**Text Analysis with word cloud using Shakespeare Data**

The aim of the following experiment is to perform text analysis with the Shakespearean dataset.

# SOURCE CODE:

library(textdata) library(readr) library(dplyr) library(e1071) library(mlbench)

#Text mining packages library(tm) library(SnowballC) library("wordcloud") library("RColorBrewer") #1 load the data View(shakespeare) str(shakespeare)

#2 Pipe the shakespeare data frame to the next line #use count to find out how may titles/types are there shakespeare %>%

count(title, type)

#3 Load tidytext/tidyverse library(tidytext) library(dplyr)

#4. create and object tidy\_shakespeare # Group by the titles of the plays

# Define a new column linenumber

# Transform the non-tidy text data to tidy text data tidy\_shakespeare <- shakespeare %>% group\_by(title) %>%

mutate(linenumber = row\_number()) %>%

unnest\_tokens(word, text) %>% # Transform the non-tidy text data to tidy text data ungroup()

tidy\_shakespeare

#5.Pipe the tidy Shakespeare data frame to the next line

# Use count to find out how many times each word is used tidy\_shakespeare %>%

count(word, sort = TRUE)

#6.Sentiment analysis of tidy\_shakespeare assin to object shakespeare\_sentiment # Implement sentiment analysis with the "bing" lexicon

shakespeare\_sentiment <- tidy\_shakespeare %>% inner\_join(get\_sentiments("bing"))

#7.shakespeare\_sentiment

# Find how many positive/negative words each play has shakespeare\_sentiment %>%

count(title, sentiment)

#8. Tragedy or comedy from tidy\_shakespeare assign to sentiment\_counts # Implement sentiment analysis using the "bing" lexicon

# Count the number of words by title, type, and sentiment sentiment\_counts <- tidy\_shakespeare %>% inner\_join(get\_sentiments("bing")) %>%

count(title,sentiment) sentiment\_counts

#9. from sentiment\_counts

# Group by the titles of the plays

# Find the total number of words in each play

# Calculate the number of words divided by the total

# Filter the results for only negative sentiment then arrange percentage in asc order

sentiment\_counts %>% group\_by(title) %>% mutate(total = sum(n),

percent = n / total) %>% filter(sentiment == "negative") %>% arrange(percent)

#10 Most common positive and negative words and assign to word\_could # Implement sentiment analysis using the "bing" lexicon

# Count by word and sentiment word\_counts <- tidy\_shakespeare %>% inner\_join(get\_sentiments("bing")) %>% count(word, sentiment)

word\_counts

#11. extract top 10 words from word\_counts and assing to top\_words # Group by sentiment

# Take the top 10 for each sentiment and ungroup it # Make word a factor in order of n

top\_words <- word\_counts %>% group\_by(sentiment) %>% # Group by sentiment

top\_n(10) %>% # Take the top 10 for each sentiment ungroup() %>% # Make word a factor in order of n mutate(word = reorder(word, n))

top\_words

#12 Use aes() to put words on the x-axis and n on the y-axis # Make a bar chart with geom\_col()

# facet\_wrap for sentiments and apply scales as free #Move x to y and y to x

# Use aes() to put words on the x-axis and n on the y-axis library(ggplot2)

ggplot(top\_words, aes(x = word, y = n, fill = sentiment)) + geom\_col(show.legend = FALSE) + facet\_wrap(~sentiment, scales = "free") +

coord\_flip()

#13. from tidy\_shakespeare Calculating a contribution score # Count by title and word

# Implement sentiment analysis using the "afinn" lexicon # Group by title

# Calculate a contribution for each word in each title sentiment\_contributions <- tidy\_shakespeare %>% count(title, word, sort = TRUE) %>% inner\_join(get\_sentiments("afinn")) %>% group\_by(title) %>%

mutate(contribution = (n \* score) / sum(n)) %>% ungroup()

sentiment\_contributions

tidyshakespeare\_reduced <- head(tidy\_shakespeare, n = 5000) #word cloud

corpus = Corpus(VectorSource(tidyshakespeare\_reduced$word)) corpus[[3]][1]

#Conversion to Lowercase

corpus = tm\_map(corpus, PlainTextDocument) corpus = tm\_map(corpus, tolower)

#Removing Punctuation

corpus = tm\_map(corpus, removePunctuation)

#Remove stopwords

corpus = tm\_map(corpus, removeWords, c("cloth", stopwords("english")))

# Stemming

corpus = tm\_map(corpus, stemDocument)

# Eliminate white spaces

corpus = tm\_map(corpus, stripWhitespace) corpus[[2]][1]

DTM <- TermDocumentMatrix(corpus) mat <- as.matrix(DTM)

