customer reviews

Morsheda Parvin

2023-03-04

The dataset contains customer reviews about retailer's products. It has been anonymized, with references to the company replaced with "retailer". The dataset consists of 23,486 rows, with each row representing a customer review and including 10 feature variables. These variables include the clothing item ID, the reviewer's age, the review title, the review body text, the rating given by the customer (ranging from 1 to 5), whether the customer recommends the product, the number of other customers who found the review positive, and categorical variables for the product's high level division, department name, and class name.

```
# read data from csv file
data <- read.csv("./MS4S09_CW_Data.csv")

# check summary of data
summary(data)
```

```
##        X          Clothing.ID          Age          Title
##  Min.   :    0   Min.   :   0.0   Min.   :18.0   Length:23486
##  1st Qu.: 5871   1st Qu.: 861.0   1st Qu.:34.0   Class :character
##  Median :11742   Median : 936.0   Median :41.0   Mode  :character
##  Mean   :11742   Mean   : 918.1   Mean   :43.2
##  3rd Qu.:17614   3rd Qu.:1078.0   3rd Qu.:52.0
##  Max.   :23485   Max.   :1205.0   Max.   :99.0
##  Review.Text            Rating      Recommended.IND  Positive.Feedback.Count
##  Length:23486       Min.   :1.000   Min.   :0.0000   Min.   :  0.000
##  Class :character   1st Qu.:4.000   1st Qu.:1.0000   1st Qu.:  0.000
##  Mode  :character   Median :5.000   Median :1.0000   Median :  1.000
##                     Mean   :4.196   Mean   :0.8224   Mean   :  2.536
##                     3rd Qu.:5.000   3rd Qu.:1.0000   3rd Qu.:  3.000
##                     Max.   :5.000   Max.   :1.0000   Max.   :122.000
##  Division.Name      Department.Name     Class.Name
##  Length:23486       Length:23486       Length:23486
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
```

```
# check structure of data
str(data)
```

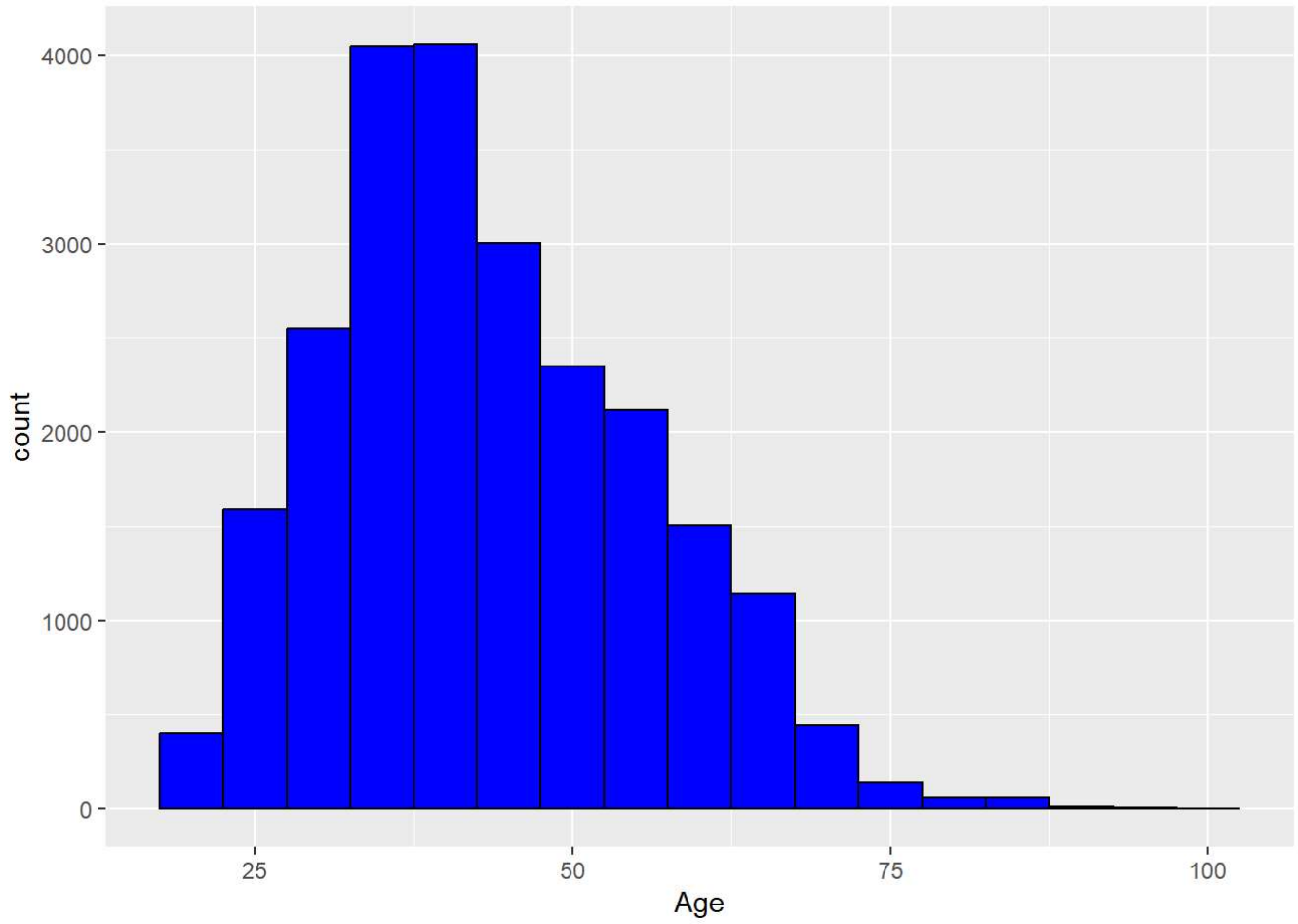
```
## 'data.frame':    23486 obs. of  11 variables:
##  $ X                      : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ Clothing.ID            : int  767 1080 1077 1049 847 1080 858 858 1077 1077 ...
##  $ Age                    : int  33 34 60 50 47 49 39 39 24 34 ...
##  $ Title                  : chr  "" "" "Some major design flaws" "My favorite buy!" ...
##  $ Review.Text            : chr  "Absolutely wonderful - silky and sexy and comfortable"
"Love this dress!  it's sooo pretty.  i happened to find it in a store, and i'm glad i did bc
i never would have"| __truncated__ "I had such high hopes for this dress and really wanted it
to work for me. i initially ordered the petite small "| __truncated__ "I love, love, love thi
s jumpsuit. it's fun, flirty, and fabulous! every time i wear it, i get nothing but great com
pliments!" ...
##  $ Rating                 : int  4 5 3 5 5 2 5 4 5 5 ...
##  $ Recommended.IND        : int  1 1 0 1 1 0 1 1 1 1 ...
##  $ Positive.Feedback.Count: int  0 4 0 0 6 4 1 4 0 0 ...
##  $ Division.Name          : chr  "Initmates" "General" "General" "General Petite" ...
##  $ Department.Name        : chr  "Intimate" "Dresses" "Dresses" "Bottoms" ...
##  $ Class.Name             : chr  "Intimates" "Dresses" "Dresses" "Pants" ...
```

```
# remove missing rows of the data
data <- na.omit(data)
```

```
library(tidytext)
library(ggplot2)

# Summary statistics for age
summary(data$Age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    18.0    34.0    41.0    43.2    52.0    99.0
```

