TABLE OF CONTENTS

Introduction

The innovations to travel further and further away enabled the tourism sector to grow exponentially for the past decade. With the ability to go and experience new places, as well as the high demand for it due to the exponential growth in the tourism sector, there are many active travel agencies. However, since March 2020, Covid-19 pandemic has affected many sectors, one that is hugely influenced being travelling. Due to travelling restrictions, lockdowns, and the overall fear of getting infected, tourism sector has taken a huge step back and decreased it's priorly forecasted growth. Since many countries are affected by this, being able to determine the customer values holds a crucial role for many travelling agencies if they are planning on staying in the market.

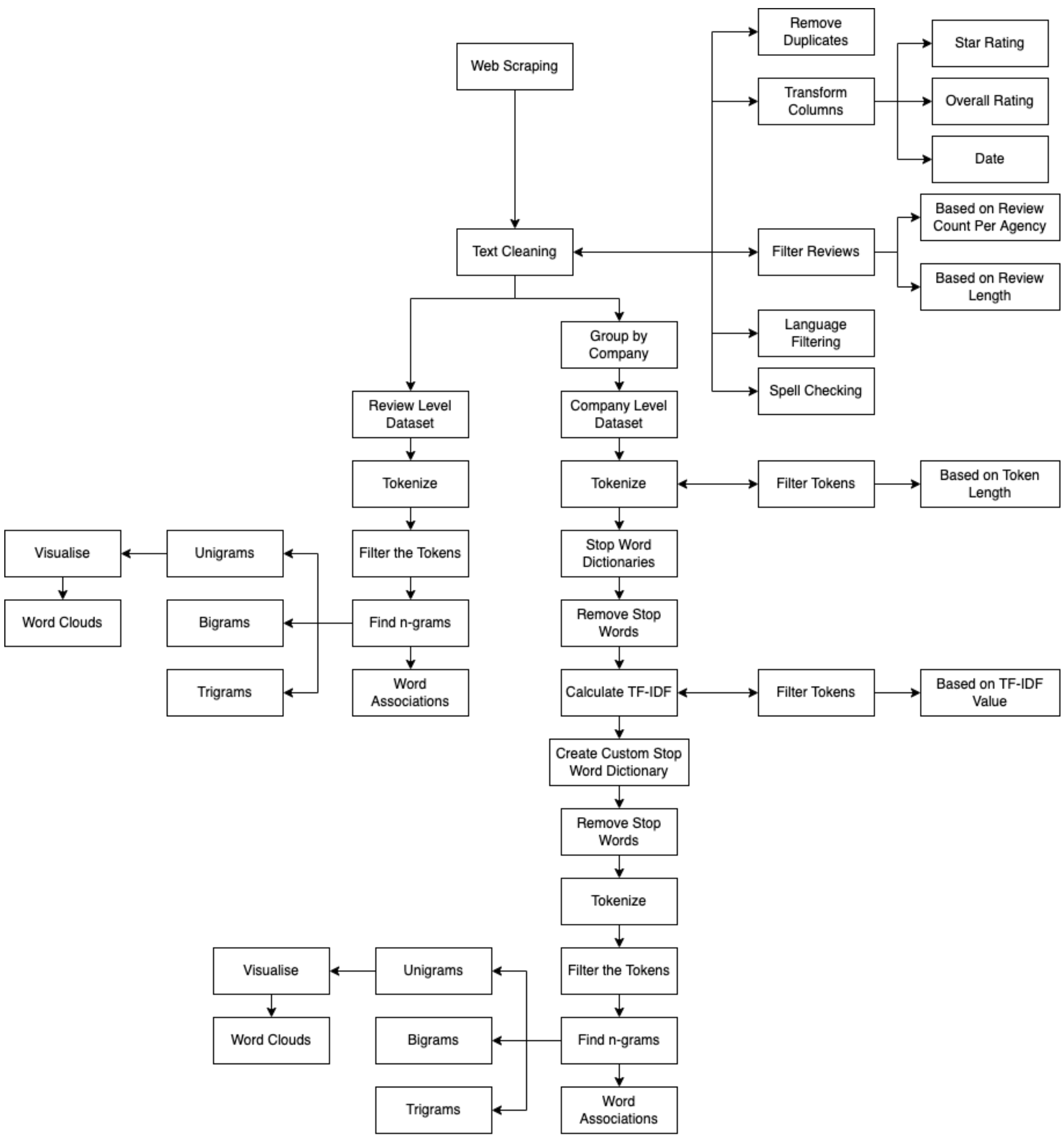
Trustpilot is a website that contains a huge variety of reviews, one section composed of reviews that people give to the travelling agencies. There, people can read the reviews of other people and be informed about their experiences, hence they can decide to try a travelling agency or not. Because of its vast number of reviews ranged from high to low, it is a great source to perform text mining and doing sentiment analysis. For companies and managers that want to focus on the areas that their customers find important and where their competitors are falling short is a clever way to improve and expand their businesses. With the uncertainty of Covid-19, while the past patterns may not hold and the customer behaviour is changing, it's core and the aspects that the customers value are most likely to stay the same.

Part A

Section Plan

In this section, the bag of words analysis is performed along with finding the dominant (most frequent) words based on the rating scores. The words are analyzed on unigram, bigram and trigram levels. The most frequent words as well as the highest ranking words based on their TF-IDF scores are visualized through various plots. Following this, most frequent words are plotted as word clouds for each rating level (1, 2, 3, 4, 5). In addition, word importance is investigated through different components available in the metadata (category of the travel agency and the country the review has been made).

While most of the steps are done for both review and company level, some were done on a specific level. For further understanding, the pipeline below is attached.



Data Preparation

- Relevant libraries are loaded.

```
library(rvest)
library(dplyr)
library(tidyverse)
library(qdap)
library(cld2)
library(tidytext)
library(hunspell)
library(stringi)
library(igraph)
library(ggraph)
library(wordcloud)
library(tm)
library(textstem)
library(gridExtra)
library(ggrepel)
```

- Functions to be used for web scraping are defined.

```
# Function for retrieving the reviewer names
get_reviewer_names <- function(sub_link) {
  agency_page <- read_html(sub_link)
  reviewer_name <- agency_page %>%
    html_nodes(".styles_consumerDetailsWrapper__p2wdr") %>%
    html_node(".styles_consumerName__dP8Um") %>%
    html_text() %>%
    paste(collapse = "<<")
  return(reviewer_name)
}

# Function for retrieving the review summaries
get_review_summaries <- function(sub_link) {
  agency_page <- read_html(sub_link)
  review_summary <- agency_page %>%
    html_nodes(".styles_reviewContent__0Q2Tg") %>%
    html_node(".styles_linkwrapper__73Tdy") %>%
    html_text() %>%
    paste(collapse = "<<")
  return(review_summary)
}

# Function for retrieving the reviews
get_reviews <- function(sub_link) {
  agency_page <- read_html(sub_link)
  review <- agency_page %>%
    html_nodes(".styles_reviewContent__0Q2Tg") %>%
    html_node("p.typography_typography__QgicV") %>%
```

```
    html_text() %>%
    paste(collapse = "<<")
  return(review)
}

# Function for retrieving the stars each reviewer gave for the travel agency
get_stars <- function(sub_link) {
  agency_page <- read_html(sub_link)
  overall_star <- agency_page %>%
    html_nodes(".styles_reviewHeader__iU9Px") %>%
    html_node(".star-rating_starRating__4rrcf img") %>%
    html_attr("alt") %>%
    paste(collapse = "<<")
  return(overall_star)
}

# Function for retrieving the dates of the reviews
get_date <- function(sub_link) {
  agency_page <- read_html(sub_link)
  date <- agency_page %>%
    html_nodes(".styles_reviewHeader__iU9Px") %>%
    html_node(".typography_typography__QgicV time") %>%
    html_attr("datetime") %>%
    str_sub(., 1,10) %>%
    paste(collapse = "<<")
  return(date)
}

# Function for retrieving the country that the reviewer is located in
get_country <- function(sub_link) {
  agency_page <- read_html(sub_link)
  country <- agency_page %>%
    html_nodes(".styles_consumerExtraDetails__fxS4S") %>%
    html_node("span.typography_typography__QgicV") %>%
    html_text() %>%
    paste(collapse = "<<")
  return(country)
}
```

- Web scraping is performed from the Trustpilot web page. We will be going with the category 'Travel Agencies'

```
# Preparing the tables to be used
travel_agencies <- data.frame()
travel_agencies_new <- data.frame()

# For loop for going through each odd numbered page on the main link
for (page_result in seq(from = 1, to = 79, by = 2)) {
  # Reading the page links from the main page
  link <- paste0("https://www.trustpilot.com/categories/travel_agency?page=",
```

```
page_result)
  page <-  read_html(link)

  # Getting the names of the travel agencies
  agency_name <- page %>%  html_nodes(".styles_displayName__1LIcI") %>% html_
text()
    Sys.sleep(1)

  # Getting the categories of the travel agencies
  category <- page %>% html_nodes(".styles_desktop__3N0-b span:nth-child(1)")
%>% html_text()
    Sys.sleep(1)

  # Getting the overall ratings of the travel agencies
  overall_rating <- page %>% html_nodes(".styles_trustScore__nLHX2 .styles_de
sktop__3N0-b") %>% html_text()
    Sys.sleep(1)

  # Getting each travel agency link on the page that is open
  agency_links <- page %>%  html_nodes(".paper_paper__29o4A a") %>% html_attr
("href") %>% paste0("https://www.trustpilot.com", .)
  travel_agencies <- data.frame()

  # For loop for going through every travel agency on the main page that is o
pen
  for (page_result1 in seq(from = 1, to = length(agency_links), by = 1)) {
    # Temporary data frame that is going to be added at the end of the iterat
ions for each travel agency
    travel_agencies_temp <- data.frame()
    # For loop for going through the first 5 sub pages of the travel agency
    for (page_result2 in seq(from = 1, to = 5, by = 1)) {
      sub_links <- agency_links[page_result1] %>% paste0(., "?page=", page_re
sult2)
      reviewer <- sapply(sub_links, FUN = get_reviewer_names, USE.NAMES = FAL
SE)
        Sys.sleep(2)
      summary <- sapply(sub_links, FUN = get_review_summaries, USE.NAMES = FA
LSE)
        Sys.sleep(2)
      review <- sapply(sub_links, FUN = get_reviews, USE.NAMES = FALSE)
        Sys.sleep(2)
      star <- sapply(sub_links, FUN = get_stars, USE.NAMES = FALSE)
        Sys.sleep(2)
      date <- sapply(sub_links, FUN = get_date, USE.NAMES = FALSE)
        Sys.sleep(2)
      country <- sapply(sub_links, FUN = get_country, USE.NAMES = FALSE)
        Sys.sleep(2)
      travel_agencies_temp <- rbind(travel_agencies_temp,
                                    data.frame(agency_name = agency_name[page
```

```
_result1],
                                                       category = category[page_resul
t1],
                                                       overall_rating = overall_ratin
g[page_result1],
                                                       reviewer = reviewer,
                                                       summary = summary,
                                                       review = review,
                                                       star = star,
                                                       date = date,
                                                       country = country,
                                                       stringsAsFactors = FALSE))

      # If all of the sub pages are gone over, take the information stored in
the temporary data frame and merge them using a separator,
      # otherwise just print the sub page that just finished being scraped fr
om
      ifelse(page_result2 == 5, travel_agencies_temp <- t(travel_agencies_tem
p) %>%
                                                                as.data.frame() %>%
                                                                unite(col = "V", sep
= "<<") %>%
                                                                t(), print(paste0("Su
bpage: ", page_result2, " - ", agency_name[page_result1])))
    }
  # Add the final version of the temporary data frame of the travel agency to
the data frame that we have been storing information into by
  # binding through rows
  travel_agencies <- rbind(travel_agencies, travel_agencies_temp)
  }
  # Add the travel agency information to a different data frame by binding th
rough rows
  travel_agencies_new <- rbind(travel_agencies_new, travel_agencies)
  print(paste0("Page: ", page_result))
}

# Saving the final data frame as RDS in order to store in the environment in
case it gets lost
# saveRDS(travel_agencies_new, "travel_agencies_1_13.rds")
# saveRDS(travel_agencies_new, "travel_agencies_15_27.rds")
# saveRDS(travel_agencies_new, "travel_agencies_29_41.rds")
# saveRDS(travel_agencies_new, "travel_agencies_43_55.rds")
# saveRDS(travel_agencies_new, "travel_agencies_57_69.rds")
# saveRDS(travel_agencies_new, "travel_agencies_65_68.rds")
# saveRDS(travel_agencies_new, "travel_agencies_69_72.rds")
# saveRDS(travel_agencies_new, "travel_agencies_73_75.rds")
# saveRDS(travel_agencies_new, "travel_agencies_76_79.rds")

# Reading all the RDS files that were saved and binding them together
travel_agencies_1_13 <- readRDS("travel_agencies_1_13.rds")
```

```r
travel_agencies_15_27 <- readRDS("travel_agencies_15_27.rds")
travel_agencies_29_41 <- readRDS("travel_agencies_29_41.rds")
travel_agencies_43_55 <- readRDS("travel_agencies_43_55.rds")
travel_agencies_57_69 <- readRDS("travel_agencies_57_69.rds")
travel_agencies_65_68 <- readRDS("travel_agencies_65_68.rds")
travel_agencies_69_72 <- readRDS("travel_agencies_69_72.rds")
travel_agencies_73_75 <- readRDS("travel_agencies_73_75.rds")
travel_agencies_76_79 <- readRDS("travel_agencies_76_79.rds")

travel_all <- rbind(travel_agencies_1_13,
                    travel_agencies_15_27,
                    travel_agencies_29_41,
                    travel_agencies_43_55,
                    travel_agencies_57_69,
                    travel_agencies_65_68,
                    travel_agencies_69_72,
                    travel_agencies_73_75,
                    travel_agencies_76_79)

# Removing the parts of data from the environment since the combined version
is there
rm(travel_agencies_1_13,
   travel_agencies_15_27,
   travel_agencies_29_41,
   travel_agencies_43_55,
   travel_agencies_57_69,
   travel_agencies_65_68,
   travel_agencies_69_72,
   travel_agencies_73_75,
   travel_agencies_76_79)
```

- Transforming the data to be used for analysis in normal form

```r
# Removing the duplicated observations that occurred during scraping since th
e data were scraped part by part and the website was updated constantly with
new ratings, hence moving the travel agencies to different rows and pages
travel_all <- travel_all %>% distinct()

# Adding a dummy variable to join the data after separating the entries based
on the separator "<<"
travel_all$id <- 1:nrow(travel_all)

# Storing the separated information separately because of different numbers o
f rows and then using inner join to join them together
a <- travel_all[, c(1, 2, 3, 10)] %>% separate_rows(1:4, sep = "<<")

# The row on 751 had the same separator "<<" as the one being used here. Once
the review was checked, it was removed
travel_all$review[751] <- str_replace(travel_all$review[751], "d. <<", "d.")
```

```
b <- travel_all[, 4:10] %>% separate_rows(1:7, sep = "<<")

travel_all <- a %>% inner_join(b)
# rm(a, b)

# Removing the dummy variable since it is no longer needed
travel_all$id <- NULL

# Since there were still duplicate rows once the data set was viewed, they we
re removed
travel_all <- travel_all %>% distinct()

# Adding ID column for companies to make it easier to differentiate the separ
ated information that was priorly collapsed later on
for_agency_ids <- travel_all %>%
  select(agency_name) %>%
  distinct()

# To be used to add agency IDs
for_agency_ids$agency_id <- 1:nrow(for_agency_ids)

# Joining the agency ID variable using the agency name
travel_all <- travel_all %>%
  left_join(for_agency_ids)
# rm(for_agency_ids)

# Reordering the variables
travel_all <- travel_all %>%
  select(agency_id, agency_name, category, overall_rating, reviewer, summary,
review, star, date, country)
```

- Formatting some of the variables so the analysis will be easier.

```
# Checking the structure of the data
str(travel_all)

## tibble [39,901 × 10] (S3: tbl_df/tbl/data.frame)
##  $ agency_id     : int [1:39901] 1 1 1 1 1 1 1 1 1 1 ...
##  $ agency_name   : chr [1:39901] "The Wildland Trekking Company" "The Wild
land Trekking Company" "The Wildland Trekking Company" "The Wildland Trekking
Company" ...
##  $ category      : chr [1:39901] "Tour Operator" "Tour Operator" "Tour Ope
rator" "Tour Operator" ...
##  $ overall_rating: chr [1:39901] "TrustScore 5.0" "TrustScore 5.0" "TrustS
core 5.0" "TrustScore 5.0" ...
##  $ reviewer      : chr [1:39901] "Gail" "Kelly" "Bob McNichols" "Bright An
gel" ...
##  $ summary       : chr [1:39901] "Excellent Experience" "Excellent Trip" "
```

```
This trip was wonderful" "Wildland Trekking did a great job!!" ...
##  $ review         : chr [1:39901] "Carlos was an excellent guide.  He was v
ery knowledgeable and kept us safe.  We would definitely use this compa"| __t
runcated__ "Excellent Trip!  Isaac was our guide and he was awesome!  Only mi
nor \"hiccup\" was that when we returned it wa"| __truncated__ "This trip was
wonderful.  The guide was excellent and very knowledgable about the Grand Can
yon.  We will travel"| __truncated__ "Wildland Trekking did a great job for o
ur hike into the Grand Canyon!  They handled everything, from getting ou"| __
truncated__ ...
##  $ star           : chr [1:39901] "Rated 5 out of 5 stars" "Rated 5 out of
5 stars" "Rated 5 out of 5 stars" "Rated 5 out of 5 stars" ...
##  $ date           : chr [1:39901] "2022-04-01" "2022-03-31" "2022-03-31" "2
022-03-16" ...
##  $ country        : chr [1:39901] "CA" "US" "US" "US" ...

# Removing unnecessary text from numerical data from overall rating and indiv
idual ratings and converting them from character to numeric values
travel_all$overall_rating <- travel_all$overall_rating %>%
  substr(12, 14) %>%
  as.numeric() %>%
  round() # Rounded to reduce the levels later on
travel_all$star <- travel_all$star %>%
  substr(7, 7) %>%
  as.numeric()

# Converting the variable type of date from character to date
travel_all$date <- as.Date(travel_all$date)

# Extracting the date from date variable so that year, month, and day are in
separate columns and converting them from character to numeric values
travel_all$year <- format(travel_all$date, "%Y") %>% as.numeric()
travel_all$month <- format(travel_all$date, "%m") %>% as.numeric()
travel_all$day <- format(travel_all$date, "%d") %>% as.numeric()

# Removing the NULL values where there were no individuals left a rating or a
review as well as "NA" string from reviews where while scraping, no text was
found
travel_all <- travel_all %>% na.omit()
travel_all <- travel_all[-which(grepl("NA", travel_all$review, ignore.case =
FALSE)),]

# Adding review ID
travel_all$review_id <- 1:nrow(travel_all)

# Finding the agency IDs that have less than 5 reviews
to_remove_companies <- travel_all %>%
  group_by(agency_id) %>%
  summarise(count_of_reviews = n()) %>%
  filter(count_of_reviews < 5) %>%
  pull(agency_id)
```

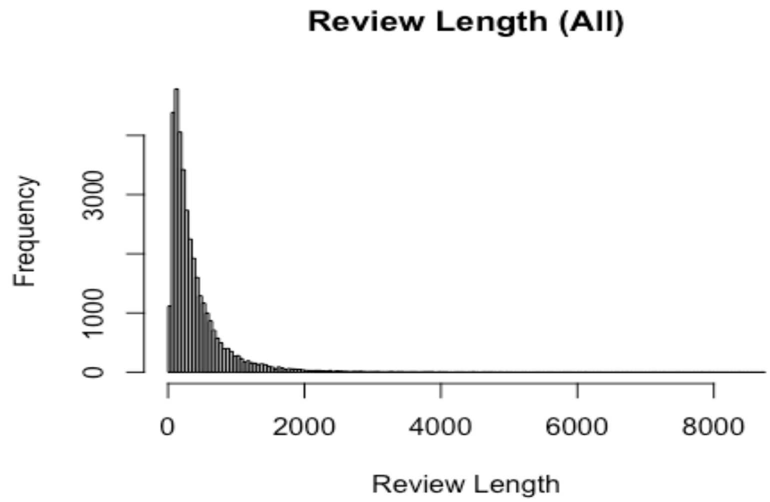
```r
# Since those have less than 5 reviews, they are removed from the data set
travel_all <- travel_all %>%
  filter(!(agency_id %in% to_remove_companies))
# rm(to_remove_companies)

# Collapsing the summary of the review as well as the review itself to work o
n them at the same time instead of repeating the same steps for both
travel_all <- travel_all %>%
  mutate(summary_review = paste(coalesce(travel_all$summary, ""), coalesce(tr
avel_all$review, ""))) %>%
  select(-summary, -review)

# Replacing common abbreviations with their longer versions to be used for di
fferent parts
travel_all$summary_review <- replace_abbreviation(travel_all$summary_review)
# Removing text in between the brackets to be used for different parts
travel_all$summary_review <- bracketX(travel_all$summary_review)
# Replacing the common symbols with their meanings to be used for different p
arts
travel_all$summary_review <- replace_symbol(travel_all$summary_review)
# Removing the digits in the text to be used for different parts
travel_all$summary_review <- gsub("[[:digit:]]+", "", travel_all$summary_revi
ew)
# Removing the white spaces in the text to be used for different parts
travel_all$summary_review <- str_squish(travel_all$summary_review)
```

Filtering The Reviews

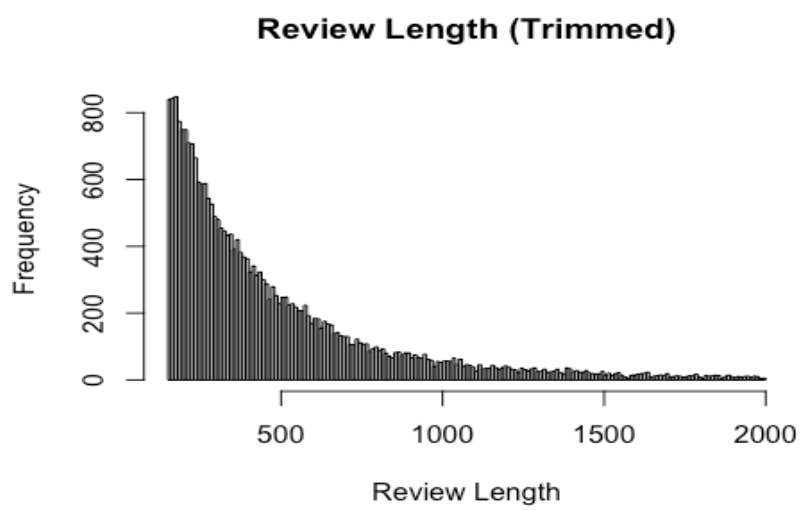
Based on Review Length

Too long reviews can be problematic for analysis and so we remove them. Also, too short reviews might not be too informatic.

```r
# Adding the length of the reviews as a variable and plotting them
travel_all$review_length <- nchar(travel_all$summary_review)
hist(travel_all$review_length, breaks = 200, main = "Review Length (All)", xl
ab = "Review Length")
```




```r
# The very short and very long reviews are removed
travel_all <- travel_all %>% filter(review_length > 150) %>% filter(review_le
ngth < 2000)
hist(travel_all$review_length, breaks = 200, main = "Review Length (Trimmed)"
, xlab = "Review Length")
```



