MS4S09 COURSEWORK

MUHAMMED ADEBISI

2023-02-22

Installing packages

```
#install.packages("tidytext")
#install.packages("textdata")
#install.packages("tidyverse")
#install.packages("glue")
#install.packages("stringr")
#install.packages("ggthemes")
#install.packages(dplyr)
#install.packages(qdap)
#install.packages(tm)
#install.packages(wordcloud)
#install.packages(plotrix)
#install.packages(dendextend)
#install.packages(ggplot2)
#install.packages(ggthemes)
#install.packages(reshape2)
#install.packages(quanteda)
#install.packages(readxl)
#install.pacakages(SnowballC)
```

Loading the relevant libraries.

```
library(dplyr) #is a data manipulation framework that offers a consistent collection of verbs, assisting in the r
esolution of the most common data manipulation challenges.
```

```
## Warning: package 'dplyr' was built under R version 4.2.2
```

```
##
## 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(stringr) #The library supports certain common use-cases like str length() and str c() (concatenate). More
over, stringr has seven additional pattern matching routines that make it considerably simpler to perform string
searches and counts. Patterns can either be regular expressions or just strings.
```

```
## Warning: package 'stringr' was built under R version 4.2.2
```

```
library(tidytext) #To enable text conversion to and from tidy formats and easy switching between tidy tools and p
re-existing text mining programmes, we provide functions and associated data sets in this package.
```

```
## Warning: package 'tidytext' was built under R version 4.2.2
```

```
library(ggplot2) # It can produce several types of data visualisations, including bar charts, pie charts, histogr
ams, scatterplots, error charts, etc.
```

```
## Warning: package 'ggplot2' was built under R version 4.2.2
```

```
library(tidyr) #Its exclusive concentration is on data tidying or cleaning linked to formatting. These are the id
eas that are used by tidyr to define tidy data. Each column has the ability to modify, each row denotes a discove
ry, and each cell only holds one value.
```

```
## Warning: package 'tidyr' was built under R version 4.2.2
```

```
library(textdata) #The purpose of textdata is to make it simple to retrieve text-related data sets without packag
ing them.
```

```
## Warning: package 'textdata' was built under R version 4.2.2
```

```
library(tidyverse) #tidyverse is about the connections between the tools that make the workflow possible
```

```
## Warning: package 'tidyverse' was built under R version 4.2.2
```

```
## Warning: package 'tibble' was built under R version 4.2.2
```

```
## Warning: package 'readr' was built under R version 4.2.2
```

```
## Warning: package 'purrr' was built under R version 4.2.2
```

```
## Warning: package 'forcats' was built under R version 4.2.2
```

```
## Warning: package 'lubridate' was built under R version 4.2.2
```

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

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the ]8;;http://conflicted.r-lib.org/conflicted package]8;; to force all conflicts to become errors
```

```
library(glue) #provides interpreted string literals that are lightweight, quick, and independent.
```

```
## Warning: package 'glue' was built under R version 4.2.2
```

```
library(tm) #a framework for text mining-based R applications.
```

```
## Warning: package 'tm' was built under R version 4.2.2
```

```
## Loading required package: NLP
##
## Attaching package: 'NLP'
##
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
library(wordcloud) #to minimise over-plotting in scatter plots with text, display differences and similarities ac
ross documents, and build lovely word clouds
```

```
## Warning: package 'wordcloud' was built under R version 4.2.2
```

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

```
library(plotrix) #package includes resources for data graphing in R.
library(dendextend) #provides a collection of functions for R dendrogram objects that allow you to compare and vi
sualise trees of hierarchical clusterings.
```

```
## Warning: package 'dendextend' was built under R version 4.2.2
```

```
##
## ---------------------
## Welcome to dendextend version 1.16.0
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues
## You may ask questions at stackoverflow, use the r and dendextend tags:
##   https://stackoverflow.com/questions/tagged/dendextend
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ---------------------
##
##
## Attaching package: 'dendextend'
##
## The following object is masked from 'package:stats':
##
##     cutree
```

```
library(ggthemes) #gives the ggplot2 package more themes, geometries, and scales.
```

```
## Warning: package 'ggthemes' was built under R version 4.2.2
```

```
library(reshape2) #flexible data reshaping
```

```
## Warning: package 'reshape2' was built under R version 4.2.2
```

```
##
## Attaching package: 'reshape2'
##
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
library(quanteda) # It was built to be used by individuals with textual data—perhaps from books, Tweets, or trans
cripts—to both manage that data (sort, label, condense, etc.) and analyze its contents.
```

```
## Warning: package 'quanteda' was built under R version 4.2.2
```

```
## Package version: 3.2.4
## Unicode version: 13.0
## ICU version: 69.1
## Parallel computing: 12 of 12 threads used.
## See https://quanteda.io for tutorials and examples.
##
## Attaching package: 'quanteda'
##
## The following object is masked from 'package:tm':
##
##     stopwords
##
## The following objects are masked from 'package:NLP':
##
##     meta, meta<-
```

```
library(readxl) #to read the excel file
```

```
## Warning: package 'readxl' was built under R version 4.2.2
```

```
library(SnowballC) #for stemming documents
library(qdap) # to assist with quantitative analysis. The program combines qualitative discussion transcripts wit
h statistical analysis and visualization
```

```
## Warning: package 'qdap' was built under R version 4.2.2
```

```
## Loading required package: qdapDictionaries
## Loading required package: qdapRegex
```

```
## Warning: package 'qdapRegex' was built under R version 4.2.2
```

```
##
## Attaching package: 'qdapRegex'
##
## The following object is masked from 'package:ggplot2':
##
##     %+%
##
## The following object is masked from 'package:dplyr':
##
##     explain
##
## Loading required package: qdapTools
```

```
## Warning: package 'qdapTools' was built under R version 4.2.2
```

```
##
## Attaching package: 'qdapTools'
##
## The following object is masked from 'package:dplyr':
##
##     id
##
##
## Attaching package: 'qdap'
##
## The following objects are masked from 'package:tm':
##
##     as.DocumentTermMatrix, as.TermDocumentMatrix
##
## The following object is masked from 'package:NLP':
##
##     ngrams
##
## The following objects are masked from 'package:base':
##
##     Filter, proportions
```

Introducing the Dataset

```
#Exporting the data from excel
Dataset<-read_excel("C:/Users/30061694/Downloads/MS4S09CWData.xlsx")
```

```
## New names:
## • `` -> `...1`
```

```
options(stringsAsFactors = FALSE) #Whether strings in a data frame should be considered as factor variables or as simple strings is determined by a logical argument. We commonly set it to FALSE for text mining in order to consider the characters as strings and properly utilise all text mining approaches.
Dataset
```

```
## # A tibble: 23,486 × 11
##     ...1 Clothing_ID   Age Title  Revie…¹ Rating Recom…² Posit…³ Divis…⁴ Depar…⁵
##    <dbl>       <dbl> <dbl> <chr>  <chr>    <dbl>   <dbl>   <dbl> <chr>   <chr>
##  1     0         767    33 <NA>   "Absol…      4       1       0 Initma… Intima…
##  2     1        1080    34 <NA>   "Love …      5       1       4 General Dresses
##  3     2        1077    60 Some … "I had…      3       0       0 General Dresses
##  4     3        1049    50 My fa… "I lov…      5       1       0 Genera… Bottoms
##  5     4         847    47 Flatt… "This …      5       1       6 General Tops
##  6     5        1080    49 Not f… "I lov…      2       0       4 General Dresses
##  7     6         858    39 Cagrc… "I ade…      5       1       1 Genera… Tops
##  8     7         858    39 Shimm… "I ord…      4       1       4 Genera… Tops
##  9     8        1077    24 Flatt… "I lov…      5       1       0 General Dresses
## 10     9        1077    34 Such … "I'm 5…      5       1       0 General Dresses
## # … with 23,476 more rows, 1 more variable: Class_Name <chr>, and abbreviated
## #   variable names ¹Review_Text, ²Recommended_IND, ³`Positive Feedback Count`,
## #   ⁴`Division Name`, ⁵Department_Name
```

TASK A (TEXT MINING) Text mining is a technique that requires a considerable deal of knowledge because it includes a human engaging with a collection of documents over time while using various analysis tools. Similar to data mining, text mining aims to extract usable information from data sources by spotting and analysing intriguing patterns. In contrast, surprising patterns are found in the unstructured textual data in the documents in these collections rather than in the formalised database entries in text mining when the data sources are document collections.