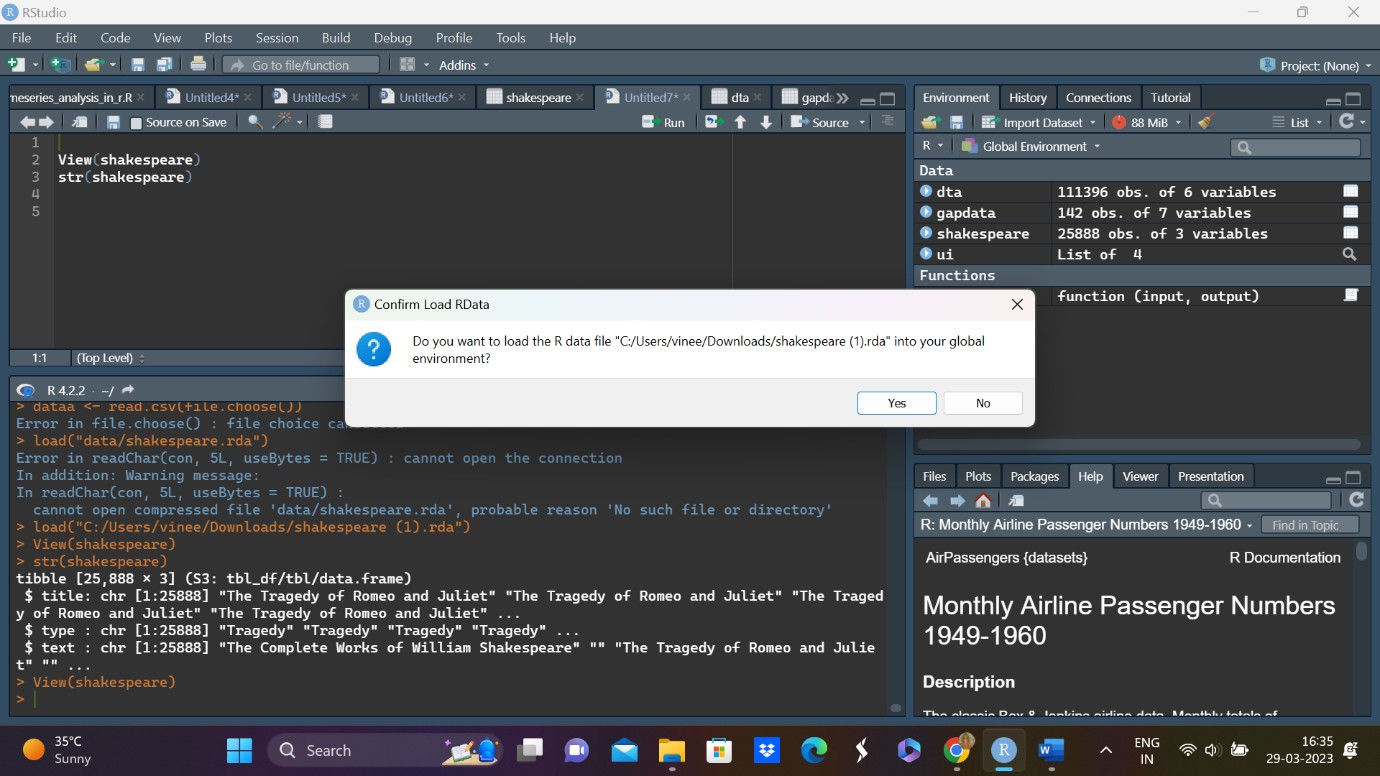
f <- sort(rowSums(mat),decreasing=TRUE) dat <- data.frame(word = names(f),freq=f) head(dat, 5)