Based on Language

- The languages of the reviews are found and only those that are in English are kept.

```r
travel_all$summary_review <- iconv(travel_all$summary_review)
travel_all$language <- detect_language(travel_all$summary_review)
table(travel_all$language)
```

```
## 
##    ca    da    de    en    es    fr    it    ja    ko    pt    sv    tl 
##     1     1     8 25741     6     6     1     1     1     1     2     1

travel_all <- travel_all%>%
  filter(language == "en")

head(travel_all)

## # A tibble: 6 × 15
##   agency_id agency_name      category  overall_rating reviewer   star date
##       <int> <chr>            <chr>              <dbl> <chr>     <dbl> <dat
e>
## 1         1 The Wildland Tr… Tour Ope…              5 Kelly         5 2022
-03-31
## 2         1 The Wildland Tr… Tour Ope…              5 Bob McNi…     5 2022
-03-31
## 3         1 The Wildland Tr… Tour Ope…              5 Bright A…     5 2022
-03-16
## 4         1 The Wildland Tr… Tour Ope…              5 Bob Flynn     5 2022
-01-18
## 5         1 The Wildland Tr… Tour Ope…              5 Mark          5 2022
-01-17
## 6         1 The Wildland Tr… Tour Ope…              5 Reg Block     5 2022
-01-12
## # … with 8 more variables: country <chr>, year <dbl>, month <dbl>, day <db
l>,
## #   review_id <int>, summary_review <chr>, review_length <int>, language <
chr>

# The final partially-cleaned data set is saved. Further cleaning is to be do
ne depending on the varying parts
# saveRDS(travel_all, "travel_all.rds")
```

Spell Checking

- Since the reviews may have spelling mistakes, the "hunspell" dictionary is used for spelling suggestions that are identified in the reviews. Following that, the dataset is downloaded as a ".csv" file and through using one's judgement, they are manually fixed. Then, the corrected version is uploaded.

```
# Getting the reviews column
reviews <- travel_all %>% select(review_id, summary_review)

# Tokenization of the reviews for spell checking, then extracting the unique
words
for_spellcheck <- unnest_tokens(reviews, word, summary_review)
unique_words <- unique(for_spellcheck$word)

# Finding the words with spelling mistakes and getting the unique spelling mi
stake
```

```
spelling_mistakes <- hunspell(unique_words)
spelling_mistakes <- unique(unlist(spelling_mistakes))

# Getting the correct word suggestions
suggested_words <- hunspell_suggest(spelling_mistakes)
suggested_words <- unlist(lapply(suggested_words, function(x) x[1]))

mistakes <- as.data.frame(cbind(spelling_mistakes, suggested_words))
mistake_freq <- count(for_spellcheck, word)
mistake_freq <- inner_join(mistake_freq, mistakes, by = c("word" = "spelling_
mistakes"))
word_suggestions <- arrange(mistake_freq, desc(n))
na_words <- word_suggestions %>% filter(is.na(suggested_words))

# To handle the spelling suggestions by hand
# write.csv(word_suggestions, "word_suggestions.csv")
# write.csv(na_words, "na_words.csv")

# Loading the prepared data
word_suggestions <- read.csv("word_suggestions.csv")
na_words <- read.csv("na_words.csv")
word_suggestions <- word_suggestions %>% select(-X) %>% na.omit()
na_words <- na_words %>% select(-X) %>% na.omit()

word_suggestions_all <- rbind(word_suggestions, na_words)
word_suggestions_all <- arrange(word_suggestions_all, desc(n))

# Final data
# write.csv(word_suggestions_all, "word_suggestions_all.csv")

# Replacing the mistakes words with suggested ones
word_suggestions_all <- read.csv("word_suggestions_all.csv", stringsAsFactors
= FALSE)
mistaken_words <- paste0(" ", word_suggestions_all$word, " ")
correct_words <- paste0(" ", word_suggestions_all$suggested_words, " ")

replace_mistakes <- function(x) {
 x$summary_review <- stri_replace_all_regex(x$summary_review, mistaken_words,
correct_words, vectorize_all = FALSE)
 return(x)
}

# Splitting reviews based on available cores
reviews <- replace_mistakes(reviews)
reviews <- reviews %>% arrange(review_id)
# saveRDS(reviews, "spell_checked_reviews.rds")

# Replacing the text with the spell checked version
travel_all$summary_review <- reviews$summary_review
```

```r
# Grouping the collapsed variable of summaries and reviews by agency IDs
text_grouped_by_company <- travel_all %>%
  unnest_tokens(word, summary_review) %>%
  group_by(agency_id) %>%
  summarise(text_grouped  = paste(word, collapse = " "))

# saveRDS(text_grouped_by_company, "text_grouped_by_company.rds")
```

Tokenizing

```r
# Tokenizing the grouped data set
tokenized_reviews_grouped_by_company <- text_grouped_by_company %>%
  unnest_tokens(word, text_grouped)
tokenized_reviews_grouped_by_company$word <- lemmatize_words(tokenized_review
s_grouped_by_company$word)
tokenized_reviews_grouped_by_company <- tokenized_reviews_grouped_by_company
%>%
  count(word, agency_id, sort = TRUE)
```

Filtering the Reviews

Based on Token Length

```r
# Calculating the token length to remove those that are very short or long
tokenized_reviews_grouped_by_company$token_length <- nchar(tokenized_reviews_
grouped_by_company$word)

# Inspecting the distribution
tokenized_reviews_grouped_by_company %>% group_by(token_length) %>% summarise
(total = n())

## # A tibble: 30 × 2
##    token_length total
##           <int> <int>
##  1            1  2734
##  2            2 13756
##  3            3 34740
##  4            4 74214
##  5            5 58129
##  6            6 53821
##  7            7 43092
##  8            8 30746
##  9            9 21415
## 10           10 15805
## # … with 20 more rows

# Since there are tokens of 1 and 2 characters that look abnormal based on th
e distribution, removing them
tokenized_reviews_grouped_by_company <- tokenized_reviews_grouped_by_company
%>%
  filter(token_length > 2)
```

```
# Inspecting the distribution from the opposite side
tokenized_reviews_grouped_by_company %>%
  group_by(token_length) %>%
  summarise(total = n()) %>%
  arrange(desc(token_length))
```

```
## # A tibble: 28 × 2
##    token_length total
##           <int> <int>
##  1           39     1
##  2           31     1
##  3           28     2
##  4           27     3
##  5           26     3
##  6           25     7
##  7           24     9
##  8           23    12
##  9           22    26
## 10           21    31
## # … with 18 more rows
```

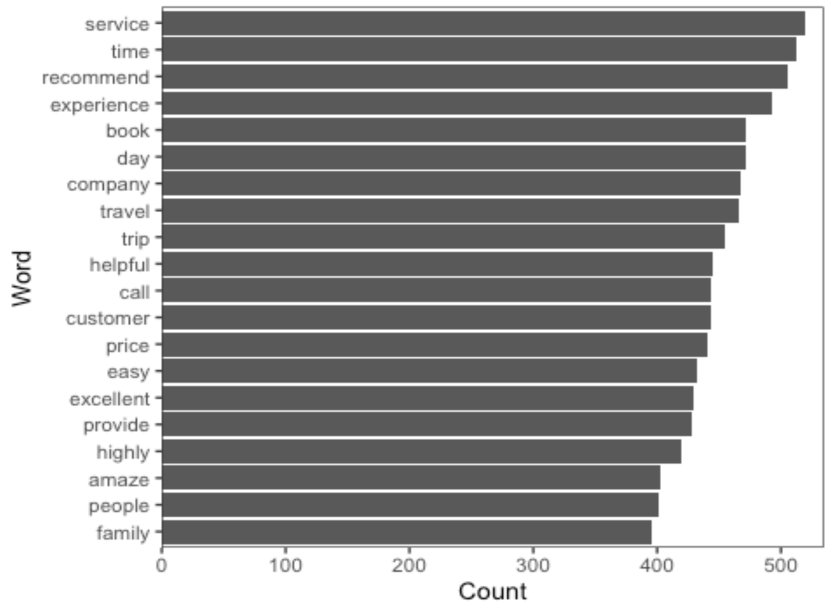
```
# After 14 characters there are some issues, probably with the tokenization p
arsing, hence those tokens with character length more than 14 are removed
tokenized_reviews_grouped_by_company <- tokenized_reviews_grouped_by_company
%>%
  filter(token_length <= 14)

# saveRDS(tokenized_reviews_grouped_by_company, "tokenized_reviews_grouped_by
_company.rds")
```

Stop Word Removal

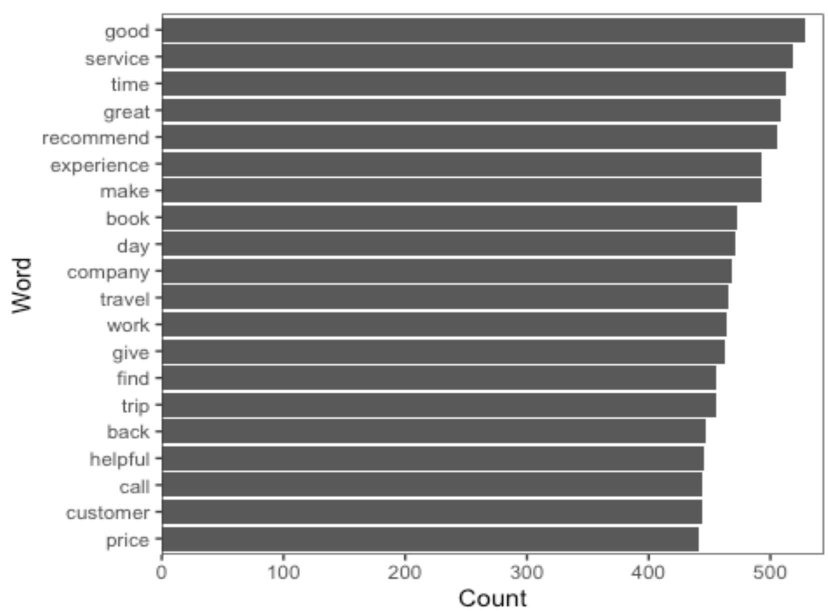
Stop Word Dictionaries

- Different stop word dictionaries are tried in order to find the one that gives the most meaningful frequent words, which is assumed to be the cleanest after stop word removal.

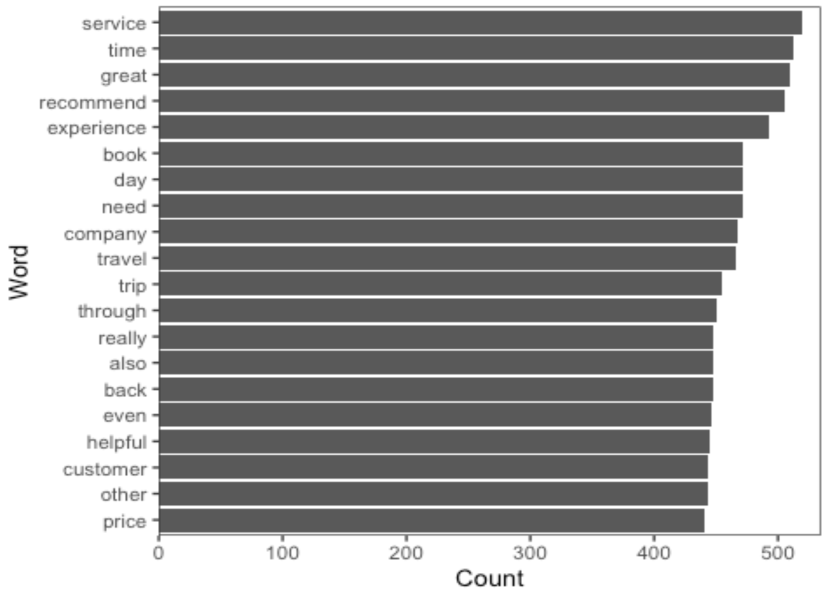
```
# Stop words from tidytext package
tokenized_reviews_regular <- tokenized_reviews_grouped_by_company %>% anti_jo
in(stop_words)
plot_regular <- plot(freq_terms(tokenized_reviews_regular$word)) + labs(title
= "After Removing Stop Words")
```



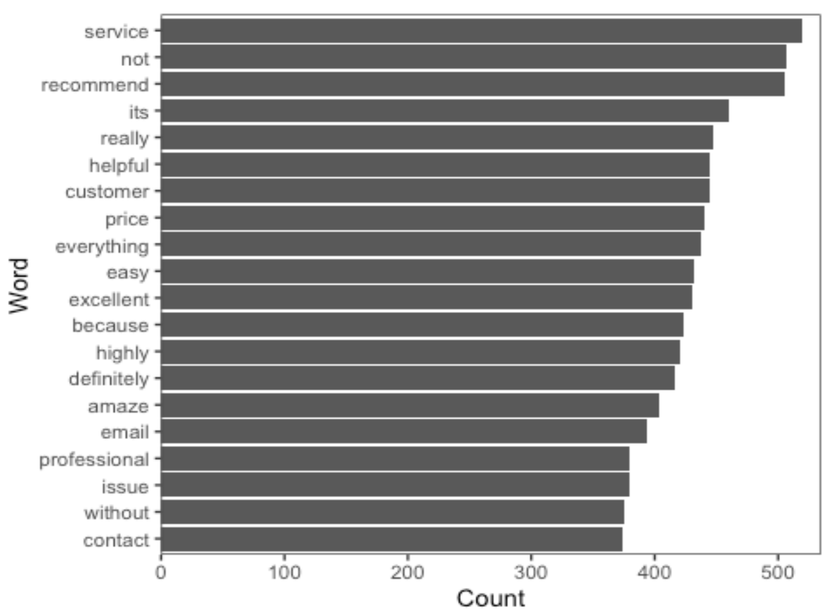

```
# BuckleySaltonSWL stop words dictionary
bsswl <- data.frame(BuckleySaltonSWL) %>% rename(word = BuckleySaltonSWL)
tokenized_reviews_BS <- tokenized_reviews_grouped_by_company %>% anti_join(bs
swl)
plot_bsswl <- plot(freq_terms(tokenized_reviews_BS$word)) + labs(title = "Aft
er Removing BuckleySaltonSWL Stop Words")
```




```
# Dolch stop words dictionary
dolch <- data.frame(Dolch) %>% rename(word = Dolch)
tokenized_reviews_dolch <- tokenized_reviews_grouped_by_company %>% anti_join
(dolch)
plot_dolch <- plot(freq_terms(tokenized_reviews_dolch$word)) + labs(title = "
After Removing Dolch Stop Words")
```

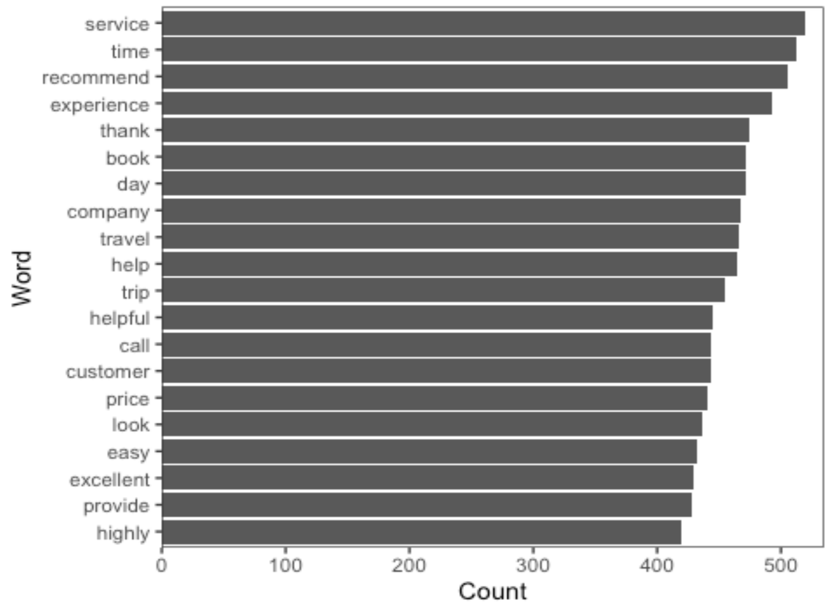



```
# Fry Top 1000 stop words dictionary
fry <- data.frame(Fry_1000) %>% rename(word = Fry_1000)
tokenized_reviews_fry <- tokenized_reviews_grouped_by_company %>% anti_join(f
ry)
plot_fry <- plot(freq_terms(tokenized_reviews_fry$word)) + labs(title = "Afte
r Removing Fry Top 1000 Stop Words")
```

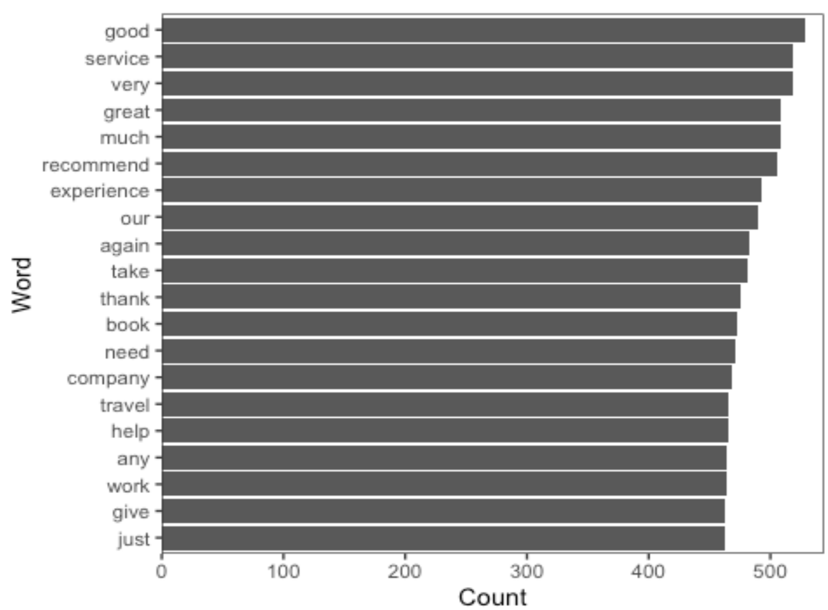



```
# OnixTxtRetToolkitSWL1 stop words dictionary
onix <- data.frame(OnixTxtRetToolkitSWL1) %>% rename(word = OnixTxtRetToolkit
SWL1)
tokenized_reviews_onix <- tokenized_reviews_grouped_by_company %>% anti_join(
onix)
```

```
plot_onix <- plot(freq_terms(tokenized_reviews_onix$word)) + labs(title = "Af
ter Removing OnixTxtRetToolkitSWL1 Stop Words")
```

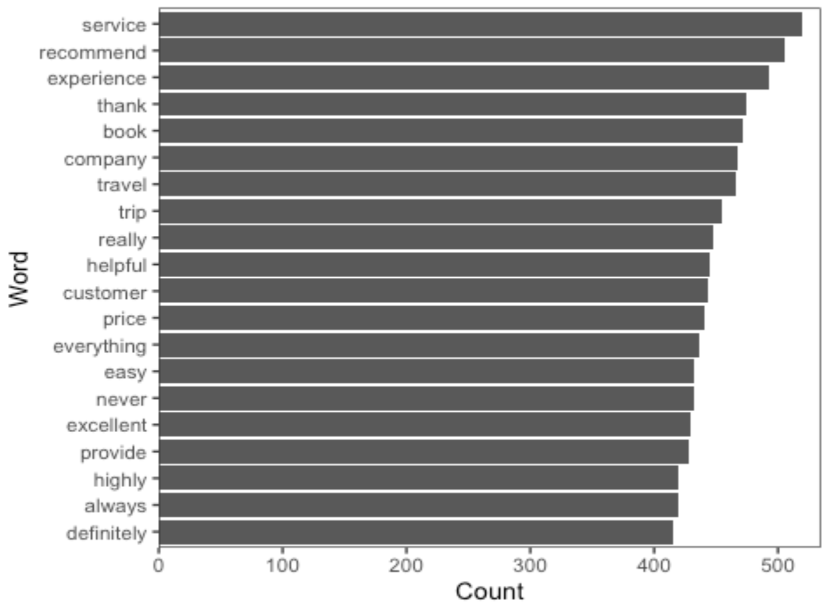



```
# Fry Top 100 stop words dictionary
fry100 <- data.frame(Top100Words) %>% rename(word = Top100Words)
tokenized_reviews_fry100 <- tokenized_reviews_grouped_by_company %>% anti_joi
n(fry100)
plot_fry100 <- plot(freq_terms(tokenized_reviews_fry100$word)) + labs(title =
"After Removing Fry Top 100 Stop Words")
```