```
names(Dataset)
```

```
##  [1] "...1"                    "Clothing_ID"
##  [3] "Age"                     "Title"
##  [5] "Review_Text"             "Rating"
##  [7] "Recommended_IND"         "Positive Feedback Count"
##  [9] "Division Name"           "Department_Name"
## [11] "Class_Name"
```

Listing the Columns in the Dataset

```
head(Dataset)
```

```
## # A tibble: 6 × 11
##    ...1 Clothing_ID   Age Title   Revie…¹ Rating Recom…² Posit…³ Divis…⁴ Depar…⁵
##   <dbl>       <dbl> <dbl> <chr>   <chr>    <dbl>   <dbl>   <dbl> <chr>   <chr>
## 1     0         767    33 <NA>    "Absol…      4       1       0 Initma… Intima…
## 2     1        1080    34 <NA>    "Love …      5       1       4 General Dresses
## 3     2        1077    60 Some m… "I had…      3       0       0 General Dresses
## 4     3        1049    50 My fav… "I lov…      5       1       0 Genera… Bottoms
## 5     4         847    47 Flatte… "This …      5       1       6 General Tops
## 6     5        1080    49 Not fo… "I lov…      2       0       4 General Dresses
## # … with 1 more variable: Class_Name <chr>, and abbreviated variable names
## #   ¹Review_Text, ²Recommended_IND, ³`Positive Feedback Count`,
## #   ⁴`Division Name`, ⁵Department_Name
```

listing the top rows in the dataset

```
# Text Transformation/Extraction
#The customer reviews for various goods are listed in the column Review_Text. Our analysis is focused on this. No
w, we'll look at how to represent text in a data frame.

#First, the Review_Text is transformed into a corpus, which is a group of related text documents. To achieve this
, we utilise the R package "tm".

#We must supply a "Source" object as an argument to the VCorpus function in order to construct a corpus using tm.

#The source we employ in this case is a "Vectorsource" that exclusively accepts character vector inputs.
#We have now created a corpus from the Review.text column that we refer to as "Customer_Dataset."

Customer_Dataset <- Corpus(VectorSource(Dataset$Review_Text))
Customer_Dataset
```

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 23486
```

```
toSpace <- content_transformer (function(x , pattern) gsub(pattern, " ", x))
Customer_Dataset <- tm_map(Customer_Dataset, toSpace, "/")
```

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

```
Customer_Dataset <- tm_map(Customer_Dataset, toSpace, "@")
```

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

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

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

```
#Change to lower case so that, for example, the phrases "Boot" and "boot" will be combined to form the word "boot
"
Customer_Dataset<- tm_map(Customer_Dataset, tolower)
```

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

```
#Remove Punctuation
Customer_Dataset <- tm_map(Customer_Dataset, removePunctuation)
```

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

```
#strip whitespace
# this Remove any excess whitespace from a text file. There is a collapse of many whitespace characters into a si
ngle blank.
Customer_Dataset <- tm_map(Customer_Dataset, stripWhitespace)
```

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

```
#Remove stopwords
#Remove stopwords: When doing text mining, understanding the idea of "stopwords" is crucial. As we write, there a
re typically several prepositions, pronouns, conjunctions, etc. in the text. Before we analyse the text, these wo
rds must be deleted. Otherwise, stopwords will show up in every list of commonly used words and will distort the
meaning of the text's main terms.
Customer_Dataset <- tm_map(Customer_Dataset, removeWords, stopwords("english"))
```

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

Stemming

```
## Stemming document
#stemming refers to the process of reducing inflected (or derived) words to their word stem, base, or root form t
ypically a written word form.For instance, using a stemming technique, the terms "replacement," "replaced," and "
replacing" are all reduced to the word "replace" as their root.
Customer_Dataset <- tm_map(Customer_Dataset, stemDocument)
```

```
## Warning in tm_map.SimpleCorpus(Customer_Dataset, stemDocument): transformation
## drops documents
```

```
##Viewing the corpus content
Customer_Dataset[[1]][1]
```

```
## $content
## [1] "absolut wonder silki sexi comfort"
```

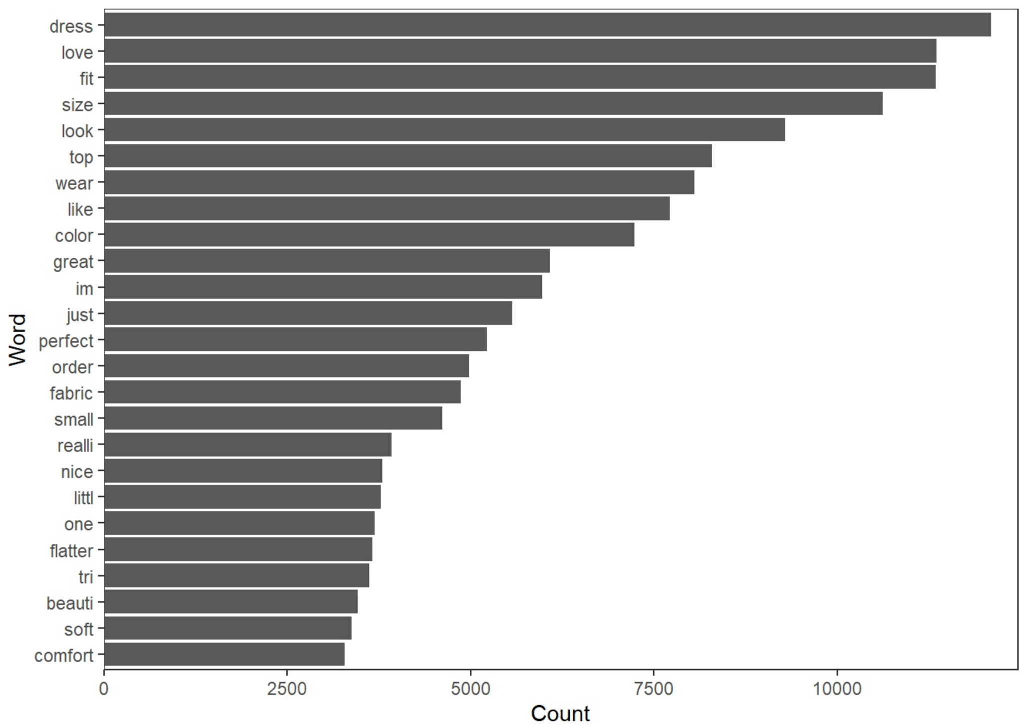
It is clear from the results that the client was satisfied with the product.

```
# Find the 25 most common terms:common_term
common_term <- freq_terms(Customer_Dataset, 25)
common_term
```

```
##    WORD     FREQ
## 1  dress   12102
## 2  love    11354
## 3  fit     11341
## 4  size    10619
## 5  look     9287
## 6  top      8295
## 7  wear     8051
## 8  like     7718
## 9  color    7234
## 10 great    6085
## 11 im       5975
## 12 just     5572
## 13 perfect  5226
## 14 order    4984
## 15 fabric   4863
## 16 small    4617
## 17 realli   3921
## 18 nice     3802
## 19 littl    3773
## 20 one      3694
## 21 flatter  3661
## 22 tri      3623
## 23 beauti   3464
## 24 soft     3382
## 25 comfort  3281
```

This are the most top 25 frequently used words

```
# Plot 25 most frequent terms
plot(common_term)
```



Creating the DTM & TDM from the corpus The cleaned-up and preprocessed corpus is then transformed into a matrix known as the document term matrix.

The document-term matrix is a mathematical matrix that may be used to calculate the frequency of words used in a group of documents. In a document-term matrix, the columns represent the collection's terms, while the rows represent its documents.

