overall analysis of the series.

Data Set Description

I downloaded from the website https://archive.org/download/A_Game_of_Thrones_Series both the *.pdf* and *.epub* files of each book. EPUB is an e-book file format that can be downloaded and read on devices like smartphones, tablets, computers, or e-readers but it is not suitable for R analysis. For this reason, I used the website https://www.epubconverter.com/epub-to-txt-converter/ in order to convert the five *.epub* files in *.txt* files.

A summary of the five books/files is given by table 3.1.

Table 3.1

#	TITLE	PAGES*	CHAPTERS	WORDS*	NAME IN THE SCRIPT	DIMENSION FILE .TXT
1	A Game of Thrones	694	73	292.727	got1	1,57 MB
2	A Clash of Kings	768	70	318.903	got2	1,71 MB
3	A Storm of Swords	973	82	414.604	got3	2,24 MB
4	A Feast for Crows	753	46	295.032	got4	1,58 MB
5	A Dance with Dragons	1040	73	414.788	got5	2,23 MB
	TOTAL	4.228	344	1.736.054	asoiaf	9,33 MB

* http://www.arbookfind.com/

After having converted the books in *.txt* format, I imported them in R using the function `readLines()` and cleaned them using the function `clean.book()`. Table 3.2 gives an overview of the dimensions of the files before and after having removed non-Latin characters, extra whitespaces and empty lines.

Table 3.2

#	NAME OF THE FILE	ELEMENTS BEFORE CLEANING	DIMENSION BEFORE CLEANING	ELEMENTS AFTER CLEANING	DIMENSION AFTER CLEANING
1	got1	28.365	3,1 MB	20.075	3 MB
2	got2	26.651	2,7 MB	17.816	2,6 MB
3	got3	43.866	4,7 MB	30.790	4,5 MB
4	got4	28.550	3,1 MB	20.482	3 MB
5	got5	37.500	4,1 MB	26.753	4 MB
	TOTAL	164.932	17,7 MB	115.916	17,1 MB

The cleaned texts were splitted in chapters using the function `split.chapter()`, then these chapters were re-read in R and organized in a table with a single chapter per row with `book.table()`. After that, I performed an analysis regarding the chapters' narrators of the books (figure 3.1).

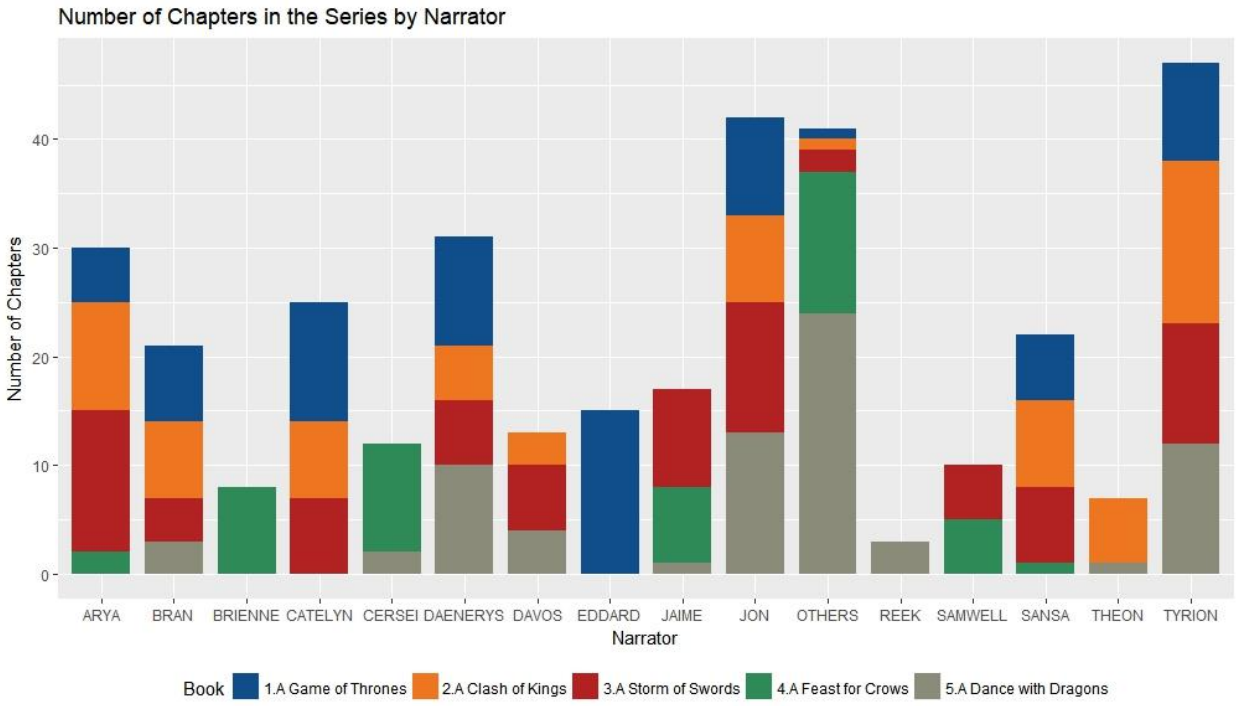

Figure 3.1

A Corpus of 344 elements was created from the `text` column of the tables obtained with `book.table()`. The TDM matrix associated to this corpus had 24.762 elements and was 94% sparse.

```
> inspect(tdm_asoiaf)
<<TermDocumentMatrix (terms: 24762, documents: 344)>>
Non-/sparse entries: 529604/7988524
Sparsity           : 94%
Maximal term length: 53
Weighting          : term frequency (tf)
Sample             :
      Docs
Terms  204 246 250 253 255 267 298 299  74  85
  and  288 218 178 234 290 318 267 279 310 267
  had   88  79 115 119  92 101  82  68 124 100
  her   83 130 135  47  51 163 210  75  71  45
  his  165  87 107 157 166  89 100 140 226 175
  she   35 136 113  15  17 179 142  46  26  27
  that  58  72  91  86  43  96  63  83  82  62
  the  523 433 378 471 575 492 499 589 594 547
  was  113 135  97 101  82 150 108  96 119  93
  with  77  64  63  70  74  70  69  81  87  73
  you   49 100 102  74  50 134  83  96 100  87
```

To reduce the dimension of the matrix and remove the words in the document that occur many times but may not be important, I applied the function `clean.corpus()` to the corpus. The associated TDM became a little bit sparser but the number of terms in it decreased to 23.714.

```
> inspect(tdm_asoiaf)
<<TermDocumentMatrix (terms: 23714, documents: 344)>>
Non-/sparse entries: 359051/7798565
Sparsity           : 96%
Maximal term length: 53
Weighting          : term frequency (tf)
Sample             :
        Docs
Terms    204 250 253 255 267 298 299 74 85 97
  black   11  11   5   6   4  12  14 16 18 22
  eyes    16   6   7   8  16   7  10 17 15 12
  hand    12   8  19  18  18  11  13 16  7  6
  jon      0   1   4   0   9   0   0  3  2 81
  king    27  28   4  13   3   7   0 19 11 10
  lord    44  23  23  37  66  12   7 92 43 35
  ser     38  60  88   6  30   0  30 14  2 10
  time    11   9   6   7  13  16  10  9  9  7
  told     6  17  16  11  28   5   7 20 16 15
  tyrion  71   1   1   0   1   0  73  0  0  0
```

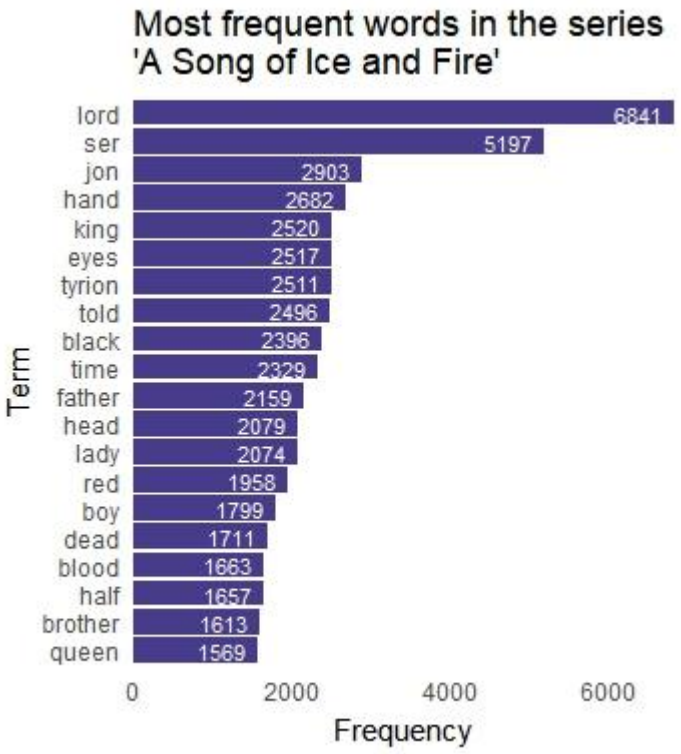

Figure 3.2

Figure 3.3

I used this TDM in order to perform a frequency analysis of the series. As we can see from figure 3.2, the two more nominated characters, Jon and Tyrion, are also the characters who narrated the most number of chapters in the series.

Moreover, we can observe that some of the most common words ("lord", "ser", "king", "lady" and "queen") are coherent with the medieval setting of the story. Words like "black", "death" and "blood" instead, give us a first idea of how the sentiment of the series could be and also reflect the substantial death rates in medieval wars.

Using the function `comparison.cloud()` from the package `wordcloud`, I created also a word cloud of the common words between the books (figure 3.3). We can deduce that the narration is more centered around key story concepts since the words presented in the figure can summarize the plot of each book.

After having performed an overall analysis of the series, I focused on the stories of Arya Stark, Daenerys Targaryen, Jon Snow and Tyrion Lannister since they are the characters who narrated the most number of chapters in the series.
A summary of the characteristics of the four corpora used is given by table 3.3.

Table 3.3	ARYA	DAENERYS	JON	TYRION
CORPUS DIMENSION	30	31	42	47
TERMS BEFORE CLEANING	8.166	9.512	10.161	12.605
NON-/SPARSE ENTRIES BEFORE CLEANING	40.483/ 20.4497	49.185/ 24.5687	62.680/ 36.4082	74.269/ 51.8166
SPARSITY BEFORE CLEANING	83%	83%	85%	87%
TERMS AFTER CLEANING	7.343	8.595	9.286	11.596
NON-/SPARSE ENTRIES AFTER CLEANING	27.367/ 19.2923	33.558/ 23.2887	42.434/ 34.7578	50.979/ 49.4033
SPARSITY AFTER CLEANING	88%	87%	89%	91%

From the graphs of the most common words in the stories of the four main characters, we can see that the terms are strongly related with the plot of the series: we find "Gendry", "sword" and "Hound" in Arya's frequent words; "Jorah", "dragons" , "Khal" and "dothraki" in Daenerys'; "snow", "wall", "watch" and "ghost" in Jon's; "dwarf", "Cersei", "Lannister" and "Bronn" in Tyrion's. Moreover, the most frequent word is always the name of the character, due to the narration in third person of the chapters. We can notice that the most common terms of figure 3.2 are also presented in figures 3.4, 3.6, 3.8, 3.10.

Regarding the word clouds, used in a reduced number of chapters, they are more able to capture the plot of the narration and so, in our case, the stories of the four characters as we can see in figure 3.5, 3.7, 3.9, 3.11.