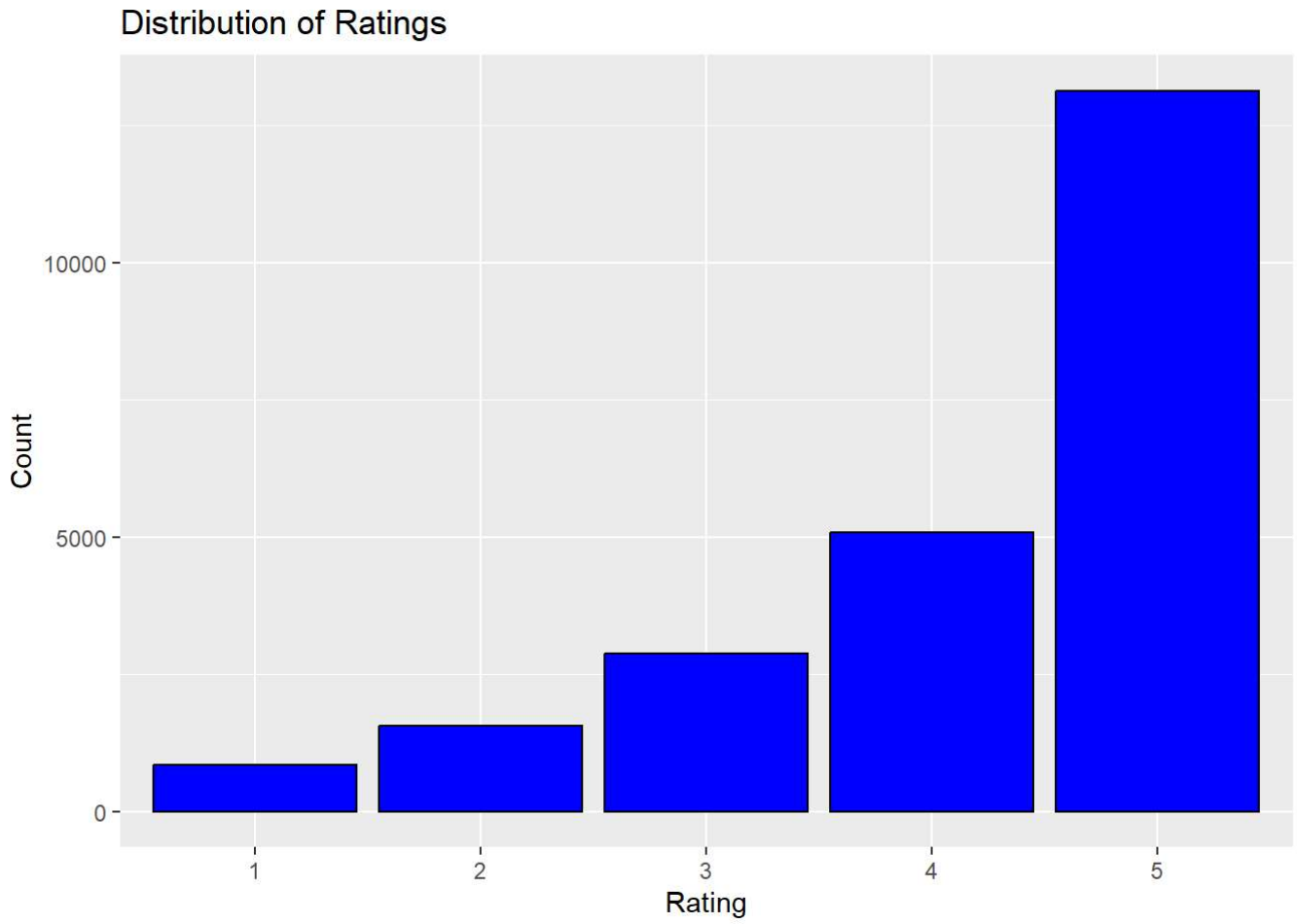
```
# Histogram of age
ggplot(data, aes(x=Age)) + geom_histogram(binwidth = 5, color="black", fill="blue")
```




```
# Summary statistics for rating
summary(data$Rating)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   4.000   5.000   4.196   5.000   5.000
```

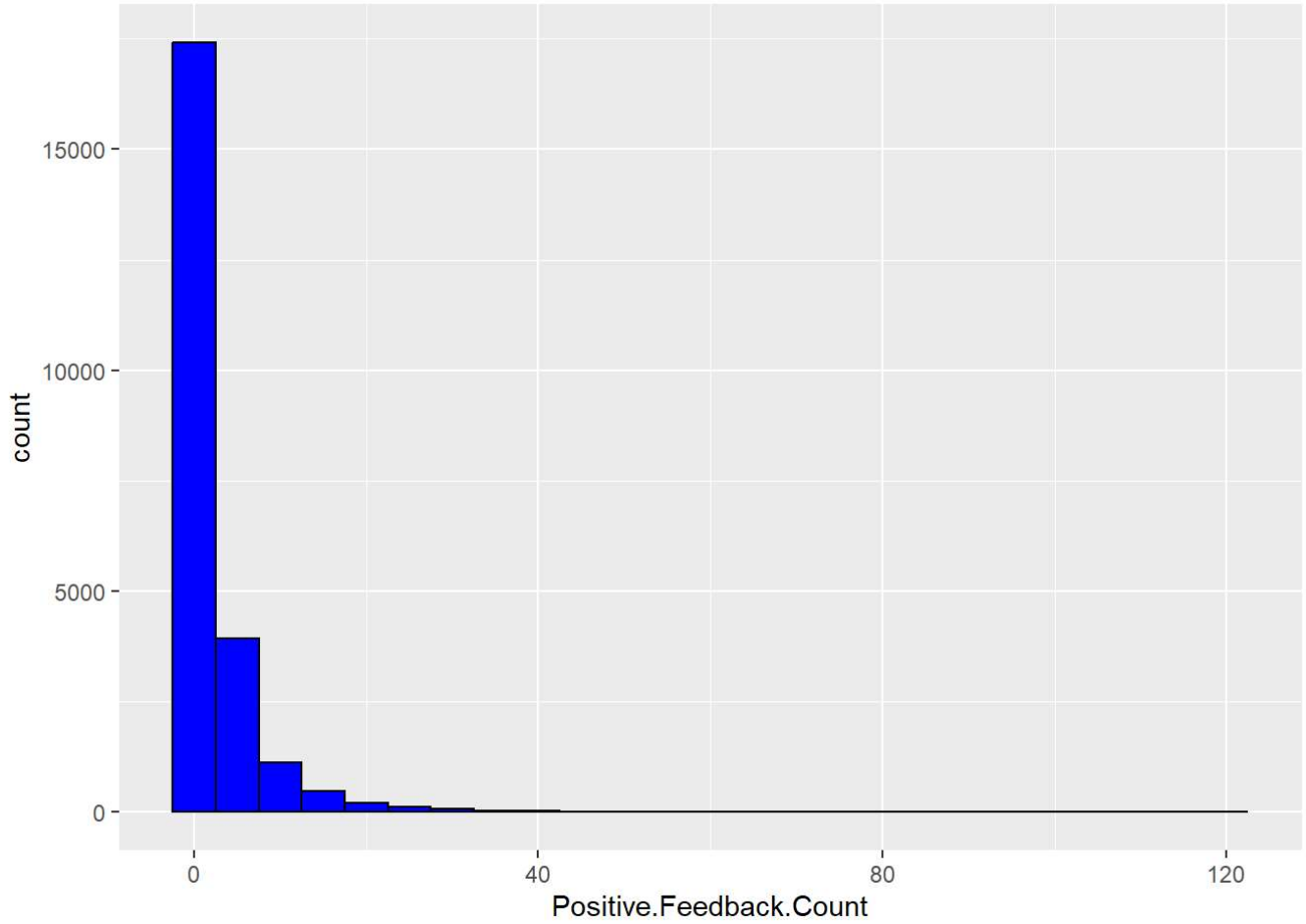
```
# Bar chart of rating
ggplot(data, aes(x=factor(Rating))) +
  geom_bar(fill="blue", color="black") +
  xlab("Rating") +
  ylab("Count") +
  ggtitle("Distribution of Ratings")
```




```
# Summary statistics for feedback count
summary(data$Positive.Feedback.Count)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   1.000   2.536   3.000 122.000
```

```
# Histogram of feedback count
ggplot(data, aes(x=Positive.Feedback.Count)) +
  geom_histogram(binwidth = 5, color="black", fill="blue")
```



The summary function provides a statistical summary of the numerical variables in the dataset. For instance, the “Age” variable has a minimum value of 18 and a maximum value of 99. The mean is 43.2, suggesting that the average customer who left a review is in their forties.For the “Rating” variable, the minimum value is 1 and the maximum value is 5, indicating that the rating scale ranges from 1 to 5. The median is 5, which suggests that most of the reviews are positive.The “Positive.Feedback.Count” variable has a minimum value of 0 and a maximum value of 122. The mean is 2.536, indicating that most reviews do not have a large number of positive feedback counts that are already shown on the bar charts and histrogram.

The structure function provides information about the data types and the number of observations in the dataset. The dataset contains 23486 observations and 11 variables, including three categorical variables and 8 numeric variables.

Task A – Text Mining

This is a product review dataset with information about the product, the reviewer, and their review.

To perform text mining on the review text, we would need to pre-process the text by removing stop words, punctuation, and any unnecessary words that do not add value to the analysis. We could also perform stemming or lemmatization to reduce words to their root form. After pre-processing the text, we could use techniques such as word frequency analysis, sentiment analysis, and topic modeling to draw insights from the text data.We would also need to use appropriate visualization techniques to effectively communicate the insights to the audience.To perform text mining on the review text, first we need to create a corpus of text. A corpus is a collection of text documents that we can analyze using text mining techniques.

In R, we can create a corpus using the tm package. Here is an example code to create a corpus of the review text:

```
#Load the data as a corpus

library(tm)
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

```
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
data <- read.csv("./MS4S09_CW_Data.csv")
# remove missing rows of the data
data <- na.omit(data)
docs <- Corpus(VectorSource(data$Review.Text))

getTransformations()
```

```
## [1] "removeNumbers"     "removePunctuation" "removeWords"
## [4] "stemDocument"      "stripWhitespace"
```

In the above code, data$Review.Text refers to the column of review text in the dataset. VectorSource is used to create a source object for the corpus, which is then passed to the VCorpus function to create a corpus of text.

It loads the data from the CSV file into the data variable. It then checks the data type of the Review.Text column using the is.character function, and prints the indices of any non-character rows.

Next, it removes any rows where the Review.Text column is empty using the subset function. It then creates a corpus from the cleaned data using the Corpus function from the tm package.

The code then performs several pre-processing steps on the corpus, including removing special characters and punctuation, removing stopwords, and removing additional words specific to this dataset.