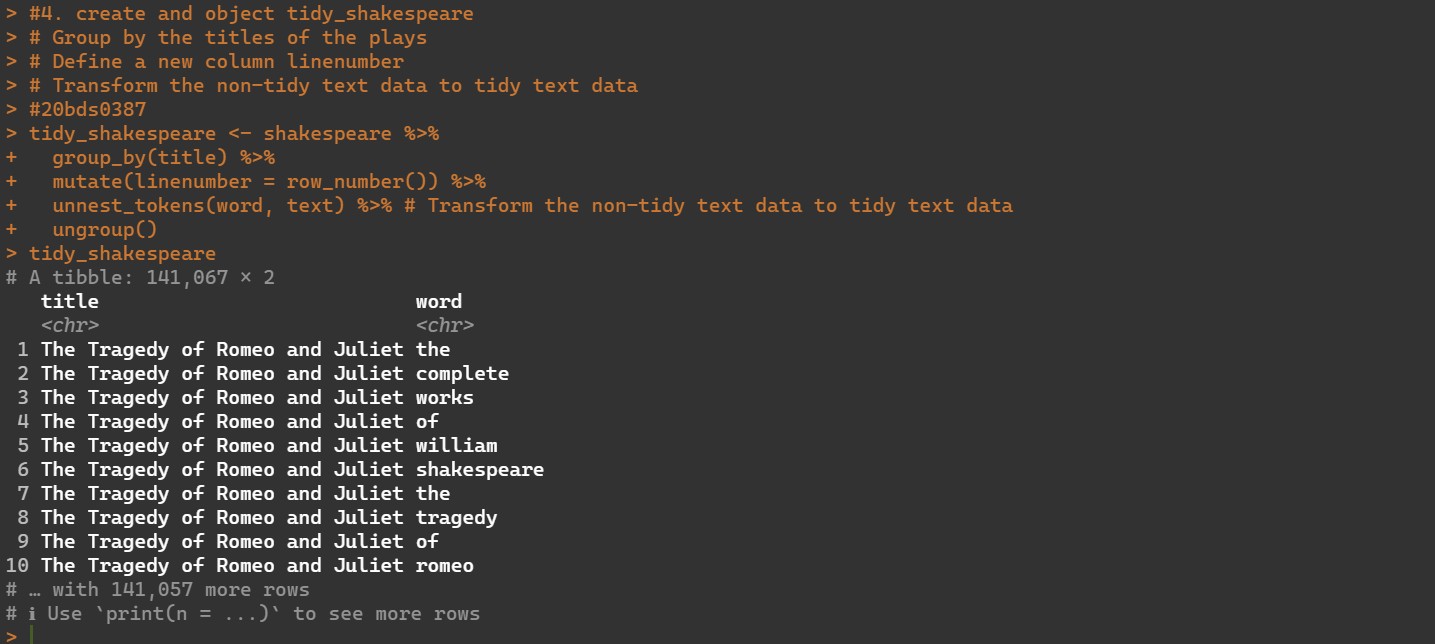
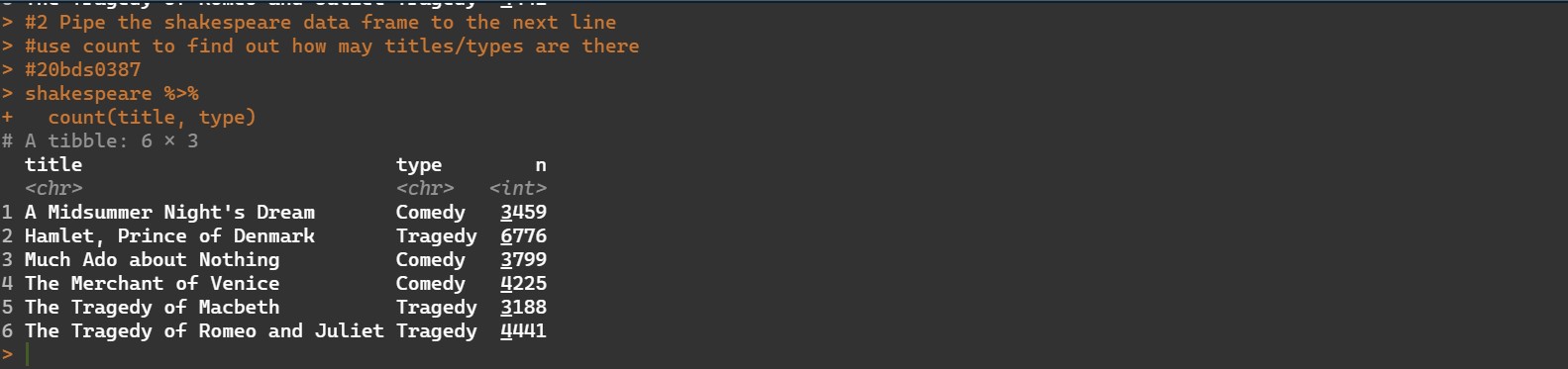
set.seed(100)

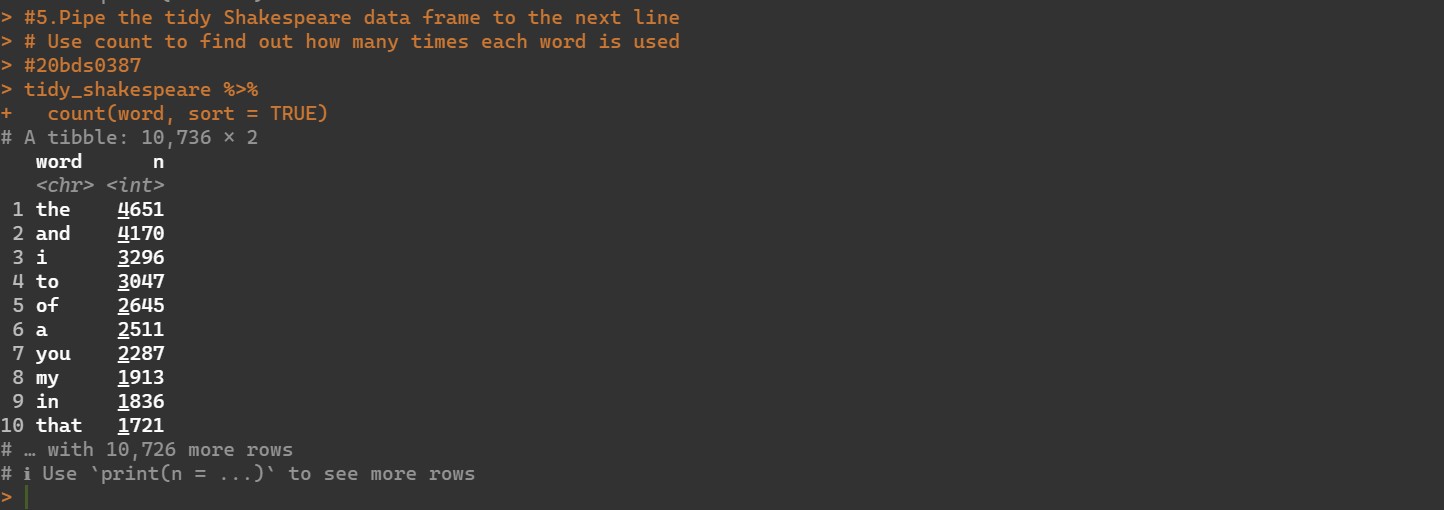
wordcloud(words = dat$word, freq = dat$freq, min.freq = 3, max.words=250, random.order=FALSE, rot.per=0.30, colors=brewer.pal(8, "Dark2"))