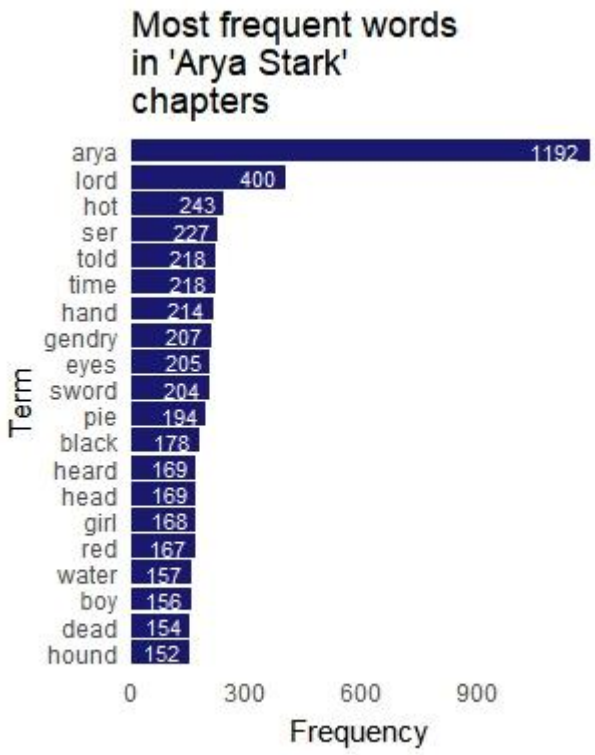

Figure 3.4

Figure 3.5

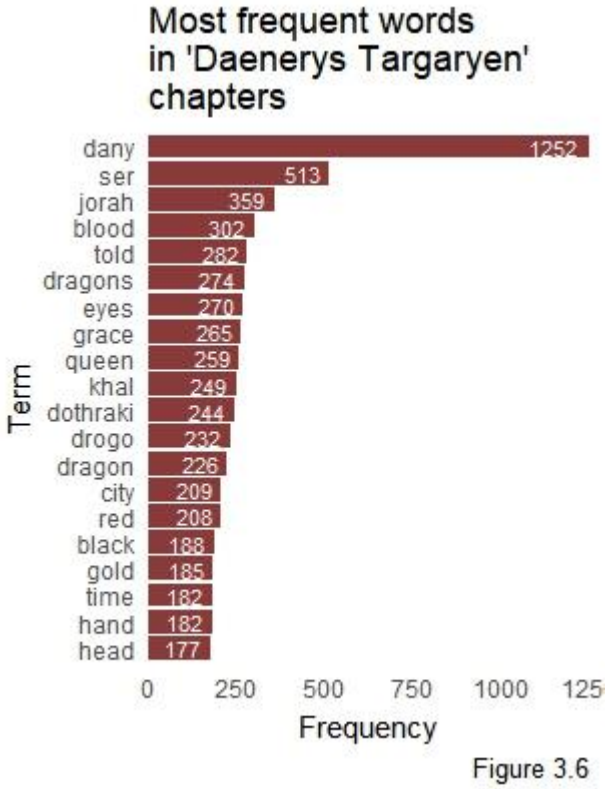

Figure 3.6

Figure 3.7

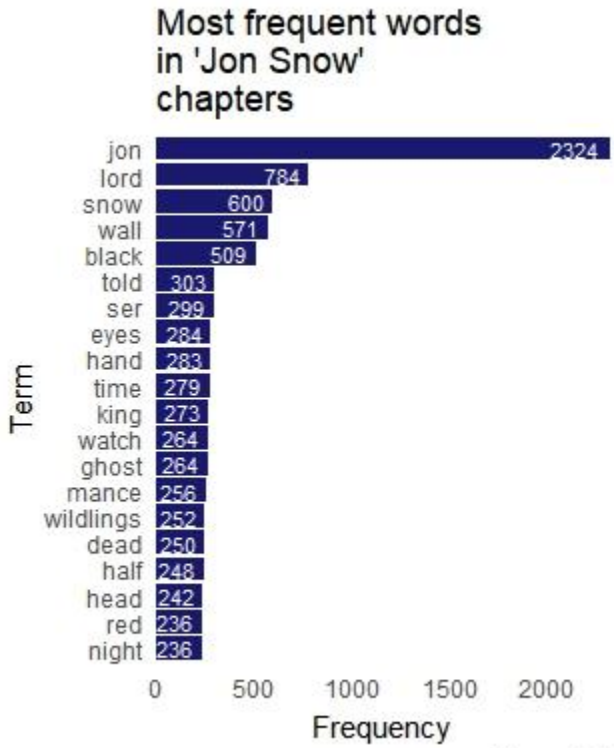

Figure 3.8

Figure 3.9

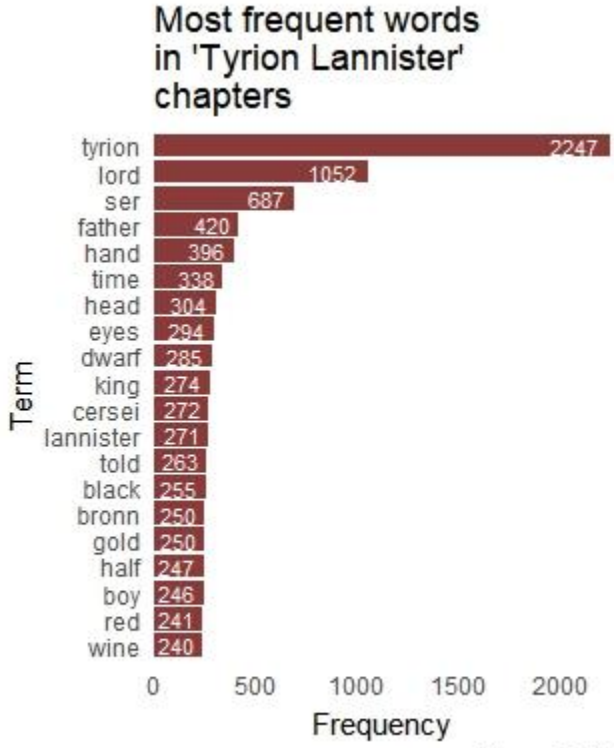

Figure 3.10

Figure 3.11

Results and Discussion

A Song of Ice and Fire

One task in this sentiment analysis was classifying the polarity of the of the five-book series *A Song of Ice and Fire*.

In order to examine how sentiment changes throughout each chapter of the series, I found a sentiment score for each word using the `bing` lexicon and `inner_join()`. Next, I counted up how many positive and negative words there are in each chapter of each book. Then, I used `spread()` to have negative and positive sentiment in separate columns, and lastly I calculated a net sentiment (positive - negative). This value is called polarity and it is represented on the y-axis of the chart in Figure 4.1. The x-axis represents instead the chapters and so the narrative time.

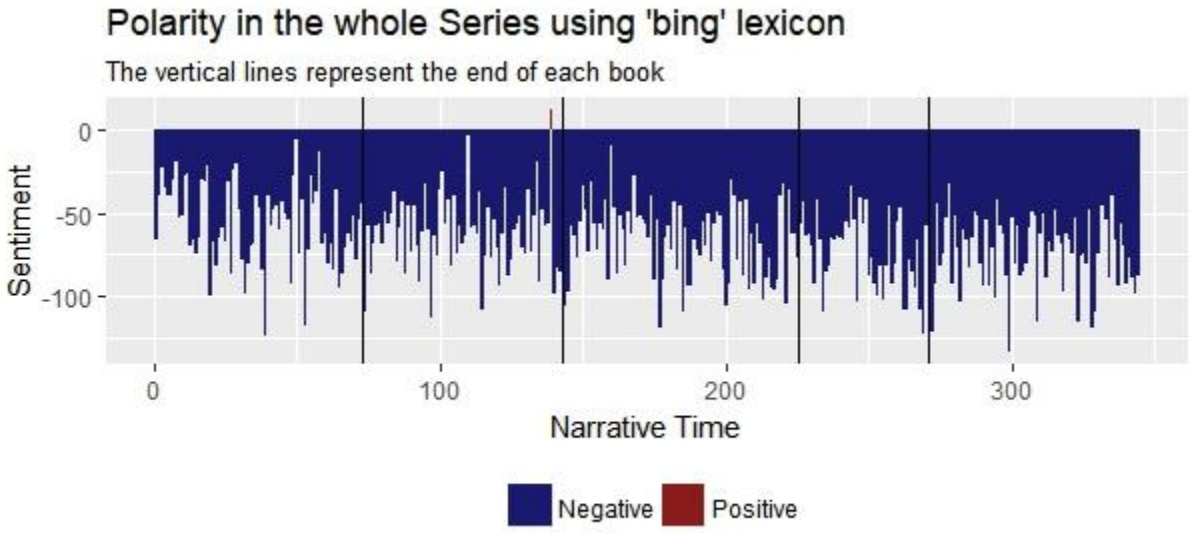

Figure 4.1

We can see in Figure 4.1 how the polarity of the series is always negative over the whole trajectory of the story, except for chapter 66 of *A Clash of Kings* which has score 12. This is an expected result since the books are known for complex characters, sudden and often violent plot twists, and political intrigue. Anyway, it is quite impressive that the polarity value it is always so low reaching negative picks of -134.

Another way to understand how dark the books written by G.R.R. Martin are, is looking at the quantity of positive and negative words classified. Using the `bing` lexicon I found 2225 different positive words but only 949 different positive words. The most used negative and positive words are represented in Figure 4.2 and 4.3.

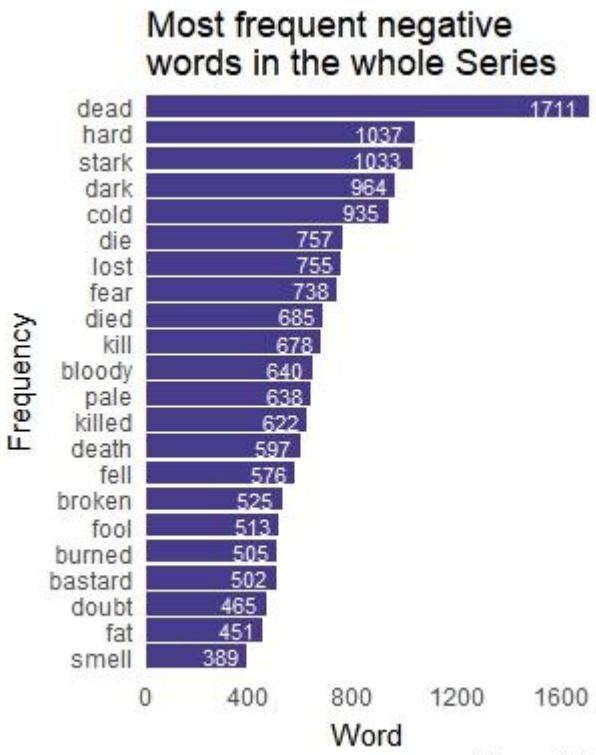

Figure 4.2

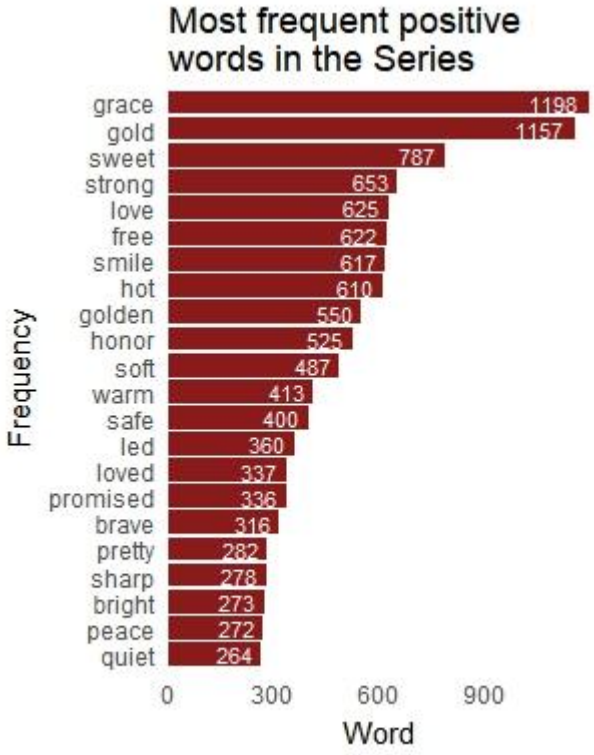

Figure 4.3

An unexpected result was instead the outcome of the emotion classification. I used `classify_emotion` from the package `sentiment` in order to categorize the words of the books according to the data set `emotion` which is a dataset containing approximately 1500 words classified into six emotion categories: anger, disgust, fear, joy, sadness, and surprise.

The categorization of the words is represented in Figure 4.4. The x-axis represents the chapters while the y-axis represents the relative frequency of each emotion in the chapter. From the bar chart, we can see that the predominant emotion is joy rather than sadness or anger, as I would expected given the previous results.

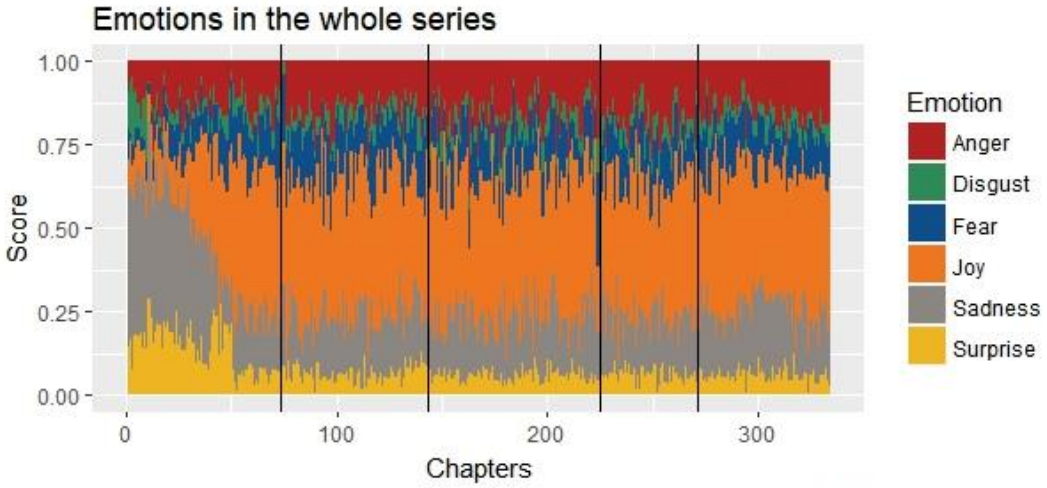

Figure 4.4

Arya Stark

The results of the analysis on the chapters regarding Arya Stark reflect the ones about the whole series. The polarity of this part of the story fluctuates between -19 and -94 for all the 31 chapters (Figure 4.5).