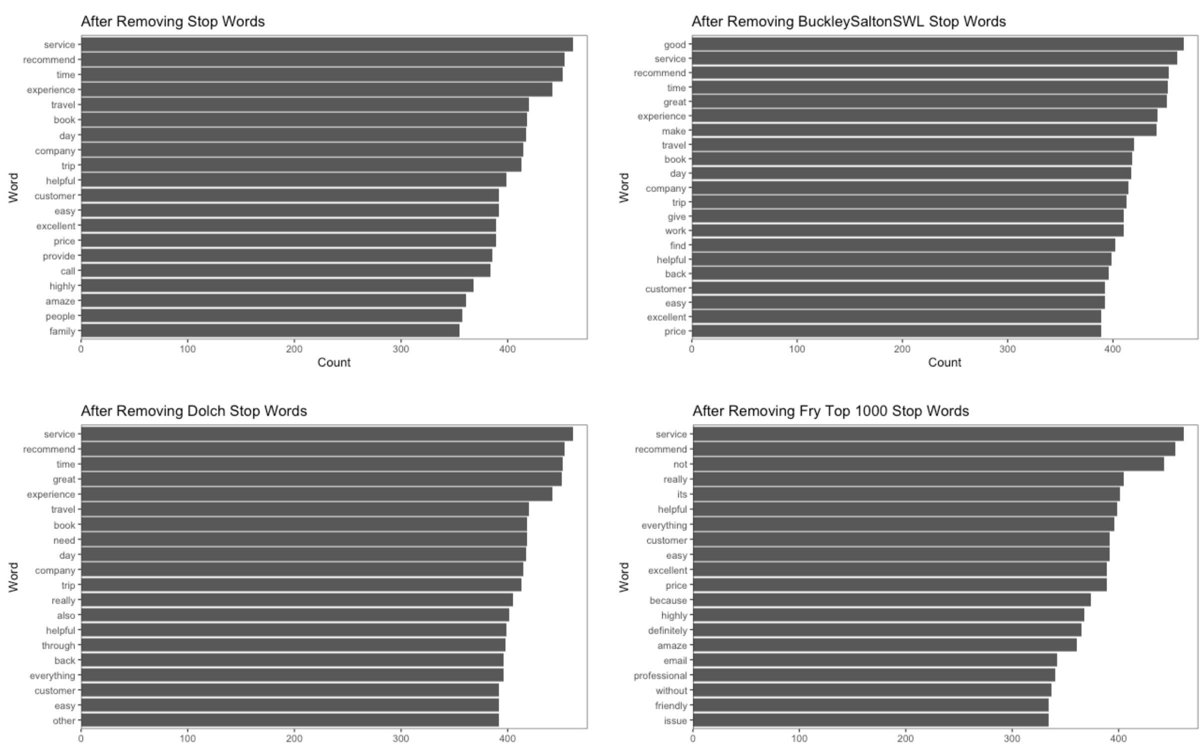


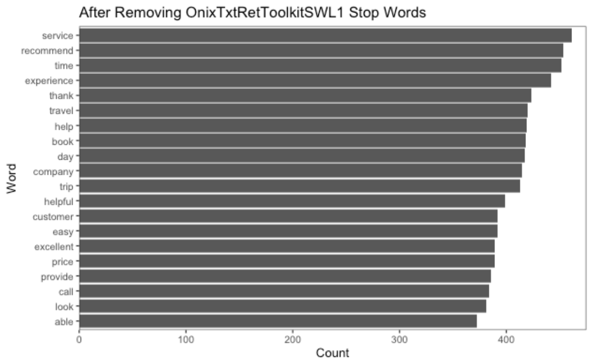

```
# Fry Top 200 stop words dictionary
fry200 <- data.frame(Top200Words) %>% rename(word = Top200Words)
tokenized_reviews_fry200 <- tokenized_reviews_grouped_by_company %>% anti_joi
```

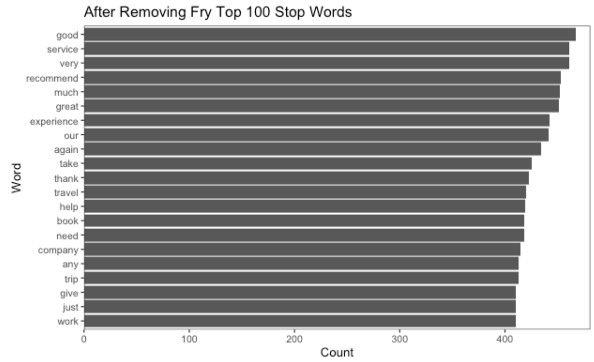
```
n(fry200)
plot_fry200 <- plot(freq_terms(tokenized_reviews_fry200$word)) + labs(title =
"After Removing Fry Top 200 Stop Words")
```

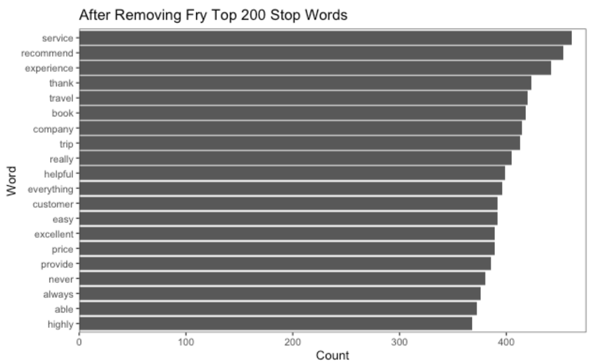



```
# Plotting the most frequent words after using each dictionary separately to
compare them
grid.arrange(plot_regular, plot_bsswl, plot_dolch, plot_fry, plot_onix, plot_
fry100, plot_fry200)
```







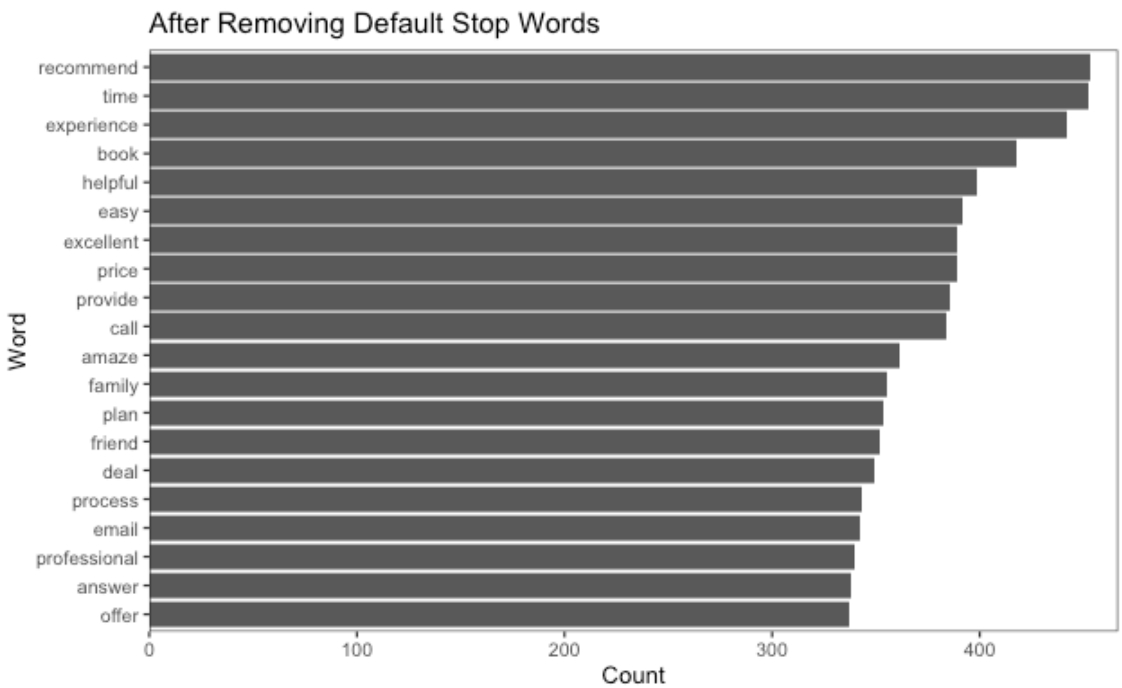



```
# Since the environment has many variables that are not going to be used again, removing everything and only loading the ones that are going to be used
# rm(list = ls())
# travel_all <- readRDS("travel_all.rds")
# text_grouped_by_company <- readRDS("text_grouped_by_company.rds")
# tokenized_reviews_grouped_by_company <- readRDS("tokenized_reviews_grouped_by_company.rds")
```

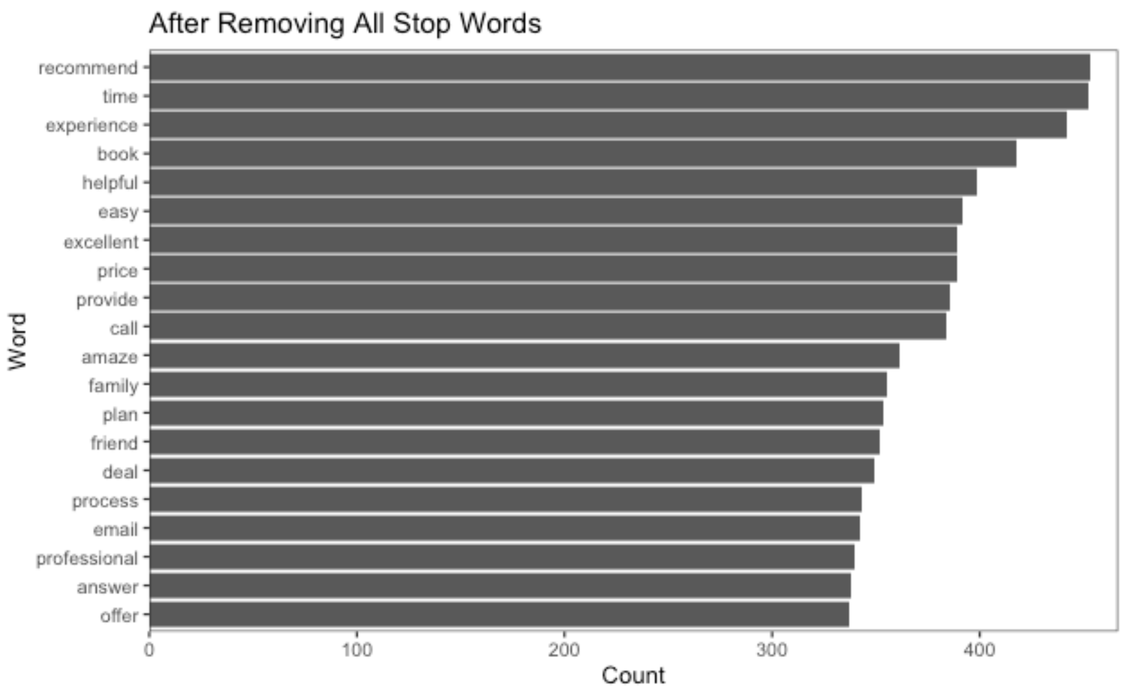
- While examining the frequent terms after removing the stop words, it is observed that the "stop_words" dictionary from tidytext package gives more meaningful and important insight, hence it was picked as the base of the stop words to be used.

Stop Word Removal

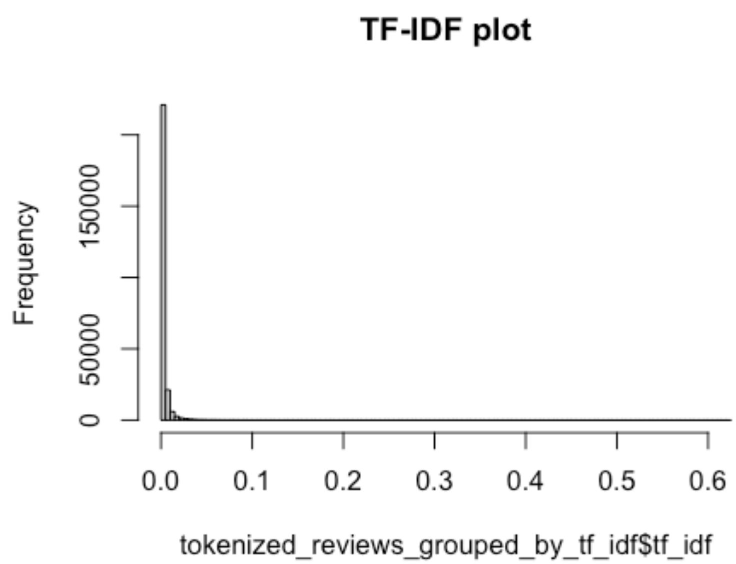
```
# Removing the stop words from tokenized reviews
tokenized_reviews_grouped_by_company <- tokenized_reviews_grouped_by_company %>% anti_join(stop_words, by = "word")

# Looking at the frequent terms
plot(freq_terms(tokenized_reviews_grouped_by_company)) + labs(title = "After Removing Default Stop Words")
```




```
# Creating a data frame for custom words
custom_stop_words_a <- c("service", "travel", "day", "company", "trip", "cust
omer", "people", "lot", "feel", "highly", "vrbo")
custom_stop_words_dfa <- data.frame(word = custom_stop_words_a, lexicon = rep
("custom", length(custom_stop_words_a)))

# Removing the custom stop words and checking the end result
tokenized_reviews_grouped_by_company <- tokenized_reviews_grouped_by_company
%>% anti_join(custom_stop_words_dfa)
plot(freq_terms(tokenized_reviews_grouped_by_company)) + labs(title = "After
Removing All Stop Words")
```

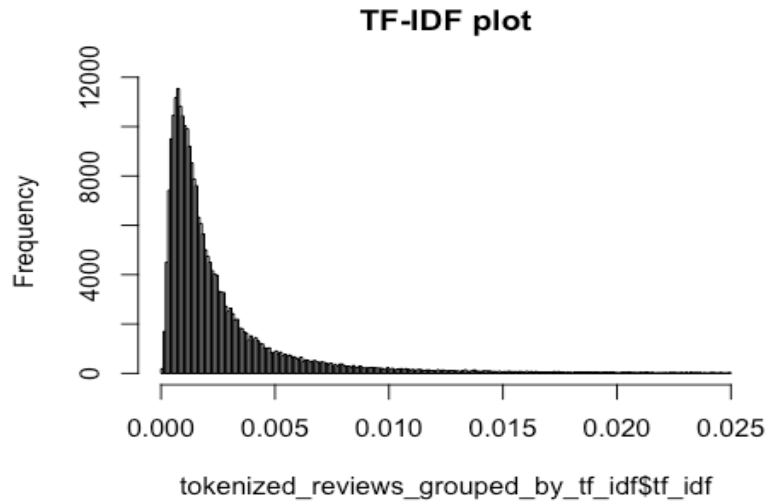



```
# Calculating tf-idf using the company as the document level
tokenized_reviews_grouped_by_tf_idf <- tokenized_reviews_grouped_by_company %
>%
  bind_tf_idf(word, agency_id, n)

# Plotting the distribution of tf-idf
hist(tokenized_reviews_grouped_by_tf_idf$tf_idf, breaks = 200, main = "TF-IDF
plot")
```

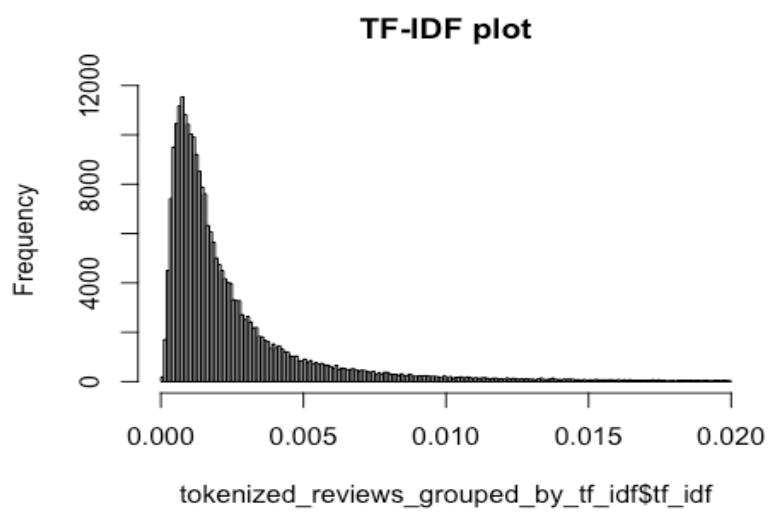



```
tokenized_reviews_grouped_by_tf_idf <- tokenized_reviews_grouped_by_tf_idf %>
%
  filter(tf_idf < 0.025)
```

```r
hist(tokenized_reviews_grouped_by_tf_idf$tf_idf, breaks = 200, main = "TF-IDF
plot")
```




```r
# From the plot, it is observed that the cut-off value is at 0.008
tokenized_reviews_grouped_by_tf_idf <- tokenized_reviews_grouped_by_tf_idf %>%
  filter(tf_idf < 0.020)

hist(tokenized_reviews_grouped_by_tf_idf$tf_idf,breaks = 200, main = "TF-IDF
plot")
```



Stop Word Removal Based on TF-IDF

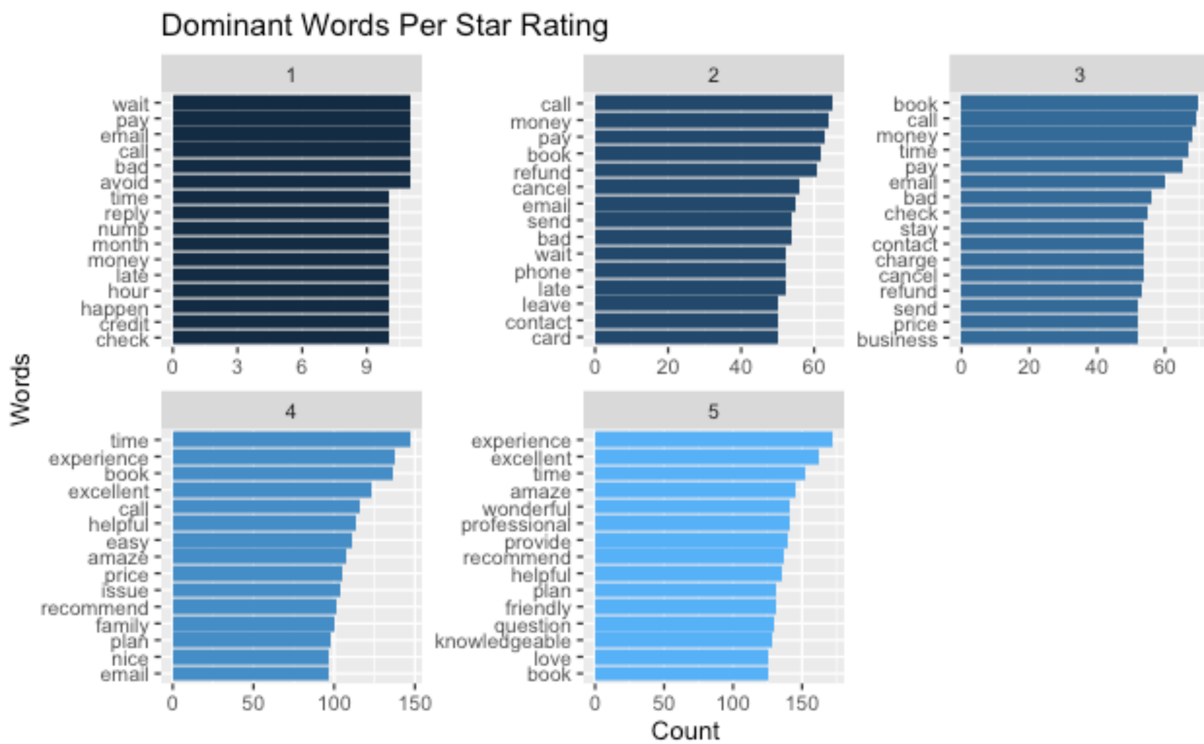
```r
# Also, it is observed that in order to remove very common terms, those with
tf-idf < 0.00025 should be removed
```

```
tokenized_reviews_grouped_by_tf_idf <- tokenized_reviews_grouped_by_tf_idf %>%
%
  filter(tf_idf > 0.0006)

tokenized_reviews_grouped_by_tf_idf %>% group_by(word) %>%
  summarise(total = n()) %>%
  arrange(desc(total)) %>%
  slice_max(total, n = 50)

## # A tibble: 50 × 2
##    word       total
##    <chr>      <int>
##  1 time         421
##  2 experience   409
##  3 book         401
##  4 excellent    354
##  5 call         345
##  6 amaze        312
##  7 helpful      310
##  8 pay          304
##  9 family       300
## 10 receive      297
## # … with 40 more rows

# It is noticed that "receive", "star", "week", "dollar", and "extremely" can
also be stop words
custom_dictionary_a <- custom_stop_words_dfa %>%
  rbind(word = "receive", lexicon = "custom",
        word = "star", lexicon = "custom",
        word = "week", lexicon = "custom",
        word = "dollar", lexicon = "custom") %>%
  rbind(stop_words) %>%
  rbind(custom_stop_words_dfa)

tokenized_reviews_grouped_by_tf_idf <- tokenized_reviews_grouped_by_tf_idf %>
%
  anti_join(custom_dictionary_a)
```

Bag-of-Words Analysis

The reviews are unnested into tokens and analysed to get informative words

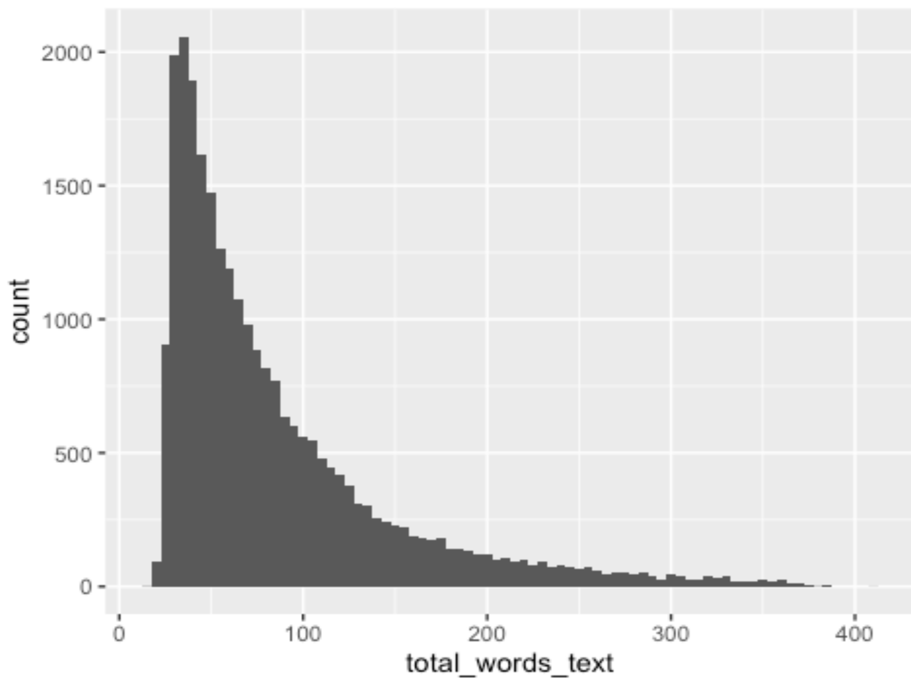
```
# Extracting agency IDs and overall ratings
for_overall <- travel_all %>% select(., c(1, 4)) %>% distinct()

# Adding the overall ratings
tokenized_reviews_grouped_by_company <- tokenized_reviews_grouped_by_company
%>%
  left_join(for_overall)

# Finding the dominant words per overall rating
dominant_words_per_star <- tokenized_reviews_grouped_by_company %>%
  right_join(tokenized_reviews_grouped_by_tf_idf) %>%
  anti_join(custom_dictionary_a) %>%
  group_by(overall_rating) %>%
  count(word) %>%
  rename(count = n) %>%
  slice_max(count, n = 15) %>%
  arrange(desc(count)) %>%
  ungroup()

# Visaulising the 15 dominant words per rating
ggplot(dominant_words_per_star, aes(x = count, y = reorder_within(word, count
, overall_rating), fill = overall_rating)) +
  geom_col(show.legend = FALSE) + facet_wrap(~ overall_rating, nrow = 2, scal
es = "free") +
  labs(x = "Count", y = "Words", title = "Dominant Words Per Star Rating") +
scale_y_reordered()
```



- It can be seen that while “bad”, “late”, and “refund” appears among the dominant words for lower star ratings, it is not apparent for the other ratings. On higher star ratings, words like “excellent”, “amaze”, “wonderful”, “professional”, and “helpful” are appearing. On all star rating categories, “time” appears as one of the dominant words, which indicates that it is an important aspect when people are writing reviews.