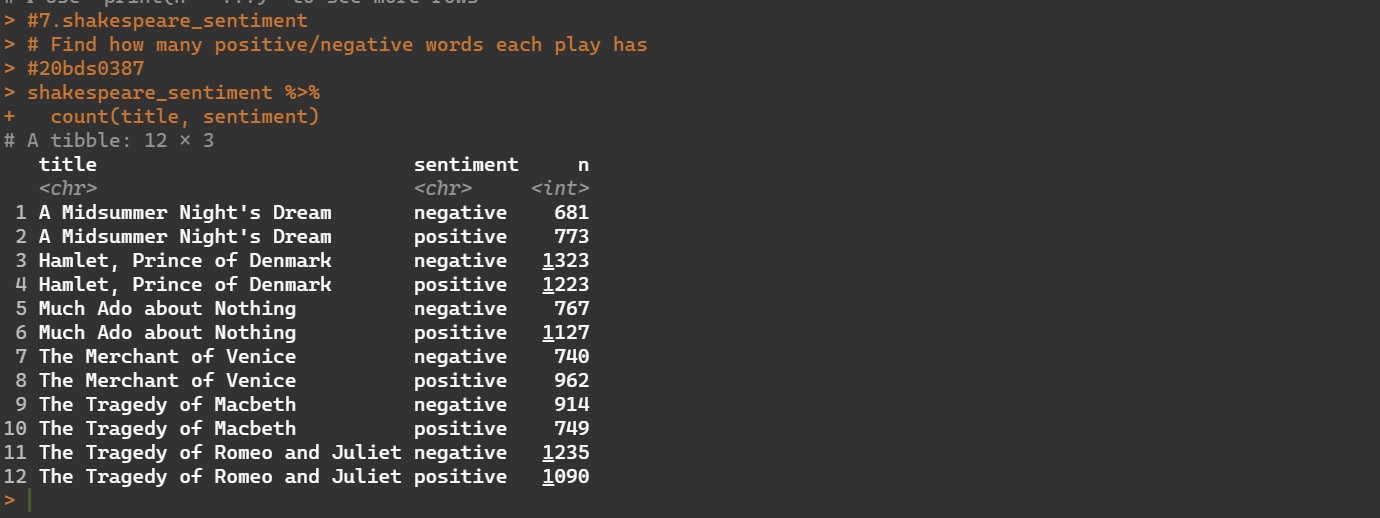
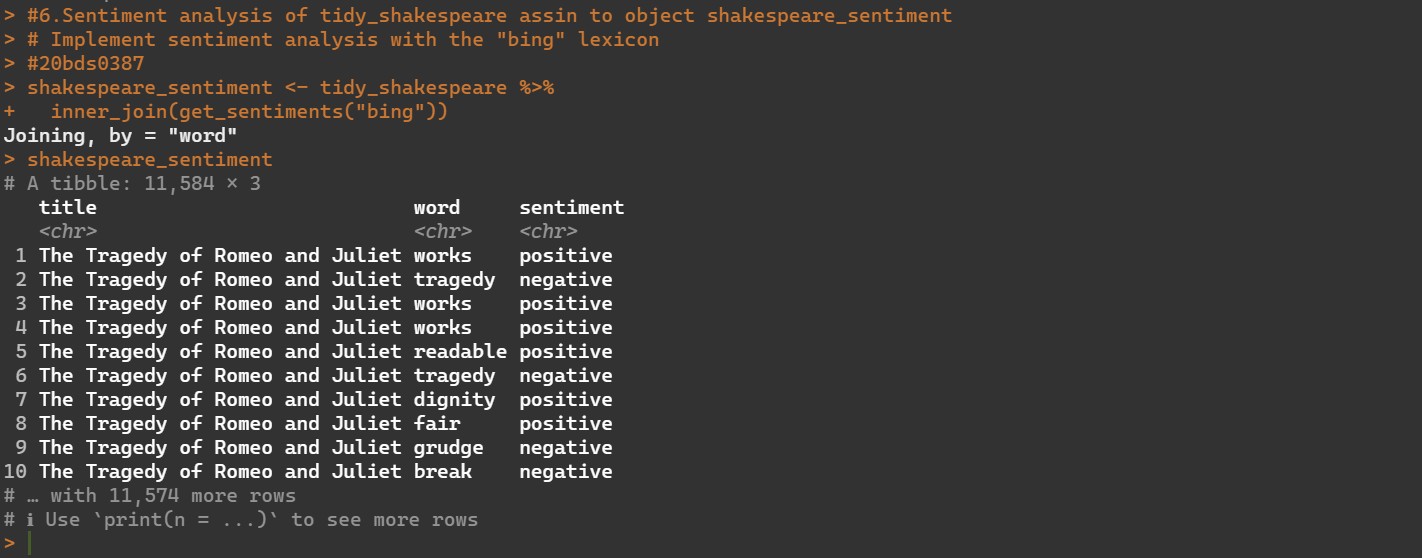
# OUTPUT:

