MA22019 2025 - Solutions for Problem Sheet 4

Text data analysis - Word frequency and sentiment

Overview

This week's problem sheet focuses on the text data analysis techniques covered in Sections 3.1-3.3.1 in the lecture notes. Exercises 1-3 are designed to help you with revising the content of the lecture from Week 5. You can check your solutions to these questions yourself by answering the Moodle quiz.

Tutorial Question 1 is similar to Exercises 1-3, while Tutorial Question 2 explores how well the type of sentiment analysis we introduced performs at assessing sentiment for product reviews. Should time permit, you may want to work on the Homework Question during the tutorial.

Your answer to the Homework Question can be submitted on Moodle to your tutor for feedback. The submission deadline is 17:00 on Thursday 13 March 2025. You should submit a single PDF or Word file that provides your R code, any created R output and all your comments.

You may want to load the following packages before starting the exercise:

```
library( dplyr )
library( ggplot2 )
library( tidytext )
library( wordcloud )
library( stringr )
```

When working on a University PC, you have to first install the tidytext, textdata and wordcloud packages.

Exercise 1 - Short stories by Edgar Allan Poe

The file "Poe.csv" contains the book *The Works of Edgar Allan Poe - Volume 1* which includes eight short stories by the American author Edgar Allen Poe. Each row in the data set gives a line from the book and the name of the short story that line belongs to. To load the data, we use

```
Poe_raw <- read.csv("Poe.csv")
```

The following exercises are designed to help you with revising the steps we used for extracting the most common words for *Jane Eyre* in the lecture.

a) Extract the individual words from the text and remove any underscores.

Looking at Section 3.1.2, we identify that we need to use unnest_tokens() and gsub():

```
Poe <- Poe_raw %>%
  unnest_tokens( word, text ) %>%
  mutate( word = gsub( "\\_", "", word ) )
```

b) Which are the five most common words (including stop words) across all stories?

We apply the count() function we used for calculating the term frequency in Section 3.1.2 and output the five words with the highest count:

```
Poe %>%
  count( word, sort=TRUE ) %>%
  slice_head(n=5)
```

```
##   word    n
## 1  the 6585
## 2   of 3788
## 3  and 2112
## 4   to 2055
## 5    a 1890
```

We find that "the", "of", "and", "to" and "a" are the most common words across the short stories.

c) For each short story, identify the word most commonly used within it (excluding stop words).

We first remove the stop words in the list provided by the tidytext package:

```
data( stop_words )
Poe <- Poe %>% anti_join( stop_words, by="word" )
```

Finally, we apply the count() function again, but we need to do it for each story. So we combine count() with group_by():

```
Poe %>%
  group_by( story ) %>%
  count( word, sort=TRUE ) %>%
  slice_head( n=1 )
```

```
## # A tibble: 8 x 3
## # Groups:   story [8]
##   story                          word          n
##   <chr>                          <chr>     <int>
## 1 Four Beasts in one             thousand     16
## 2 Hans Pfaal                     earth        61
## 3 MS. Found in a Bottle          ship         27
```

```
## 4 The Baloon-Hoax                balloon     38
## 5 The Gold-Bug                   de          73
## 6 The Murders in the Rue Morgue  voice       42
## 7 The Mystery of Marie Roget     marie       75
## 8 The Oval Portrait              deep         7
```

Exercise 2 - Baby names in the USA between 1880 and 2017

The file "Babynames.csv" provides a comprehensive record of the names given to newborns in the United States between 1880 and 2017 according to the Social Security Administration. For each name, year and sex, we are given the number of newborn babies given that name (as long as the name was used at least five times in that year). We are further provided with the proportion of newborn babies who were given that name amongst babies of the same sex.

a) What was the most common baby name over the period 1880-2017?

We start by loading the data set:

```
Babies <- read.csv("Babynames.csv")
```

To find the most common name, we have to sum up the counts across the different years:

```
Babies %>%
  group_by( Name=name ) %>%
  summarise( Count = sum(n) ) %>%
  arrange( desc(Count) ) %>%
  slice_head( n=1 )
```

```
## # A tibble: 1 x 2
##   Name    Count
##   <chr>   <int>
## 1 James 5173828
```

We find that "James" was the most common name given to a baby over the period 1880-2017.

b) Create a word cloud which visualizes the 30 most common baby names (in terms of total number) for a girl between 1880 and 2017.

