Table of Contents

Introduction:

Electronics and technology is a dynamic category of consumer products which will always have demand but evolves fast. Various retailers sell the same products all over the world and it is difficult to maintain the same level of service over multiple locations. Hence, when choosing retailers and distributing products to them, manufacturers of electronics need some way to ensure the commitment of quality will be abide by. Hence, managers need to find primary feedback from customers to get unbiased insight while also having reach to customer feedbacks over greater geographical regions.

Trustpilot is the world's most popular site for customer reviews about companies and it acts as a third party to enable customers to provide their honest opinions. This repertoire of information is quite good for the purposes of text mining to get business insights and using this, we can easily figure out the customer sentiments and service areas to improve on. Managers can also potentially utilize the results of the analysis to understand the elements to improve on and which stores should be removed from retailers' list.

Thus, text mining insights seem necessary in this scenario and thus the project is taken up.

Part A:

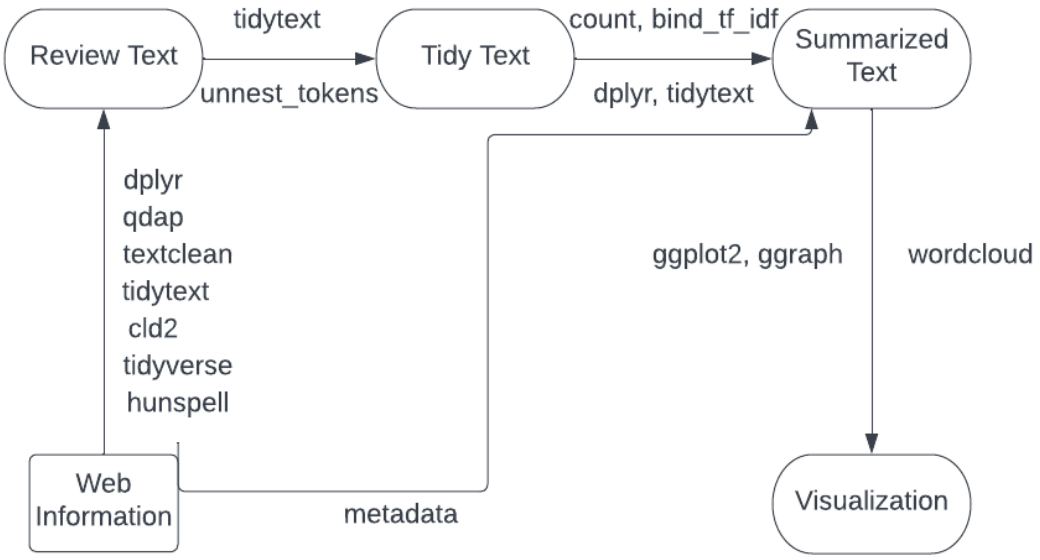
A.1: Section Plan

In this section, we will be performing the bag of words analysis. After relevant analysis, the results will enable us to find the important words by ratings. The word analysis will be done in different n-gram levels (unigram, bigram and trigram).

Upon finding out the important words, we will visualize them in plots and word clouds.

Finally, we will also perform some analysis to connect word importance with different components of the available metadata. For this dataset, we will perform each step on the review level as well as on the company level.

Overall, the pipeline for this section can be portrayed as follows:



A.2: Data Preparation

- Loading the relevant libraries

```
library(ggplot2)
library(tidyverse)
library(dplyr)
library(tidytext)
library(jsonlite)
library(tidyr)
library(readr)
library(topicmodels)
library(geojsonR)
library(stringr)
library(dplyr)
library(tidyr)
library(stm)
library(udpipe)
library(cld2)
library(wordcloud)
library(xml2)
library(rvest)
library(janeaustenr)
library(data.table)
library(rjson)
library(lubridate)
library(textcat)
library(rJava)
library(qdap)
library(magrittr)
library(jsonlite)
library(topicmodels)
library(geojsonR)
library(hunspell)
library(ggrepel)
library(gridExtra)
library(widyr)
library(igraph)
library(textstem)
library(ggraph)
library(stringr)
library(future)
library(future.apply)
library(influential)
library(stringi)
library(tm)
```

We will now perform web-scraping with rvest to retrieve information from Trustpilot webpage and compile them into a dataset.

```
   #Creating a dataset to store all the scraped data
rm(store_reviews)
store_reviews <- data.frame()

#Defining functions to repeatedly perform specific crawling
get_description = function(store_links) {
  store_page = read_html(store_links)
  description = store_page %>% html_nodes("div.card_cardContent__sFUOe") %>%
html_node("div.styles_container__9nZxD") %>% html_text() %>% .[c(1,2,3)] %>%
paste0(collapse = " ")
  return(description)
}

get_email = function(store_links) {
  store_page = read_html(store_links)
  email = store_page %>% html_nodes("div.card_cardContent__sFUOe") %>% html_n
ode(".styles_contactInfoElement__SxlS3:nth-child(1) .styles_contactInfoLink__
gMJWX") %>% html_text() %>% .[c(1,2,3)] %>% paste0(collapse = " ")
  return(email)
}

get_phone = function(store_links) {
  store_page = read_html(store_links)
  phone = store_page %>% html_nodes("div.card_cardContent__sFUOe") %>% html_n
ode(".styles_contactInfoElement__SxlS3+ .styles_contactInfoElement__SxlS3 .st
yles_contactInfoLink__gMJWX") %>% html_text() %>% .[c(1,2,3)] %>% paste0(coll
apse = " ")
  return(phone)
}

get_address = function(store_links) {
  store_page = read_html(store_links)
  address = store_page %>% html_nodes("div.card_cardContent__sFUOe") %>% html
_node("ul.typography_typography__QgicV") %>% html_text() %>% .[c(1,2,3)] %>%
paste0(collapse = " ")
  return(address)
}

#Looping through multiple pages and subpages and retrieving data into the des
ired dataset
for (page_result in seq(from = 1, to = 750, by = 15)) {
  rm(sub_contents)
  sub_contents <- data.frame()

  link = paste0("https://www.trustpilot.com/categories/travel_agency?page=",
```

```
page_result)
  page = read_html(link)

  name = page %>% html_nodes("div.styles_content__3mwYr") %>% html_node(".sty
les_displayName__1LIcI") %>% html_text()


  overall = page %>% html_nodes("div.styles_rating__2FRLX") %>% html_nodes("s
pan.styles_mobile__17MXv") %>% html_text()

  category = page %>% html_nodes("div.card_cardContent__3Idve") %>% html_node
("div.styles_wrapper__nFO8_") %>% html_text() %>% sub("·.*", "", .)

           Sys.sleep(1)
  total_reviews = page %>% html_nodes("div.styles_rating__2FRLX") %>% html_no
de("p.typography_typography__23IQz") %>% html_text()


  store_links <- page %>%
                 html_nodes("div.paper_paper__29o4A") %>%
                 html_node(".styles_linkWrapper__3x9X7") %>%
                 html_attr("href") %>%
                 paste0("https://uk.trustpilot.com",.)

  store_description = sapply(store_links, FUN = get_description, USE.NAMES =
F)
  Sys.sleep(2)
  phone_number = sapply(store_links, FUN = get_phone, USE.NAMES = F)

  email = sapply(store_links, FUN = get_email, USE.NAMES = F)
  Sys.sleep(2)
  store_address = sapply(store_links, FUN = get_address, USE.NAMES = F)

  for (i in store_links) {
    rm(sub_contents_catch)
    sub_contents_catch <- data.frame()

    for (subpage_result in seq(from = 1, to = 3, by = 1)) {
      sub_links = paste0(i, "?page=", subpage_result)
      sub_page = read_html(sub_links)

      reviewer = sub_page %>% html_nodes("div.styles_consumerDetailsWrapper__
p2wdr") %>%
                 html_node("div.typography_typography__QgicV") %>%
                html_text(trim = TRUE) %>% paste0(collapse = "//")

           Sys.sleep(1)
      review_summary = sub_page %>% html_nodes("div.styles_reviewContent__0Q2
Tg") %>%
                 html_node("a.link_internal__7XN06") %>%
```

```
            html_text(trim = TRUE) %>% paste0(collapse = "//")

          Sys.sleep(2)

     review = sub_page %>% html_nodes("div.styles_reviewContent__0Q2Tg") %>%
 html_node("p.typography_typography__QgicV") %>% html_text() %>%
       paste0(collapse = "//")

          Sys.sleep(2)
     date = sub_page %>% html_nodes("div.styles_reviewHeader__iU9Px") %>%
           html_node("div.typography_typography__QgicV time") %>%
            html_attr("datetime") %>% str_sub(.,1, 10) %>% paste0(collapse =
 "//")


     country = sub_page %>% html_elements(xpath = "//div[contains(@class, 's
tyles_consumerDetailsWrapper__p2wdr')]") %>%
           html_elements(xpath = ".//div[contains(@class, 'styles_consumerEx
traDetails__fxS4S')]") %>%
           html_node(".styles_detailsIcon__Fo_ua+ .styles_detailsIcon__Fo_ua
") %>%
           html_text(trim = TRUE) %>% paste0(collapse = "//")

          Sys.sleep(2)
     reviewer_rating = sub_page %>% html_nodes("div.styles_reviewHeader__iU9
Px") %>%
           html_node("div.star-rating_starRating__4rrcf img") %>% html_attr(
"alt") %>%  paste0(collapse = "//")

     sub_contents_catch = rbind(sub_contents_catch, data.frame(reviewer, rev
iew_summary, review, date, country, reviewer_rating))

     ifelse(subpage_result == 5, t(sub_contents_catch) %>% as.data.frame() %
>% unite(col = "1", sep = "//") %>% t(.) -> sub_contents_catch, print(paste("
Sub_Page", subpage_result)))

   }
   sub_contents = rbind(sub_contents, sub_contents_catch)
 }


 store_reviews <- rbind(store_reviews,
                        data.frame(name, overall, category, store_descriptio
n, phone_number, email, store_address, sub_contents, stringsAsFactors = FALSE
))


 print(paste("Page:", page_result))
}
#saveRDS(store_reviews, "store_reviews_final.rds)
```

- Transforming the scraped data

Now that the raw data is imported, we will perform some steps to convert it into analyzable form.

```
   #Reading the saved file
dataset <- readRDS("store_reviews_finals.rds")
dataset <- distinct(dataset)

#Correcting an anomaly
dataset$review[400] <- str_replace(dataset$review[400], "https://earnapp", "h
ttps:earnapp")
dataset$review[68] <- str_replace(dataset$review[68], " // ", "")
dataset$review[609] <- str_replace(dataset$review[609], " //M", "M")
dataset$review[862] <- str_replace(dataset$review[862], "on 2//7/09-as", "on
2/7/09-as")
dataset$review[902] <- str_replace(dataset$review[902], "https:// t.me", "htt
ps: t.me")
dataset$review[930] <- str_replace(dataset$review[930], "\\\\WHY///", "WHY")

#Separating the individual reviews
dataset <- dataset %>% separate_rows(7:12, sep = "//")

#Removing duplicates
dataset <- distinct(dataset)
dataset <- na.omit(dataset)
dataset <- dataset %>% filter(reviewer_rating != "")

#Saving the final dataset
#saveRDS(dataset, "raw_data_final.rds")
```

- Cleaning the variables

For the benefit of analysis, we will perform some cleaning to normalize the dataset. We utilized some techniques while scraping to ensure the row numbers match. We will now remove those unnecessary characters.

```
   review_data <- readRDS("raw_data_final.rds")
#Removing unwanted characters
review_data$store_description <- review_data$store_description %>% str_replac
e_all("NA", "")
review_data$email <- review_data$email %>% str_replace_all("NA", "")
review_data$store_address <- review_data$store_address %>% str_replace_all("N
A", "")
review_data$phone_number <- review_data$phone_number %>% str_replace_all("NA"
, "")
review_data$reviewer_rating <- review_data$reviewer_rating %>% str_sub(start
= 6, end = 8)

#Fixing the datatypes
review_data <- review_data %>% separate(date, into = c("year", "month", "day"
```

```
), sep = "-")
review_data <- review_data %>% mutate(year = as.numeric(year),
                                      month = as.numeric(month),
                                      day = as.numeric(day))
review_data <- review_data %>% mutate(date = make_date(year, month, day))
review_data$date <- as.Date(review_data$date, format = "%Y %b,%d")

#Formatting the target variable
review_data$overall <- as.numeric(review_data$overall)
review_data$overall <- round(review_data$overall)
review_data$reviewer_rating <- as.numeric(review_data$reviewer_rating)

review_data <- review_data %>% arrange(desc(date))
```

The dataset is now ready for text analysis and is in normal form.

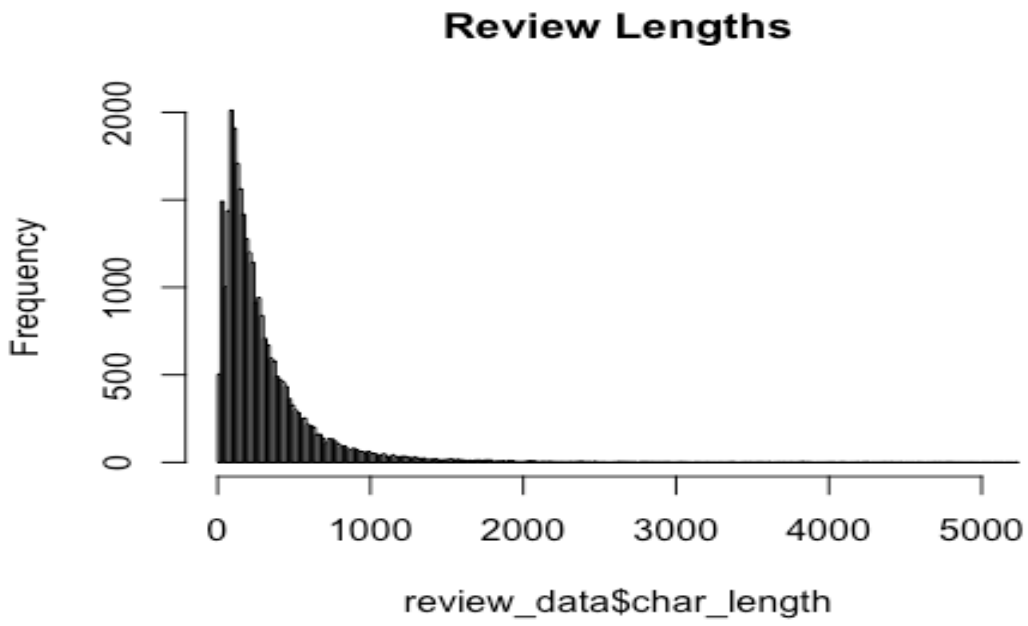
We will now perform some filtering on the reviews to bring it into a better state to analyze that is efficient.

```
   #Remove companies with less than 5 reviews
remove_names <- review_data %>%
                group_by(name) %>%
                summarise(total = n()) %>%
                filter(total <= 5) %>% pull(name)

review_data <- review_data %>%
               filter(!(name %in% remove_names))

#Assign unique keys to the stores
name_id <- review_data %>% distinct(name)
name_id$company_id <- 1:nrow(name_id)
review_data <- name_id %>%
               inner_join(review_data)

#Create a new key to track each review
review_data$review_id <- 1:nrow(review_data)

#Create new object with only the full review text and the review_id
reviews<- review_data %>%
          dplyr::select(review_id, review_summary, review, reviewer_rating)

reviews <- reviews %>%
           mutate(reviewText =
           paste(
             coalesce(review_data$review_summary, ""),
             coalesce(review_data$review, "")
             )
         )  %>%
  dplyr::select(-(review_summary:review))

review_data <- review_data %>% left_join(reviews)
```

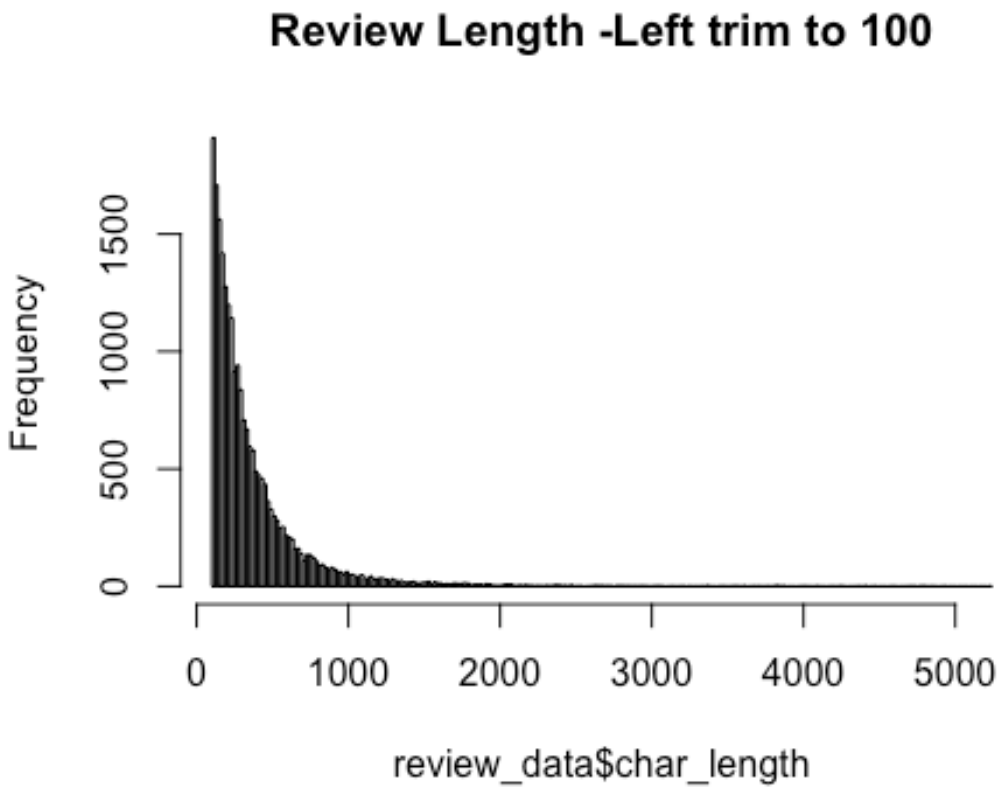
- Cleaning the reviews

We will perform some basic text cleaning functions, which can be utilized throughout all the parts of this project. Other addition cleanings will be performed as required in each section.

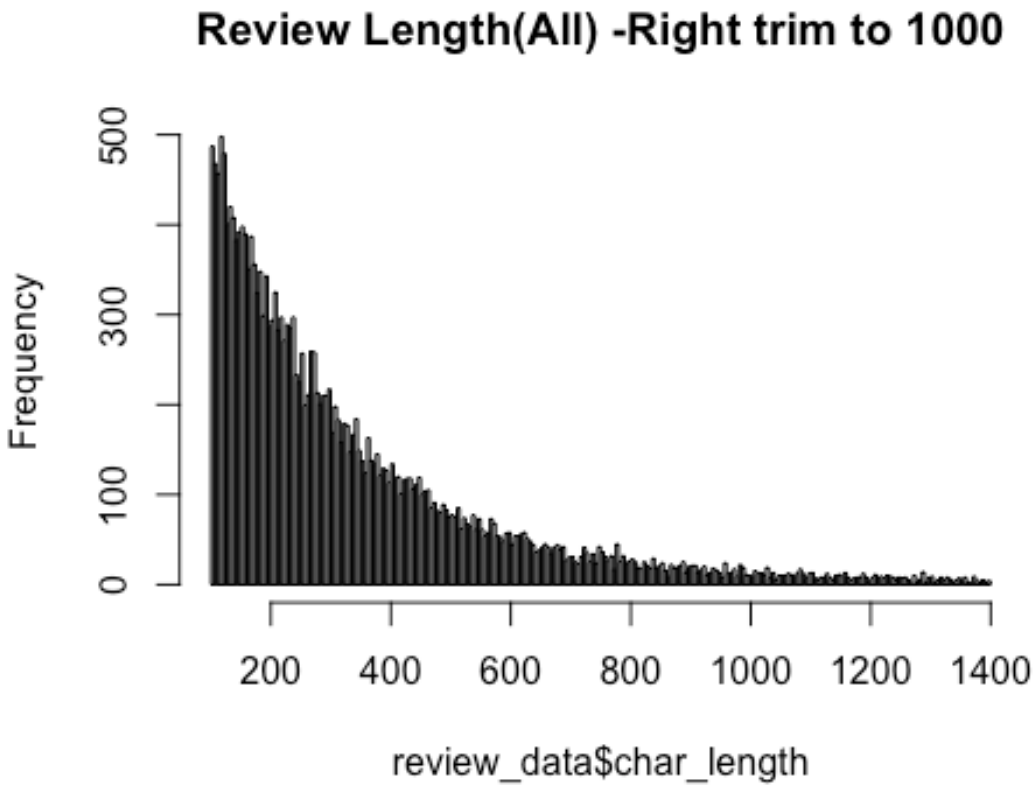
```
   #Replacing the digits with empty space
review_data$reviewText <- gsub('[[:digit:]]+',' ', review_data$reviewText)

#Replacing the punctuations with empty space
review_data$reviewText <- gsub('[[:punct:]]+',' ', review_data$reviewText)

#Additional cleaning
review_data$reviewText <- bracketX(review_data$reviewText)
review_data$reviewText <- replace_contraction(review_data$reviewText)
review_data$reviewText <- replace_symbol(review_data$reviewText)

# Remove spaces and newlines
review_data$reviewText <- gsub("\n", " ", review_data$reviewText)
review_data$reviewText <- gsub("^\\s+", "", review_data$reviewText)
review_data$reviewText <- gsub("\\s+$", "", review_data$reviewText)
review_data$reviewText <- gsub("[ |\t]+", " ", review_data$reviewText) #space
s


#Getting the character lengths
review_data$char_length <- nchar(review_data$reviewText)

#Plotting the lengths
hist(review_data$char_length, breaks = 300, main = "Review Lengths")
```



```
   #Filtering for minimum length of 100
review_data <- review_data %>% filter(char_length>100)
hist(review_data$char_length,breaks = 300,main = "Review Length -Left trim to
 100")
```




```
   #Filtering for maximum length of 1400
review_data <- review_data %>% filter(char_length<1400)
hist(review_data$char_length,breaks = 300,main = "Review Length(All) -Right t
rim to 1000")
```




```
   #Cleaning the language of the reviews
review_data$reviewText <- iconv(review_data$reviewText)
review_data$language <- cld2::detect_language(review_data$reviewText)
review_data %>% filter(language =="en") -> review_data_en
```

- Cleaning the store descriptions

We will apply basic cleaning functions on the store descriptions that will be used throughout the project.

```
   #Replacing the digits with empty space
review_data_en$store_description <- gsub('[[:digit:]]+',' ', review_data_en$s
tore_description)

#Replacing the punctuations with empty space
review_data_en$store_description <- gsub('[[:punct:]]+',' ', review_data_en$s
tore_description)

#Additional cleaning
review_data_en$store_description <- bracketX(review_data_en$store_description
)
review_data_en$store_description <- replace_contraction(review_data_en$store_
description)
review_data_en$store_description <- replace_symbol(review_data_en$store_descr
iption)

# Remove spaces and newlines
review_data_en$store_description <- gsub("\n", " ", review_data_en$store_desc
ription)
review_data_en$store_description <- gsub("^\\s+", "", review_data_en$store_de
scription)
review_data_en$store_description <- gsub("\\s+$", "", review_data_en$store_de
scription)
review_data_en$store_description <- gsub("[ |\t]+", " ", review_data_en$store
_description) #spaces

#Lemmatizing
lemma <- review_data_en %>% dplyr::select(company_id, store_description) %>%
distinct(.)
lemma <- lemma %>% unnest_tokens(word, store_description)
lemma$word <- lemmatize_words(lemma$word)
lemma <- lemma %>% group_by(company_id) %>% summarise(store_description = pas
te(word, collapse = " "))
review_data_en$store_description <- NULL
review_data_en <- review_data_en %>% left_join(lemma)

#Cleaning the language
review_data_en$store_description <- iconv(review_data_en$store_description)
```

- Spell Check

We will use the 'hunspell' dictionary spelling suggestions to deal will the spelling mistakes identified in the reviews. Then, using judgement, we will manually fix the mistakes and integrate the corrected reviews in the dataset.

```
   #Getting the reviews column
reviews <- review_data_en %>% arrange(review_id) %>% dplyr::select(review_id,
reviewText)

#Tokenizing the reviews
tokens_for_spellcheck <- unnest_tokens(reviews,word,reviewText)

#Getting the unique words
unique_words <- unique(tokens_for_spellcheck$word)
head(unique_words)
   ## [1] "poor"    "service" "i"       "made"    "an"      "over"
   length(unique_words)
   ## [1] 24440
   #Finding the misspelled words
mispells <- hunspell(unique_words)

#Getting the unique spelling mistake
mispells <- unique(unlist(mispells))
length(mispells)
   ## [1] 9742
   #Getting correct word suggestions
suggestive_words <- hunspell_suggest(mispells)
suggestive_words <- unlist(lapply(suggestive_words, function(x) x[1]))
head(suggestive_words)
   ## [1] "awns"      "talk talk" "did"       "TV"        "fiber"     "would"
   mistakes.list <- as.data.frame(cbind(mispells,suggestive_words))
freq_mistake <- count(tokens_for_spellcheck,word)
freq_mistake <- inner_join(freq_mistake, mistakes.list, by = c("word" = "misp
ells"))
words_suggestion<-arrange(freq_mistake,desc(n))
word_NA <- words_suggestion %>% filter(is.na(suggestive_words))

#Manually dealing with the spelling suggestions
#write.csv(words_suggestion,"words_suggestion.csv")
#write.csv(word_NA,"word_NA.csv")

#Reading the prepared data
words_suggestion <- read.csv("words_suggestion.csv")
words_suggestion <- words_suggestion %>% dplyr::select(-X) %>% na.omit()
word_NA <- read.csv("word_NA.csv")
word_NA <- word_NA %>% dplyr::select(-X) %>% na.omit()

words_suggestions <- rbind(words_suggestion,word_NA)
```

```
words_suggestions <- arrange(words_suggestions)
sum(is.na(words_suggestions))
   ## [1] 0
   #Saving the final data
#write.csv(words_suggestions, "words_suggestions.csv")

#Replacing the mistakes words with suggestive ones
word.list <- read.csv("words_suggestions.csv", stringsAsFactors = FALSE)
mistake.words <- paste0(" ", word.list$word, " ")
correct.words <- paste0(" ", word.list$suggestive_words, " ")

mistake.replace <- function(df) {
 df$reviewText <- stri_replace_all_regex(df$reviewText, mistake.words, correc
t.words, vectorize_all = FALSE)
 return(df)
}

#Defining the number of cores
ncores <- 6L
plan(multiprocess, workers = ncores)

# split comments based on available cores
corpus_splitted <- split(reviews, seq(1, nrow(reviews), by=5))
reviews <- future_lapply(corpus_splitted, mistake.replace)
reviews <- rbindlist(reviews)
reviews <- as.data.frame(reviews)
reviews <- reviews %>% arrange(review_id)

#Creating a backup of the correct reviews
#saveRDS(reviews, "spell_mistake_correction.rds")

#Replacing the reviews with the corrected reviews
review_data_en$reviewText <- reviews$reviewText

#Creating a backup
#saveRDS(review_data_en, "review_data_en.rds")
```

A.4: Bag of Words Analysis

Now that we have filtered and trimmed review words, we can take the tokens and perform tokenization to look for terms.

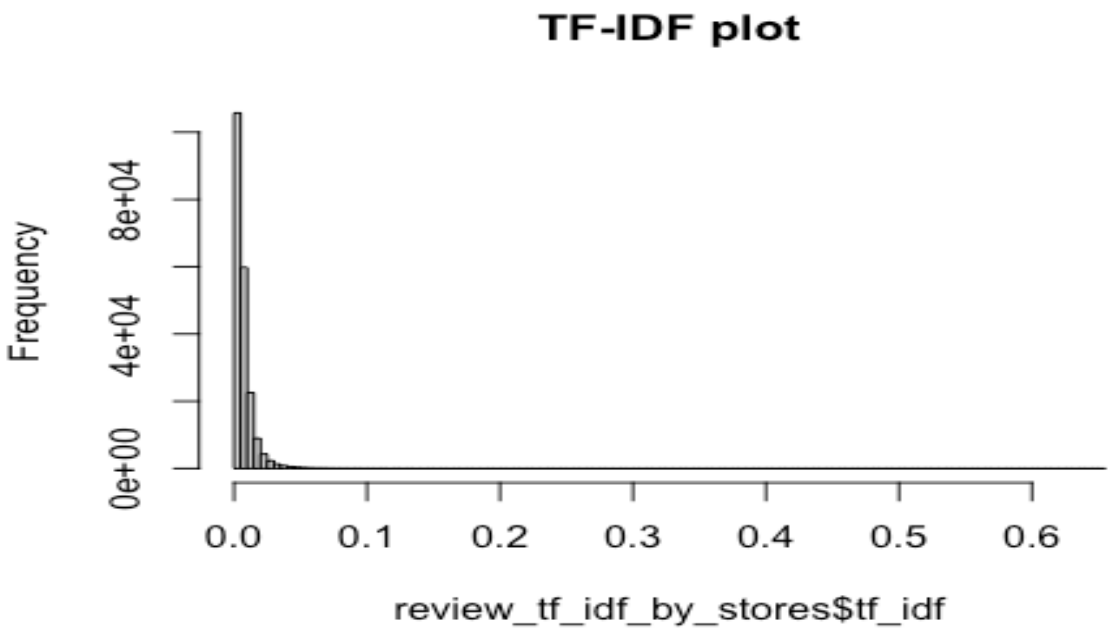
```
   #Grouping the reviews by stores
review_data_by_stores <- review_data_en %>%
 unnest_tokens(word,reviewText)%>%
 group_by(company_id) %>%
 summarise(grouped_reviews = paste(word,collapse = " "))

#saveRDS(review_data_by_stores, "review_data_by_stores.rds")

#Splitting the dataset
split_size <- 100

tokens_list <- split(review_data_by_stores,
                     rep(1:ceiling(nrow(review_data_by_stores)/split_size),
                     each=split_size,
                     length.out=nrow(review_data_by_stores)))
# Tokenization
review_tokens_by_stores <- data.frame()

for(i in 1:length(tokens_list)){
      review_tokens_k <- tokens_list[[i]] %>%
      unnest_tokens(word,grouped_reviews) %>%
      count(word,company_id) %>%
      anti_join(stop_words)
      print(i)

      review_tokens_by_stores <- bind_rows(review_tokens_by_stores,review_tok
ens_k)
}
   ## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
   #Lemmatizing the tokens
review_tokens_by_stores$word <- lemmatize_words(review_tokens_by_stores$word)
```

- Filtering the tokens

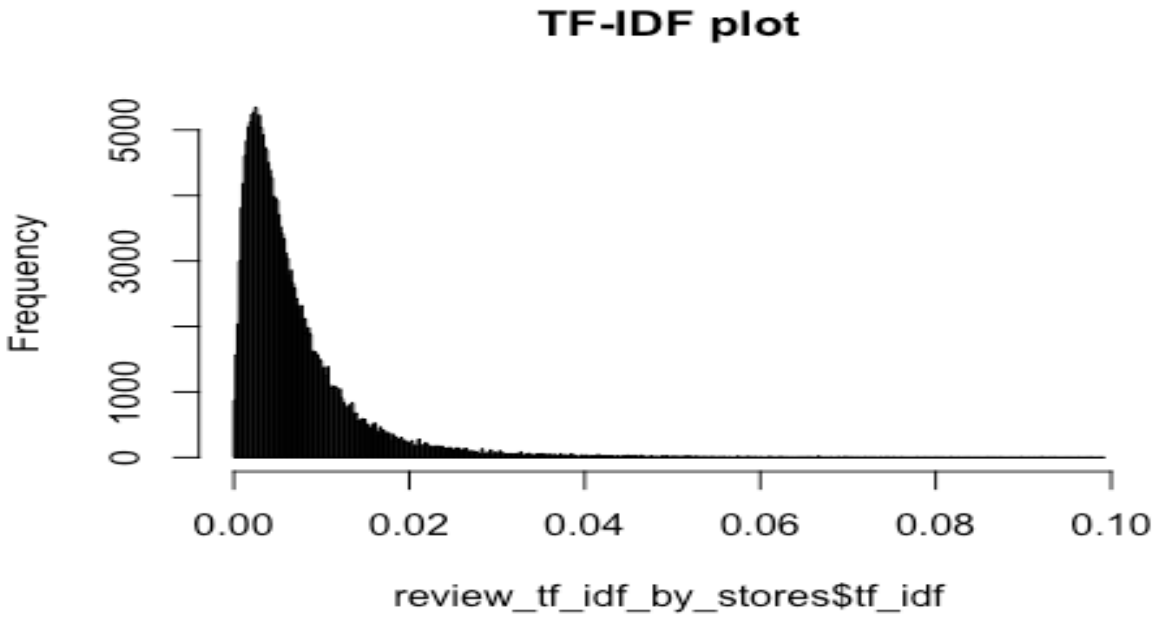
We will try to even trim the reviews by looking at the word lengths. Unnaturally long tokens indicate some abnormal words that might be typos that the cleaning functions didn't detect. Similarly, too short tokens might be prepositions or articles that add no meaning to the analysis.

```
   #Finding the token lengths
review_tokens_by_stores$token_length <- nchar(review_tokens_by_stores$word)

#Checking the distribution of the lengths
view(review_tokens_by_stores %>% group_by(token_length) %>% summarise(total =
n()))

#Removing the short tokens
review_tokens_by_stores <- review_tokens_by_stores %>% filter(token_length >
2)

#Checking the longest tokens
view(review_tokens_by_stores %>% group_by(token_length) %>% summarise(total =
n()) %>% arrange(desc(token_length)))

#Removing excessively long tokens
review_tokens_by_stores <- review_tokens_by_stores %>% filter(token_length<=1
6)

#Creating backups
#saveRDS(review_data_by_stores, "review_data_by_stores.rds")

#Getting the frequency of each token
review_tokens_by_stores <- review_tokens_by_stores %>% count(company_id, word
, sort = TRUE)
get_overall <- review_data_en %>% dplyr::select(., c(2,3)) %>% distinct(compa
ny_id, overall)
review_tokens_by_stores <- review_tokens_by_stores %>% left_join(get_overall)
total_words <- review_tokens_by_stores %>%
                group_by(word) %>%
                 summarize(total = sum(n))
review_tokens_by_stores <- review_tokens_by_stores %>% left_join(total_words)

#Calculating tf_idf of the tokens
review_tf_idf_by_stores <- review_tokens_by_stores %>% count(company_id, word
) %>%
                            bind_tf_idf(word,company_id,n) %>% group_by(word)
 %>% arrange(desc(tf-idf))
```

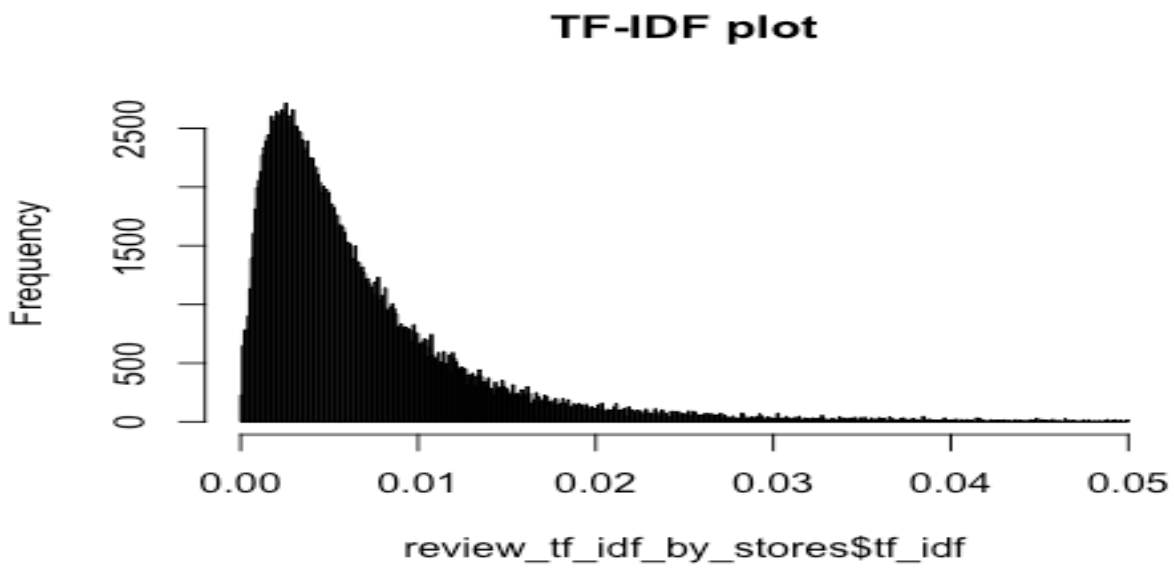
```r
   #Checking the tf-idf distribution
hist(review_tf_idf_by_stores$tf_idf,breaks = 200,main="TF-IDF plot")
```



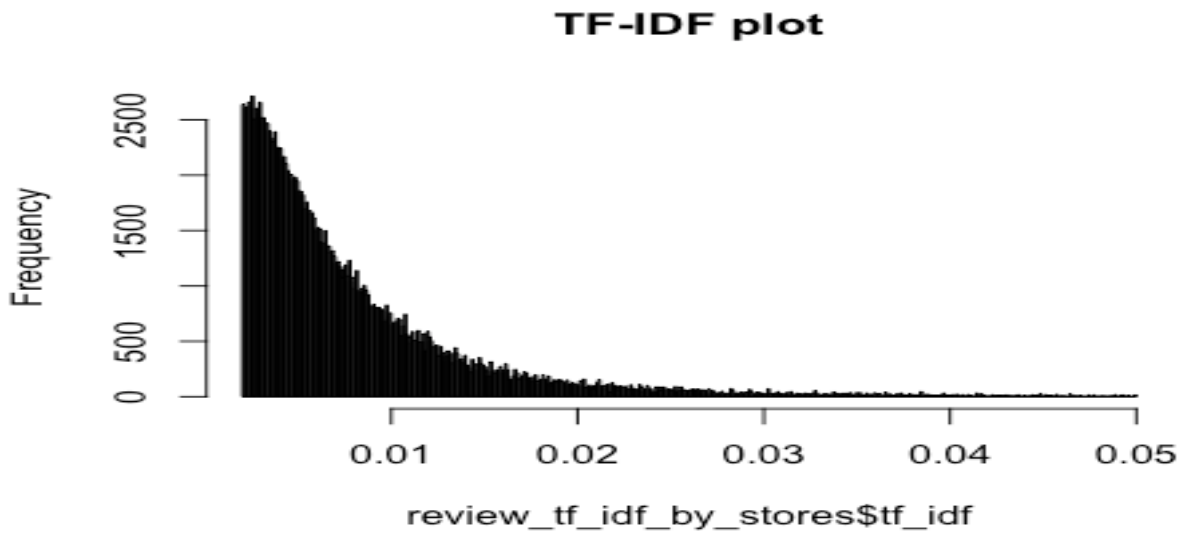

```r
   #Taking the cut-off value from histogram
review_tf_idf_by_stores <- review_tf_idf_by_stores %>%
                           filter(tf_idf < 0.1)
hist(review_tf_idf_by_stores$tf_idf,breaks = 400,main="TF-IDF plot")
```



```
   review_tf_idf_by_stores <- review_tf_idf_by_stores %>%
                              filter(tf_idf<0.05)
hist(review_tf_idf_by_stores$tf_idf,breaks = 400,main="TF-IDF plot")
```




```
   review_tf_idf_by_stores <- review_tf_idf_by_stores %>%
                              filter(tf_idf>0.002)
hist(review_tf_idf_by_stores$tf_idf,breaks = 400,main="TF-IDF plot")
```



```
   review_tf_idf_by_stores %>% group_by(word) %>%
                              summarise(total =n()) %>%
                              arrange(desc(total)) %>%
                              top_n(50)
```

- Visualizing wordclouds by ratings

We now use our filtered and cleaned text to visualize the important words. While visualizing, the necessary stop words are identified through repeated plotting and judgement.

```
   custom_words <- c("class", "please", "job", "shop", "hand", "friend", "dat
e", "service", "review", "product", "customer", "day", "days", "received", "m
oney", "week", "buy", "company", "phone", "don", "ve", "ofcom", "ninja", "moo
nfruit", "cancelled", "virgin", "told", "people", "arrived", "paul", "av", "b
en", "josh", "andy", "tom", "richer", "plugin", "amp", "hifi", "aquiss", "sou
nds", "controller", "zoltan", "ian", "idnet", "glue", "proxy", "helen", "eno"
, "oled", "sypwai", "george", "peter", "panamoz", "pine", "dotsquares", "laur
a", "vape", "amiga", "biz", "technoworld", "proxies", "star", "hesit", "futur
", "easi", "instal", "satisfi", "effici", "solv", "refus", "disgrac", "compet
it", "ensur", "refus", "incr", "wix", "sb", "laskys", "responsiv", "wayv", "d
ab", "novads", "harvey")

tibble(word = custom_words,lexicon = rep("custom",length(custom_words))) %>%
bind_rows(stop_words) -> custom_stopwords
   review_5 <- review_tokens_by_stores %>%
            anti_join(custom_stopwords) %>%
            filter(overall == 5) %>%
            group_by(word) %>%
            summarise(total = mean(total)) %>%
            arrange(desc(total))
```

```
wordcloud(review_5$word, review_5$total, max.words = 25,
          colors = c("gold3", "blue4", "darkgoldenrod4"))
```