```
library(tm)     # For text mining
library(dplyr)  # For data manipulation
```

```
##
## 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
```

```r
library(tidyr)  # For data manipulation

# Load the data from CSV file


# Create a corpus from the cleaned data
docs <- Corpus(VectorSource(as.character(data$Review.Text)))

# Perform pre-processing steps
docs <- tm_map(docs, content_transformer(gsub), pattern = "/", replacement = " ")
```

```
## Warning in tm_map.SimpleCorpus(docs, content_transformer(gsub), pattern = "/",
## : transformation drops documents
```

```r
toSpace <- content_transformer(function(x, pattern) gsub(pattern, " ", x))
docs <- tm_map(docs, toSpace, "@")
```

```
## Warning in tm_map.SimpleCorpus(docs, toSpace, "@"): transformation drops
## documents
```

```r
docs <- tm_map(docs, toSpace, "\\|")
```

```
## Warning in tm_map.SimpleCorpus(docs, toSpace, "\\|"): transformation drops
## documents
```

```r
docs <- tm_map(docs, content_transformer(tolower))
```

```
## Warning in tm_map.SimpleCorpus(docs, content_transformer(tolower)):
## transformation drops documents
```

```r
docs <- tm_map(docs, removeNumbers)
```

```
## Warning in tm_map.SimpleCorpus(docs, removeNumbers): transformation drops
## documents
```

```r
docs <- tm_map(docs, removeWords, stopwords("english"))
```

```
## Warning in tm_map.SimpleCorpus(docs, removeWords, stopwords("english")):
## transformation drops documents
```

```r
docs <- tm_map(docs, removePunctuation)
```

```
## Warning in tm_map.SimpleCorpus(docs, removePunctuation): transformation drops
## documents
```

```
docs <- tm_map(docs, stripWhitespace)
```

```
## Warning in tm_map.SimpleCorpus(docs, stripWhitespace): transformation drops
## documents
```

```
# Remove additional stopwords
docs <- tm_map(docs, removeWords, c("xxs","xs", "cami","hem","thats","isnt","tts","tee","iv
e","cant","doesnt","didnt","dont","lbs","theres","wont","wouldnt","youre","couldnt"))
```

```
## Warning in tm_map.SimpleCorpus(docs, removeWords, c("xxs", "xs", "cami", :
## transformation drops documents
```

```
getTransformations()
```

```
## [1] "removeNumbers"     "removePunctuation" "removeWords"
## [4] "stemDocument"      "stripWhitespace"
```

In the above code, data-cleaned$Review.Text refers to the column of review text in the dataset. VectorSource is used to create a source object for the corpus, which is then passed to the VCorpus function to create a corpus of text.A corpus is created from the cleaned data using the Corpus() and VectorSource() functions from the tm package. Pre-processing steps are performed on the corpus to clean the data. These steps include removing special characters, converting text to lowercase, removing numbers, removing stop words, removing punctuation, and removing additional stopwords. Finally, the resulting corpus is inspected using the inspect() function from the tm package to ensure that the pre-processing steps were performed correctly.

```
library(tm)
library(SnowballC)
# view the top 100 most frequent words
dtm <- TermDocumentMatrix(docs)
m <- as.matrix(dtm)
v <- sort(rowSums(m),decreasing=TRUE)
d <- data.frame(word = names(v),freq=v)
head(d, 100)
```

	word <chr>	freq <dbl>
dress	dress	10542
love	love	8926
size	size	8728
top	top	7385
fit	fit	7283
like	like	6996
wear	wear	6427
great	great	6084

	word <chr>	freq <dbl>
just	just	5575
fabric	fabric	4781

1-10 of 100 rows Previous **1** 2 3 4 5 6 … 10 Next

This code block loads the tm and SnowballC libraries and creates a TermDocumentMatrix object from the docs corpus. The TermDocumentMatrix object represents the word frequency matrix of the corpus where the rows correspond to words and the columns correspond to the documents in the corpus.

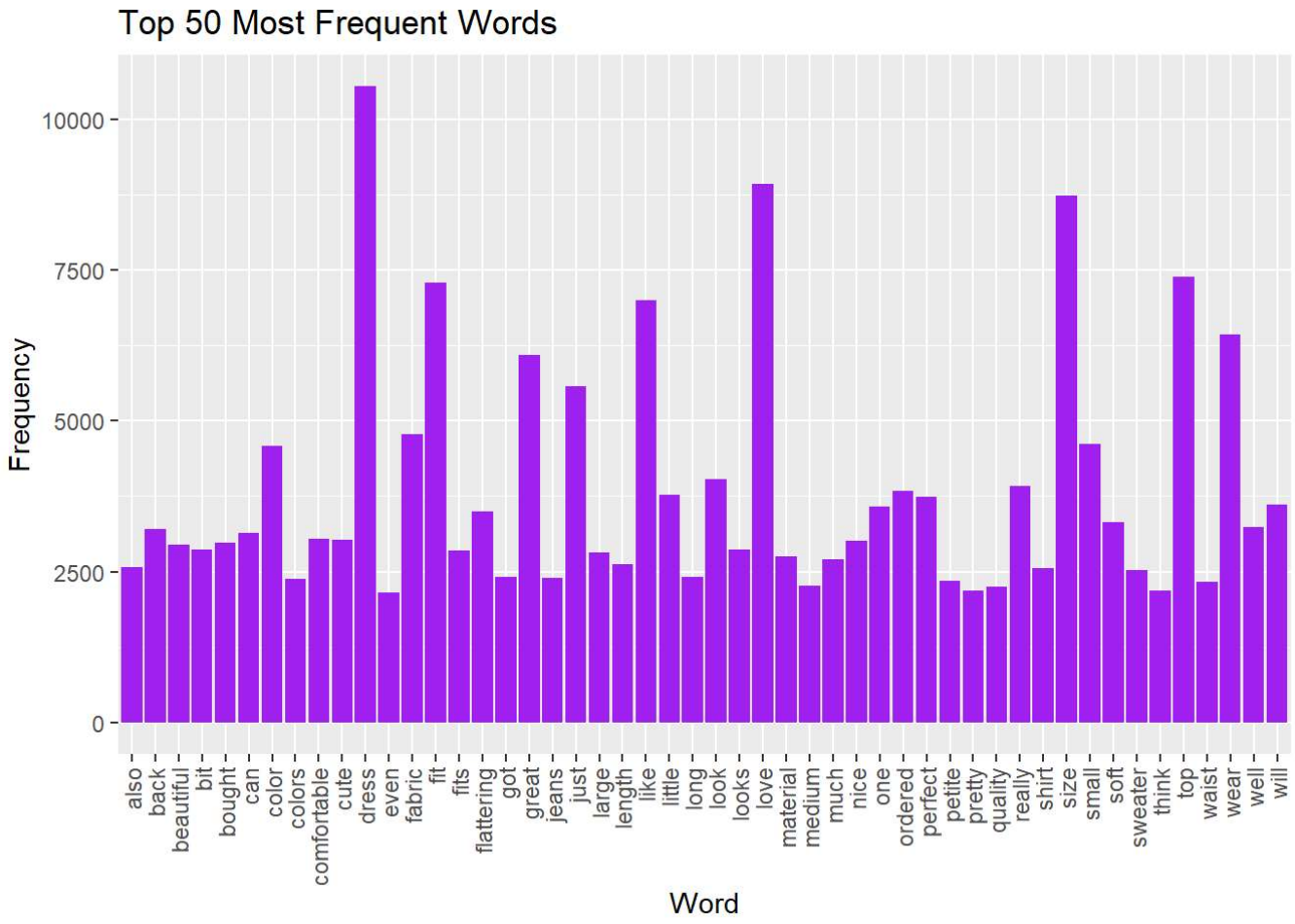
Next, the matrix is converted to a data frame (d) where the rows are sorted in descending order based on the frequency of occurrence of each word. Finally, the top 100 most frequent words are printed using the head function.

The resulting data frame d contains the top 100 most frequent words in the corpus, sorted in descending order of frequency. The first column of the data frame shows the word itself, and the second column shows the frequency of that word in the corpus. The output you provided shows the first six rows of the data frame, which display the word, the word again, and the frequency of that word in the corpus, respectively.This output shows the 100 most frequent words in the corpus, along with their frequency count. From this output, it appears that “dress” is the most frequent word in the corpus with a frequency count of 10,542, followed by “love” with a count of 8,926, and “size” with a count of 8,728. This suggests that the reviews may be about dresses and that size and fit are important factors that people are mentioning in their reviews. However, body and going are the least frequent words in the corpus with a frequency count of 1185 and 1182 respectively.