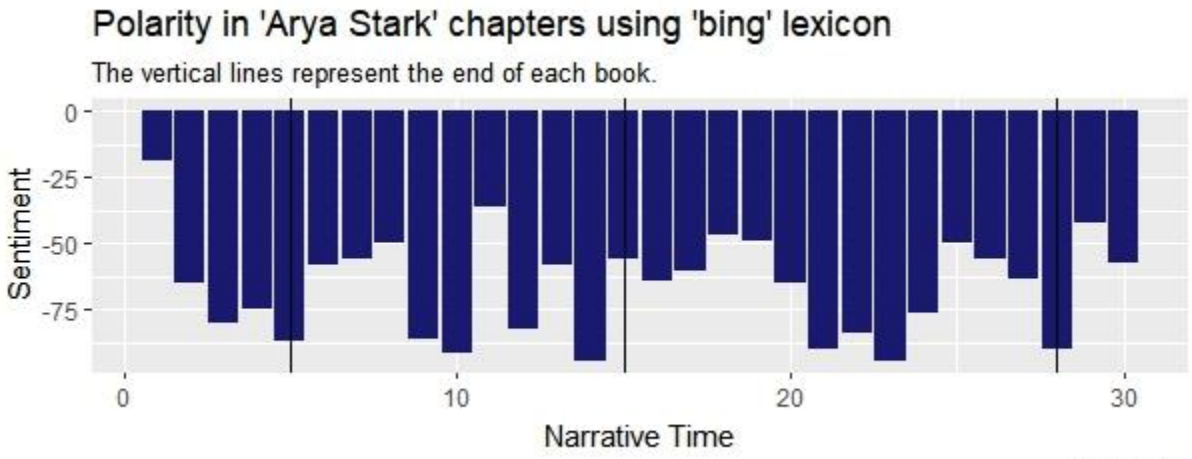

Figure 4.5

Moreover, the negative words are more than double the positive words: 808 different negative words and only 295 different positive words are used in these chapters.

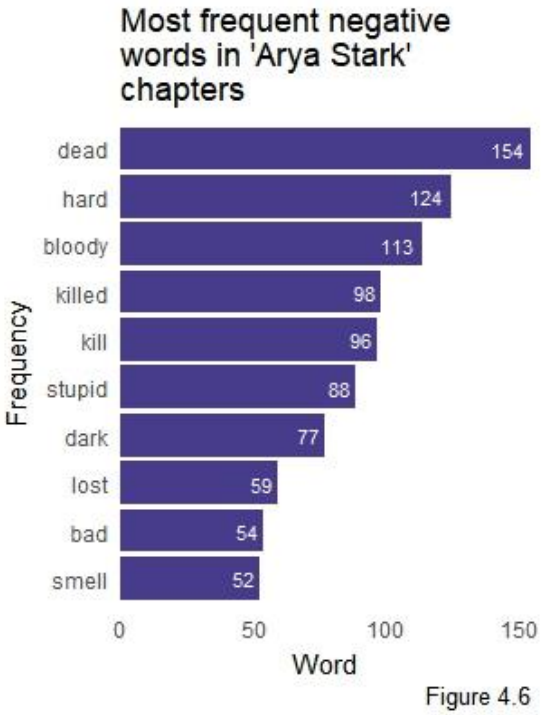

Figure 4.6

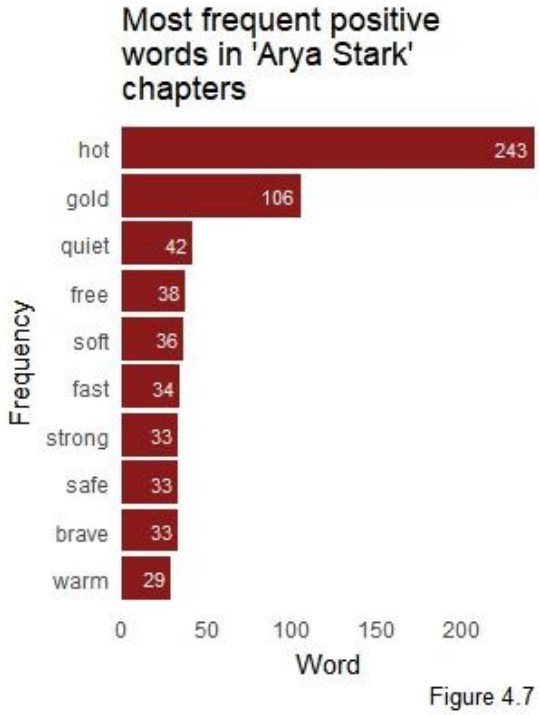

Figure 4.7

Daenerys Targaryen

The trend of the polarity over the chapters about Daenerys Targaryen is more similar to the one of the whole series where the fifth book, *A Dance with Dragons*, seems to be the most negative one. (Figure 4.8).

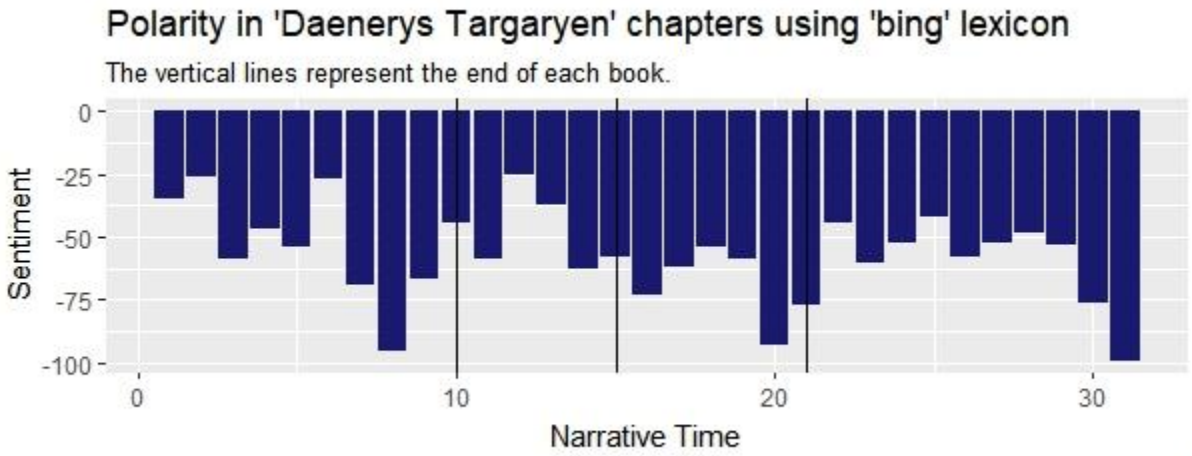

Figure 4.8

In this part of the story, G.R.R. Martin used 490 different positive words and 1024 different negative words. “slaves” and “slave” stand out between the most used negative words (Figure 4.9). This is due to the fact that Daenerys’ mission in the series is to free all of the slaves in the region of Meereen.

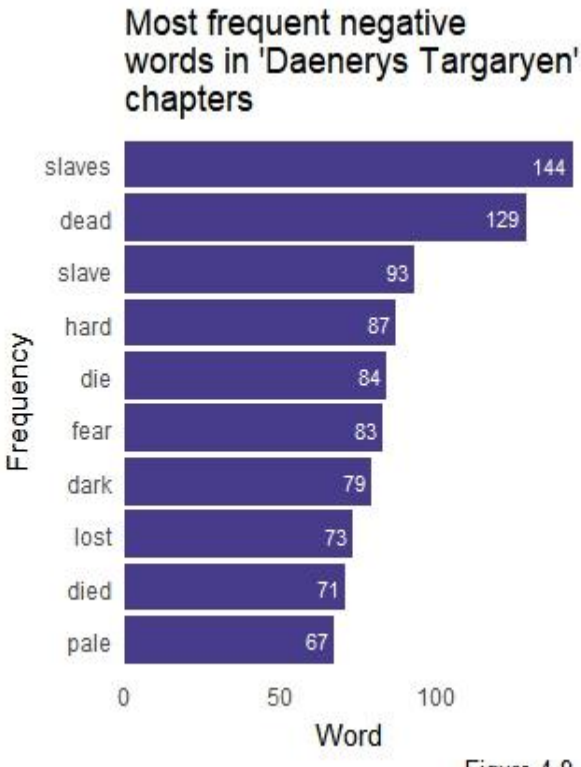

Figure 4.9

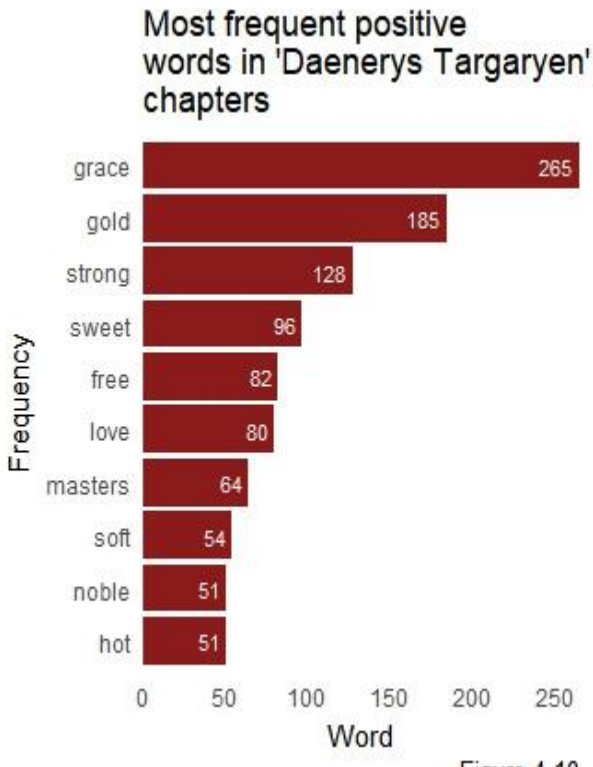

Figure 4.10

Jon Snow

The polarity of Jon Snow's story (Figure 4.11), even if it is always below -27, has a more negative trend between the end of the third book and the beginning of the fifth one (there are not chapters about Jon Snow in *A Feast for Crows*) .

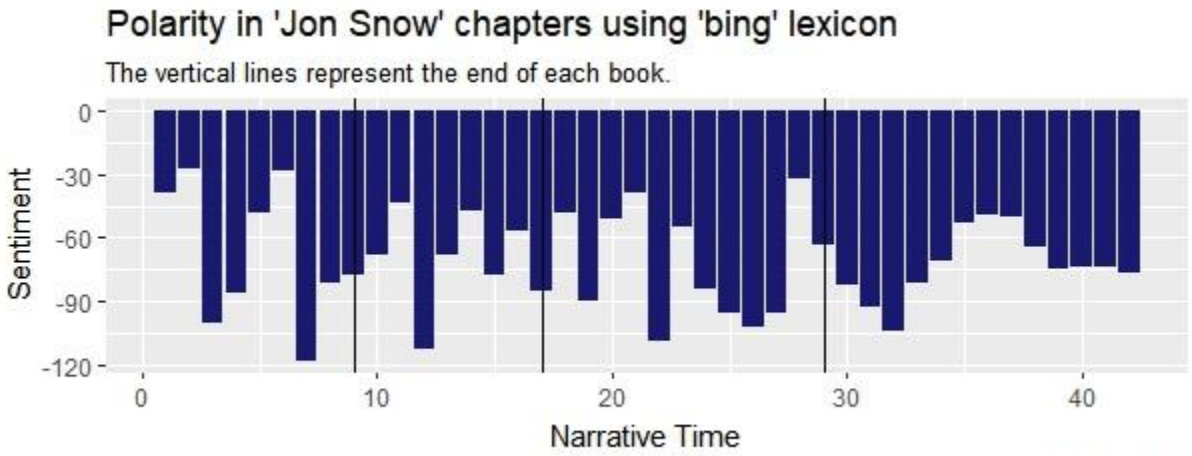

Figure 4.11

Between the 1106 different negative words presented in Jon Snow's chapters, two of the most used are "bastard" and "stark" (Figure 4.12). This happens because from infancy, and so from the beginning of the series, Jon is presented as the bastard son of Lord Eddard Stark, and raised by Eddard alongside his lawful children.

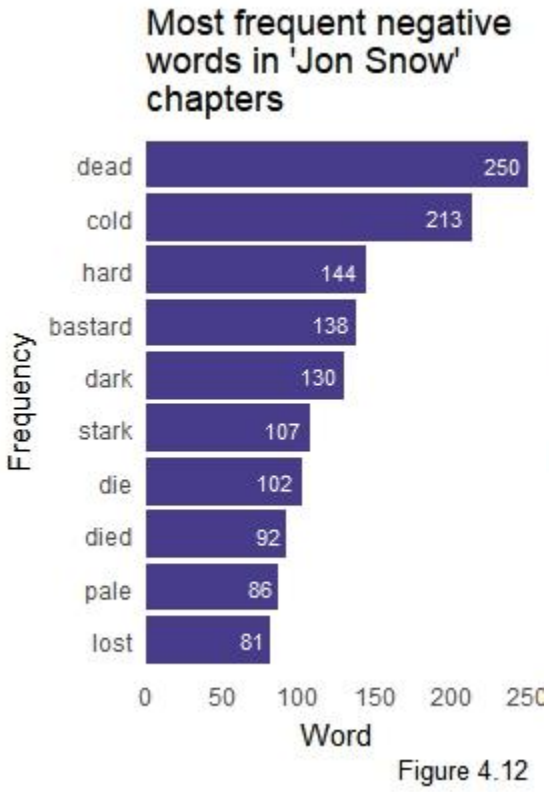

Figure 4.12

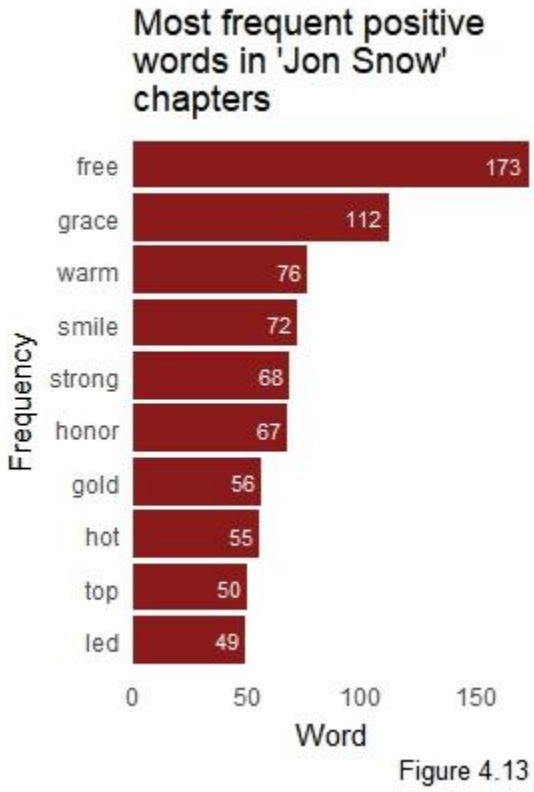

Figure 4.13

Tyrion Lannister

Also in this case the polarity is constantly negative (Figure 4.14). The second book seems to be the "more positive" one. Probably this is because Tyrion, in *A Clash of Kings*, spends most of his time in King's Landing, which is the capital and richest city of the Seven Kingdoms, and, in order to describe the city, G.R.R. Martin makes a huge use of the words "gold" and "golden" which are between the most used positive words (Figure 4.16).