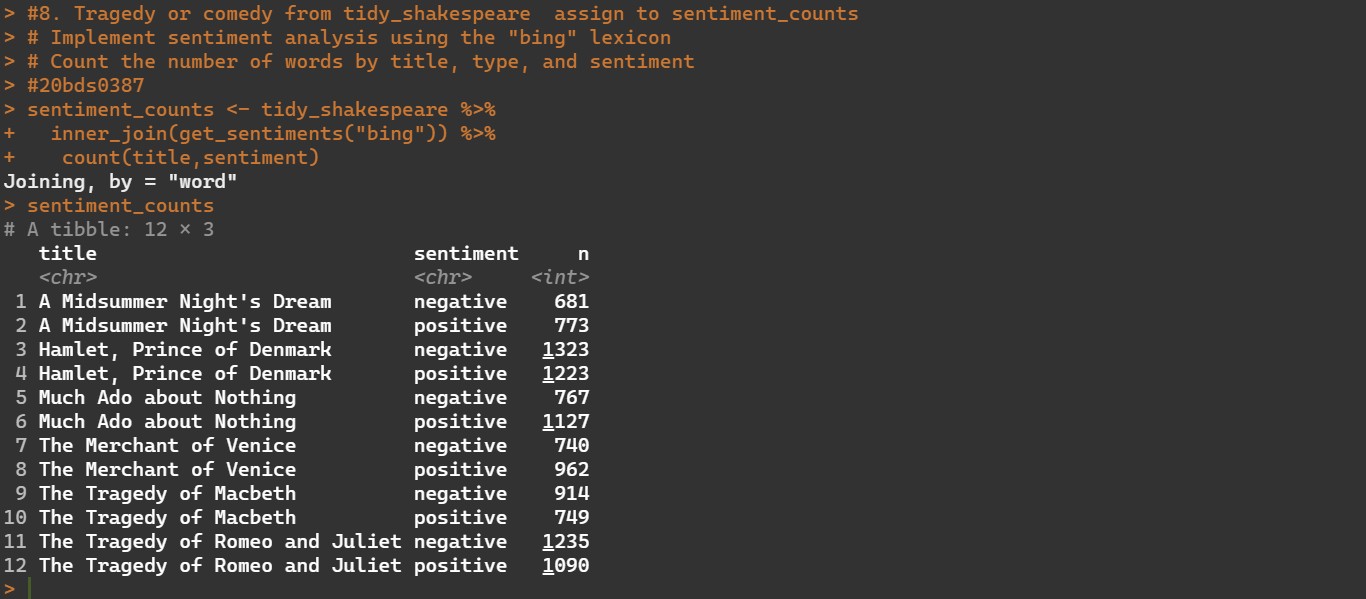
Loading the dataset