```
# Create a bar chart of the top 50 most frequent terms
library(tm)
library(SnowballC)
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
library(dplyr)
library(RColorBrewer)
library(ggplot2)

# Create a data frame with the top 50 most frequent terms
d <- data.frame(word=names(v), freq=v)
top_words <- head(d, 50)


ggplot(top_words, aes(x=word, y=freq)) +
  geom_bar(stat="identity", fill="purple") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) +
  labs(title = "Top 50 Most Frequent Words", x="Word", y="Frequency")
```




```
# Create a word cloud of the top 100 most frequent terms
wordcloud(words = d$word, freq = d$freq,scale = c(2.5, 0.5), min.freq = 10,
          max.words=100, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2"))
```

From the barchart and wordcloud it is easily be observed that dress,love,size, top……are the most The most frequent words in the review text appear to be positive in nature, such as “love”, “comfortable”, and “perfect”. This suggests that overall, customers tend to have positive experiences with the clothing products. Otherwise, the most frequent words related to specific product categories, such as “dress”, “top”, and “shirt”. This could indicate that certain types of clothing items are more popular or frequently reviewed than others.

Text miniing by tidy text format

```
library(dplyr)
library(conflicted)
library(SnowballC)
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ forcats   1.0.0     ✓ readr     2.1.4
## ✓ lubridate 1.9.2     ✓ stringr   1.5.0
## ✓ purrr     1.0.1     ✓ tibble    3.1.8
```

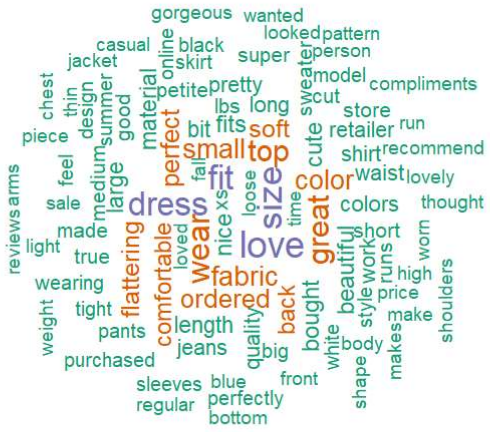
```r
library(tidytext)
library(wordcloud)
library(RColorBrewer)
library(ggplot2)
library(stringr)

# Load the data from CSV file
data <- read.csv("./MS4S09_CW_Data.csv")

# Remove rows with missing values in 'Review.Text' column
Data_cleaned <- na.omit(data)


# Convert character vector to data frame
Data_cleaned <- as.data.frame(Data_cleaned)

# Convert 'Review.Text' column to character type
Data_cleaned$Review.Text <- as.character(Data_cleaned$Review.Text)

Data_cleaned
```

	X <int>	Clothing.ID <int>	A.. <int>	Title <chr>
1	0	767	33	
2	1	1080	34	
3	2	1077	60	Some major design flaws
4	3	1049	50	My favorite buy!
5	4	847	47	Flattering shirt
6	5	1080	49	Not for the very petite
7	6	858	39	Cagrcoal shimmer fun
8	7	858	39	Shimmer, surprisingly goes with lots
9	8	1077	24	Flattering
10	9	1077	34	Such a fun dress!

```
# Tokenize the 'Review.Text' column
tidy_reviews <- Data_cleaned %>%
  select(Review.Text) %>%
  unnest_tokens(word, Review.Text, drop = FALSE) %>%
  mutate(word = gsub("[/@\\|]", " ", word),                    # replace /, @ and | with spac
es
         word = tolower(word),                                   # convert to lowercase
         word = gsub("\\d+", "", word),                           # remove digits
         word = removeWords(word, stopwords::stopwords(source = "smart")), # remove stop word
s
         word = gsub("[[:punct:]]", "", word),                    # remove punctuation
         word = trimws(word))                                     # trim leading/trailing whi
tespace

# Count the frequency of each word
distinct_tidy_reviews <- tidy_reviews %>%
  distinct() %>%
  count(word,sort = TRUE)

# Remove words that are either stop words or unimportant for analysis
tidy_reviews <- tidy_reviews %>%
  anti_join(data.frame(word = c("xxs", "sooo","oops","xs","xxsp","bc","midi","cami","xl","d
d", "hem","ft", "thats","lb","ve", "waas","hte","teh","isnt", "tts", "ag","tee", "ive", "can
t", "doesnt", "didnt", "wil","tees","xsp", "idecide","dont","im","ish", "lbs","esp", "there
s", "wont", "wouldnt", "youre", "couldnt")), by = "word")



distinct_tidy_reviews
```

word <chr>	n <int>
	22617
love	7419
size	6593
fit	6197
dress	6082
wear	5530
top	5322
great	5198
fabric	4151
color	4059

```
# Show the top 20 most common words
top_20_words <- distinct_tidy_reviews %>%
  slice_head(n = 20) %>%
  pull(word)

# Create the wordcloud
wordcloud(words = distinct_tidy_reviews$word, freq = distinct_tidy_reviews$n, scale = c(2.5,
0.5),
          min.freq = 1, max.words = 100, random.order = FALSE, rot.per = 0.35,
          colors = brewer.pal(8, "Dark2"))
```




```
# Load the data from CSV file
data <- read.csv("./MS4S09_CW_Data.csv")

# Check the data type of 'Review.Text' column
is_char <- sapply(data$Review.Text, is.character)

# Print the indices of non-character rows
which(!is_char)
```

```
## named integer(0)
```

```r
library(tidyverse)

# Load the data from CSV file
data <- read.csv("./MS4S09_CW_Data.csv")

# Clean the data by removing empty reviews
data_cleaned <- data %>%
  mutate(Review.Text = na_if(Review.Text, "")) %>%
  drop_na(Review.Text)

# Check if the data has been cleaned
sum(data_cleaned$Review.Text == "")
```

```
## [1] 0
```

Task B – Sentiment Analysis

```r
library(dplyr)
library(tidytext)
library(tidyr)


library(conflicted)

# Load the data from CSV file
data <- read.csv("./MS4S09_CW_Data.csv")

# Clean the data by removing empty reviews
data_cleaned <- data %>%
  mutate(Review.Text = na_if(Review.Text, "")) %>%
  drop_na(Review.Text)

# Check if the data has been cleaned
sum(data_cleaned$Review.Text == "")
```

```
## [1] 0
```

```
# Convert 'Review.Text' column to character type
data_cleaned$Review.Text <- as.character(data_cleaned$Review.Text)

# Tokenize the text column
tidy_reviews <- data_cleaned %>%
  unnest_tokens(word, Review.Text, drop = FALSE) %>%
  select(word, Clothing.ID, Age, Positive.Feedback.Count,Division.Name,Department.Name,Class.
Name,Rating)

# Perform pre-processing steps
tidy_reviews <- tidy_reviews %>%
  mutate(word = gsub("[/@\\|]", " ", word),,  # replace /, @ and | with spaces
         word = tolower(word),                                    # convert to lowercase
         word = gsub("\\d+", "", word),                            # remove digits
         word = removeWords(word, stopwords::stopwords(source = "smart")), # remove stop word
s
         word = gsub("[[:punct:]]", "", word),                     # remove punctuation
         word = trimws(word))                                      # trim leading/trailing whi
tespace
# Remove words that are either stop words or unimportant for analysis
tidy_reviews <- tidy_reviews %>%
  anti_join(data.frame(word = stopwords::stopwords(source = "smart")), by = c("word" = "wor
d")) %>%
  anti_join(data.frame(word = c("xxs", "xs", "cami", "hem", "thats", "isnt", "tts", "tee", "i
ve", "cant", "doesnt", "didnt", "dont", "lbs", "theres", "wont", "wouldnt", "youre", "couldn
t")), by = "word")
```

```
library(dplyr)
library(tidytext)

# Perform sentiment analysis using AFINN lexicon
distinct_tidy_reviews_sentiment <- tidy_reviews %>%
  inner_join(get_sentiments("afinn")) %>%
  group_by(Clothing.ID, Rating) %>%
  summarize(sentiment_score = sum(value))
```

```
## Joining with `by = join_by(word)`
## `summarise()` has grouped output by 'Clothing.ID'. You can override using the
## `.groups` argument.
```

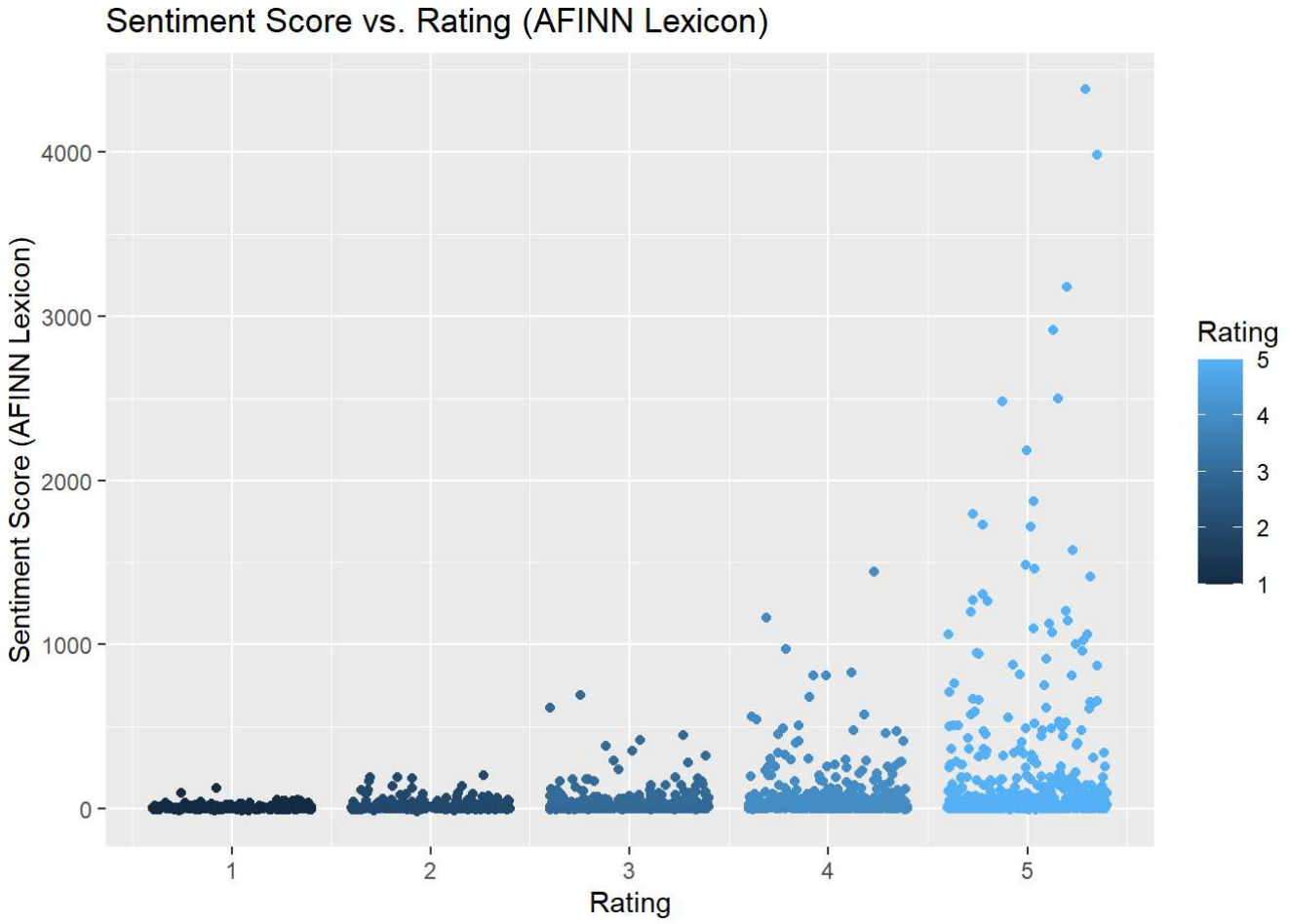
```
# Perform sentiment analysis using Bing lexicon
distinct_tidy_bing_sentiment <- tidy_reviews %>%
  inner_join(get_sentiments("bing")) %>%
  group_by(Clothing.ID, Rating) %>%
  summarize(sentiment_score = sum(as.numeric(sentiment == "positive") - as.numeric(sentiment
== "negative")))
```