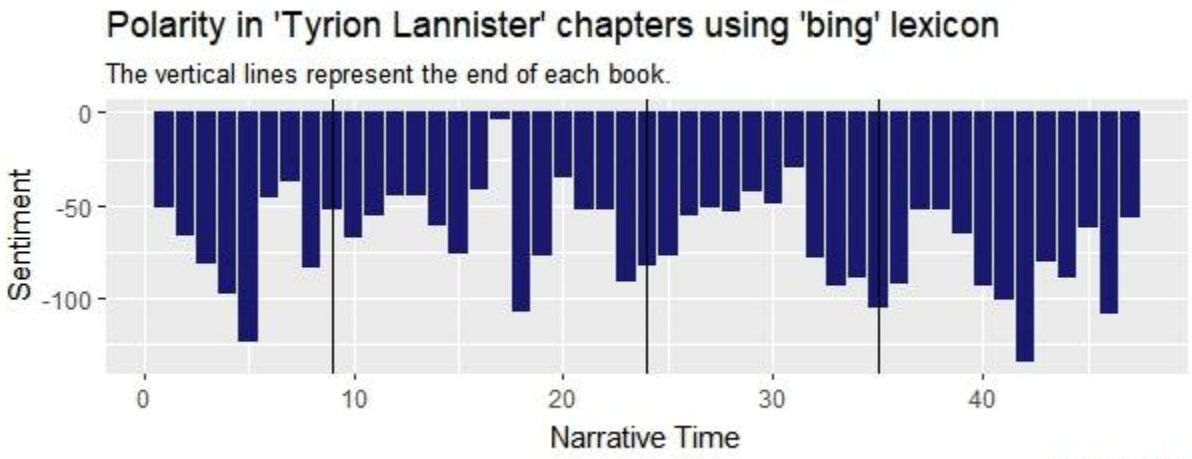

Figure 4.14

In Tyrion Lannister's chapters have been used 1441 different negative words and 622 different positive words.

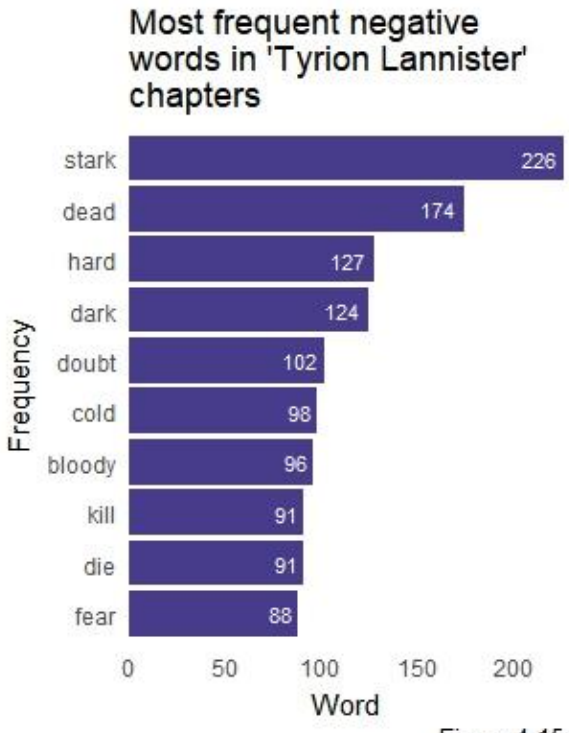

Figure 4.15

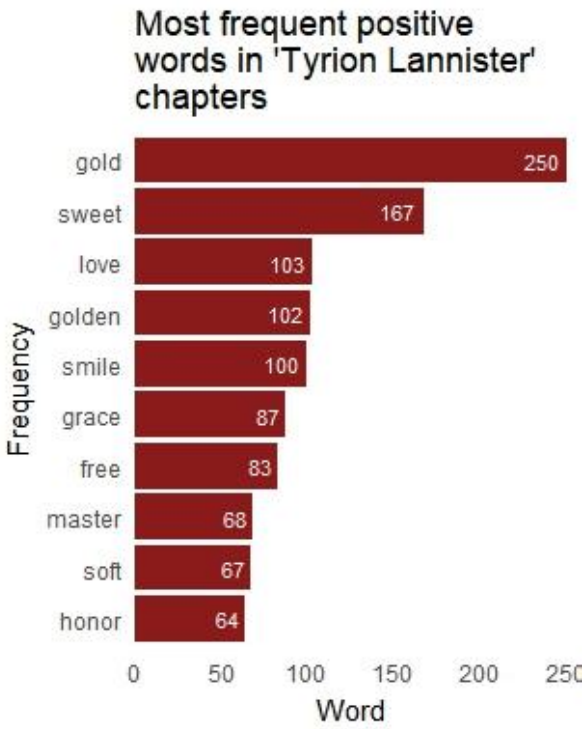

Figure 4.16

Summary

The task of this project was dividing the series *A Song of Ice and Fire* according to the narrators of each chapter, in order to perform a sentiment analysis with the objective of understand the story of a character using polarity value. Even if the goal was not successfully reached, the analysis revealed some fascinating peculiarity of the G.R.R. Martin's narrative style and some tools that could be used also for future works.

Interesting results came out from the analysis of the most frequent negative and positive words used in the chapters narrated by the main characters.

In all the four parts, the number of negative words used is more than double the positive words. Moreover, in the negative words "dead", "die", and "died" are always presented, underling how violent and vicious the series is and validating the warning that Cersei Lannister address to Ned Stark in the first book: "When you play the game of thrones, you win or you die. There is no middle ground."

Regarding the positive words, they are mostly related to the settings of the events as "warm", "hot" and "soft". "grace" instead, is another word related to the medieval time of the series since the style 'Your Grace' is used to address the King or Queen.

Although fantasy comes from an imaginative realm, G.R.R. Martin reflects the real world where people die sometimes ugly deaths, even beloved people. Main characters are killed off so that the reader will not expect the supposed hero to survive, and instead will feel the same tension and fear that the characters might. This came out also from the quite impressive result of the polarity analysis of the series, where the value of the polarity is always negative over the whole trajectory of the story, reaching negative picks of -134.

Consequently, also the analysis of the stories of the main characters was affected by this problem. I chose then to analyse the stories of Arya Stark, Tyrion Lannister, Jon Snow, and Daenerys Targaryen since, being the characters who narrate the most number of chapters, they generate the most feedback from readers. The results are not particularly interesting, also because the words with a negative score are often referred to events that happen around the character and not to the character itself. For this reason, it is very difficult extract and understand the story of the single character using the polarity (we should think about a different approach to achieve this goal) but, we can instead interpret this value as the perception of the readers about his story.

Moreover, the functions written specifically for this project can be used in future works. `split.chap()` can split a book in chapters given just the table of content and `book.table()`creates a table containing for each chapter text, number, narrator and number of the book in which it is contained. Both functions perform tasks of text mining that could be useful in the future analysis of books, particularly of book series, for this reason they could be considered a positive outcome of this project.

References

- https://archive.org/details/ASongOfIceAndFire5bookspdf
- https://www.epubconverter.com/epub-to-txt-converter/
- http://www.arbookfind.com/
- http://moodle.labmasters.pl/course/view.php?id=20
- https://stackoverflow.com/questions/
- https://www.tidytextmining.com/tidytext.html
- http://www.cookbook-r.com/
- http://awoiaf.westeros.org/index.php/Main_Page
- https://en.wikipedia.org/wiki/A_Song_of_Ice_and_Fire

Appendix

Packages

```
library(tm)
library(ggplot2)
library(qdap)
library(tidytext)
library(tidyr)
library(dplyr)
library(broom)
library(Rstem)
library(sentiment)
library(openNLP)
library(igraph)
library(stringr)
library(wordcloud)
library(RDRPOSTagger)
library(tokenizers)
library(readtext)
```