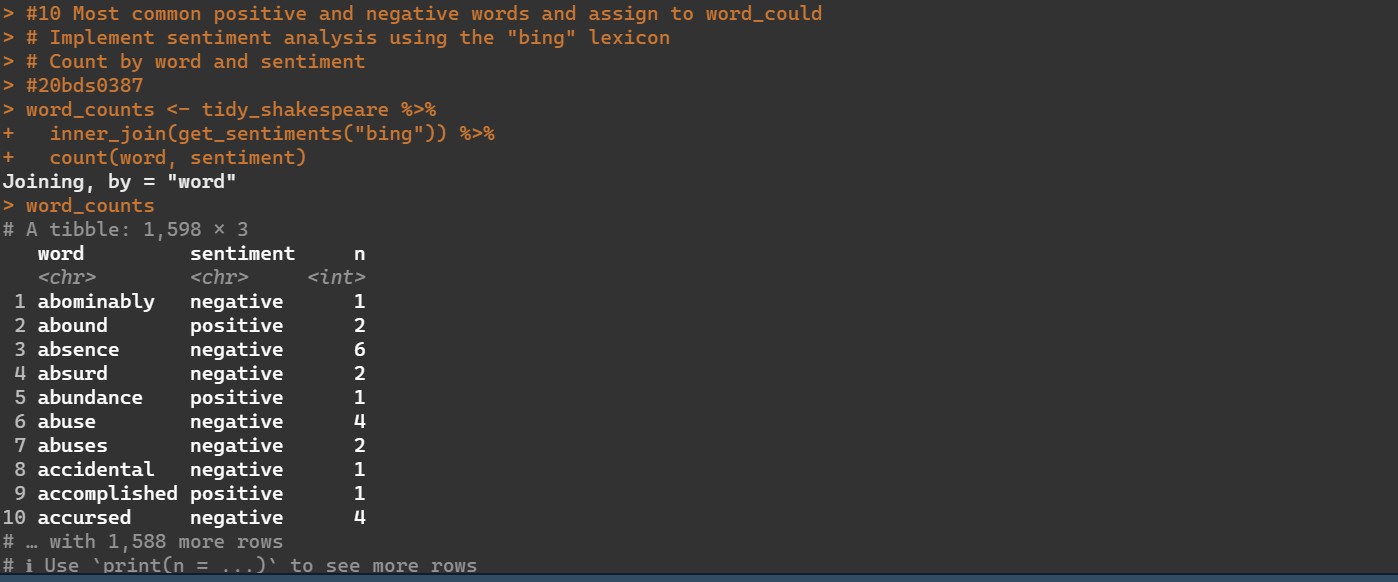
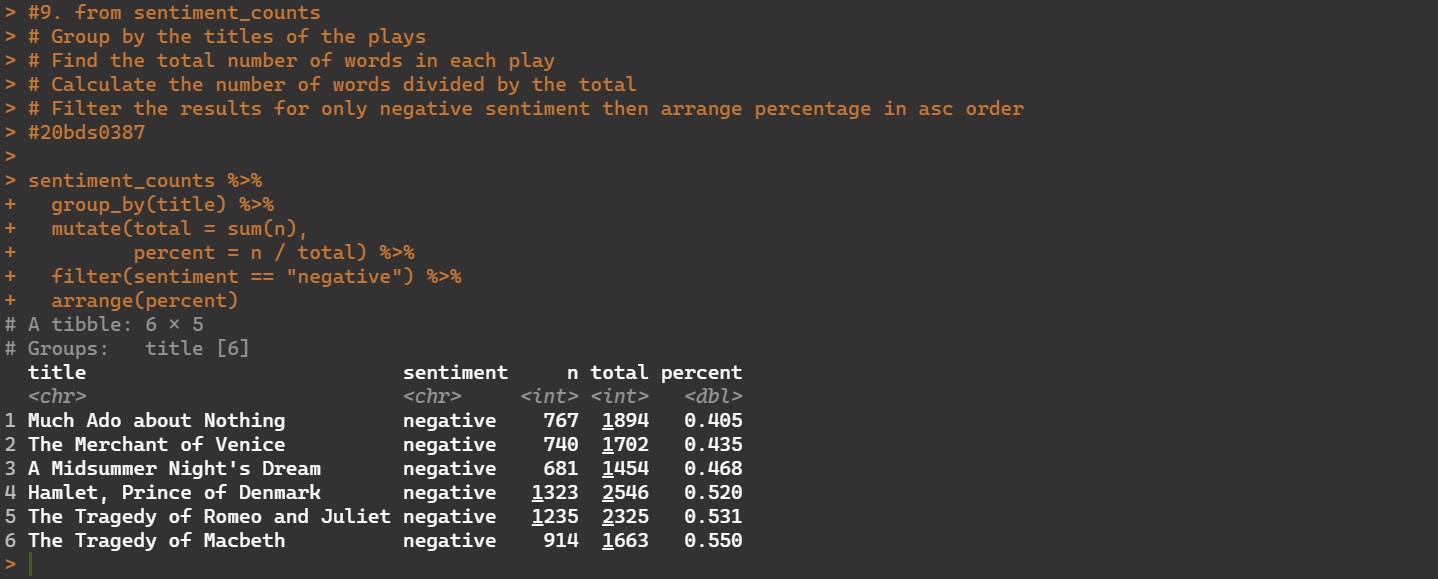