```
## Joining with `by = join_by(word)`
```

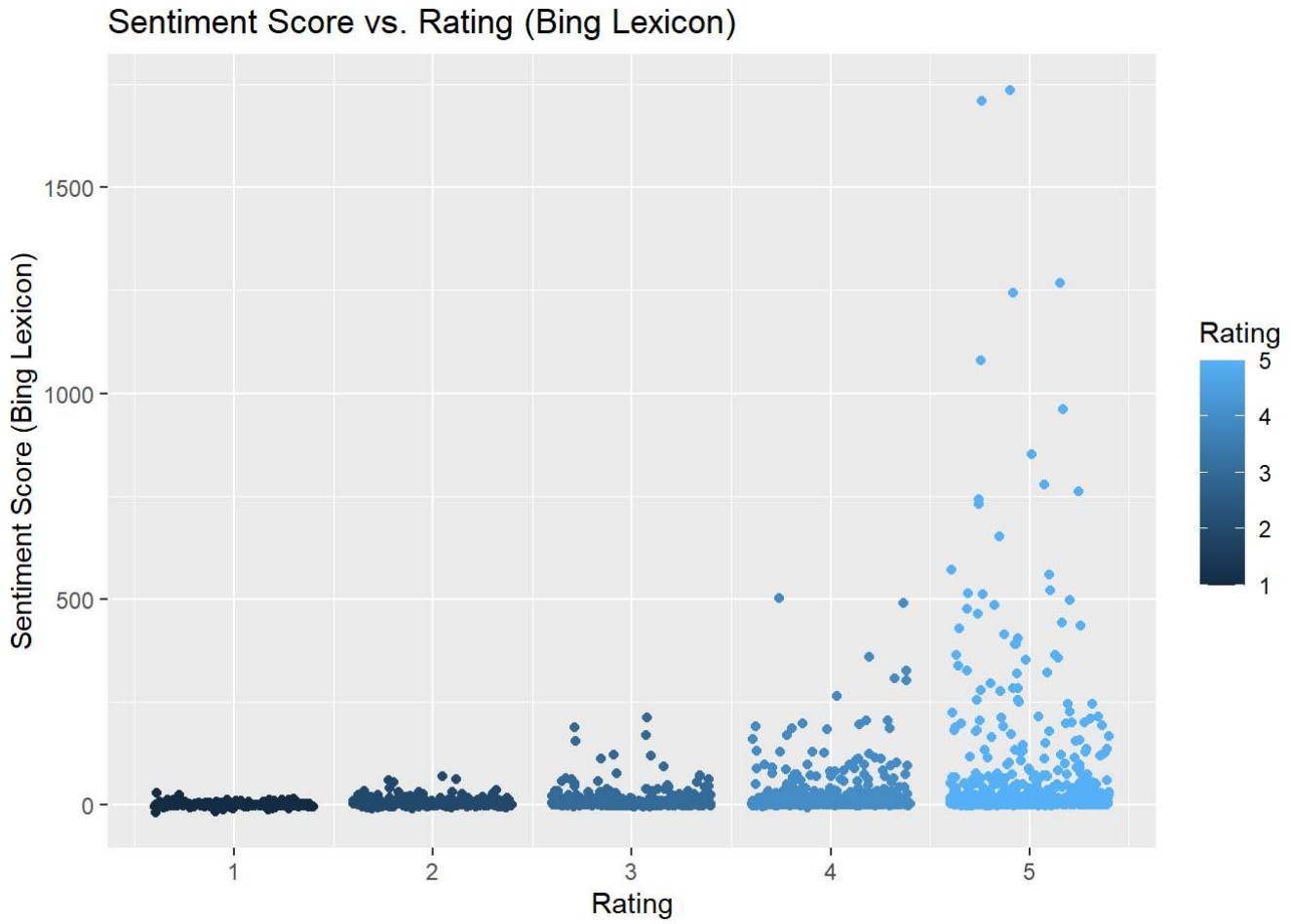
```
## Warning in inner_join(., get_sentiments("bing")): Each row in `x` is expected to match at
most 1 row in `y`.
## ℹ Row 65278 of `x` matches multiple rows.
## ℹ If multiple matches are expected, set `multiple = "all"` to silence this
##   warning.
```

```
## `summarise()` has grouped output by 'Clothing.ID'. You can override using the
## `.groups` argument.
```

```
reviews_sentiments <- tidy_reviews %>%
  mutate(sentiment = ifelse(Rating >= 4, "Positive", ifelse(Rating == 3, "Neutral", "Negativ
e")))

# Combine both sentiment analyses
combined_sentiment <- inner_join(distinct_tidy_reviews_sentiment, distinct_tidy_bing_sentimen
t , by = c("Clothing.ID", "Rating"), suffix = c("_afinn", "_bing"))

# Plot the sentiment scores against the rating
library(ggplot2)

ggplot(combined_sentiment, aes(x = Rating, y = sentiment_score_afinn, color = Rating)) +
  geom_jitter() +
  labs(x = "Rating", y = "Sentiment Score (AFINN Lexicon)", title = "Sentiment Score vs. Rati
ng (AFINN Lexicon)")
```



```
ggplot(combined_sentiment, aes(x = Rating, y = sentiment_score_bing, color = Rating)) +
  geom_jitter() +
  labs(x = "Rating", y = "Sentiment Score (Bing Lexicon)", title = "Sentiment Score vs. Ratin
g (Bing Lexicon)")
```



The code calculates the sentiment score for each review based on the lexicons and summarizes the results based on the age and rating of the reviews.

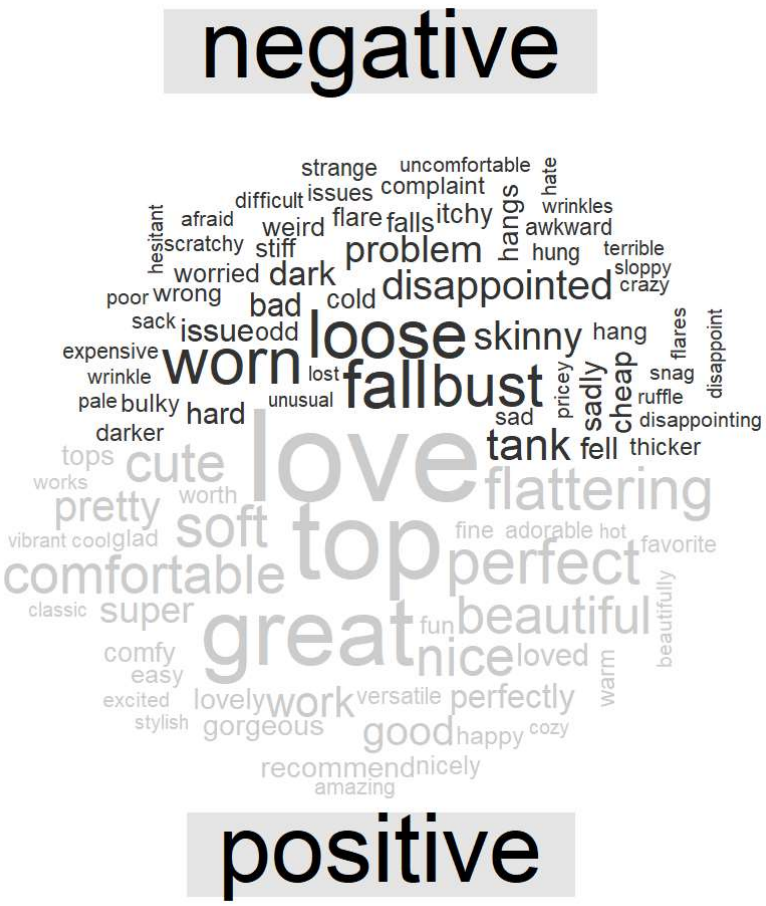
The code creates two plots, one for each lexicon, that show the relationship between the sentiment score and the rating of the reviews. The sentiment score is shown on the y-axis, while the rating is shown on the x-axis. The plots use the ggplot2 library to create a scatter plot of the sentiment scores for each rating, where each point represents a group of reviews with the same rating. The color of the points indicates the rating of the reviews.

The plots show that there is a positive relationship between the sentiment score and the rating of the reviews for both lexicons. This suggests that reviews with higher ratings tend to have more positive sentiment scores. However, the AFINN lexicon seems to have a wider range of sentiment scores compared to the Bing lexicon, with some reviews having very low or very high sentiment scores. This could be because the AFINN lexicon assigns numeric scores to words, which allows for more fine-grained analysis, while the Bing lexicon only classifies words as positive or negative. Overall, the code provides insights into the sentiment of the reviews and the relationship between the sentiment score and the rating of the reviews.