R Code

```
##### Split Files into Chapters #####

### Book1 ###

setwd('C:/Users/Amministratore/Desktop/Text Mining/Project/Data')
got1<-readLines("1.AGameOfThrones-GeorgeR.R.Martin.txt")
#View(got1)

clean.text<-function(text){
  text<-iconv(text, "latin1", "ASCII", sub="")
  text<-stripWhitespace(text)
  text<-text[which(text !="")]
}

got1<-clean.text(got1)
chap1<-readLines('Chapters1.txt')


split.chap<-function(text, chapters){
  for(i in (2:length(chapters))){
    for(j in (1:length(text))){
      if (chapters[i]==text[j]){
        z<-j-1
        new.chapter<-text[1:z]
        write.table(new.chapter[2:length(new.chapter)],
                    file = paste(new.chapter[1], (i-1),".txt", sep=""))
        text<-text[j:length(text)]
        break
      }
    }
  }
  write.table(text[2:length(text)], file =
paste(text[1],(length(chapters)),".txt", sep=""))
}

setwd('C:/Users/Amministratore/Desktop/Text Mining/Project/Splitted
Chapters/1')
split.chap(text = got1, chapters = chap1)

### Book2 ###

setwd('C:/Users/Amministratore/Desktop/Text Mining/Project/Data')
got2<-readLines("2.AClashOfKings-GeorgeR.R.Martin.txt")
got2<-clean.text(got2)
#View(got2)

chap2<-readLines('Chapters2.txt')
setwd('C:/Users/Amministratore/Desktop/Text Mining/Project/Splitted
Chapters/2')
```

```
split.chap(got2, chap2)

### Book3 ###
setwd('C:/Users/Amministratore/Desktop/Text Mining/Project/Data')
got3<-readLines("3.AStormOfSwords-GeorgeR.R.Martin.txt")
got3<-clean.text(got3)
#View(got3)

chap3<-readLines('Chapters3.txt')
setwd('C:/Users/Amministratore/Desktop/Text Mining/Project/Splitted
Chapters/3')
split.chap(got3, chap3)

### Book4 ###
setwd('C:/Users/Amministratore/Desktop/Text Mining/Project/Data')
got4<-readLines("4.AFeastForCrows-GeorgeR.R.Martin.txt")
got4<-clean.text(got4)
#View(got4)

chap4<-readLines('Chapters4.txt')
setwd('C:/Users/Amministratore/Desktop/Text Mining/Project/Splitted
Chapters/4')
split.chap(got4, chap4)

### Book5 ###
setwd('C:/Users/Amministratore/Desktop/Text Mining/Project/Data')
got5<-readLines("5.ADanceWithDragons-GeorgeR.R.Martin.txt")
got5<-clean.text(got5)
#View(got5)

chap5<-readLines('Chapters5.txt')
setwd('C:/Users/Amministratore/Desktop/Text Mining/Project/Splitted
Chapters/5')
split.chap(got5, chap5)
##### Import Books ####
text1<-readtext("C:/Users/Amministratore/Desktop/Text
Mining/Project/Splitted Chapters/1/*.txt")

book.table<-function(text,n){
  text$num<-gsub('.* ([0-9]+).txt','\\1',text$doc_id)
  text$num <- as.numeric(as.character( text$num ))
  text$narrator<-gsub("^([[:alpha:]]*).([0-9]+).txt", "\\1",
                      text$doc_id)
  text$book_n<-as.numeric(n)
  text$book_n <- as.numeric(as.character( text$book_n ))
  text<-text[,2:5]
  text<-text[order(text$num, decreasing=F),]
}

text1<-book.table(text1,1)
```

```
text2<-readtext("C:/Users/Amministratore/Desktop/Text
Mining/Project/Splitted Chapters/2/*.txt")
text2<-book.table(text2,2)

text3<-readtext("C:/Users/Amministratore/Desktop/Text
Mining/Project/Splitted Chapters/3/*.txt")
text3<-book.table(text3,3)

text4<-readtext("C:/Users/Amministratore/Desktop/Text
Mining/Project/Splitted Chapters/4/*.txt")
text4<-book.table(text4,4)

text5<-readtext("C:/Users/Amministratore/Desktop/Text
Mining/Project/Splitted Chapters/5/*.txt")
text5<-book.table(text5,5)


##### Chapters Analysis ####

sum1<-cbind(as.matrix(count(text1, text1$narrator)),matrix(rep("1.A
Game of Thrones",9),9,1))
sum1[7,1]<-"OTHERS"
sum2<-cbind(as.matrix(count(text2, text2$narrator)),matrix(rep("2.A
Clash of Kings",10),10,1))
sum2[7,1]<-"OTHERS"
sum3<-cbind(as.matrix(count(text3, text3$narrator)),matrix(rep("3.A
Storm of Swords",12),12,1))
sum3[c(6,9),1]<-"OTHERS"
sum4<-cbind(as.matrix(count(text4, text4$narrator)),matrix(rep("4.A
Feast for Crows",18),18,1))
sum4[c(1,4,7,10:18),1]<-"OTHERS"
sum5<-cbind(as.matrix(count(text5, text5$narrator)),matrix(rep("5.A
Dance with Dragons",33),33,1))
sum5[c(1,6,9:10,12:30,33),1]<-"OTHERS"

sum<-as.data.frame(rbind(sum1,sum2,sum3,sum4,sum5))
sum[, 2] <- as.numeric(as.character( sum[, 2] ))
sum[, 1] <- as.character( sum[, 1] )

ggplot(data=sum, aes(x=sum$`text1$narrator`, y=n, fill=V3)) +
  geom_bar(na.rm=TRUE, position="stack", width=0.8, stat="identity")+
  labs(x="Narrator", y="Number of Chapters", caption="Figure 3.1")+
  ggtitle("Number of Chapters in the Series by Narrator ")+
  labs(fill="Book") +
  theme(legend.position = 'bottom')+
  scale_fill_manual(values=c("dodgerblue4", "chocolate2",
                             "firebrick", "seagreen", "lightyellow4"))
```

```
##### SERIES ANALYSIS #####


##### Overall Frequency #####
asoiaf<-as.data.frame(rbind(text1,text2,text3,text4,text5))
asoiafCorpus <- Corpus(VectorSource(asoiaf$text))
tdm_asoiaf <- TermDocumentMatrix(asoiafCorpus)
inspect(tdm_asoiaf)

## <<TermDocumentMatrix (terms: 24761, documents: 344)>>
## Non-/sparse entries: 529598/7988186
## Sparsity           : 94%
## Maximal term length: 53
## Weighting          : term frequency (tf)
## Sample             :
##       Docs
## Terms  204 246 250 253 255 267 298 299  74  85
##   and  288 218 178 234 290 318 267 279 310 267
##   had   88  79 115 119  92 101  82  68 124 100
##   her   83 130 135  47  51 163 210  75  71  45
##   his  165  87 107 157 166  89 100 140 226 175
##   she   35 136 113  15  17 179 142  46  26  27
##   that  58  72  91  86  43  96  63  83  82  62
##   the  523 433 378 471 575 492 499 589 594 547
##   was  113 135  97 101  82 150 108  96 119  93
##   with  77  64  63  70  74  70  69  81  87  73
##   you   49 100 102  74  50 134  83  96 100  87

clean.corpus <- function(corpus){
  corpus <- tm_map(corpus, removeNumbers)
  corpus <- tm_map(corpus, tolower)
  corpus <- tm_map(corpus, removeWords, c("x","dont"))
  corpus <- tm_map(corpus, removeWords, c(stopwords("en")))
  corpus <- tm_map(corpus, removeWords, stop_words$word)
  corpus <- tm_map(corpus, removePunctuation,
preserve_intra_word_dashes=T)
  corpus <- tm_map(corpus, stripWhitespace)
  return(corpus)
}

asoiafCorpus<-clean.corpus(asoiafCorpus)

tdm_asoiaf <- TermDocumentMatrix(asoiafCorpus)
inspect(tdm_asoiaf)
```

```
## <<TermDocumentMatrix (terms: 23713, documents: 344)>>
## Non-/sparse entries: 359049/7798223
## Sparsity           : 96%
## Maximal term length: 53
## Weighting          : term frequency (tf)
## Sample             :
##         Docs
## Terms    204 250 253 255 267 298 299 74 85 97
##   black   11  11   5   6   4  12  14 16 18 22
##   eyes    16   6   7   8  16   7  10 17 15 12
##   hand    12   8  19  18  18  11  13 16  7  6
##   jon      0   1   4   0   9   0   0  3  2 81
##   king    27  28   4  13   3   7   0 19 11 10
##   lord    44  23  23  37  66  12   7 92 43 35
##   ser     38  60  88   6  30   0  30 14  2 10
##   time    11   9   6   7  13  16  10  9  9  7
##   told     6  17  16  11  28   5   7 20 16 15
##   tyrion  71   1   1   0   1   0  73  0  0  0

abs.freq_asoiaf<-rowSums(as.matrix(tdm_asoiaf))
abs.df_asoiaf <- data.frame(term = names(abs.freq_asoiaf),
                            frequency = abs.freq_asoiaf)
abs.df_asoiaf <- abs.df_asoiaf[order(abs.df_asoiaf[,2], decreasing=F),]

abs.df_asoiaf$term <- factor(abs.df_asoiaf$term,
                             levels = unique(abs.df_asoiaf$term))

ggplot(abs.df_asoiaf[23694:23713,], aes(x = term, y = frequency)) +
  geom_bar(stat="identity", fill='slateblue4') +
  coord_flip() +
  geom_text(aes(label = frequency), colour = "white",
            hjust = 1.25, size = 3) +
  theme_minimal() +
  theme(panel.grid = element_blank()) +
  scale_y_continuous(expand = c(0, 0)) +
  labs(title = "Most frequent words in the series\n'A Song of Ice and
       Fire'", x="Term", y="Frequency", caption ="Figure 3.2")

##### Common Words Between Books #####


asoiaf1.vec <- paste(as.vector(asoiafCorpus[1:73]), collapse=" ")
asoiaf2.vec <- paste(as.vector(asoiafCorpus[74:143]), collapse=" ")
asoiaf3.vec <- paste(as.vector(asoiafCorpus[144:225]), collapse=" ")
asoiaf4.vec <- paste(as.vector(asoiafCorpus[226:271]), collapse=" ")
asoiaf5.vec <- paste(as.vector(asoiafCorpus[272:344]), collapse=" ")

all_asoiaf <- c(asoiaf1.vec, asoiaf2.vec, asoiaf3.vec, asoiaf4.vec,
asoiaf5.vec)

asoiafCorpus.all <- VCorpus(VectorSource(all_asoiaf))
tdm.asoiaf.all <- TermDocumentMatrix(asoiafCorpus.all)
```

```
tdm.m.asoiaf.all <- as.matrix(tdm.asoiaf.all)
colnames(tdm.m.asoiaf.all) = c("A Game\nof Thrones",
                               "A Clash\nof Kings",
                               "A Storm\nof Swords",
                               "A Feast\nfor Crows",
                               "A Dance\nwith Dragons")

comparison.cloud(tdm.m.asoiaf.all, max.words=100, random.order=FALSE,
                 title.size=1.0,
                 colors=c("midnightblue", "darkorange3","darkgreen",
                          "indianred4","grey34"))


##### ARYA ####
##### Frequency ######
arya1<-text1[which(text1$narrator =="ARYA"),]
arya1 <- arya1[order(arya1[,2], decreasing=F),]

arya2<-text2[which(text2$narrator =="ARYA"),]
arya2<- arya2[order(arya2[,2], decreasing=F),]

arya3<-text3[which(text3$narrator =="ARYA"),]
arya3 <- arya3[order(arya3[,2], decreasing=F),]

arya4<-text4[which(text4$narrator =="ARYA"),]
arya4<- arya4[order(arya4[,2], decreasing=F),]

arya<-rbind(arya1,arya2,arya3,arya4)

aryaCorpus <- Corpus(VectorSource(arya$text))
arya.dtm <- DocumentTermMatrix(aryaCorpus)
inspect(arya.dtm)

## <<DocumentTermMatrix (documents: 30, terms: 8166)>>
## Non-/sparse entries: 40483/204497
## Sparsity           : 83%
## Maximal term length: 15
## Weighting          : term frequency (tf)
## Sample             :
##     Terms
## Docs and arya had her his she that the was you
##   10 192   53  89  82  47 108   48 325  83  86
##   12 183   42  47  77  66 104   58 282  83  21
##   14 211   63  53  97  74 143   44 437  79  77
##   15 199   45  59  97  72 162   42 432  91  62
##   17 179   45  32  85  60  85   45 258  48 114
##   22 198   31  40  34  87  37   53 291  66  72
##   28 169   53  46  82 123 115   23 362  74  48
##   29 154   49  47  83  53 107   36 345  70  21
##   30 238   57  61 152  26 175   59 436  86 134
##   9  192   52  28  74  67  98   23 404  82  42
```

```
aryaCorpus<-clean.corpus(aryaCorpus)
arya.dtm <- DocumentTermMatrix(aryaCorpus)
inspect(arya.dtm)
```

```
## <<DocumentTermMatrix (documents: 30, terms: 7343)>>
## Non-/sparse entries: 27367/192923
## Sparsity           : 88%
## Maximal term length: 15
## Weighting          : term frequency (tf)
## Sample             :
##     Terms
## Docs arya eyes gendry hand hot lord ser sword time told
##   10   53    5     38   10  40    0   3     5   12   14
##   14   63    7      7   11  15   35  25     7    8   10
##   15   45   13     12    5  14   52  18     6   10   10
##   17   45    7     18   13  34    3   0    12    1    4
##   21   32    6      8    6   1   49   5    26    2    4
##   22   31    4     13    8   1   55   6    11   14    6
##   28   53    7      0   11   0    2  12    10    8    8
##   29   49   13      1   12   1    2   3     3    7   16
##   30   57   12      1   10   5    1  12     8   15   15
##   9    52    7     19   10  24    4   7     4    9    5
```

```
arya.tidy <- tidy(arya.dtm)
colnames(arya.tidy)<-c('chapter_number','word','count')
arya.tidy$chapter_number<-as.numeric(arya.tidy$chapter_number)

count.words.arya<-count(arya.tidy, word)
total.words.arya<-sum(count.words.arya$n)

tdm_arya<-TermDocumentMatrix(aryaCorpus)
inspect(tdm_arya)
```

```