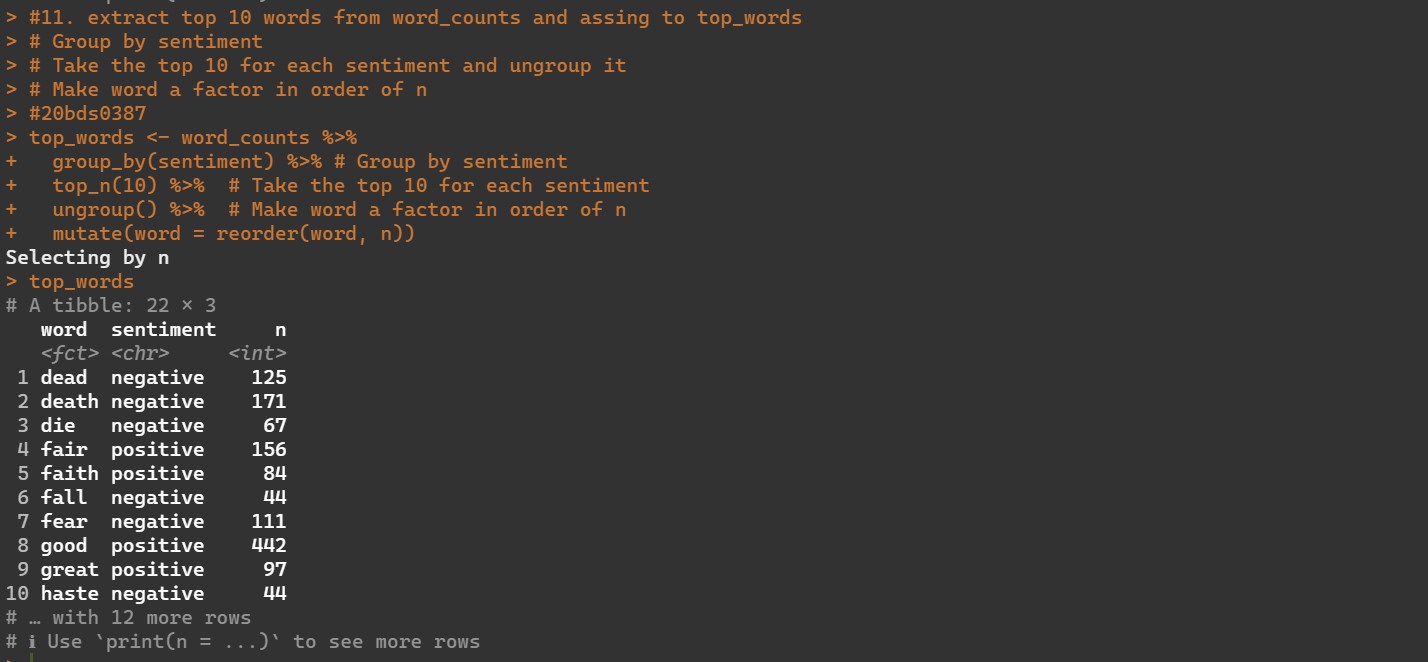


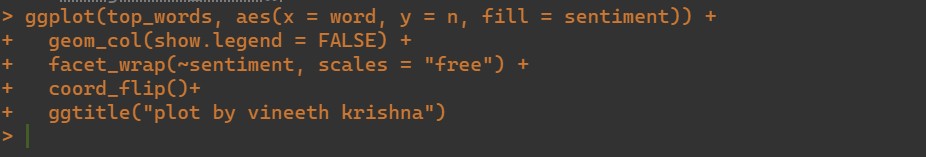


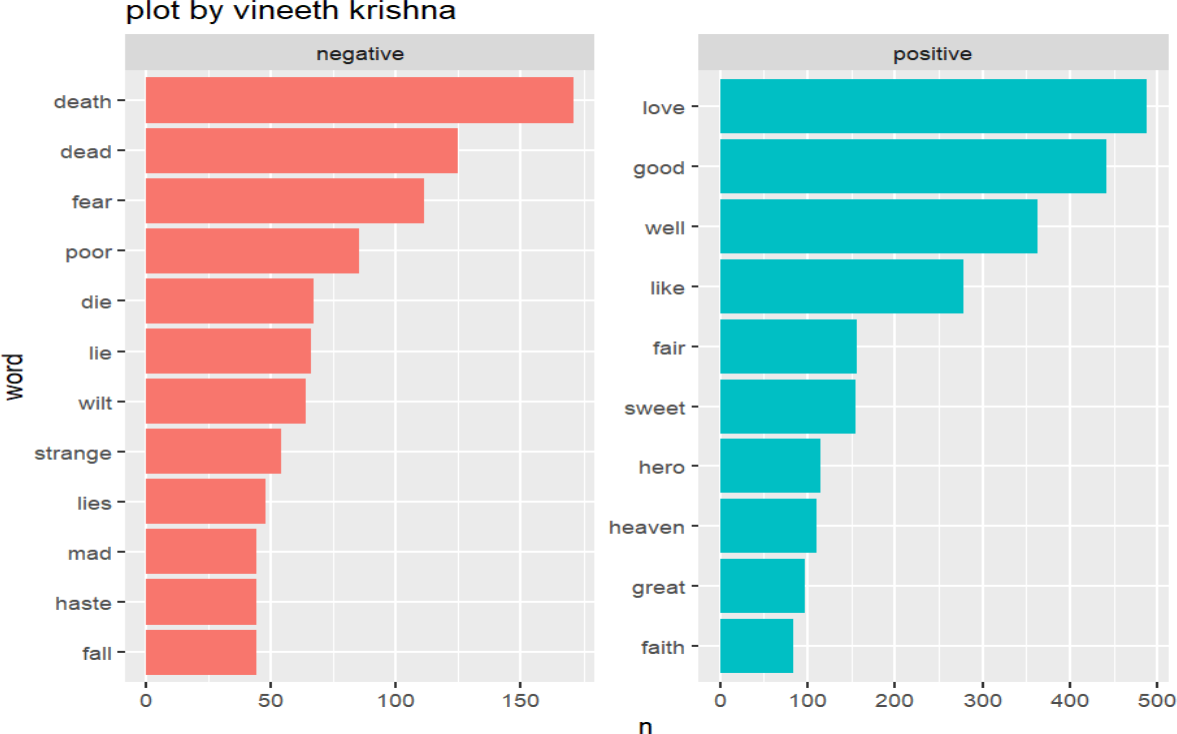


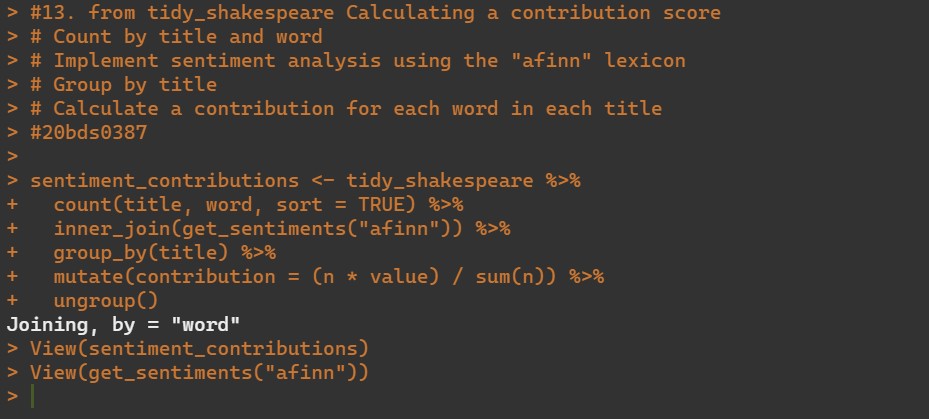


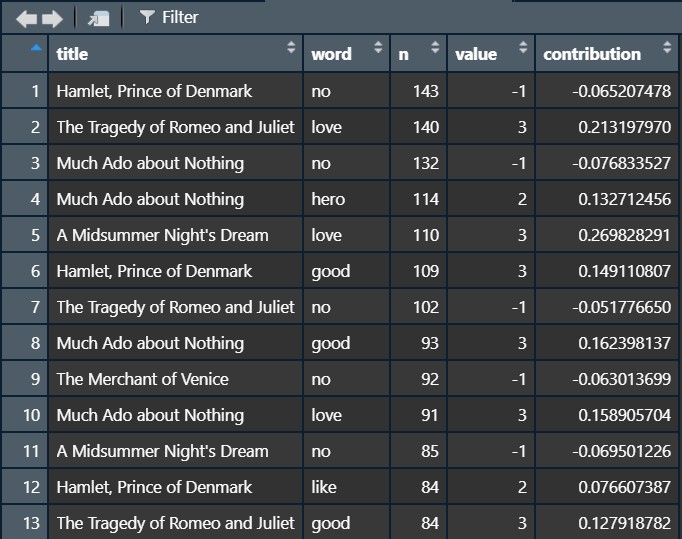