The document-term matrix has been superseded by the term-document matrix. In language analysis, it is commonly employed. By converting the DTM/TDM into a simple matrix using as.matrix, it is easy to begin analysing the data ().

```
examine_dtm <- DocumentTermMatrix(Customer_Dataset)
examine_tdm <- TermDocumentMatrix(Customer_Dataset)
```

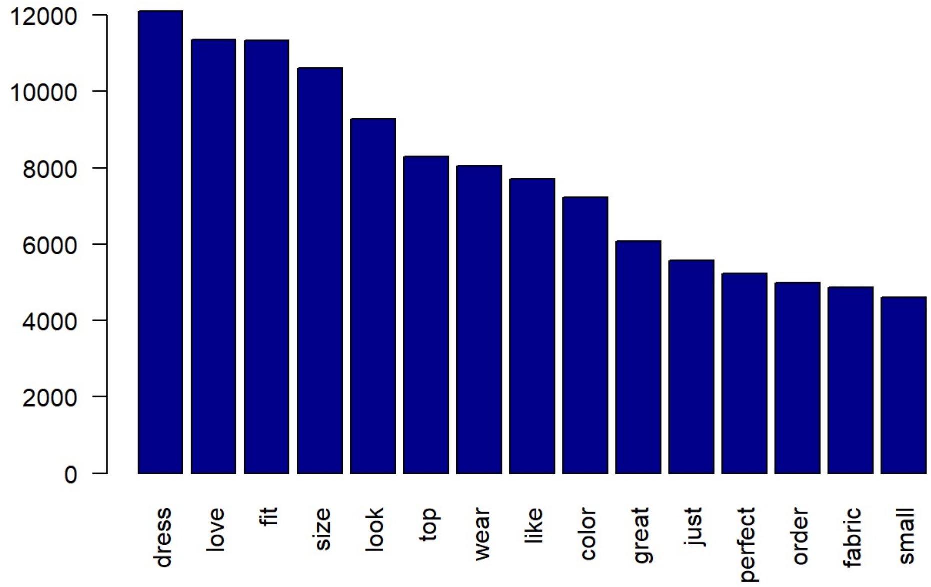
Searching for popular phrases with the TDM

```
# Convert TDM to matrix
examine_m <- as.matrix(examine_tdm)
# Sum rows and frequency data frame
examine_term_freq <- rowSums(examine_m)
# Sort term_frequency in descending order
examine_term_freq <- sort(examine_term_freq, decreasing = T)
# View the top 20 most common words
examine_term_freq[1:20]
```

```
##   dress    love     fit    size    look     top    wear    like   color   great
##   12102   11354   11335   10615    9287    8295    8051    7718    7234    6085
##    just perfect   order  fabric   small  realli    nice   littl     one flatter
##    5572    5225    4984    4863    4616    3921    3802    3773    3694    3661
```

Plotting a Bar chart of 30 most common words

```
# Plot a barchart of the 30 most common words
barplot(examine_term_freq[1:15], col = "darkblue", las = 2)
```



Word cloud A word cloud is a well-liked technique for determining the phrases that appear most frequently in a text corpus. The size of the terms in the word cloud varies depending on how frequently they are used.Want to check for the top 60

```
check_word_freq <- data.frame(term = names(examine_term_freq),
  num = examine_term_freq)
# Create a wordcloud for the values in word_freqs
wordcloud(check_word_freq$term, check_word_freq$num,
  max.words = 50, colors = brewer.pal(4,"Dark2"))
```

TASK B (SENTIMENT ANALYSIS) Using machine learning and natural language processing (nlp), sentiment analysis, commonly referred to as opinion mining, is a text mining technique that automatically examines texts for the author's sentiment (positive, negative, neutral, and beyond). Text mining's major objective is to extract useful data and insights from texts so that businesses may make informed decisions. With the use of sophisticated machine learning algorithms, it is simple to determine whether a comment is good, negative, or neutral. Much more specific results can be obtained by using aspect-based sentiment analysis. Aspect-based sentiment analysis analyses material, such as customer reviews or product testimonials, first by category to determine whether categories are positive or negative (Features, Shipping, Customer Service, etc.).

Tokenization The process of tokenization in natural language processing involves breaking down the given text into tokens, which are the smallest grammatical units of a sentence. Punctuation, words, and numbers can all be viewed as tokens. Tokenization is required because we could wish to count the number of times each word appears in the provided text by dividing it up into tokens.

```
#We then need to split the text into tokens.
tidy_data <- Dataset %>%
  unnest_tokens(word, Review_Text)
tidy_data
```

```
## # A tibble: 1,369,578 × 11
##     ...1 Clothing_ID   Age Title Rating Recomm…¹ Posit…² Divis…³ Depar…⁴ Class…⁵
##    <dbl>       <dbl> <dbl> <chr>  <dbl>    <dbl>   <dbl> <chr>   <chr>   <chr>
##  1     0         767    33 <NA>       4        1       0 Initma… Intima… Intima…
##  2     0         767    33 <NA>       4        1       0 Initma… Intima… Intima…
##  3     0         767    33 <NA>       4        1       0 Initma… Intima… Intima…
##  4     0         767    33 <NA>       4        1       0 Initma… Intima… Intima…
##  5     0         767    33 <NA>       4        1       0 Initma… Intima… Intima…
##  6     0         767    33 <NA>       4        1       0 Initma… Intima… Intima…
##  7     0         767    33 <NA>       4        1       0 Initma… Intima… Intima…
##  8     1        1080    34 <NA>       5        1       4 General Dresses Dresses
##  9     1        1080    34 <NA>       5        1       4 General Dresses Dresses
## 10     1        1080    34 <NA>       5        1       4 General Dresses Dresses
## # … with 1,369,568 more rows, 1 more variable: word <chr>, and abbreviated
## #   variable names ¹Recommended_IND, ²`Positive Feedback Count`,
## #   ³`Division Name`, ⁴Department_Name, ⁵Class_Name
```

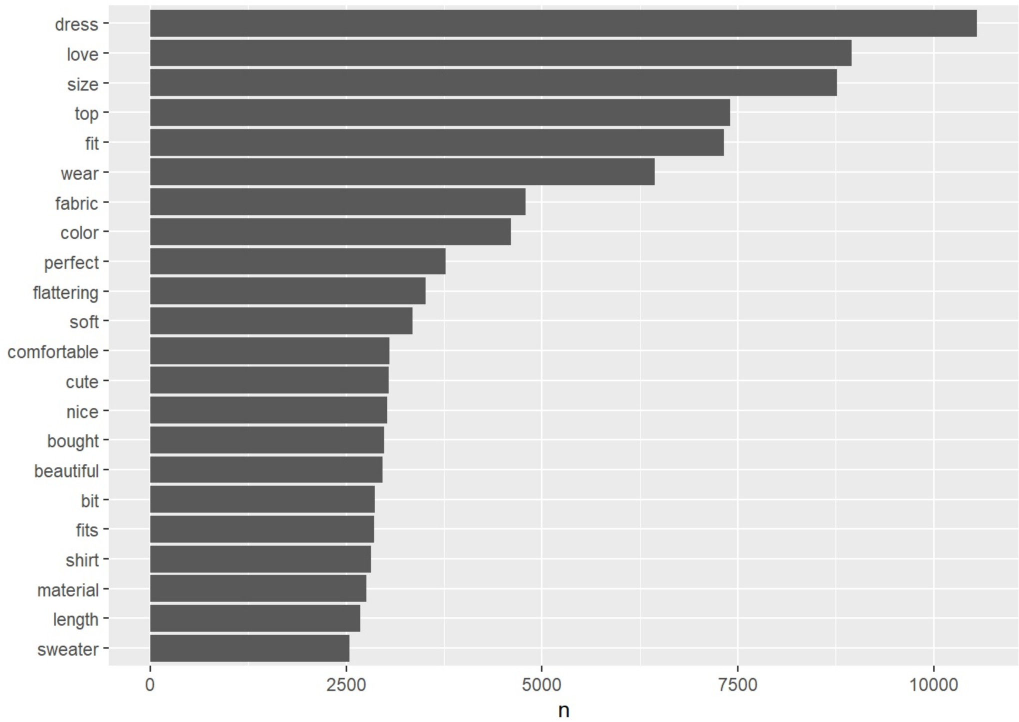
```
#Once again we need to remove stop words from the data.
data(stop_words)
tidy_data <- tidy_data %>%
  anti_join(stop_words, by = "word")
```

```
#Use the count function to identify the most common words.
tidy_data %>%
  count(word, sort = TRUE)
```

```
## # A tibble: 14,144 × 2
##    word           n
##    <chr>      <int>
##  1 dress      10553
##  2 love        8948
##  3 size        8768
##  4 top         7405
##  5 fit         7318
##  6 wear        6439
##  7 fabric      4790
##  8 color       4605
##  9 perfect     3772
## 10 flattering  3517
## # … with 14,134 more rows
```

.

```
#Produce a plot of the all words which appear more then 2500 times
tidy_data %>%
  count(word, sort = TRUE) %>%
  filter(n > 2500) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n)) +
  geom_col() +
  xlab(NULL) +
  coord_flip()
```