Word Cloud for 5 star reviews

```
review_4 <- review_tokens_by_stores %>%
            anti_join(custom_stopwords) %>%
            filter(overall == 4) %>%
            group_by(word) %>%
            anti_join(custom_stopwords) %>%
            summarise(total = mean(total)) %>%
            arrange(desc(total))

wordcloud(review_4$word, review_4$total, max.words = 25,
          colors = c("gold3", "blue4", "darkgoldenrod4"))
```

Word Cloud for 4 star reviews

```
   review_3 <- review_tokens_by_sto
res %>%
             anti_join(custom_stopwo
rds) %>%
             filter(overall == 3) %>
%
             group_by(word) %>%
             anti_join(custom_stopwo
rds) %>%
             summarise(total = mean(
total)) %>%
             arrange(desc(total))

wordcloud(review_3$word, review_3$t
otal, max.words = 25,
          colors = c("gold3", "blue
4", "darkgoldenrod4"))
```

Word Cloud for 3 star reviews

```
   review_2 <- review_tokens_by_sto
res %>%
             anti_join(custom_stopwo
rds) %>%
             filter(overall == 2) %>
%
             group_by(word) %>%
             anti_join(custom_stopwo
rds) %>%
             summarise(total = mean(
total)) %>%
             arrange(desc(total))

wordcloud(review_2$word, review_2$t
otal, max.words = 25,
          colors = c("gold3", "blue
4", "darkgoldenrod4"))
```

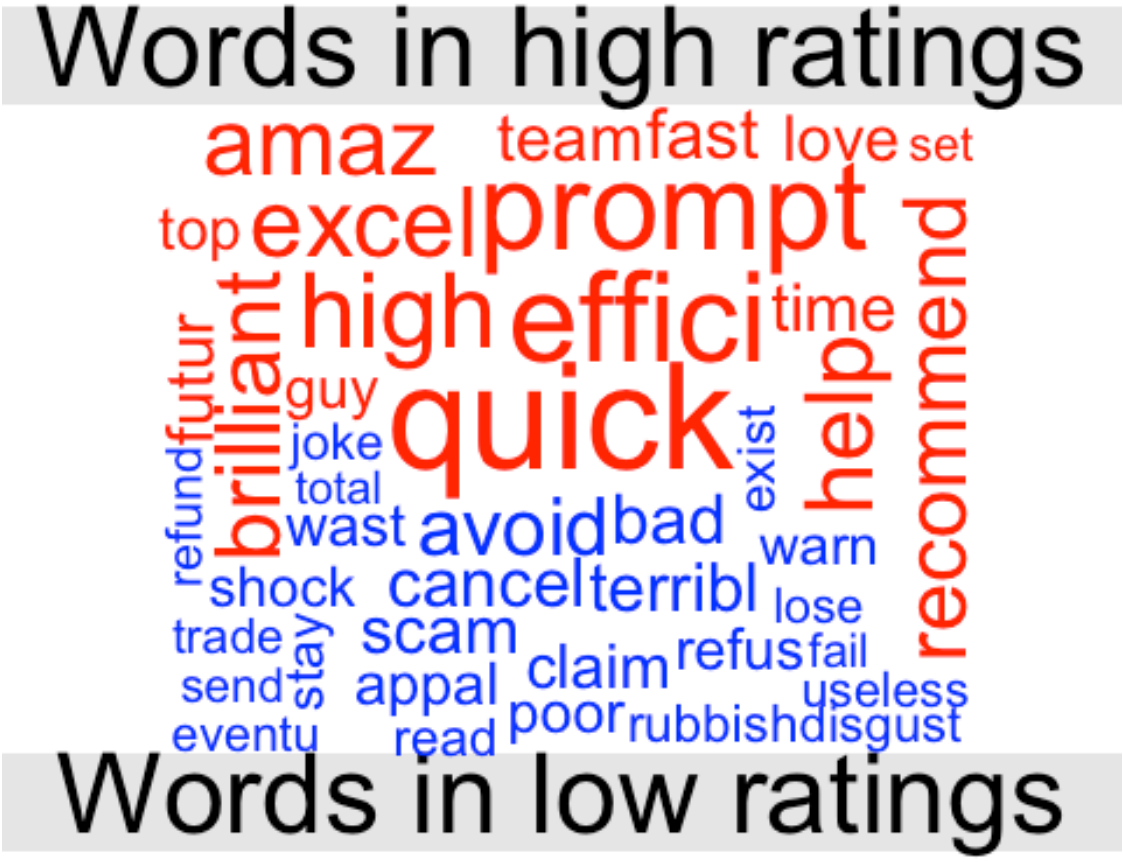
Word Cloud for 2 star reviews

```
   review_1 <- review_tokens_by_sto
res %>%
             anti_join(custom_stopwo
rds) %>%
             filter(overall == 1) %>
%
             group_by(word) %>%
             anti_join(custom_stopwo
rds) %>%
             summarise(total = mean(
total)) %>%
             arrange(desc(total))

wordcloud(review_1$word, review_1$t
otal, max.words = 25,
          colors = c("gold3", "blue
4", "darkgoldenrod4"))
```

Word Cloud for 1 star reviews

- Checking the wordcloud for dominant words across low and high ratings

```
   #Ratings of 4 and above are considered as high ratings
positive_reviews <- review_tokens_by_stores %>% anti_join(custom_stopwords) %
>%
                    filter (overall == 4| overall == 5)

positive_reviews <- positive_reviews %>% dplyr::select(word)

#Ratings of 2 and below are considered as low ratings
negative_reviews <- review_tokens_by_stores %>% anti_join(custom_stopwords) %
>%
                    filter (overall == 2| overall ==1 )

negative_reviews <- negative_reviews %>% dplyr::select(word)

combine_review <- c(positive_reviews,negative_reviews)

#Creating a corpus
corpus_combine = VCorpus(VectorSource(combine_review))

#Stemming the document
corpus_combine=tm_map(corpus_combine, stemDocument)
review_dtm_combine <- TermDocumentMatrix(corpus_combine)
all=as.matrix(review_dtm_combine)
colnames(all)=c("Words in high ratings","Words in low ratings")

#Creating comparison cloud
comparison.cloud(all,
 colors = c("red", "blue"),
 max.words = 100)
```



The comparison cloud shows the most prominent words in terms of the high and low ratings. Overall, the words seem to reflect the reasons for the respective ratings.

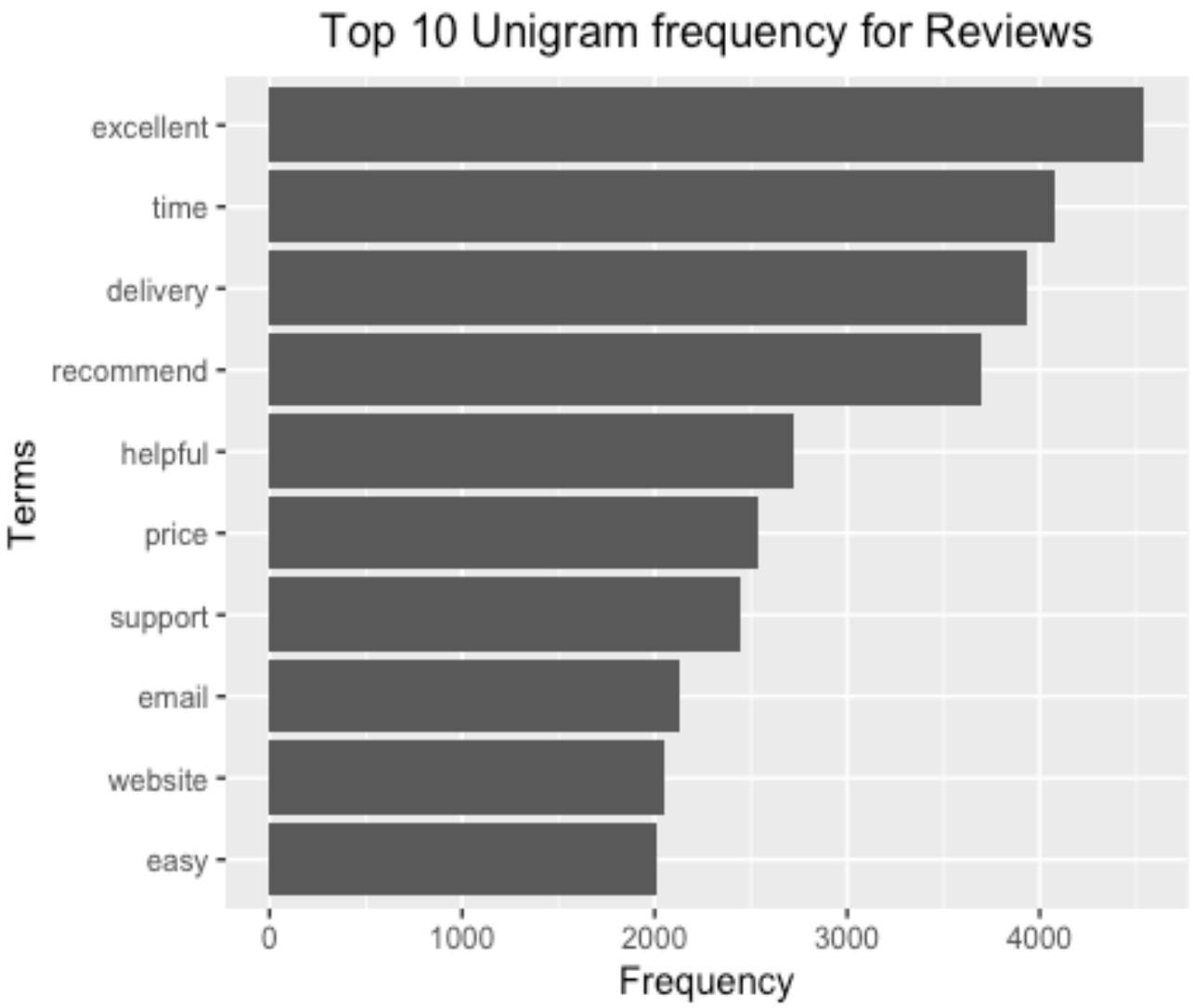
A.5: Word importance by n-grams

A.5.1: Reviews

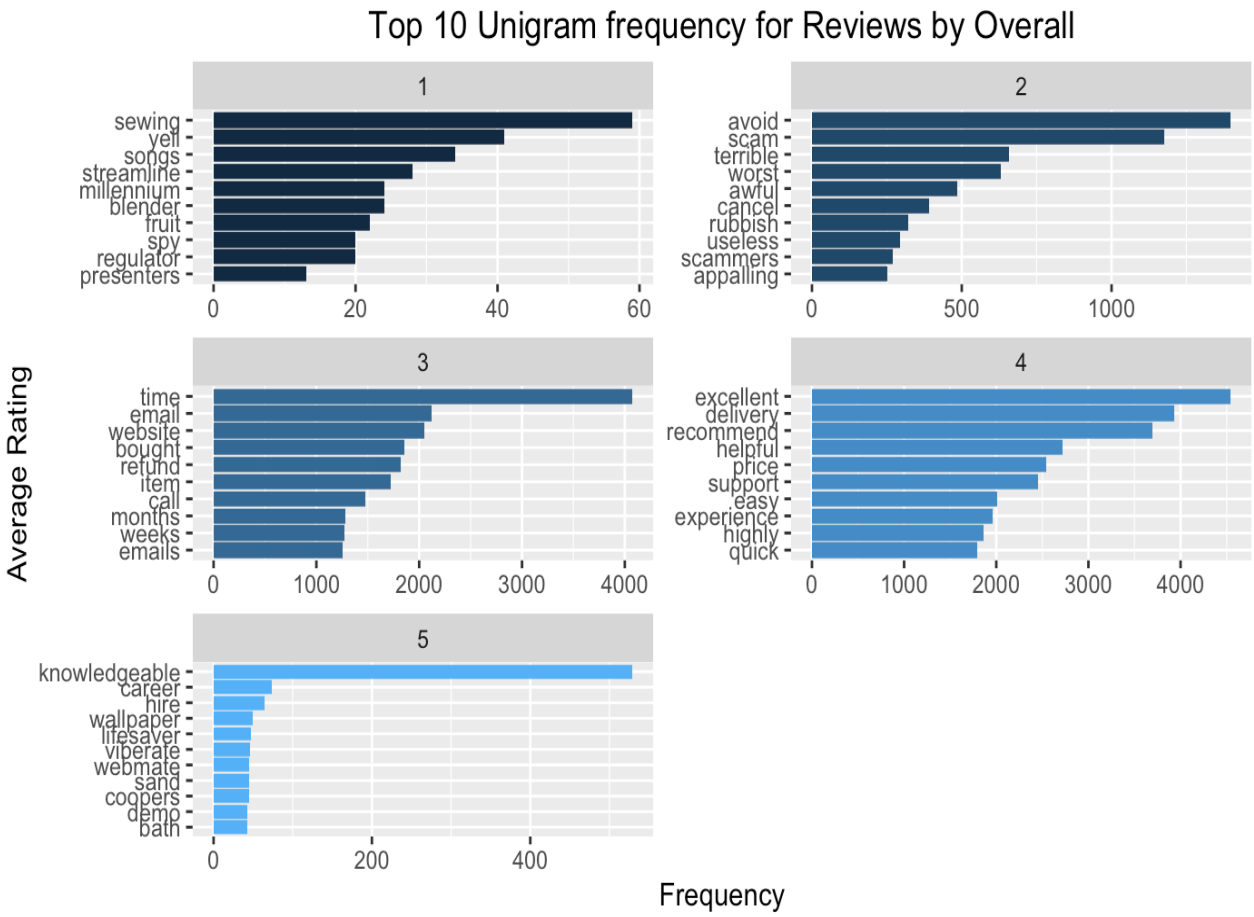
Now, using the bag of words approach, we will get the top n-grams for both reviews and store descriptions.

- Unigrams of reviews

```
   #Bringing the data for using in n-grams
df <- review_data_en %>% dplyr::select(company_id, store_description, overall
) %>% left_join(review_data_by_stores) %>% distinct(.)

#Tokenizing into unigrams
ngram_1_review <- df %>%
                  unnest_tokens(word, grouped_reviews, token = "ngrams", n =1
) %>%
                  anti_join(custom_stopwords)

#Finding the most frequent unigrams
ngram_1_review <- ngram_1_review %>%
                  group_by(word) %>%
                  summarise(Frequency = n(),
                  Overall = round(mean(overall))) %>%
                  arrange(desc(Frequency))

#Visualizing the top frequent unigrams
df %>% unnest_tokens(word, grouped_reviews, token = "ngrams", n =1) %>%
       anti_join(custom_stopwords) %>%
       count(word, sort = TRUE) %>%
       top_n(10) %>%
       ggplot(aes(reorder(word, n), n)) +
       geom_col() +
       labs(title="Top 10 Unigram frequency for Reviews",
       y = "Frequency",
       x = "Terms") +
       theme(plot.title = element_text(hjust = 0.5)) +
       coord_flip()
```


Top 10 Unigram frequency for Reviews
excellent
time
delivery
recommend
helpful
price
support
email
website
easy
Terms
0
1000
2000
3000
4000
Frequency

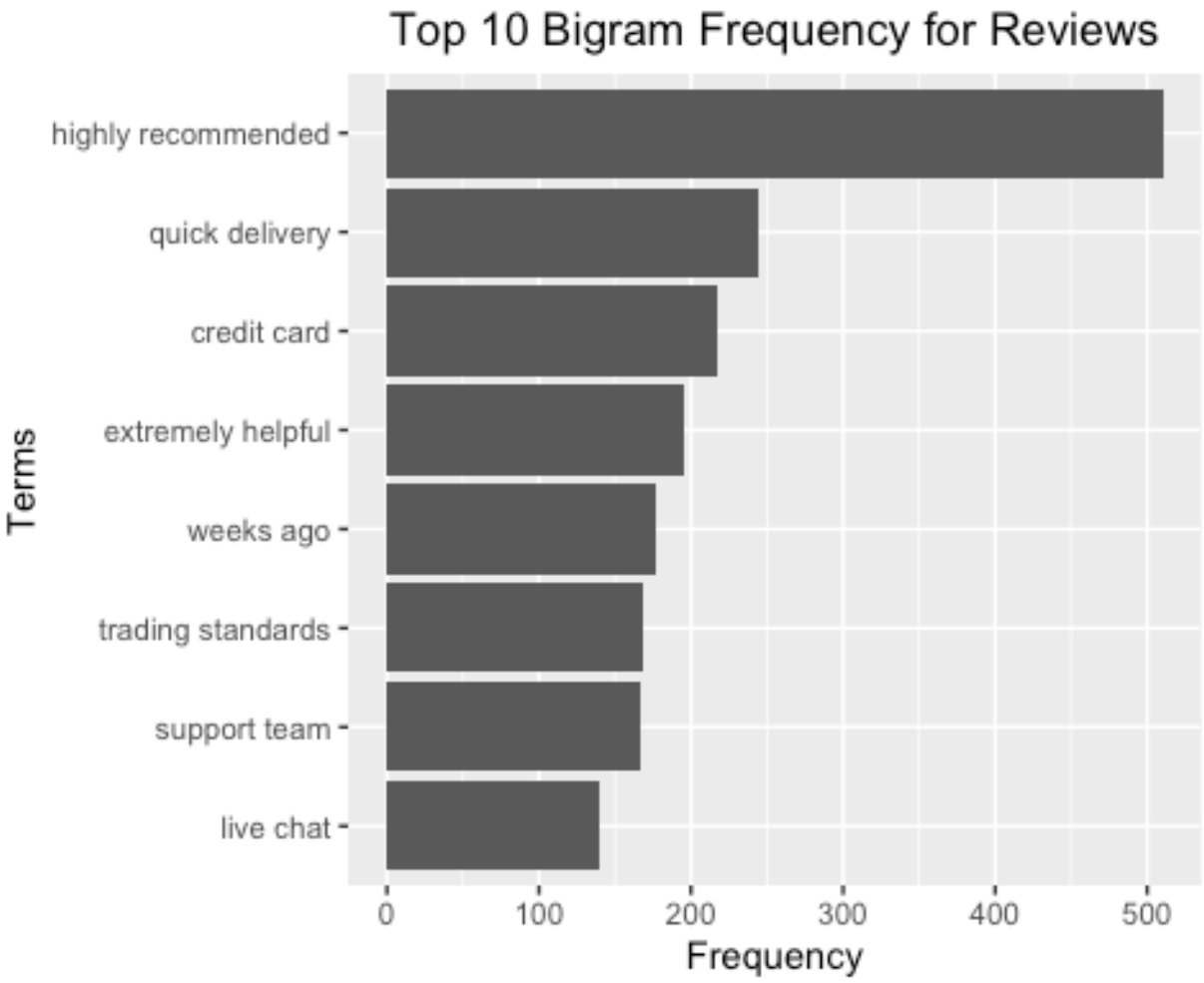
```r
   #Visualizing the top frequent unigrams by ratings
ngram_1_review %>% group_by(Overall) %>%
                   slice_max(Frequency, n = 10) %>%
                   ungroup() %>%
                   ggplot(aes(Frequency, fct_reorder(word, Frequency), fill =
 Overall)) +
                   geom_col(show.legend = FALSE) +
                   facet_wrap(~ Overall, ncol = 2, scales = "free") +
                   labs(title="Top 10 Unigram frequency for Reviews by Overal
l",
                   y = "Average Rating",
                   x = "Frequency") +
                   theme(plot.title = element_text(hjust = 0.5))
```



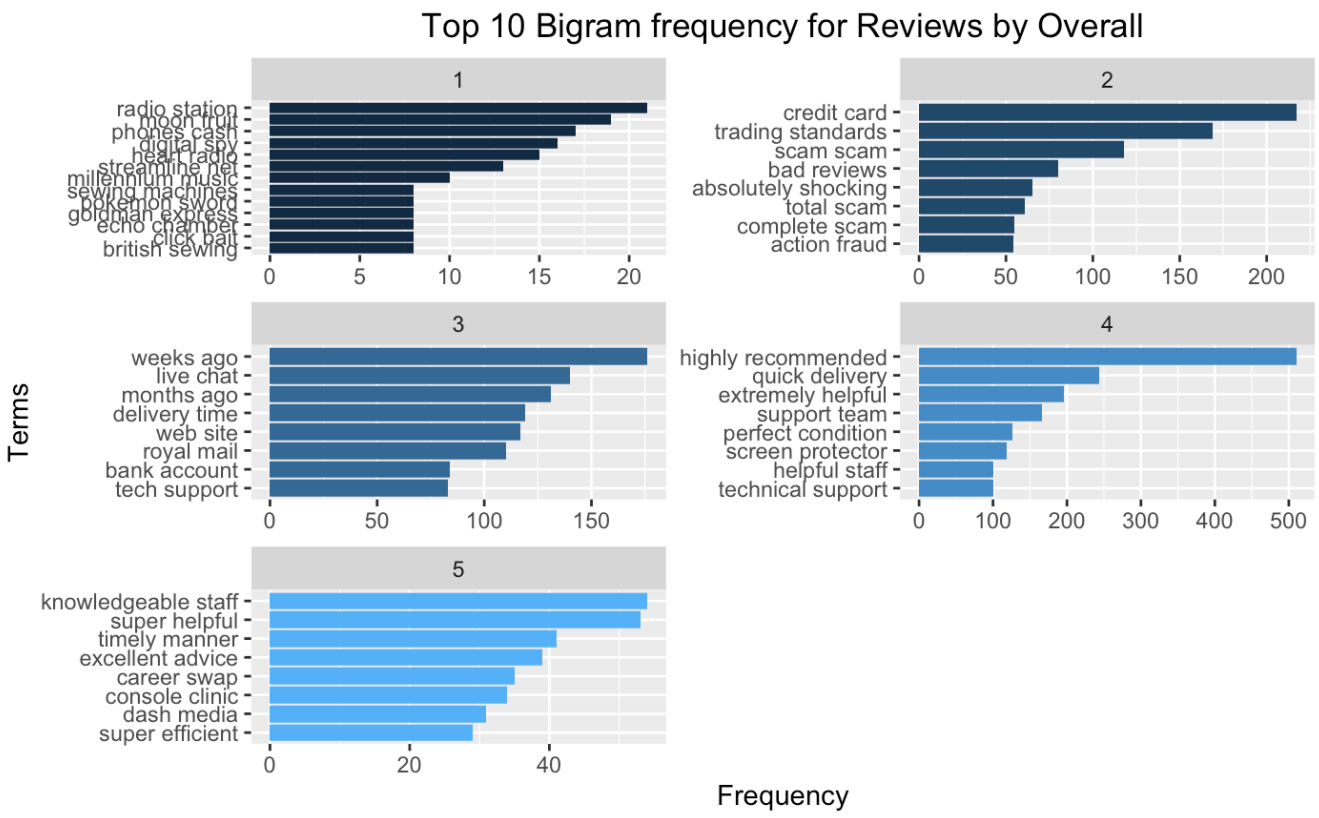
- Bigrams of reviews

```
   #Adding stop words specifically for bigrams
bi_custom_words <- c("class", "please", "job", "shop", "hand", "friend", "dat
e", "service", "review", "product", "customer", "day", "days", "received", "m
oney", "week", "buy", "company", "phone", "don", "ve", "ofcom", "ninja", "moo
nfruit", "cancelled", "virgin", "told", "people", "arrived", "paul", "av", "b
en", "josh", "andy", "tom", "richer", "plugin", "amp", "hifi", "aquiss", "sou
nds", "controller", "zoltan", "ian", "idnet", "glue", "proxy", "helen", "eno"
, "oled", "sypwai", "george", "peter", "panamoz", "pine", "dotsquares", "laur
a", "vape", "amiga", "biz", "technoworld", "proxies", "recommend", "fast", "m
achine", "avoid", "fridge", "freezer", "sb", "trust", "pilot", "pc")

tibble(word = bi_custom_words,lexicon = rep("custom",length(bi_custom_words))
) %>% bind_rows(stop_words) -> bi_custom_stopwords

#Bringing the data
ngram_2_review <- df %>%
                  unnest_tokens(word, grouped_reviews, token = "ngrams", n =2
)

#Tokenizing for bigrams
ngram_2_review$index <- seq.int(nrow(ngram_2_review))

df2 <- ngram_2_review %>%
       separate(word, c("word1", "word2"), sep = " ") %>%
       dplyr::select('index', 'word1','word2')

ngram_2_review <- left_join(ngram_2_review, df2, by = "index")
rm(df2)

ngram_2_review <- ngram_2_review %>%
                  filter(!word1 %in% bi_custom_stopwords$word) %>%
                  filter(!word2 %in% bi_custom_stopwords$word)

ngram_2_review_splitted <- ngram_2_review %>%
                  count(word1, word2, sort = TRUE)

#Finding the frequencies of the bigrams
ngram_2_review_combined <- ngram_2_review %>%
                  group_by(word) %>%
                  summarise(Frequency = n(),
                            Overall = round(mean(overall))) %>%
                  arrange(desc(Frequency)) %>%
                  na.omit()

#Visualizing the top frequent bigrams
ngram_2_review_combined %>% slice_max(Frequency, n = 8) %>%
                   ggplot(aes(reorder(word, Frequency), Frequency)) +
                   geom_col() +
```

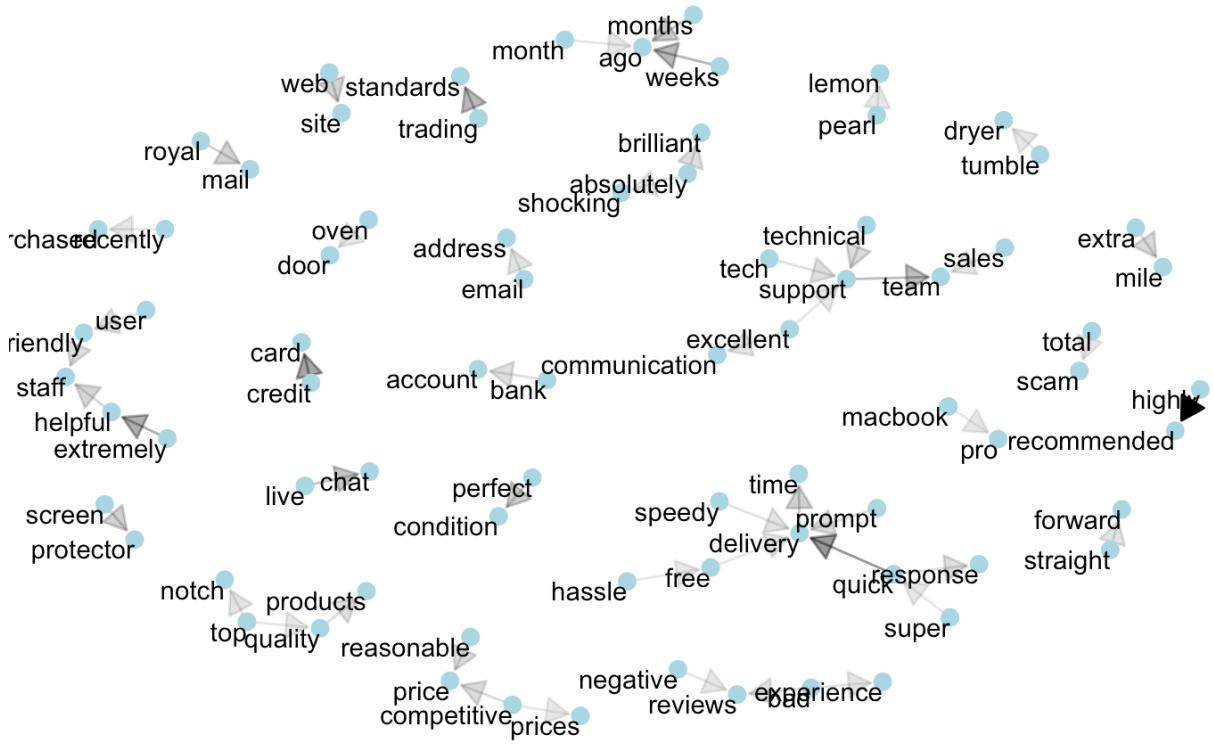
```
                labs(title="Top 10 Bigram Frequency for Reviews",
                y = "Frequency",
                x = "Terms") +
                theme(plot.title = element_text(hjust = 0.5)) +
                coord_flip()
```



```r
   #Visualizing the top frequent bigrams by ratings
ngram_2_review_combined %>% group_by(Overall) %>%
                    slice_max(Frequency, n = 8) %>%
                    ungroup() %>%
                    ggplot(aes(Frequency, fct_reorder(word, Frequency), fill =
 Overall)) +
                    geom_col(show.legend = FALSE) +
                    facet_wrap(~ Overall, ncol = 2, scales = "free") +
                    labs(title="Top 10 Bigram frequency for Reviews by Overall
",
                    y = "Terms",
                    x = "Frequency") +
                    theme(plot.title = element_text(hjust = 0.5))
```



```
   #Common bigrams in reviews
ngram_2_graph <- ngram_2_review_splitted %>%
                 filter(n > 60) %>%
                 graph_from_data_frame()


set.seed(321)
a <- grid::arrow(type = "closed", length = unit(.15, "inches"))

ggraph(ngram_2_graph, layout = "fr") +
       geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
                      arrow = a, end_cap = circle(.07, 'inches')) +
       geom_node_point(color = "lightblue", size = 3) +
       geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
       theme_void()
```

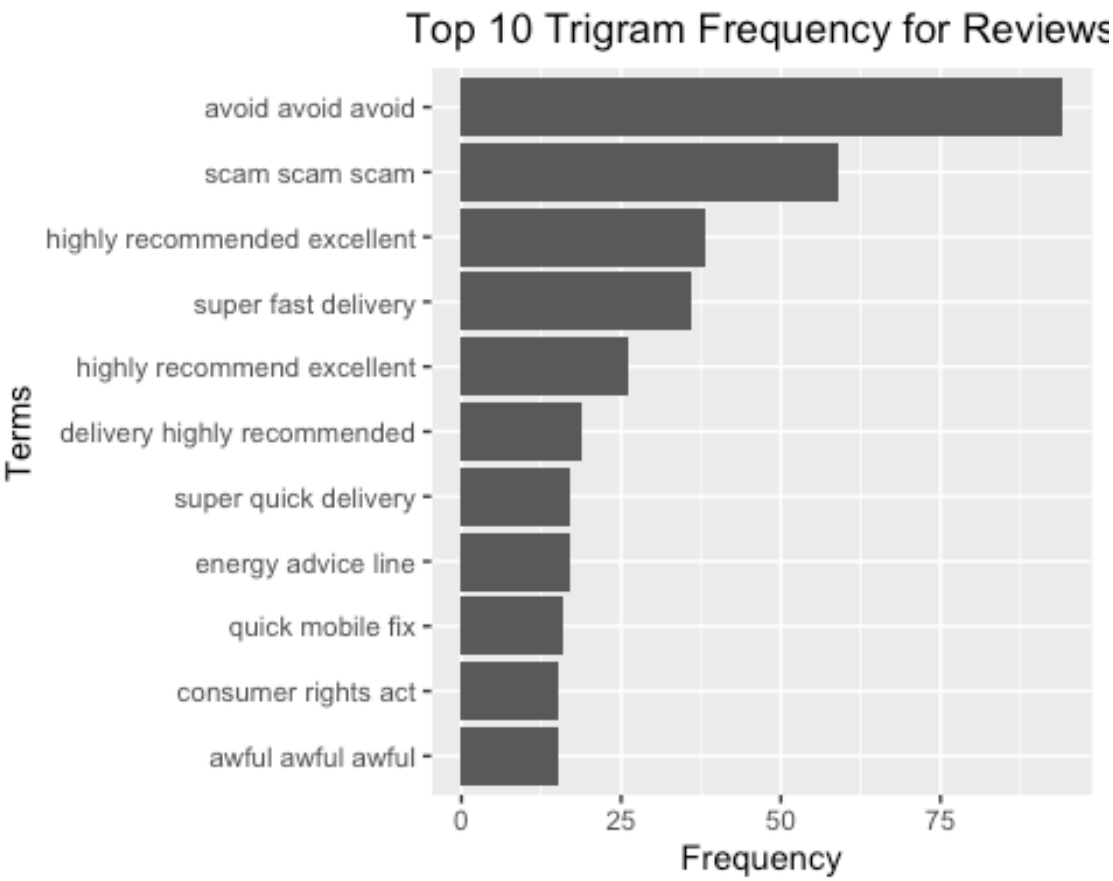


The word associations show a connection between the common words. It is giving a sense of which word pairs can be expected in the reviews and the direction of the pairs to show the dominant word in the pairs.

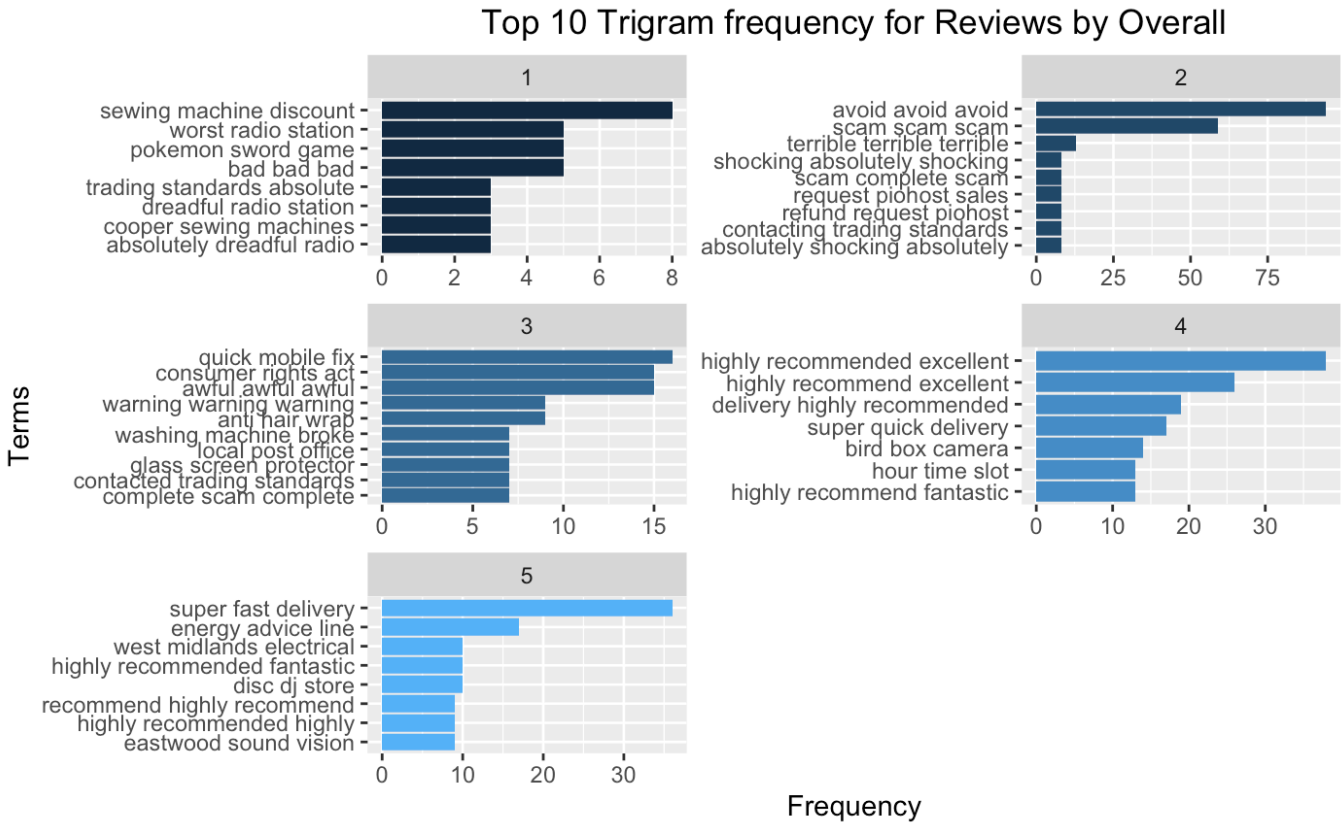
- Trigrams of reviews

```
   #Adding stop words specifically for bigrams
tri_custom_words <- c("class", "please", "job", "service", "shop", "hand", "f
riend", "date", "review", "product", "customer", "day", "days", "money", "wee
k", "buy", "company", "phone", "don", "ve", "ofcom", "ninja", "moonfruit", "v
irgin", "told", "people", "arrived", "paul", "av", "ben", "josh", "andy", "to
m", "richer", "plugin", "amp", "hifi", "aquiss", "sounds", "zoltan", "ian", "
idnet", "glue", "proxy", "helen", "eno", "oled", "sypwai", "george", "peter",
 "panamoz", "pine", "dotsquares", "laura", "vape", "amiga", "biz", "technowor
ld", "fridge", "freezer", "iphone", "pro", "trust", "pilot", "st", "uber", "r
eviews", "boots", "sh", "london", "british", "xl", "ms", "avern", "pm", "june
", "bullguard")

tibble(word = tri_custom_words,lexicon = rep("custom",length(tri_custom_words
))) %>% bind_rows(stop_words) -> tri_custom_stopwords

#Bringing the data
ngram_3_review <- df %>%
                  unnest_tokens(word, grouped_reviews, token = "ngrams", n =3
)

#Tokenizing into trigrams
ngram_3_review$index <- seq.int(nrow(ngram_3_review))

df3 <- ngram_3_review %>%
       separate(word, c("word1", "word2", "word3"), sep = " ") %>%
       dplyr::select('index', 'word1','word2', 'word3')

ngram_3_review <- left_join(ngram_3_review, df3, by = "index")
rm(df3)

ngram_3_review <- ngram_3_review %>%
                  filter(!word1 %in% tri_custom_stopwords$word) %>%
                  filter(!word2 %in% tri_custom_stopwords$word) %>%
                  filter(!word3 %in% tri_custom_stopwords$word)

ngram_3_review_splitted <- ngram_3_review %>%
                  count(word1, word2, word3, sort = TRUE)

#Finding the frequencies
ngram_3_review <- ngram_3_review %>%
                  group_by(word) %>%
                  summarise(Frequency = n(),
                            Overall = round(mean(overall))) %>%
                  arrange(desc(Frequency)) %>%
                  na.omit()

#Visualizing the top frequent trigrams
ngram_3_review %>% slice_max(Frequency, n = 10) %>%
                   ggplot(aes(reorder(word, Frequency), Frequency)) +
```

```
                    geom_col() +
                    labs(title="Top 10 Trigram Frequency for Reviews",
                    y = "Frequency",
                    x = "Terms") +
                    theme(plot.title = element_text(hjust = 0.5)) +
                    coord_flip()
```

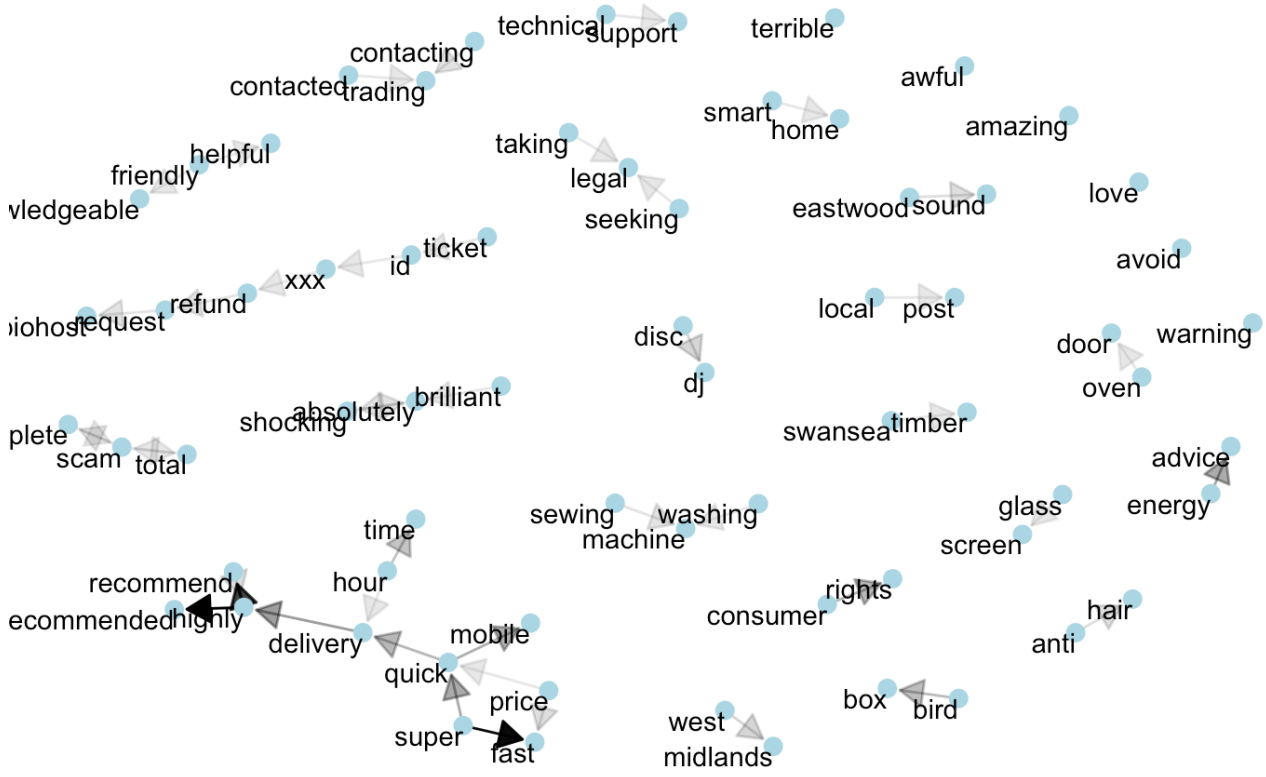


```r
   #Visualizing the top frequent trigrams by ratings
ngram_3_review %>% group_by(Overall) %>%
                  slice_max(Frequency, n = 7) %>%
                  ungroup() %>%
                  ggplot(aes(Frequency, fct_reorder(word, Frequency), fill = O
verall)) +
                  geom_col(show.legend = FALSE) +
                  facet_wrap(~ Overall, ncol = 2, scales = "free") +
                  labs(title="Top 10 Trigram frequency for Reviews by Overall"
,
                  y = "Terms",
                  x = "Frequency") +
                  theme(plot.title = element_text(hjust = 0.5))
```



The trigrams seem moderately meaningful. While the context is at times clear, some of the trigrams seem to be a random mix of unrelated words.

