week 5 assignmnet

Ajay

2025-05-01

## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

#Exercises

# Load the packages needed to carry out text mining and create visual summaries  
library(gutenbergr) # Lets us download literary works from a public archive  
library(dplyr) # Supports data wrangling such as filtering and arranging

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(tidytext) # Designed for breaking text into analyzable units  
library(stringr) # Handles tasks involving matching and modifying text  
library(ggplot2) # Allows us to visualize patterns and data insights effectively

gutenberg\_metadata

## # A tibble: 72,569 × 8  
## gutenberg\_id title author gutenberg\_author\_id language gutenberg\_bookshelf  
## <int> <chr> <chr> <int> <chr> <chr>   
## 1 1 "The De… Jeffe… 1638 en "Politics/American…  
## 2 2 "The Un… Unite… 1 en "Politics/American…  
## 3 3 "John F… Kenne… 1666 en ""   
## 4 4 "Lincol… Linco… 3 en "US Civil War"   
## 5 5 "The Un… Unite… 1 en "United States/Pol…  
## 6 6 "Give M… Henry… 4 en "American Revoluti…  
## 7 7 "The Ma… <NA> NA en ""   
## 8 8 "Abraha… Linco… 3 en "US Civil War"   
## 9 9 "Abraha… Linco… 3 en "US Civil War"   
## 10 10 "The Ki… <NA> NA en "Banned Books List…  
## # ℹ 72,559 more rows  
## # ℹ 2 more variables: rights <chr>, has\_text <lgl>

#1 Use str\_detect to find the ID of the novel Pride and Prejudice.

# Search for "Pride and Prejudice" in the title column  
pride\_info <- gutenberg\_metadata %>%  
 filter(str\_detect(title, "Pride and Prejudice"))  
  
# View results  
pride\_info

## # A tibble: 7 × 8  
## gutenberg\_id title author gutenberg\_author\_id language gutenberg\_bookshelf  
## <int> <chr> <chr> <int> <chr> <chr>   
## 1 1342 Pride an… Auste… 68 en "Best Books Ever L…  
## 2 20686 Pride an… Auste… 68 en "Best Books Ever L…  
## 3 20687 Pride an… Auste… 68 en "Harvard Classics/…  
## 4 26301 Pride an… Auste… 68 en "Best Books Ever L…  
## 5 37431 Pride an… Auste… 68 en ""   
## 6 37431 Pride an… MacKa… 38839 en ""   
## 7 42671 Pride an… Auste… 68 en "Best Books Ever L…  
## # ℹ 2 more variables: rights <chr>, has\_text <lgl>

#2. We notice that there are several versions. The gutenberg\_works() function filters this table to remove replicates and include only English language works. Read the help file and use this function to find the ID for Pride and Prejudice.

# Get cleaned metadata: English language, no duplicates  
cleaned\_books <- gutenberg\_works()  
  
# Search for "Pride and Prejudice" in the cleaned dataset  
pride\_cleaned <- cleaned\_books %>%  
 filter(str\_detect(title, "Pride and Prejudice"))  
  
# View the results  
pride\_cleaned

## # A tibble: 3 × 8  
## gutenberg\_id title author gutenberg\_author\_id language gutenberg\_bookshelf  
## <int> <chr> <chr> <int> <chr> <chr>   
## 1 1342 Pride an… Auste… 68 en "Best Books Ever L…  
## 2 37431 Pride an… Auste… 68 en ""   
## 3 37431 Pride an… MacKa… 38839 en ""   
## # ℹ 2 more variables: rights <chr>, has\_text <lgl>

#3Use the gutenberg\_download function to download the text for Pride and Prejudice. Save it to an object called book.

# Fetch the complete text of Jane Austen's "Pride and Prejudice" using its unique ID (1342) from an alternate Gutenberg mirror server  
text\_content <- gutenberg\_download(1342, mirror = "http://mirror.csclub.uwaterloo.ca/gutenberg/")  
   
# Gutenberg ID 1342 uniquely identifies this classic novel. The specified mirror is used to speed up the download and reduce the load on the main server.  
  
# Preview the beginning of the downloaded text to make sure it loaded correctly and to get a quick look at the format.  
head(text\_content)

## # A tibble: 6 × 2  
## gutenberg\_id text   
## <int> <chr>   
## 1 1342 " [Illustration:"   
## 2 1342 ""   
## 3 1342 " GEORGE ALLEN"   
## 4 1342 " PUBLISHER"   
## 5 1342 ""   
## 6 1342 " 156 CHARING CROSS ROAD"

# This quick check confirms that the data has been retrieved and lets us see what the raw content looks like before processing.