Sentiment analysis with the tidytext package using the “bing” lexicon

```
text_df <- tibble(text = str_to_lower(Dataset$Review_Text))
text_df
```

```
## # A tibble: 23,486 × 1
##    text
##    <chr>
##  1 "absolutely wonderful - silky and sexy and comfortable"
##  2 "love this dress!  it's sooo pretty.  i happened to find it in a store, and …
##  3 "i had such high hopes for this dress and really wanted it to work for me. i…
##  4 "i love, love, love this jumpsuit. it's fun, flirty, and fabulous! every tim…
##  5 "this shirt is very flattering to all due to the adjustable front tie. it is…
##  6 "i love tracy reese dresses, but this one is not for the very petite. i am j…
##  7 "i aded this in my basket at hte last mintue to see what it would look like …
##  8 "i ordered this in carbon for store pick up, and had a ton of stuff (as alwa…
##  9 "i love this dress. i usually get an xs but it runs a little snug in bust so…
## 10 "i'm 5\"5' and 125 lbs. i ordered the s petite to make sure the length wasn'…
## # … with 23,476 more rows
```

```
#Review the sentiment lexicons in the tidyverse package.
sentiments
```

```
## # A tibble: 6,786 × 2
##    word        sentiment
##    <chr>       <chr>
##  1 2-faces     negative
##  2 abnormal    negative
##  3 abolish     negative
##  4 abominable  negative
##  5 abominably  negative
##  6 abominate   negative
##  7 abomination negative
##  8 abort       negative
##  9 aborted     negative
## 10 aborts      negative
## # … with 6,776 more rows
```

```
#sentiment analysis with the tidytext package using the "bing" lexicon
bing_word_counts <- text_df %>% unnest_tokens(output = word, input = text) %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE)
```

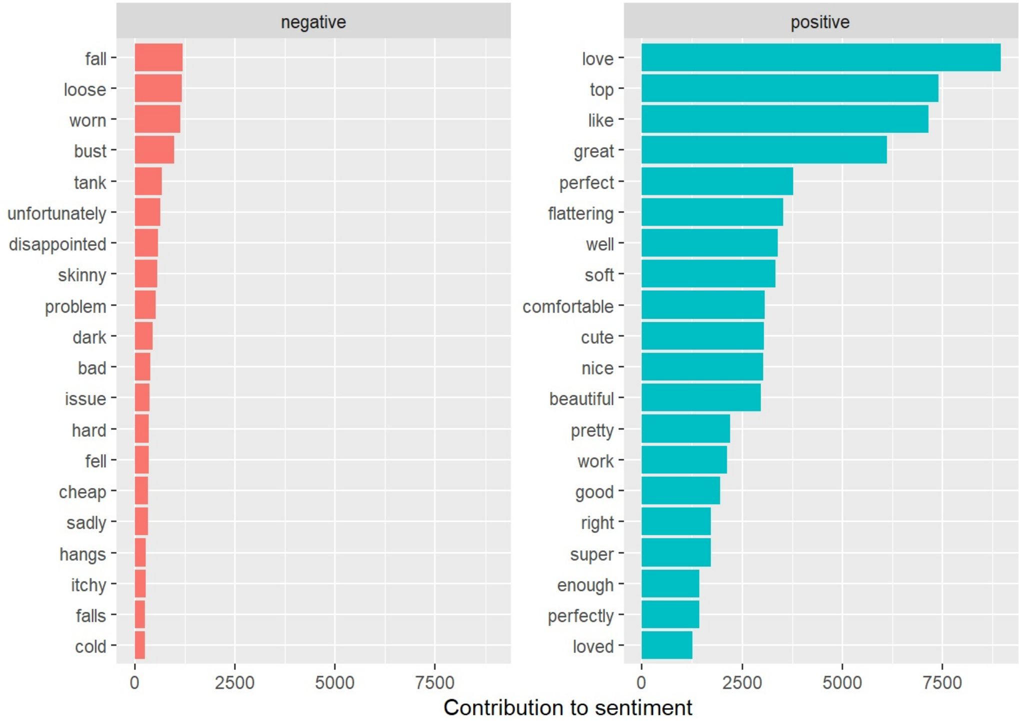
```
## Joining with `by = join_by(word)`
```

```
## Warning in inner_join(., get_sentiments("bing")): Each row in `x` is expected to match at most 1 row in `y`.
## ℹ Row 65660 of `x` matches multiple rows.
## ℹ If multiple matches are expected, set `multiple = "all"` to silence this
##   warning.
```

```
#select top 20 words by sentiment using bing(Positve and Negative)
bing_top_10_word_sentiment <- bing_word_counts %>%
  group_by(sentiment) %>%
  slice_max(order_by = n, n = 20) %>%
  ungroup()%>%
  mutate(word = reorder(word, n))
bing_top_10_word_sentiment
```

```
## # A tibble: 40 × 3
##    word          sentiment     n
##    <fct>         <chr>     <int>
##  1 fall          negative   1200
##  2 loose         negative   1181
##  3 worn          negative   1137
##  4 bust          negative    992
##  5 tank          negative    679
##  6 unfortunately negative    633
##  7 disappointed  negative    584
##  8 skinny        negative    565
##  9 problem       negative    522
## 10 dark          negative    447
## # … with 30 more rows
```

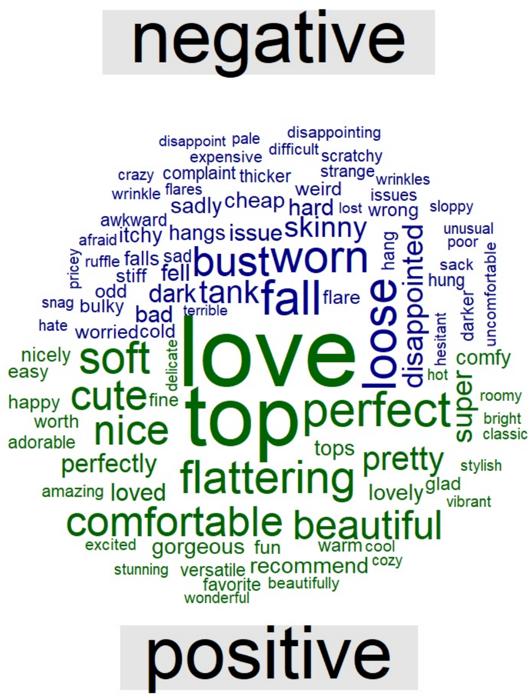
```
#create a barplot showing contribution of words to sentiment
bing_top_10_word_sentiment %>%
  ggplot(aes(word, n, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  labs(y = "Contribution to sentiment", x =  NULL) +
  coord_flip()
```




```
#comparison of the negative and positive sentiment using wordcloud
library(wordcloud)
library(reshape2)

 bing_word_counts <- tidy_data %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("darkblue", "darkgreen"),
                   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 22650 of `x` matches multiple rows.
## i If multiple matches are expected, set `multiple = "all"` to silence this
##   warning.
```



The comparison cloud makes a clear distinction between terms used by individuals who are pleased with the product (Positive) and those who are not (Negative). Those who didn’t like it used words like “disappointing,” “flare,” “cheap,” and other unfavourable adjectives, while those that are please with it used words like “love”, “perfect”, “cute” and other favourable adjectives.

Using SYUZHET package for generating sentiment score Sentiments can also be represented numerically in order to better communicate the degree of positive or negative strength included in a body of text.

This example generates sentiment scores using the Syuzhet package, which includes four sentiment dictionaries and a mechanism for accessing the sentiment extraction tool built by Stanford's NLP lab.

The function get sentiment takes two parameters: a character vector (containing sentences or words) and a method. Which of the four available sentiment extraction methods will be utilised is determined by the approach chosen. The four available techniques are syuzhet, bing, afinn, and nrc.

However,we will be making use of the syuzhet package.

installing packages

```
#install.packages("syuzhet")
#install.packages("lubridate")
#install.packages("scales")
```

loading the relevant libraries

```
library(syuzhet)
```

```
## Warning: package 'syuzhet' was built under R version 4.2.2
```

```
##
## Attaching package: 'syuzhet'
```

```
## The following object is masked from 'package:plotrix':
##
##     rescale
```

```
library(lubridate)
library(scales)
```

```
## Warning: package 'scales' was built under R version 4.2.2
```

```
##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:syuzhet':
##
##     rescale
```

```
## The following object is masked from 'package:plotrix':
##
##     rescale
```

```
## The following object is masked from 'package:purrr':
##
##     discard
```

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
#Taking the review_text column
review <- iconv(Dataset$Review_Text)
```

```
#Obtaining the sentiment score named sent
Syuzhet_vector <- get_sentiment(review, method="syuzhet")
```

```
#viewing the first row of the vector
head(Syuzhet_vector)
```

```
## [1] 2.00 3.35 2.95 2.50 3.20 1.35
```

The Syuzhet vector's first element has a value of 2.00, according to a visual inspection. This indicates that the first response's (line) sentiment scores for all significant words in the text file add up to 2.00. The syuzhet method uses a decimal scale for sentiment scores that ranges from -1 (indicating the most negative) to +1 (indicating the most positive).