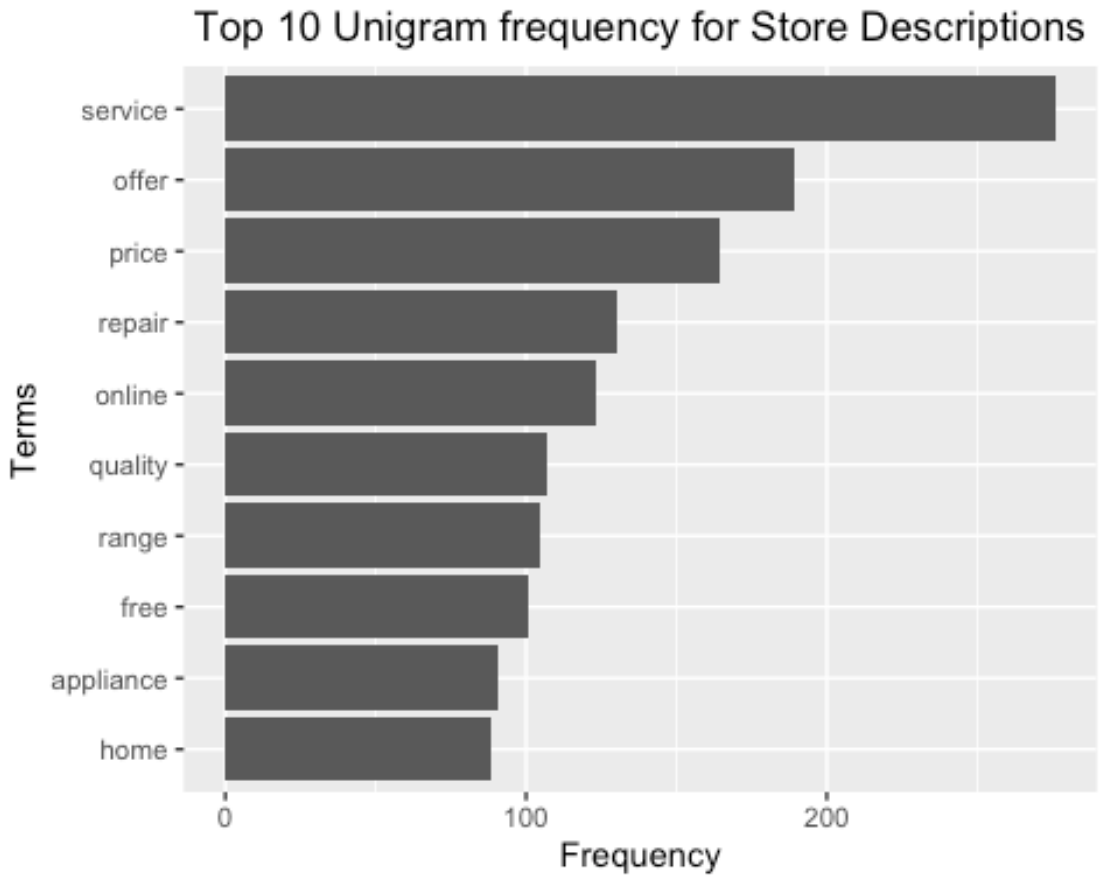
```r
   #Common trigrams in reviews
ngram_3_graph <- ngram_3_review_splitted %>%
                 filter(n > 6) %>%
                 graph_from_data_frame()

set.seed(321)
a <- grid::arrow(type = "closed", length = unit(.15, "inches"))

ggraph(ngram_3_graph, layout = "fr") +
       geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
                          arrow = a, end_cap = circle(.07, 'inches')) +
       geom_node_point(color = "lightblue", size = 3) +
       geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
       theme_void()
```



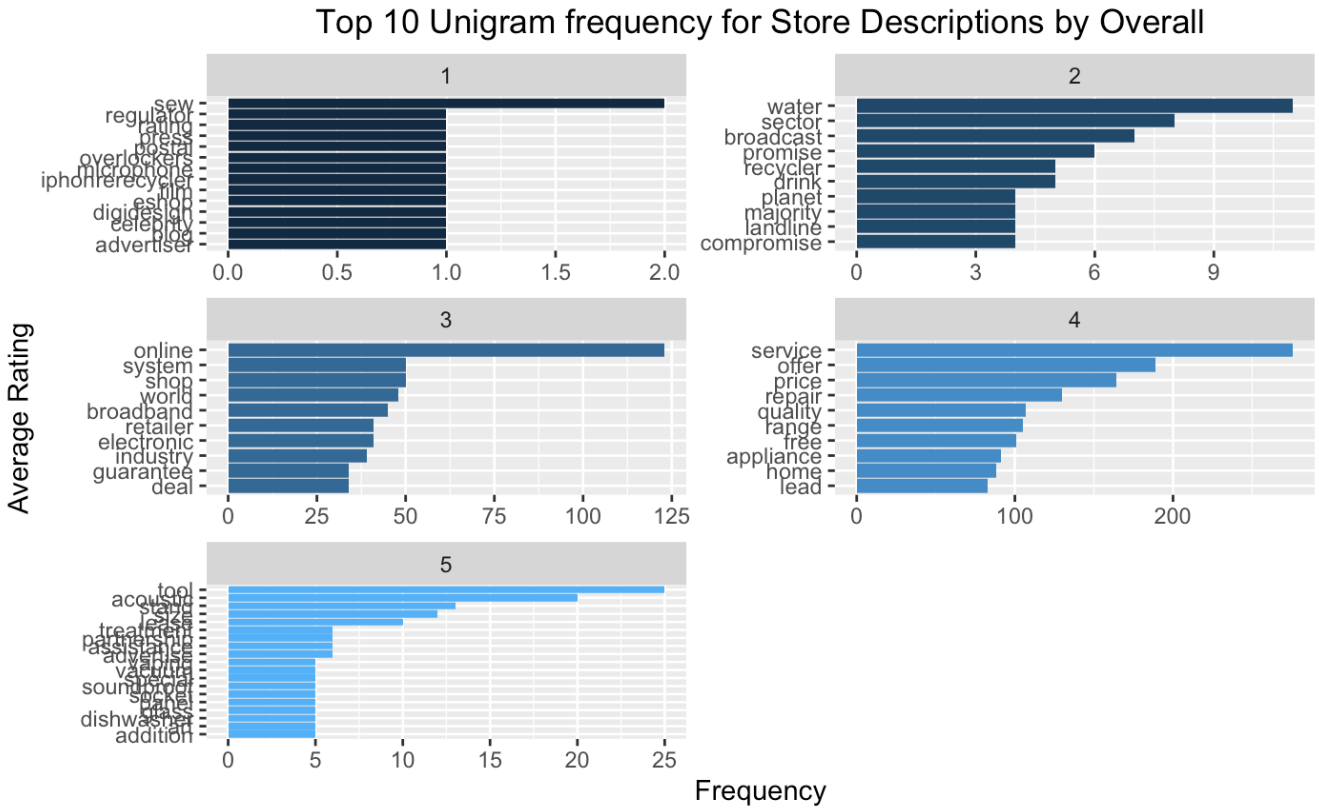
A.5.2: Store Descriptions

- Unigrams of store description

```
   #Adding special stop words
custom_words_desc <- c("uk", "company", "customer", "business", "na", "mobile
", "provide", "product", "facebook", "sony", "ireland", "tv", "computer", "ph
one", "oven", "laskys", "moonfruit", "cooper", "goldmanlife", "airwave", "ime
ndmacs", "north", "battery", "lcd", "tnp", "mo", "tablet", "sellmyiphone", "i
phonererecycler", "iphonerecycler", "st", "sia", "toshiba", "sonos", "nokia",
 "plusnet", "plasma", "virgin", "adapter")

tibble(word = custom_words_desc,lexicon = rep("custom",length(custom_words_de
sc))) %>% bind_rows(stop_words) -> custom_stopwords_desc

#Bringing the data for using in n-grams
df <- review_data_en %>% dplyr::select(company_id, store_description, overall
) %>% left_join(review_data_by_stores) %>% distinct(.)

#Tokenizing into unigrams
ngram_1_desc <- df %>%
                unnest_tokens(word, store_description, token = "ngrams", n =1
) %>%
                anti_join(custom_stopwords_desc)

#Finding the most frequent unigrams
ngram_1_desc <- ngram_1_desc %>%
 group_by(word) %>%
 summarise(Frequency = n(),
           Overall = round(mean(overall))) %>%
 arrange(desc(Frequency))

#Visualizing the top frequent unigrams
df %>%
 unnest_tokens(word, store_description, token = "ngrams", n =1) %>%
 anti_join(custom_stopwords_desc) %>%
 count(word, sort = TRUE) %>%
 top_n(10) %>%
 ggplot(aes(reorder(word, n), n)) +
 geom_col() +
 labs(title="Top 10 Unigram frequency for Store Descriptions",
 y = "Frequency",
 x = "Terms") +
 theme(plot.title = element_text(hjust = 0.5)) +
 coord_flip()
```

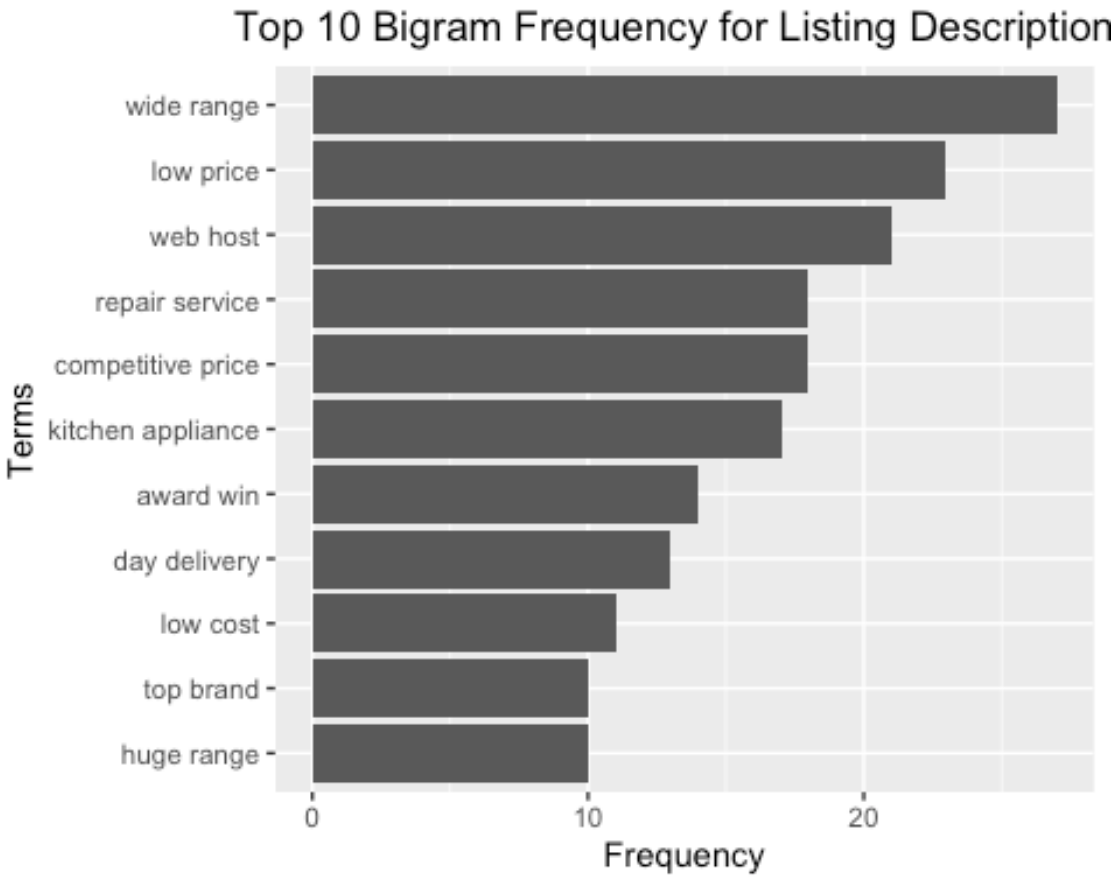

Top 10 Unigram frequency for Store Descriptions
service
offer
price
repair
online
quality
range
free
appliance
home
Terms
0
100
200
Frequency

```r
  #Visualizing the top frequent unigrams by ratings
ngram_1_desc %>%
 group_by(Overall) %>%
 slice_max(Frequency, n = 10) %>%
 ungroup() %>%
 ggplot(aes(Frequency, fct_reorder(word, Frequency), fill = Overall)) +
 geom_col(show.legend = FALSE) +
 facet_wrap(~ Overall, ncol = 2, scales = "free") +
 labs(title="Top 10 Unigram frequency for Store Descriptions by Overall",
 y = "Average Rating",
 x = "Frequency") +
 theme(plot.title = element_text(hjust = 0.5))
```

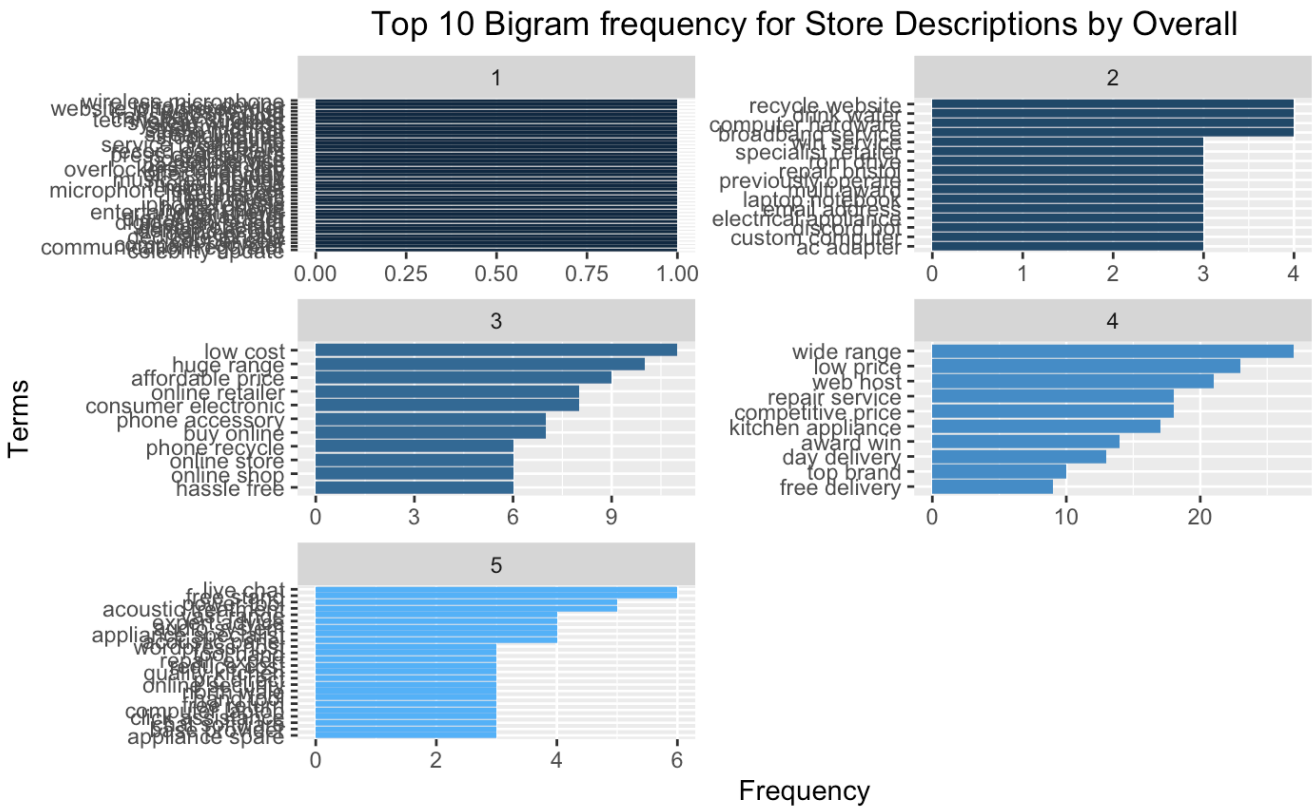


- Bigrams of store description

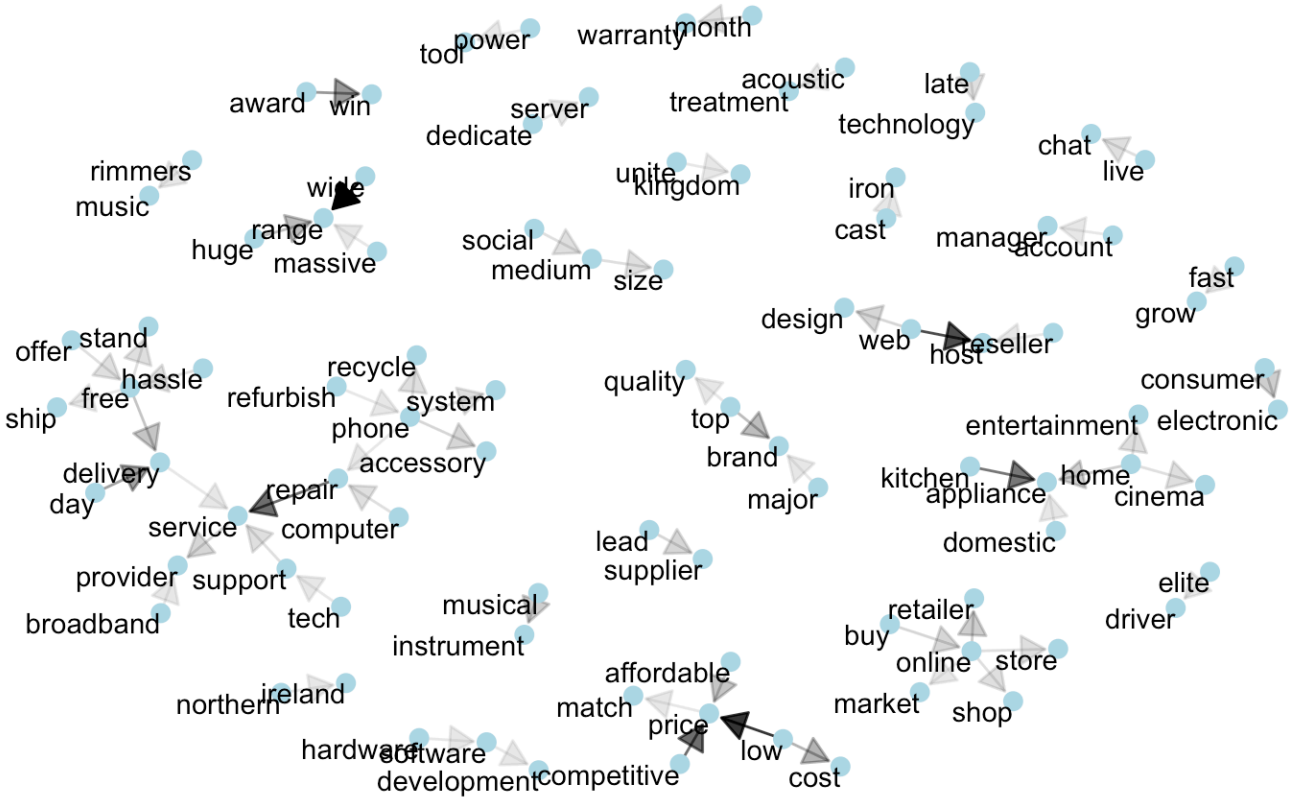
```
#Adding special stop words
custom_words_desc <- c("uk", "company", "customer", "business", "na", "mobile
", "provide", "product", "machine", "game", "console", "camera", "virgin", "s
sl", "pc", "oven", "fridge", "freezer", "battery", "mo", "lcd", "tv", "cable"
, "tablet", "cable", "charger", "samsung", "sony")

tibble(word = custom_words_desc,lexicon = rep("custom",length(custom_words_de
sc))) %>% bind_rows(stop_words) -> custom_stopwords_desc

#Bringing the data
ngram_2_desc <- df %>%
                unnest_tokens(word, store_description, token = "ngrams", n =2
)

#Tokenizing for bigrams
ngram_2_desc$index <- seq.int(nrow(ngram_2_desc))

df2 <- ngram_2_desc %>%
           separate(word, c("word1", "word2"), sep = " ") %>%
           dplyr::select('index', 'word1','word2')

ngram_2_desc <- left_join(ngram_2_desc, df2, by = "index")

ngram_2_desc <- ngram_2_desc %>%
                filter(!word1 %in% custom_stopwords_desc$word) %>%
                filter(!word2 %in% custom_stopwords_desc$word)

ngram_2_desc_splitted <- ngram_2_desc %>%
                  count(word1, word2, sort = TRUE) %>% na.omit()

#Finding the frequencies of the bigrams
ngram_2_desc_combined <- ngram_2_desc %>%
                group_by(word) %>%
                summarise(Frequency = n(),
                          Overall = round(mean(overall))) %>%
                arrange(desc(Frequency)) %>%
                na.omit()

#Visualizing the top frequent bigrams
ngram_2_desc_combined %>% slice_max(Frequency, n = 10) %>%
                 ggplot(aes(reorder(word, Frequency), Frequency)) +
                 geom_col() +
                 labs(title="Top 10 Bigram Frequency for Listing Descriptions
",
                 y = "Frequency",
                 x = "Terms") +
                 theme(plot.title = element_text(hjust = 0.5)) +
                 coord_flip()
```


Top 10 Bigram Frequency for Listing Description
wide range
low price
web host
repair service
competitive price
kitchen appliance
award win
day delivery
low cost
top brand
huge range
Terms
0
10
20
Frequency

```
   #VIsualizing the top frequent bigrams by ratings
ngram_2_desc_combined %>% group_by(Overall) %>%
                  slice_max(Frequency, n = 10) %>%
                  ungroup() %>%
                  ggplot(aes(Frequency, fct_reorder(word, Frequency), fill = O
verall)) +
                  geom_col(show.legend = FALSE) +
                  facet_wrap(~ Overall, ncol = 2, scales = "free") +
                  labs(title="Top 10 Bigram frequency for Store Descriptions b
y Overall",
                  y = "Terms",
                  x = "Frequency") +
                  theme(plot.title = element_text(hjust = 0.5))
```



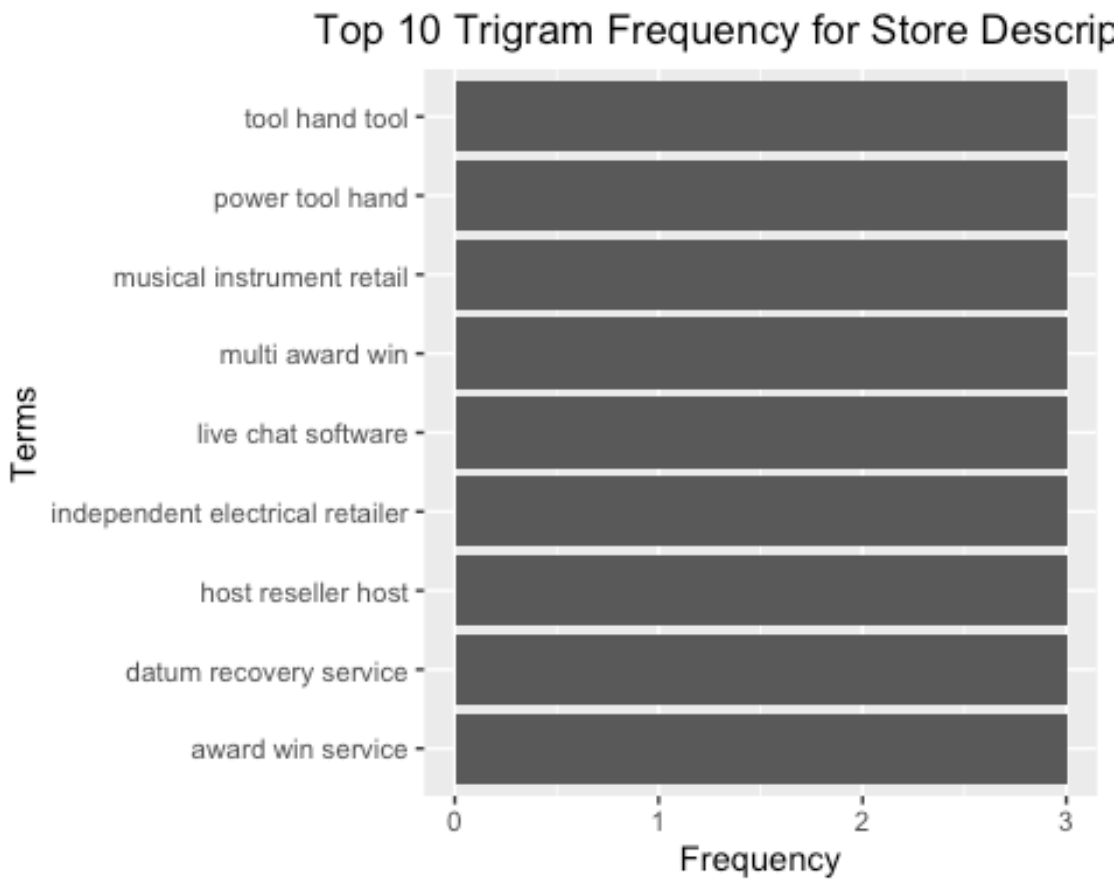
```
#Common bigrams in reviews
ngram_2_graph <- ngram_2_desc_splitted %>%
                 filter(n > 4) %>%
                 graph_from_data_frame()

set.seed(2017)
a <- grid::arrow(type = "closed", length = unit(.15, "inches"))

ggraph(ngram_2_graph, layout = "fr") +
       geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
                           arrow = a, end_cap = circle(.07, 'inches')) +
       geom_node_point(color = "lightblue", size = 3) +
       geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
       theme_void()
```



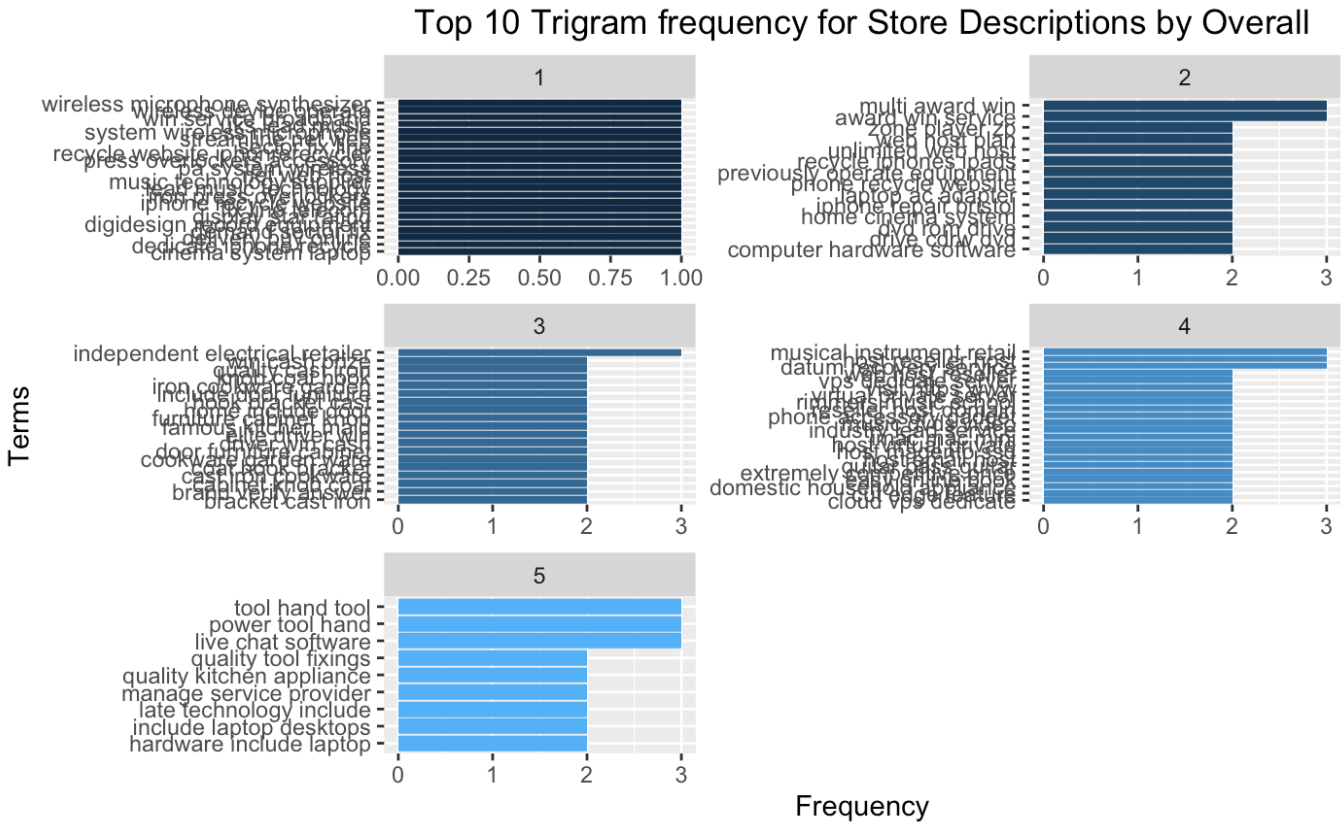
* Trigrams of store description

```
#Adding special stop words
custom_words_desc3 <- c("uk", "company", "customer", "business", "na", "mobil
e", "provide", "product", "machine", "game", "console", "camera", "virgin", "
ssl", "pc", "oven", "fridge", "freezer", "battery", "mo", "lcd", "tv", "cable
", "tablet", "cable", "charger", "samsung", "sony")

tibble(word = custom_words_desc,lexicon = rep("custom",length(custom_words_de
sc))) %>% bind_rows(stop_words) -> custom_stopwords_desc3

#Bringing the data
ngram_3_desc <- df %>%
                unnest_tokens(word, store_description, token = "ngrams", n =3
)

#Tokenizing into trigrams
ngram_3_desc$index <- seq.int(nrow(ngram_3_desc))

df3 <- ngram_3_desc %>%
       separate(word, c("word1", "word2", "word3"), sep = " ") %>%
       dplyr::select('index', 'word1','word2', 'word3')

ngram_3_desc <- left_join(ngram_3_desc, df3, by = "index")
rm(df3)

ngram_3_desc <- ngram_3_desc %>%
                filter(!word1 %in% custom_stopwords_desc3$word) %>%
                filter(!word2 %in% custom_stopwords_desc3$word) %>%
                filter(!word3 %in% custom_stopwords_desc3$word)

ngram_3_desc_splitted <- ngram_3_desc %>%
                  count(word1, word2, word3, sort = TRUE) %>% na.omit()

#Finding the frequencies
ngram_3_desc_combined <- ngram_3_desc %>%
                group_by(word) %>%
                summarise(Frequency = n(),
                          Overall = round(mean(overall))) %>%
                arrange(desc(Frequency)) %>%
                na.omit()

#Visualizing the top frequent trigrams
ngram_3_desc_combined %>% slice_max(Frequency, n = 5) %>%
                 ggplot(aes(reorder(word, Frequency), Frequency)) +
                 geom_col() +
                 labs(title="Top 10 Trigram Frequency for Store Descriptions"
,
                 y = "Frequency",
                 x = "Terms") +
```

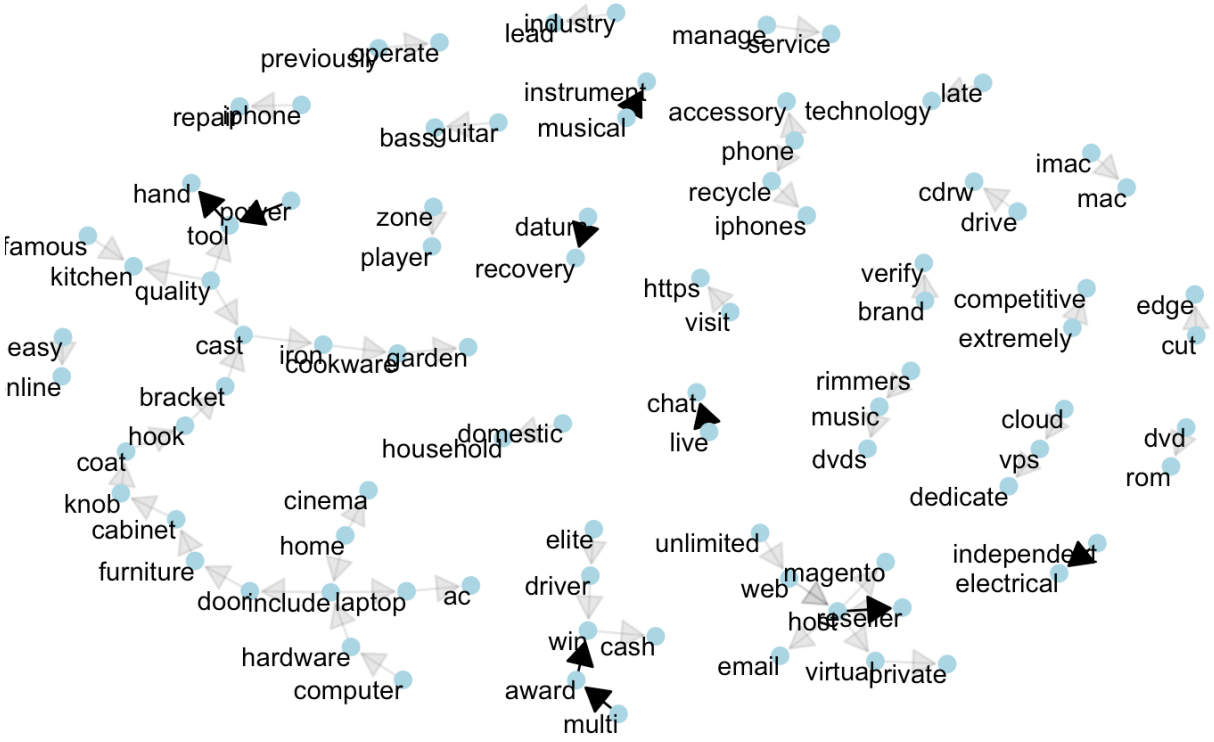
```
                theme(plot.title = element_text(hjust = 0.5)) +
                coord_flip()
```



```
#Visualizing the top frequent trigrams by ratings
ngram_3_desc_combined %>% group_by(Overall) %>%
                  slice_max(Frequency, n = 5) %>%
                  ungroup() %>%
                  ggplot(aes(Frequency, fct_reorder(word, Frequency), fill = O
verall)) +
                  geom_col(show.legend = FALSE) +
                  facet_wrap(~ Overall, ncol = 2, scales = "free") +
                  labs(title="Top 10 Trigram frequency for Store Descriptions
by Overall",
                  y = "Terms",
                  x = "Frequency") +
                  theme(plot.title = element_text(hjust = 0.5))
```



```
   ngram_3_graph <- ngram_3_desc_splitted %>%
                    filter(n > 1) %>%
                    graph_from_data_frame()

set.seed(2017)
a <- grid::arrow(type = "closed", length = unit(.15, "inches"))

ggraph(ngram_3_graph, layout = "fr") +
       geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
                          arrow = a, end_cap = circle(.07, 'inches')) +
       geom_node_point(color = "lightblue", size = 3) +
       geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
       theme_void()
```



A.6: Word Associations

Now that we have looked at the n-grams, we will try to examine some important words and look at the word associations to get a sense of what reviewers are saying.

```
   #Casting a Document Term Matrix
dtm_1_review <- df %>%
                unnest_tokens(word, grouped_reviews, token = "ngrams", n =1)
%>%
                anti_join(custom_stopwords) %>%
                count(company_id, word) %>%
                cast_dtm(company_id, word, n)

inspect(dtm_1_review)
   ## <<DocumentTermMatrix (documents: 695, terms: 21720)>>
## Non-/sparse entries: 231135/14864265
## Sparsity           : 98%
## Maximal term length: 45
## Weighting          : term frequency (tf)
## Sample             :
##      Terms
## Docs  delivery easy email excellent helpful price recommend support time
##   3          2    2    10         2       0     8        10       0   58
##   36         0    1     3         1       4     0         0       1   28
##   361       20    0    14        20      24    14        20      18   16
##   44         0    1     5         1       6     1         4      12    8
##   49         0    3     4         0       0     0         0       5   11
##   51         2    2     2         1       5     3         9       4   11
##   639       65    3    14         4       4     9         2       1   21
##   67         4    1     4         4       3     7         4       5   22
##   92        20    0    10         4       2     2         3      10   11
##   93         5    1     0         5       3     3         6       6    8
##      Terms
## Docs  website
##   3         4
##   36        0
##   361      12
##   44        4
##   49        4
##   51        3
##   639       3
##   67        5
##   92        8
##   93        5
   #Checking which words correlate with "recommend"
findAssocs(dtm_1_review,"recommended",corlimit = 0.7)
   ## $recommended
## highly
##    0.7
   findAssocs(dtm_1_review,"excellent",corlimit = 0.6)
```

```
   ## $excellent
## recommend
##      0.63
   findAssocs(dtm_1_review,"easy",corlimit = 0.4)
   ## $easy
##  simple process
##    0.48    0.47
   findAssocs(dtm_1_review,"price",corlimit = 0.5)
   ## $price
## offered  quoted
##    0.53    0.51
   findAssocs(dtm_1_review,"time",corlimit = 0.4)
   ## $time
##       hours       times         bad        wait        left   customers
##        0.55        0.55        0.52        0.52        0.51        0.49
##        poor       speak       spent       worst        call      cancel
##        0.49        0.49        0.49        0.49        0.48        0.47
##       worse  absolutely      called  department       leave     contact
##        0.47        0.46        0.46        0.46        0.46        0.45
##         due        hold        sort      person       awful   complaint
##        0.45        0.45        0.45        0.45        0.44        0.44
##         cut    terrible     useless       weeks      option      happen
##        0.44        0.44        0.44        0.44        0.44        0.43
##       month     waiting  complaints  cancelling  experience      issues
##        0.43        0.43        0.43        0.42        0.42        0.42
##      reason      taking    complain        ring      expect      change
##        0.42        0.42        0.42        0.42        0.42        0.41
## frustrating        hour      months     package     provide    services
##        0.41        0.41        0.41        0.41        0.41        0.41
##       stars    shocking  understand
##        0.41        0.41        0.40
   findAssocs(dtm_1_review,"helpful",corlimit = 0.6)
   ## $helpful
##      friendly knowledgeable
##          0.64          0.61
   findAssocs(dtm_1_review,"support",corlimit = 0.5)
   ## $support
## technical
##      0.51
   # To check and confirm what word combinations have recommended to get an i
dea about things which are recommended in the reviews by the user
ngram_3_review %>% group_by(word) %>%
                   count(sort = TRUE) %>%
                   filter(stringr::str_detect(word,"recommend | recommended")
)
   ## # A tibble: 1,017 × 2
## # Groups:   word [1,017]
##    word                                n
##    <chr>                           <int>
##  1 absolutely recommend azooma         1
```

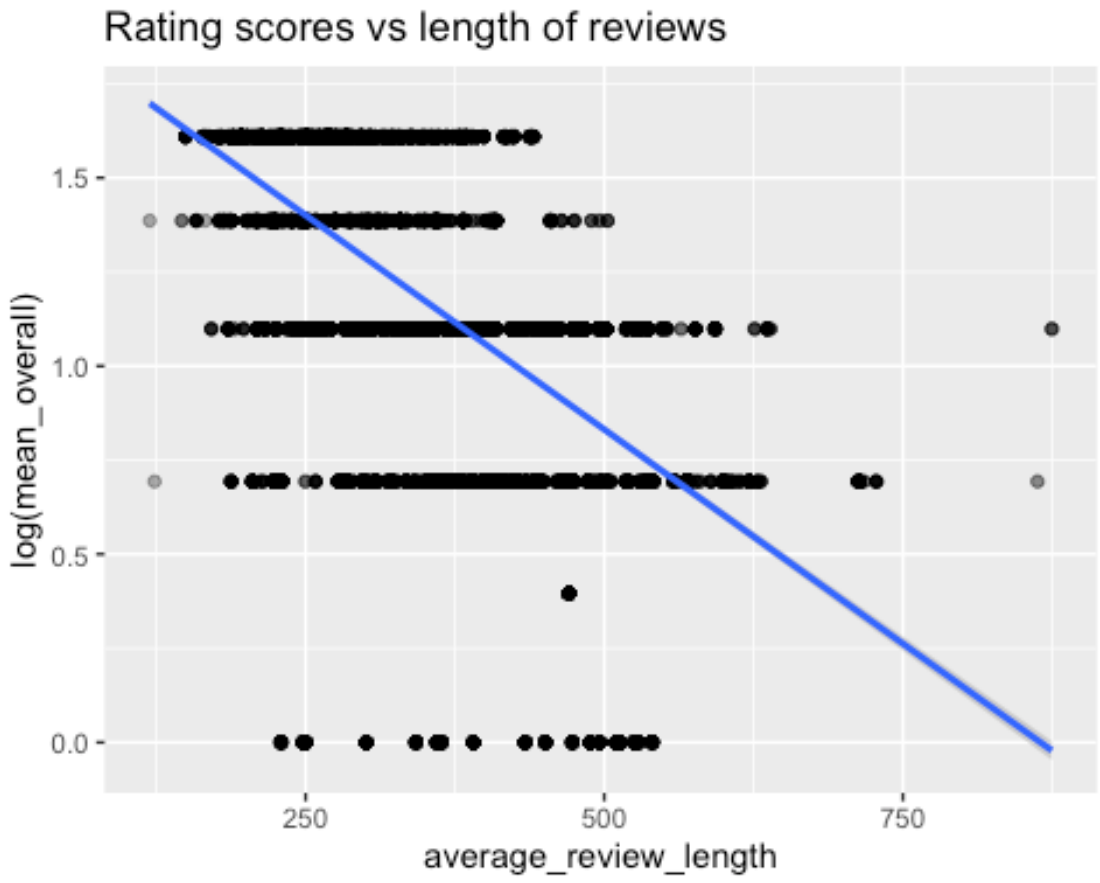
```
##  2 absolutely recommend building        1
##  3 absolutely recommend dd              1
##  4 absolutely recommend ismash          1
##  5 absolutely recommend linnworks       1
##  6 absolutely recommend selling         1
##  7 absolutely recommend ttnc            1
##  8 absolutely recommended excellent     1
##  9 account highly recommended           1
## 10 advertised seller recommended        1
## # … with 1,007 more rows
   ngram_2_review %>% group_by(word) %>%
                   count(sort = TRUE) %>%
                   filter(stringr::str_detect(word,"excellent"))
   ## # A tibble: 1,146 × 2
## # Groups:   word [1,146]
##    word                         n
##    <chr>                    <int>
##  1 excellent communication    73
##  2 excellent support          63
##  3 excellent experience       56
##  4 excellent price            53
##  5 recommended excellent      53
##  6 excellent excellent        51
##  7 excellent quality          49
##  8 excellent condition        46
##  9 delivery excellent         45
## 10 excellent advice           39
## # … with 1,136 more rows
   ngram_2_review %>% group_by(word) %>%
                   count(sort = TRUE) %>%
                   filter(stringr::str_detect(word,"easy | time"))
   ## # A tibble: 501 × 2
## # Groups:   word [501]
##    word                  n
##    <chr>             <int>
##  1 delivery time       119
##  2 multiple times       60
##  3 numerous times       48
##  4 easy process         42
##  5 delivery times       32
##  6 response time        30
##  7 response times       27
##  8 easy transaction     25
##  9 quick time           21
## 10 lead time            19
## # … with 491 more rows
```

z

We are now going to investigate if there are different sets of words that come up depending on the metadata available.