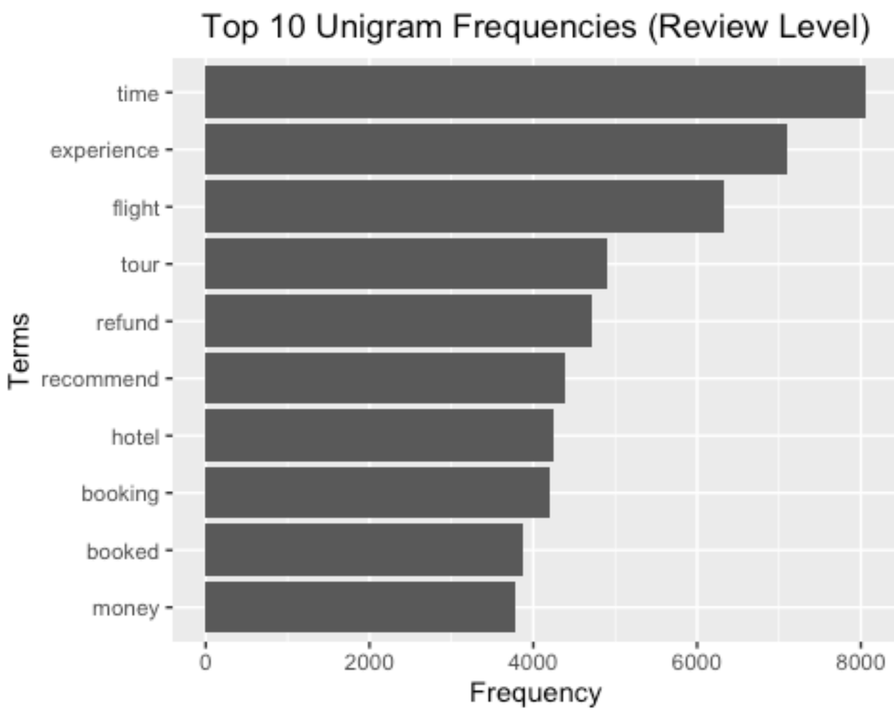
Unigrams and n-grams For Ungrouped Reviews

```
# Counting the number of total words in the collapsed variable of review and
its summary
travel_all$total_words_text <- stringr::str_count(travel_all$summary_review,
"\\S+")

travel_all %>%
  ggplot(., aes(total_words_text)) +
  geom_histogram(binwidth = 5)
```

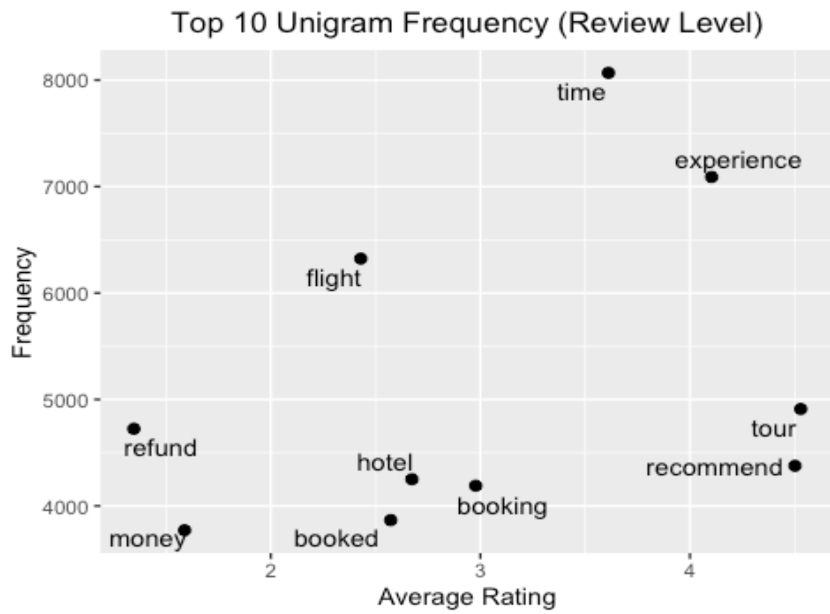


Unigrams

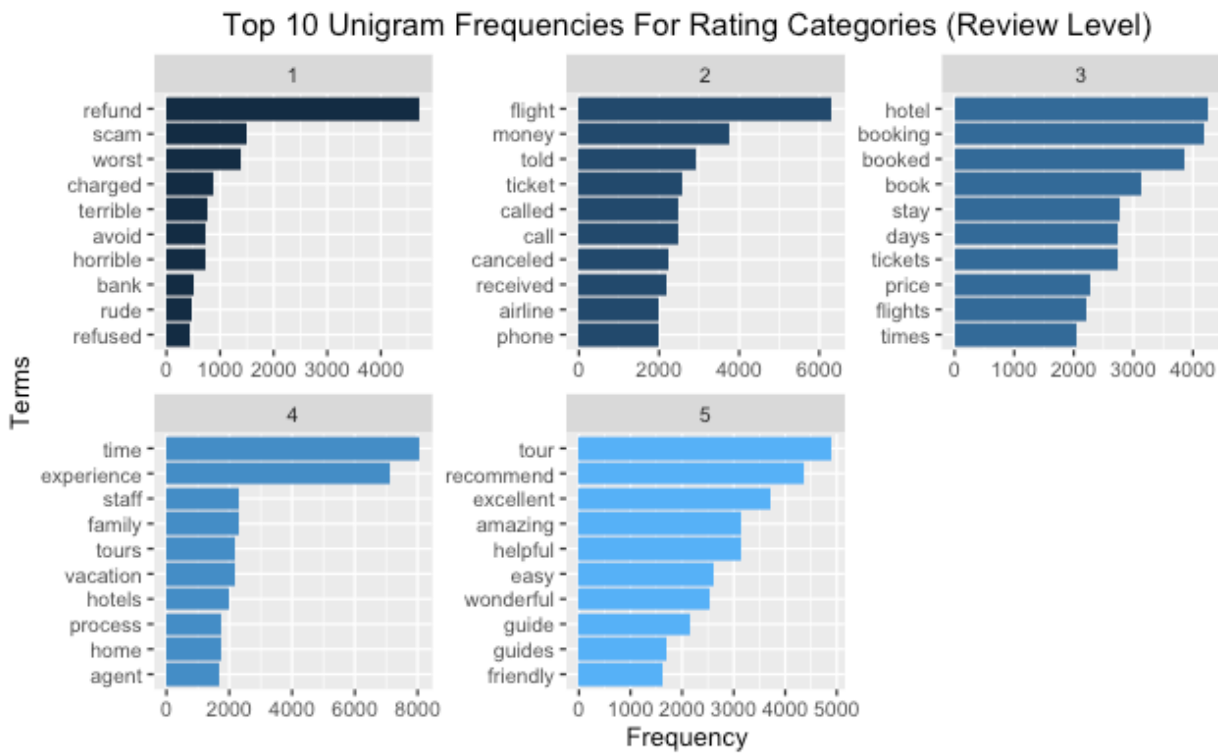
```
# It appears that no filtering is required based on the number of words

# Tokenizing the reviews and their summaries and removing the custom stop wor
ds for unigrams
unigram <- travel_all %>%
  unnest_tokens(word, summary_review, token = "ngrams", n = 1) %>%
  anti_join(custom_dictionary_a)

# Finding the top 10 unigrams and showing their frequencies as well as the av
erage ratings they were used for
unigram_top10 <- unigram %>%
  group_by(word) %>%
  summarise(Frequency = n(),
            Average_Rating = mean(star)) %>%
  top_n(10, Frequency) %>%
  arrange(desc(Frequency))

# Visualising the top 10 unigram frequency for ungrouped reviews
unigram_top10 %>%
  ggplot(aes(Frequency, reorder(word, Frequency))) +
  geom_col() +
  labs(title = "Top 10 Unigram Frequencies (Review Level)",
       x = "Frequency",
       y = "Terms") +
  theme(plot.title = element_text(hjust = 0.5))
```




```
# Visualising the top 10 unigrams based on their frequencies and the average ratings
unigram_top10 %>%
  ggplot(aes(Average_Rating, Frequency)) +
  geom_point(size = 2) +
  geom_text_repel(aes(label = word), max.overlaps = 15) +
  labs(title = "Top 10 Unigram Frequency (Review Level)",
       x = "Average Rating",
       y = "Frequency") +
  theme(plot.title = element_text(hjust = 0.5))
```



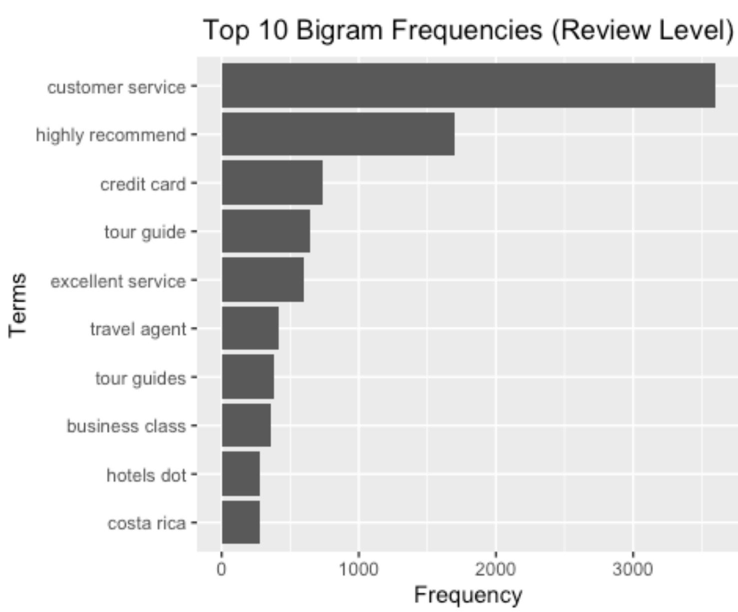

```
# Visualising the top 10 unigrams for each rating
unigram %>%
  group_by(word) %>%
  summarise(n = n(),
            avg_star = round(mean(star))) %>%
  arrange(desc(n)) %>%
  group_by(avg_star) %>%
  slice_max(n, n = 10) %>%
  ungroup() %>%
  ggplot(aes(n, fct_reorder(word, n), fill = avg_star)) +
  geom_col(show.legend = FALSE) +
  labs(title = "Top 10 Unigram Frequencies For Rating Categories (Review Leve
l)",
         x = "Frequency",
         y = "Terms") +
  facet_wrap(~ avg_star, nrow = 2, scales = "free") +
  theme(plot.title = element_text(hjust = 0.5))
```



Bigrams

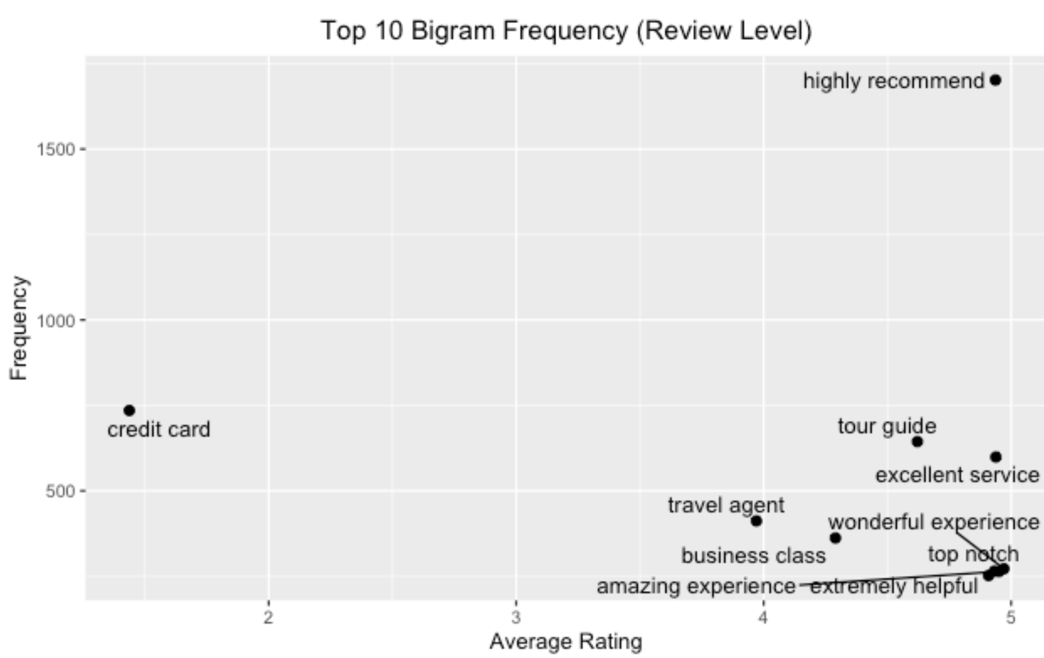
```
# Tokenizing the reviews and their summaries as bigrams
bigram <- travel_all %>%
  unnest_tokens(word, summary_review, token = "ngrams", n = 2)

# Adding indices for each row to separate the words, so the stop words can be
removed
bigram$index <- seq.int(nrow(bigram))

# Separating the bigrams
bigrams_separated <- bigram %>%
  separate(word, c("word1", "word2"), sep = " ") %>%
  select("index", "word1","word2")

# Adding the separated versions of bigrams to the data frame
bigram <- bigram %>%
  left_join(bigrams_separated, by = "index")

# Since words that have been a stop word for unigrams can be more informative
and specific for bigrams and trigrams, a separate custom stop word dictionary
is created
custom_stop_words_ngram <- data.frame()

# Adding the base level stop words to the dictionary
custom_dictionary_ngram <- custom_stop_words_ngram %>%
  rbind(stop_words)
```

```r
# Removing the instances where the bigram contains a stop word from the custo
m ngram dictionary
bigram <- bigram %>%
  filter(!word1 %in% custom_dictionary_ngram$word) %>%
  filter(!word2 %in% custom_dictionary_ngram$word)

# Finding the top 10 bigrams and showing their frequencies as well as the ave
rage ratings they were used for
bigram_top10 <- bigram %>%
  group_by(word) %>%
  summarise(Frequency = n(),
            Average_Rating = mean(star)) %>%
  top_n(10, Frequency) %>%
  arrange(desc(Frequency))

# Visualising the top 10 bigram frequency for ungrouped reviews
bigram_top10 %>%
  ggplot(aes(Frequency, reorder(word, Frequency))) +
  geom_col() +
  labs(title = "Top 10 Bigram Frequencies (Review Level)",
       x = "Frequency",
       y = "Terms") +
  theme(plot.title = element_text(hjust = 0.5))
```

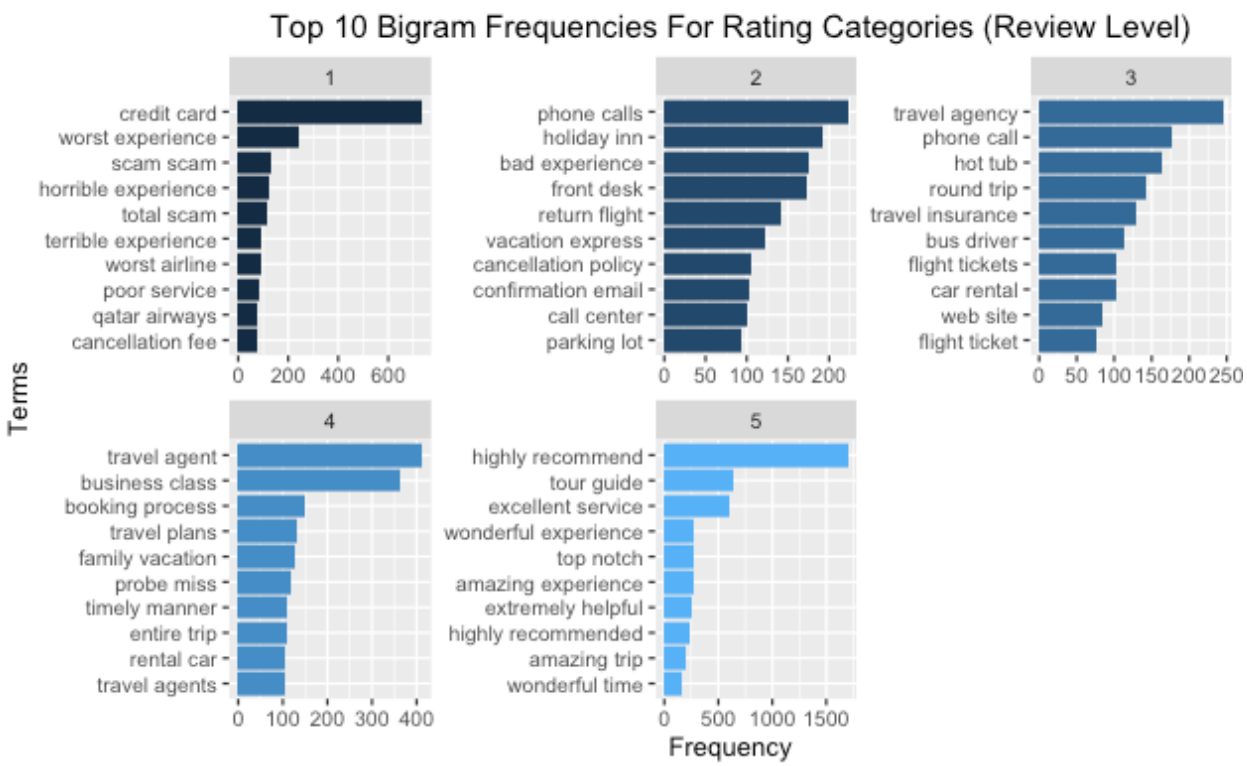



```r
# Adding custom stop words for bigrams and trigrams
custom_stop_words_ngram1 <- c("guides", "costa", "rica", "american", "hong",
"kong", "dot", "customer", "due", "cathay", "pacific", "sky", "dollar", "mere
dith", "lodging", "multiple", "proble", "las", "vegas", "ago", "diamond", "re
sorts", "company")
```

```
custom_stop_words_ngram <- data.frame(word = custom_stop_words_ngram1, lexico
n = rep("custom", length(custom_stop_words_ngram1)))

custom_dictionary_ngram <- custom_stop_words_ngram %>%
  rbind(stop_words)

# Removing the stop words from bigrams and the separated bigram data frame wh
ich will be used later on
bigram <- bigram %>%
  filter(!word1 %in% custom_dictionary_ngram$word) %>%
  filter(!word2 %in% custom_dictionary_ngram$word)

bigrams_separated <- bigrams_separated %>%
  filter(!word1 %in% custom_dictionary_ngram$word) %>%
  filter(!word2 %in% custom_dictionary_ngram$word)

# Counting the bigrams
bigrams_count <- bigrams_separated %>%
  count(word1, word2, sort = TRUE)

# Visualising the top 10 bigrams based on their frequencies and the average r
atings
bigram_top10 %>%
  ggplot(aes(Average_Rating, Frequency)) +
  geom_point(size = 2) +
  geom_text_repel(aes(label = word), max.overlaps = 15) +
  labs(title = "Top 10 Bigram Frequency (Review Level)",
         x = "Average Rating",
         y = "Frequency") +
  theme(plot.title = element_text(hjust = 0.5))
```



```
# Visualising the top 10 bigrams for each rating
bigram %>%
  group_by(word) %>%
  summarise(n = n(),
            avg_star = round(mean(star))) %>%
  arrange(desc(n)) %>%
  group_by(avg_star) %>%
  slice_max(n, n = 10) %>%
  ungroup() %>%
  ggplot(aes(n, fct_reorder(word, n), fill = avg_star)) +
  geom_col(show.legend = FALSE) +
  labs(title = "Top 10 Bigram Frequencies For Rating Categories (Review Level
)",
         x = "Frequency",
         y = "Terms") +
  facet_wrap(~ avg_star, nrow = 2, scales = "free") +
  theme(plot.title = element_text(hjust = 0.5))
```




```
# Filtering bigrams that repeat at least 100 times to be used for graphing as
sociations
bigram_assossiation <- bigrams_count %>%
  filter(n > 100) %>%
  graph_from_data_frame()

set.seed(1)

arrow_dimensions <- grid::arrow(type = "closed", length = unit(.2, "inches"))
```

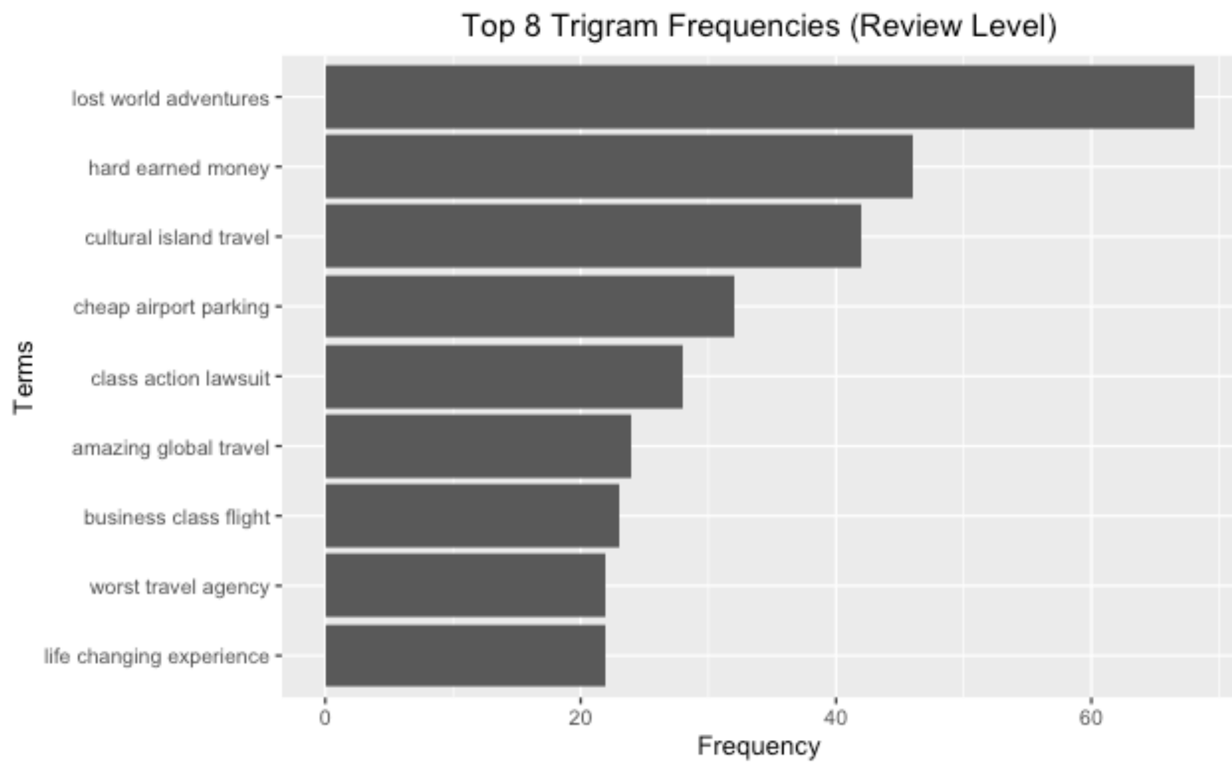
```
# Visualising the word associations on bigram level
ggraph(bigram_assossiation, layout = "fr") +
       geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
                          arrow = arrow_dimensions, end_cap = circle(.07, "in
ches")) +
       geom_node_point(color = "lightgreen", size = 3) +
       geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
       theme_void()
```

- The word associations show the directions of connections that each word has in when pairing up. We can see words like experience and trip have high association with multiple words in the bigrams.