```
#checking for summary
summary(Syuzhet_vector)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -4.050   1.250   2.300   2.416   3.500   9.550
```

However, the median value of the suyzhet vector's summary statistics is 2.300, which is greater than zero and indicates that the overall average response sentiment is positive.

TASK C (TOPIC MODELLING) The use of topic models makes it simple to analyse large amounts of unlabeled text. A theme is a collection of words that are regularly used together. Using contextual clues, topic models can connect words with similar meanings and distinguish between distinct uses of words with different meanings.

```
#install.package("topicmodels")

library(topicmodels)
```

```
## Warning: package 'topicmodels' was built under R version 4.2.2
```

```
#From the DTM in the Text Mining
examine_dtm <- DocumentTermMatrix(Customer_Dataset)
examine_dtm
```

```
## <<DocumentTermMatrix (documents: 23486, terms: 13228)>>
## Non-/sparse entries: 581211/310091597
## Sparsity           : 100%
## Maximal term length: 32
## Weighting          : term frequency (tf)
```

```
#filling zeros rows with number
all_zero_rows <- which(apply(examine_dtm, 1, function(x) all (x==0)))
z_dtm <- examine_dtm[-all_zero_rows,]
z_dtm
```

```
## <<DocumentTermMatrix (documents: 22641, terms: 13228)>>
## Non-/sparse entries: 581211/298913937
## Sparsity           : 100%
## Maximal term length: 32
## Weighting          : term frequency (tf)
```

```
#we turn the DTM data into data frame
#and name it :Mod_td
Mod_td <- tidy(z_dtm)
Mod_td
```

```
## # A tibble: 581,211 × 3
##    document term    count
##    <chr>    <chr>   <dbl>
##  1 1        absolut     1
##  2 1        comfort     1
##  3 1        sexi        1
##  4 1        silki       1
##  5 1        wonder      1
##  6 2        bought      1
##  7 2        definit     1
##  8 2        dress       1
##  9 2        find        1
## 10 2        glad        1
## # … with 581,201 more rows
```

```
#Notice that only the non-zero values are included in the tidied output
```

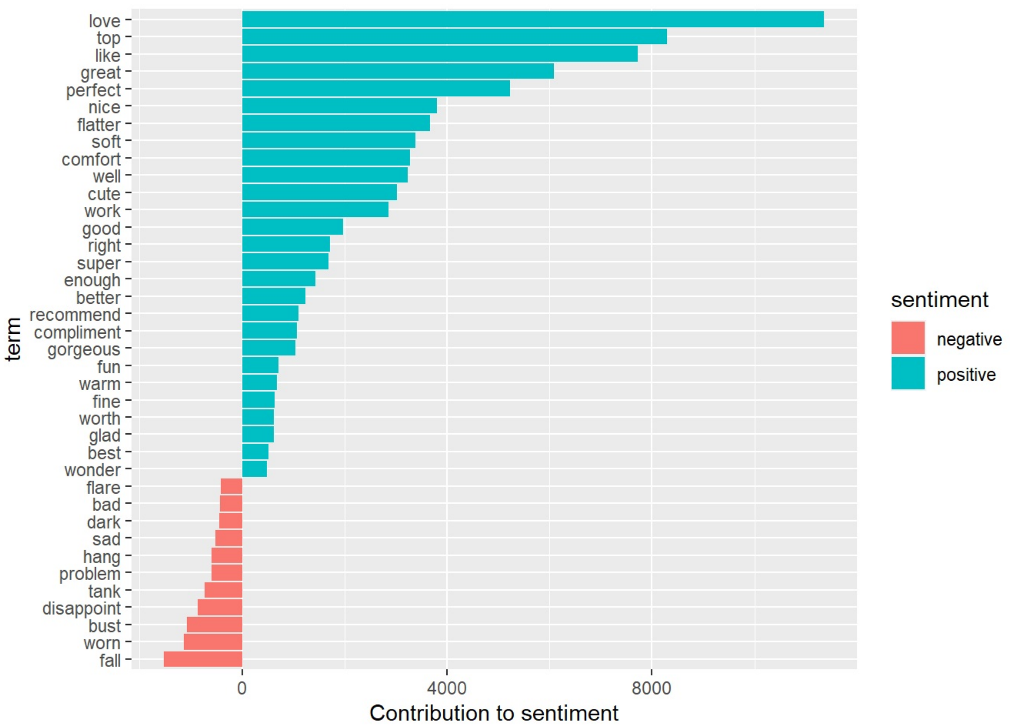
```
#We now have the sentiment analysis as follows
Mod_sentiments <- Mod_td %>%
  inner_join(get_sentiments("bing"), by = c(term = "word"))
```

```
## Warning in inner_join(., get_sentiments("bing"), by = c(term = "word")): Each row in `x` is expected to match
at most 1 row in `y`.
## i Row 27859 of `x` matches multiple rows.
## i If multiple matches are expected, set `multiple = "all"` to silence this
##   warning.
```

```
Mod_sentiments
```

```
## # A tibble: 93,609 × 4
##    document term     count sentiment
##    <chr>    <chr>    <dbl> <chr>
##  1 1        comfort      1 positive
##  2 1        wonder       1 positive
##  3 2        glad         1 positive
##  4 2        love         2 positive
##  5 3        cheap        1 negative
##  6 3        comfort      1 positive
##  7 3        flaw         1 negative
##  8 3        nice         1 positive
##  9 3        sever        1 negative
## 10 3        top          1 positive
## # … with 93,599 more rows
```

```
#Contribution sentiment for words that appears more than 400 times
#For negative and positive sentiment
library(ggplot2)
Mod_sentiments %>%
  count(sentiment, term, wt = count) %>%
  ungroup() %>%
  filter(n >=400) %>%
  mutate(n = ifelse(sentiment == "negative", -n, n)) %>%
  mutate(term = reorder(term, n)) %>%
  ggplot(aes(term, n, fill = sentiment)) +
  geom_bar(stat = "identity") +
  ylab("Contribution to sentiment") +
  coord_flip()
```



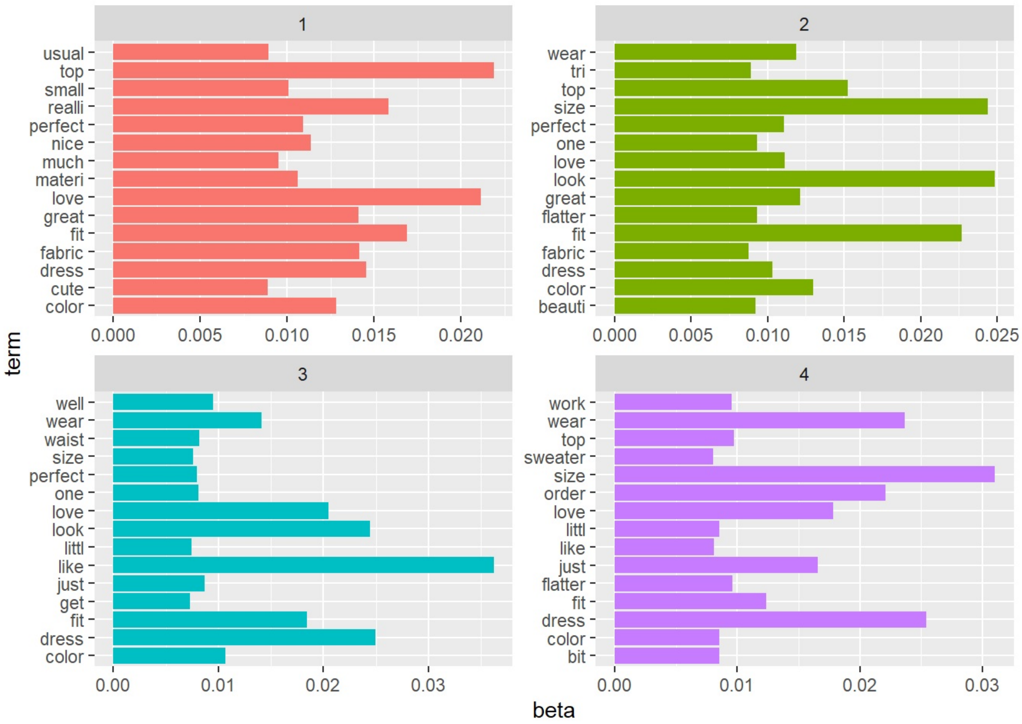

```
#We can use the LDA() function from the topicmodels package, setting k = 4,
#to create a four-topic LDA model.

# set a seed so that the output of the model is predictable
Mod_lda <- LDA(z_dtm, k = 4, control = list(seed = 1234))
Mod_lda
```

```
## A LDA_VEM topic model with 4 topics.
```

```
# extracting the per-topic-per-word probabilities, called "beta", from the model.