- Checking for correlation between ratings and review_lengths

```
   #Finding the review lengths
corr_data_1 <- review_data %>%
               mutate(review_length = str_count(reviewText)) %>%
               group_by(company_id) %>%
               mutate(average_review_length = mean(review_length),
                      mean_overall = mean(overall)) %>%
               ungroup()
#Plotting
corr_data_1 %>% ggplot(aes(x=average_review_length,
                           y=log(mean_overall))) +
                geom_point(alpha=0.2) +
                geom_smooth(method="lm") +
                ggtitle("Rating scores vs length of reviews")
```

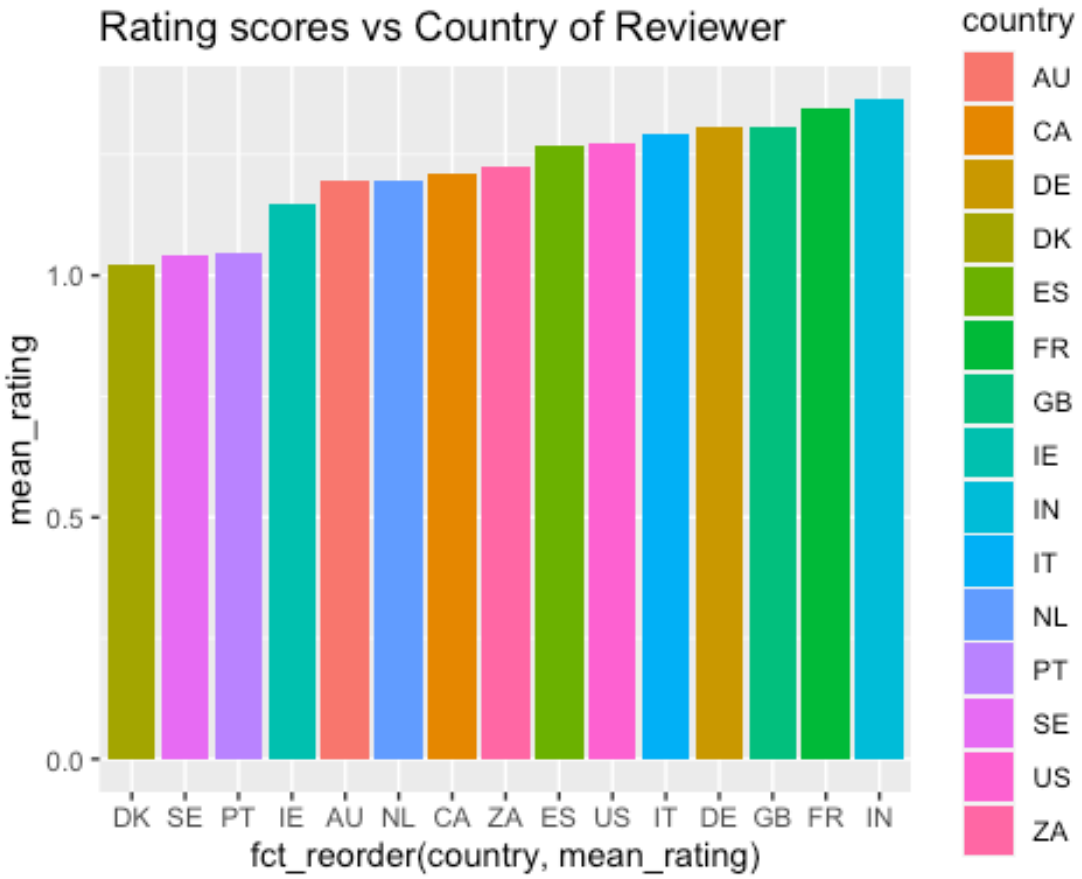


```
   #Performing correlation test
cor.test(corr_data_1$mean_overall,
         corr_data_1$average_review_length,
         method="pearson")
   ##
##  Pearson's product-moment correlation
##
## data:  corr_data_1$mean_overall and corr_data_1$average_review_length
## t = -106.56, df = 22144, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.5908448 -0.5734313
## sample estimates:
##        cor
## -0.5822048
```

The correlation test shows that there is a negative correlation between the ratings and review lengths. It seems that the reviews with lower ratings tend to be longer. This may be due to the unsatisfaction of the customers that more words are needed to explain what their discontent is about.

- Checking for correlation between ratings and country

```
   #Finding top 15 countries with highest reviews
top_15_country <- review_data %>%
                  mutate(country = as.factor(country)) %>%
                  group_by(country) %>%
                  summarise(mean_rating = log(mean(reviewer_rating)),
                            frequency = n()) %>%
                  slice_max(frequency, n = 15)

#Plotting
top_15_country %>% ggplot(aes(x = fct_reorder(country, mean_rating),
                              y = mean_rating, fill = country)) +
                   geom_col() +
                   ggtitle("Rating scores vs Country of Reviewer")
```



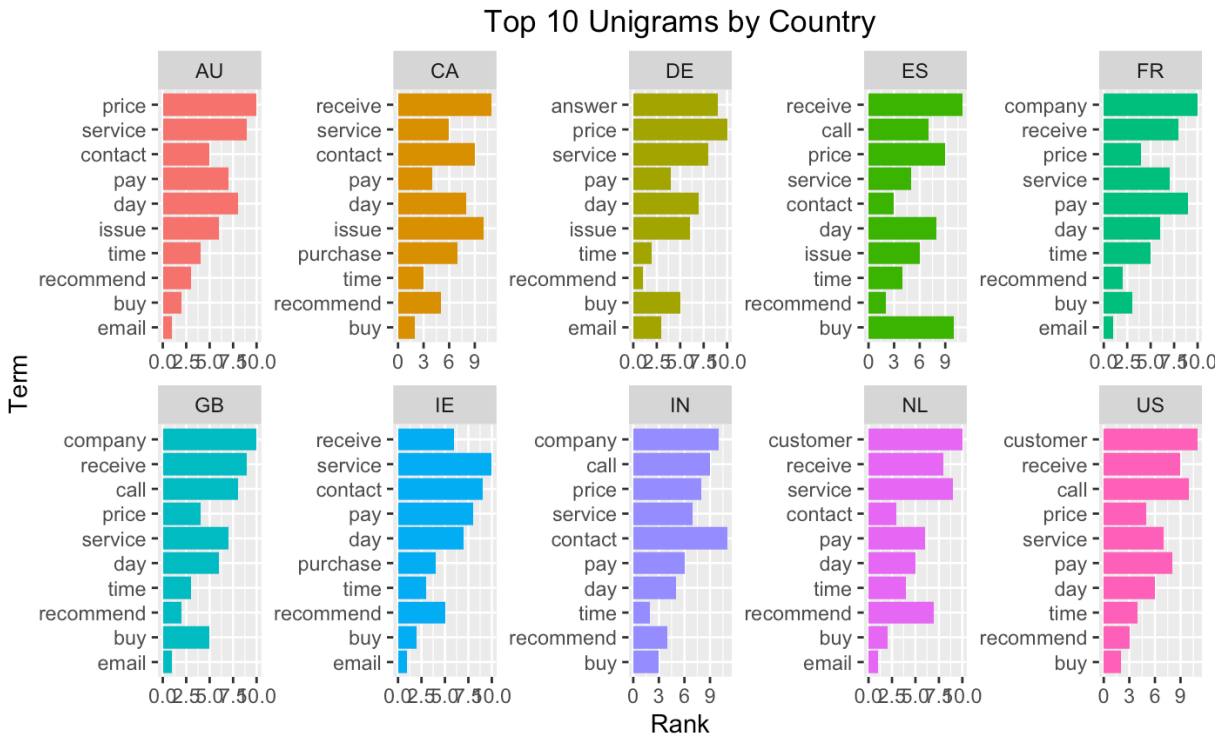
We took the top 15 countries based on the number of reviews to see if there exists any correlation between the location and ratings. We have identified the countries where the ratings seem to be much better than the others. India, France and Britain seems to have higher ratings in general.

- Word importance by metadata

We will now look for words that come up by country

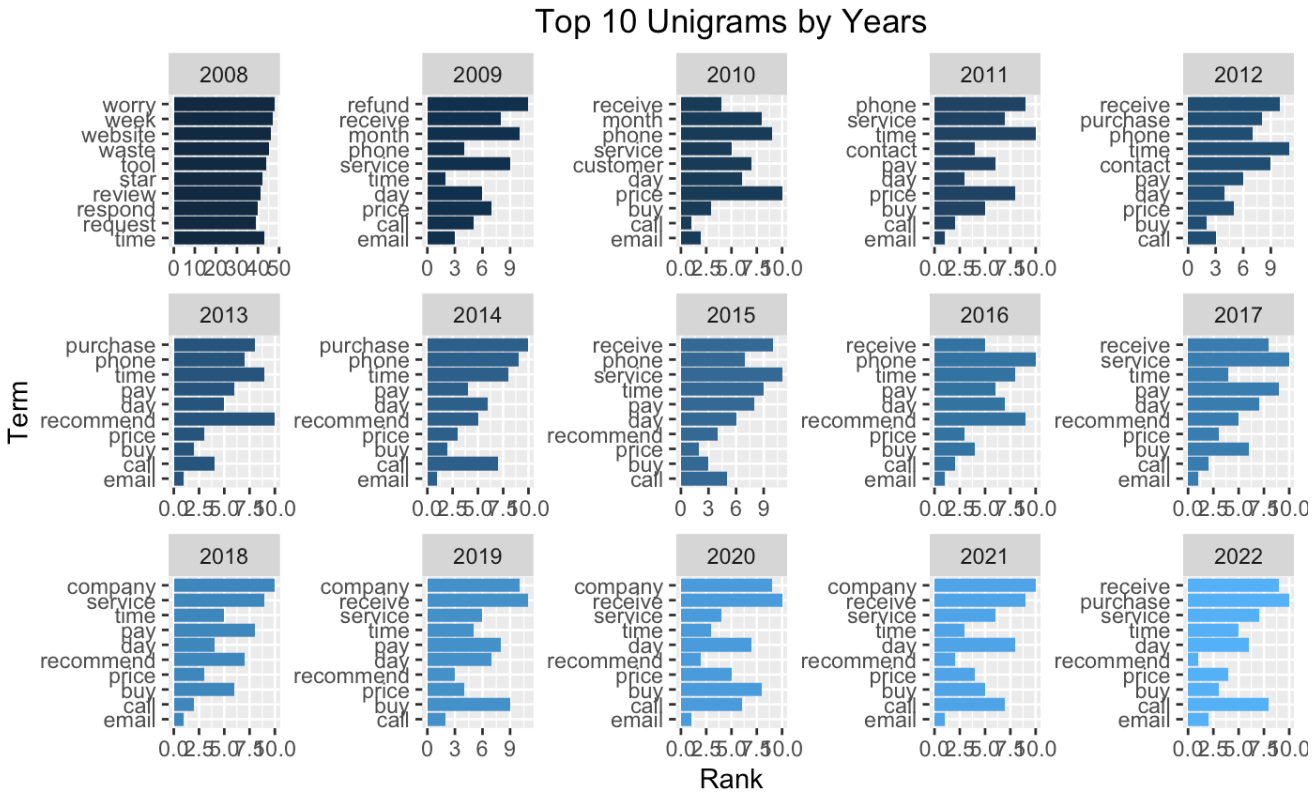
```
   top_country <- review_data_en %>%
  dplyr::select(company_id,country) %>%
  unique(.) %>%
  group_by(country) %>%
  summarise(total =n()) %>%
  arrange(desc(total)) %>%
  top_n(10)

top_tokenized_country <- review_data_en %>%
  dplyr::select(company_id,country) %>%
  filter(country %in% top_country$country) %>%
  unique(.)
```

```r
country_tokens <- review_tokens_by_stores %>%
  right_join(top_tokenized_country) %>% group_by(country,word) %>%
  summarise(total=sum(n)) %>%
  arrange(desc(total))

output_country <- data.frame()
for(countries in 1:nrow(top_country)){
  print(paste0("For year: ",top_country$country[countries]))

  output <- country_tokens %>% ungroup() %>% filter(country == top_country$co
untry[countries]) %>% top_n(10,total) %>% dplyr::select(-total) %>% mutate(ra
nk = row_number())
  output_country <- rbind(output_country, output)
}

output_country %>% group_by(country) %>%
                   slice_max(rank, n = 10) %>%
                   ungroup() %>%
                   ggplot(aes(rank, fct_reorder(word, rank), fill = country))
 +
                   geom_col(show.legend = FALSE) +
                   facet_wrap(~ country, ncol = 5, scales = "free") +
                   labs(title="Top 10 Unigrams by Country",
                   y = "Term",
                   x = "Rank") +
                   theme(plot.title = element_text(hjust = 0.5))
```



We also try to see what word are becoming more important over the years.

```
   top_year <- review_data_en %>%
  dplyr::select(company_id,year) %>%
  unique(.) %>%
  group_by(year) %>%
  summarise(total =n()) %>%
  arrange(desc(total))

top_tokenized_year <- review_data_en %>%
  dplyr::select(company_id,year) %>%
  filter(year %in% top_year$year) %>%
  unique(.)

year_tokens <- review_tokens_by_stores %>%
  right_join(top_tokenized_year) %>% group_by(year,word) %>%
  summarise(total=sum(n)) %>%
  arrange(desc(total))

output_year <- data.frame()
for(years in 1:nrow(top_year)){
  print(paste0("For year: ",top_year$year[years]))

  output1 <- year_tokens %>% ungroup() %>% filter(year == top_year$year[years
]) %>% top_n(10,total) %>% dplyr::select(-total) %>% mutate(rank = row_number
())
  output_year <- rbind(output_year, output1)
}

output_year %>% group_by(year) %>%
                    slice_max(rank, n = 10) %>%
                    ungroup() %>%
                    ggplot(aes(rank, fct_reorder(word, rank), fill = year)) +
                    geom_col(show.legend = FALSE) +
                    facet_wrap(~ year, ncol = 5, scales = "free") +
                    labs(title="Top 10 Unigrams by Years",
                    y = "Term",
                    x = "Rank") +
                    theme(plot.title = element_text(hjust = 0.5))
```


Top 10 Unigrams by Years
2008
worry
week
website
waste
tool
star
review
respond
request
time
0 10 20 30 40 50
2009
refund
receive
month
phone
service
time
day
price
call
email
0 3 6 9
2010
receive
month
phone
service
customer
day
price
buy
call
email
0.0 2.5 5.0 7.5 10.0
2011
phone
service
time
contact
pay
day
price
buy
call
email
0.0 2.5 5.0 7.5 10.0
2012
receive
purchase
phone
time
contact
pay
day
price
buy
call
0 3 6 9
2013
purchase
phone
time
pay
day
recommend
price
buy
call
email
0.0 2.5 5.0 7.5 10.0
2014
purchase
phone
time
pay
day
recommend
price
buy
call
email
0.0 2.5 5.0 7.5 10.0
2015
receive
phone
service
time
pay
day
recommend
price
buy
call
0 3 6 9
2016
receive
phone
time
pay
day
recommend
price
buy
call
email
0.0 2.5 5.0 7.5 10.0
2017
receive
service
time
pay
day
recommend
price
buy
call
email
0.0 2.5 5.0 7.5 10.0
2018
company
service
time
pay
day
recommend
price
buy
call
email
0.0 2.5 5.0 7.5 10.0
2019
company
receive
service
time
pay
day
recommend
price
buy
call
0 3 6 9
2020
company
receive
service
time
day
recommend
price
buy
call
email
0.0 2.5 5.0 7.5 10.0
2021
company
receive
service
time
day
recommend
price
buy
call
email
0.0 2.5 5.0 7.5 10.0
2022
receive
purchase
service
time
day
recommend
price
buy
call
email
0.0 2.5 5.0 7.5 10.0
Term
Rank

Part B

In this section, we will perform sentiment analysis and feature extraction from the reviews

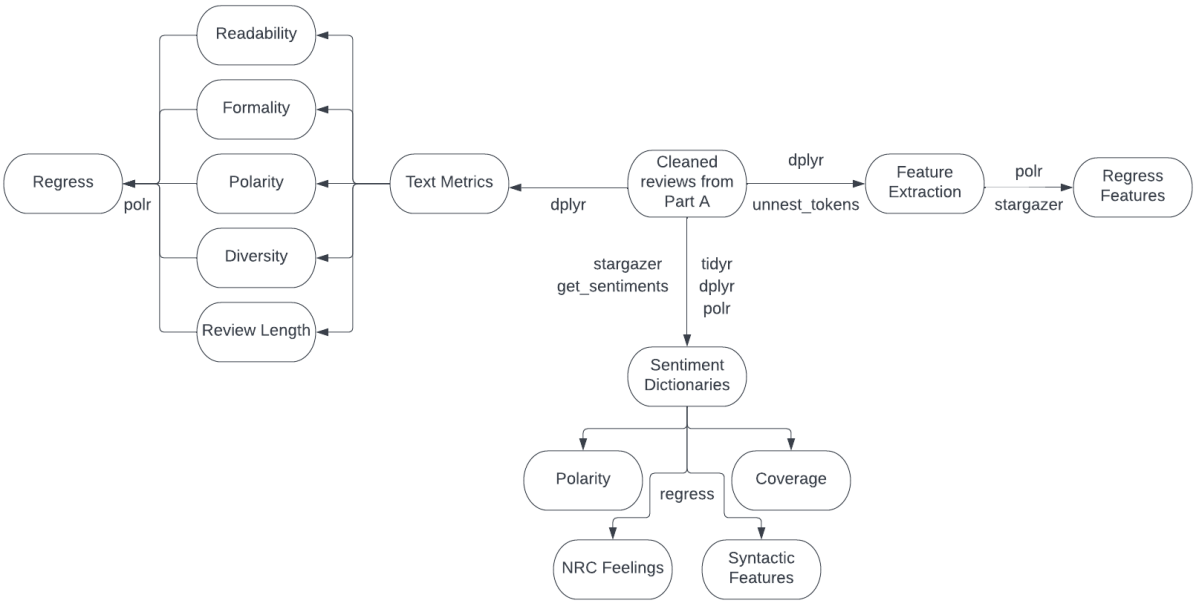
We will start off by finding some text features and connecting them with the target variable. This will be done through regression models. We will compare the models to find which features have significant effect on the target variable.

Next, we will examine the text metrics to evaluate the effects on the target. The readability, formality, diversity, polarity and review length metrics will be utilized.

Then, we will try to understand the sentiments of the texts. Using the analysed sentiments, we will find the best fit sentiment dictionary for this dataset.

Finally, we will combine the sentiments with some syntactic features to predict the target variable as well.

The NLP pipeline used in this section is as follows:



B.1: Dataset Preparation

- Loading the relevant libraries for this section

```
   library(qdap)
   library(tidytext)
   library(tidyverse)
   library(readr)
   library(tm)
   library(qdap)
   library(textclean)
   library(wordcloud)
   library(reshape2)
   library(stringi)
   library(textstem)
   library(sentimentr)
   library(lexicon)
   library(FSelector)
   library(future.apply)
   library(rJava)
   #install.packages('egg')
   library(egg)
   library(stargazer)
   library(MASS)
   library(caret)
   library(magrittr)
   #install.packages('ordinal')
   library(ordinal)
   library(data.table)
   #install.packages('textdata')
   library(textdata)
   library(hunspell)
```

- We will use the prepared dataset from the previous section.

```
   #Importing the dataset
dataset <- readRDS("review_data_en.rds")
dataset_stores <- readRDS("review_data_by_stores.rds")
   #Setting the correct datatype of the target variable
dataset$reviewer_rating <- as.numeric(dataset$reviewer_rating)
dataset$overall <- as.numeric(dataset$overall)
```

- Text cleaning

  Applying some additional text cleaning relevant for the analysis of this section.

```
   #Getting sample reviews to understand the required cleaning
#dataset$reviewText %>%sample(100)%>% check_text(n=1)
```

```
#Cleaning the reviews
dataset$reviewText <- replace_emoticon(dataset$reviewText) #convert emoticons
 to words
dataset$reviewText <- replace_emoji(dataset$reviewText) #convert emojis to wo
rds
dataset$reviewText <- replace_incomplete(dataset$reviewText) #dealing with in
complete sentences
dataset$reviewText <- add_missing_endmark(dataset$reviewText) #putting requir
ed endmarks
dataset$reviewText <- textshape::split_sentence(dataset$reviewText)
dataset$reviewText <- replace_non_ascii(dataset$reviewText) #replacing common
 non-ascii words
dataset$reviewText <- gsub("[ |\t]+", " ", dataset$reviewText) #removing whit
e space
```

- Getting the tokenized reviews at both store and review level

```
   #Using selective stop words from part A
custom_words <- c("class", "job", "shop", "hand", "friend", "date", "service"
, "review", "product", "customer", "day", "days", "received", "money", "week"
, "buy", "company", "phone", "don", "ve", "ofcom", "ninja", "moonfruit", "can
celled", "virgin", "told", "people", "arrived", "paul", "av", "ben", "josh",
"andy", "tom", "richer", "plugin", "amp", "hifi", "aquiss", "sounds", "contro
ller", "zoltan", "ian", "idnet", "glue", "proxy", "helen", "eno", "oled", "sy
pwai", "george", "peter", "panamoz", "pine", "dotsquares", "laura", "vape", "
amiga", "biz", "technoworld", "proxies")

#Binding them with the default stop word dictionary
tibble(word = custom_words,lexicon = rep("custom",length(custom_words))) %>%
bind_rows(stop_words) -> custom_stopwords

#Tokenizing the reviews at store level
tokens_all <- dataset_stores %>%
              unnest_tokens(word,grouped_reviews) %>%
              count(word,company_id) %>%
              anti_join(custom_stopwords)

#Tokenizing the reviews at review level
tokens_all_review <- dataset %>%
              unnest_tokens(word,reviewText) %>%
              count(word,review_id) %>%
              anti_join(custom_stopwords)

count <- tokens_all_review %>% group_by(review_id) %>% summarise(wordcount =
n())

tokens_all_review <- tokens_all_review %>% left_join(count)
```

B.2: Feature Extraction

- Finding the features to extract

We divide the ratings into two categories and then look for top terms to select some terms to use as features to predict ratings

```
   #Getting the average ratings for each store for rating categorizing
rating_categories <- dataset %>%
  group_by(company_id) %>%
  summarise(avg_rating = mean(reviewer_rating)) %>%
  ungroup()

#Finding the levels to aggregate the words
quantile(rating_categories$avg_rating)
   ##       0%      25%      50%      75%     100%
## 1.000000 2.625160 4.214286 4.886752 5.000000
   #Assigning ratings to two categories
rating_categories$rating_category <- ifelse(rating_categories$avg_rating < 4.
2, 1, 2)

ratings_categories_tokens <- tokens_all %>% left_join(rating_categories) %>%
  group_by(rating_category,word) %>% summarise(total =sum(n))

#Looking at potential features to extract
ratings_categories_tokens %>% filter(rating_category == 1) %>% arrange(desc(t
otal)) %>% top_n(10)
   ## # A tibble: 10 × 3
## # Groups:   rating_category [1]
##    rating_category word      total
##              <dbl> <chr>     <int>
##  1               1 time       2220
##  2               1 delivery   1981
##  3               1 refund     1661
##  4               1 email      1633
##  5               1 avoid      1280
##  6               1 price      1215
##  7               1 item       1208
##  8               1 website    1206
##  9               1 bought     1152
## 10               1 excellent  1137

   ratings_categories_tokens %>% filter(rating_category == 2) %>% arrange(des
c(total)) %>% top_n(10)
   ## # A tibble: 10 × 3
## # Groups:   rating_category [1]
##    rating_category word      total
##              <dbl> <chr>     <int>
```

```
##  1              2 excellent  3407
##  2              2 recommend  2592
##  3              2 helpful    2005
##  4              2 delivery   1950
##  5              2 time       1825
##  6              2 highly     1478
##  7              2 support    1404
##  8              2 easy       1382
##  9              2 price      1319
## 10              2 quick      1251
```

- Extracting features from text and creating regression models

From the previous output, we will now select some words as features and explore the regressions.

We have formatted the features are binary values in the code. Hence, we will use polr function to perform logistical regression as the appropriate method.

```
   reviews_for_feature <- dataset %>% dplyr::select(review_id, reviewText, st
ore_description, reviewer_rating, overall)
reviews_for_feature$reviewText <- tolower(reviews_for_feature$reviewText)
reviews_for_feature$store_description <- tolower(reviews_for_feature$store_de
scription)

#Finding reviews that contain "delivery"
reviews_for_feature %>% group_by(review_id) %>%
 summarise(TF=sum(str_detect(reviewText, "delivery"))) %>%
 mutate(delivery=ifelse(TF>0, "1", "0")) %>%
 dplyr::select(review_id,delivery) -> delivery_feature

delivery_feature_final <- delivery_feature %>%
 inner_join(dataset) %>%
 dplyr::select(review_id, reviewer_rating, delivery)
model_delivery <- polr(factor(reviewer_rating) ~ delivery, data=delivery_feat
ure_final, Hess=TRUE, method = c("logistic"))

#Finding reviews that contain "time"
reviews_for_feature %>% group_by(review_id) %>%
 summarise(TF=sum(str_detect(reviewText, "time"))) %>%
 mutate(time=ifelse(TF>0, "1", "0")) %>%
 dplyr::select(review_id,time) -> time_feature

time_feature_final <- time_feature %>%
 inner_join(dataset) %>%
 dplyr::select(review_id, reviewer_rating, time)
model_time <- polr(factor(reviewer_rating) ~ time, data=time_feature_final, H
ess=TRUE, method = c("logistic"))

#Finding reviews that contain "price"
reviews_for_feature %>% group_by(review_id) %>%
```

```
 summarise(TF=sum(str_detect(reviewText, "price"))) %>%
 mutate(price=ifelse(TF>0, "1", "0")) %>%
 dplyr::select(review_id,price) -> price_feature

price_feature_final <- price_feature %>%
 inner_join(dataset) %>%
 dplyr::select(review_id, reviewer_rating, price)
model_price <- polr(factor(reviewer_rating) ~ price, data=price_feature_final
, Hess=TRUE, method = c("logistic"))

#Finding reviews that contain "helpful"
reviews_for_feature %>% group_by(review_id) %>%
 summarise(TF=sum(str_detect(reviewText, "helpful"))) %>%
 mutate(helpful=ifelse(TF>0, "1", "0")) %>%
 dplyr::select(review_id,helpful) -> helpful_feature

helpful_feature_final <- helpful_feature %>%
 inner_join(dataset) %>%
 dplyr::select(review_id, reviewer_rating, helpful)
model_helpful <- polr(factor(reviewer_rating) ~ helpful, data=helpful_feature
_final, Hess=TRUE, method = c("logistic"))

#Finding reviews that contain "easy"
reviews_for_feature %>% group_by(review_id) %>%
 summarise(TF=sum(str_detect(reviewText, "easy"))) %>%
 mutate(easy=ifelse(TF>0, "1", "0")) %>%
 dplyr::select(review_id,easy) -> easy_feature

easy_feature_final <- easy_feature %>%
 inner_join(dataset) %>%
 dplyr::select(review_id, reviewer_rating, easy)
model_easy <- polr(factor(reviewer_rating) ~ easy, data=easy_feature_final, H
ess=TRUE, method = c("logistic"))

#Finding reviews that contain "excellent"
reviews_for_feature %>% group_by(review_id) %>%
 summarise(TF=sum(str_detect(reviewText, "excellent"))) %>%
 mutate(excellent=ifelse(TF>0, "1", "0")) %>%
 dplyr::select(review_id,excellent) -> excellent_feature

excellent_feature_final <- excellent_feature %>%
 inner_join(dataset) %>%
 dplyr::select(review_id, reviewer_rating, excellent)
model_excellent <- polr(factor(reviewer_rating) ~ excellent, data=excellent_f
eature_final, Hess=TRUE, method = c("logistic"))

#Finding reviews that contain upper case letter
dataset %>% group_by(review_id) %>%
 summarise(TF=sum(str_detect(reviewText, "\\b[A-Z]+\\b"))) %>%
 mutate(upper_case=ifelse(TF>0, "1", "0")) %>%
```

```
 dplyr::select(review_id,upper_case) -> upper_case_feature

upper_case_feature_final <- upper_case_feature %>%
 inner_join(dataset) %>%
 dplyr::select(review_id, reviewer_rating, upper_case)
model_upper <- polr(factor(reviewer_rating) ~ upper_case, data=upper_case_fea
ture_final, Hess=TRUE, method = c("logistic"))

#Finding reviews that have email
dataset %>% group_by(review_id) %>%
  summarise(TF=sum(str_detect(email, "  "))) %>%
  mutate(emails=ifelse(TF>0, "0", "1")) %>%
 dplyr::select(review_id,emails) -> email_feature

email_feature_final <- email_feature %>%
 inner_join(dataset) %>%
 dplyr::select(review_id, reviewer_rating, emails)
model_email <- polr(factor(reviewer_rating) ~ emails, data=email_feature_fina
l, Hess=TRUE, method = c("logistic"))

#Finding reviews that have phone number
dataset %>% group_by(review_id) %>%
 summarise(TF=sum(str_detect(phone_number, "  "))) %>%
 mutate(phone=ifelse(TF>0, "0", "1")) %>%
 dplyr::select(review_id,phone) -> phone_feature

phone_feature_final <- phone_feature %>%
 inner_join(dataset) %>%
 dplyr::select(review_id, reviewer_rating, phone)
model_phone <- polr(factor(reviewer_rating) ~ phone, data=phone_feature_final
, Hess=TRUE, method = c("logistic"))

#Finding reviews that contain exclamations
dataset %>% group_by(review_id) %>%
 summarise(TF=sum(str_detect(review, "!"))) %>%
 mutate(exclm=ifelse(TF>0, "1", "0")) %>%
 dplyr::select(review_id,exclm) -> exclm_feature

exclm_feature_final <- exclm_feature %>%
 inner_join(dataset) %>%
 dplyr::select(review_id, reviewer_rating, exclm)
model_exclm <- polr(factor(reviewer_rating) ~ exclm, data=exclm_feature_final
, Hess=TRUE, method = c("logistic"))

#Comparing all the feature models
stargazer::stargazer(model_delivery, model_time, model_upper, model_easy, mod
el_excellent, model_exclm, model_helpful, model_price, model_phone, model_ema
il,
                     type = "text", add.lines = list(c("AIC", round(AIC(model
_delivery),1),
```

```
                                                                  round(AIC(model
_time),1),
                                                                  round(AIC(model
_upper),1),
                                                                  round(AIC(model
_easy),1),
                                                                  round(AIC(model
_excellent),1),
                                                                  round(AIC(model
_helpful),1),
                                                                  round(AIC(model
_price),1),
                                                                  round(AIC(model
_email),1),
                                                                  round(AIC(model
_exclm),1),
                                                                  round(AIC(model
_phone),1))),
                     out = "features.txt", single.row = T, align = T, flip =
T)
   ##
## ============================================================================
============================================================================
=================================
##
               Dependent variable:

##              ---------------------------------------------------------
-----------------------------------------------------------------------------
---------------------------------
##
                 reviewer_rating

##                     (1)               (2)               (3)              (4
)              (5)               (6)              (7)              (8)
       (9)              (10)
## ----------------------------------------------------------------------------
-----------------------------------------------------------------------------
---------------------------------
## delivery1    0.309*** (0.041)


## time1                         -0.358*** (0.033)


## upper_case1                                      -0.883*** (0.032)


## easy1                                                                  1.875***
 (0.085)
```

```
## excellent1
         2.680*** (0.081)

## exclm1
                          -0.266*** (0.031)

## helpful1
                                                 1.659*** (0.064)

## price1
                                                                    0.722*** (0.045)

## phone1

 1.006*** (0.028)
## emails1

                   1.806*** (0.032)
## ------------------------------------------------------------------------
------------------------------------------------------------------------
---------------------------------
## AIC               42734.4           42678.5           41973.5          4203
2.3           40568             41858.6           42511.1           39436.2
     42717.8           41479.4
## Observations      22,119            22,119            22,119            22,
119           22,119            22,119             22,119            22,119
      22,119            22,119
## ========================================================================
========================================================================
=================================
## Note:

       *p<0.1; **p<0.05; ***p<0.01
```

The outcome of the analysis shows that each of the features are significant to predict the target variable and have a high AIC value.

- Finding the effects of each feature on ratings

Now, we use all the features together to create a regression model to predict the target variable.

```
   #Combining all features into a data.frame
review_feature_final <- reviews_for_feature %>%
                        left_join(delivery_feature) %>%
                        left_join(time_feature) %>%
                        left_join(upper_case_feature)%>%
                        left_join(easy_feature)%>%
                        left_join(excellent_feature)%>%
                        left_join(helpful_feature)%>%
                        left_join(price_feature)%>%
                        left_join(email_feature)%>%
                        left_join(phone_feature)%>%
                        left_join(exclm_feature)

#Creating a backup
#saveRDS(review_feature_final,"review_feature_final.rds")

#Conducting regression with all the features
regress_all_features <- review_feature_final %>% dplyr::select(,-c(review_id,
 reviewText, store_description, overall))
model_features_all <- polr(factor(reviewer_rating) ~ ., data = regress_all_fe
atures)
```

B.3: Text Metrics Analysis

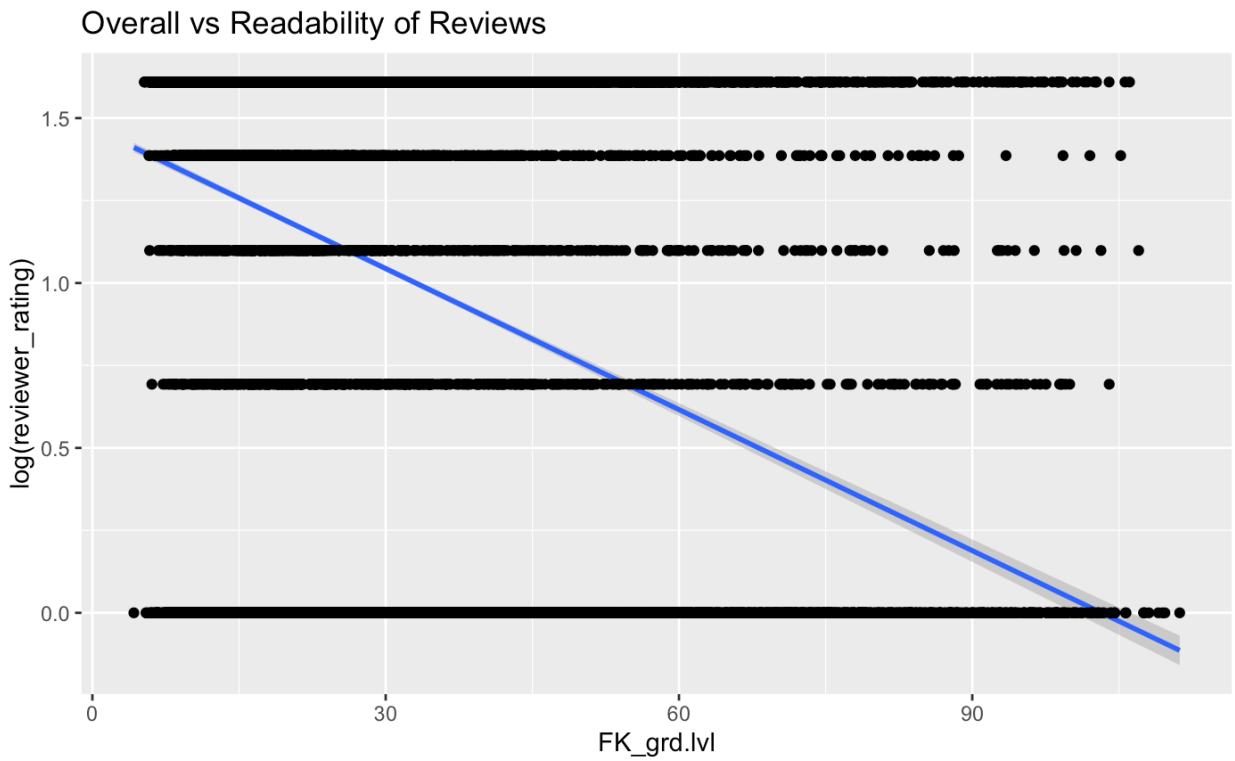
Here, we will analyse the readability, formality, polarity, diversity and review lengths of both the reviews and the store descriptions.