```
library(tm)
library(reshape2)


data(stop_words)
stop_words <- stop_words %>%
  anti_join(data.frame(word = c("but", "a", "and")), by = "word")

sentiment_scores <- tidy_reviews %>%
  select(word, Rating) %>%
  anti_join(stop_words, by = "word") %>%
  inner_join(get_sentiments("afinn"), by = "word") %>%
  mutate(sentiment_label = ifelse(value >= 0, "Positive", "Negative"))

tidy_reviews %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("gray20", "gray80"),
                   max.words = 100)
```

```
## Joining with `by = join_by(word)`
```

```
## Warning in inner_join(., get_sentiments("bing")): Each row in `x` is expected to match at
most 1 row in `y`.
## ℹ Row 65278 of `x` matches multiple rows.
## ℹ If multiple matches are expected, set `multiple = "all"` to silence this
##   warning.
```



This code performs sentiment analysis on the reviews dataset using the AFINN and Bing lexicons. It also creates a word cloud of the most frequently occurring positive and negative words in the reviews dataset.

First, the stop words are loaded and filtered to remove specific words using the anti_join() function. Then, the reviews dataset is cleaned and tokenized using the unnest_tokens() function. Stop words are removed using anti_join(). The AFINN lexicon is used to score each word in the dataset, and a sentiment label is assigned based on the score using the ifelse() function.

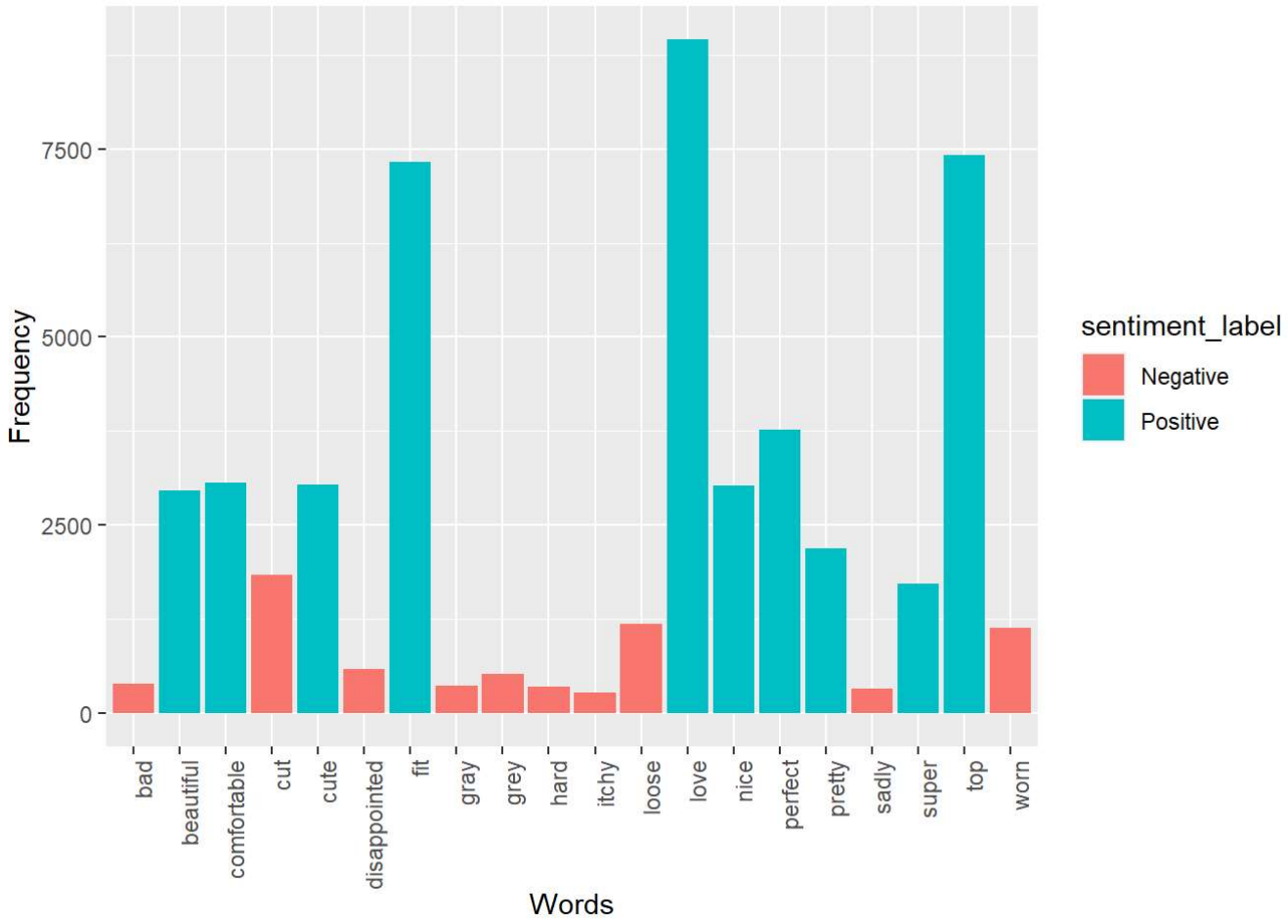
For the word cloud, the get_sentiments() function is used to load the Bing lexicon. The count() function is used to count the occurrences of each word and sentiment in the dataset. The acast() function is used to pivot the data into a format suitable for creating the word cloud. Finally, the comparison.cloud() function is used to create the word cloud.

The wordcloud can be used to get a quick overview of the most frequently occurring positive and negative words in the reviews dataset.

```r
library(conflicted)
library(tidytext)
library(ggplot2)

# Remove stopwords (including "but", "a", "ill", "and "and")
data(stop_words)
stop_words <- stop_words %>%
  anti_join(data.frame(word = c("but", "a", "and")), by = "word")

# Get the sentiment scores for each word in the reviews
sentiment_scores <- tidy_reviews %>%
  select(word, Rating) %>%
  anti_join(stop_words, by = "word") %>%
  inner_join(get_sentiments("afinn"), by = "word") %>%
  mutate(sentiment_label = ifelse(value >= 0, "Positive", "Negative"))

# Create a bar chart of most common positive and negative words
word_counts <- sentiment_scores %>%
  group_by(sentiment_label, word) %>%
  summarise(freq = n()) %>%
  arrange(sentiment_label, desc(freq)) %>%
  group_by(sentiment_label) %>%
  slice_head(n = 10)
```

```
## `summarise()` has grouped output by 'sentiment_label'. You can override using
## the `.groups` argument.
```

```r
ggplot(word_counts, aes(x = word, y = freq, fill = sentiment_label)) +
  geom_bar(stat = "identity", position = "dodge") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  xlab("Words") +
  ylab("Frequency")
```



Task C – Topic Modelling

```
library(topicmodels)
library(tm)
library(tidyr)

# Load the data from CSV file
data <- read.csv("./MS4S09_CW_Data.csv")

# Remove rows with missing values in the 'Review.Text' column
data <- data %>% drop_na(Review.Text)

# Convert 'Review.Text' column to character type and create a corpus
docs <- Corpus(VectorSource(as.character(data$Review.Text)))

# Perform pre-processing steps
docs <- tm_map(docs, content_transformer(gsub), pattern = "/", replacement = " ")
```

```
## Warning in tm_map.SimpleCorpus(docs, content_transformer(gsub), pattern = "/",
## : transformation drops documents
```

```
toSpace <- content_transformer(function(x, pattern) gsub(pattern, " ", x))
docs <- tm_map(docs, toSpace, "@")
```

```
## Warning in tm_map.SimpleCorpus(docs, toSpace, "@"): transformation drops
## documents
```

```
docs <- tm_map(docs, toSpace, "\\|")
```

```
## Warning in tm_map.SimpleCorpus(docs, toSpace, "\\|"): transformation drops
## documents
```

```
docs <- tm_map(docs, content_transformer(tolower))
```

```
## Warning in tm_map.SimpleCorpus(docs, content_transformer(tolower)):
## transformation drops documents
```