We start by calculating for each name how many girls were given that name.

```
Names_Girls <- Babies %>%
  filter( sex=="F" ) %>%
  group_by( name ) %>%
  summarize( Count=sum(n) )
```

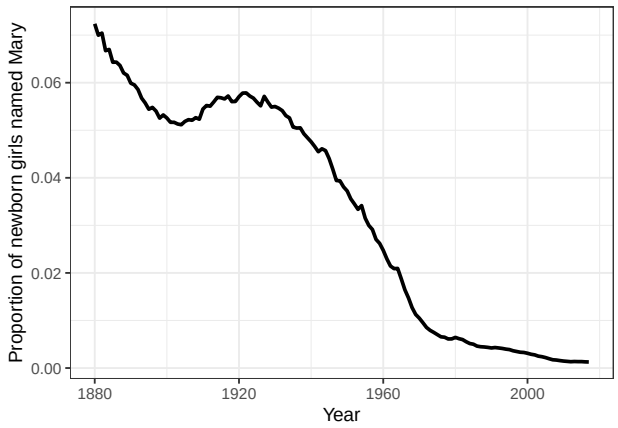
Using the code in Section 3.1.2, we create a word cloud as follows:

```
Names_Girls %>% with( wordcloud( name, Count, max.words=30 ) )
```

c) Explore how the popularity of the girl name "Mary" evolved over the period 1880-2017. When did its popularity peak?

One way is to create a plot for the proportions over time

```
Babies %>%
  filter( sex=="F", name=="Mary" ) %>%
  ggplot( aes("x"=year, "y"=prop) ) + geom_line( linewidth=1 ) +
  labs( x="Year", y="Proportion of newborn girls named Mary" ) + theme_bw()
```



From the plot, we conclude that "Mary" has become less popular over the period, falling from a peak of more than 7% in 1880 to 0.1-0.2% in the 2010s.

Exercise 3 - Analysis of the book *Frankenstein*

The novel *Frankenstein* by Mary Shelley tells the story of a young scientist who creates a creature via an experiment and is subsequently horrified by what he has made. The book is provided in the file "Frankenstein.csv" and can be loaded using

```
Frankenstein_raw <- read.csv( "Frankenstein.csv", fileEncoding = "UTF-8" )
```

Perform an analysis that addresses the following questions for *Frankenstein*:

a) Which five words appear the most often (excluding stop words)?

We first split the lines of text and remove any punctuation:

```
Frankenstein <- Frankenstein_raw %>% unnest_tokens( word, text )
Frankenstein <- Frankenstein %>% mutate( word = gsub( "_", "", word ) )
```

We now count the number of occurrences for each word and remove any words that are categorized as stop words. Finally, we print the five most common words (excluding stop words):

```
Frankenstein %>%
  count( word, sort=TRUE ) %>%
  anti_join( stop_words, by=c("word") ) %>%
  slice_head( n=5 )
```

```
##     word   n
## 1   life 115
## 2 father 113
## 3   eyes 104
## 4   time  98
## 5  night  92
```

We conclude that that five most common words (excluding stop words) are "life", "father", "eyes", "time" and "night".

b) Calculate the AFINN sentiment score (sum of the AFINN score of all words) for each chapter. Which chapter has the highest/lowest sentiment score?

We want to study chapters, and thus we take the code from Section 3.2.2 to derive the chapter the individual lines belong to:

```
Frankenstein_chapters <- Frankenstein_raw %>%
  mutate( chapter = cumsum( str_detect(
    text, regex("^chapter ", ignore_case = TRUE)
  ) ) )
```

The next step is to remove all the text before the first chapter begins, and to then split the remaining text into the indiviudal words:

```
Frankenstein_chapters <- Frankenstein_chapters %>%
  filter( chapter > 0 ) %>%
  unnest_tokens( word, text ) %>%
  mutate( word = gsub( "\\_", "", word ) )
```

Since we are asked to use the AFINN sentiment lexicon, we load it:

```
AFINN <- get_sentiments( "afinn" )
```