B.3.1: Checking if readability of reviews has effect on ratings

Readability refers to the ease of reading a text. We calculate the readability of the reviews to see if it has any effect on the ratings

```
   all_review <- dataset %>% dplyr::select(review_id, reviewText, reviewer_ra
ting) %>% unique(.)

readability_all_review <- data.frame()

for(i in 1:nrow(all_review)){

  readability <- data.frame()

  desc <- iconv(all_review$reviewText[i])
  desc <- removeNumbers(desc)
```

```
  desc <- removePunctuation(desc)

  tryCatch(readability <- flesch_kincaid(desc),error=function(e){
    cat("Error parsing")
  })

  if(!is.null(readability$Readability)){

     readability <- readability$Readability
     readability$review_id <- all_review$review_id[i]
     readability_all_review <- bind_rows(readability_all_review,readability)
   }

  print(i)
}
   #Adding readability output to dataset
dataset %>% left_join(readability_all_review) -> dataset_read_review

readability_review_plot <- dataset_read_review %>% group_by(review_id) %>%
  dplyr::select(FK_grd.lvl, reviewer_rating) %>%
  na.omit() %>%
  ggplot(aes(x = FK_grd.lvl,y = log(reviewer_rating))) +
  geom_smooth(method="lm") +
  geom_point() +
  ggtitle("Overall vs Readability of Reviews")
#ggsave("readability_review_plot.png")

#Plotting the relation between the readability of the store description and t
he rating scores
cor.test(dataset_read_review$reviewer_rating,
 dataset_read_review$FK_grd.lvl,
 method="pearson")
```



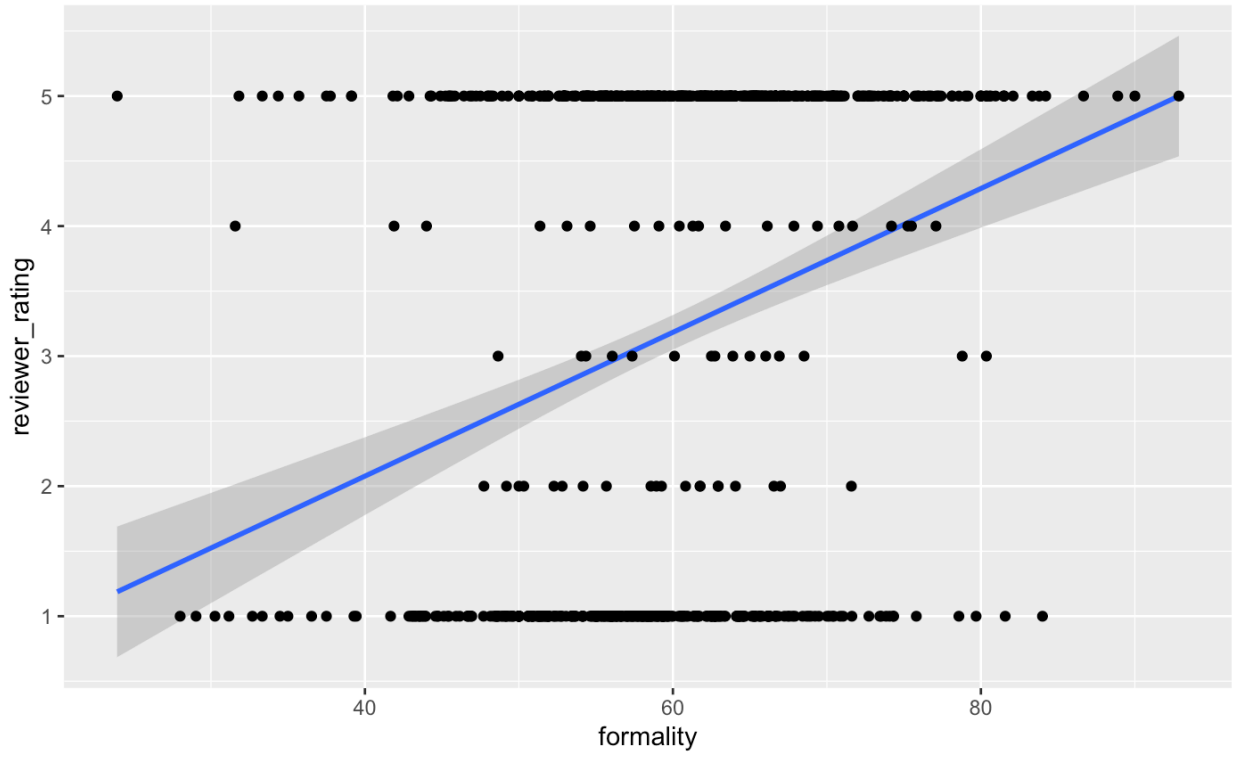
It is seen that there is a negative correlation between readability and reviewer ratings

B.3.2: Checking the formality of the reviews

Formality refers to the conforming to standard language conventions. We will calculate the formality scores for the reviews and regress it against the ratings

```
   formality_all_review <- data.frame()

for(i in 1:nrow(all_review)){

  formality <- data.frame()

  desc <- iconv(all_review$reviewText[i])
  desc <- removeNumbers(desc)
  desc <- removePunctuation(desc)

  tryCatch(formality <- formality(desc),error=function(e){
    cat("Error parsing")
  })

  if(!is.null(formality$formality)){

     formality <- formality$formality
```

```
    formality$review_id <- all_review$review_id[i]
    formality_all_review <- bind_rows(formality_all_review,formality)
   }

  print(i)
}

#saveRDS(formality_all_review, "formality_all_review.rds")
   #Adding formality output to dataset
dataset_read_review %>% left_join(formality_all_review) -> dataset_read_revie
w

formality_review_plot <- dataset_read_review %>% group_by(review_id) %>%
  dplyr::select(formality, reviewer_rating) %>%
  na.omit() %>%
  ggplot(aes(x=formality,y=reviewer_rating))+geom_smooth(method="lm")+geom_po
int()
#ggsave("formality_review_plot.png")
```

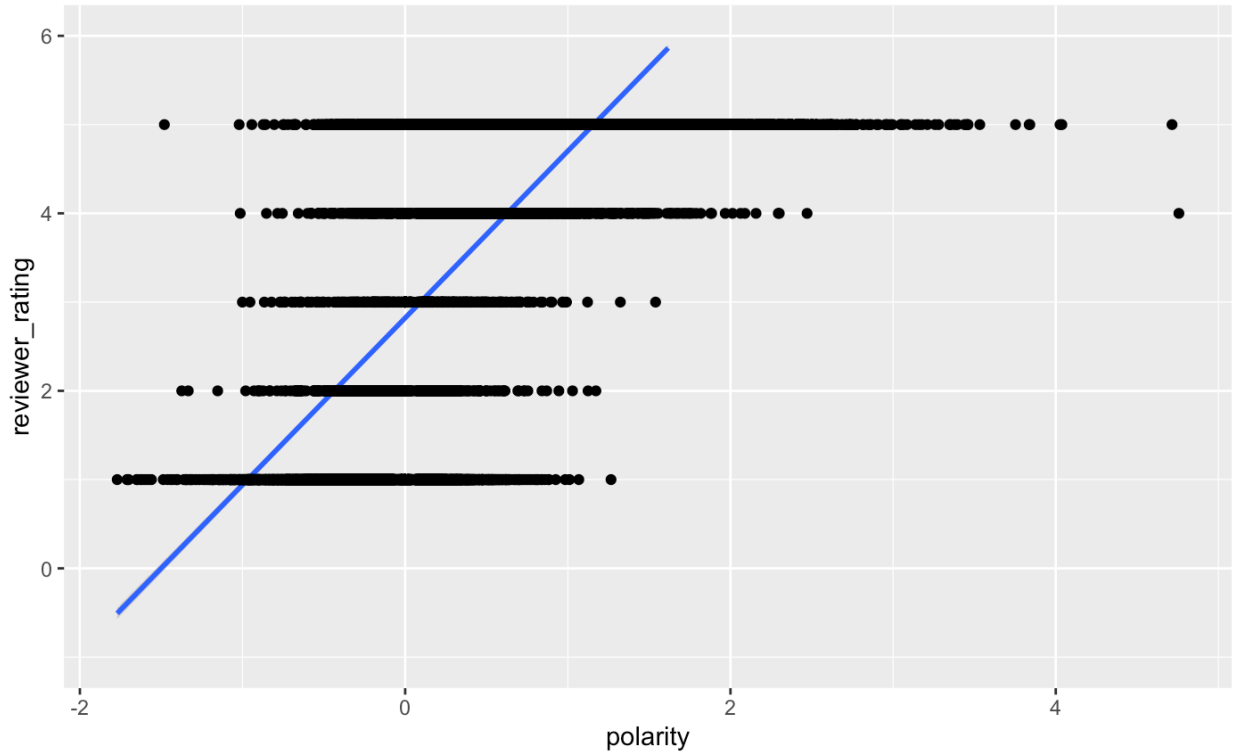


Formality has a positive relationship with reviewer ratings in the case of reviews

B.3.3: Checking the polarity of the reviews

Now, we will analyse whether the inclination of the sentence sentiments of the reviews have any effect on the ratings

```
   polarity_all_review <- data.frame()

for(i in 1:nrow(all_review)){

  polarity <- data.frame()

  desc <- iconv(all_review$reviewText[i])
  desc <- removeNumbers(desc)
  desc <- removePunctuation(desc)

  tryCatch(polarity <- polarity(desc),error=function(e){
    cat("Error parsing")
  })

  if(!is.null(polarity$all)){

     polarity <- polarity$all
     polarity$review_id <- all_review$review_id[i]
     polarity_all_review <- bind_rows(polarity_all_review,polarity)
   }

  print(i)
}
   #Adding polarity output to dataset
dataset_read_review %>% left_join(polarity_all_review) -> dataset_read_review

polarity_review_plot <- dataset_read_review %>% group_by(review_id) %>%
  dplyr::select(polarity, reviewer_rating) %>%
  na.omit() %>%
  ggplot(aes(x = polarity,y = reviewer_rating)) + geom_smooth(method="lm") +
geom_point() + ylim(-1,6)
#ggsave("polarity_review_plot.png")
```



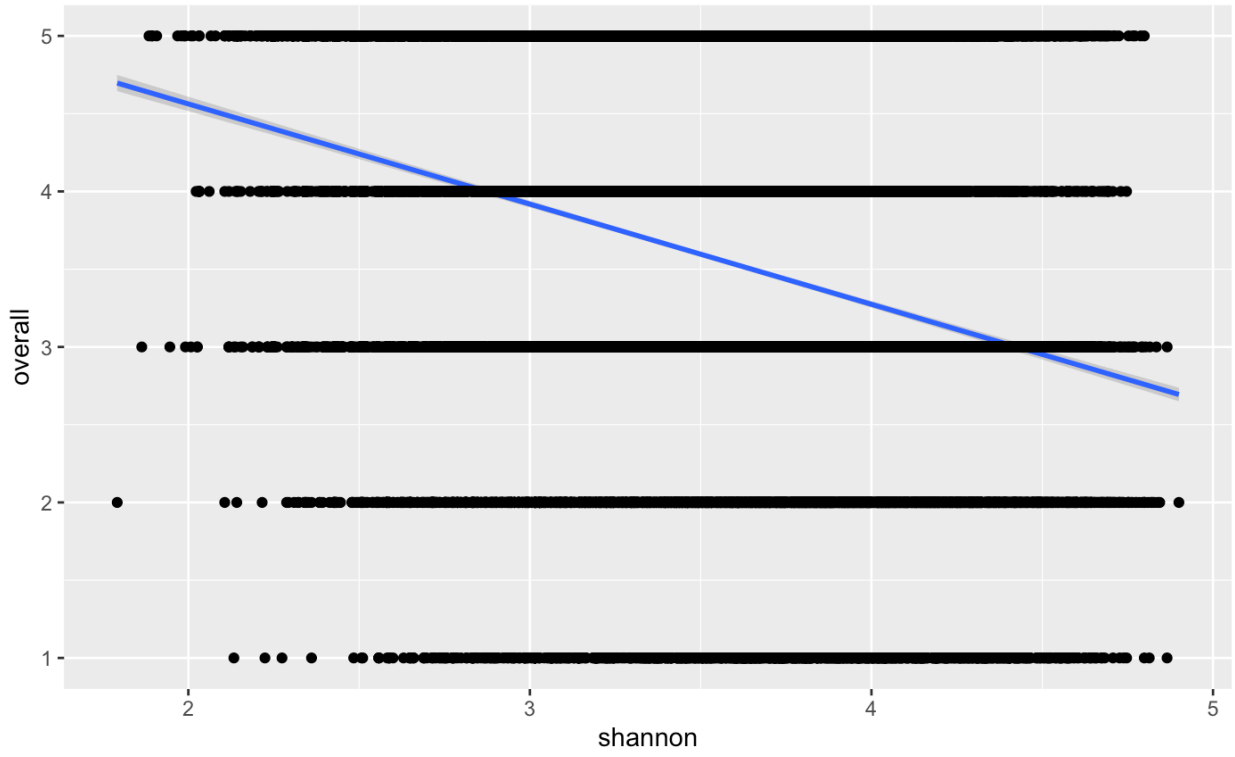
The reviews have a positive relation between ratings and polarity

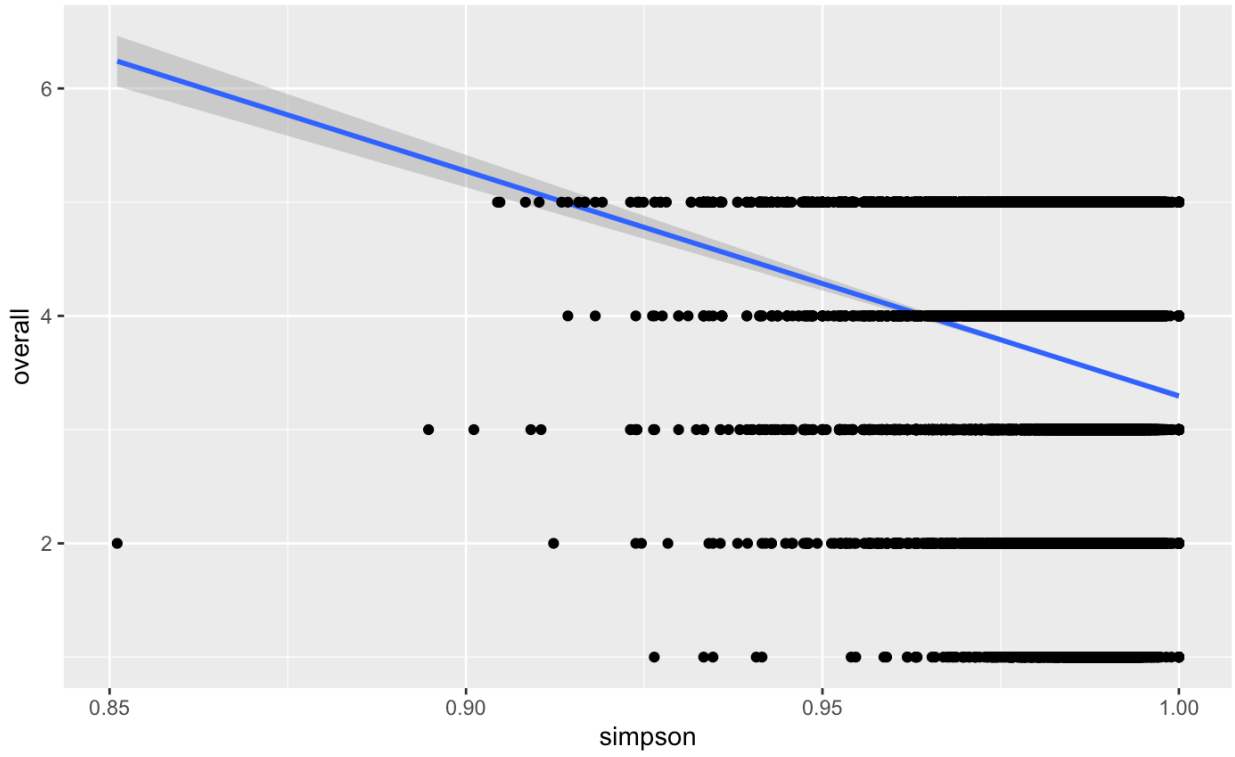
B.3.4: Checking for the diversity of the reviews against ratings

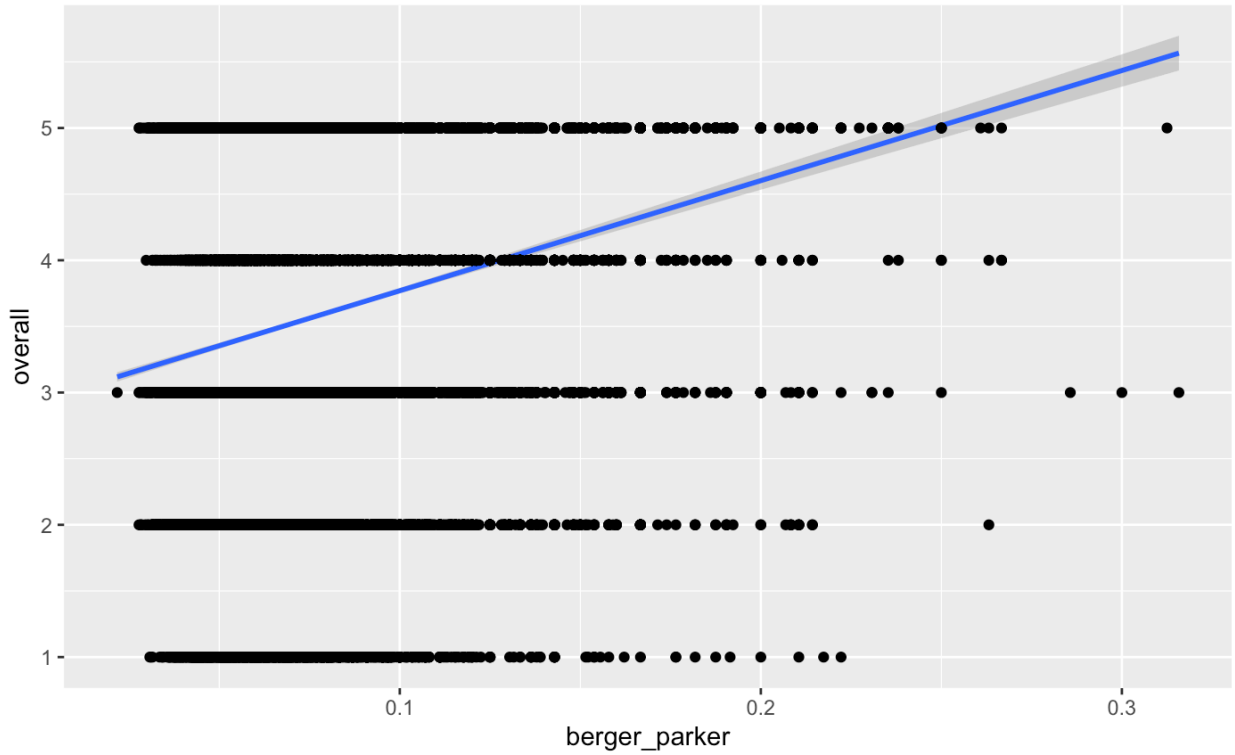
We also look at different diversity scores to see if those have any relation with ratings

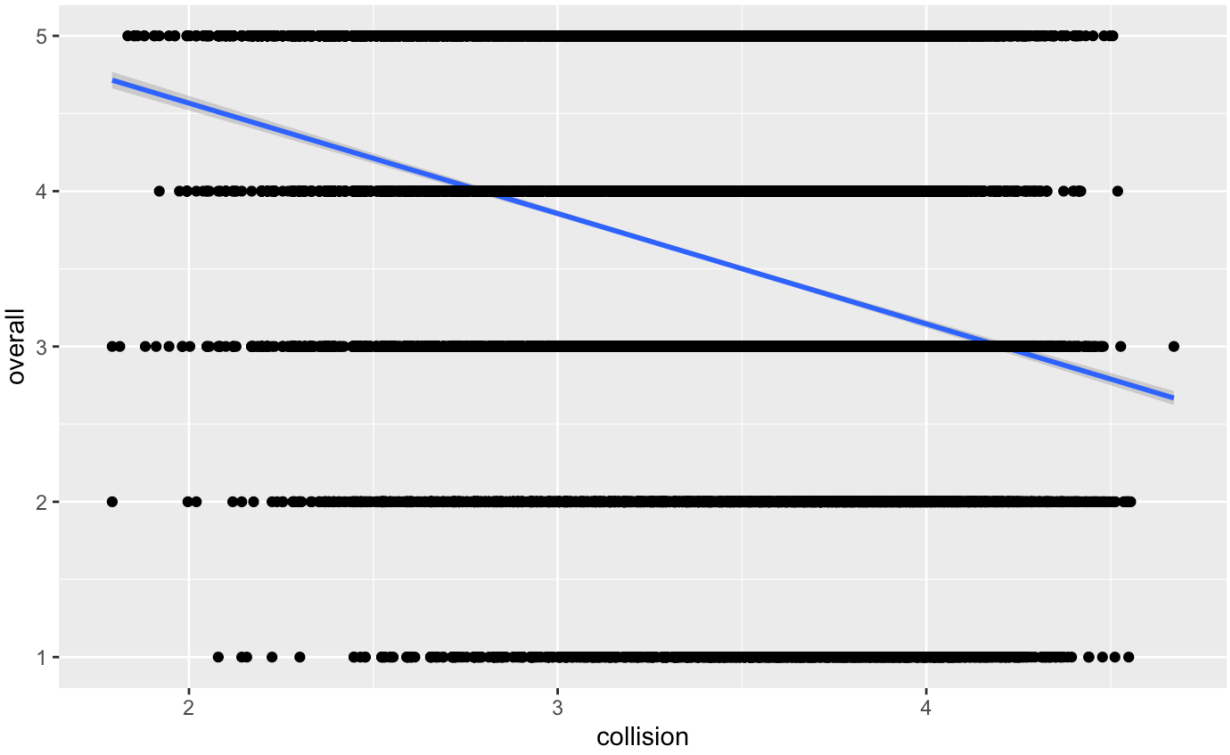
```
   diversity_review <- qdap::diversity(all_review$reviewText,all_review$revie
w_id)
diversity_review$review_id <- as.numeric(diversity_review$review_id)

#Adding polarity output to dataset
dataset_read_review %>% left_join(diversity_review) -> dataset_read_review

shannon_review_plot <- dataset_read_review %>% group_by(review_id) %>%
  dplyr::select(shannon, overall) %>%
  na.omit() %>%
  ggplot(aes(x = shannon,y = overall)) + geom_smooth(method="lm") + geom_poin
t()
ggsave("shannon_review_plot.png")

simpson_review_plot <- dataset_read_review %>% group_by(company_id) %>%
  dplyr::select(simpson, overall) %>%
  na.omit() %>%
  ggplot(aes(x = simpson,y = overall)) + geom_smooth(method="lm") + geom_poin
```

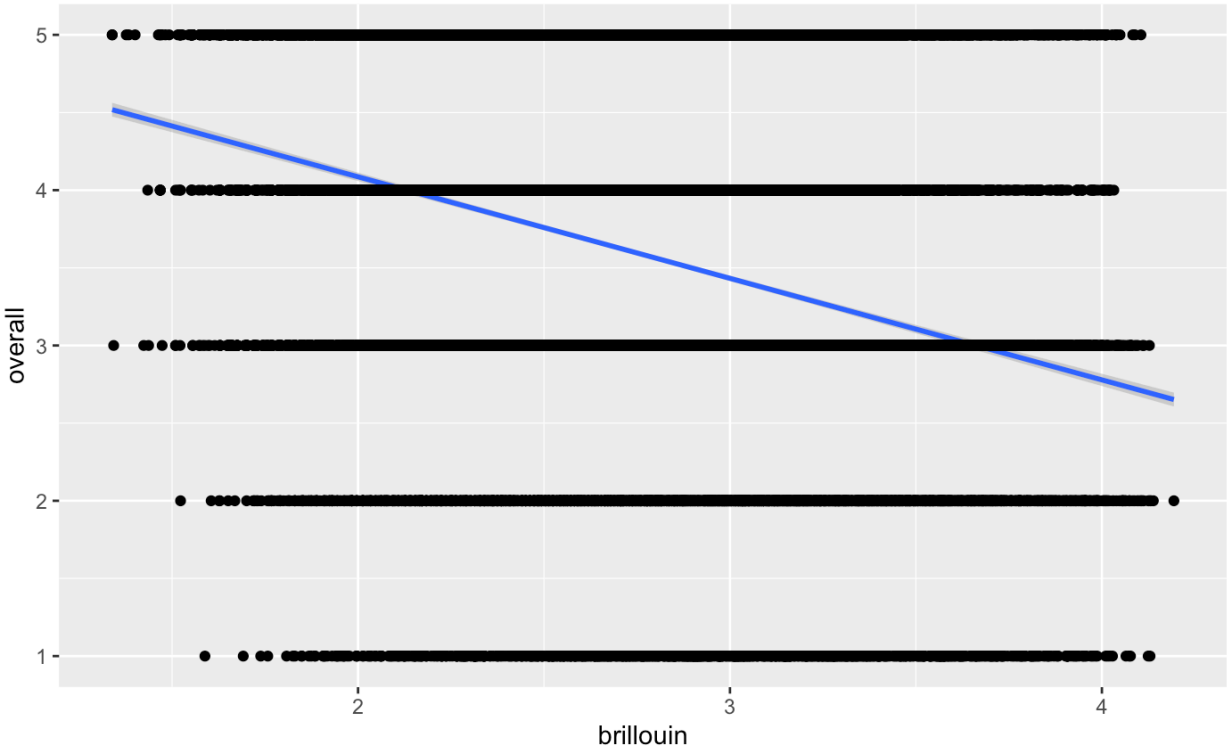
```
t()
ggsave("simpson_review_plot.png")

berger_parker_review_plot <- dataset_read_review %>% group_by(company_id) %>%
  dplyr::select(berger_parker, overall) %>%
  na.omit() %>%
  ggplot(aes(x = berger_parker,y = overall)) + geom_smooth(method="lm") + geo
m_point()
ggsave("berger_parker_review_plot.png")

collision_review_plot <-dataset_read_review %>% group_by(company_id) %>%
  dplyr::select(collision, overall) %>%
  na.omit() %>%
  ggplot(aes(x = collision,y = overall)) + geom_smooth(method="lm") + geom_po
int()
ggsave("collision_review_plot.png")

brillouin_review_plot <- dataset_read_review %>% group_by(company_id) %>%
  dplyr::select(brillouin, overall) %>%
  na.omit() %>%
  ggplot(aes(x = brillouin,y = overall)) + geom_smooth(method="lm") + geom_po
int()
ggsave("brilouin_review_plot.png")
```




overall
6
4
2
0.85
0.90
0.95
1.00
simpson


overall
5
4
3
2
1
0.1
0.2
0.3
berger_parker


overall
5
4
3
2
1
2
3
4
collision


overall
5
4
3
2
1
2
3
4
brillouin

B.3.5: Regressing the readability, formality, polarity and wordcount of reviews against overall

Now, we combine all the text metric scores and perform regression to find out which ones are significant predictors of ratings

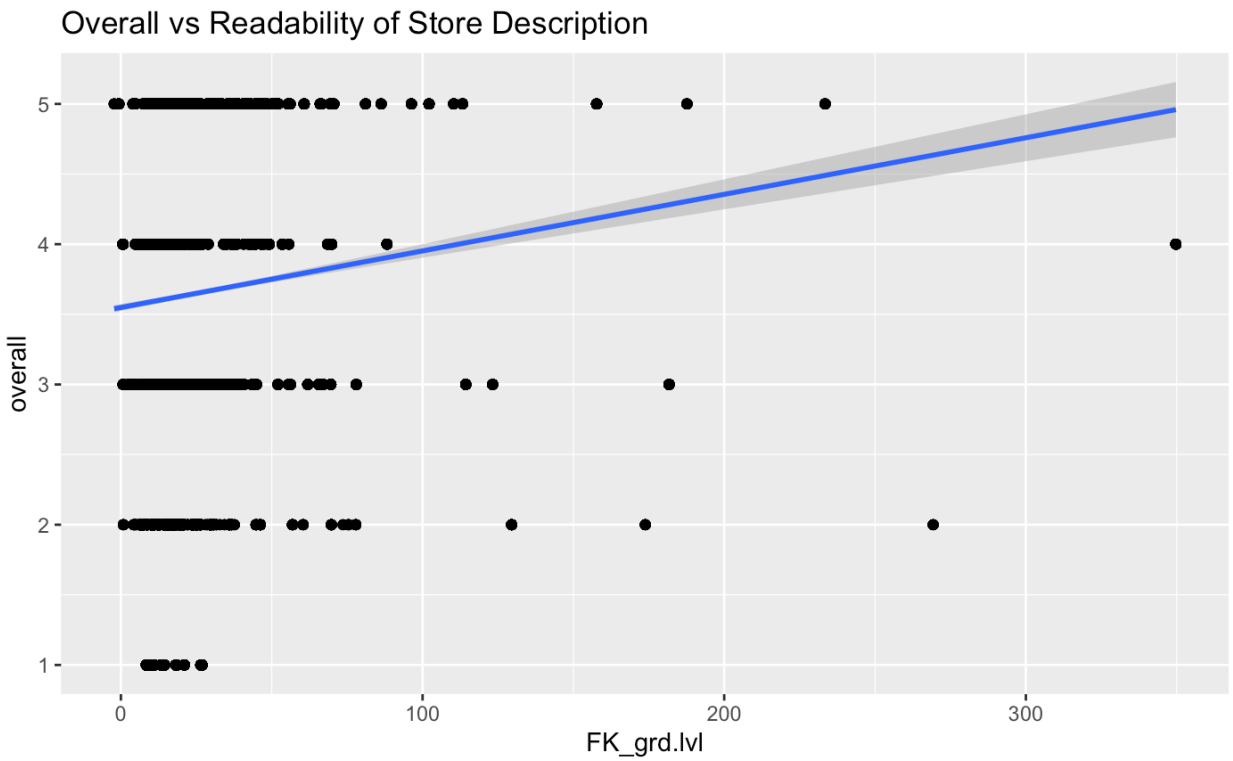
```
   dataset_read_review
model1 <- polr(factor(reviewer_rating) ~ FK_grd.lvl, data = na.omit(dataset_r
ead_review))
model2 <- polr(factor(reviewer_rating) ~ formality, data = na.omit(dataset_re
ad_review))
model3 <- polr(factor(reviewer_rating) ~ polarity, data = na.omit(dataset_rea
d_review))
model4 <- polr(factor(reviewer_rating) ~ shannon, data = na.omit(dataset_read
_review))
model5 <- polr(factor(reviewer_rating) ~ simpson, data = na.omit(dataset_read
_review))
model6 <- polr(factor(reviewer_rating) ~ collision, data = na.omit(dataset_re
ad_review))
model7 <- polr(factor(reviewer_rating) ~ brillouin, data = na.omit(dataset_re
ad_review))
model8 <- polr(factor(reviewer_rating) ~ berger_parker, data = na.omit(datase
t_read_review))

stargazer::stargazer(model1,model2,model3,model4,model5,model6,model7,model8,
type = "text",
                     add.lines = list(c("AIC", round(AIC(model1),1),
                                              round(AIC(model2),1),
                                              round(AIC(model3),1),
                                              round(AIC(model4),1),
                                              round(AIC(model5),1),
                                              round(AIC(model6),1),
                                              round(AIC(model7),1),
                                              round(AIC(model8),1))),
                     out = "features.txt", single.row = T, align = T, flip =
T)
```

Now, we perform the similar analysis on the store descriptions

B.3.6: Checking to see if readability of store description has effect on ratings

```
   all_company <- dataset %>% dplyr::select(company_id, store_description, ov
erall) %>% unique(.)

readability_all_desc <- data.frame()

for(i in 1:nrow(all_company)){

  readability <- data.frame()

  desc <- iconv(all_company$store_description[i])
  desc <- removeNumbers(desc)
  desc <- removePunctuation(desc)

  tryCatch(readability <- flesch_kincaid(desc),error=function(e){
    cat("Error parsing")
  })

  if(!is.null(readability$Readability)){

     readability <- readability$Readability
     readability$company_id <- all_company$company_id[i]
     readability_all_desc <- bind_rows(readability_all_desc,readability)
   }

  print(i)
}

#Adding readability output to dataset
dataset %>% left_join(readability_all_desc) -> dataset_read_desc

readability_desc_plot <- dataset_read_desc %>% group_by(company_id) %>%
  dplyr::select(FK_grd.lvl, overall) %>%
  na.omit() %>%
  ggplot(aes(x = FK_grd.lvl,y = overall)) +
  geom_smooth(method="lm") +
  geom_point() +
  ggtitle("Overall vs Readability of Store Description")
ggsave("readability_desc_plot.png")

#Plotting the relation between the readability of the store description and t
he rating scores
cor.test(dataset_read_desc$overall,
 dataset_read_desc$FK_grd.lvl,
 method="pearson")
   ##
##  Pearson's product-moment correlation
##
```

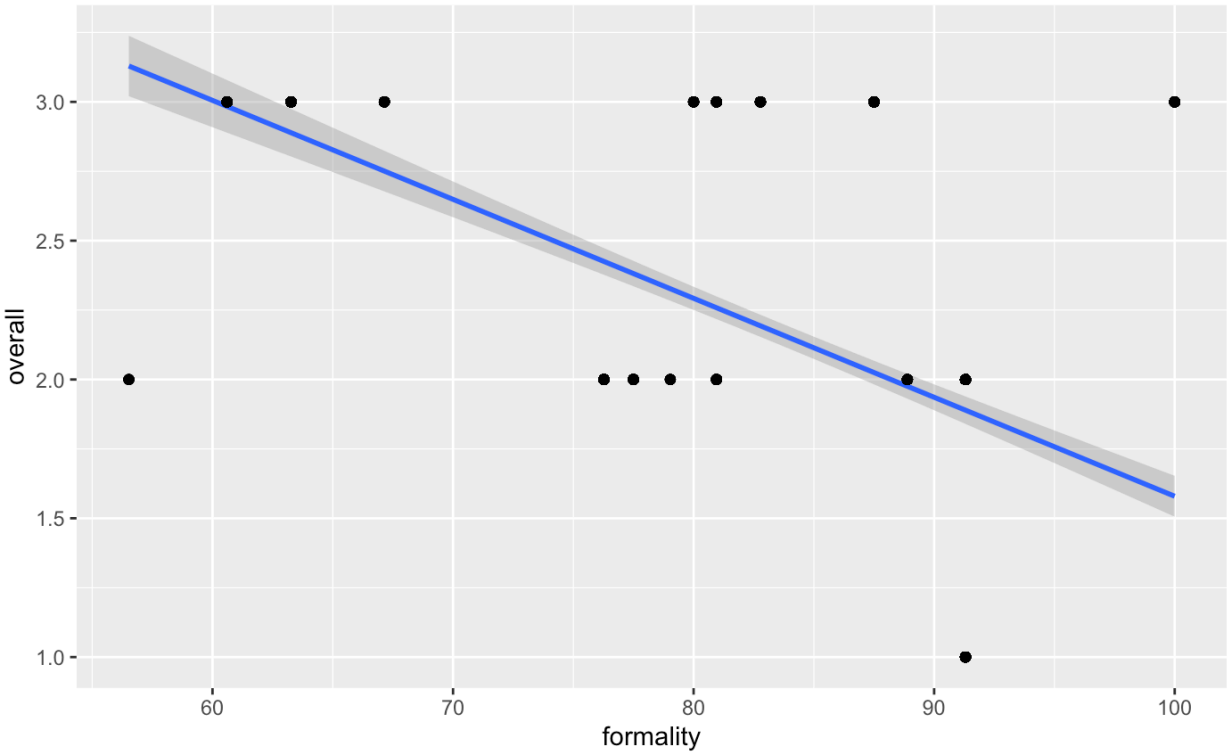
```
## data:  dataset_read_desc$overall and dataset_read_desc$FK_grd.lvl
## t = 13.034, df = 18296, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.08153775 0.11025009
## sample estimates:
##       cor
## 0.09591387
```



B.3.7: Checking to see if formality of store description has effect on ratings

```
   formality_all_desc <- data.frame()

for(i in 1:nrow(all_company)){

  formality <- data.frame()

  desc <- iconv(all_company$store_description[i])
  desc <- removeNumbers(desc)
  desc <- removePunctuation(desc)

  tryCatch(formality <- formality(desc),error=function(e){
    cat("Error parsing")
  })

  if(!is.null(formality$formality)){
```

```
    formality <- formality$formality
    formality$company_id <- all_company$company_id[i]
    formality_all_desc <- bind_rows(formality_all_desc,formality)
   }

  print(i)
}

#Adding formality output to dataset
dataset_read_desc %>% left_join(formality_all_desc) -> dataset_read_desc

formality_desc_plot <- formality_desc_plot <- dataset_read_desc %>% group_by(
company_id) %>%
  dplyr::select(formality, overall) %>%
  na.omit() %>%
  ggplot(aes(x=formality,y=overall))+geom_smooth(method="lm")+geom_point()
ggsave("formality_desc_plot.png")

#Plotting the relation between the formality of the store description and the
 rating scores
cor.test(dataset_read_desc$overall,
 dataset_read_desc$formality,
 method="pearson")
   ##
##  Pearson's product-moment correlation
##
## data:  dataset_read_desc$overall and dataset_read_desc$formality
## t = -24.136, df = 6168, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.3163990 -0.2707975
## sample estimates:
##        cor
## -0.2937654
```

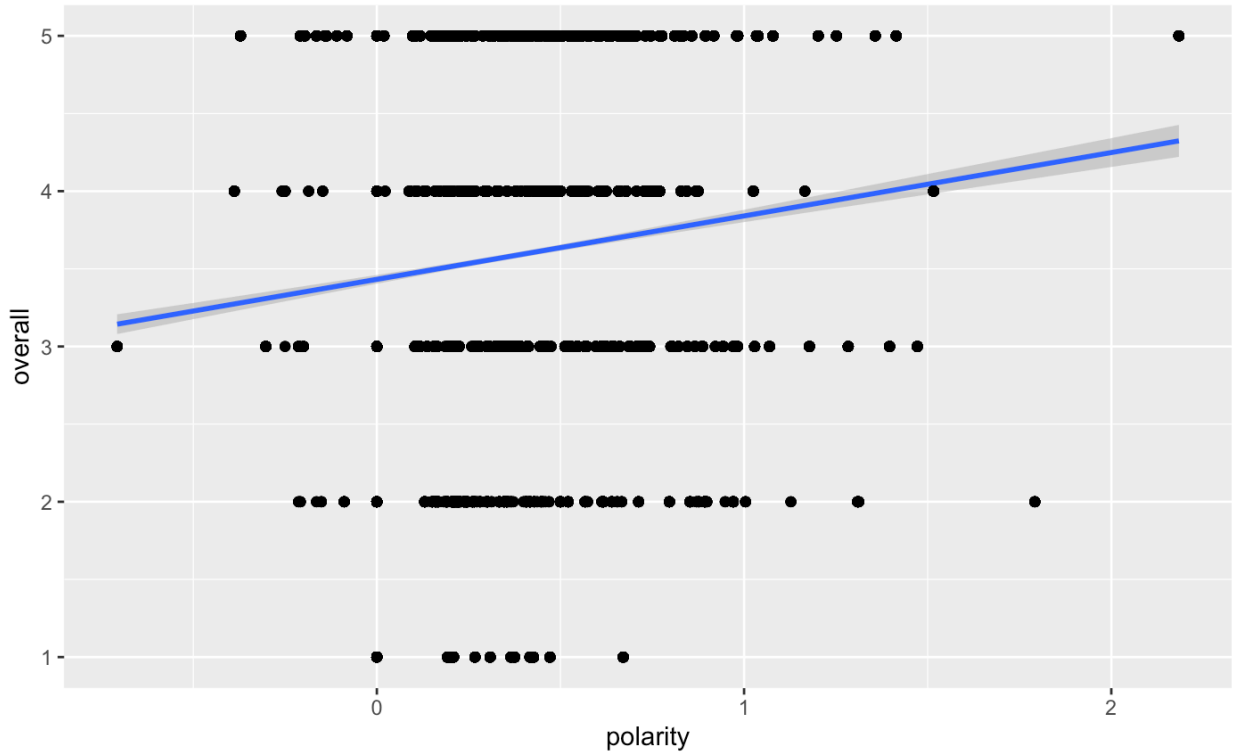


In case of the store descriptions, there seems to be a negative relationship between formality and ratings