Trigrams

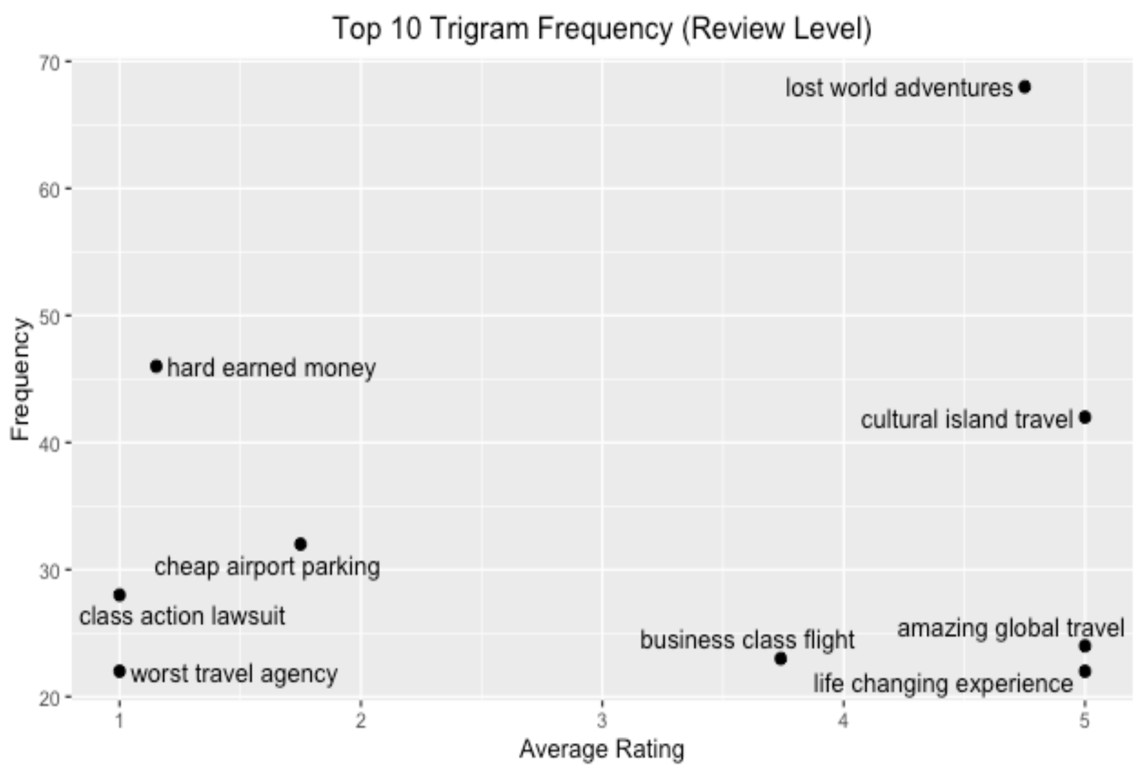
```
# Tokenizing the reviews and their summaries as trigrams
trigram <- travel_all %>%
  unnest_tokens(word, summary_review, token = "ngrams", n = 3)

# Adding indices for each row to separate the words, so the stop words can be
removed
trigram$index <- seq.int(nrow(trigram))

# Separating the trigrams
trigrams_separated <- trigram %>%
  separate(word, c("word1", "word2", "word3"), sep = " ") %>%
```

```r
  select("index", "word1","word2", "word3")

# Adding the separated versions of trigrams to the data frame
trigram <- trigram %>%
  left_join(trigrams_separated, by = "index")

# Since there already is a customised dictionary for bigrams and trigrams, re
moving the stop words from the trigram data frame
trigram <- trigram %>%
  filter(!word1 %in% custom_dictionary_ngram$word) %>%
  filter(!word2 %in% custom_dictionary_ngram$word) %>%
  filter(!word3 %in% custom_dictionary_ngram$word)

# Finding the top 8 trigrams and showing their frequencies as well as the ave
rage ratings they were used for
trigram_top8 <- trigram %>%
  group_by(word) %>%
  summarise(Frequency = n(),
            Average_Rating = mean(star)) %>%
  top_n(8, Frequency) %>%
  arrange(desc(Frequency))

# Visualising the top 8 trigram frequency for ungrouped reviews
trigram_top8 %>%
  ggplot(aes(Frequency, reorder(word, Frequency))) +
  geom_col() +
  labs(title = "Top 8 Trigram Frequencies (Review Level)",
         x = "Frequency",
         y = "Terms") +
  theme(plot.title = element_text(hjust = 0.5))
```



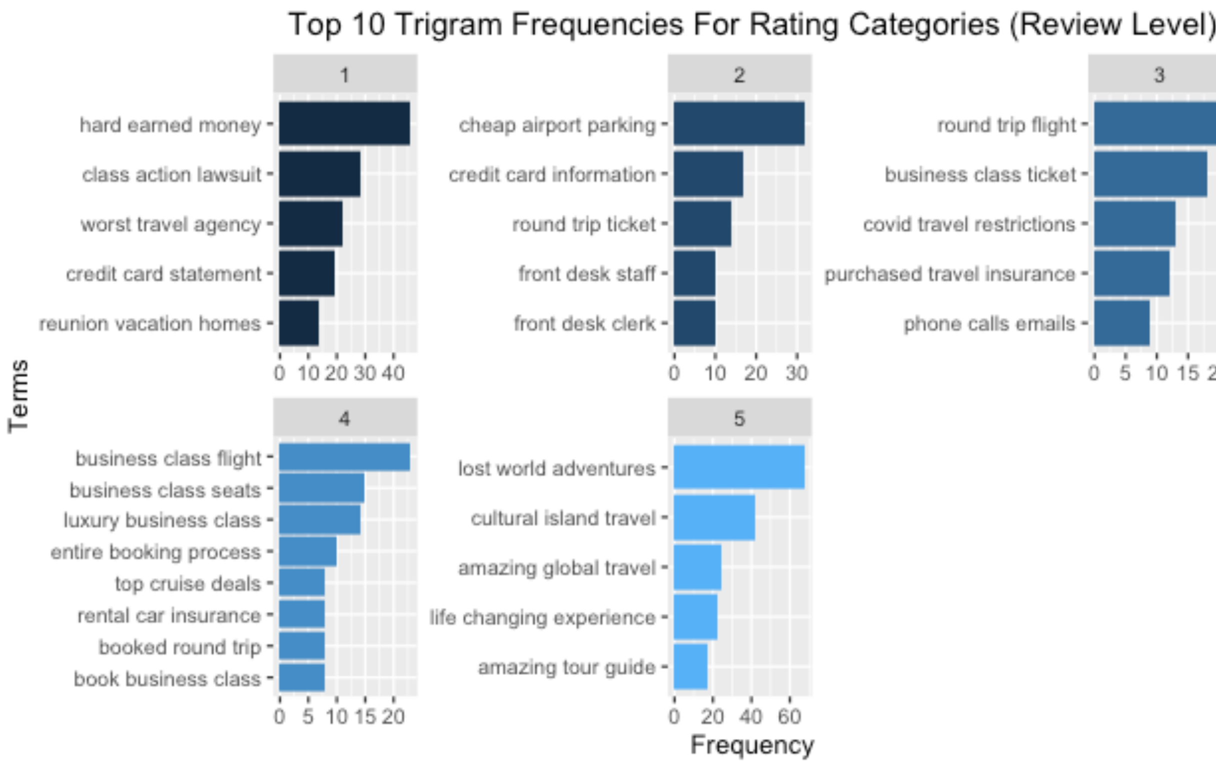

```
# Adding additional custom stop words and storing them separately
custom_stop_words_ngram2 <- c("african", "excellent", "tahiti", "dacey's", "e
u", "cornish", "de", "johnnie", "america", "chicago", "alpha", "safaris", "fl
ights", "scam", "avoid", "horrible", "gbg", "florida", "info", "chi", "roo",
"salt", "magnolia", "starmark", "sitters", "military", "charlene's", "tickets
", "sleep", "customer", "israel", "inn", "pro", "executive", "del", "corolla"
)

custom_stop_words_trigram <- data.frame(word = custom_stop_words_ngram2, lexi
con = rep("custom", length(custom_stop_words_ngram2)))

custom_dictionary_trigram <- custom_stop_words_trigram %>%
  rbind(stop_words)

# Removing the stop words from trigrams and the separated trigram data frame
which will be used later on
trigram <- trigram %>%
  filter(!word1 %in% custom_stop_words_trigram$word) %>%
  filter(!word2 %in% custom_stop_words_trigram$word) %>%
  filter(!word3 %in% custom_stop_words_trigram$word)

trigrams_separated <- trigrams_separated %>%
  filter(!word1 %in% custom_stop_words_trigram$word) %>%
  filter(!word2 %in% custom_stop_words_trigram$word) %>%
  filter(!word3 %in% custom_stop_words_trigram$word)
```

```r
# Counting the trigrams
trigrams_count <- trigrams_separated %>%
  count(word1, word2, word3, sort = TRUE)

# Visualising the top 8 trigrams based on their frequencies and the average r
atings
trigram_top8 %>%
  ggplot(aes(Average_Rating, Frequency)) +
  geom_point(size = 2) +
  geom_text_repel(aes(label = word), max.overlaps = 15) +
  labs(title = "Top 10 Trigram Frequency (Review Level)",
       x = "Average Rating",
       y = "Frequency") +
  theme(plot.title = element_text(hjust = 0.5))
```

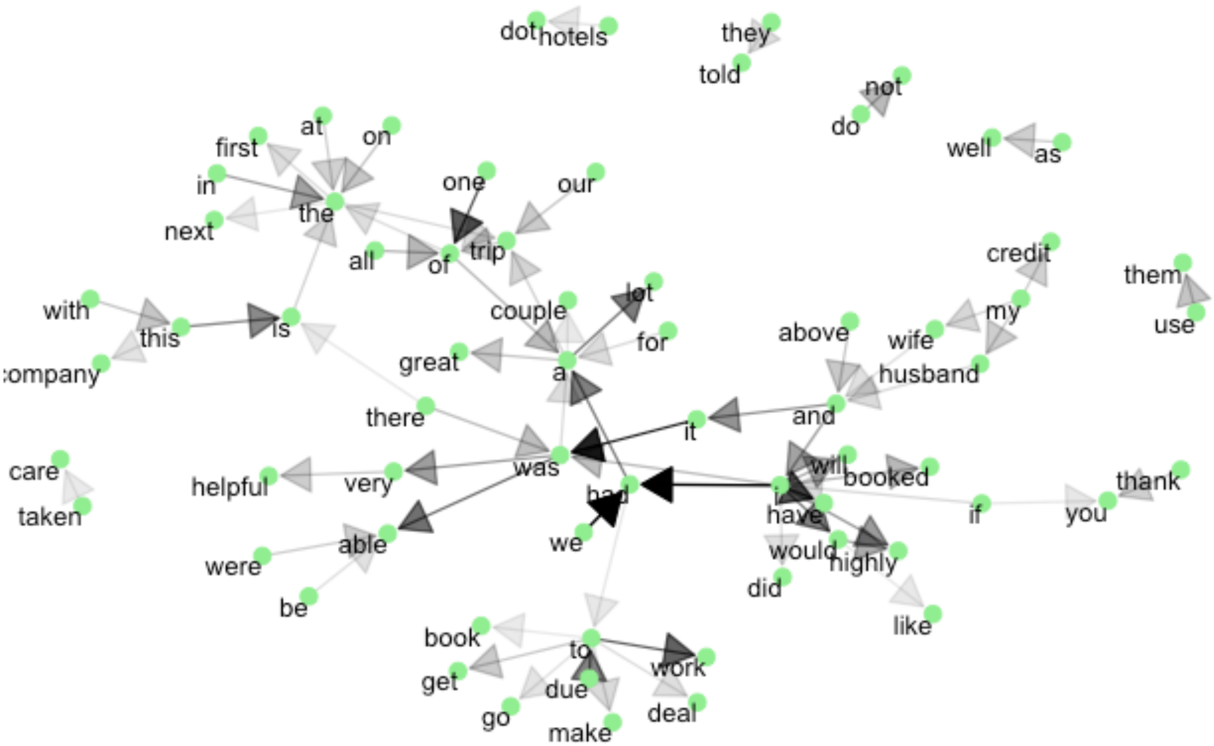



```r
# Visualising the top 8 trigrams for each rating
trigram %>%
  group_by(word) %>%
  summarise(n = n(),
            avg_star = round(mean(star))) %>%
  arrange(desc(n)) %>%
  group_by(avg_star) %>%
  slice_max(n, n = 5) %>%
  ungroup() %>%
  ggplot(aes(n, fct_reorder(word, n), fill = avg_star)) +
```

```
  geom_col(show.legend = FALSE) +
  labs(title = "Top 10 Trigram Frequencies For Rating Categories (Review Leve
l)",
       x = "Frequency",
       y = "Terms") +
  facet_wrap(~ avg_star, nrow = 2, scales = "free") +
  theme(plot.title = element_text(hjust = 0.5))
```




```
# Filtering trigrams that repeat at least 260 times to be used for graphing a
ssociations
trigram_assossiation <- trigrams_count %>%
  filter(n > 260) %>%
  graph_from_data_frame()

set.seed(1)

arrow_dimensions_tri <- grid::arrow(type = "closed", length = unit(.2, "inche
s"))

# Visualising the word associations on trigram level
ggraph(trigram_assossiation, layout = "fr") +
       geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
                      arrow = arrow_dimensions_tri, end_cap = circle(.07,
"inches")) +
       geom_node_point(color = "lightgreen", size = 3) +
       geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
       theme_void()
```



Unigrams and n-grams For Grouped Reviews

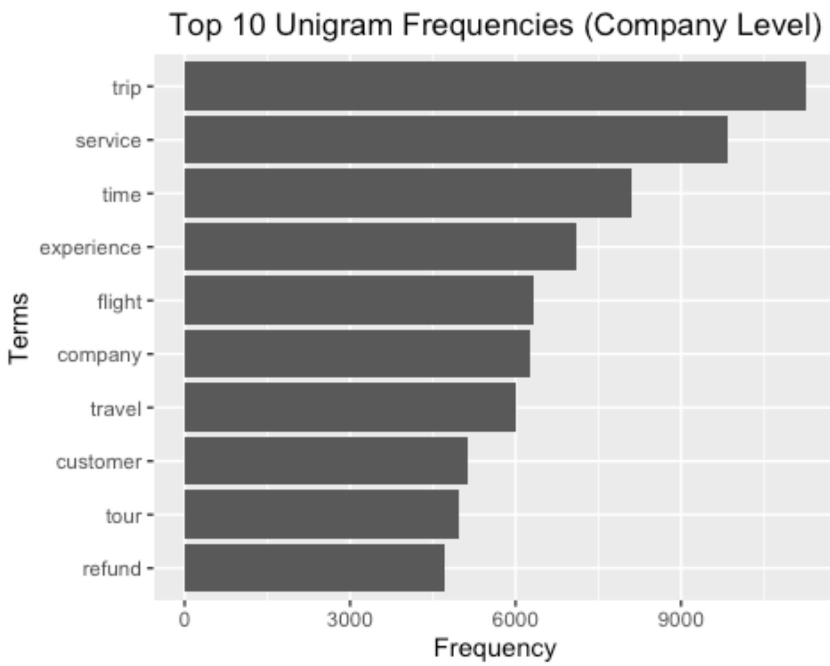
```
# Counting the number of total words in the collapsed variable of review and its summary for each travel agency
text_grouped_by_company$total_words_text <- stringr::str_count(text_grouped_by_company$text_grouped, "\\S+")

# Adding the overall ratings of the travel agencies to the data frame
text_grouped_by_company <- text_grouped_by_company %>%
  left_join(for_overall)
```

Unigrams

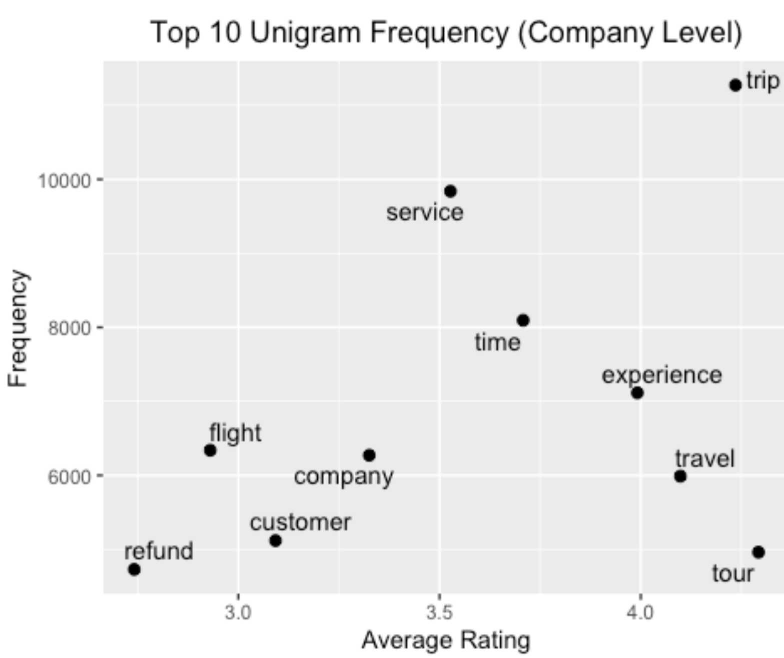
```
# Visualising unigram frequency for grouped reviews
unigram_grouped <- text_grouped_by_company %>%
  unnest_tokens(word, text_grouped, token = "ngrams", n = 1) %>%
  anti_join(stop_words)

# Finding the top 10 unigrams and showing their frequencies as well as the travel agency ratings they were used for
unigram_grouped_top10 <- unigram_grouped %>%
  group_by(word) %>%
  summarise(Frequency = n(),
            Average_Rating = mean(overall_rating)) %>%
  top_n(10, Frequency) %>%
  arrange(desc(Frequency))
```

```r
# Adding custom stop words to be removed from the unigrams
custom_stop_words2 <- c("travel", "service", "company", "customer", "booked")
custom_stop_words_df2 <- data.frame(word = custom_stop_words2, lexicon = rep(
"custom", length(custom_stop_words2)))
custom_dictionary_for_grouped_a <- custom_stop_words_df2 %>%
  rbind(word = "swiss", lexicon = "custom",
        word = "jetblue", lexicon = "custom",
        word = "mmt", lexicon = "custom",
        word = "makemytrip", lexicon = "custom",
        word = "wizzair", lexicon = "custom",
        word = "adele", lexicon = "custom",
        word = "wizz", lexicon = "custom",
        word = "protime", lexicon = "custom",
        word = "rs", lexicon = "custom",
        word = "inn", lexicon = "custom",
        word = "blue", lexicon = "custom",
        word = "qatar", lexicon = "custom",
        word = "ctrip", lexicon = "custom",
        word = "singapore", lexicon = "custom",
        word = "kiwi", lexicon = "custom",
        word = "southwest", lexicon = "custom",
        word = "msc", lexicon = "custom",
        word = "dollar", lexicon = "custom",
        word = "told", lexicon = "custom",
        word = "people", lexicon = "custom",
        word = "africa", lexicon = "custom",
        word = "croatia", lexicon = "custom",
        word = "cuba", lexicon = "custom",
        word = "ireland", lexicon = "custom",
        word = "safari", lexicon = "custom",
        word = "days", lexicon = "custom",
        word = "tickets", lexicon = "custom",
        word = "turkish", lexicon = "custom",
        word = "western", lexicon = "custom",
        word = "liftopia", lexicon = "custom",
        word = "bean", lexicon = "custom",
        word = "walker", lexicon = "custom",
        word = "airways", lexicon = "custom",
        word = "simplio", lexicon = "custom",
        word = "zach", lexicon = "custom",
        word = "swiz", lexicon = "custom",
        word = "protimetours", lexicon = "custom",
        word = "mapmaker", lexicon = "custom",
        word = "coldplay", lexicon = "custom",
        word = "wembley", lexicon = "custom",
        word = "watchdog", lexicon = "custom",
        word = "oyo", lexicon = "custom",
        word = "regency", lexicon = "custom",
        word = "luton", lexicon = "custom",
```