Mod_topics <- tidy(Mod_lda, matrix = "beta")
Mod_topics
```

```
## # A tibble: 52,912 × 3
##    topic term        beta
##    <int> <chr>       <dbl>
##  1     1 absolut 0.00131
##  2     2 absolut 0.0000679
##  3     3 absolut 0.00220
##  4     4 absolut 0.00167
##  5     1 comfort 0.00365
##  6     2 comfort 0.00750
##  7     3 comfort 0.00613
##  8     4 comfort 0.00313
##  9     1 sexi    0.000381
## 10     2 sexi    0.0000893
## # … with 52,902 more rows
```

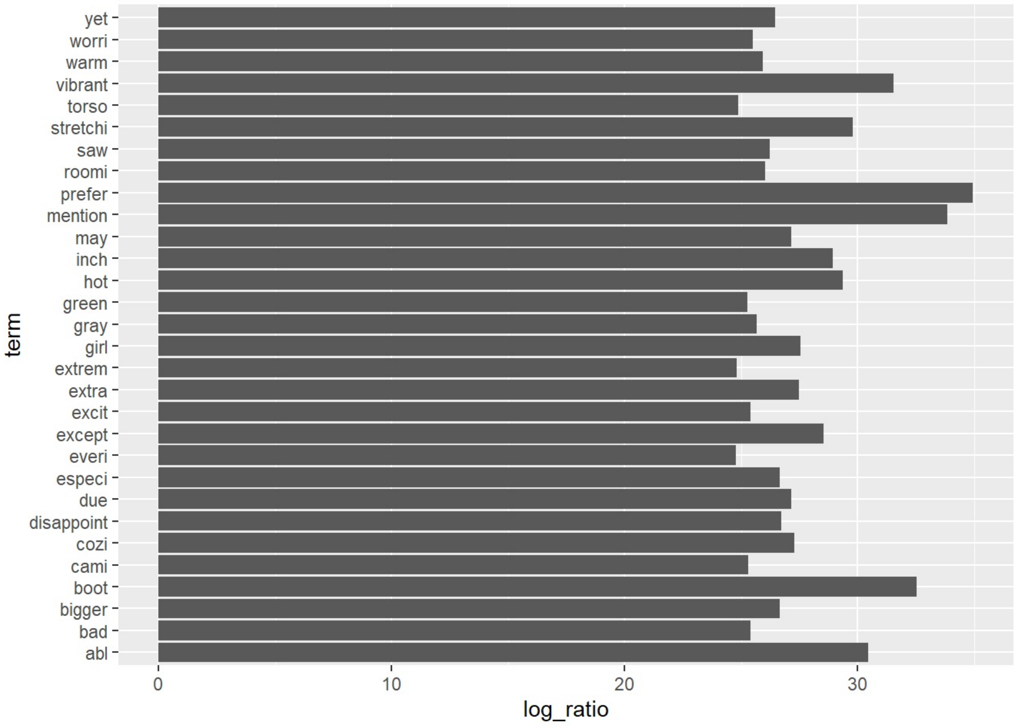
```
#Examine most common 15 terms in each topic
Mod_top_terms <- Mod_topics %>%
  group_by(topic) %>%
  top_n(15, beta) %>%
  ungroup() %>%
  arrange(topic, -beta)

Mod_top_terms %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  coord_flip()
```




```
#As an alternative, we could consider the terms that had the greatest difference in beta between topic 1 ,topic 2
, topic 3 and topic 4.

beta_spread <- Mod_topics %>%
  mutate(topic = paste0("topic", topic)) %>%
  spread(topic, beta) %>%
  filter(topic1 > .001 | topic2 > .001 | topic3 > .001 | topic4 > .001) %>%
  mutate(log_ratio = log2(topic4 / topic3 / topic2 / topic1))

beta_spread
```

```
## # A tibble: 337 × 6
##    term       topic1    topic2   topic3   topic4 log_ratio
##    <chr>       <dbl>     <dbl>    <dbl>    <dbl>     <dbl>
##  1 abl      0.000580 0.0000135 0.000162 0.00187       30.5
##  2 absolut  0.00131  0.0000679 0.00220  0.00167       23.0
##  3 actual   0.000171 0.000385  0.00376  0.000379      20.5
##  4 add      0.00170  0.00142   0.000163 0.000391      19.9
##  5 ador     0.00172  0.00157   0.000791 0.000221      16.7
##  6 agre     0.000126 0.000362  0.00123  0.000658      23.5
##  7 almost   0.000992 0.000256  0.00118  0.00180       22.5
##  8 also     0.00282  0.00418   0.00523  0.00378       15.9
##  9 although 0.00171  0.000493  0.000131 0.000284      21.3
## 10 alway    0.000587 0.000230  0.00132  0.00151       23.0
## # … with 327 more rows
```

```
#Examine top 30 beta scores
beta_top_terms <- beta_spread %>%
  top_n(30, log_ratio)

beta_top_terms %>%
  ggplot(aes(term, log_ratio)) +
  geom_col(show.legend = FALSE) +
  coord_flip()
```

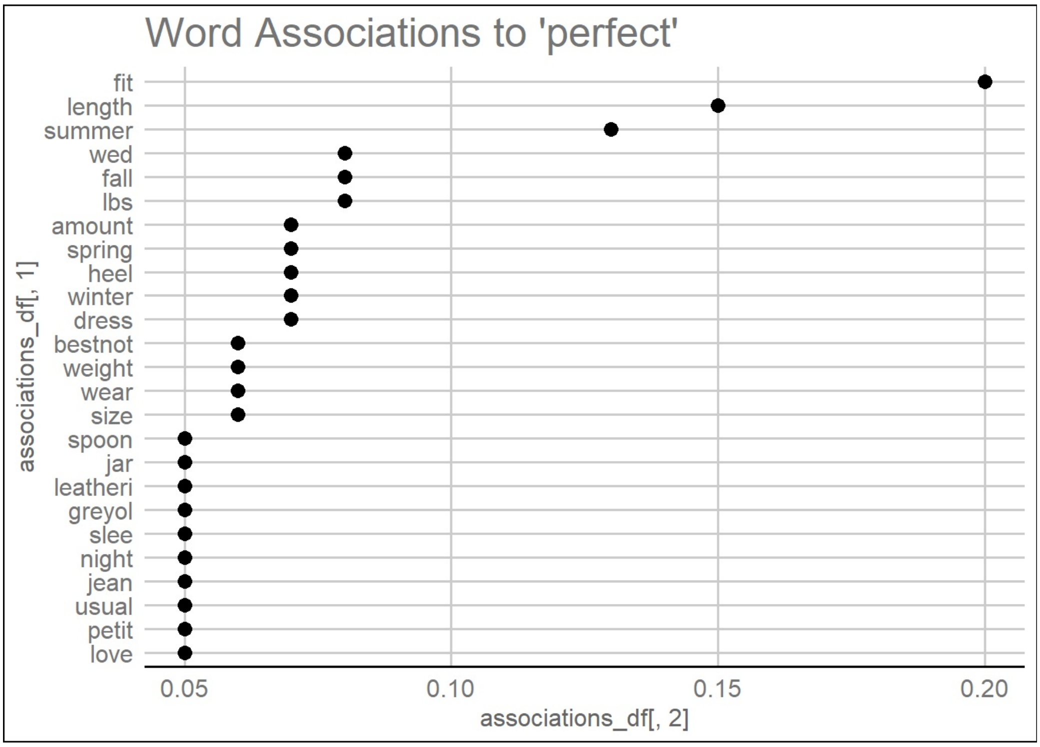



```
#To get the document topic probability
#We can examine the per-document-per-topic probabilities, called "gamma"
Mod_documents <- tidy(Mod_lda, matrix = "gamma")
Mod_documents
```

```
## # A tibble: 90,564 × 3
##    document topic gamma
##    <chr>    <int> <dbl>
##  1 1            1 0.250
##  2 2            1 0.246
##  3 3            1 0.256
##  4 4            1 0.249
##  5 5            1 0.249
##  6 6            1 0.249
##  7 7            1 0.240
##  8 8            1 0.251
##  9 9            1 0.250
## 10 10           1 0.245
## # … with 90,554 more rows
```

Word associations Word association is a way of identifying the correlation between two words in a DTM or TDM. It is another way to recognize terms that are frequently used together. The word association plot reveals a relationship between various terms and the word “perfect” in our corpus.However, we will be making use of the TDM.