```
docs <- tm_map(docs, removeNumbers)
```

```
## Warning in tm_map.SimpleCorpus(docs, removeNumbers): transformation drops
## documents
```

```
docs <- tm_map(docs, removeWords, stopwords("english"))
```

```
## Warning in tm_map.SimpleCorpus(docs, removeWords, stopwords("english")):
## transformation drops documents
```

```
docs <- tm_map(docs, removePunctuation)
```

```
## Warning in tm_map.SimpleCorpus(docs, removePunctuation): transformation drops
## documents
```

```
docs <- tm_map(docs, stripWhitespace)
```

```
## Warning in tm_map.SimpleCorpus(docs, stripWhitespace): transformation drops
## documents
```

```
# Remove additional stopwords
docs <- tm_map(docs, removeWords, c("xxs","xs", "cami","hem","thats","isnt","tts","tee","iv
e","cant","doesnt","didnt","dont","lbs","theres","wont","wouldnt","youre","couldnt"))
```

```
## Warning in tm_map.SimpleCorpus(docs, removeWords, c("xxs", "xs", "cami", :
## transformation drops documents
```

```
# Create a DocumentTermMatrix object
dtm <- DocumentTermMatrix(docs)

# Inspect the resulting DocumentTermMatrix object
dtm
```

```
## <<DocumentTermMatrix (documents: 23486, terms: 15087)>>
## Non-/sparse entries: 576435/353756847
## Sparsity           : 100%
## Maximal term length: 32
## Weighting          : term frequency (tf)
```

```
library(topicmodels)
library(tm)
library(stringr)
library(ggplot2)
library(tidyr)
library(dplyr)
library(tidytext)

# Load the data from CSV file
data <- read.csv("./MS4S09_CW_Data.csv")

# Remove rows with missing values in the 'Review.Text' column
data <- data %>% drop_na(Review.Text)

# Convert 'Review.Text' column to character type
data$Review.Text <- as.character(data$Review.Text)

# Tokenize the text column
tidy_reviews <- data %>%
  unnest_tokens(word, Review.Text, drop = FALSE) %>%
  select(word, Clothing.ID, Age, Positive.Feedback.Count, Division.Name, Department.Name, Cla
ss.Name, Rating)

# Perform pre-processing steps
tidy_reviews <- tidy_reviews %>%
  mutate(word = gsub("[/@\\|]", " ", word),                                  # replace /, @ and |
with spaces
         word = tolower(word),                                               # convert to lowercas
e
         word = gsub("\\d+", "", word),                                       # remove digits
         word = removeWords(word, stopwords::stopwords(source = "smart")), # remove stop word
s
         word = gsub("[[:punct:]]", "", word),                                # remove punctuation
         word = trimws(word))                                                 # trim leading/trail
ing whitespace

# Remove words that are either stop words or unimportant for analysis
tidy_reviews <- tidy_reviews %>%
  anti_join(data.frame(word = stopwords::stopwords(source = "smart")), by = c("word" = "wor
d")) %>%
  anti_join(data.frame(word = c("xxs", "xs", "cami", "hem", "thats", "isnt", "tts", "tee", "i
ve", "cant", "doesnt", "didnt", "dont", "lbs", "theres", "wont", "wouldnt", "youre", "couldn
t")), by = "word")

# Find document-word counts
word_counts <- tidy_reviews %>%
  count(word, sort = TRUE)

# Top 20 terms
top_terms <- word_counts %>%
  top_n(20) %>%
  arrange(desc(n))
```

```
## Selecting by n
```

```r
library(tidytext)
library(dplyr)

docs_dtm <- tidy_reviews %>%
  count(Clothing.ID, word) %>%
  cast_dtm(Clothing.ID, word, n)
```

```r
library(topicmodels)
library(tm)
library(tidyr)

# Load the data from CSV file
data <- read.csv("./MS4S09_CW_Data.csv")

# Remove rows with missing values in the 'Review.Text' column
data <- data %>% drop_na(Review.Text)

# Convert 'Review.Text' column to character type and create a corpus
docs <- Corpus(VectorSource(as.character(data$Review.Text)))

# Perform pre-processing steps
docs <- tm_map(docs, content_transformer(gsub), pattern = "/", replacement = " ")
```

```
## Warning in tm_map.SimpleCorpus(docs, content_transformer(gsub), pattern = "/",
## : transformation drops documents
```

```r
toSpace <- content_transformer(function(x, pattern) gsub(pattern, " ", x))
docs <- tm_map(docs, toSpace, "@")
```

```
## Warning in tm_map.SimpleCorpus(docs, toSpace, "@"): transformation drops
## documents
```

```r
docs <- tm_map(docs, toSpace, "\\|")
```

```
## Warning in tm_map.SimpleCorpus(docs, toSpace, "\\|"): transformation drops
## documents
```

```r
docs <- tm_map(docs, content_transformer(tolower))
```

```
## Warning in tm_map.SimpleCorpus(docs, content_transformer(tolower)):
## transformation drops documents
```

```r
docs <- tm_map(docs, removeNumbers)
```

```
## Warning in tm_map.SimpleCorpus(docs, removeNumbers): transformation drops
## documents
```

```
docs <- tm_map(docs, removeWords, stopwords("english"))
```

```
## Warning in tm_map.SimpleCorpus(docs, removeWords, stopwords("english")):
## transformation drops documents
```

```
docs <- tm_map(docs, removePunctuation)
```

```
## Warning in tm_map.SimpleCorpus(docs, removePunctuation): transformation drops
## documents
```

```
docs <- tm_map(docs, stripWhitespace)
```

```
## Warning in tm_map.SimpleCorpus(docs, stripWhitespace): transformation drops
## documents
```

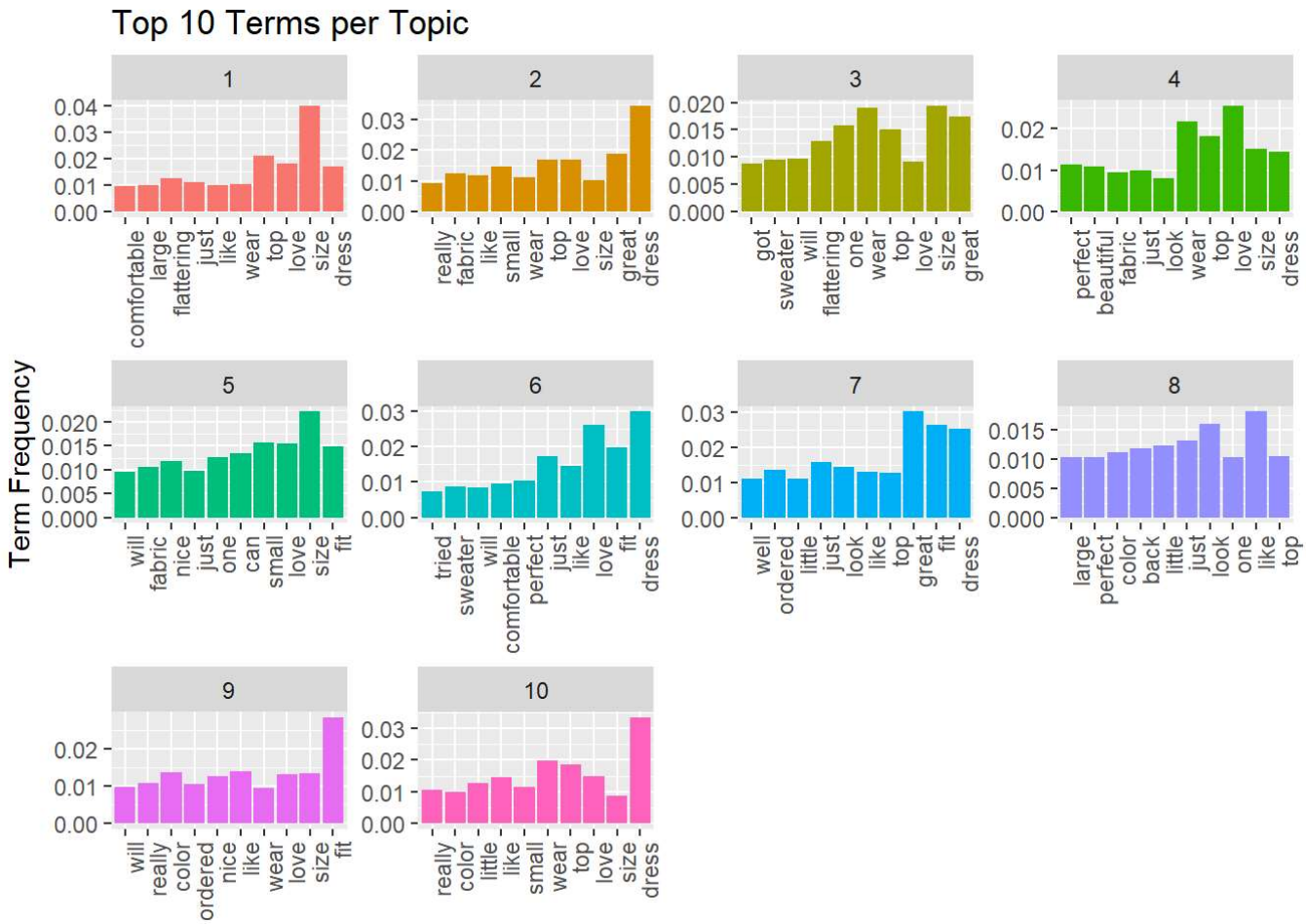
```
# Remove additional stopwords
docs <- tm_map(docs, removeWords, c("xxs","xs", "cami","hem","thats","isnt","tts","tee","iv
e","cant","doesnt","didnt","dont","lbs","theres","wont","wouldnt","youre","couldnt"))
```