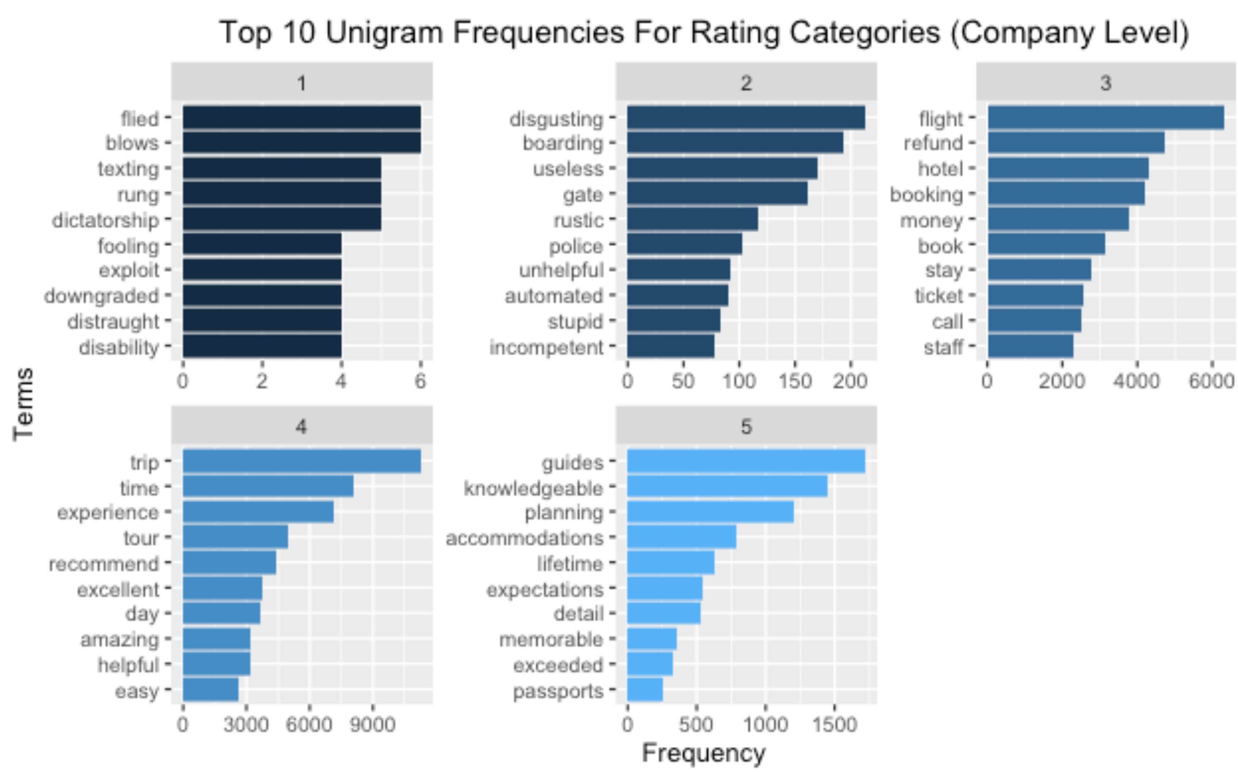
```
        word = "chf", lexicon = "custom",
        word = "dwayne", lexicon = "custom",
        word = "david", lexicon = "custom",
        word = "called", lexicon = "custom",
        word = "meredith", lexicon = "custom",
        word = "ll", lexicon = "custom",
        word = "dc", lexicon = "custom",
        word = "transavia", lexicon = "custom",
        word = "tesco", lexicon = "custom",
        word = "eurorest", lexicon = "custom",
        word = "lodz", lexicon = "custom",
        word = "lycafly", lexicon = "custom",
        word = "makemy", lexicon = "custom",
        word = "martires", lexicon = "custom",
        word = "nn", lexicon = "custom",
        word = "sau", lexicon = "custom",
        word = "seatles", lexicon = "custom",
        word = "sheeran", lexicon = "custom",
        word = "sst", lexicon = "custom",
        word = "swissair", lexicon = "custom",
        word = "waittime", lexicon = "custom",
        word = "koukoullis", lexicon = "custom",
        word = "surya", lexicon = "custom",
        word = "egyptair", lexicon = "custom",
        word = "istanbul", lexicon = "custom",
        word = "aa", lexicon = "custom",
        word = "spokane", lexicon = "custom",
        word = "johnnie", lexicon = "custom",
        word = "pegasus", lexicon = "custom",
        word = "doha", lexicon = "custom",
        word = "peru", lexicon = "custom") %>%
  rbind(stop_words)

# Removing the stop words
unigram_grouped <- unigram_grouped %>%
  anti_join(custom_dictionary_for_grouped_a)

# Visualising the top 10 unigram frequency for ungrouped reviews
unigram_grouped_top10 %>%
  ggplot(aes(Frequency, reorder(word, Frequency))) +
  geom_col() +
  labs(title = "Top 10 Unigram Frequencies (Company Level)",
       x = "Frequency",
       y = "Terms") +
  theme(plot.title = element_text(hjust = 0.5))
```




```
# Visualising the top 10 unigrams based on their frequencies and the overall
ratings
unigram_grouped_top10 %>%
  ggplot(aes(Average_Rating, Frequency)) +
  geom_point(size = 2) +
  geom_text_repel(aes(label = word), max.overlaps = 15) +
  labs(title = "Top 10 Unigram Frequency (Company Level)",
       x = "Average Rating",
       y = "Frequency") +
  theme(plot.title = element_text(hjust = 0.5))
```



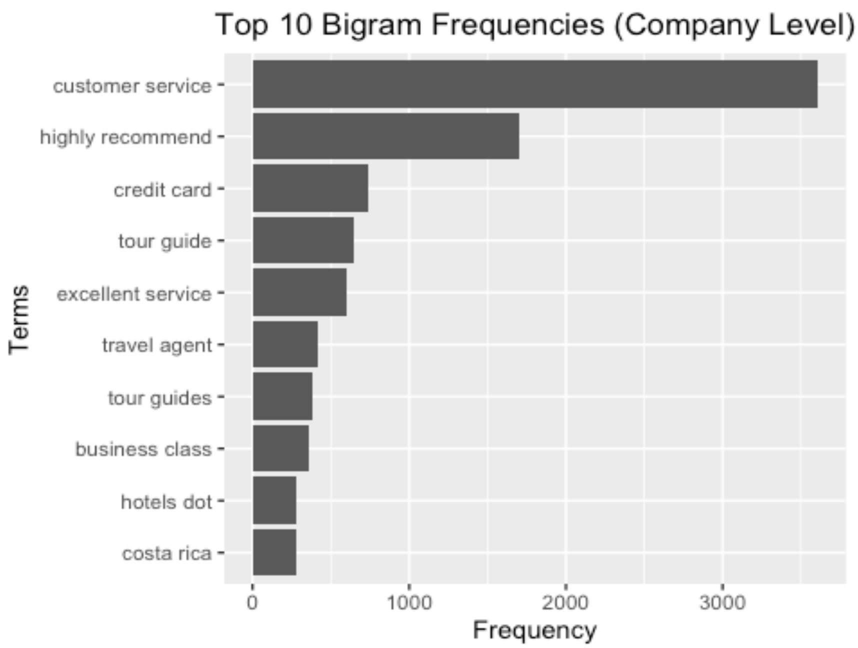
```
# Visualising the top 10 unigrams for each rating
unigram_grouped %>%
  group_by(word) %>%
  summarise(n = n(),
            avg_rating = round(mean(overall_rating))) %>%
  arrange(desc(n)) %>%
  group_by(avg_rating) %>%
  slice_max(n, n = 10) %>%
  ungroup() %>%
  ggplot(aes(n, fct_reorder(word, n), fill = avg_rating)) +
  geom_col(show.legend = FALSE) +
  labs(title = "Top 10 Unigram Frequencies For Rating Categories (Company Lev
el)",
         x = "Frequency",
         y = "Terms") +
  facet_wrap(~ avg_rating, nrow = 2, scales = "free") +
  theme(plot.title = element_text(hjust = 0.5))
```



Bigrams

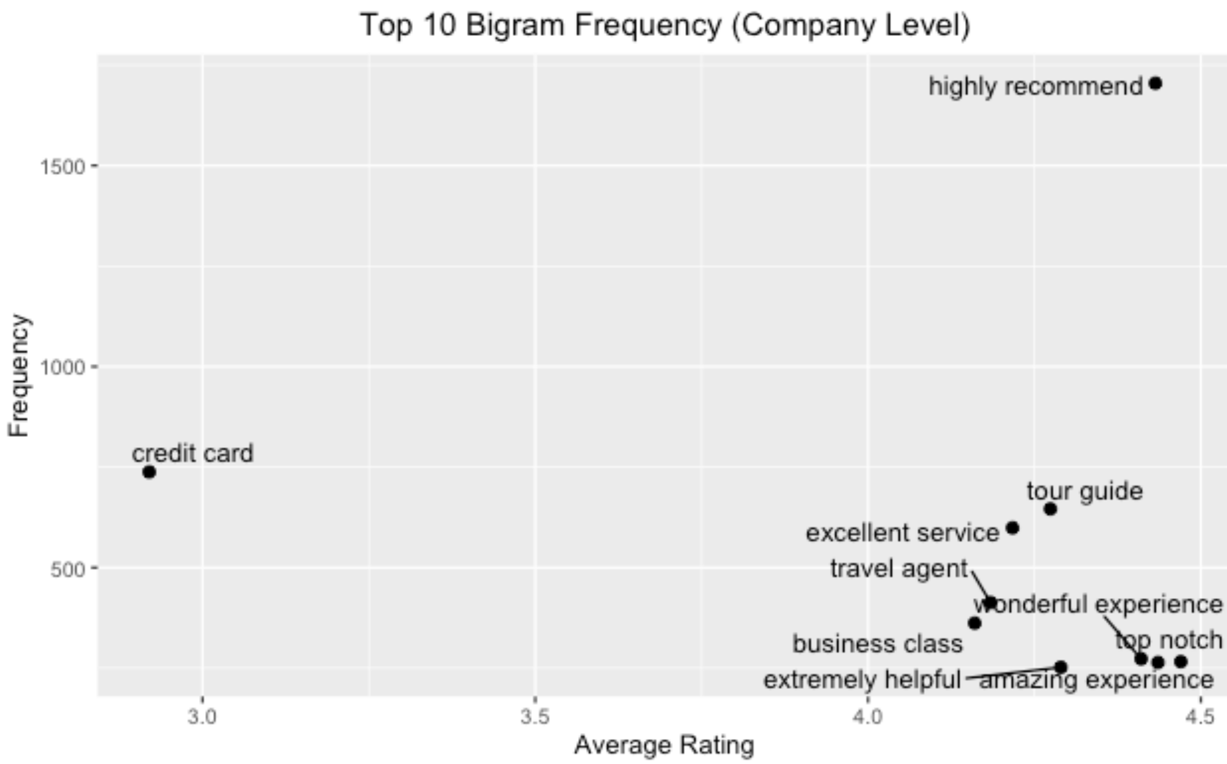
```
# Tokenizing the reviews and their summaries as bigrams for grouped reviews
bigram_grouped <- text_grouped_by_company %>%
  unnest_tokens(word, text_grouped, token = "ngrams", n = 2)

# Adding indices for each row to separate the words, so the stop words can be
removed
bigram_grouped$index <- seq.int(nrow(bigram_grouped))
```

```r
# Separating the bigrams
bigrams_company_separated <- bigram_grouped %>%
  separate(word, c("word1", "word2"), sep = " ") %>%
  select("index", "word1","word2")

# Adding the separated versions of bigrams to the data frame
bigram_grouped <- bigram_grouped %>%
  left_join(bigrams_company_separated, by = "index")

# Since words that have been a stop word for unigrams can be more informative
and specific for bigrams and trigrams, a separate custom stop word dictionary
is created
custom_dictionary_ngram_grouped <- data.frame()

# Adding the base level stop words to the dictionary
custom_dictionary_ngram_grouped <- custom_dictionary_ngram_grouped %>%
  rbind(stop_words)

# Removing the instances where the bigram contains a stop word from the custo
m ngram dictionary
bigram_grouped <- bigram_grouped %>%
  filter(!word1 %in% custom_dictionary_ngram_grouped$word) %>%
  filter(!word2 %in% custom_dictionary_ngram_grouped$word)

# Finding the top 10 bigrams and showing their frequencies as well as the ave
rage ratings they were used for
bigram_company_top10 <- bigram_grouped %>%
  group_by(word) %>%
  summarise(Frequency = n(),
            Average_Rating = mean(overall_rating)) %>%
  top_n(10, Frequency) %>%
  arrange(desc(Frequency))

# Visualising the top 10 bigram frequency for grouped reviews
bigram_company_top10 %>%
  ggplot(aes(Frequency, reorder(word, Frequency))) +
  geom_col() +
  labs(title = "Top 10 Bigram Frequencies (Company Level)",
         x = "Frequency",
         y = "Terms") +
  theme(plot.title = element_text(hjust = 0.5))
```

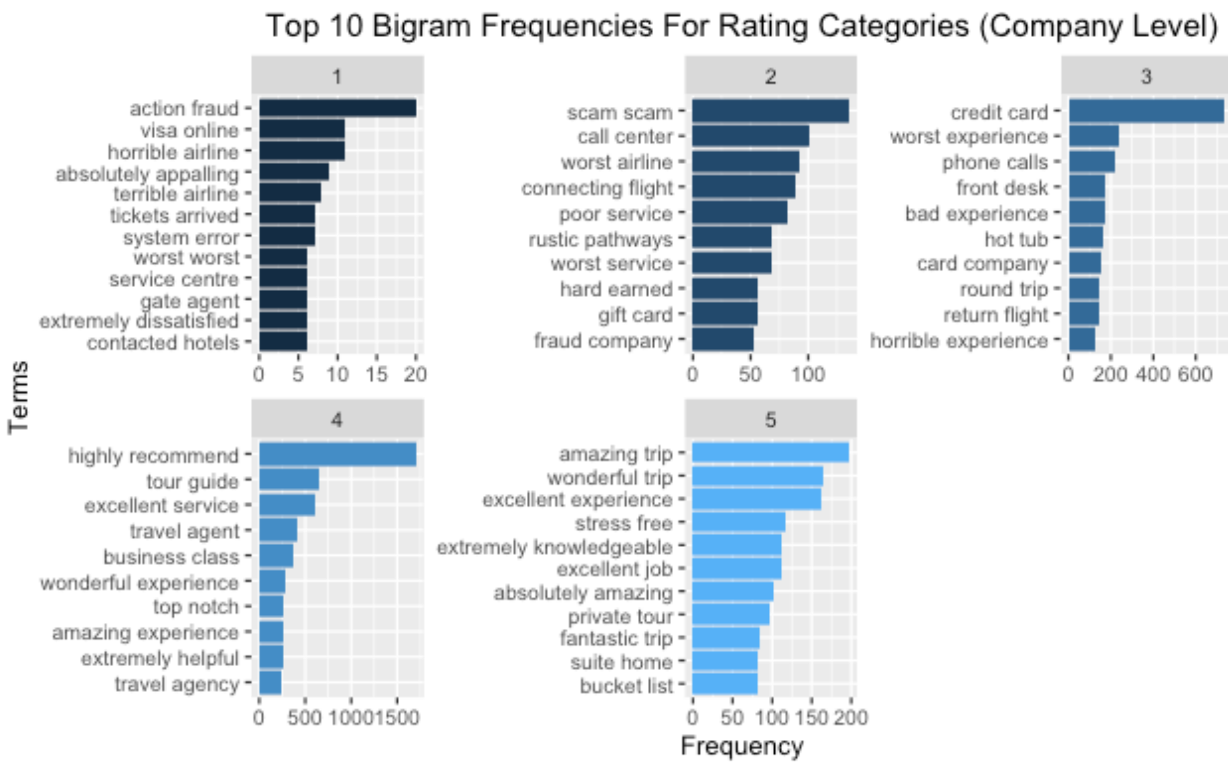



```
# Adding custom stop words for bigrams and trigrams
custom_stop_words_ngram3 <- c("guides", "costa", "rica", "american", "hong",
"kong", "dot", "customer", "due", "cathay", "pacific", "sky", "dollar", "mere
dith", "lodging", "multiple", "proble", "las", "vegas", "ago", "diamond", "re
sorts", "africa", "irish", "inn", "singapore", "turkish", "ll", "qatar", "jet
", "swiss", "pro", "wizz", "adele", "protime", "kenya", "african", "sa", "ita
ly", "johnnie", "choice", "tours", "wiz", "surya", "jetblue", "southern")

custom_stop_words_ngram_grouped <- data.frame(word = custom_stop_words_ngram3
, lexicon = rep("custom", length(custom_stop_words_ngram3)))

custom_dictionary_ngram_grouped <- custom_stop_words_ngram_grouped %>%
  rbind(stop_words)

# Removing the stop words from bigrams and the separated bigram data frame wh
ich will be used later on
bigram_grouped <- bigram_grouped %>%
  filter(!word1 %in% custom_dictionary_ngram_grouped$word) %>%
  filter(!word2 %in% custom_dictionary_ngram_grouped$word)

bigrams_company_separated <- bigrams_company_separated %>%
  filter(!word1 %in% custom_dictionary_ngram_grouped$word) %>%
  filter(!word2 %in% custom_dictionary_ngram_grouped$word)

# Counting the bigrams
bigrams_grouped_count <- bigrams_company_separated %>%
  count(word1, word2, sort = TRUE)

# Visualising the top 10 bigrams based on their frequencies and the overall r
atings
```

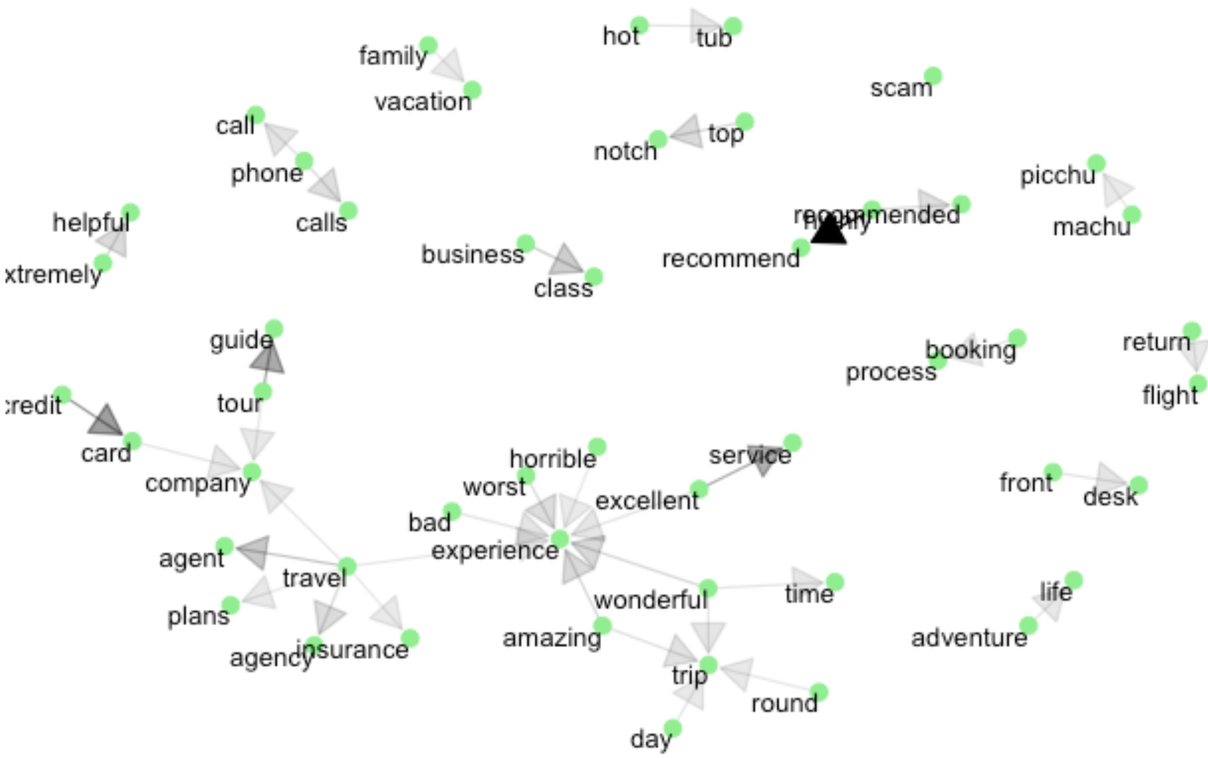
```r
bigram_company_top10 %>%
  ggplot(aes(Average_Rating, Frequency)) +
  geom_point(size = 2) +
  geom_text_repel(aes(label = word), max.overlaps = 15) +
  labs(title = "Top 10 Bigram Frequency (Company Level)",
         x = "Average Rating",
         y = "Frequency") +
  theme(plot.title = element_text(hjust = 0.5))
```




```r
# Visualising the top 10 bigrams for each rating
bigram_grouped %>%
  group_by(word) %>%
  summarise(n = n(),
            avg_rating = round(mean(overall_rating))) %>%
  arrange(desc(n)) %>%
  group_by(avg_rating) %>%
  slice_max(n, n = 10) %>%
  ungroup() %>%
  ggplot(aes(n, fct_reorder(word, n), fill = avg_rating)) +
  geom_col(show.legend = FALSE) +
  labs(title = "Top 10 Bigram Frequencies For Rating Categories (Company Leve
l)",
         x = "Frequency",
         y = "Terms") +
  facet_wrap(~ avg_rating, nrow = 2, scales = "free") +
  theme(plot.title = element_text(hjust = 0.5))
```



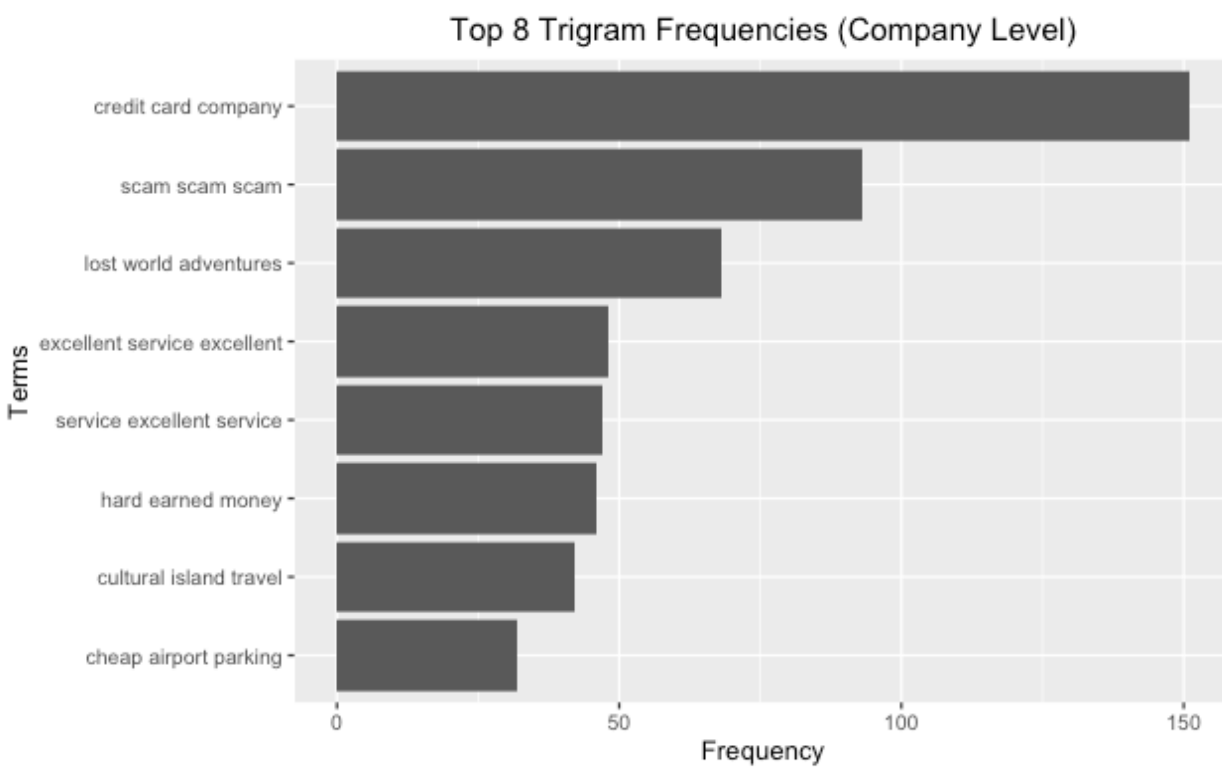

```
# Filtering bigrams that repeat at least 125 times to be used for graphing as
sociations
bigram_company_assossiation <- bigrams_grouped_count %>%
  filter(n > 125) %>%
  graph_from_data_frame()

set.seed(1)

arrow_dimensions_company <- grid::arrow(type = "closed", length = unit(.2, "i
nches"))

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



Trigrams

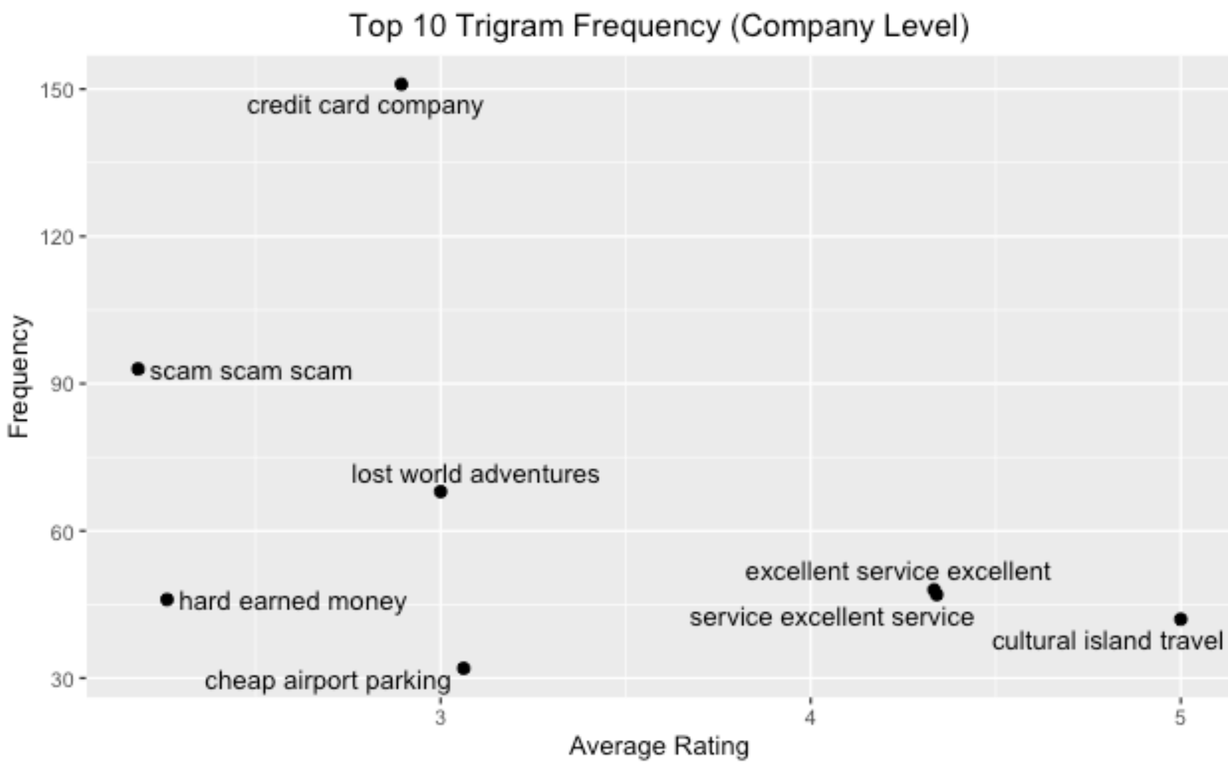
```
# Tokenizing the reviews and their summaries as trigrams for grouped reviews
trigram_grouped <- text_grouped_by_company %>%
  unnest_tokens(word, text_grouped, token = "ngrams", n = 3)