We can now use the code from Section 3.2.2 to derive the sentiment score for each chapter:

```
Frankenstein_sentiment <- Frankenstein_chapters %>%
  inner_join( AFINN, by="word" ) %>%
  group_by( chapter ) %>%
  summarise( sentiment = sum(value) )
```

Finally, let's extract the chapters with the lowest and highest AFINN sentiment score:

```
slice_min( Frankenstein_sentiment, sentiment, n=1 )
```

```
## # A tibble: 1 x 2
##   chapter sentiment
##     <int>     <dbl>
## 1      24      -290
```

```
slice_max( Frankenstein_sentiment, sentiment, n=1 )
```

```
## # A tibble: 1 x 2
##   chapter sentiment
##     <int>     <dbl>
## 1       6       157
```

We conclude that Chapter 6 has the highest score (157), while Chapter 24 has the lowest score (-290).

Tutorial Question 1 - Word frequency analysis for *Les Misérables*

We want to analyse the book *Les Misérables* by Victor Hugo in terms of the words used therein. The file "LesMis.csv" contains the data as provided by Project Gutenberg, but with the metadata at the beginning and end already removed. Consider the following questions:

a) Extract the individual words from the text, and remove any stop words and underscores.

We adapt the code from the lectures and use the functions unnest_tokens(), gsub() and anti_join():

```
LesMis_raw <- read.csv( "LesMis.csv", fileEncoding="UTF-8" )
data( stop_words )

LesMis <- LesMis_raw %>%
  unnest_tokens( word, text ) %>%
  mutate( word = gsub( "\\_", "", word ) )  %>%
  anti_join( stop_words, by="word" )
```

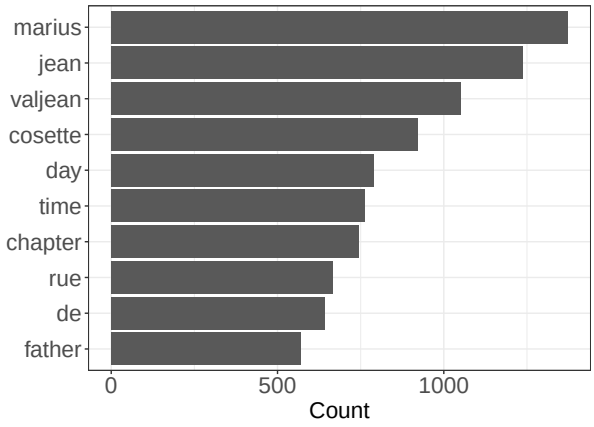
b) Find the 10 most common in *Les Misérables* (excluding stop words) and create a bar plot which visualizes their number of occurrences.

We first create a data frame that gives the number of times each word appeared in the text and we sort the words based on this count

```
LesMis_Count <- LesMis %>% count( word, sort=TRUE )
```

We now extract and reorder the top 10 words before creating a bar plot as in Section 3.1.2 of the lecture notes:

```
LesMis_Count %>%
  slice_head( n=10 ) %>%
  mutate( word = reorder(word,n) ) %>%
  ggplot( aes( x=n, y=word) ) + geom_col() +
  theme_bw() + labs( x="Count", y="" ) +
  theme( axis.text=element_text(size=17), axis.title=element_text(size=17) )
```



We find that the four most frequent words (excluding stop words) refer to three of the protagonists in the book. We also see that the word "chapter" appears very often. This finding results from the chapter titles being labeled "Chapter.." and thus this is not really accurate (the book also provides a table of contents that uses "chapter" a lot).

c) Create a word cloud for the 30 most frequently used words (excluding stop words).

We adapt the code from Section 3.1.2 in the lecture notes:

```
LesMis_Count %>% with( wordcloud( word, n, max.words=30 ) )
```

We again see that the 30 most common words include the main characters and the city of Paris, where most of the story takes place. The remaining words are hard to group within a single theme.