B.3.8: Checking to see if polarity of store description has effect on ratings

```
   polarity_all_desc <- data.frame()

for(i in 1:nrow(all_company)){

  polarity <- data.frame()

  desc <- iconv(all_company$store_description[i])
  desc <- removeNumbers(desc)
  desc <- removePunctuation(desc)

  tryCatch(polarity <- polarity(desc),error=function(e){
    cat("Error parsing")
  })

  if(!is.null(polarity$all)){

     polarity <- polarity$all
     polarity$company_id <- all_company$company_id[i]
     polarity_all_desc <- bind_rows(polarity_all_desc,polarity)
   }
```

```
  print(i)
}

#Adding polarity output to dataset
dataset_read_desc %>% left_join(polarity_all_desc) -> dataset_read_desc

polarity_desc_plot <- dataset_read_desc %>% group_by(company_id) %>%
  dplyr::select(polarity, overall) %>%
  na.omit() %>%
  ggplot(aes(x = polarity,y = overall)) + geom_smooth(method="lm") + geom_poi
nt()
ggsave("polarity_desc_plot.png")

#Plotting the relation between the polarity of the store description and the
rating scores
cor.test(dataset_read_desc$overall,
 dataset_read_desc$polarity,
 method="pearson")
   ##
##  Pearson's product-moment correlation
##
## data:  dataset_read_desc$overall and dataset_read_desc$polarity
## t = 14.343, df = 18914, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.08961231 0.11780724
## sample estimates:
##       cor
## 0.1037306
```



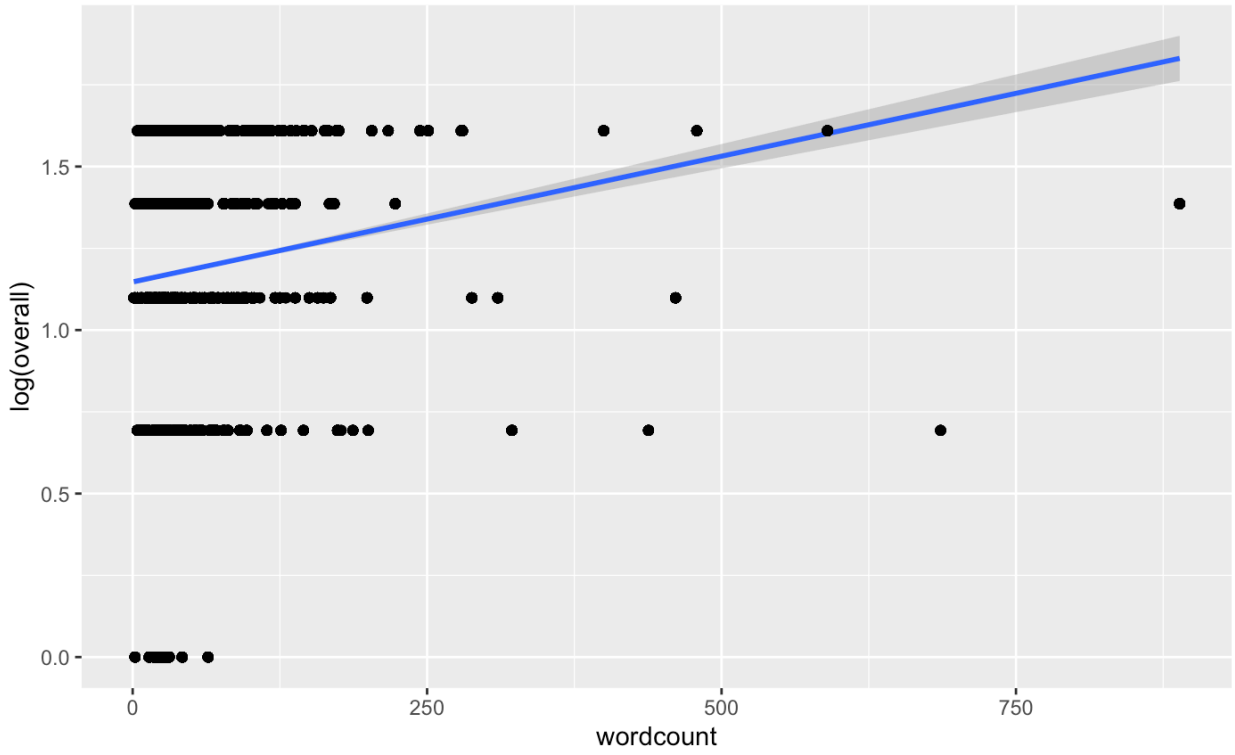
B.3.9: Checking to see if diversity of store description has effect on ratings

```
   diversity_desc <- qdap::diversity(all_company$store_description,all_compan
y$company_id)
diversity_desc$company_id <- as.numeric(diversity_desc$company_id)

#Adding diversity output to dataset
dataset_read_desc %>% left_join(diversity_desc) -> dataset_read_desc

shannon_desc_plot <- dataset_read_desc %>% group_by(company_id) %>%
  dplyr::select(shannon, overall) %>%
  na.omit() %>%
  ggplot(aes(x = shannon,y = overall)) + geom_smooth(method="lm") + geom_poin
t()
ggsave("shannon_desc_plot.png")

simpson_desc_plot <- dataset_read_desc %>% group_by(company_id) %>%
  dplyr::select(simpson, overall) %>%
  na.omit() %>%
  ggplot(aes(x = simpson,y = overall)) + geom_smooth(method="lm") + geom_poin
t()
ggsave("simpson_desc_plot.png")

berger_parker_desc_plot <- dataset_read_desc %>% group_by(company_id) %>%
  dplyr::select(berger_parker, overall) %>%
  na.omit() %>%
```

```
  ggplot(aes(x = berger_parker,y = overall)) + geom_smooth(method="lm") + geo
m_point()
ggsave("berger_parker_desc_plot.png")

collision_desc_plot <- dataset_read_desc %>% group_by(company_id) %>%
  dplyr::select(collision, overall) %>%
  na.omit() %>%
  ggplot(aes(x = collision,y = overall)) + geom_smooth(method="lm") + geom_po
int()
ggsave("collision_desc_plot.png")

brillouin_desc_plot <- dataset_read_desc %>% group_by(company_id) %>%
  dplyr::select(brillouin, overall) %>%
  na.omit() %>%
  ggplot(aes(x = brillouin,y = overall)) + geom_smooth(method="lm") + geom_po
int()
ggsave("brillouin_desc_plot.png")
```

B.3.10: Checking to see if wordcount of store description has effect on ratings

```
   #Adding word count column
dataset_read_desc$wordcount = as.numeric(qdap::wc(dataset_read_desc$store_des
cription, byrow = TRUE))

#Plotting Word Count against price
wordcount_desc_plot <- dataset_read_desc %>% dplyr::select(wordcount,overall)
 %>% na.omit() %>%
  ggplot(aes(x = wordcount,y = log(overall))) + geom_smooth(method = "lm") +
geom_point()
ggsave("wordcount_desc_plot.png")

cor.test(dataset_read_desc$overall,
 dataset_read_desc$wordcount,
 method="pearson")
   ##
##  Pearson's product-moment correlation
##
## data:  dataset_read_desc$overall and dataset_read_desc$wordcount
## t = 16.27, df = 18914, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.1034054 0.1315136
## sample estimates:
##      cor
## 0.117483
```



B.3.11: Regressing the readability, formality, polarity and wordcount of store description against overall

```
   dataset_read_desc
   ## # A tibble: 22,840 × 38
##    name     company_id overall phone_number email store_address     review
er
##    <chr>         <dbl>   <dbl> <chr>        <chr> <chr>             <chr>

##  1 TalkTalk          1       2 "  "         "  "  "  United States" Ian Da
rroch
##  2 TalkTalk          1       2 "  "         "  "  "  United States" Tom

##  3 TalkTalk          1       2 "  "         "  "  "  United States" Anon

##  4 TalkTalk          1       2 "  "         "  "  "  United States" Brian
West
##  5 TalkTalk          1       2 "  "         "  "  "  United States" Sharon
 Tayl…
##  6 TalkTalk          1       2 "  "         "  "  "  United States" Carol
Smith
##  7 TalkTalk          1       2 "  "         "  "  "  United States" Custom
```

```
er PA…
##  8 TalkTalk          1       2 "  "         "  "  "  United States" Enoch

##  9 TalkTalk          1       2 "  "         "  "  "  United States" Michel
le Ti…
## 10 TalkTalk          1       2 "  "         "  "  "  United States" Nick

## # … with 22,830 more rows, and 31 more variables: review_summary <chr>,
## #   review <chr>, year <dbl>, month <dbl>, day <dbl>, country <chr>,
## #   reviewer_rating <dbl>, date <date>, review_id <int>, reviewText <chr>,
## #   char_length <int>, language <chr>, store_description <chr>, all <chr>,
## #   word.count <int>, sentence.count <int>, syllable.count <dbl>,
## #   FK_grd.lvl <dbl>, FK_read.ease <dbl>, formality <dbl>, wc <dbl>,
## #   polarity <dbl>, pos.words <list>, neg.words <list>, text.var <chr>, …
   model9 <- polr(factor(overall) ~ FK_grd.lvl, data = na.omit(dataset_read_d
esc))
model10 <- polr(factor(overall) ~ formality, data = na.omit(dataset_read_desc
))
model11 <- polr(factor(overall) ~ polarity, data = na.omit(dataset_read_desc)
)
model12 <- polr(factor(overall) ~ shannon, data = na.omit(dataset_read_desc))
model13 <- polr(factor(overall) ~ simpson, data = na.omit(dataset_read_desc))
model14 <- polr(factor(overall) ~ collision, data = na.omit(dataset_read_desc
))
model15 <- polr(factor(overall) ~ brillouin, data = na.omit(dataset_read_desc
))
model16 <- polr(factor(overall) ~ berger_parker, data = na.omit(dataset_read_
desc))

stargazer::stargazer(model9,model10,model11,model12,model13,model14,model15,m
odel16,type = "text",
                     add.lines = list(c("AIC", round(AIC(model9),1),
                                              round(AIC(model10),1),
                                              round(AIC(model11),1),
                                              round(AIC(model12),1),
                                              round(AIC(model13),1),
                                              round(AIC(model14),1),
                                              round(AIC(model15),1),
                                              round(AIC(model16),1))),
                     out = "features.txt", single.row = T, align = T, flip =
T)
   ##
## ============================================================================
==========================================================================
=
##                                                                         D
ependent variable:

##               ------------------------------------------------------------
----------------------------------------------------------------------------
```

```
-
##
     overall

##                     (1)              (2)              (3)              (4
)             (5)              (6)              (7)               (8)

## ---------------------------------------------------------------------------
---------------------------------------------------------------------------
-
## FK_grd.lvl    0.015*** (0.001)


## formality                      -0.032*** (0.002)


## polarity                                          0.913*** (0.072)


## shannon                                                                0.838***
 (0.037)

## simpson
         7.882*** (2.710)

## collision
                         0.953*** (0.042)

## brillouin
                                        0.817*** (0.037)

## berger_parker
                                                                -10.433*** (0.490
)
## ---------------------------------------------------------------------------
---------------------------------------------------------------------------
-
## AIC               15996.6           16062.7          16085.7          1573
7.5          16247.1          15720.9          15743.3             15743

## Observations       5,758             5,758            5,758            5,7
58            5,758             5,758            5,758             5,758

## ===========================================================================
===========================================================================
=
## Note:
                                                      *p<0.1; **p<0.05; ***p<0.0
1
```

B.4: Sentiment Analysis

At this stage, we will try to understand the sentiments of the customers from the reviews using various dictionaries.

B.4.1: Checking the number of positive and negative sentiments of reviews

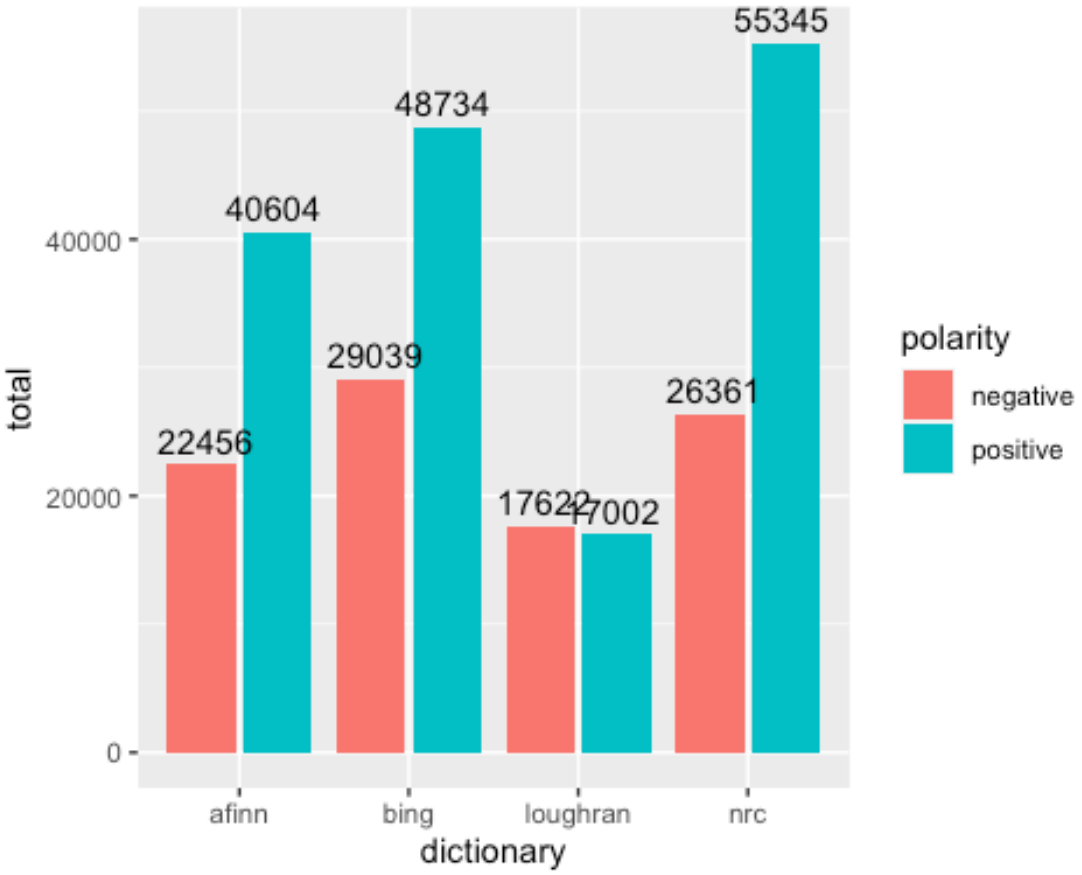
We will start off by importing different dictionaries and getting the number of positive and negative sentiments of each

```
   #Check afinn
#get_afinn <- get_sentiments("afinn")
#saveRDS(get_afinn, "get_afinn.rds")
get_afinn <- readRDS("get_afinn.rds")
afinn<- inner_join(tokens_all_review, get_afinn)

afinn$sentiment<-ifelse(afinn$value>0,"positive",ifelse(afinn$value<0,"negati
ve","neutral"))

afinnsentiments = afinn %>%
  group_by(review_id,sentiment) %>%
  summarise(total=n(), wordcount = mean(wordcount)) %>%
pivot_wider(names_from=sentiment, values_from=total,values_fill=list(total=0)
) %>%
mutate(sentiment = (positive-negative)/(positive+negative))

afinnsentiments
   ## # A tibble: 20,239 × 5
## # Groups:   review_id [20,239]
##    review_id wordcount negative positive sentiment
##        <int>     <dbl>    <int>    <int>     <dbl>
##  1         1        12        1        0    -1
##  2         2        45        4        3    -0.143
##  3         6        72        3        4     0.143
##  4         7        19        1        1     0
##  5         8        20        3        2    -0.2
##  6         9         4        0        3     1
##  7        11        25        2        2     0
##  8        12        16        1        1     0
##  9        13         7        0        1     1
## 10        15        27        5        1    -0.667
## # … with 20,229 more rows

#check bing
get_bing <- readRDS("get_bing.rds")
#get_bing <- get_sentiments("bing")
bing <- inner_join(tokens_all_review,get_bing)
#saveRDS(get_bing, "get_bing.rds")

#sentiment score of bing
```

```r
bingsentiments = bing %>%
  group_by(review_id,sentiment) %>%
  summarise(total=n(), wordcount = mean(wordcount)) %>%
pivot_wider(names_from=sentiment, values_from=total,values_fill=list(total=0)
) %>%
mutate(sentiment = (positive-negative)/(positive+negative))

bingsentiments
   ## # A tibble: 20,988 × 5
## # Groups:   review_id [20,988]
##    review_id wordcount negative positive sentiment
##        <int>     <dbl>    <int>    <int>     <dbl>
##  1         1        12        1        1     0
##  2         2        45        2        3     0.2
##  3         6        72        3        6     0.333
##  4         7        19        3        0    -1
##  5         8        20        3        0    -1
##  6         9         4        0        1     1
##  7        11        25        2        4     0.333
##  8        12        16        1        1     0
##  9        13         7        0        2     1
## 10        15        27        8        0    -1
## # … with 20,978 more rows

#calculate sentiment score of nrc
#get_nrc <- get_sentiments("nrc")
get_nrc <- readRDS("get_nrc.rds")
nrc <- inner_join(tokens_all_review,get_nrc)
#saveRDS(get_nrc, "get_nrc.rds")

nrcsentiments = nrc%>%
  group_by(review_id,sentiment) %>%
  summarise(total=n(), wordcount = mean(wordcount)) %>%
  filter(sentiment %in% c("positive","negative")) %>%
pivot_wider(names_from=sentiment, values_from=total,values_fill=list(total=0)
) %>%
mutate(sentiment = (positive-negative)/(positive+negative))

nrcsentiments
   ## # A tibble: 20,677 × 5
## # Groups:   review_id [20,677]
##    review_id wordcount negative positive sentiment
##        <int>     <dbl>    <int>    <int>     <dbl>
##  1         1        12        1        1    0
##  2         2        45        4        5    0.111
##  3         6        72        6        7    0.0769
##  4         7        19        2        3    0.2
##  5         8        20        3        3    0
##  6         9         4        0        2    1
##  7        11        25        0       11    1
```

```
##  8        12        16        1        1    0
##  9        13         7        0        3    1
## 10        15        27        5        1   -0.667
## # … with 20,667 more rows

#Check loughran
#get_loughran <- get_sentiments("loughran")
get_loughran <- readRDS("get_loughran.rds")
loughran<- inner_join(tokens_all_review,get_loughran)
#saveRDS(get_loughran, "get_loughran.rds")

loughransentiments = loughran %>%
  group_by(review_id,sentiment) %>%
  summarise(total=n(), wordcount = mean(wordcount)) %>%
  filter(sentiment %in% c("positive","negative")) %>%
pivot_wider(names_from=sentiment, values_from=total,values_fill=list(total=0)
) %>%

mutate(sentiment = (positive-negative)/(positive+negative))

loughransentiments
   ## # A tibble: 16,708 × 5
## # Groups:   review_id [16,708]
##    review_id wordcount negative positive sentiment
##        <int>     <dbl>    <int>    <int>     <dbl>
##  1         1        12        1        0    -1
##  2         2        45        2        0    -1
##  3         6        72        4        2    -0.333
##  4         7        19        1        0    -1
##  5         9         4        0        1     1
##  6        11        25        1        0    -1
##  7        15        27        6        0    -1
##  8        16         6        0        1     1
##  9        17        14        1        0    -1
## 10        18        22        1        0    -1
## # … with 16,698 more rows


#calculate the number of positive and negative tokens can be detected by each
 dictionary
afinn_1<-afinn %>%
  group_by(sentiment) %>%
  summarise(total=n())

bing_1<-bing %>%
  filter(sentiment %in% c("positive","negative")) %>%
  group_by(sentiment) %>%
  summarise(total=n())

loughran_1<-loughran%>%
```

```r
  filter(sentiment %in% c("positive","negative")) %>%
  group_by(sentiment) %>%
  summarise(total=n())

nrc_1<-nrc%>%
  filter(sentiment %in% c("positive","negative")) %>%
  group_by(sentiment) %>%
  summarise(total=n())


#plot the total number of positive and negative tokens respect to each dictio
nary
evaluation1<-data.frame(polarity=c('negative','positive'),
            afinn=afinn_1$total,
            bing=bing_1$total,
            loughran=loughran_1$total,
            nrc=nrc_1$total)

evaluation1<-evaluation1%>%
  pivot_longer(!polarity,names_to = "dictionary", values_to ="total")

evaluation1%>%
  group_by(dictionary,polarity)%>%
  ggplot(aes(x=dictionary,y=total))+geom_col(aes(fill=polarity),position = po
sition_dodge2(preserve = 'single'))+geom_text(aes(label=total),position = pos
ition_dodge2(width=0.8,preserve = 'single'),vjust=-0.5,hjust=0.5)
```



Hence, we can see that nrc dictionary is able to capture maximum number of positive and negative sentiments of the reviews. Following it is Bing dictionary. So, these can be considered.

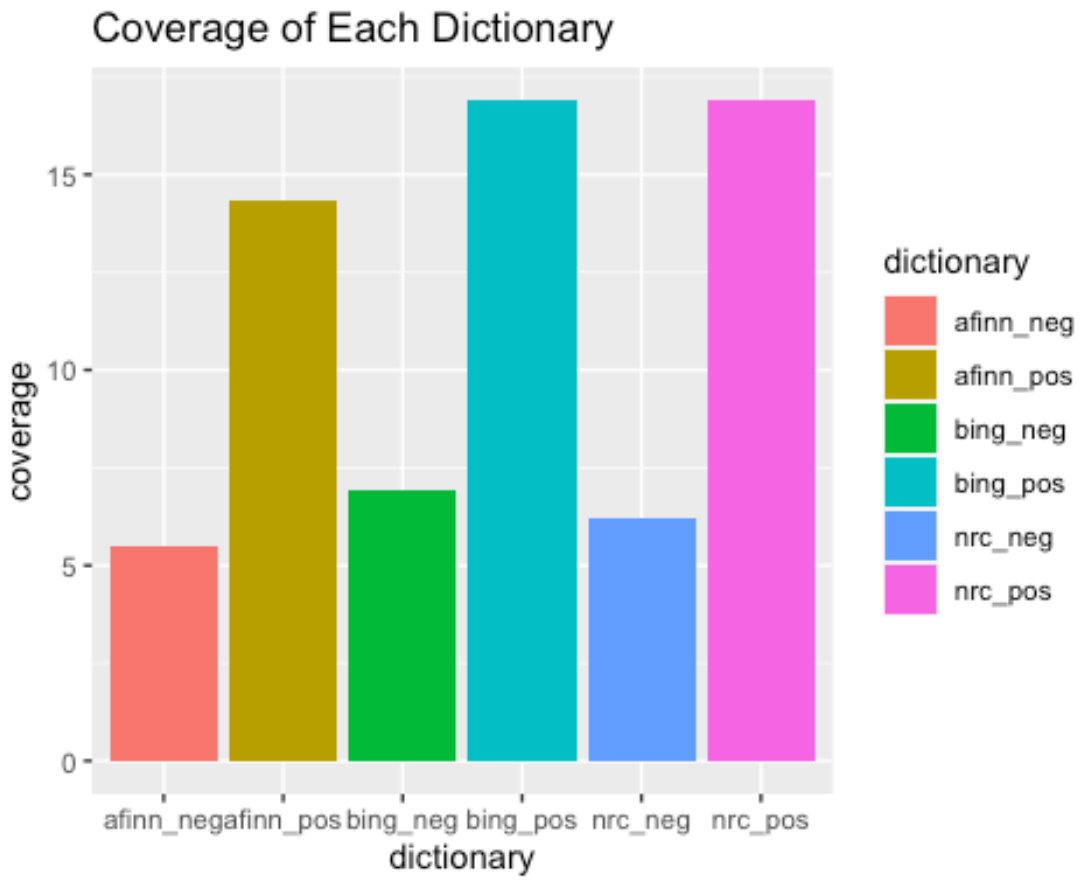
B.4.2: Understanding the dictionary coverages

We will try to analyse how much of the reviews are being covered by the dictionaries

```
   afinnsentiments_coverage<-afinnsentiments%>%
  mutate(afinn_pos_coverage=positive/wordcount*100)%>%
  mutate(afinn_neg_coverage=negative/wordcount*100)%>%
  mutate(afinn_coverage=(negative+positive)/wordcount*100)

bingsentiments_coverage<-bingsentiments%>%
  mutate(bing_pos_coverage=positive/wordcount*100)%>%
  mutate(bing_neg_coverage=negative/wordcount*100)%>%
  mutate(bing_coverage=(negative+positive)/wordcount*100)

nrcsentiments_coverage<-nrcsentiments%>%
  mutate(nrc_pos_coverage=positive/wordcount*100)%>%
  mutate(nrc_neg_coverage=negative/wordcount*100)%>%
  mutate(nrc_coverage=(negative+positive)/wordcount*100)
```

```r
loughransentiments_coverage<-loughransentiments%>%
  mutate(loughran_pos_coverage=positive/wordcount*100)%>%
  mutate(loughran_neg_coverage=negative/wordcount*100)%>%
  mutate(loughran_coverage=(negative+positive)/wordcount*100)

afinn_coverage<-
  afinnsentiments_coverage%>%
  dplyr::select(review_id,afinn_pos_coverage,afinn_neg_coverage,afinn_coverag
e)

bing_coverage<-
  bingsentiments_coverage%>%
  dplyr::select(review_id,bing_pos_coverage,bing_neg_coverage,bing_coverage)

nrc_coverage<-
  nrcsentiments_coverage%>%
  dplyr::select(review_id,nrc_pos_coverage,nrc_neg_coverage,nrc_coverage)

loughran_coverage<-
  loughransentiments_coverage%>%
  dplyr::select(review_id,loughran_pos_coverage,loughran_neg_coverage,loughra
n_coverage)


all_coverage<- afinn_coverage %>% left_join(bing_coverage)
all_coverage<- all_coverage %>% left_join(nrc_coverage)
all_coverage<- all_coverage %>% left_join(loughran_coverage)

all_coverage[is.na(all_coverage)] <- 0

all_coverage<-all_coverage%>%
  dplyr::select(review_id,afinn_pos_coverage,afinn_neg_coverage,afinn_coverag
e,
         bing_pos_coverage,bing_neg_coverage,bing_coverage,
         nrc_pos_coverage,nrc_neg_coverage,nrc_coverage)


dictionary_coverage<-data.frame(cc='coverage',
                        afinn_pos= mean(all_coverage$afinn_pos_coverage),
                        afinn_neg= mean(all_coverage$afinn_neg_coverage),
                       # afinn= mean(All_dictionary_coverage$afinn_coverage),
                        bing_pos= mean(all_coverage$bing_pos_coverage),
                        bing_neg= mean(all_coverage$bing_neg_coverage),
                        #bing= mean(All_dictionary_coverage$bing_coverage),
                        nrc_pos= mean(all_coverage$nrc_pos_coverage),
                        nrc_neg= mean(all_coverage$nrc_neg_coverage))
                       # nrc= mean(All_dictionary_coverage$nrc_coverage)
dictionary_coverage<-dictionary_coverage%>%
  pivot_longer(!cc,names_to = "dictionary", values_to ="coverage")
```

```
dictionary_coverage
   ## # A tibble: 6 × 3
##   cc       dictionary coverage
##   <chr>    <chr>          <dbl>
## 1 coverage afinn_pos      14.3
## 2 coverage afinn_neg       5.51
## 3 coverage bing_pos       16.9
## 4 coverage bing_neg        6.94
## 5 coverage nrc_pos        16.9
## 6 coverage nrc_neg         6.21
   ggplot(dictionary_coverage,aes(x=dictionary,y=coverage))+geom_col(aes(fill
=dictionary))+ggtitle("Coverage of Each Dictionary")
```



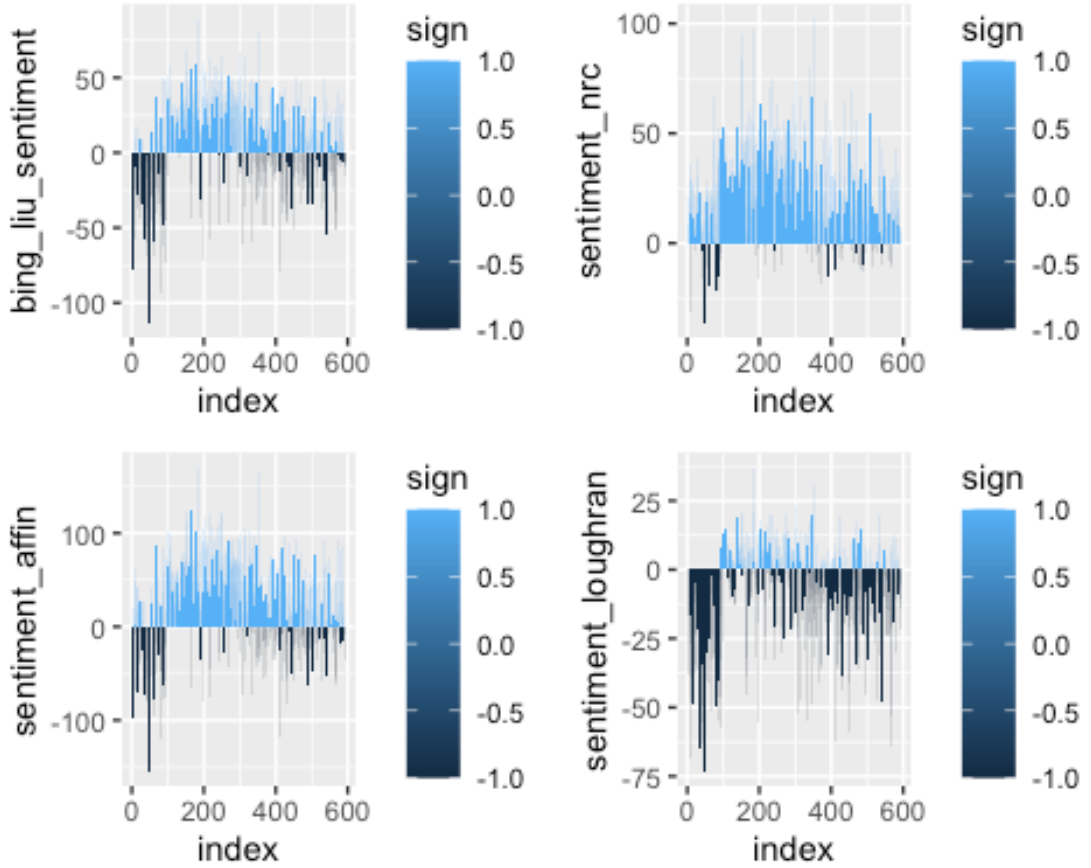
Thus, in terms of coverage, we can also see that nrc and bing both are similarly covering the total sentiments of the reviews.

B.4.3: Sentiment dictionary analysis for store level

We will now use regression analysis to find the best fit dictionary to predict the target variable.

```
   # Getting average reviewer ratings by each stores for better relating the
ratings with the curent dataset
company_id_avg_rating <- dataset %>% group_by(company_id) %>% summarise(avg_r
ating = round(mean(reviewer_rating)))

# Bing Liu - Sentiment Dictionary
tokens_all %>% inner_join(get_bing) %>%
  count(sentiment,company_id) %>% spread(sentiment,n) %>%
  mutate(bing_liu_sentiment = positive-negative) %>%
  dplyr::select(company_id,bing_liu_sentiment) -> bing_liu_sentiment_company

bing_liu_sentiment_company <- bing_liu_sentiment_company %>%
  left_join(company_id_avg_rating)

# NRC Dictionary
tokens_all %>% inner_join(get_nrc) %>%
  count(sentiment,company_id) %>%
  spread(sentiment,n)  -> emotions_nrc

emotions_nrc <- emotions_nrc %>%
  left_join(company_id_avg_rating) %>%
  mutate(sentiment_nrc = positive-negative)

# Afinn Dictionary
tokens_all %>% inner_join(get_afinn) %>%
  group_by(company_id) %>%
  summarise(sentiment_affin = sum(value)) -> sentiment_affin

sentiment_affin <- sentiment_affin %>%
  left_join(company_id_avg_rating)

# Loughran Dictionary
tokens_all %>% inner_join(get_loughran) %>%
  count(sentiment,company_id) %>%
  spread(sentiment,n) -> sentiments_loughran

sentiments_loughran <- sentiments_loughran %>%
  left_join(company_id_avg_rating) %>%
  mutate(sentiment_loughran = positive-negative) %>%
  dplyr::select(company_id,avg_rating,sentiment_loughran)

all_together_sentiments_company <- data.frame()
all_together_sentiments_company <- bing_liu_sentiment_company %>%
  left_join(emotions_nrc)
all_together_sentiments_company <- all_together_sentiments_company %>%
  left_join(sentiment_affin)
```

```r
all_together_sentiments_company <- all_together_sentiments_company %>%
  left_join(sentiments_loughran) %>%
  na.omit()

# Plot the sentiment
p1 <- all_together_sentiments_company %>%
        mutate(index = row_number(), sign = sign(bing_liu_sentiment)) %>%
        ggplot(aes(x=index,y=bing_liu_sentiment, fill = sign))+
        geom_bar(stat="identity")
p2 <- all_together_sentiments_company %>%
        mutate(index = row_number(), sign = sign(sentiment_nrc)) %>%
        ggplot(aes(x=index,y=sentiment_nrc, fill = sign))+
        geom_bar(stat="identity")
p3 <- all_together_sentiments_company %>%
        mutate(index = row_number(), sign = sign(sentiment_affin)) %>%
        ggplot(aes(x=index,y=sentiment_affin, fill = sign))+
        geom_bar(stat="identity")
p4 <- all_together_sentiments_company %>%
        mutate(index = row_number(), sign = sign(sentiment_loughran)) %>%
        ggplot(aes(x=index,y=sentiment_loughran, fill = sign))+
        geom_bar(stat="identity")
ggarrange(p1,p2,p3,p4)
```



```
   # Bing Liu
model17 <- lm(log(avg_rating)~bing_liu_sentiment,
              data=all_together_sentiments_company)

# NRC affection
model18 <- lm(log(avg_rating)~sentiment_nrc,
              data=all_together_sentiments_company)

#Afinn
model19 <- lm(log(avg_rating)~sentiment_affin,
              data=all_together_sentiments_company)
# Loughran
model20 <- lm(log(avg_rating)~sentiment_loughran,
              data=all_together_sentiments_company)

stargazer::stargazer(model17,model18,model19,model20,type = "text")
   ##
## ================================================================================
====
##                                                Dependent variable:

##                                    ---------------------------------------
----
##                                                  log(avg_rating)

##                                       (1)         (2)         (3)         (4)

## --------------------------------------------------------------------------
----
## bing_liu_sentiment                  0.015***

##                                     (0.0004)

##

## sentiment_nrc                                    0.018***

##                                                  (0.001)

##

## sentiment_affin                                                 0.009***

##                                                                 (0.0003)

##

## sentiment_loughran                                                          0.022
***
##                                                                             (0.00
```

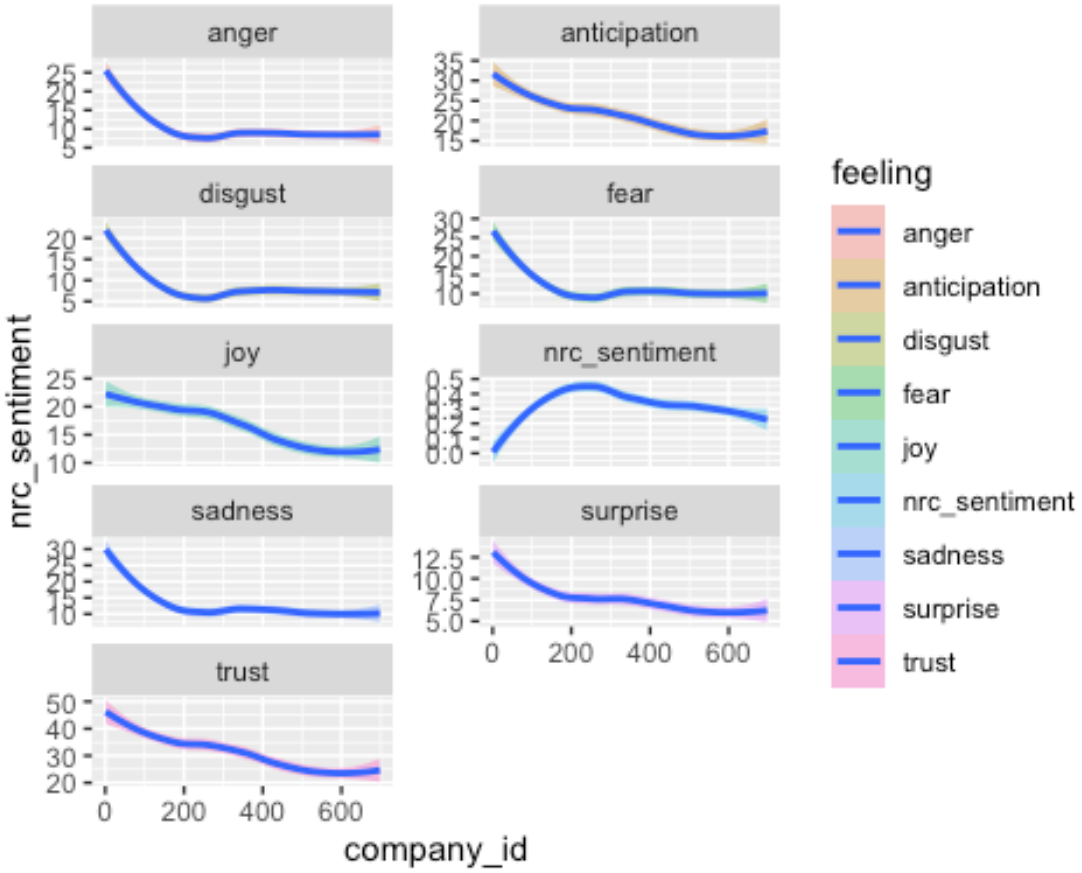
```
1)
##

## Constant                          1.069***    0.798***    0.946***    1.342
***
##                                   (0.013)     (0.023)     (0.014)     (0.01
6)
##

## ------------------------------------------------------------------------
----
## Observations                        591         591         591         591

## R2                                 0.653       0.444       0.676       0.52
2
## Adjusted R2                        0.652       0.443       0.675       0.52
1
## Residual Std. Error (df = 589)     0.307       0.389       0.297       0.36
0
## F Statistic (df = 1; 589)      1,106.092*** 470.655*** 1,227.796*** 642.45
4***
## ========================================================================
====
## Note:                                                *p<0.1; **p<0.05; ***p<
0.01
```

It can be seen after regression that all the dictionaries are significant in predicting the ratings of the reviews. However, looking at the adjusted R2 value is the highest for the Afinn dictionary, followed by the Bing dictionary. Hence, these are the better models that are fit to the dataset as per the outcome of the analysis.

B.4.4: Extract feelings from NRC dictionary

The NRC dictionary provides the affection levels of the text as well. We will analyse the different affection categories and perform regression with them

```
   nrc_feelings <- tokens_all %>%
                inner_join(get_nrc) %>%
                count(sentiment,company_id) %>%
                spread(sentiment,n) %>%
                mutate(nrc_sentiment = (positive-negative)/(positive+negative
)) %>%
                dplyr::select(-c(positive,negative)) %>%
                pivot_longer(anger:nrc_sentiment,names_to = "feeling",values_
to="nrc_sentiment")

# Plot the feelings extracted
nrc_feelings %>% ggplot(aes(x=company_id,y=nrc_sentiment,fill=feeling))+
                 geom_smooth()+
                 facet_wrap(~feeling,scales="free_y",ncol=2)
```




```
   #Running regressions for nrc feelings
nrc_feelings_model = nrc_feelings %>% spread(feeling, nrc_sentiment, fill = 0
)
nrc_feelings_model = left_join(nrc_feelings_model, company_id_avg_rating)
nrc_feelings_model$avg_rating = as.numeric(nrc_feelings_model$avg_rating)

model21 <- lm(log(avg_rating)~.-company_id, data=nrc_feelings_model)
stargazer::stargazer(model21, type = "text")
   ##
## ================================================
##                         Dependent variable:
##                     ---------------------------
##                           log(avg_rating)
## ------------------------------------------------
## anger                          -0.011**
##                                 (0.006)
##
## anticipation                    -0.006*
##                                 (0.004)
##
## disgust                        -0.023***
##                                 (0.005)
##
```

```
## fear                            -0.009**
##                                  (0.005)
##
## joy                             0.027***
##                                  (0.004)
##
## nrc_sentiment                   0.790***
##                                  (0.068)
##
## sadness                         0.014***
##                                  (0.004)
##
## surprise                         -0.004
##                                  (0.005)
##
## trust                            -0.0003
##                                  (0.002)
##
## Constant                        0.893***
##                                  (0.033)
##
## ------------------------------------------------
## Observations                       695
## R2                                0.640
## Adjusted R2                       0.635
## Residual Std. Error      0.311 (df = 685)
## F Statistic          135.346*** (df = 9; 685)
## ================================================
## Note:                *p<0.1; **p<0.05; ***p<0.01
```

The analysis shows that of all the feelings available in the nrc dictionary, surprise and trust feelings do not have any significant effect for the reviews. Other than that, all the remaining feelings seem to have significant effects.