# Adding indices for each row to separate the words, so the stop words can be
removed
trigram_grouped$index <- seq.int(nrow(trigram_grouped))

# Separating the trigrams
trigrams_company_separated <- trigram_grouped %>%
  separate(word, c("word1", "word2", "word3"), sep = " ") %>%
  select("index", "word1","word2", "word3")

# Adding the separated versions of trigrams to the data frame
trigram_grouped <- trigram_grouped %>%
  left_join(trigrams_company_separated, by = "index")

# Since there already is a customised dictionary for bigrams and trigrams, re
moving the stop words from the trigram data frame
trigram_grouped <- trigram_grouped %>%
  filter(!word1 %in% custom_dictionary_ngram_grouped$word) %>%
  filter(!word2 %in% custom_dictionary_ngram_grouped$word) %>%
  filter(!word3 %in% custom_dictionary_ngram_grouped$word)

# Finding the top 8 trigrams and showing their frequencies as well as the ave
```

```r
rage ratings they were used for
trigram_company_top8 <- trigram_grouped %>%
  group_by(word) %>%
  summarise(Frequency = n(),
            Average_Rating = mean(overall_rating)) %>%
  top_n(8, Frequency) %>%
  arrange(desc(Frequency))

# Visualising the top 8 trigram frequency for grouped reviews
trigram_company_top8 %>%
  ggplot(aes(Frequency, reorder(word, Frequency))) +
  geom_col() +
  labs(title = "Top 8 Trigram Frequencies (Company Level)",
       x = "Frequency",
       y = "Terms") +
  theme(plot.title = element_text(hjust = 0.5))
```

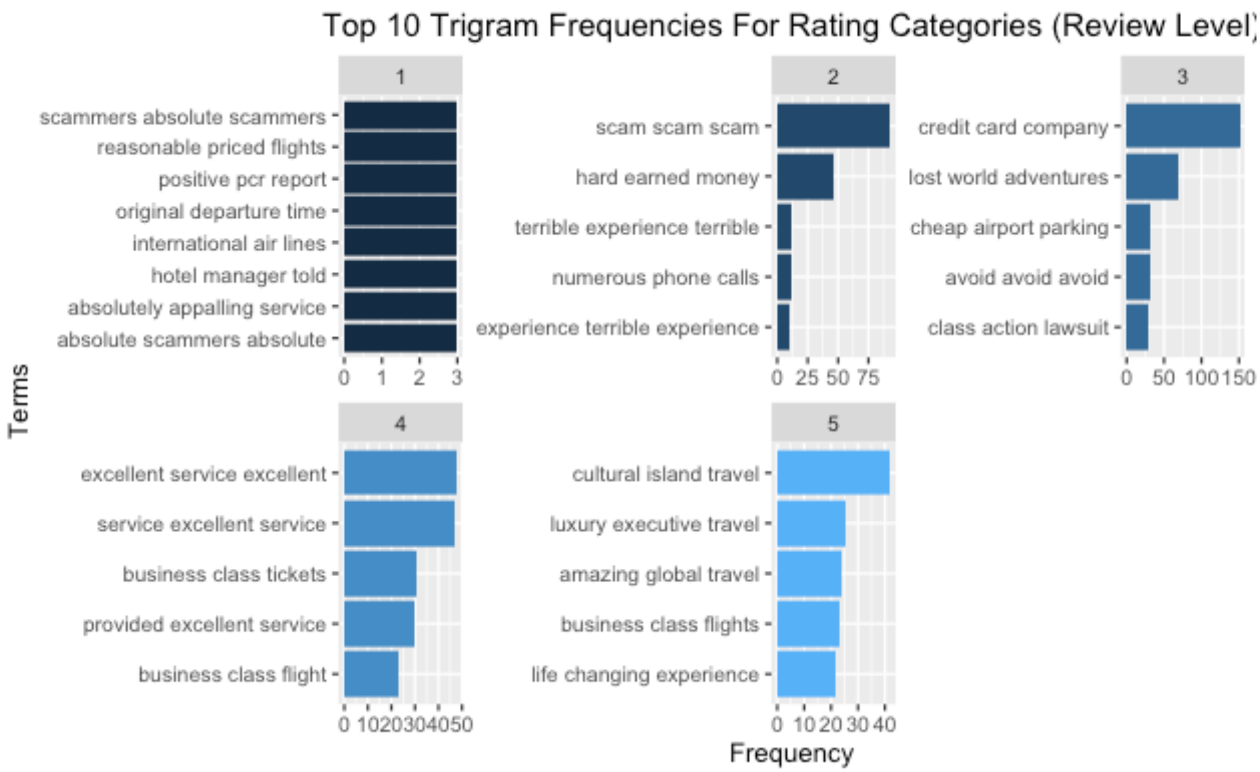



```r
# Adding additional custom stop words and storing them separately
custom_stop_words_ngram4 <- c("safaris", "tahiti", "eu", "chicago", "south",
"use.active", "corolla", "pm", "regency", "nn", "alpha", "")

custom_stop_words_ngram_grouped_trigram <- data.frame(word = custom_stop_word
s_ngram4, lexicon = rep("custom", length(custom_stop_words_ngram4)))

custom_dictionary_grouped_trigram <- custom_stop_words_ngram_grouped_trigram
%>%
  rbind(stop_words)
```

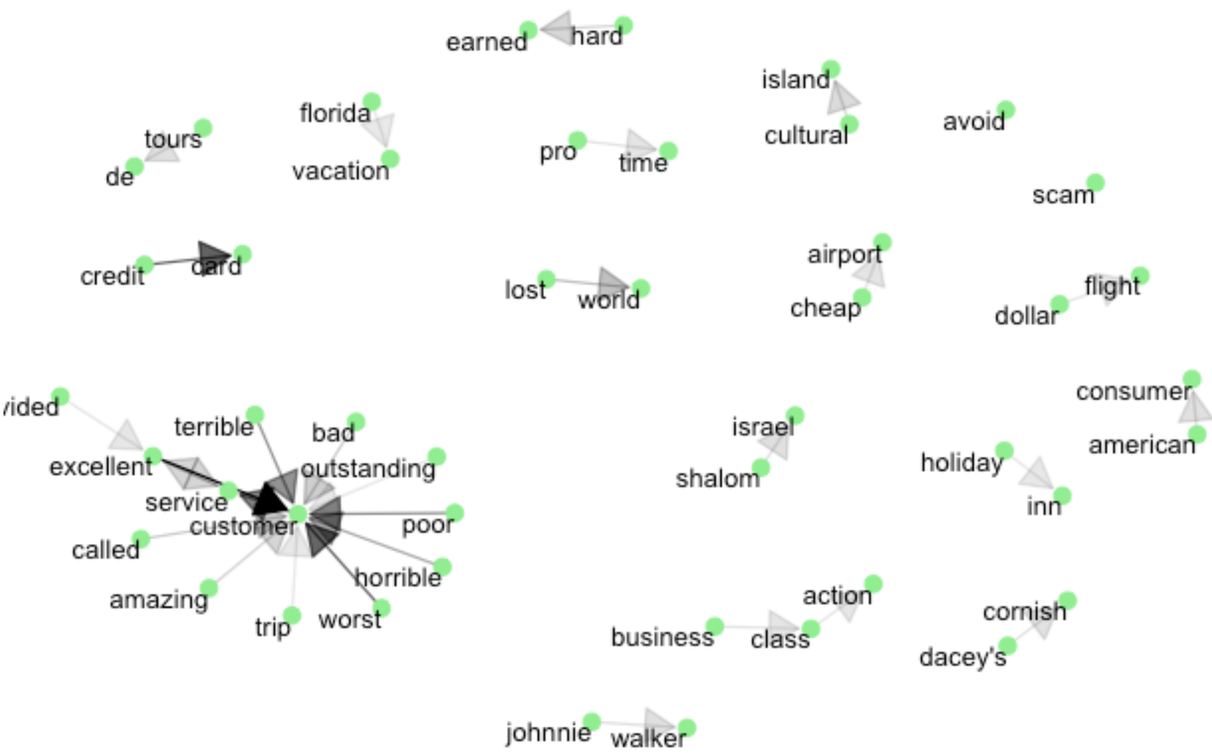
```r
# Removing the stop words from trigrams and the separated trigram data frame
which will be used later on
trigram_grouped <- trigram_grouped %>%
  filter(!word1 %in% custom_dictionary_grouped_trigram$word) %>%
  filter(!word2 %in% custom_dictionary_grouped_trigram$word) %>%
  filter(!word3 %in% custom_dictionary_grouped_trigram$word)

trigrams_company_separated <- trigrams_company_separated %>%
  filter(!word1 %in% custom_dictionary_grouped_trigram$word) %>%
  filter(!word2 %in% custom_dictionary_grouped_trigram$word) %>%
  filter(!word3 %in% custom_dictionary_grouped_trigram$word)

# Counting the trigrams
trigrams_grouped_count <- trigrams_company_separated %>%
  count(word1, word2, word3, sort = TRUE)

# Visualising the top 8 trigrams based on their frequencies and the overall r
atings
trigram_company_top8 %>%
  ggplot(aes(Average_Rating, Frequency)) +
  geom_point(size = 2) +
  geom_text_repel(aes(label = word), max.overlaps = 15) +
  labs(title = "Top 10 Trigram Frequency (Company Level)",
         x = "Average Rating",
         y = "Frequency") +
  theme(plot.title = element_text(hjust = 0.5))
```




```
# Visualising the top 8 trigrams for each rating
trigram_grouped %>%
  group_by(word) %>%
  summarise(n = n(),
            avg_rating = round(mean(overall_rating))) %>%
  arrange(desc(n)) %>%
  group_by(avg_rating) %>%
  slice_max(n, n = 5) %>%
  ungroup() %>%
  ggplot(aes(n, fct_reorder(word, n), fill = avg_rating)) +
  geom_col(show.legend = FALSE) +
  labs(title = "Top 10 Trigram Frequencies For Rating Categories (Review Leve
l)",
         x = "Frequency",
         y = "Terms") +
  facet_wrap(~ avg_rating, nrow = 2, scales = "free") +
  theme(plot.title = element_text(hjust = 0.5))
```




```r
# Filtering bigrams that repeat at least 25 times to be used for graphing ass
ociations
trigram_company_assossiation <- trigrams_grouped_count %>%
  filter(n > 25) %>%
  graph_from_data_frame()

set.seed(1)

arrow_dimensions_grouped_tri <- grid::arrow(type = "closed", length = unit(.2
, "inches"))

ggraph(trigram_company_assossiation, layout = "fr") +
       geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
                      arrow = arrow_dimensions_grouped_tri, end_cap = cir
cle(.07, "inches")) +
       geom_node_point(color = "lightgreen", size = 3) +
       geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
       theme_void()
```



Word Clouds for Company Level Based on Overall Ratings

```
set.seed(1)

# Storing the unigrams used for overall rating of 5 and counting them
overall_5 <- unigram_grouped %>%
  filter(overall_rating == 5) %>%
  group_by(word) %>%
  summarise(total = n()) %>%
  arrange(desc(total))

# Visualising the word cloud for overall rating of 5
wordcloud(overall_5$word, overall_5$total, max.words = 32,
          colors = c("forestgreen", "violetred", "darkgoldenrod3"))
```


```r
# Storing the unigrams used for overall rating of 4 and counting them
overall_4 <- unigram_grouped %>%
  filter(overall_rating == 4) %>%
  group_by(word) %>%
  summarise(total = n()) %>%
  arrange(desc(total))

# Visualising the word cloud for overall rating of 4
wordcloud(overall_4$word, overall_4$total, max.words = 32,
          colors = c("forestgreen", "violetred", "darkgoldenrod3"))
```


```r
# Storing the unigrams used for overall rating of 3 and counting them
overall_3 <- unigram_grouped %>%
  filter(overall_rating == 3) %>%
  group_by(word) %>%
  summarise(total = n()) %>%
```

```
  arrange(desc(total))

# Visualising the word cloud for overall rating of 3
wordcloud(overall_3$word, overall_3$total, max.words = 32,
          colors = c("forestgreen", "violetred", "darkgoldenrod3"))
```


```
# Storing the unigrams used for overall rating of 2 and counting them
overall_2 <- unigram_grouped %>%
  filter(overall_rating == 2) %>%
  group_by(word) %>%
  summarise(total = n()) %>%
  arrange(desc(total))

# Visualising the word cloud for overall rating of 2
wordcloud(overall_2$word, overall_2$total, max.words = 32,
          colors = c("forestgreen", "violetred", "darkgoldenrod3"))
```


```
# Storing the unigrams used for overall rating of 1 and counting them
overall_1 <- unigram_grouped %>%
  filter(overall_rating == 1) %>%
  group_by(word) %>%
  summarise(total = n()) %>%
  arrange(desc(total))

# Visualising the word cloud for overall rating of 1
wordcloud(overall_1$word, overall_1$total, max.words = 32,
          colors = c("forestgreen", "violetred", "darkgoldenrod3"))
```

Word Associations

Word Associations for Review Level

```
# DTM is built for unigrams, bigrams, and trigrams in order to be used for wo
rd correlations
dtm_unigram <- unigram %>%
  count(review_id, word) %>%
  cast_dtm(review_id, word, n)

inspect(dtm_unigram)

## <<DocumentTermMatrix (documents: 25741, terms: 37469)>>
## Non-/sparse entries: 634093/963855436
## Sparsity           : 100%
## Maximal term length: 39
## Weighting          : term frequency (tf)
## Sample             :
##        Terms
## Docs    booked booking experience flight hotel money recommend refund time
tour
##   10368      0       0          2      0     2     0         0      0    1
3
##   12579      1       2          0      0     0     0         0      0    1
1
##   14212      0       0          1      0     3     0         0      0    1
6
##   15367      0       0          0      0     0     0         0      0    0
3
##   22308      0       0          0      0     3     1         0      0    3
1
##   26555      0       0          0      0     0     0         0      0    0
0
##   29545      0       0          0      0     0     2         0      7    0
0
##   34238      0       0          0      7     1     0         0      0    1
0
##   35614      0       0          0      5     1     0         0      1    1
0
##   5051       0       0          0      0     0     0         0      0    2
0

dtm_bigram <- bigram %>%
  count(review_id, word) %>%
  cast_dtm(review_id, word, n)

inspect(dtm_bigram)

## <<DocumentTermMatrix (documents: 25163, terms: 124394)>>
## Non-/sparse entries: 210325/3129915897
## Sparsity           : 100%
```

```
## Maximal term length: 45
## Weighting          : term frequency (tf)
## Sample             :
##        Terms
## Docs    amazing experience business class credit card excellent service
##   1107                   0              0           0                 0
##   1133                   0              0           0                 0
##   11423                  0              0           0                 0
##   12210                  0              0           0                 0
##   20620                  0              0           0                 0
##   22308                  0              0           0                 0
##   26555                  0              0           0                 0
##   33421                  0              0           0                 0
##   3559                   0              0           0                 0
##   7195                   0              0           0                 0
##        Terms
## Docs    extremely helpful highly recommend top notch tour guide travel age
nt
##   1107                  0                0         0           0
0
##   1133                  0                0         1           0
0
##   11423                 0                1         1           0
0
##   12210                 0                0         0           0
0
##   20620                 0                0         0           0
0
##   22308                 0                0         0           1
0
##   26555                 0                0         0           0
0
##   33421                 0                0         0           0
0
##   3559                  0                0         0           0
0
##   7195                  0                0         0           0
0
##        Terms
## Docs    wonderful experience
##   1107                     0
##   1133                     0
##   11423                    0
##   12210                    0
##   20620                    0
##   22308                    0
##   26555                    0
##   33421                    0
##   3559                     0
##   7195                     0
```

```
dtm_trigram <- trigram %>%
  count(review_id, word) %>%
  cast_dtm(review_id, word, n)

inspect(dtm_trigram)
```

```
## <<DocumentTermMatrix (documents: 17803, terms: 52051)>>
## Non-/sparse entries: 57962/926605991
## Sparsity           : 100%
## Maximal term length: 56
## Weighting          : term frequency (tf)
## Sample             :
##        Terms
## Docs    amazing global travel business class flight cheap airport parking
##   1133                      0                     0                     0
##   11423                     0                     0                     0
##   14858                     0                     0                     0
##   20620                     0                     0                     0
##   22308                     0                     0                     0
##   25235                     0                     0                     0
##   28593                     0                     0                     0
##   28904                     0                     0                     0
##   35022                     0                     0                     0
##   3559                      0                     0                     0
##        Terms
## Docs    class action lawsuit cultural island travel hard earned money
##   1133                     0                      0                 0
##   11423                    0                      0                 0
##   14858                    0                      0                 0
##   20620                    0                      0                 0
##   22308                    0                      0                 0
##   25235                    0                      0                 0
##   28593                    0                      0                 0
##   28904                    0                      0                 0
##   35022                    0                      0                 0
##   3559                     0                      0                 0
##        Terms
## Docs    life changing experience lost world adventures round trip flight
##   1133                         0                     0                 0
##   11423                        0                     0                 0
##   14858                        0                     0                 0
##   20620                        0                     0                 0
##   22308                        0                     0                 0
##   25235                        0                     0                 0
##   28593                        0                     0                 0
##   28904                        0                     0                 0
##   35022                        0                     0                 0
##   3559                         0                     0                 0
##        Terms
## Docs    worst travel agency
```

```
##   1133                    0
##   11423                   0
##   14858                   0
##   20620                   0
##   22308                   0
##   25235                   0
##   28593                   0
##   28904                   0
##   35022                   0
##   3559                    0
```

```
# Unigram, bigram, and trigram are all giving 100% sparsity, hence unigram is
used from now on

dtm_unigram_sparse <- removeSparseTerms(dtm_unigram, 0.97)
inspect(dtm_unigram_sparse)
```

```
## <<DocumentTermMatrix (documents: 25741, terms: 111)>>
## Non-/sparse entries: 162899/2694352
## Sparsity           : 94%
## Maximal term length: 13
## Weighting          : term frequency (tf)
## Sample             :
##        Terms
## Docs    booked booking experience flight hotel money recommend refund time
tour
##   20271      0       1          0      8     0     1         0      0    0
0
##   20749      0       0          0      0     0     2         0      7    0
0
##   25211      1      10          0      0     4     1         0      1    1
0
##   28569      1      10          0      0     4     1         0      1    1
0
##   29792      3       0          0      5     0     3         0     12    2
0
##   30191      0       0          0      0     0     8         0      3    1
0
##   32251      1       2          1      0     4     1         0      9    0
0
##   32395      2       1          0     17     0     0         0      0    1
0
##   33421      0       0          0      0     0     0         0      0    0
0
##   9980       0       0          2      0     0     0         0      8    0
0
```

```
# Finding the words that are associated with "recommend", "refund", and "tour
" according to the DTM of unigrams
findAssocs(dtm_unigram_sparse, "recommend", corlimit = 0.1)
```

```
## $recommend
## friends    tour
##    0.12    0.10
```

```r
findAssocs(dtm_unigram_sparse, "refund", corlimit = 0.3)
```

```
## $refund
## canceled
##     0.33
```

```r
findAssocs(dtm_unigram_sparse, "tour", corlimit = 0.4)
```

```
## $tour
## guide
##  0.43
```

```r
# Checking the word combinations that have "excellent" or "amazing" in order
to get an idea about things that are liked in the reviews by the users
bigram %>% group_by(word) %>%
  count(sort = TRUE) %>%
  filter(stringr::str_detect(word,"excellent | amazing"))
```

```
## # A tibble: 693 × 2
## # Groups:   word [693]
##    word                        n
##    <chr>                   <int>
##  1 excellent service         599
##  2 excellent experience      161
##  3 excellent job             111
##  4 absolutely amazing        101
##  5 excellent tour             88
##  6 excellent trip             57
##  7 excellent travel           36
##  8 excellent guide            34
##  9 service amazing            28
## 10 excellent communication    27
## # … with 683 more rows
```