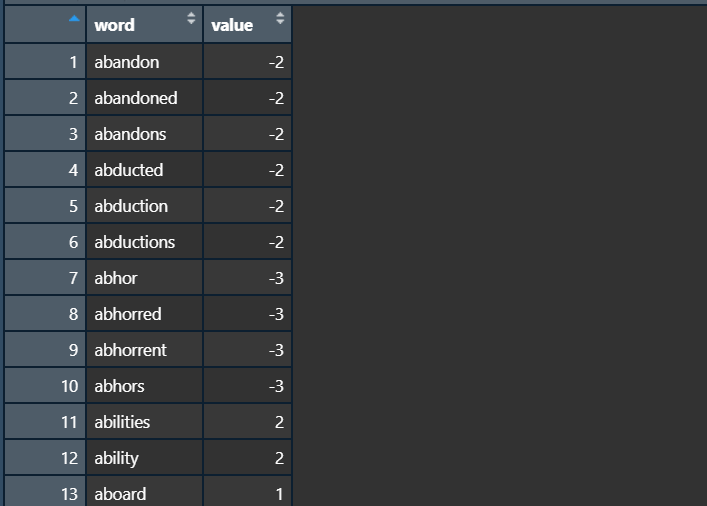












Text Analysis using word cloud:



Due, the last size of the data entries, only the first 5000 entries were considered while constructing this word cloud.



## 