B.4.5: Syntactic Feature Analysis

For syntactic features, we will use exclamation marks and capital letter combine them with different sentiments in regression models.

```
# Calculate the number of exclamation marks in each store
syntactical_info_excl <-  dataset %>% left_join(company_id_avg_rating) %>%
                          left_join(all_together_sentiments_company) %>%
                          mutate(excl_count = str_count(reviewText, "!")) %>%
                          group_by(company_id) %>%
                          mutate(total_excl_count = sum(excl_count),
                                 exl_prop = total_excl_count/char_length)

model22 <- lm(log(avg_rating)~total_excl_count,data = syntactical_info_excl)
model23 <- lm(log(avg_rating)~exl_prop,data = syntactical_info_excl)

#Calculating the number of capital letter words in each store
syntactical_info_caps <-  dataset %>% left_join(company_id_avg_rating) %>%
                          left_join(all_together_sentiments_company) %>%
                          mutate(cap_count = str_count(reviewText, "[[:upper:
]]")) %>%
                          group_by(company_id) %>%
                          mutate(total_cap_count = sum(cap_count),
                                 cap_prop = total_cap_count/char_length)

model24 <- lm(log(avg_rating)~total_cap_count,data = syntactical_info_caps)
model25 <- lm(log(avg_rating)~cap_prop,data = syntactical_info_caps)

stargazer::stargazer(model22,model23,model24,model25,type = "text")
   ##
## ============================================================================
============================
##                                                  Dependent variable
:
##                              ---------------------------------------------
---------------------------
##                                                    log(avg_rating)

##                                    (1)               (2)               (
3)               (4)
## ----------------------------------------------------------------------------
---------------------------
## total_excl_count

##

##

## exl_prop
```

```
##

##

## total_cap_count                                                    -0.0
01***
##                                                                    (0.0
0001)
##

## cap_prop
               -0.077***
##
                (0.002)
##

## Constant                          1.201***           1.201***           1.6
31***           1.359***
##                                   (0.003)            (0.003)            (0.
005)            (0.005)
##

## ---------------------------------------------------------------------------
----------------------------
## Observations                       22,119             22,119             22
,119             22,119
## R2                                 0.000              0.000              0.
283              0.060
## Adjusted R2                        0.000              0.000              0.
283              0.060
## Residual Std. Error         0.518 (df = 22118) 0.518 (df = 22118) 0.439 (d
f = 22117) 0.502 (df = 22117)
## F Statistic (df = 1; 22117)                                           8,713
.212***       1,410.830***
## ===========================================================================
============================
## Note:
  *p<0.1; **p<0.05; ***p<0.01
```

```r
#Regressing exclamations with sentiment
model26 <- lm(bing_liu_sentiment~total_excl_count, data = syntactical_info_ex
cl)
model27 <- lm(sentiment_nrc~total_excl_count, data = syntactical_info_excl)
model28 <- lm(sentiment_affin~total_excl_count, data = syntactical_info_excl)
model29 <- lm(sentiment_loughran~total_excl_count, data = syntactical_info_ex
cl)

stargazer::stargazer(model26,model27,model28,model29,type = "text")
   ##
## ================================================================================
==========================
##                                                          Dependent variabl
e:
##                                 ---------------------------------------
-------------------------
##                                 bing_liu_sentiment sentiment_nrc sentimen
t_affin sentiment_loughran
##                                        (1)             (2)            (3
)              (4)
## --------------------------------------------------------------------------------
-------------------------
## total_excl_count

##

##

## Constant                                6.150***        23.656***      25.35
2***        -10.425***
##                                        (0.225)          (0.144)        (0.3
53)          (0.136)
##

## --------------------------------------------------------------------------------
-------------------------
## Observations                             20,873          20,873         20,8
73            20,873
## R2                                       0.000            0.000          0.0
00            0.000
## Adjusted R2                              0.000            0.000          0.0
00            0.000
## Residual Std. Error (df = 20872)         32.562          20.852         51.0
67            19.713
## ================================================================================
==========================
## Note:                                                                  *
p<0.1; **p<0.05; ***p<0.01
```

```
#Regressing capital letters with sentiment
model30 <- lm(bing_liu_sentiment~total_cap_count, data = syntactical_info_cap
s)
model31 <- lm(sentiment_nrc~total_cap_count, data = syntactical_info_caps)
model32 <- lm(sentiment_affin~total_cap_count, data = syntactical_info_caps)
model33 <- lm(sentiment_loughran~total_cap_count, data = syntactical_info_cap
s)

stargazer::stargazer(model30,model31,model32,model33,type = "text")
   ##
## ================================================================================
==========================
##                                                         Dependent variabl
e:
##                                 ---------------------------------------
-------------------------
##                                bing_liu_sentiment sentiment_nrc sentimen
t_affin sentiment_loughran
##                                       (1)             (2)            (3
)              (4)
## --------------------------------------------------------------------------------
-------------------------
## total_cap_count                    -0.053***       -0.014***      -0.07
4***        -0.039***
##                                     (0.001)         (0.0004)        (0.0
01)          (0.0003)
##
##
## Constant                           35.174***       31.216***      65.59
2***        10.812***
##                                     (0.363)          (0.270)        (0.5
95)          (0.197)
##
##
## --------------------------------------------------------------------------------
-------------------------
## Observations                         20,873          20,873         20,8
73            20,873
## R2                                   0.296            0.049          0.2
31            0.433
## Adjusted R2                          0.296            0.049          0.2
31            0.432
## Residual Std. Error (df = 20871)     27.320          20.336         44.7
72            14.851
## F Statistic (df = 1; 20871)       8,777.683***    1,075.174***   6,282.7
64***     15,906.570***
## ================================================================================
==========================
## Note:                                                                 *
p<0.1; **p<0.05; ***p<0.01
```

Part C:

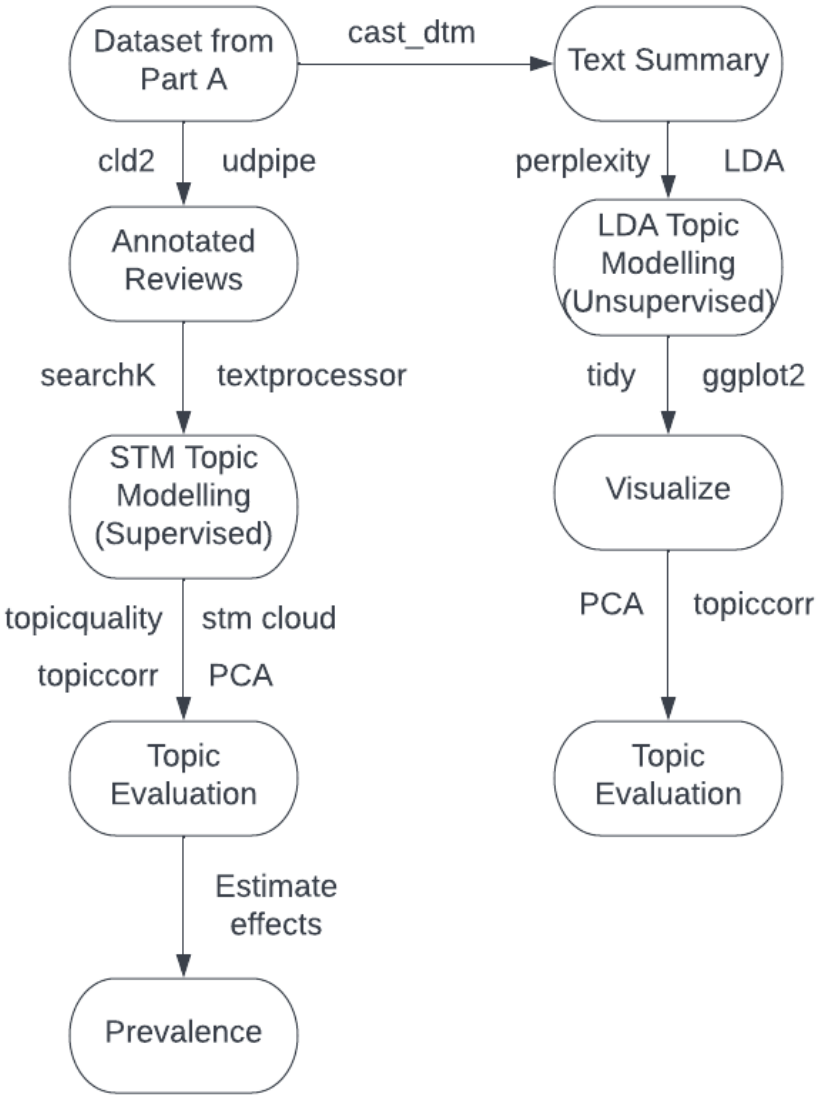
C.1: Section Plan

In this section, we will perform topic modelling using the Latent Dirichlet allocation method.

We will initially perform an unsupervised topic modelling approach and then compare it with a supervised approach.

Upon analysis, we will try to identify the optimum kappa value for the number of topics to be selected. Using the kappa value, we will identify the topics using semantic coherence and exclusivity. Finally, we will try to associate the effects of the topics and metadata with the ratings. We will also try to sense the importance of topics over regions and years.

For this section, the following pipeline would be observed:



C.2: Data Preparation

- Loading relevant libraries

```
library(tidyverse)
library(dplyr)
library(qdap)
library(stringr)
library(tidytext)
library(cld2)
library(tm)
library(textmineR)
library(textclean)
library(textstem)
library(wordcloud)
library(reshape2)
library(topicmodels)
library(lubridate)
library(stm)
```

- Loading the dataset from previous section

```
#Importing the dataset
dataset <- readRDS("review_data_en.rds")
dataset_stores <- readRDS("review_data_by_stores.rds")
```

- Cleaning the reviews

We will now do some text cleaning that will be necessary for this section

```
#Checking the structure of the dataset
str(dataset)
## tibble [22,119 × 20] (S3: tbl_df/tbl/data.frame)
##  $ name             : chr [1:22119] "TalkTalk" "TalkTalk" "TalkTalk" "Talk
Talk" ...
##  $ company_id       : int [1:22119] 1 1 1 1 1 1 1 1 1 1 ...
##  $ overall          : num [1:22119] 2 2 2 2 2 2 2 2 2 2 ...
##  $ phone_number     : chr [1:22119] "  " "  " "  " "  " ...
##  $ email            : chr [1:22119] "  " "  " "  " "  " ...
##  $ store_address    : chr [1:22119] "  United States" "  United States" "
 United States" "  United States" ...
##  $ reviewer         : chr [1:22119] "Ian Darroch" "Tom" "Anon" "Brian West
"  ...
##  $ review_summary   : chr [1:22119] "Poor service!!!" "2 years ago I saw a
 deal" "I had taken out a new account after my…" "Frustration" ...
##  $ review           : chr [1:22119] "I made an over- payment to my account
 and phoned up about it.Although the guy was telling me I would be getting"|
```

```
__truncated__ "2 years ago I saw a deal. 1 year of Amazon and 2 years broadba
nd. I signed up. I didn’t see the small print tha"| __truncated__ "I had take
n out a new account after my late parents' account had to be closed and had a
 price of £27.50 agreed."| __truncated__ "FrustrationI have gone down the on-
line route because I have got nowhere over the last month on the telephone. "
| __truncated__ ...
##  $ year             : num [1:22119] 2022 2022 2022 2022 2022 ...
##  $ month            : num [1:22119] 4 4 4 4 4 4 4 4 4 4 ...
##  $ day              : num [1:22119] 14 13 13 13 13 13 13 13 13 13 ...
##  $ country          : chr [1:22119] "GB" "GB" "GB" "GB" ...
##  $ reviewer_rating  : num [1:22119] 1 1 2 1 1 5 5 1 5 1 ...
##  $ date             : Date[1:22119], format: "2022-04-14" "2022-04-13" ...
##  $ review_id        : int [1:22119] 1 2 6 7 8 9 11 12 13 15 ...
##  $ reviewText       : chr [1:22119] "Poor service I made an over payment t
o my account and phoned up about it Although the guy was telling me I woul"|
__truncated__ "years ago I saw a deal years ago I saw a deal year of Amazon a
nd years broadband I signed up I didn t see the s"| __truncated__ "I had take
n out a new account after my I had taken out a new account after my late pare
nts account had to be cl"| __truncated__ "Frustration FrustrationI have gone
down the on line route because I have got nowhere over the last month on the"
| __truncated__ ...
##  $ char_length      : int [1:22119] 315 842 1341 399 466 110 386 283 141 5
15 ...
##  $ language         : chr [1:22119] "en" "en" "en" "en" ...
##  $ store_description: chr [1:22119] "the uk s lead value for money telecom
 company" "the uk s lead value for money telecom company" "the uk s lead valu
e for money telecom company" "the uk s lead value for money telecom company"
...
```

```r
#data cleaning
dataset$reviewText <- iconv(dataset$reviewText)
dataset$reviewText <- replace_non_ascii(dataset$reviewText) #replacing common
 non-ascii words
dataset$reviewText <- gsub("[ |\t]+", " ", dataset$reviewText) #removing whit
e space
dataset$reviewText <- gsub("[^\x01-\x7F]", "", dataset$reviewText) #removing
emoticons
dataset$reviewText <- gsub("<.*>", "", dataset$reviewText) #removing emojis
dataset$reviewText <- lemmatize_words(dataset$reviewText) #lemmatizing words
dataset$reviewText <- gsub("[ |\t]+", " ", dataset$reviewText) #removing whit
e space
```

C.3: LDA Topic modelling approach (Unsupervised)

Here, we will perform unsupervised topic modelling. We will try to identify the topics without taking the prevalence target variable into account.

We are going to use the manual LDA method for this approach. However, the following method could also be used to do unsupervised modelling. We will use the supervised version of the similar code later.

```
{
    reviews_fit_11 <- stm(documents = outdocuments,
                          vocab = outvocab,
                          K = 0,
                          max.em.its = 75,
                          data = out$meta,
                          sigma.prior = 0.5,
                          init.type = "Spectral") #to allow the function to find the
                      optimum number of topics
}
```

For the unsupervised approach, we will group all the individual reviews into one company review to find the topics. This should allow us to explore the topics without any effect of review levels or ratings.

C.3.1: LDA Topic Modelling

- Casting a dtm
-

```
   LDA_custom_words <- c("ve", "don", "service", "company", "customer", "busi
ness", "product")
tibble(word = LDA_custom_words,lexicon = rep("custom",length(LDA_custom_words
))) %>% bind_rows(stop_words) -> LDA_custom_stopwords

topic_dtm <- dataset_stores %>%
             unnest_tokens(word,grouped_reviews) %>% anti_join(LDA_custom_sto
pwords) %>%
             count(company_id,word) %>% cast_dtm(company_id,word,n)
inspect(topic_dtm)
   ## <<DocumentTermMatrix (documents: 695, terms: 21783)>>
## Non-/sparse entries: 237635/14901550
## Sparsity           : 98%
## Maximal term length: 45
## Weighting          : term frequency (tf)
## Sample             :
##       Terms
## Docs  day days delivery excellent helpful money phone received recommend t
ime
##   3     6   14        1         1       0     9    35        9         5
 29
##   36    8   13        0         1       4     1    13        3         0
 28
##   361   0   10       20        20      24     8     8       18        20
 16
##   44   10    5        0         1       6    10    13        4         4
  8
##   51   18    9        2         1       5    10     1        3         9
 11
##   639  28   24       65         4       4     8    12       15         2
 21
##   67    6    5        4         4       3     9     9        8         4
 22
##   78   22   21       24         0       2    22    70       16         2
 17
##   92    5   14       20         4       2    13    98       16         3
 11
##   93    7   13        5         5       3     7     9        6         6
  8
   topic_dtm_sparse <- tm::removeSparseTerms(topic_dtm,sparse = 0.68)
inspect(topic_dtm_sparse)
   ## <<DocumentTermMatrix (documents: 695, terms: 156)>>
## Non-/sparse entries: 50878/57542
## Sparsity           : 53%
## Maximal term length: 13
```

```
## Weighting          : term frequency (tf)
## Sample             :
##      Terms
## Docs  day days delivery excellent helpful money phone received recommend t
ime
##   203   5   25        8         0       3    20     5       23         4
 11
##   26    3   27       22         9       3    16     5       15         3
  8
##   3     6   14        1         1       0     9    35        9         5
 29
##   361   0   10       20        20      24     8     8       18        20
 16
##   412   7   21       10         3       1    22     5       43         2
  3
##   48   15   23       31         1       0    12    17       20         3
 15
##   5    13   29       24         3       0    15    39       28         5
 19
##   621  15   10        6         0       0    17   113       33         1
  3
##   639  28   24       65         4       4     8    12       15         2
 21
##   92    5   14       20         4       2    13    98       16         3
 11
```

- Creating an LDA model

We will initiate by choosing a random kappa value

```
   #Creating model with random kappa value for evaluation
topic_model <- LDA(topic_dtm, k = 2, method = "Gibbs")
topic_model_sparse <- LDA(topic_dtm_sparse, k = 2, method = "Gibbs")
```

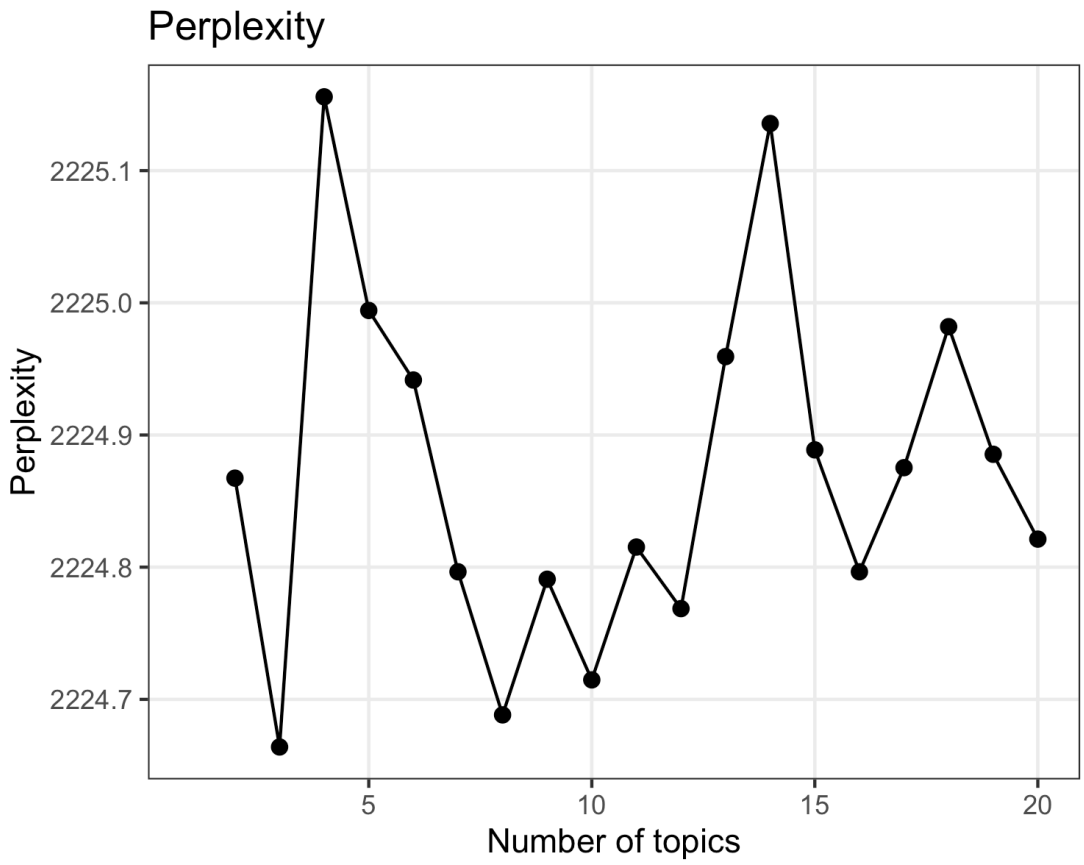
C.3.2: Kappa (K) Evaluation

- For kappa selection

Now that we have got a dummy model, we search for an optimum value to k by using perplexity values. We will aim for the lowest perplexity value excluding the terminal ends.

```
   set.seed(123)
#Setting a number range
topics <- c(2:20)

#Getting perplexity values
perplexity_df <- data.frame(perp_value=numeric())
for (i in topics){
  fitted <- LDA(topic_dtm, k = i, method = "Gibbs")
```

```
  perplexity_df[i,1] <- perplexity(topic_model,topic_dtm)
}

#Plotting the results
perplexity_plot <- ggplot(data=perplexity_df,
       aes(x=as.numeric(row.names(perplexity_df)))) +
  labs(y="Perplexity",x="Number of topics", title="Perplexity") +
  geom_line(aes(y=perp_value), colour="black") +
  geom_point(aes(y=perp_value),size=2) +
  theme_bw() +
  theme(panel.grid.minor = element_blank())
ggsave("perplexity.png")
```



From the graph, we see that perplexity is lowest at k = 8. We will now search for 12 topics and go for 8 topics for the sparse dtm

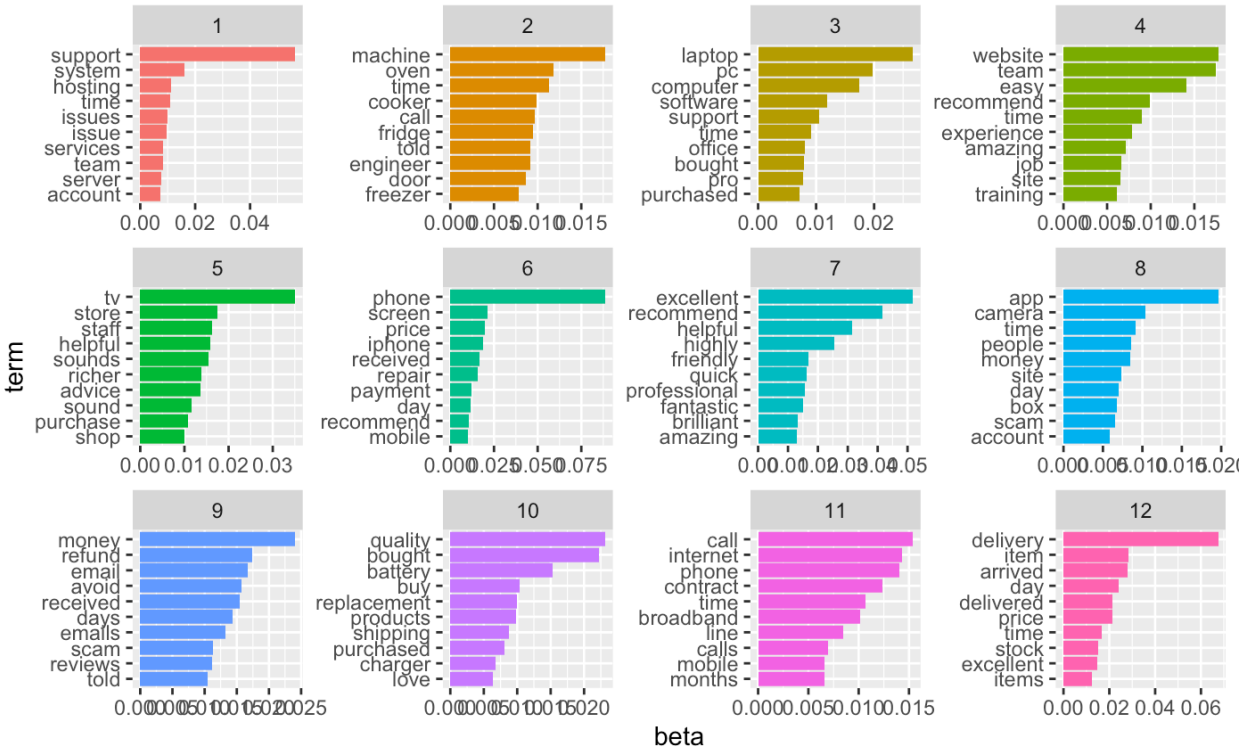
- Topic modelling for k = 12

```
  #Creating the model
topic_model_12 <- LDA(topic_dtm, k = 12, method = "Gibbs", control = list(see
d = 1234))
topic_model_8_sparse <- LDA(topic_dtm_sparse, k = 8, method = "Gibbs", contro
l = list(seed = 123))
```

```
#Tidying the output
lda_12_topics_beta <- tidy(topic_model_12, matrix = "beta")
lda_8_topics_beta_sparse <- tidy(topic_model_8_sparse, matrix = "beta")
```

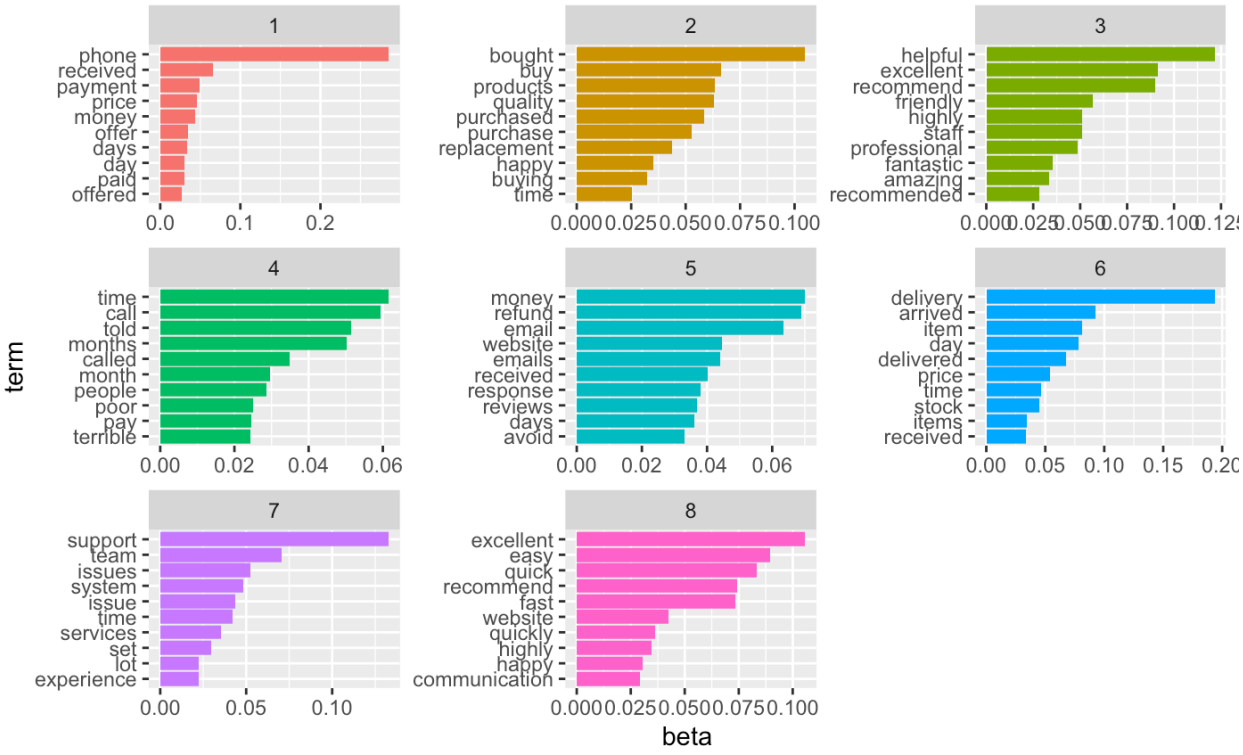
C.3.3: Visualization of Topics

```
   #Getting the top terms for each topic
lda_12_beta_top_terms <- lda_12_topics_beta %>%
  group_by(topic) %>%
  slice_max(beta, n = 10) %>%
  ungroup() %>%
  arrange(topic, -beta)

lda_8_beta_top_terms_sparse <- lda_8_topics_beta_sparse %>%
  group_by(topic) %>%
  slice_max(beta, n = 10) %>%
  ungroup() %>%
  arrange(topic, -beta)

#Plotting the topic words
lda_12_beta_top_terms %>%
  mutate(term = reorder_within(term, beta, topic)) %>%
  ggplot(aes(beta, term, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  scale_y_reordered()
```



From the plot, we can observe the high probable words for each topic. In the 12 topic model, we can see that most of the topics make sense, however, some topics do not seem to have good coherence. For example, it is hard to figure out the topic from the bar plot of topic 2.

```
  lda_8_beta_top_terms_sparse %>%
  mutate(term = reorder_within(term, beta, topic)) %>%
  ggplot(aes(beta, term, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  scale_y_reordered()
```



From the 8-topic model, we can see that due to the number of topics calculated being fewer, the topics make better sense. It seems to be a better model due to the dtm used has less sparsity.

Thus, based on the plot, we will now understand the topic labels that can be used for the word sets.

```
   topic_labels <- c("phone_payment", "product_quality", "friendly_staff", "c
all_issues", "refunds", "item_delivery", "customer_support", "website_quality
")
```

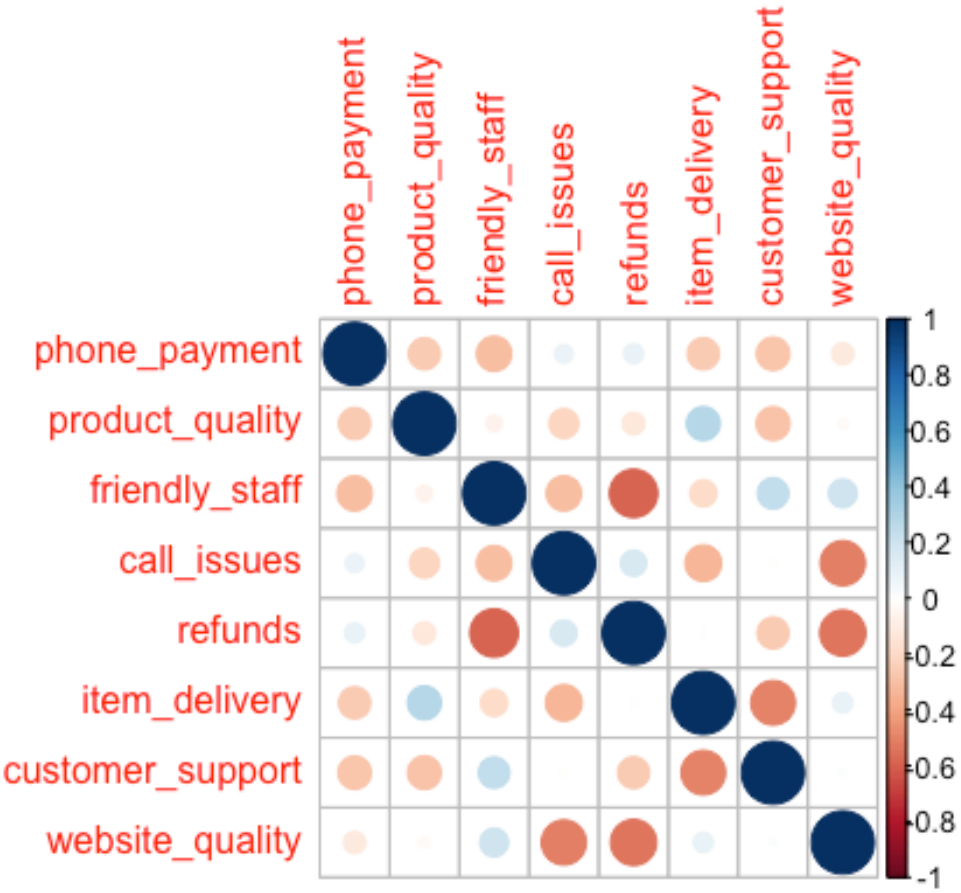
C.3.4: Evaluation of Topics

- Getting the document level values(gamma)

```
   #Tidying the model
gamma_topics <- tidy(topic_model_8_sparse,matrix="gamma")
gamma_topics <- gamma_topics %>%
                pivot_wider(names_from = topic, values_from = gamma)

#Setting the column names
colnames(gamma_topics) <- c("document",topic_labels)

#Removing the document from the column
rownames(gamma_topics) <- gamma_topics$document
gamma_topics$document <- NULL
```

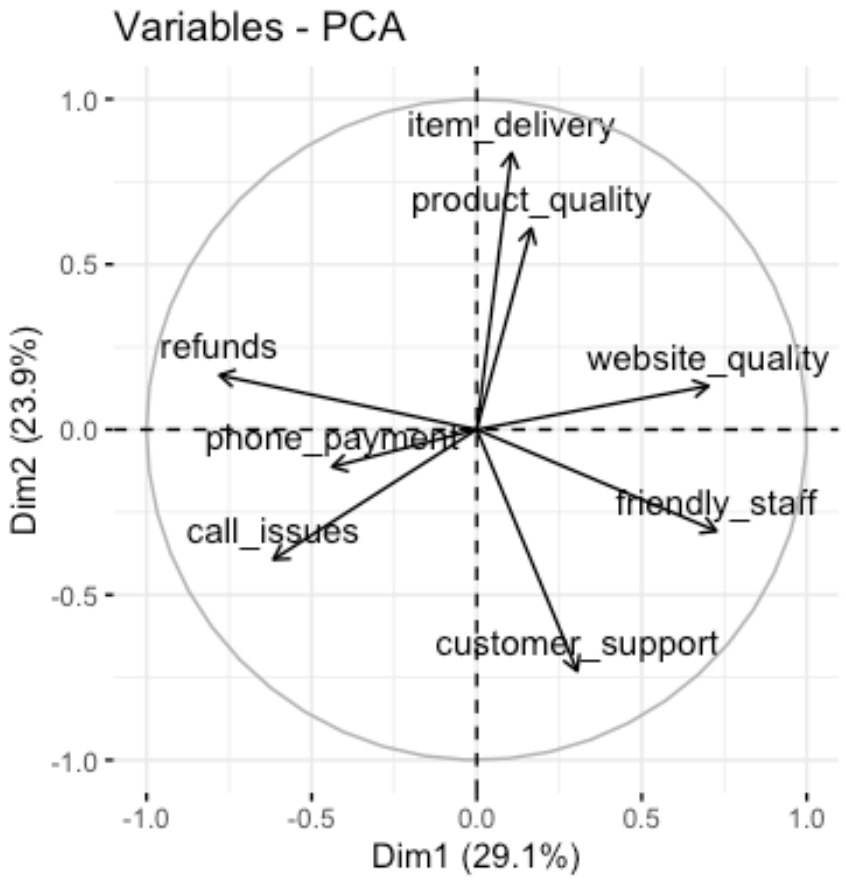
- Plotting topic relations

```
   #Topic correlations
corrplot::corrplot(cor(gamma_topics))
```



From the plot, we can observe that there is higher correlation between topics, such as refunds and friendly staff or refunds and website quality. These signal which topics often appear together.

```
   #Factor relation
pcah <- FactoMineR::PCA(gamma_topics,graph = FALSE)
factoextra::fviz_pca_var(pcah)
```



From the PCA graph, we can observe the directions of variance explained by the factor components and that explains whether multi-collinearity is present between the topics. In graph, we can see that the first two factors explain 53% of the variations present in the dataset.

C.3.5: Bigram topics

- Checking topics for bigrams

```
   topic_bi_dtm <- dataset_stores %>%
                unnest_tokens(word,grouped_reviews, token = "ngrams", n = 2)
%>%
                separate(word, c("word1", "word2"), sep = " ") %>%
                filter(!word1 %in% LDA_custom_stopwords$word) %>%
                filter(!word2 %in% LDA_custom_stopwords$word) %>%
                unite("word", word1, word2, sep = " ") %>%
                count(company_id,word) %>% cast_dtm(company_id,word,n)
   #Creating the model
topic_model_bi <- LDA(topic_bi_dtm, k = 8, method = "Gibbs", control = list(s
eed = 1234))

#Tidying the output
lda_bi_topics_beta <- tidy(topic_model_bi, matrix = "beta")

#Getting the top terms for each topic
lda_bi_beta_top_terms <- lda_bi_topics_beta %>%
  group_by(topic) %>%
  slice_max(beta, n = 5) %>%
  ungroup() %>%
  arrange(topic, -beta)

#Plotting the topic words
lda_bi_beta_top_terms %>%
  mutate(term = reorder_within(term, beta, topic)) %>%
  ggplot(aes(beta, term, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  scale_y_reordered()
```

The topics generated using bigrams help to understand the context of the topics better. Topic 1 seems to be relate to store staff, Topic 2 could be web communications team, topic 3 doesn't seem to have any meaning, topic 4 is related to delivery of products, topic 5 refers to price, topic 6 isn't too meaningful, topic 7 isn't meaningful and topic 8 doesn't make sense either.

Hence, the bigrams don't seem to give much effectiveness in understanding topics in this context.