## <<TermDocumentMatrix (terms: 7343, documents: 30)>>
## Non-/sparse entries: 27367/192923
## Sparsity           : 88%
## Maximal term length: 15
## Weighting          : term frequency (tf)
## Sample             :
##         Docs
## Terms    10 14 15 17 21 22 28 29 30  9
##   arya   53 63 45 45 32 31 53 49 57 52
##   eyes    5  7 13  7  6  4  7 13 12  7
##   gendry 38  7 12 18  8 13  0  1  1 19
##   hand   10 11  5 13  6  8 11 12 10 10
##   hot    40 15 14 34  1  1  0  1  5 24
##   lord    0 35 52  3 49 55  2  2  1  4
##   ser     3 25 18  0  5  6 12  3 12  7
##   sword   5  7  6 12 26 11 10  3  8  4
##   time   12  8 10  1  2 14  8  7 15  9
##   told   14 10 10  4  4  6  8 16 15  5
```

```
abs.freq_arya<-rowSums(as.matrix(tdm_arya))
abs.df_arya<-data.frame(term = names(abs.freq_arya ),
                        frequency = abs.freq_arya)
abs.df_arya<-abs.df_arya[order(abs.df_arya[,2], decreasing=F),]

abs.df_arya$term<-factor(abs.df_arya$term,
                         levels = unique(abs.df_arya$term))

ggplot(abs.df_arya[7324:7343,], aes(x = term, y = frequency)) +
  geom_bar(stat="identity", fill='midnightblue') +
  coord_flip() +
  geom_text(aes(label = frequency), colour = "white",
            hjust = 1.25, size = 3) +
  theme_minimal() +
  theme(panel.grid = element_blank()) +
  scale_y_continuous(expand = c(0, 0)) +
  labs(title = "Most frequent words\nin 'Arya Stark'\nchapters",
       x="Term",y="Frequency",caption ="Figure 3.4")



##### Common Words #####
arya1.vec <- paste(as.vector(aryaCorpus[1:5]), collapse=" ")
arya2.vec <- paste(as.vector(aryaCorpus[6:15]), collapse=" ")
arya3.vec <- paste(as.vector(aryaCorpus[16:28]), collapse=" ")
arya4.vec <- paste(as.vector(aryaCorpus[29:30]), collapse=" ")

all_arya <- c(arya1.vec, arya2.vec, arya3.vec, arya4.vec)

aryaCorpus.all <- VCorpus(VectorSource(all_arya))
tdm.arya.all <- TermDocumentMatrix(aryaCorpus.all)
inspect(tdm.arya.all)

## <<TermDocumentMatrix (terms: 7438, documents: 4)>>
## Non-/sparse entries: 13364/16388
## Sparsity           : 55%
## Maximal term length: 17
## Weighting          : term frequency (tf)
## Sample             :
##         Docs
## Terms      1   2   3   4
##   arya   284 391 411 106
##   eyes    52  57  71  25
##   gendry   0  87 118   2
##   hand    61  60  71  21
##   hot      3 144  89   6
##   lord    29 145 223   3
##   ser     29 121  62  15
##   sword   45  65  83  11
##   time    30  71  95  22
##   told    36  69  82  31
```

```
tdm.m.arya.all <- as.matrix(tdm.arya.all)
colnames(tdm.m.arya.all) = c("A Game of Thrones", "A Clash of Kings",
                             "A Storm of Swords", "A Feast for Crows")
#Figure 3.5
comparison.cloud(tdm.m.arya.all, max.words=100, random.order=FALSE,
title.size=1.0,
                 colors=c("midnightblue", "darkorange3","darkgreen",
                          "indianred4"))

##### DAENERYS ####


##### Frequency ######
daenerys1<-text1[which(text1$narrator =="DAENERYS"),]
daenerys1 <- daenerys1[order(daenerys1[,2], decreasing=F),]

daenerys2<-text2[which(text2$narrator =="DAENERYS"),]
daenerys2<- daenerys2[order(daenerys2[,2], decreasing=F),]

daenerys3<-text3[which(text3$narrator =="DAENERYS"),]
daenerys3 <- daenerys3[order(daenerys3[,2], decreasing=F),]

daenerys5<-text5[which(text5$narrator =="DAENERYS"),]
daenerys5<- daenerys5[order(daenerys5[,2], decreasing=F),]

daenerys<-rbind(daenerys1,daenerys2,daenerys3,daenerys5)

daenerysCorpus <- Corpus(VectorSource(daenerys$text))
daenerys.dtm <- DocumentTermMatrix(daenerysCorpus)
inspect(daenerys.dtm)

## <<DocumentTermMatrix (documents: 31, terms: 9512)>>
## Non-/sparse entries: 49184/245688
## Sparsity           : 83%
## Maximal term length: 17
## Weighting          : term frequency (tf)
## Sample             :
##     Terms
## Docs and dany had her his she that the was you
##   16 219   46  63  78  86  58   68 330  79  70
##   17 253   63  49 121  70  93   83 428  58  63
##   19 221   60  45 102  83  73   68 346  82 103
##   20 222   49  53  91 104  77   44 380  79  73
##   21 207   59  73  93  55 128   59 379  90 108
##   22 175   49 118 106  76  78   46 418  69  48
##   24 159   36  64  82  65  57   68 328  70 117
##   28 185   43  52 127  89  82   64 282  80  86
##   30 169   41  38  91  98  74   36 377  61  38
##   31 234   52 105 219  57 202   65 408 115  70
```

```
daenerysCorpus<-clean.corpus(daenerysCorpus)
daenerys.dtm <- DocumentTermMatrix(daenerysCorpus)
inspect(daenerys.dtm)
```

```
## <<DocumentTermMatrix (documents: 31, terms: 8595)>>
## Non-/sparse entries: 33558/232887
## Sparsity           : 87%
## Maximal term length: 17
## Weighting          : term frequency (tf)
## Sample             :
##     Terms
## Docs blood dany dragons eyes grace jorah khal queen ser told
##   16     3   46      26    4    16    27    5    10  33   10
##   17    17   63      14   17    14    14    3    10  18    7
##   19     9   60      13    4    15    26    0    11  27   11
##   20    13   49       7   10    11    31    3     3  35   13
##   21     5   59       9    8    14    20    3    13  36   17
##   22     8   49       9   10    18     2    1    21  15   12
##   24     4   36      10    9    10     2    1    19   9    6
##   28     4   43       1   13    21     0    2    37  21    7
##   30    10   41       4   11    14     1    2    19  14    3
##   31    16   52      12   11     2     6   11     4   7   13
```

```
daenerys.tidy <- tidy(daenerys.dtm)
colnames(daenerys.tidy)<-c('chapter_number','word','count')
daenerys.tidy$chapter_number<-as.numeric(daenerys.tidy$chapter_number)

count.words.daenerys<-count(daenerys.tidy, word)
total.words.daenerys<-sum(count.words.daenerys$n)

tdm_daenerys<-TermDocumentMatrix(daenerysCorpus)
inspect(tdm_daenerys)
```

```
## <<TermDocumentMatrix (terms: 8595, documents: 31)>>
## Non-/sparse entries: 33558/232887
## Sparsity           : 87%
## Maximal term length: 17
## Weighting          : term frequency (tf)
## Sample             :
##          Docs
## Terms     16 17 19 20 21 22 24 28 30 31
##   blood    3 17  9 13  5  8  4  4 10 16
##   dany    46 63 60 49 59 49 36 43 41 52
##   dragons 26 14 13  7  9  9 10  1  4 12
##   eyes     4 17  4 10  8 10  9 13 11 11
##   grace   16 14 15 11 14 18 10 21 14  2
##   jorah   27 14 26 31 20  2  2  0  1  6
##   khal     5  3  0  3  3  1  1  2  2 11
##   queen   10 10 11  3 13 21 19 37 19  4
##   ser     33 18 27 35 36 15  9 21 14  7
##   told    10  7 11 13 17 12  6  7  3 13
```

```
abs.freq_daenerys<-rowSums(as.matrix(tdm_daenerys))
abs.df_daenerys<-data.frame(term = names(abs.freq_daenerys ),
                            frequency = abs.freq_daenerys)
abs.df_daenerys<-abs.df_daenerys[order(abs.df_daenerys[,2],
decreasing=F),]

abs.df_daenerys$term<-factor(abs.df_daenerys$term,
                             levels = unique(abs.df_daenerys$term))

ggplot(abs.df_daenerys[8576:8595,], aes(x = term, y = frequency)) +
  geom_bar(stat="identity", fill='indianred4') +
  coord_flip() +
  geom_text(aes(label = frequency), colour = "white",
            hjust = 1.25, size = 3) +
  theme_minimal() +
  theme(panel.grid = element_blank()) +
  scale_y_continuous(expand = c(0, 0)) +
  labs(title = "Most frequent words\nin 'Daenerys
Targaryen'\nchapters",
       x="Term", y="Frequency",caption ="Figure 3.6")



##### Common Words #####
daenerys1.vec <- paste(as.vector(daenerysCorpus[1:10]), collapse=" ")
daenerys2.vec <- paste(as.vector(daenerysCorpus[11:15]), collapse=" ")
daenerys3.vec <- paste(as.vector(daenerysCorpus[16:21]), collapse=" ")
daenerys5.vec <- paste(as.vector(daenerysCorpus[22:31]), collapse=" ")

all_daenerys <- c(daenerys1.vec, daenerys2.vec, daenerys3.vec,
daenerys5.vec)

daenerysCorpus.all <- VCorpus(VectorSource(all_daenerys))
tdm.daenerys.all <- TermDocumentMatrix(daenerysCorpus.all)
inspect(tdm.daenerys.all)

## <<TermDocumentMatrix (terms: 8710, documents: 4)>>
## Non-/sparse entries: 16546/18294
## Sparsity           : 53%
## Maximal term length: 17
## Weighting          : term frequency (tf)
## Sample             :
##          Docs
## Terms       1   2   3   4
##   blood   104  36  54 106
##   dany    394 166 323 369
##   dragons  39  70  82  83
##   eyes     97  30  53  90
##   grace     6  15  85 159
##   jorah   155  61 129  14
##   khal    197  18  14  20
##   queen    11  30  50 168
```

```
##   ser     159  65 158 131
##   told     88  34  67  93

tdm.m.daenerys.all <- as.matrix(tdm.daenerys.all)
colnames(tdm.m.daenerys.all) = c("A Game of Thrones",
                                 "A Clash of Kings",
                                 "A Storm of Swords",
                                 "A Dance with Dragons ")

#Figure 3.7
comparison.cloud(tdm.m.daenerys.all, max.words=100, random.order=FALSE,
title.size=1.0,
                 colors=c("midnightblue", "darkorange3","darkgreen",
"indianred4"))


##### JON ####


##### Frequency ######
jon1<-text1[which(text1$narrator =="JON"),]
jon1 <- jon1[order(jon1[,2], decreasing=F),]

jon2<-text2[which(text2$narrator =="JON"),]
jon2<- jon2[order(jon2[,2], decreasing=F),]

jon3<-text3[which(text3$narrator =="JON"),]
jon3 <- jon3[order(jon3[,2], decreasing=F),]

jon5<-text5[which(text5$narrator =="JON"),]
jon5<- jon5[order(jon5[,2], decreasing=F),]

jon<-rbind(jon1,jon2,jon3,jon5)

jonCorpus <- Corpus(VectorSource(jon$text))
jon.dtm <- DocumentTermMatrix(jonCorpus)
inspect(jon.dtm)

## <<DocumentTermMatrix (documents: 42, terms: 10161)>>
## Non-/sparse entries: 62680/364082
## Sparsity           : 85%
## Maximal term length: 48
## Weighting          : term frequency (tf)
## Sample             :
##     Terms
## Docs and had him his jon that the was with you
##   12 223  57  51 123  81   65 449  89   55  83
##   19 231  54  48  90  73   62 390 114   60  80
##   24 246  87  36  98  78   49 516  86   44  30
##   30 151  65  34  91  52   51 359  57   41 104
##   31 207  42  79 101  80   80 437  78   49 125
##   32 199  73  32 116  51   46 444  61   61  28
```

```
##   40 154  60  44  76  68   83 413  57   43 111
##   41 176  69  36  81  74   58 337  55   69  58
##   42 170  58  57  93  73   47 355  66   52  52
##   7  157  45  67 129  88   43 376  96   49  35
```

```
jonCorpus<-clean.corpus(jonCorpus)
jon.dtm <- DocumentTermMatrix(jonCorpus)
inspect(jon.dtm)
```

```
## <<DocumentTermMatrix (documents: 42, terms: 9286)>>
## Non-/sparse entries: 42434/347578
## Sparsity           : 89%
## Maximal term length: 48
## Weighting          : term frequency (tf)
## Sample             :
##     Terms
## Docs black eyes hand jon lord ser snow time told wall
##   12    22   12    6  81   35  10   11    7   15   15
##   19    14   13   10  73   10   1   28   12    6   10
##   24    26    4    7  78   10   0    9   10   15   18
##   31    14   12   11  80   49   5    8   16    8   25
##   32    14   11   11  51   29  11   11    3    7   34
##   33     5    4    7  53   42  14   21    6    3    9
##   40    17    8   12  68   40   6   24    8   12   32
##   41    23    4    7  74   13   0   37    7    6   18
##   42    14    0    5  73   22  20   39    9    8   11
##   7     17   17   17  88   31  20    6    9   11   16
```

```
jon.tidy <- tidy(jon.dtm)
colnames(jon.tidy)<-c('chapter_number','word','count')
jon.tidy$chapter_number<-as.numeric(jon.tidy$chapter_number)
```

```
count.words.jon<-count(jon.tidy, word)
total.words.jon<-sum(count.words.jon$n)
```

```
tdm_jon<-TermDocumentMatrix(jonCorpus)
inspect(tdm_jon)
```

```
## <<TermDocumentMatrix (terms: 9286, documents: 42)>>
## Non-/sparse entries: 42434/347578
## Sparsity           : 89%
## Maximal term length: 48
## Weighting          : term frequency (tf)
## Sample             :
##        Docs
## Terms   12 19 24 31 32 33 40 41 42  7
##   black 22 14 26 14 14  5 17 23 14 17
##   eyes  12 13  4 12 11  4  8  4  0 17
##   hand   6 10  7 11 11  7 12  7  5 17
##   jon   81 73 78 80 51 53 68 74 73 88
##   lord  35 10 10 49 29 42 40 13 22 31
##   ser   10  1  0  5 11 14  6  0 20 20
```

```