```
associations <- findAssocs(examine_tdm, "perfect", 0.05)

associations_df <- list_vect2df(associations)[, 2:3]

ggplot(associations_df, aes(y=associations_df[, 1])) +
  geom_point(aes(x = associations_df[, 2]),
             data = associations_df, size = 3) +
  ggtitle("Word Associations to 'perfect'") +
  theme_gdocs()
```

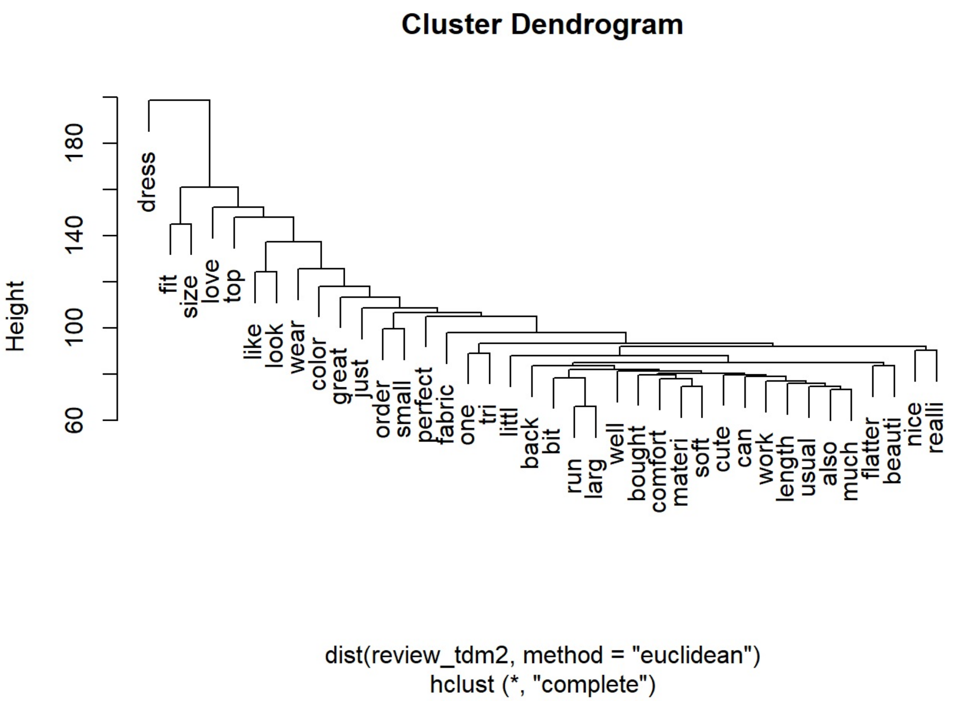


The word “perfect” are most commonly associated with “fit” and “length”.

Word Clustering Based on the distance in frequency, word clustering helps identify word groups that are frequently used together. This is a method of dimension reduction. It aids in putting words into clusters that are related. We can now use a dendogram to visualize the word cluster as follow:

```
review_tdm2 <- removeSparseTerms(examine_tdm, sparse = 0.9)
dnd <- hclust(d = dist(review_tdm2,method = "euclidean"), method="complete")
```

```
plot(dnd)
```



The cluster dendogram reveals the relationships between various word groups. For instance, the terms “usual” ,“also”, and “much” have been used together. The cluster identifies the most frequently occurring group of words since the clustering is based on frequency distances.

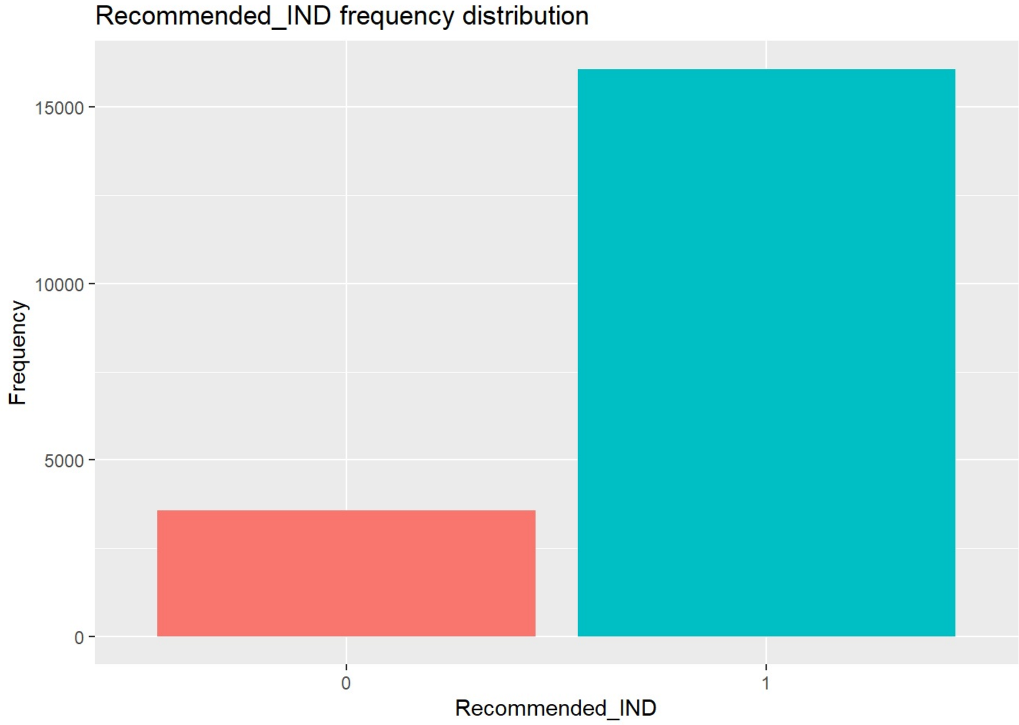
Task D Further exploration

```
#Removing all the NA's in the Dataset
datamod <- na.omit(Dataset)
```

We wish to investigate goods that costumers have not recommended.

Firstly,lets check for the frequency distribution of the recommended_IND

```
#convert to factor
datamod$Recommended_IND <- as.factor(datamod$Recommended_IND)

#Frequency distribution of Recommended IND sorted by frequency count
ggplot(datamod, aes(x = reorder(Recommended_IND, -table('Recommended IND')[Recommended_IND]), fill = Recommended_
IND)) +
  geom_bar() +
  ggtitle("Recommended_IND frequency distribution") +
  xlab("Recommended_IND") +
  ylab("Frequency") +
  theme(legend.position = "none")
```



Customers indicate that they would recommend the product more

want to know the total count and percentage of recommendations(Recommended and Not recommended) using frequency table

```
#frequency table for Recommended_IND
datamod %>%
  group_by(Recommended_IND) %>%
  summarize(TotalCount = n())%>%
            mutate(Prop = round(TotalCount/sum(TotalCount)*100, digits = 2))
```

```
## # A tibble: 2 × 3
##   Recommended_IND TotalCount  Prop
##   <fct>                <int> <dbl>
## 1 0                     3575  18.2
## 2 1                    16087  81.8
```

Now we know that 81.82% of customers recommend the products

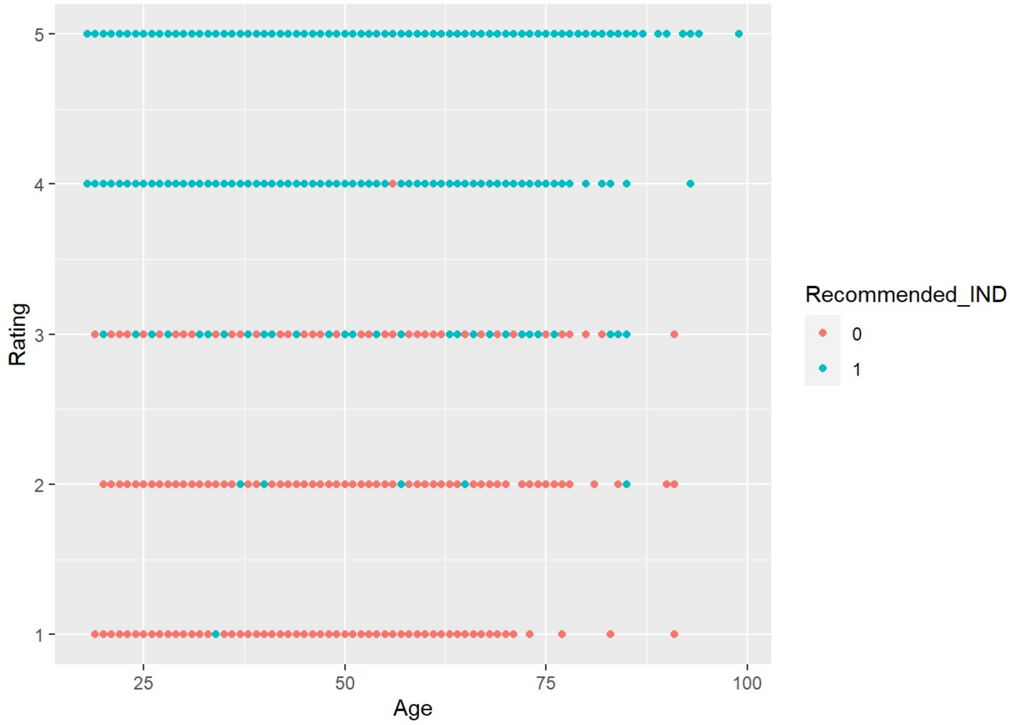
```
datamod %>%
  group_by(Recommended_IND) %>%
  summarize(Average_Rating = mean(Rating), TotalCount= n())
```