```
## Warning in tm_map.SimpleCorpus(docs, removeWords, c("xxs", "xs", "cami", :
## transformation drops documents
```

```
# Create a DocumentTermMatrix object
dtm <- DocumentTermMatrix(docs)

# Inspect the resulting DocumentTermMatrix object
dtm
```

```
## <<DocumentTermMatrix (documents: 23486, terms: 15087)>>
## Non-/sparse entries: 576435/353756847
## Sparsity           : 100%
## Maximal term length: 32
## Weighting          : term frequency (tf)
```

```r
library(tidytext)
library(dplyr)
library(topicmodels)
library(ggplot2)
# Create a DocumentTermMatrix object
dtm <- DocumentTermMatrix(docs)

# Fit an LDA model with 10 topics
# create a DTM with minimum document frequency of 5
dtm <- DocumentTermMatrix(docs, control = list(minDocFreq = 5))

# remove all-zero rows
dtm <- dtm[!rowSums(as.matrix(dtm)) == 0, ]

# fit LDA model

lda_model <- LDA(dtm, k = 10)

# Get the top 10 terms for each topic
terms <- tidy(lda_model, matrix = "beta") %>%
  group_by(topic) %>%
  top_n(10, beta) %>%
  ungroup()

# Print the topics and their associated terms
terms %>%
  arrange(topic, -beta) %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~topic, scales = "free") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(x = NULL, y = "Term Frequency", title = "Top 10 Terms per Topic")
```



The code fits an LDA (Latent Dirichlet Allocation) model with 10 topics to a document-term matrix (DTM), using the LDA() function from the topicmodels package. The resulting model is then used to extract the top 10 terms for each topic, using the tidy() function from the tidytext package. These terms are then visualized using ggplot2, with each topic shown in a separate facet and the terms ordered by their frequency within that topic. The resulting plot can be used to gain insights into the topics and their associated terms.

Task D – Further exploration

```
library(topicmodels)
library(tm)
library(stringr)
library(ggplot2)
library(tidyr)
library(dplyr)
library(tidytext)
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

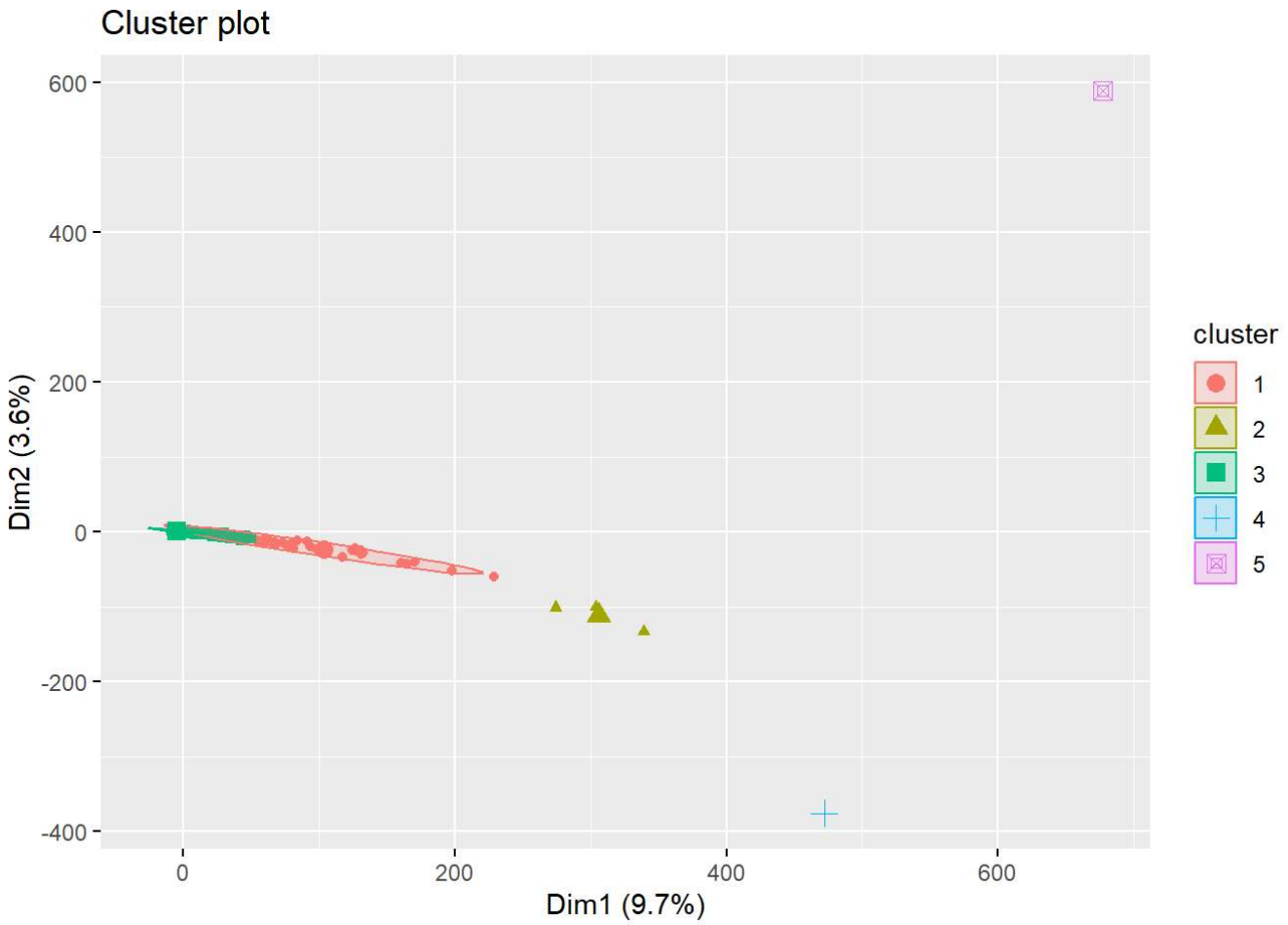
```r
# Load the data from CSV file
data <- read.csv("./MS4S09_CW_Data.csv")

# Remove rows with missing values in the 'Review.Text' column
data <- data %>% drop_na(Review.Text)

# Convert 'Review.Text' column to character type
data$Review.Text <- as.character(data$Review.Text)

# Tokenize the text column
tidy_reviews <- data %>%
  unnest_tokens(word, Review.Text, drop = FALSE) %>%
  select(word, Clothing.ID, Age, Positive.Feedback.Count, Division.Name, Department.Name, Cla
ss.Name, Rating)

# Perform pre-processing steps
tidy_reviews <- tidy_reviews %>%
  mutate(word = gsub("[/@\\|]", " ", word),                              # replace /, @ and |
with spaces
         word = tolower(word),                                           # convert to lowercas
e
         word = gsub("\\d+", "", word),                                   # remove digits
         word = removeWords(word, stopwords::stopwords(source = "smart")), # remove stop word
s
         word = gsub("[[:punct:]]", "", word),                            # remove punctuation
         word = trimws(word))                                             # trim leading/trail
ing whitespace

# Remove words that are either stop words or unimportant for analysis
tidy_reviews <- tidy_reviews %>%
  anti_join(data.frame(word = stopwords::stopwords(source = "smart")), by = c("word" = "wor
d")) %>%
  anti_join(data.frame(word = c("xxs", "xs", "cami", "hem", "thats", "isnt", "tts", "tee", "i
ve", "cant", "doesnt", "didnt", "dont", "lbs", "theres", "wont", "wouldnt", "youre", "couldn
t")), by = "word")

# Create a document-term matrix
doc_term_matrix <- tidy_reviews %>%
  count(Clothing.ID, word) %>%
  cast_dtm(Clothing.ID, word, n)

# Perform K-means clustering with 5 clusters
kmeans_model <- kmeans(scale(doc_term_matrix), centers = 5)

# Add cluster assignments to the data
tidy_reviews <- tidy_reviews %>%
  group_by(Clothing.ID) %>%
  reframe(cluster = as.factor(kmeans_model$cluster))

# Plot the clusters using the first two principal components
library(factoextra)
fviz_cluster(kmeans_model, data = scale(doc_term_matrix), stand = FALSE, geom = "point", fram
e.type = "norm")
```

```
## Warning: argument frame is deprecated; please use ellipse instead.
```

```
## Warning: argument frame.type is deprecated; please use ellipse.type instead.
```

```
## Too few points to calculate an ellipse
```

```
## Too few points to calculate an ellipse
## Too few points to calculate an ellipse
```



This code performs text clustering on customer reviews data using K-means clustering.A document-term matrix is created, and K-means clustering is performed with 5 clusters. The cluster assignments are then added to the data, and the clusters are plotted using the first two principal components.

Finally, the cluster summaries are printed, showing the number of reviews in each cluster and the top 10 words associated with each cluster. This information can help understand the common themes and topics of the customer reviews in each cluster.