##   snow  11 28  9  8 11 21 24 37 39  6
##   time   7 12 10 16  3  6  8  7  9  9
##   told  15  6 15  8  7  3 12  6  8 11
##   wall  15 10 18 25 34  9 32 18 11 16

abs.freq_jon<-rowSums(as.matrix(tdm_jon))
abs.df_jon<-data.frame(term = names(abs.freq_jon ),
                       frequency = abs.freq_jon)
abs.df_jon<-abs.df_jon[order(abs.df_jon[,2], decreasing=F),]

abs.df_jon$term<-factor(abs.df_jon$term,
                        levels = unique(abs.df_jon$term))

ggplot(abs.df_jon[9267:9286,], aes(x = term, y = frequency)) +
  geom_bar(stat="identity", fill='midnightblue') +
  coord_flip() +
  geom_text(aes(label = frequency), colour = "white",
            hjust = 1.25, size = 3) +
  theme_minimal() +
  theme(panel.grid = element_blank()) +
  scale_y_continuous(expand = c(0, 0)) +
  labs(title = "Most frequent words\nin 'Jon Snow'\nchapters",
       caption ="Figure 3.8", x="Term", y="Frequency")



##### Common Words #####
jon1.vec <- paste(as.vector(jonCorpus[1:9]), collapse=" ")
jon2.vec <- paste(as.vector(jonCorpus[10:17]), collapse=" ")
jon3.vec <- paste(as.vector(jonCorpus[18:29]), collapse=" ")
jon5.vec <- paste(as.vector(jonCorpus[30:42]), collapse=" ")

all_jon <- c(jon1.vec, jon2.vec, jon3.vec, jon5.vec)

jonCorpus.all <- VCorpus(VectorSource(all_jon))
tdm.jon.all <- TermDocumentMatrix(jonCorpus.all)
inspect(tdm.jon.all)

## <<TermDocumentMatrix (terms: 9447, documents: 4)>>
## Non-/sparse entries: 18316/19472
## Sparsity           : 52%
## Maximal term length: 48
## Weighting          : term frequency (tf)
## Sample             :
##        Docs
## Terms     1   2   3   4
##   black  89  90 154 166
##   eyes   77  56  73  77
##   hand   68  41  77  95
##   jon   544 378 655 747
##   lord  142 110 171 361
##   ser    85  32  59 123
```

```
##   snow   72  50 162 312
##   time   51  45  86  96
##   told   76  53  84  90
##   wall   83  56 205 227

tdm.m.jon.all <- as.matrix(tdm.jon.all)
colnames(tdm.m.jon.all) = c("A Game of Thrones", "A Clash of Kings",
                            "A Storm of Swords", "A Dance with Dragons")

#Figure 3.9
comparison.cloud(tdm.m.jon.all, max.words=100, random.order=FALSE,
title.size=1.0, colors=c("midnightblue", "darkorange3","darkgreen",
"indianred4"))

##### TYRION ####


##### Frequency ######
tyrion1<-text1[which(text1$narrator =="TYRION"),]
tyrion1 <- tyrion1[order(tyrion1[,2], decreasing=F),]

tyrion2<-text2[which(text2$narrator =="TYRION"),]
tyrion2<- tyrion2[order(tyrion2[,2], decreasing=F),]

tyrion3<-text3[which(text3$narrator =="TYRION"),]
tyrion3 <- tyrion3[order(tyrion3[,2], decreasing=F),]

tyrion5<-text5[which(text5$narrator =="TYRION"),]
tyrion5<- tyrion5[order(tyrion5[,2], decreasing=F),]

tyrion<-rbind(tyrion1,tyrion2,tyrion3,tyrion5)

tyrionCorpus <- Corpus(VectorSource(tyrion$text))
tyrion.dtm <- DocumentTermMatrix(tyrionCorpus)
inspect(tyrion.dtm)

## <<DocumentTermMatrix (documents: 47, terms: 12605)>>
## Non-/sparse entries: 74269/518166
## Sparsity           : 87%
## Maximal term length: 27
## Weighting          : term frequency (tf)
## Sample             :
##     Terms
## Docs and had him his that the tyrion was with you
##   27 183  58  31 106   55 349     49  46   48  95
##   32 288  88  51 165   58 523     71 113   77  49
##   33 171  62  74  96   76 294     66  71   35 124
##   34 170  54  45 139   74 355     55 100   39  86
##   36 190  62  54 133   66 343     59  81   56  83
##   41 210  33  39 100   78 425     56  90   54 100
##   42 279  68  36 140   83 589     73  96   81  96
##   45 188  60  46 106   51 376     56  71   43  57
```

```
##   46 163  62  45  80   75 375     57  77   66  80
##   8  213  69  87 237   55 408     94  91   79  56

tyrionCorpus<-clean.corpus(tyrionCorpus)
tyrion.dtm <- DocumentTermMatrix(tyrionCorpus)
inspect(tyrion.dtm)

## <<DocumentTermMatrix (documents: 47, terms: 11596)>>
## Non-/sparse entries: 50979/494033
## Sparsity           : 91%
## Maximal term length: 27
## Weighting          : term frequency (tf)
## Sample             :
##     Terms
## Docs dwarf eyes father hand head king lord ser time tyrion
##   27     2    7     28    4    3   17   99  30    5     49
##   32    13   16     11   12    7   27   44  38   11     71
##   33     9    3     20   10   11   11   30  42    8     66
##   34     9    6     14   13   13    3   23  36    6     55
##   36    21    5     11    6   12    4   16   1    9     59
##   41    26    9     15    9    9    5   15   0    6     56
##   42    36   10      4   13   17    0    7  30   10     73
##   45    11   14      7    4    6    1    9   3   10     56
##   46     8   12      3    8    8    1    7   9    9     57
##   8      6   10     23   10   17    1   38  22    9     94

tyrion.tidy <- tidy(tyrion.dtm)
colnames(tyrion.tidy)<-c('chapter_number','word','count')
tyrion.tidy$chapter_number<-as.numeric(tyrion.tidy$chapter_number)

count.words.tyrion<-count(tyrion.tidy, word)
total.words.tyrion<-sum(count.words.tyrion$n)

tdm_tyrion<-TermDocumentMatrix(tyrionCorpus)
inspect(tdm_tyrion)

## <<TermDocumentMatrix (terms: 11596, documents: 47)>>
## Non-/sparse entries: 50979/494033
## Sparsity           : 91%
## Maximal term length: 27
## Weighting          : term frequency (tf)
## Sample             :
##         Docs
## Terms    27 32 33 34 36 41 42 45 46  8
##   dwarf   2 13  9  9 21 26 36 11  8  6
##   eyes    7 16  3  6  5  9 10 14 12 10
##   father 28 11 20 14 11 15  4  7  3 23
##   hand    4 12 10 13  6  9 13  4  8 10
##   head    3  7 11 13 12  9 17  6  8 17
##   king   17 27 11  3  4  5  0  1  1  1
##   lord   99 44 30 23 16 15  7  9  7 38
##   ser    30 38 42 36  1  0 30  3  9 22
```

```
##   time    5 11  8  6  9  6 10 10  9  9
##   tyrion 49 71 66 55 59 56 73 56 57 94

abs.freq_tyrion<-rowSums(as.matrix(tdm_tyrion))
abs.df_tyrion<-data.frame(term = names(abs.freq_tyrion ),
                          frequency = abs.freq_tyrion)
abs.df_tyrion<-abs.df_tyrion[order(abs.df_tyrion[,2], decreasing=F),]

abs.df_tyrion$term<-factor(abs.df_tyrion$term,
                           levels = unique(abs.df_tyrion$term))

ggplot(abs.df_tyrion[11577:11596,], aes(x = term, y = frequency)) +
  geom_bar(stat="identity", fill='indianred4') +
  coord_flip() +
  geom_text(aes(label = frequency), colour = "white",
            hjust = 1.25, size = 3) +
  theme_minimal() +
  theme(panel.grid = element_blank()) +
  scale_y_continuous(expand = c(0, 0)) +
  labs(title = "Most frequent words\nin 'Tyrion Lannister'\nchapters",
       x="Term", y="Frequency", caption="Figure 3.10")

##### Common Words #####
tyrion1.vec <- paste(as.vector(tyrionCorpus[1:9]), collapse=" ")
tyrion2.vec <- paste(as.vector(tyrionCorpus[10:24]), collapse=" ")
tyrion3.vec <- paste(as.vector(tyrionCorpus[25:35]), collapse=" ")
tyrion5.vec <- paste(as.vector(tyrionCorpus[36:47]), collapse=" ")

all_tyrion <- c(tyrion1.vec, tyrion2.vec, tyrion3.vec, tyrion5.vec)

tyrionCorpus.all <- VCorpus(VectorSource(all_tyrion))
tdm.tyrion.all <- TermDocumentMatrix(tyrionCorpus.all)
inspect(tdm.tyrion.all)

## <<TermDocumentMatrix (terms: 11767, documents: 4)>>
## Non-/sparse entries: 22810/24258
## Sparsity           : 52%
## Maximal term length: 27
## Weighting          : term frequency (tf)
## Sample             :
##         Docs
## Terms      1   2   3   4
##   dwarf   28  21  52 184
##   eyes    56  82  58  98
##   father  82  74 169  95
##   hand    66 147  89  93
##   head    49  78  74 101
##   king    29 121  98  26
##   lord   195 359 382 116
##   ser    138 224 245  80
##   time    54 105  78 100
##   tyrion 520 611 511 605
```

```
tdm.m.tyrion.all <- as.matrix(tdm.tyrion.all)
colnames(tdm.m.tyrion.all) = c("A Game of Thrones", "A Clash of Kings",
                               "A Storm of Swords", "A Dance with Dragons")

#Figure 3.11
comparison.cloud(tdm.m.tyrion.all, max.words=100, random.order=FALSE,
title.size=1.0,
                 colors=c("midnightblue", "darkorange3","darkgreen",
"indianred4"))


##### SERIES #####

##### Sentimental Analysis #####
nrc.joy <-subset(sentiments,sentiments$lexicon=='nrc' &
sentiments$sentiment=='joy')
bing <- subset(sentiments, sentiments$lexicon=='bing')[,-4]

asoiaf.dtm<-DocumentTermMatrix(asoiafCorpus)
asoiaf.tidy <- tidy(asoiaf.dtm)
colnames(asoiaf.tidy)<-c('chapter_number','word','count')
asoiaf.tidy$chapter_number<-as.numeric(asoiaf.tidy$chapter_number)

joy.words.asoiaf <- inner_join(asoiaf.tidy,nrc.joy, by="word")
count.joy.words.asoiaf <- count(joy.words.asoiaf, word)

asoiaf.sentiment <- inner_join(asoiaf.tidy,bing, by="word")
asoiaf.sentiment <- count(asoiaf.sentiment,sentiment,
index=chapter_number)
asoiaf.sentiment <- spread(asoiaf.sentiment, sentiment, n, fill = 0)

asoiaf.sentiment$polarity <- asoiaf.sentiment$positive -
asoiaf.sentiment$negative
asoiaf.sentiment$pos <- ifelse(asoiaf.sentiment$polarity >= 0, "pos",
"neg")

ggplot(asoiaf.sentiment, aes(x=index, y=polarity, fill=pos)) +
  geom_bar(stat="identity", position="identity", width=1) +
  guides(fill=guide_legend(title=NULL))+
  scale_fill_manual(values=c("midnightblue", "firebrick4"),
                    labels=c("Negative", "Positive")) +
  ggtitle("Polarity in the whole Series using 'bing' lexicon ") +
  labs(x="Narrative Time", y="Sentiment", caption="Figure 4.1",
       subtitle = "The vertical lines represent the end of each book")+
  theme_grey()+
  theme(legend.position="bottom")+
  geom_vline(xintercept = c(73,143,225,271))

## `geom_smooth()` using method = 'loess'
```

```
asoiaf.sentiment <- inner_join(asoiaf.tidy,bing, by="word")
asoiaf.negative<-
asoiaf.sentiment[which(asoiaf.sentiment$sentiment=="negative"),]
asoiaf.negative<-aggregate(asoiaf.negative$count,
list(asoiaf.negative$word), sum)
asoiaf.negative<-asoiaf.negative[order(asoiaf.negative$x,
decreasing=F),]
asoiaf.negative<-as.data.frame(asoiaf.negative)
asoiaf.negative$Group.1 <- factor(asoiaf.negative$Group.1,
                              levels = unique(asoiaf.negative$Group.1))

ggplot(asoiaf.negative[2204:2225,], aes(x = Group.1, y = x)) +
  geom_bar(stat="identity", fill='slateblue4') +
  coord_flip() +
  geom_text(aes(label = x), colour = "white",
            hjust = 1.25, size = 3) +
  theme_minimal() +
  theme(panel.grid = element_blank()) +
  scale_y_continuous(expand = c(0, 0)) +
  labs(title = "Most frequent negative\nwords in the whole Series",
       x="Frequency", y="Word",  caption="Figure 4.2")

asoiaf.positive<-
asoiaf.sentiment[which(asoiaf.sentiment$sentiment=="positive"),]
asoiaf.positive<-aggregate(asoiaf.positive$count,
list(asoiaf.positive$word), sum)
asoiaf.positive<-asoiaf.positive[order(asoiaf.positive$x,
decreasing=F),]
asoiaf.positive<-as.data.frame(asoiaf.positive)
asoiaf.positive$Group.1 <- factor(asoiaf.positive$Group.1,
                                  levels =
unique(asoiaf.positive$Group.1))

ggplot(asoiaf.positive[928:949,], aes(x = Group.1, y = x)) +
  geom_bar(stat="identity", fill='firebrick4') +
  coord_flip() +
  geom_text(aes(label = x), colour = "white",
            hjust = 1.25, size = 3) +
  theme_minimal() +
  theme(panel.grid = element_blank()) +
  scale_y_continuous(expand = c(0, 0)) +
  labs(title = "Most frequent positive\nwords in the Series",
       x="Frequency", y="Word", caption="Figure 4.3")



##### Emotion Classification ####


data(emotions)

emo.asoiaf.df <- as.data.frame(classify_emotion(asoiaf$text))
```

```
score<-round(as.numeric(as.matrix(emo.asoiaf.df[,1:6])),2)
chapter<-as.numeric(matrix(rep(seq(1,334),6), nrow = 334*6, ncol = 1))
Emotion<-as.character(matrix(rep(c("Anger",
"Disgust","Fear","Joy","Sadness","Surprise"),
                                 c(334,334,334,334,334,334)),
                             nrow = 334*6, ncol = 1))

asoiaf.plot<-as.data.frame(cbind(score,chapter,Emotion))[1:2004,]