```
## # A tibble: 2 × 3
##   Recommended_IND Average_Rating TotalCount
##   <fct>                    <dbl>      <int>
## 1 0                         2.31       3575
## 2 1                         4.60      16087
```

the table indicates a decrease in rating from customers who do not recommend to the ones that recommend.This analysis could help identify the specific areas in which improvements are needed to increase customer satisfaction and consequently,the likelihood of recommendations.

Observing product that were not recommended by the customers

```
#Using a scatter plot
ggplot(datamod, aes(x=Age, y= Rating, color = Recommended_IND)) +
  geom_point()
```



This demonstrates that you recommend products to other customers when you rate them highly. The chart also demonstrates that an average rating of 3 results in roughly equal numbers of recommendations and non-recommendations, while ratings of 4-5 indicate good ratings (recommended) and ratings of 1-2 indicate poor ratings (not recommended).

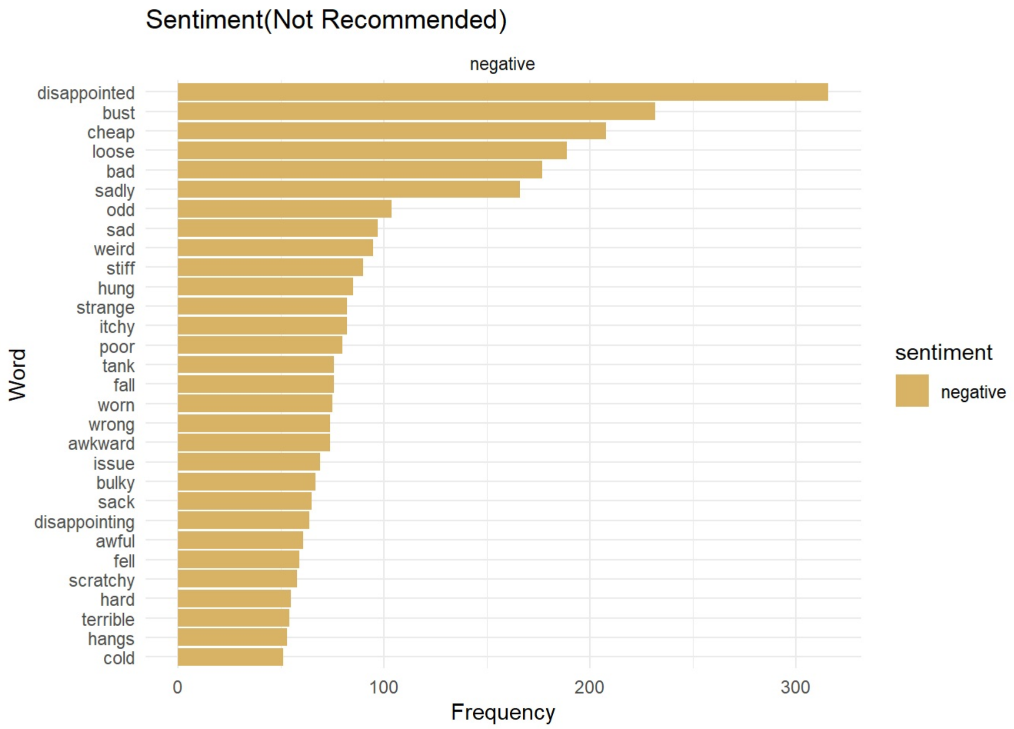
Visualizing the negative sentiment from customers(that is,customers that will not recommend)

```
#check negative sentiment
tokens_nonrec <- datamod %>%
  filter(Recommended_IND == 0) %>%
  unnest_tokens(word, Review_Text) %>%
  anti_join(stop_words)
```

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

```
#using bing for the tokenization to determine wether the words are positive or negative
sentiment_nonrec <- tokens_nonrec %>%
  inner_join(get_sentiments("bing"), by = "word", multiple = "all")

#now to visualise the negative sentiment(Not recommended products)
sentiment_nonrec %>%
  count(word, sentiment) %>%
  filter(sentiment == "negative") %>%
  top_n(30) %>%
  ungroup() %>%
  arrange(desc(sentiment), n) %>%
  ggplot(aes(x=n, y = reorder(word, n), fill = sentiment)) +
  geom_col() +
  facet_wrap(~sentiment, scales = "free") +
  labs(title = "Sentiment(Not Recommended)", x = "Frequency", y = "Word") +
  scale_fill_brewer(palette = "BrBG") +
  theme_minimal()
```

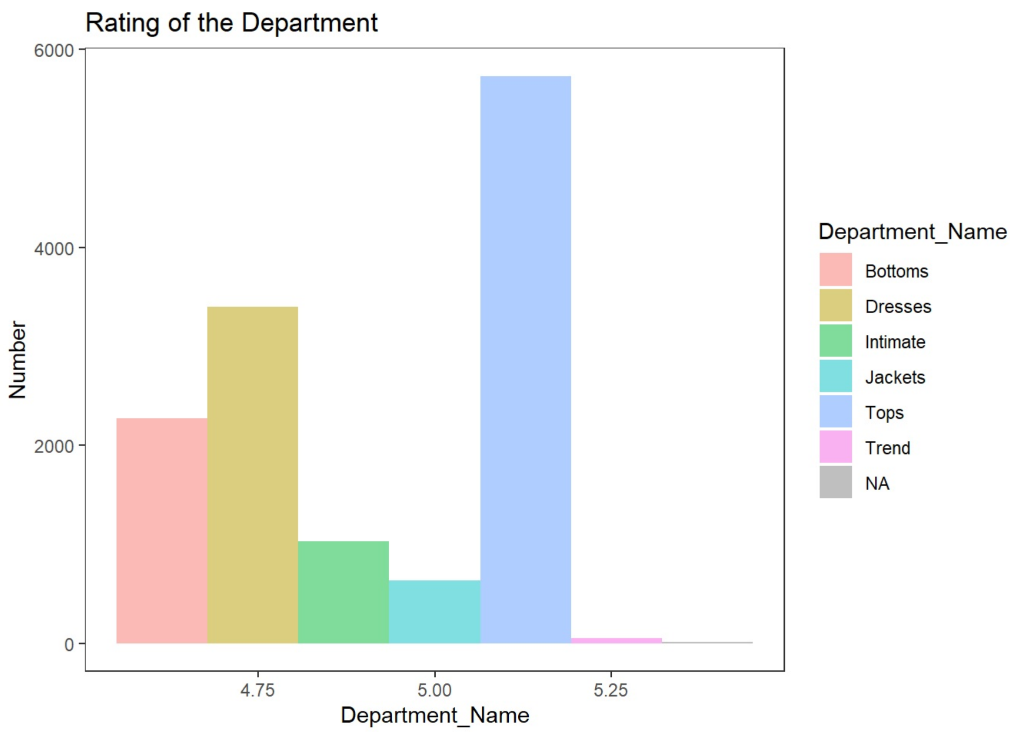
```
## Selecting by n
```



An analysis of the negative sentiment associated with the graphic shows that customers who gave the product a unfavorable audit used words like “disappointed”, “burst”, “cheap”, “loose”, “bad”, etc. to describe the quality of the products.Even words like ‘strange’ and ‘awkward’ suggest that some products are not what they appear to be, or perhaps not what they expect. Through this, the store should try to improve the quality of the products and please the customers with the product.

Additional visualisations

```
#Frequency distribution
Dataset %>%
  mutate(Department_Name) %>%
  filter(Rating == 5)%>%
  ggplot(aes(Rating, fill = Department_Name)) +
  geom_bar(position = "dodge", alpha = 0.5) +
  theme_bw() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank()) +
  labs(title = "Rating of the Department",
       x = "Department_Name",
       y = "Number")
```



In the visualization, the Tops appear to be the most popular picks and the least popular is the trend.

```
#Wordcloud with shape
library(wordcloud2)
```

```
## Warning: package 'wordcloud2' was built under R version 4.2.2
```

```
wordcloud2(check_word_freq, size= 0.3,  shape = "star")
```

Using the frequent term count,we visualize a wordcloud in a STAR shape with the help of wordcloud2 library