C.4: STM topic modelling approach (Supervised)

C.4.1: Data Processing

- POS Tagging

```
   #Downloading the model for the english language
langmodel_download <- udpipe::udpipe_download_model("english")

#Loading the model
langmodel <- udpipe::udpipe_load_model(langmodel_download$file_model)
```

- Annotating the reviews with POS

```
   #creating an empty dataframe
annotated_reviews_all <- data.frame()

#setting a split size for the data
split_size <- 1000

#splitting the dataset according to set split size
for_pos_list <- split(dataset,
                      rep(1:ceiling(nrow(dataset)
                                    /split_size),
                          each=split_size,
                          length.out=nrow(dataset)))


#annotating the parts of speech using the language model
for(i in 1:length(for_pos_list)){
    udpipe::udpipe_annotate(for_pos_list[[i]]$reviewText,
                doc_id = for_pos_list[[i]]$review_id,
                object = langmodel) %>%
    as.data.frame() %>%
    filter(upos %in% c("NOUN","ADJ","ADV")) %>%
    dplyr::select(doc_id,lemma) %>%
    group_by(doc_id) %>%
    summarise(annotated_reviews = paste(lemma, collapse = " ")) %>%
    rename(review_id = doc_id) -> this_annotated_reviews

  print(paste(i,"from",length(for_pos_list)))
  annotated_reviews_all <- bind_rows(annotated_reviews_all,
                                     this_annotated_reviews)
}
```

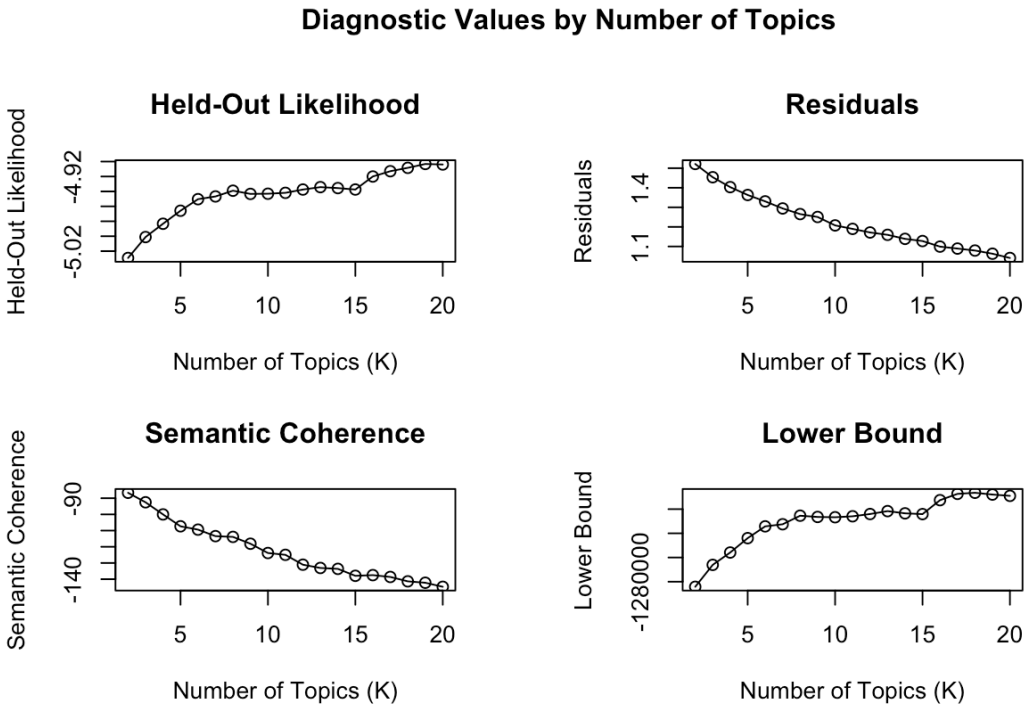
- Prepping data for STM model

```
   #Setting the datatype of the review_id variable for use in stm model
annotated_reviews_all$review_id <- as.integer(annotated_reviews_all$review_id
)

#combining the annotated reviews with the main dataset to be used for stm mod
elling
data_for_stm <- annotated_reviews_all %>%
```

```
                left_join(dataset)

#Setting the datatype of the reviewer ratings for use in stm model
data_for_stm$reviewer_rating <- as.integer(data_for_stm$reviewer_rating)
   #processing the data through automated cleaning of function
processed <- textProcessor(data_for_stm$annotated_reviews,
                           metadata = data_for_stm,
                           customstopwords = c("customer", "review", "ever",
"even", "site", "business", "also",  "just", "still", "one", "able", "many"),
                           stem = F)

   #keep those words who appear more than 1% in the document corpus
threshold <- round(1/100 * length(processed$documents),0)

#getting the final data to be used in stm model
out <- prepDocuments(processed$documents,
                     processed$vocab,
                     processed$meta,
                     lower.thresh = threshold)
```

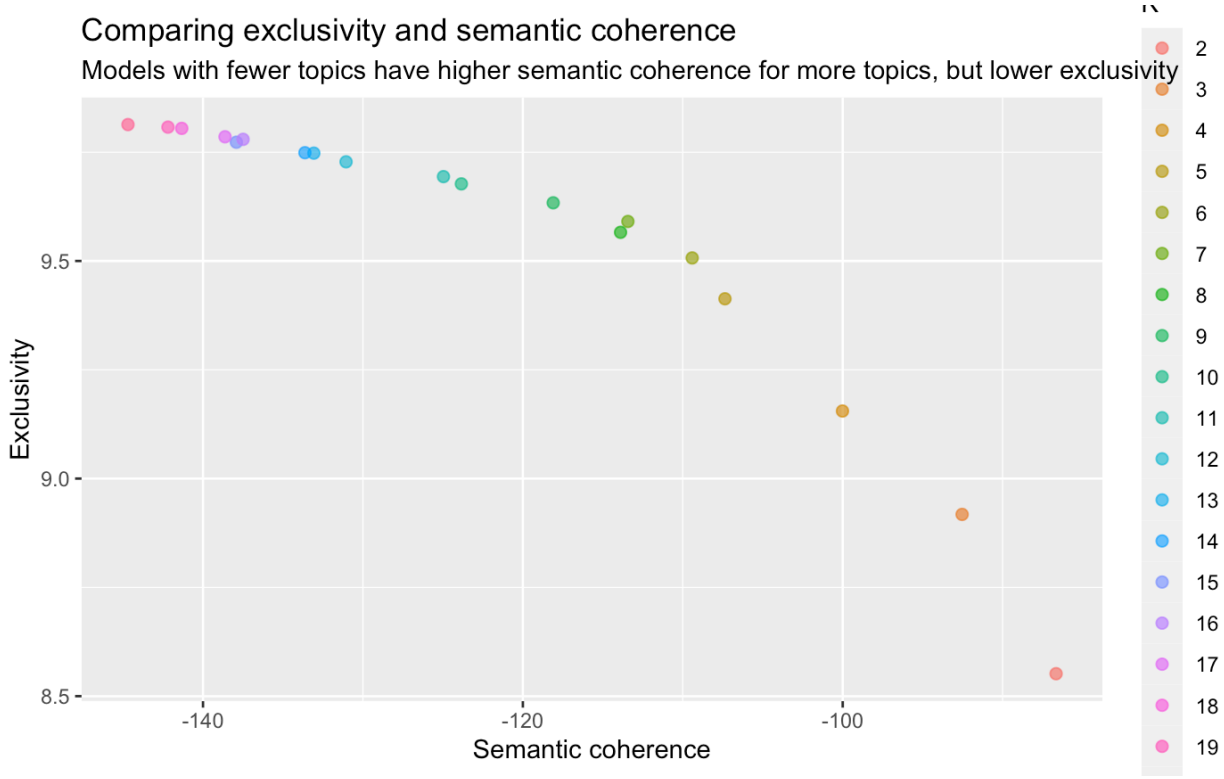
C.4.2: Kappa (K) Evaluation

```
   #searching for optimal k value
numtopics <- searchK(out$documents,out$vocab,K=seq(from=2, to=20,by=1))
   #plotting the result
plot(numtopics)
```



Based on the outcomes of searchK, we are going to find a value that has high held-out likelihood, low residuals and good semantic coherence. The range from 10-15 seems to be promising. We are going to further analyse the results of search to be sure about the exact value of K to choose.

```
   numtopics$results %>%
  dplyr::select(K, exclus, semcoh) %>%
  filter(K %in% c(1:20)) %>%
  unnest() %>%
  mutate(K = as.factor(K)) %>%
  ggplot(aes(semcoh, exclus, color = K)) +
  geom_point(size = 2, alpha = 0.7) +
  labs(x = "Semantic coherence",
       y = "Exclusivity",
       title = "Comparing exclusivity and semantic coherence",
       subtitle = "Models with fewer topics have higher semantic coherence fo
r more topics, but lower exclusivity")
```



After plotting the K values in terms of exclusivity vs semantic coherence, we can see that the K value of 11 seems to have the better position as it has a good balance of both semantic coherence and exclusivity. Thus, we are going to use STM model to search for 11 topics.

C.4.3: Topic Modelling

- STM for 11 topics

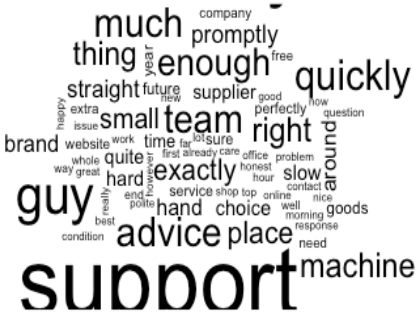
```
   # for 11 topics
reviews_fit_11 <- stm(documents = out$documents,
                      vocab = out$vocab,
                      prevalence = ~ reviewer_rating,
                      K = 11,
                      max.em.its = 75,
                      data = out$meta,
                      sigma.prior = 0.5,
                      init.type = "LDA")
```

C.4.4: Visualizing the topics

```
   #visualise top words in wordclou
d
set.seed(1)
stm::cloud(reviews_fit_11,topic = 1
)
```


```
   stm::cloud(reviews_fit_11,topic
= 3)
```


```
   stm::cloud(reviews_fit_11,topic
= 2)
```




```
   stm::cloud(reviews_fit_11,topic
= 4)
```

```
stm::cloud(reviews_fit_11,topic = 5)
```


```
stm::cloud(reviews_fit_11,topic = 6)
```


```
stm::cloud(reviews_fit_11,topic = 7)
```

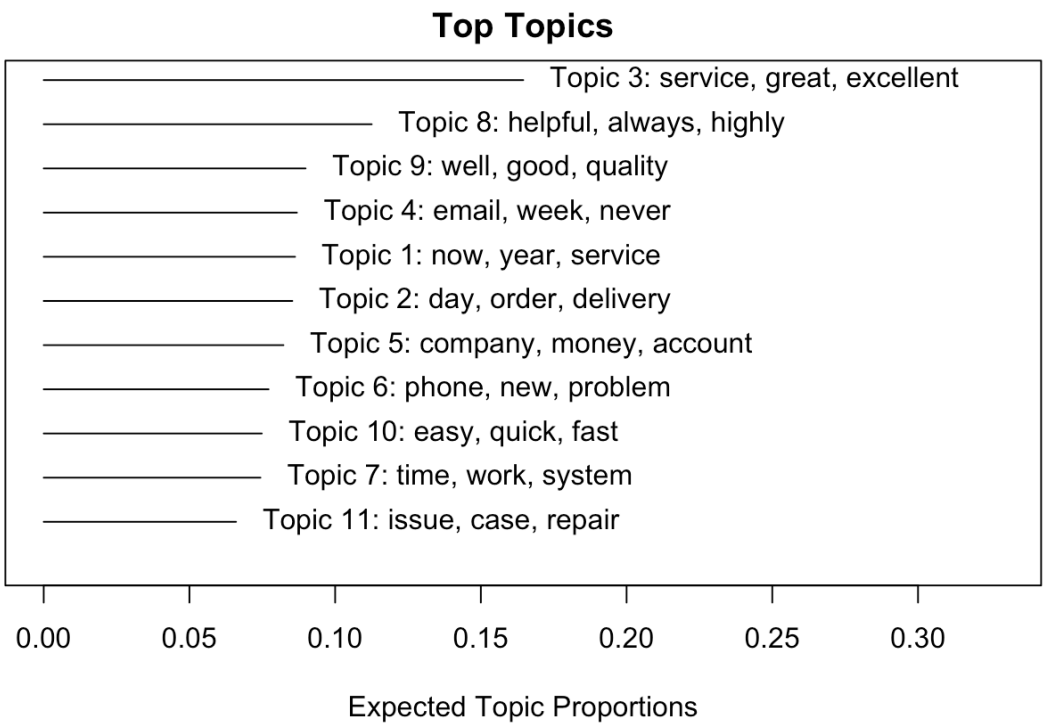

```
stm::cloud(reviews_fit_11,topic = 8)
```

```
stm::cloud(reviews_fit_11,topic
= 9)
```


```
stm::cloud(reviews_fit_11,topic
= 11)
```


```
stm::cloud(reviews_fit_11,topic
= 10)
```

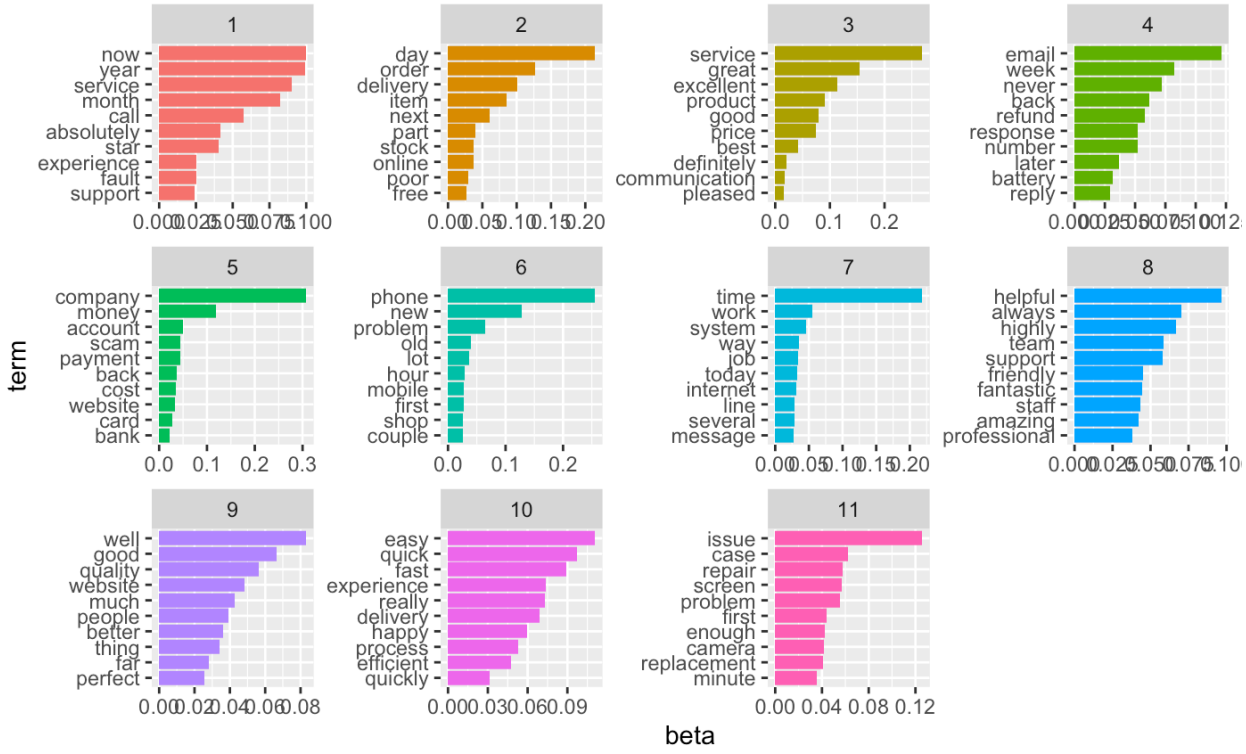
```
   #visualize topics
plot(reviews_fit_11, type="summary")
```



The plot shows the top topics in the order of higher proportions and the top three terms included in the topics.

- Visualizing the top words in each topic

```
stm_11_topics_beta <- tidy(reviews_fit_11, matrix = "beta")

#Getting the top terms for each topic
stm_11_topics_beta <- stm_11_topics_beta %>%
                      group_by(topic) %>%
                      slice_max(beta, n = 10) %>%
                      ungroup() %>%
                      arrange(topic, -beta)

#Plotting the topic words
stm_11_topics_beta %>% mutate(term = reorder_within(term, beta, topic)) %>%
                       ggplot(aes(beta, term, fill = factor(topic))) +
                       geom_col(show.legend = FALSE) +
                       facet_wrap(~ topic, scales = "free") +
                       scale_y_reordered()
```

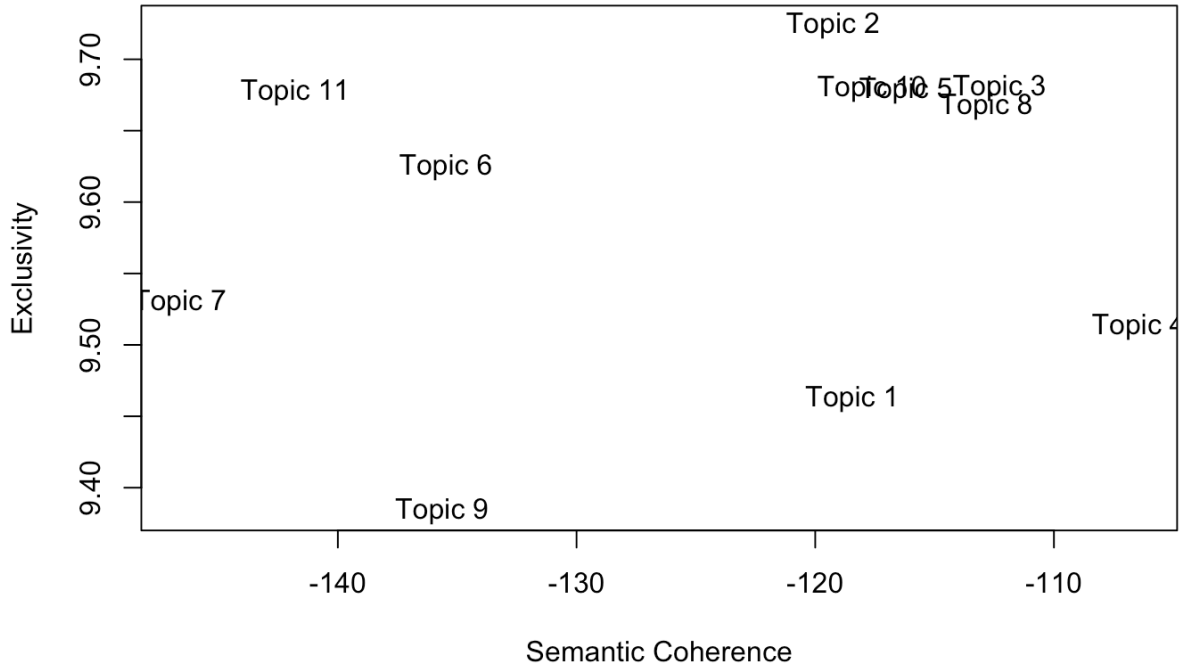


- Finding labels for the topics

```
   #getting the model summary
topic_summary <- summary(reviews_fit_11)
   ## A topic model with 11 topics, 22074 documents and a 289 word dictionary
.
   #see top reviews for each topic
topic_proportions <- colMeans(reviews_fit_11$theta)

#assigning the topic names as column headers
topic_labels <- c("Transactional Issues", "Reliable delivery", "Waiting Time"
, "Product Satisfaction", "Service", "Phone Issues", "Work", "Team Behavior",
 "Webpage", "Delivery Process", "Repairs")

#assembling the topic words in table according to proportions
table_to_write_labels <- data.frame()
for(i in 1:length(topic_summary$topicnums)){

   row_here <- tibble(topicnum= topic_summary$topicnums[i],
                      topic_label = topic_labels[i],
                      proportion = 100*round(topic_proportions[i],3),
                     frex_words = paste(topic_summary$frex[i,1:7],
                                        collapse = ", "))
   table_to_write_labels <- rbind(row_here,table_to_write_labels)
}
(table_to_write_labels %>% arrange(topicnum))

##       <int> <chr>                     <dbl> <chr>

##  1        1 Transactional Issues        6.7 camera, line, deal, repair, pr
ovide…
##  2        2 Reliable delivery           7   system, guy, machine, always,
right…
##  3        3 Waiting Time                9.3 internet, last, today, enginee
r, so…
##  4        4 Product Satisfaction        7.6 part, case, laptop, screen, bo
x, wr…
##  5        5 Service                    12.7 quality, good, fast, product,
cable…
##  6        6 Phone Issues                8.1 help, really, process, thanks,
 quic…
##  7        7 Work                       12.1 definitely, best, store, happy
, val…
##  8        8 Team Behavior               9.7 friendly, highly, staff, fanta
stic,…
##  9        9 Webpage                     9.5 battery, call, phone, bad, mob
ile, …
## 10       10 Delivery Process            8.6 item, next, stock, day, order,
 deli…
## 11       11 Repairs                     8.7 scam, refund, money, payment,
card,…
```

C.3.5: Evaluation of topics

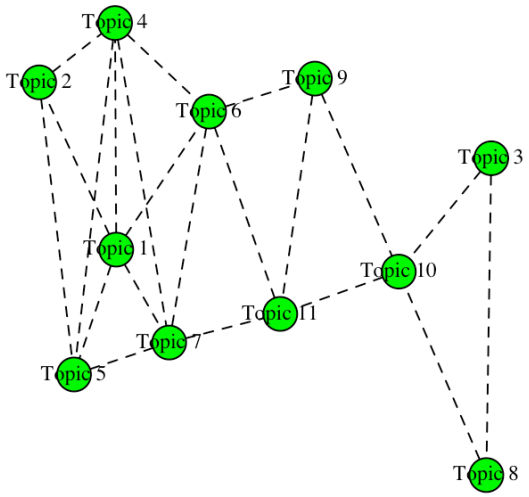
```
   #Visualizing topic quality with respect to exclusivity and semantic cohere
nce

   topicQuality(reviews_fit_11,documents = out$documents)
```



We can see that the topics are spread out over the graph. However, a 5 of the topics are positioned similarly at the top-right end of the graph, which is good. Topic 7 and Topic 9 seem to be the unreliable ones of all the topics due to lower coherence and exclusivity. Overall, the topics seem to be of good quality.

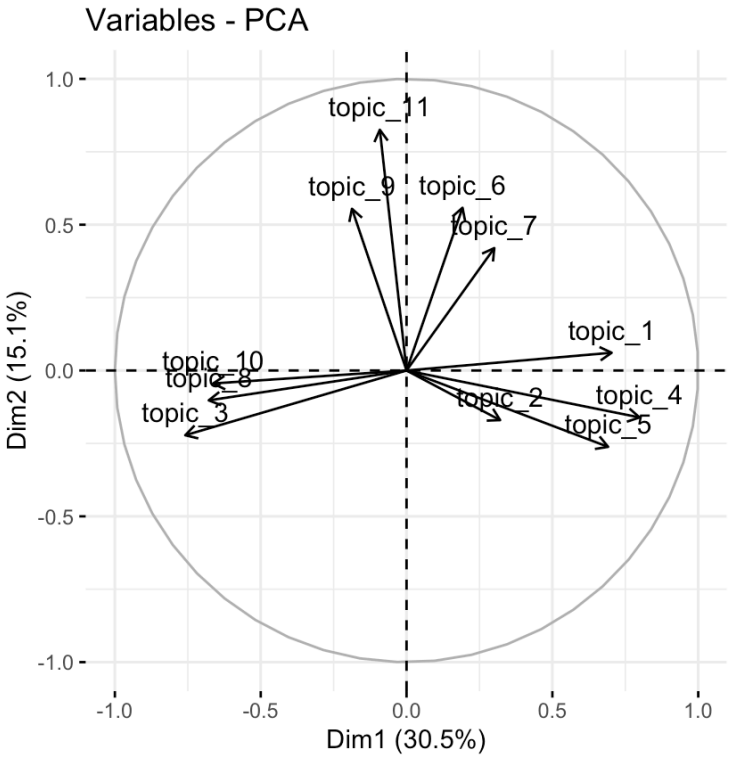
- Finding the correlations and factors for the topics

```
library(FactoMineR)

#extracting metadata and gamma values
gamma_topics <- cbind(out$meta, reviews_fit_11$theta)

#processing dataset to use for correlation
gamma_topics$doc_id <- 1:nrow(gamma_topics)
gamma_topics <- gamma_topics %>% dplyr::select(22:33)
rownames(gamma_topics) <- gamma_topics$doc_id
gamma_topics$doc_id <- NULL
colnames(gamma_topics) <- paste0("topic_", 1:11)

#checking topic correlations
topic.corr <- topicCorr(reviews_fit_11)
plot(topic.corr)
```



From the topic correlations, we can see that topic 11 seems to be an important topic, as a lot of the other topics are correlated with it. Topic 10 and topic 6 also seem to be of high significance. Evidently, it makes sense that customers would talk about repairs and phone issues/store services together and so on.

```
   #checking factor relations
pcah <- FactoMineR::PCA(gamma_topics, graph = FALSE)
factoextra::fviz_pca_var(pcah)
```

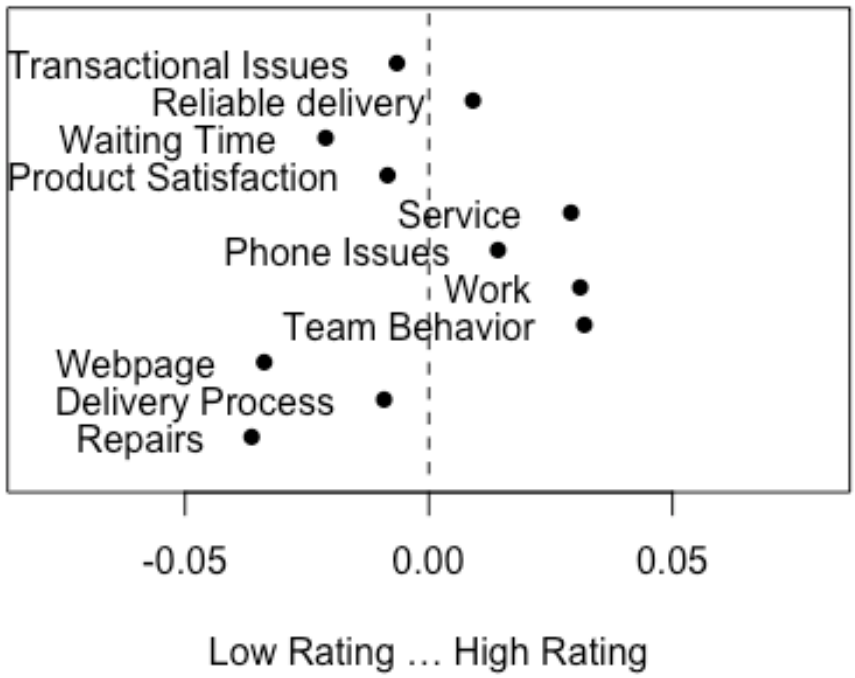


The factor component analysis says that the top two factor components successfully explain 45.6% of the total variation in the dataset of the topics. There seems to be some closeness in the direction of the topics in clusters in three common directions.

C.4.6: Looking at topic effects on metadata

- Looking at the effects between topics and rating

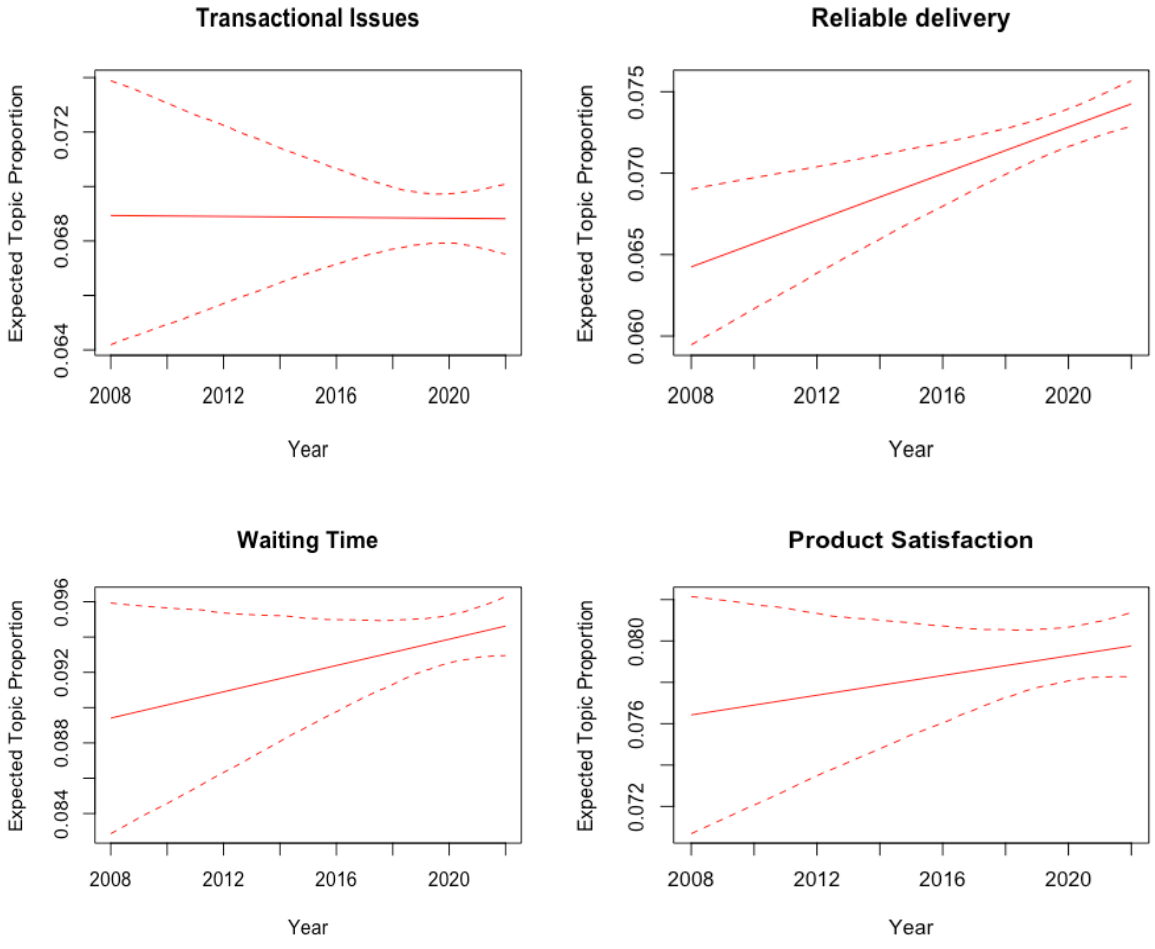
```
   #Predicting topics by reviewer ratings
predict_topics_ratings <- estimateEffect(~ reviewer_rating, stmobj = reviews_
fit_11, metadata = out$meta)

#Visualizing the effects of reviewer ratings
plot(predict_topics_ratings, covariate = "reviewer_rating", topics = c(1:11),
 model = reviews_fit_11, method = "difference",
 cov.value1 = "5", cov.value2 = "1",
 xlab = "Low Rating … High Rating",
 main = "",
 xlim = c(-0.08,0.08),
 custom.labels = topic_labels,
 labeltype = "custom")
```

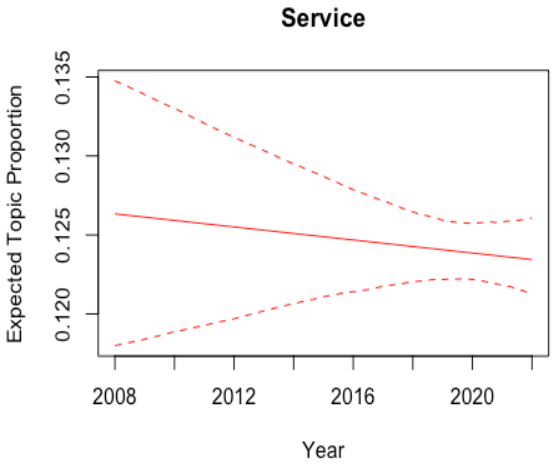


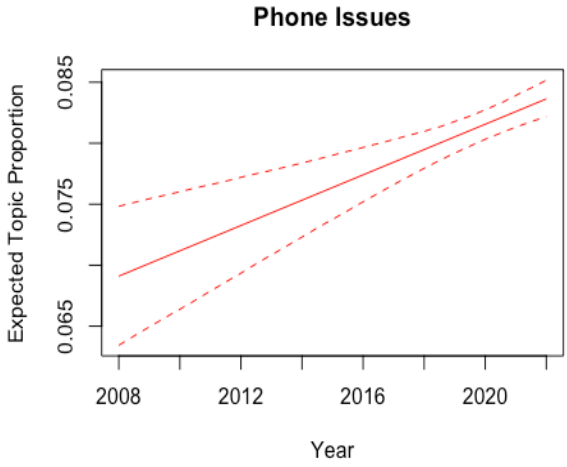
The above plot gives an idea about the relation between the topics and ratings. We can understand that when reviewers talked about topics like team behavior or service, they tended to give higher ratings, whereas reviews related to repairs and webpage seems to get lower ratings.

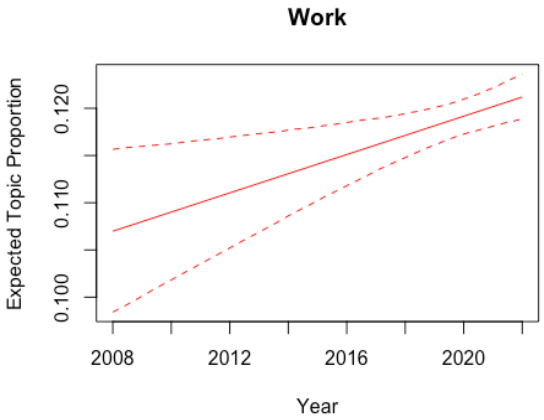
- Checking the topic prevalence over years

```
   #Observing topic prevalence over time
predict_topics_years <- estimateEffect(~ year, stmobj = reviews_fit_11, metad
ata = out$meta)

#Visualizing the effects of reviewer ratings
for(i in 1:length(topic_labels)){

  i <-   (plot(predict_topics_years, covariate = "year",
           topics = i,
           model = review_fit_11, method = "continuous",
           xlab = "Year",
           main = topic_labels[i],
           printlegend = FALSE,
           custom.labels =topic_labels[i],
           labeltype = "custom"))
}
```

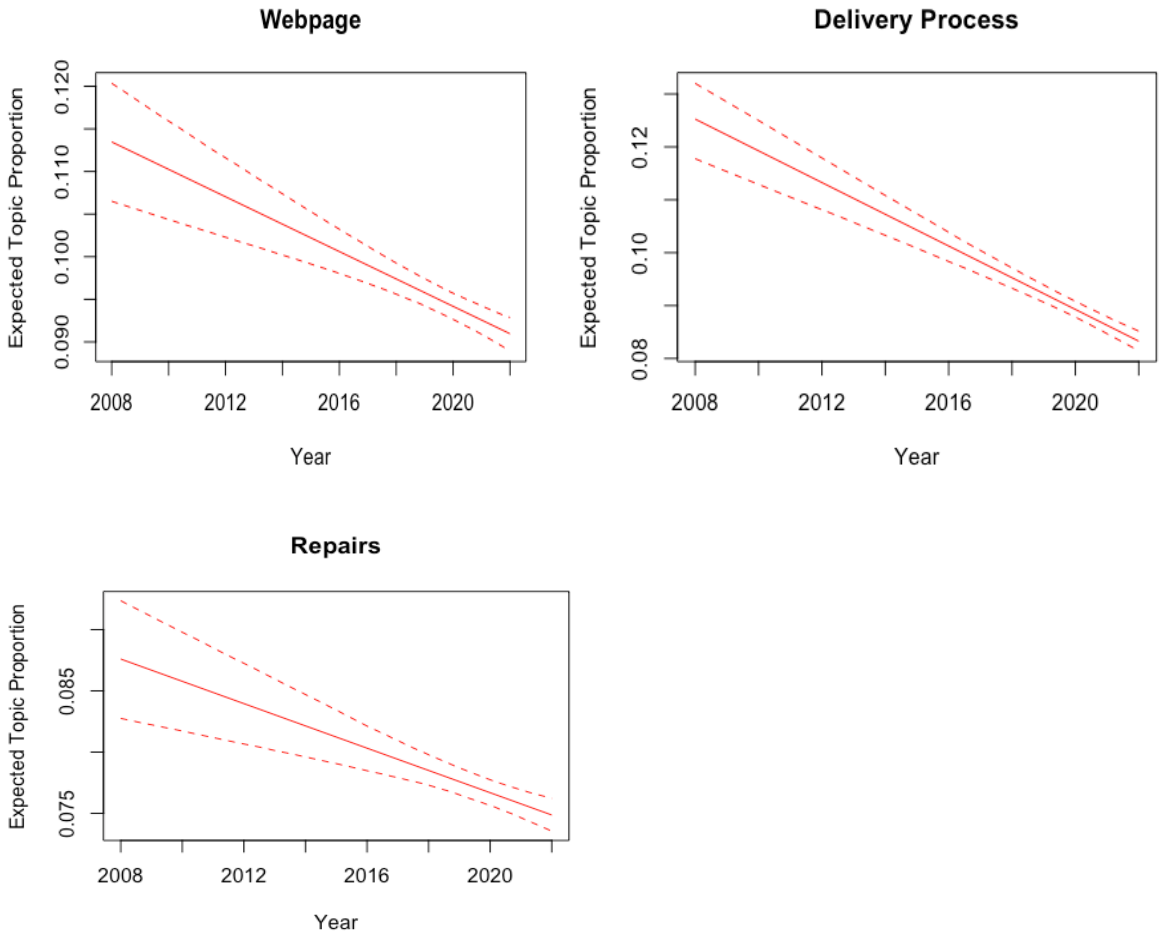



Service
Expected Topic Proportion
0.120
0.125
0.130
0.135
2008
2012
2016
2020
Year


Phone Issues
Expected Topic Proportion
0.065
0.075
0.085
2008
2012
2016
2020
Year


Work
Expected Topic Proportion
0.100
0.110
0.120
2008
2012
2016
2020
Year

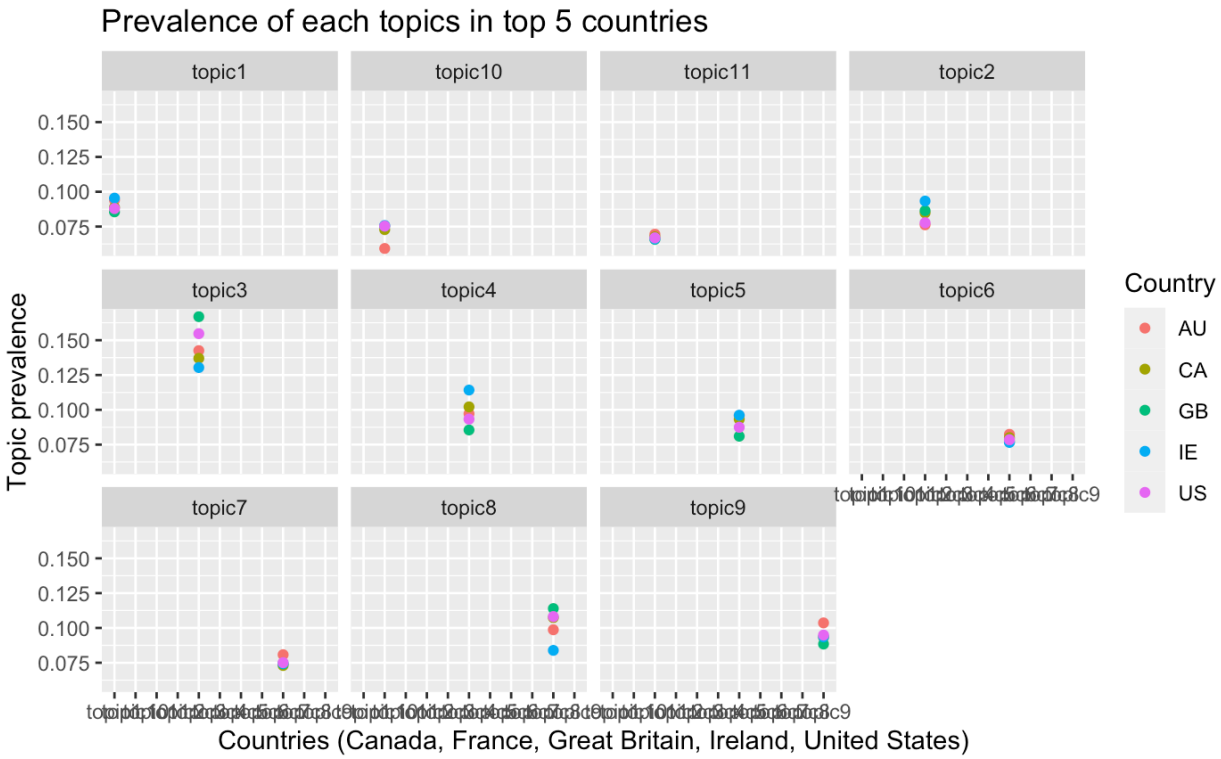

Team Behavior
Expected Topic Proportion
0.07
0.09
2008
2012
2016
2020
Year



The plots show how the prevalence of each topic has changed over the years that the reviews have been collected from 2008 to 2020. Overall, we can see that the topics such as webpage, delivery process and repairs have had a significantly decreasing proportions over the years. On the other hand, team behavior, phone issues and delivery related topics have started to gain in importance recently.

- Checking the prevalence of topics by top countries

```
   #Finding the top 5 countries with most reviews
top_countries <- dataset %>% group_by(country) %>% summarise(total_reviews =
n()) %>% arrange(desc(total_reviews)) %>% top_n(5) %>% dplyr::select(country)

#Getting the theta values and countries
stm_object <- reviews_fit_11$theta
colnames(stm_object) <- paste0("topic_",1:11)
country_topics <- cbind(out$meta, stm_object)
country_topics <- top_countries %>% left_join(country_topics)

#Summarizing the theta values by top 5 countries
country_topic_prevalence <- country_topics %>% group_by(country) %>% summaris
e(topic1=mean(topic_1),
            topic2 = mean(topic_2),
            topic3 = mean(topic_3),
            topic4 = mean(topic_4),
            topic5 = mean(topic_5),
            topic6 = mean(topic_6),
            topic7 = mean(topic_7),
            topic8 = mean(topic_8),
            topic9 = mean(topic_9),
            topic10 = mean(topic_10),
            topic11 = mean(topic_11))
country_topic_prevalence <- pivot_longer(country_topic_prevalence, 2:12, name
s_to = "topics", values_to = "theta")

#Visualizing the outcome
country_topic_prevalence %>% group_by(topics, country) %>% ggplot(aes(x = top
ics, y = theta, color = country)) + geom_point() + facet_wrap(~ topics) + lab
s(color = "Country", x = "Countries (Canada, France, Great Britain, Ireland,
United States)", y = "Topic prevalence", title = "Prevalence of each topics i
n top 5 countries")
```



The plot show the proportions that each topic tended to cover in terms of countries. Topic 3 seems to have high importance in all the countries in general, while topic 10 seems the least importance. We can understand from this plot which topic to focus on when it comes to any specific country.

C.5: Limitations of Kappa

The unsupervised model is the better model in this case, as it cover much more of the variance in the data than that of the supervised model. However, the selection of K has some limitations, in that, it is ultimately inferior to human judgement and can always create topics that are quite similar. Also, the K value can change every time the perplexity function is run, hence, there the selection is somewhat arbitrary.

Conclusion

Overall, the text analysis has examined various aspects of the reviews ranging from bag of words analysis to sentiment analysis to topic modelling. Thoughout the results of these analyses, various aspects of the reviews and the metadata have been connected with the ratings to give the managers of the electronics and technology industry a thorough insight into the thinking of the customers