asoiaf.plot[, 1] <- as.numeric(as.character( asoiaf.plot[, 1] ))
asoiaf.plot[, 2] <- as.numeric(as.character( asoiaf.plot[, 2] ))

ggplot(data=asoiaf.plot, aes(x=asoiaf.plot$chapter, y=score,
fill=Emotion)) +
  geom_bar(na.rm=TRUE, position = "fill", stat="identity", width = 1)+
  labs(x="Chapters", y="Score", caption="Figure 4.4")+
  ggtitle("Emotions in the whole series ")+
  scale_fill_manual(values=c("firebrick", "seagreen", "dodgerblue4",
                             "chocolate2", "seashell4", "goldenrod2"))+
  theme_gray()+
  geom_vline(xintercept = c(73,143,225,271))



##### ARYA #####


##### Sentimental Analysis #####

joy.words.arya <- inner_join(arya.tidy,nrc.joy, by="word")
count.joy.words.arya <- count(joy.words.arya, word)

arya.sentiment <- inner_join(arya.tidy,bing, by="word")
arya.sentiment <- count(arya.sentiment,sentiment, index=chapter_number)
arya.sentiment <- spread(arya.sentiment, sentiment, n, fill = 0)

arya.sentiment$polarity <- arya.sentiment$positive -
arya.sentiment$negative
arya.sentiment$pos <- ifelse(arya.sentiment$polarity >= 0, "pos",
"neg")

ggplot(arya.sentiment, aes(x=index, y=polarity, fill=pos)) +
  geom_bar(stat="identity", position="identity", width=0.91,
           fill="midnightblue")) +
  ggtitle("Polarity in 'Arya Stark' chapters using 'bing' lexicon ") +
  labs(x="Narrative Time", y="Sentiment", caption="Figure 4.5",
       subtitle = "The vertical lines represent the end of each
book.")+
  theme_grey()+
  geom_vline(xintercept = c(5,15,28))

## `geom_smooth()` using method = 'loess'
```

```
arya.sentiment <- inner_join(arya.tidy,bing, by="word")
arya.negative<-
arya.sentiment[which(arya.sentiment$sentiment=="negative"),]
arya.negative<-aggregate(arya.negative$count, list(arya.negative$word),
sum)
arya.negative<-arya.negative[order(arya.negative$x, decreasing=F),]
arya.negative<-as.data.frame(arya.negative)
arya.negative$Group.1 <- factor(arya.negative$Group.1,
                                levels = unique(arya.negative$Group.1))

ggplot(arya.negative[799:808,], aes(x = Group.1, y = x)) +
  geom_bar(stat="identity", fill='slateblue4') +
  coord_flip() +
  geom_text(aes(label = x), colour = "white",
            hjust = 1.25, size = 3) +
  theme_minimal() +
  theme(panel.grid = element_blank()) +
  scale_y_continuous(expand = c(0, 0)) +
  labs(title = "Most frequent negative\nwords in 'Arya
Stark'\nchapters",
       x="Frequency", y="Word",  caption="Figure 4.6"

arya.positive<-
arya.sentiment[which(arya.sentiment$sentiment=="positive"),]
arya.positive<-aggregate(arya.positive$count, list(arya.positive$word),
sum)
arya.positive<-arya.positive[order(arya.positive$x, decreasing=F),]
arya.positive<-as.data.frame(arya.positive)
arya.positive$Group.1 <- factor(arya.positive$Group.1,
                                levels = unique(arya.positive$Group.1))

ggplot(arya.positive[286:295,], aes(x = Group.1, y = x)) +
  geom_bar(stat="identity", fill='firebrick4') +
  coord_flip() +
  geom_text(aes(label = x), colour = "white",
            hjust = 1.25, size = 3) +
  theme_minimal() +
  theme(panel.grid = element_blank()) +
  scale_y_continuous(expand = c(0, 0)) +
  labs(title = "Most frequent positive\nwords in 'Arya
Stark'\nchapters",
       x="Frequency", y="Word", caption="Figure 4.7")

##### DAENERYS #####


##### Sentimental Analysis #####

joy.words.daenerys <- inner_join(daenerys.tidy,nrc.joy, by="word")
count.joy.words.daenerys <- count(joy.words.daenerys, word)
```

```
daenerys.sentiment <- inner_join(daenerys.tidy,bing, by="word")
daenerys.sentiment <- count(daenerys.sentiment,sentiment,
index=chapter_number)
daenerys.sentiment <- spread(daenerys.sentiment, sentiment, n, fill =
0)

daenerys.sentiment$polarity <- daenerys.sentiment$positive -
daenerys.sentiment$negative
daenerys.sentiment$pos <- ifelse(daenerys.sentiment$polarity >= 0,
"pos", "neg")

ggplot(daenerys.sentiment, aes(x=index, y=polarity, fill=pos)) +
  geom_bar(stat="identity", position="identity", width=0.91,
           fill="midnightblue")) +
  ggtitle("Polarity in 'Daenerys Targaryen' chapters using 'bing'
lexicon ") +
  labs(x="Narrative Time", y="Sentiment", caption="Figure 4.8",
       subtitle = "The vertical lines represent the end of each
book.")+
  theme_grey()+
  geom_vline(xintercept = c(10,15,21))

## `geom_smooth()` using method = 'loess'

daenerys.sentiment <- inner_join(daenerys.tidy,bing, by="word")
daenerys.negative<-
daenerys.sentiment[which(daenerys.sentiment$sentiment=="negative"),]
daenerys.negative<-aggregate(daenerys.negative$count,
list(daenerys.negative$word), sum)
daenerys.negative<-daenerys.negative[order(daenerys.negative$x,
decreasing=F),]
daenerys.negative<-as.data.frame(daenerys.negative)
daenerys.negative$Group.1 <- factor(daenerys.negative$Group.1,
                                    levels =
unique(daenerys.negative$Group.1))

ggplot(daenerys.negative[1015:1024,], aes(x = Group.1, y = x)) +
  geom_bar(stat="identity", fill='slateblue4') +
  coord_flip() +
  geom_text(aes(label = x), colour = "white",
            hjust = 1.25, size = 3) +
  theme_minimal() +
  theme(panel.grid = element_blank()) +
  scale_y_continuous(expand = c(0, 0)) +
  labs(title = "Most frequent negative\nwords in 'Daenerys
Targaryen'\nchapters",
       x="Frequency", y="Word",  caption="Figure 4.9")
```

```
daenerys.positive<-
daenerys.sentiment[which(daenerys.sentiment$sentiment=="positive"),]
daenerys.positive<-aggregate(daenerys.positive$count,
list(daenerys.positive$word), sum)
daenerys.positive<-daenerys.positive[order(daenerys.positive$x,
decreasing=F),]
daenerys.positive<-as.data.frame(daenerys.positive)
daenerys.positive$Group.1 <- factor(daenerys.positive$Group.1,
                                    levels =
unique(daenerys.positive$Group.1))

ggplot(daenerys.positive[481:490,], aes(x = Group.1, y = x)) +
  geom_bar(stat="identity", fill='firebrick4') +
  coord_flip() +
  geom_text(aes(label = x), colour = "white",
            hjust = 1.25, size = 3) +
  theme_minimal() +
  theme(panel.grid = element_blank()) +
  scale_y_continuous(expand = c(0, 0)) +
  labs(title = "Most frequent positive\nwords in 'Daenerys
Targaryen'\nchapters",
       x="Frequency", y="Word", caption="Figure 4.10")



##### JON ####


##### Sentimental Analysis #####

joy.words.jon <- inner_join(jon.tidy,nrc.joy, by="word")
count.joy.words.jon <- count(joy.words.jon, word)

jon.sentiment <- inner_join(jon.tidy,bing, by="word")
jon.sentiment <- count(jon.sentiment,sentiment, index=chapter_number)
jon.sentiment <- spread(jon.sentiment, sentiment, n, fill = 0)

jon.sentiment$polarity <- jon.sentiment$positive -
jon.sentiment$negative
jon.sentiment$pos <- ifelse(jon.sentiment$polarity >= 0, "pos", "neg")

ggplot(jon.sentiment, aes(x=index, y=polarity, fill=pos)) +
  geom_bar(stat="identity", position="identity", width=0.91,
           fill="midnightblue")) +
  ggtitle("Polarity in 'Jon Snow' chapters using 'bing' lexicon") +
  labs(x="Narrative Time", y="Sentiment", caption="Figure 4.11",
       subtitle = "The vertical lines represent the end of each
book.")+
  theme_grey()+
  geom_vline(xintercept = c(9,17,29))

## `geom_smooth()` using method = 'loess'
```

```
jon.sentiment <- inner_join(jon.tidy,bing, by="word")
jon.negative<-
jon.sentiment[which(jon.sentiment$sentiment=="negative"),]
jon.negative<-aggregate(jon.negative$count, list(jon.negative$word),
sum)
jon.negative<-jon.negative[order(jon.negative$x, decreasing=F),]
jon.negative<-as.data.frame(jon.negative)
jon.negative$Group.1 <- factor(jon.negative$Group.1,
                               levels = unique(jon.negative$Group.1))

ggplot(jon.negative[1097:1106,], aes(x = Group.1, y = x)) +
  geom_bar(stat="identity", fill='slateblue4') +
  coord_flip() +
  geom_text(aes(label = x), colour = "white",
            hjust = 1.25, size = 3) +
  theme_minimal() +
  theme(panel.grid = element_blank()) +
  scale_y_continuous(expand = c(0, 0)) +
  labs(title = "Most frequent negative\nwords in 'Jon Snow'\nchapters",
       x="Frequency", y="Word",  caption="Figure 4.12")

jon.positive<-
jon.sentiment[which(jon.sentiment$sentiment=="positive"),]
jon.positive<-aggregate(jon.positive$count, list(jon.positive$word),
sum)
jon.positive<-jon.positive[order(jon.positive$x, decreasing=F),]
jon.positive<-as.data.frame(jon.positive)
jon.positive$Group.1 <- factor(jon.positive$Group.1,
                               levels = unique(jon.positive$Group.1))

ggplot(jon.positive[428:437,], aes(x = Group.1, y = x)) +
  geom_bar(stat="identity", fill='firebrick4') +
  coord_flip() +
  geom_text(aes(label = x), colour = "white",
            hjust = 1.25, size = 3) +
  theme_minimal() +
  theme(panel.grid = element_blank()) +
  scale_y_continuous(expand = c(0, 0)) +
  labs(title = "Most frequent positive\nwords in 'Jon Snow'\nchapters",
       x="Frequency", y="Word", caption="Figure 4.13")



##### TYRION ####


##### Sentimental Analysis #####

joy.words.tyrion <- inner_join(tyrion.tidy,nrc.joy, by="word")
count.joy.words.tyrion <- count(joy.words.tyrion, word)
```

```
tyrion.sentiment <- inner_join(tyrion.tidy,bing, by="word")
tyrion.sentiment <- count(tyrion.sentiment,sentiment,
index=chapter_number)
tyrion.sentiment <- spread(tyrion.sentiment, sentiment, n, fill = 0)

tyrion.sentiment$polarity <- tyrion.sentiment$positive -
tyrion.sentiment$negative
tyrion.sentiment$pos <- ifelse(tyrion.sentiment$polarity >= 0, "pos",
"neg")

ggplot(tyrion.sentiment, aes(x=index, y=polarity, fill=pos)) +
  geom_bar(stat="identity", position="identity", width=0.91,
           fill="midnightblue")) +
  ggtitle("Polarity in 'Tyrion Lannister' chapters using 'bing'
lexicon") +
  labs(x="Narrative Time", y="Sentiment",caption="Figure 4.14",
       subtitle = "The vertical lines represent the end of each
book.")+
  theme_grey()+
  geom_vline(xintercept = c(9,24,35))

## `geom_smooth()` using method = 'loess'

tyrion.sentiment <- inner_join(tyrion.tidy,bing, by="word")
Tyrion.negative<-
tyrion.sentiment[which(tyrion.sentiment$sentiment=="negative"),]
tyrion.negative<-aggregate(tyrion.negative$count,
list(tyrion.negative$word), sum)
tyrion.negative<-tyrion.negative[order(tyrion.negative$x,
decreasing=F),]
tyrion.negative<-as.data.frame(tyrion.negative)
tyrion.negative$Group.1 <- factor(tyrion.negative$Group.1,
                                  levels =
unique(tyrion.negative$Group.1))

ggplot(tyrion.negative[1432:1441,], aes(x = Group.1, y = x)) +
  geom_bar(stat="identity", fill='slateblue4') +
  coord_flip() +
  geom_text(aes(label = x), colour = "white",
            hjust = 1.25, size = 3) +
  theme_minimal() +
  theme(panel.grid = element_blank()) +
  scale_y_continuous(expand = c(0, 0)) +
  labs(title = "Most frequent negative\nwords in 'Tyrion
Lannister'\nchapters",
       x="Frequency", y="Word",  caption="Figure 4.15")

tyrion.positive<-
tyrion.sentiment[which(tyrion.sentiment$sentiment=="positive"),]
tyrion.positive<-aggregate(tyrion.positive$count,
list(tyrion.positive$word), sum)
```

```
tyrion.positive<-tyrion.positive[order(tyrion.positive$x,
decreasing=F),]
tyrion.positive<-as.data.frame(tyrion.positive)
tyrion.positive$Group.1 <- factor(tyrion.positive$Group.1,
                                  levels =
unique(tyrion.positive$Group.1))

ggplot(tyrion.positive[613:622,], aes(x = Group.1, y = x)) +
  geom_bar(stat="identity", fill='firebrick4') +
  coord_flip() +
  geom_text(aes(label = x), colour = "white",
            hjust = 1.25, size = 3) +
  theme_minimal() +
  theme(panel.grid = element_blank()) +
  scale_y_continuous(expand = c(0, 0)) +
  labs(title = "Most frequent positive\nwords in 'Tyrion
Lannister'\nchapters",
       x="Frequency", y="Word", caption="Figure 4.16")
```