Tutorial Question 2 - Amazon Reviews

One important application area for sentiment analysis concerns the analysis of customer reviews. To explore how good the AFINN lexicon performs at capturing the overall sentiment of a review, we consider 1,000 Amazon product reviews. The data is stored in the file "AmazonReviews.csv".

```
Reviews_raw <- read.csv( "AmazonReviews.csv" )
```

The data provides the rating ("1 Star" to "5 Stars") and the accompanying text for each review. We want to derive the sentiment for each review and then compare it to the given score to assess how well we perform at capturing the sentiment of the text.

a) Using the AFINN sentiment lexicon, for each review, calculate the average sentiment per word.

We start by splitting the reviews into individual words:

```
Reviews <- Reviews_raw %>% unnest_tokens( word, Text )
```

The question asks us to use th AFINN sentiment lexicon, and so we have to load it:

```
AFINN <- get_sentiments( "afinn" )
```

Instead of calculating the aggregate sentiment, we are asked to consider average sentiment per word. This means we cannot simply remove the words that are not in the sentiment lexicon, i.e, the anti_join() function would be a poor choice. Instead, we need to keep all the words and replace any NA entries by 0. This can, for instance, be achieved using the following code

```
Reviews_Sentiment <- Reviews %>%
  left_join( AFINN, by="word" ) %>%
  mutate( value = case_when( is.na(value)==TRUE ~ 0, .default = value ) ) %>%
  group_by( Review ) %>%
  summarise( Sentiment=mean(value) )
```

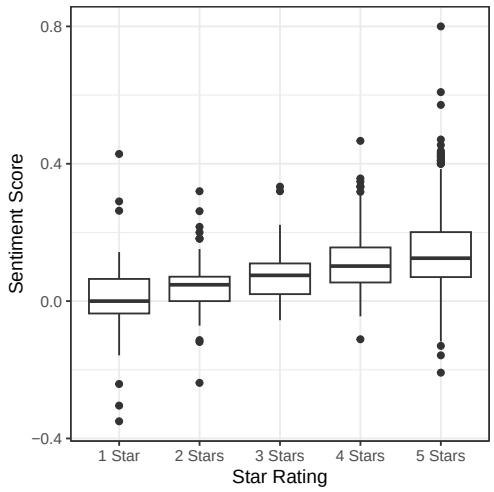
b) Plot the average sentiments calculated in part a) against the star ratings. What do you conclude?

To create the plot, we first need to link up the sentiment score derived in part a) with the star ratings:

```
Reviews_Sentiment <- Reviews_Sentiment %>% mutate( Score = Reviews_raw$Score )
```

Now we can create box plots of the calculated sentiment scores for each category of the star rating:

```
ggplot( Reviews_Sentiment, aes(x=Score, y=Sentiment) ) +
  geom_boxplot() + theme_bw() + labs( x="Star Rating", y="Sentiment Score")
```



We see that higher star ratings are associated with slightly higher sentiment scores. However, we should note that sentiment scores are quite variable, and thus the sentiment score is not necessarily a great predictor in terms of star rating.

Homework Question - The work of H.G. Wells

We want to analyze and compare the novels *The Time Machine* and *War of the Worlds* by the British author Herbert George Wells. The books are provided in the files "Time Machine.csv" and "War of Worlds.csv".

a) Compare the two books in terms of the words used within them.

We start by loading the two text data sets:

```
Time_raw <- read.csv( "Time Machine.csv" )
War_raw <- read.csv( "War of Worlds.csv" )
```

As in previous analyses, we split the texts into individual words and remove any underscores:

```
Time <- Time_raw %>%
  unnest_tokens( word, text ) %>%
  mutate( word = gsub( "\\_", "", word ) )
War <- War_raw %>%
  unnest_tokens( word, text ) %>%
  mutate( word = gsub( "\\_", "", word ) )
```

With the data ready for analysis, we calculate the number of occurrences and term frequency for each word:

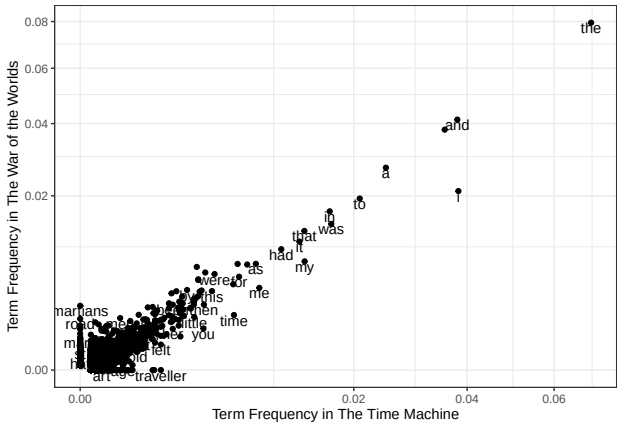
```
Time <- Time %>%
  count( word, sort=TRUE ) %>%
  mutate( tf = n / sum(n) )
War <- War %>%
  count( word, sort=TRUE ) %>%
  mutate( tf = n / sum(n) )
```

To compare the two books, we merge the calculated term frequencies into one data frame and replace any missing entries (corresponding to the word only occurring in one book) with a value of 0:

```
Time_War <- full_join( Time, War, by = "word" ) %>%
  rename( tf.Time=tf.x, tf.War=tf.y ) %>%
  mutate( tf.Time = case_when( is.na(tf.Time) == TRUE ~ 0, .default = tf.Time ),
          tf.War = case_when( is.na(tf.War) == TRUE ~ 0, .default = tf.War ) )
```

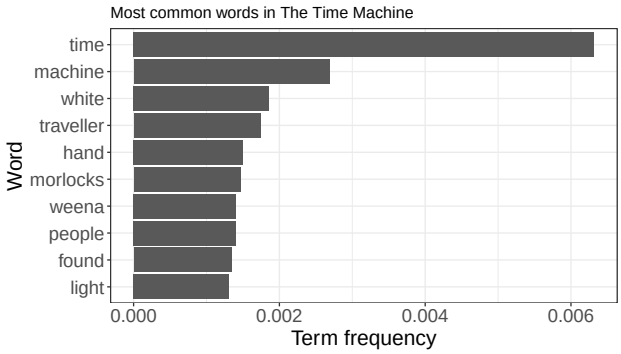
As in Section 3.3.1, we plot the term frequencies for the different words against each other:

```
ggplot( Time_War, aes( x=tf.Time, y=tf.War ) ) + geom_point() +
  geom_abline( color="gray40", linetype=2 ) +
  geom_text( aes(label=word), check_overlap = TRUE, vjust=1 ) +
  coord_trans( x="sqrt", y="sqrt" ) + theme_bw() +
  labs( x="Term Frequency in The Time Machine",
        y="Term Frequency in The War of the Worlds" )
```

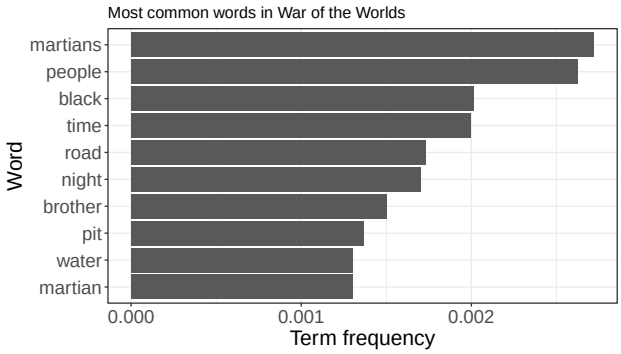


We see that the term frequency of the most common words is quite similar across the two books. The plot also shows some points along the axes, which indicates that some words only occur relatively frequently in one of the books (of they occur in two books at all). To explore some aspects in more detail, we remove stop words and extract the ten most common words (excluding stop words) for each book:

```
Time_War %>%
  anti_join( stop_words, by="word") %>%
  slice_max( tf.Time, n=10 ) %>%
  mutate( word = reorder(word,tf.Time) ) %>%
  ggplot( aes( x=tf.Time, y=word ) ) + geom_col() +
  labs( x="Term frequency", y="Word", title = "Most common words in The Time Machine" )
  theme_bw() +
  theme( axis.title=element_text(size=17), axis.text=element_text(size=15) )
```




```
Time_War %>%
  anti_join( stop_words, by="word") %>%
  slice_max( tf.War, n=10 ) %>%
  mutate( word = reorder(word,tf.War) ) %>%
  ggplot( aes( x=tf.War, y=word ) ) + geom_col() +
  labs( x="Term frequency", y="Word", title = "Most common words in War of the Worlds" )
  theme_bw() +
  theme( axis.title=element_text(size=17), axis.text=element_text(size=15) )
```



We find some words which may be quite specific to one of the two books: "martian", "morlocks" and "weena". The other words are likely less specific and occur in both books. So we may conclude that the two books differ in terms of words describing the setting and the character names, but the other words exhibit a similar term frequency.

b) How do "The Time Machine" and "The War of the Worlds" differ regarding their sentiment?