#4Use the tidytext package to create a tidy table with all the words in the text. Save the table in an object called words

# Load required libraries  
library(tidytext)  
library(dplyr)  
  
# Tokenize the text: convert lines to individual words  
words <- text\_content %>%  
 unnest\_tokens(word, text)  
  
# View the first few words  
head(words)

## # A tibble: 6 × 2  
## gutenberg\_id word   
## <int> <chr>   
## 1 1342 illustration  
## 2 1342 george   
## 3 1342 allen   
## 4 1342 publisher   
## 5 1342 156   
## 6 1342 charing

#5We will later make a plot of sentiment versus location in the book. For this, it will be useful to add a column with the word number to the table.

# Add a word number column to the tidy word table  
words <- words %>%  
 mutate(word\_number = row\_number())  
  
# View the first few rows  
head(words)

## # A tibble: 6 × 3  
## gutenberg\_id word word\_number  
## <int> <chr> <int>  
## 1 1342 illustration 1  
## 2 1342 george 2  
## 3 1342 allen 3  
## 4 1342 publisher 4  
## 5 1342 156 5  
## 6 1342 charing 6

#6. Remove the stop words and numbers from the words object. Hint: use the anti\_join

# Load stop words from tidytext  
data("stop\_words")  
  
# Remove stop words using anti\_join  
clean\_words <- words %>%  
 anti\_join(stop\_words, by = "word")

#7 Now use the AFINN lexicon to assign a sentiment value to each word.

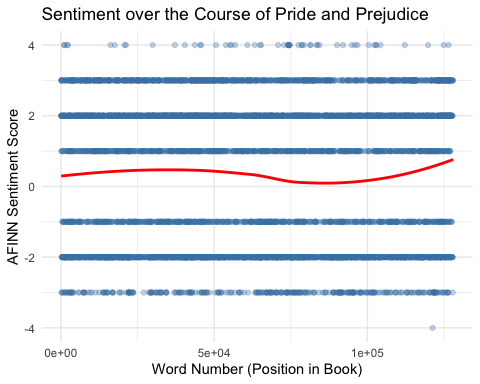
library(tidytext)  
# Load AFINN lexicon from tidytext  
afinn <- get\_sentiments("afinn")  
  
# Join sentiment scores to your cleaned word list  
sentiment\_words <- clean\_words %>%  
 inner\_join(afinn, by = "word")  
  
# View result  
head(sentiment\_words)

## # A tibble: 6 × 4  
## gutenberg\_id word word\_number value  
## <int> <chr> <int> <dbl>  
## 1 1342 fine 87 2  
## 2 1342 loving 92 2  
## 3 1342 loving 96 2  
## 4 1342 love 99 3  
## 5 1342 affection 129 3  
## 6 1342 curious 133 1

#8 Make a plot of sentiment score versus location in the book and add a smoother.

# Load ggplot2  
library(ggplot2)  
  
# Plot sentiment value over word number with a smooth trend line  
ggplot(sentiment\_words, aes(x = word\_number, y = value)) +  
 geom\_point(alpha = 0.3, color = "steelblue") + # Light dots for each word's sentiment  
 geom\_smooth(method = "loess", se = FALSE, color = "red") + # Smooth trend line  
 labs(title = "Sentiment over the Course of Pride and Prejudice",  
 x = "Word Number (Position in Book)",  
 y = "AFINN Sentiment Score") +  
 theme\_minimal()

## `geom\_smooth()` using formula = 'y ~ x'

 #9. Assume there are 300 words per page. Convert the locations to pages and then compute the average sentiment in each page. Plot that average score by page. Add a smoother that appears to go through data.

# Convert word number to page (300 words per page)  
sentiment\_words$page <- sentiment\_words$word\_number %/% 300 + 1  
  
# Calculate average sentiment per page  
avg\_sentiment <- sentiment\_words %>%  
 group\_by(page) %>%  
 summarise(mean\_sentiment = mean(value))  
  
# Plot average sentiment per page with smoother  
ggplot(avg\_sentiment, aes(x = page, y = mean\_sentiment)) +  
 geom\_line(color = "blue") +  
 geom\_smooth(se = FALSE, color = "red") +  
 labs(title = "Sentiment by Page", x = "Page", y = "Average Sentiment")

## `geom\_smooth()` using method = 'loess' and formula = 'y ~ x'