We take the AFINN sentiment lexicon (but you can also take the Bing lexicon):

```
AFINN <- get_sentiments( "afinn" )
```

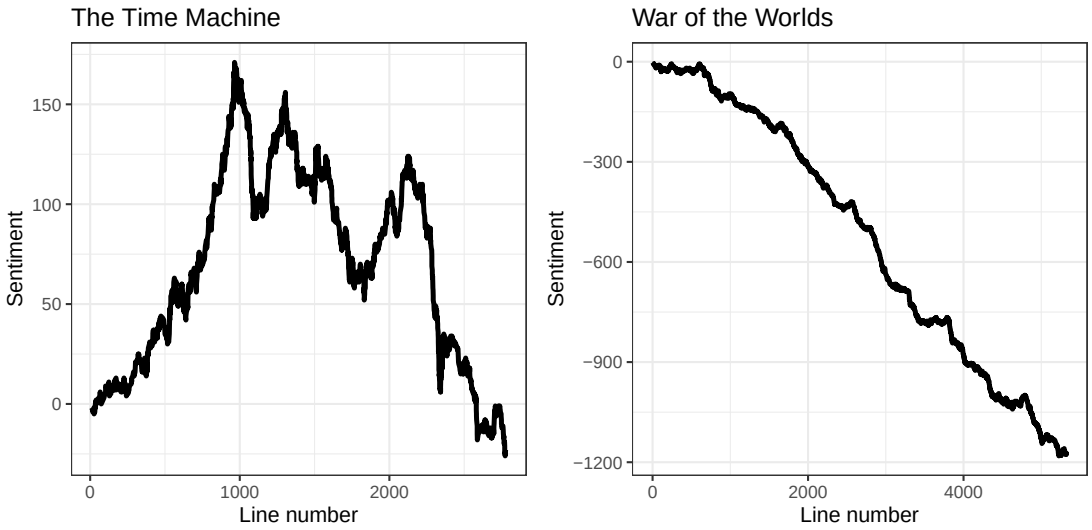
We aim to explore how the AFINN sentiment score, which we take as the sum of sentiments up to that point, evolves across the book. One option (not necessarily the best) is to produce plots as in Section 3.2.2:

```
Time_sentiment <- Time_raw %>%
  mutate( line = row_number() ) %>%
  unnest_tokens( word, text ) %>%
  mutate( word = gsub( "_", "", word ) ) %>%
  inner_join( AFINN, by="word" ) %>%
  mutate( sentiment = cumsum( value ) ) %>%
  ggplot( aes( x=line, y=sentiment ) ) + geom_line( linewidth=1.2 ) +
  theme_bw() + labs( x="Line number", y="Sentiment",
                     title="The Time Machine" )

War_sentiment <- War_raw %>%
  mutate( line = row_number() ) %>%
  unnest_tokens( word, text ) %>%
  mutate( word = gsub( "_", "", word ) ) %>%
  inner_join( AFINN, by="word" ) %>%
  mutate( sentiment = cumsum( value ) ) %>%
  ggplot( aes( x=line, y=sentiment ) ) + geom_line( linewidth=1.2 ) +
  theme_bw() + labs( x="Line number", y="Sentiment",
                     title="War of the Worlds" )

library(patchwork)
```

```
Time_sentiment + War_sentiment
```



Looking at the plots, the analysis suggests that the two books are markedly different in terms of sentiment: The first half of "The Time Machine" seems positive, before turning "negative", while "War of the Worlds" seems to have a "negative" sentiment throughout. However, we have to be careful since the sentiment has been aggregated. For those of you familiar with the books, you may argue that these plots are not reflective of how the sentiment evolves and that it would be better to calculate sentiment for the individual chapters.