```r
# Checking the word combinations that have "refund" in order to get an idea a
bout the things that are mentioned with having refunds in the reviews by the
users
bigram %>% group_by(word) %>%
  count(sort = TRUE) %>%
  filter(stringr::str_detect(word,"refund"))
```

```
## # A tibble: 818 × 2
## # Groups:   word [818]
##    word                  n
##    <chr>             <int>
##  1 refund policy        62
##  2 partial refund       59
##  3 refund request       46
```

```
##   4 percent refund       39
##   5 refund process       27
##   6 refund amount        20
##   7 refund months        18
##   8 refund money         17
##   9 refundable ticket    15
##  10 cash refund          14
## # … with 808 more rows
```

```
# Checking the word combinations that have "time" in order to get an idea abo
ut things that are liked in the reviews by the users
bigram %>% group_by(word) %>%
  count(sort = TRUE) %>%
  filter(stringr::str_detect(word,"time"))
```

```
## # A tibble: 1,810 × 2
## # Groups:   word [1,810]
##    word                      n
##    <chr>                 <int>
##  1 wonderful time          166
##  2 timely manner           107
##  3 amazing time             95
##  4 free time                66
##  5 time share               62
##  6 life time                58
##  7 numerous times           53
##  8 time frame               53
##  9 lifetime experience      50
## 10 time booking             50
## # … with 1,800 more rows
```

Word Associations for Company Level

```
# DTM is built for unigrams, bigrams, and trigrams in order to be used for wo
rd correlations
dtm_unigram_grouped <- unigram_grouped %>%
  count(agency_id, word) %>%
  cast_dtm(agency_id, word, n)

inspect(dtm_unigram_grouped)
```

```
## <<DocumentTermMatrix (documents: 548, terms: 37404)>>
## Non-/sparse entries: 309548/20187844
## Sparsity           : 98%
## Maximal term length: 39
## Weighting          : term frequency (tf)
## Sample             :
##      Terms
## Docs  booking experience flight hotel money recommend refund time tour tri
p
##   112      81         17      1    11    78        11    114   64    0   3
```

```
9
##   113      18          42      6    10    52          4       8  108    4    1
6
##   618      20          38      2     2    26         14      16   40    0    1
8
##   625      28          24      6    64    40         10      34   51   11    8
3
##   716     132          19    168    58    56          5      94   47    1    5
6
##   766      17          11      0   168    42          5      27   39    0
8
##   773      28          36    189    15    29          5      48   57    0    1
7
##   788       4          27    185    18    16          6      36   69    0    3
2
##   790      13          41    225    15    29          3      24   66    0    1
6
##   792     153          34      3   417    82         11     113   75    0    2
0

dtm_bigram_grouped <- bigram_grouped %>%
  count(agency_id, word) %>%
  cast_dtm(agency_id, word, n)

inspect(dtm_bigram_grouped)

## <<DocumentTermMatrix (documents: 548, terms: 129922)>>
## Non-/sparse entries: 185062/71012194
## Sparsity           : 100%
## Maximal term length: 47
## Weighting          : term frequency (tf)
## Sample             :
##      Terms
## Docs  amazing experience business class credit card excellent service
##   112                  0              0          17                 0
##   113                  0              0           8                 0
##   136                  0              0           0                 0
##   618                  0              0           0                 0
##   625                  0              0           0                 0
##   716                  0              1          15                 0
##   773                  0             19           2                 1
##   781                  0              0           5                 0
##   788                  0              2           4                 0
##   792                  0              0          12                 2
##      Terms
## Docs  extremely helpful highly recommend top notch tour guide travel agent
##   112                 0                0         0          0            0
##   113                 0                2         0          0            0
##   136                 0               19         3          2            0
##   618                 0                6         2          0            0
```

```
##   625                 0                 4         0          2            0
##   716                 0                 0         0          0            2
##   773                 0                 0         0          0            1
##   781                 1                 1         0          0            2
##   788                 0                 2         0          0            0
##   792                 0                 0         0          0            0
##      Terms
## Docs  wonderful experience
##   112                    0
##   113                    0
##   136                    2
##   618                    0
##   625                    0
##   716                    0
##   773                    0
##   781                    0
##   788                    0
##   792                    0
```

```
dtm_trigram_grouped <- trigram_grouped %>%
  count(agency_id, word) %>%
  cast_dtm(agency_id, word, n)

inspect(dtm_trigram_grouped)
```

```
## <<DocumentTermMatrix (documents: 544, terms: 61152)>>
## Non-/sparse entries: 63956/33202732
## Sparsity           : 100%
## Maximal term length: 56
## Weighting          : term frequency (tf)
## Sample             :
##      Terms
## Docs  avoid avoid avoid business class tickets cheap airport parking
##   112                 1                      0                     0
##   113                 0                      0                     0
##   136                 0                      0                     0
##   339                 0                      0                     0
##   599                 0                      0                     0
##   618                 0                      0                     0
##   625                 0                      0                     0
##   781                 0                      0                     0
##   789                 0                      0                     0
##   792                 5                      0                     0
##      Terms
## Docs  credit card company cultural island travel excellent service excelle
nt
##   112                   4                       0
0
##   113                   0                       0
0
```

```
##   136                    0                      0
0
##   339                    0                      0
0
##   599                    0                      0
0
##   618                    0                      0
0
##   625                    0                      0
0
##   781                    3                      0
0
##   789                    0                      0
0
##   792                    3                      0
0
##      Terms
## Docs  hard earned money lost world adventures scam scam scam
##   112                  0                      0              0
##   113                  0                      0              0
##   136                  0                      0              0
##   339                  0                      0              0
##   599                  0                      0              0
##   618                  0                      0              0
##   625                  0                      0              0
##   781                  0                      0              0
##   789                  7                      0              0
##   792                  3                      0              0
##      Terms
## Docs  service excellent service
##   112                          0
##   113                          0
##   136                          0
##   339                          0
##   599                          0
##   618                          0
##   625                          0
##   781                          0
##   789                          0
##   792                          0
```

```
# Bigram and trigram are giving 100% sparsity, hence bigrams are to be used i
nstead of trigrams and the sparsity of DTM of bigram is reduced

dtm_bigram_grouped_sparse <- removeSparseTerms(dtm_bigram_grouped, 0.97)
inspect(dtm_bigram_grouped_sparse)
```

```
## <<DocumentTermMatrix (documents: 548, terms: 524)>>
## Non-/sparse entries: 17262/269890
## Sparsity           : 94%
```

```
## Maximal term length: 26
## Weighting          : term frequency (tf)
## Sample             :
##      Terms
## Docs  amazing experience business class credit card excellent service
##   112                  0              0          17                 0
##   119                  6              0           2                 2
##   596                  0              0          14                 6
##   637                  0              0           9                 0
##   699                  0              1          18                 0
##   703                  0              0           9                 0
##   716                  0              1          15                 0
##   748                  0              0          28                 0
##   761                  0              0           3                 0
##   792                  0              0          12                 2
##      Terms
## Docs  extremely helpful highly recommend top notch tour guide travel agent
##   112                 0                0         0          0            0
##   119                 0                4         4         17            2
##   596                 0                2         0          0            0
##   637                 0                0         0          0            2
##   699                 0                1         0          0            5
##   703                 0                0         0          0            2
##   716                 0                0         0          0            2
##   748                 1                0         0          0            0
##   761                 0                1         0          0            0
##   792                 0                0         0          0            0
##      Terms
## Docs  wonderful experience
##   112                    0
##   119                    2
##   596                    0
##   637                    0
##   699                    0
##   703                    0
##   716                    0
##   748                    0
##   761                    0
##   792                    0
```

```
dtm_trigram_grouped_sparse <- removeSparseTerms(dtm_trigram_grouped, 0.97)
inspect(dtm_trigram_grouped_sparse)
```

```
## <<DocumentTermMatrix (documents: 544, terms: 9)>>
## Non-/sparse entries: 251/4645
## Sparsity           : 95%
## Maximal term length: 27
## Weighting          : term frequency (tf)
## Sample             :
##      Terms
```

```
## Docs  avoid avoid avoid business class tickets credit card company
##   120                 0                      0                  7
##   297                 0                      0                  0
##   663                 0                      0                 13
##   699                 0                      0                  6
##   703                 1                      0                  4
##   704                 1                      0                 11
##   748                 1                      0                  9
##   785                 0                      0                  9
##   789                 0                      0                  0
##   792                 5                      0                  3
##      Terms
## Docs  excellent service excellent excellent tour guide hard earned money
##   120                           0                    0                 1
##   297                           5                    0                 0
##   663                           0                    0                 3
##   699                           0                    0                 3
##   703                           0                    0                 3
##   704                           0                    0                 1
##   748                           0                    0                 0
##   785                           0                    0                 0
##   789                           0                    0                 7
##   792                           0                    0                 3
##      Terms
## Docs  provided excellent service round trip flight service excellent servi
ce
##   120                          0                 0
0
##   297                          0                 0
5
##   663                          0                 0
0
##   699                          0                 0
0
##   703                          0                 0
0
##   704                          0                 0
0
##   748                          0                 0
0
##   785                          0                 0
0
##   789                          0                 0
0
##   792                          0                 0
0

# Finding the words that are associated with "booking", "experience", "recomm
end" and "time" according to the DTM of unigrams
findAssocs(dtm_unigram_grouped, "booking", corlimit = 0.6)
```

```
## $booking
##          book confirmation    confirmed      website      contact cancellat
ion
##          0.81         0.77         0.75         0.73         0.72          0
.72
##        double        email     directly     bookings         card         av
oid
##          0.70         0.68         0.68         0.68         0.68          0
.66
##         hotel    contacted       cancel      confirm        error          b
ank
##          0.66         0.66         0.65         0.65         0.64          0
.64
##   reservation      charged       actual       charge        money         is
sue
##          0.62         0.62         0.62         0.62         0.61          0
.61
##       request      payment      receive      checked
##          0.61         0.61         0.60         0.60
```

```
findAssocs(dtm_unigram_grouped, "experience", corlimit = 0.6)
```

```
## $experience
##   recommend    amazing       time  wonderful     highly incredible      sta
rt
##        0.72       0.69       0.69       0.66       0.65       0.64       0.
63
##        trip        met absolutely     entire     forget    culture     count
ry
##        0.63       0.62       0.61       0.61       0.61       0.61       0.
60
##   extremely  fantastic
##        0.60       0.60
```

```
findAssocs(dtm_unigram_grouped, "recommend", corlimit = 0.6)
```

```
## $recommend
##          highly     experience        amazing      excellent      wonderful
##            0.89           0.72           0.71           0.71           0.70
##       fantastic           trip      questions knowledgeable      organized
##            0.68           0.65           0.64           0.63           0.63
##         perfect        friends      itinerary        planned       provided
##            0.62           0.61           0.61           0.61           0.61
##        arranged       fabulous            met
##            0.61           0.60           0.60
```

```
findAssocs(dtm_unigram_grouped, "time", corlimit = 0.55)
```

```
## $time
##           day        times   experience         left        spent    absolut
ely
```

```
##         0.74         0.72         0.69         0.69         0.67          0
.66
##      finally        can’t    extremely       person         wife          h
ope
##         0.65         0.65         0.64         0.63         0.63          0
.63
##      husband        leave         week      started        spend    complet
ely
##         0.62         0.62         0.62         0.61         0.61          0
.61
##      offered         care      decided       didn’t         wait          i
t’s
##         0.61         0.60         0.60         0.60         0.60          0
.60
##         feel        hours        start        weeks         hour         st
ars
##         0.59         0.59         0.59         0.59         0.59          0
.59
##         call          due     informed       taking        don’t           
bit
##         0.58         0.58         0.58         0.58         0.58          0
.57
## disappointed       entire       expect         half    recommend         wo
rth
##         0.57         0.57         0.57         0.57         0.57          0
.57
##         mind          ago       reason         paid       couple        iss
ues
##         0.57         0.57         0.56         0.56         0.55          0
.55
##     happened     horrible
##         0.55         0.55

# Finding the words that are associated with "top notch", "credit card","high
ly recommend", and "tour guide" according to the DTM of bigrams
findAssocs(dtm_bigram_grouped_sparse, "top notch", corlimit = 0.35)

## $`top notch`
##        highly recommend              amazing trip          fantastic trip
##                    0.50                      0.49                    0.48
##          wonderful trip               entire trip          fabulous trip
##                    0.47                      0.45                    0.45
##          travel advisor            memorable trip             bucket list
##                    0.44                      0.44                    0.41
##       travel experience              tour company extremely knowledgeable
##                    0.40                      0.40                    0.38
##                pick ups             trip exceeded       absolutely amazing
##                    0.38                      0.38                    0.37
##      amazing experience            excellent trip     wonderful experience
##                    0.37                      0.37                    0.37
```

```
##       planning process       beautiful country             family trip 
##                   0.37                    0.37                    0.36 
##           job planning 
##                   0.36
```

```r
findAssocs(dtm_bigram_grouped_sparse, "credit card", corlimit = 0.5)
```

```
## $`credit card`
##    card company  hotel directly booked directly      debit card 
##            0.86            0.61            0.53            0.51
```

```r
findAssocs(dtm_bigram_grouped_sparse, "highly recommend", corlimit = 0.4)
```

```
## $`highly recommend`
##             amazing trip            wonderful trip      absolutely amazing 
##                     0.56                      0.54                    0.50 
##                top notch extremely knowledgeable    wonderful experience 
##                     0.50                      0.49                    0.48 
##           fabulous trip               entire trip          wonderful time 
##                     0.47                      0.46                    0.46 
##          fantastic trip                  day trip               week trip 
##                     0.45                      0.43                    0.42 
##       beautiful country                  day tour            private tour 
##                     0.42                      0.41                    0.40 
##            tour company                  pick ups           lifetime trip 
##                     0.40                      0.40                    0.40
```

```r
findAssocs(dtm_bigram_grouped_sparse, "tour guide", corlimit = 0.5)
```

```
## $`tour guide`
##          excellent tour            amazing tour knowledgeable friendly 
##                    0.72                    0.60                   0.51
```

```r
# Checking the word combinations that have "excellent" or "amazing" in order
to get an idea about things that are liked in the reviews by the users
trigram_grouped %>% group_by(word) %>%
  count(sort = TRUE) %>%
  filter(stringr::str_detect(word,"excellent | amazing"))
```

```
## # A tibble: 1,723 × 2
## # Groups:   word [1,723]
##    word                                n
##    <chr>                           <int>
##  1 excellent service excellent        48
##  2 service excellent service          47
##  3 provided excellent service         30
##  4 excellent tour guide               21
##  5 amazing service amazing            15
##  6 excellent experience excellent     14
##  7 received excellent service         13
##  8 spoke excellent english            13
##  9 experience excellent experience    12
```

```
## 10 service amazing service            11
## # … with 1,713 more rows
```

```
# Checking the word combinations that have "refund" in order to get an idea a
bout the things that are mentioned with having refunds in the reviews by the
users
trigram_grouped %>% group_by(word) %>%
  count(sort = TRUE) %>%
  filter(stringr::str_detect(word,"refund"))
```

```
## # A tibble: 537 × 2
## # Groups:   word [537]
##    word                            n
##    <chr>                       <int>
##  1 refundable service fee          5
##  2 credit card refund              4
##  3 refund empty promises           4
##  4 refund absolutely terrible      3
##  5 air canada refunded             2
##  6 amex amex refunded              2
##  7 amsterdam requested refunds     2
##  8 canceled ba refunded            2
##  9 day refunds constant            2
## 10 dishonest refund practices      2
## # … with 527 more rows
```

```
# Checking the word combinations that have "time" in order to get an idea abo
ut things that are liked in the reviews by the users
trigram_grouped %>% group_by(word) %>%
  count(sort = TRUE) %>%
  filter(stringr::str_detect(word,"time"))
```

```
## # A tibble: 1,353 × 2
## # Groups:   word [1,353]
##    word                        n
##    <chr>                   <int>
##  1 life time experience       12
##  2 called numerous times       5
##  3 lifetime adventure life     5
##  4 airport parking times       4
##  5 family time vacation        4
##  6 flight time change          4
##  7 guys prove time             4
##  8 hard time finding           4
##  9 hour wait time              4
## 10 night time tour             4
## # … with 1,343 more rows
```

Word Importance By Metadata

- Some top words are identified based on the metadata available in the dataset.

```r
# Categories
# For adding the category of the travel agency
for_category <- travel_all %>%
  select(agency_id, category) %>%
  unique()

# Adding the category of the travel agency to the data frame
tokenized_reviews_grouped_by_tf_idf <- tokenized_reviews_grouped_by_tf_idf %>
%
  left_join(for_category)

# Finding the top 5 categories based on tf-idf values
top_5_categories <- tokenized_reviews_grouped_by_tf_idf %>%
  arrange(desc(tf_idf)) %>%
  select(agency_id, category) %>%
  unique() %>%
  group_by(category) %>%
  summarise(total = n()) %>%
  arrange(desc(total)) %>%
  top_n(5)

# Filtering the words that are used in the reviews for the travel agencies fr
om the top 5 categories
tokens_top_5_categories <- tokenized_reviews_grouped_by_tf_idf %>%
  select(agency_id, category) %>%
  filter(category %in% top_5_categories$category) %>%
  unique()

# Calculating the total number of occurrences of the tokens that are kept for
each category
category_tokens <- tokenized_reviews_grouped_by_tf_idf %>%
  right_join(tokens_top_5_categories) %>%
  group_by(category, word) %>%
  summarise(total = sum(n)) %>%
  arrange(desc(total))

# Loop for getting the top 10 tokens per category for the travel agency
result1 <- data.frame()
for(categ in 1:nrow(top_5_categories)) {
  print(paste0("For category: ", top_5_categories$category[categ]))
  result1_temp <- category_tokens %>%
    ungroup() %>%
    filter(category == top_5_categories$category[categ]) %>%
    slice_max(total, n = 10) %>%
    select(-total) %>%
    mutate(rank = row_number())
  result1 <- rbind(result1, result1_temp)
}
```

```
## [1] "For category: Travel Agency"
## [1] "For category: Tour Operator"
## [1] "For category: Flights Search Site"
## [1] "For category: Vacation Rental"
## [1] "For category: Hotel"
```

```
(result1)
```

```
## # A tibble: 50 × 3
##    category            word          rank
##    <chr>               <chr>         <int>
##  1 Travel Agency       book              1
##  2 Travel Agency       time              2
##  3 Travel Agency       experience        3
##  4 Travel Agency       flight            4
##  5 Travel Agency       call              5
##  6 Travel Agency       refund            6
##  7 Travel Agency       recommend         7
##  8 Travel Agency       cancel            8
##  9 Travel Agency       price             9
## 10 Travel Agency       money            10
## 11 Tour Operator       experience        1
## 12 Tour Operator       time              2
## 13 Tour Operator       plan              3
## 14 Tour Operator       hotel             4
## 15 Tour Operator       amaze             5
## 16 Tour Operator       wonderful         6
## 17 Tour Operator       excellent         7
## 18 Tour Operator       recommend         8
## 19 Tour Operator       book              9
## 20 Tour Operator       knowledgeable    10
## 21 Flights Search Site flight            1
## 22 Flights Search Site book              2
## 23 Flights Search Site time              3
## 24 Flights Search Site cancel            4
## 25 Flights Search Site call              5
## 26 Flights Search Site ticket            6
## 27 Flights Search Site airline           7
## 28 Flights Search Site refund            8
## 29 Flights Search Site money             9
## 30 Flights Search Site bad              10
## 31 Vacation Rental     stay              1
## 32 Vacation Rental     time              2
## 33 Vacation Rental     clean             3
## 34 Vacation Rental     book              4
## 35 Vacation Rental     experience        5
## 36 Vacation Rental     vacation          6
## 37 Vacation Rental     family            7
## 38 Vacation Rental     home              8
## 39 Vacation Rental     nice              9
```

```
## 40 Vacation Rental       check          10
## 41 Hotel                 book            1
## 42 Hotel                 stay            2
## 43 Hotel                 call            3
## 44 Hotel                 time            4
## 45 Hotel                 cancel          5
## 46 Hotel                 reservation     6
## 47 Hotel                 pay             7
## 48 Hotel                 refund          8
## 49 Hotel                 site            9
## 50 Hotel                 money          10
```

```
# Countries
# For adding the country of the reviews for travel agencies
for_country <- travel_all %>%
  select(agency_id, country) %>%
  unique()

# Adding the country of the reviews to the data frame
tokenized_reviews_grouped_by_tf_idf <- tokenized_reviews_grouped_by_tf_idf
%>%
  left_join(for_country)

# Finding the top 10 countries based on tf-idf values
top_10_countries <- tokenized_reviews_grouped_by_tf_idf %>%
  arrange(desc(tf_idf)) %>%
  select(agency_id, country) %>%
  unique(.) %>%
  group_by(country) %>%
  summarise(total = n()) %>%
  arrange(desc(total)) %>%
  top_n(10)

# Filtering the words that are used in the reviews for the travel agencies
from the top 10 countries
tokens_top_10_countries <- tokenized_reviews_grouped_by_tf_idf %>%
  select(agency_id, country) %>%
  filter(country %in% top_10_countries$country) %>%
  unique(.)

# Calculating the total number of occurrences of the tokens that are kept for
each country
country_tokens <- tokenized_reviews_grouped_by_tf_idf %>%
  right_join(tokens_top_10_countries) %>%
  group_by(country, word) %>%
  summarise(total = sum(n)) %>%
  arrange(desc(total))
```

```r
# Loop for getting the top 10 tokens per country for the reviews per travel agency
result2 <- data.frame()
for(cntry in 1:nrow(top_10_countries)){
  print(paste0("For country: ", top_10_countries$country[cntry]))
  result2_temp <- country_tokens %>%
    ungroup() %>%
    filter(country == top_10_countries$country[cntry]) %>%
    slice_max(total, n = 10) %>%
    select(-total) %>%
    mutate(rank = row_number())
  result2 <- rbind(result2, result2_temp)
}
```

```
## [1] "For country: US"
## [1] "For country: CA"
## [1] "For country: GB"
## [1] "For country: AU"
## [1] "For country: IN"
## [1] "For country: DE"
## [1] "For country: FR"
## [1] "For country: ES"
## [1] "For country: NL"
## [1] "For country: IT"
```

```r
(result2)
```

```
## # A tibble: 100 × 3
##    country   word            rank
##    <chr>     <chr>          <int>
##  1 US        book               1
##  2 US        time               2
##  3 US        experience         3
##  4 US        flight             4
##  5 US        call               5
##  6 US        refund             6
##  7 US        cancel             7
##  8 US        stay               8
##  9 US        recommend          9
## 10 US        money             10
## 11 CA        book               1
## 12 CA        time               2
## 13 CA        experience         3
## 14 CA        flight             4
## 15 CA        call               5
## 16 CA        refund             6
## 17 CA        cancel             7
## 18 CA        stay               8
## 19 CA        money              9
```

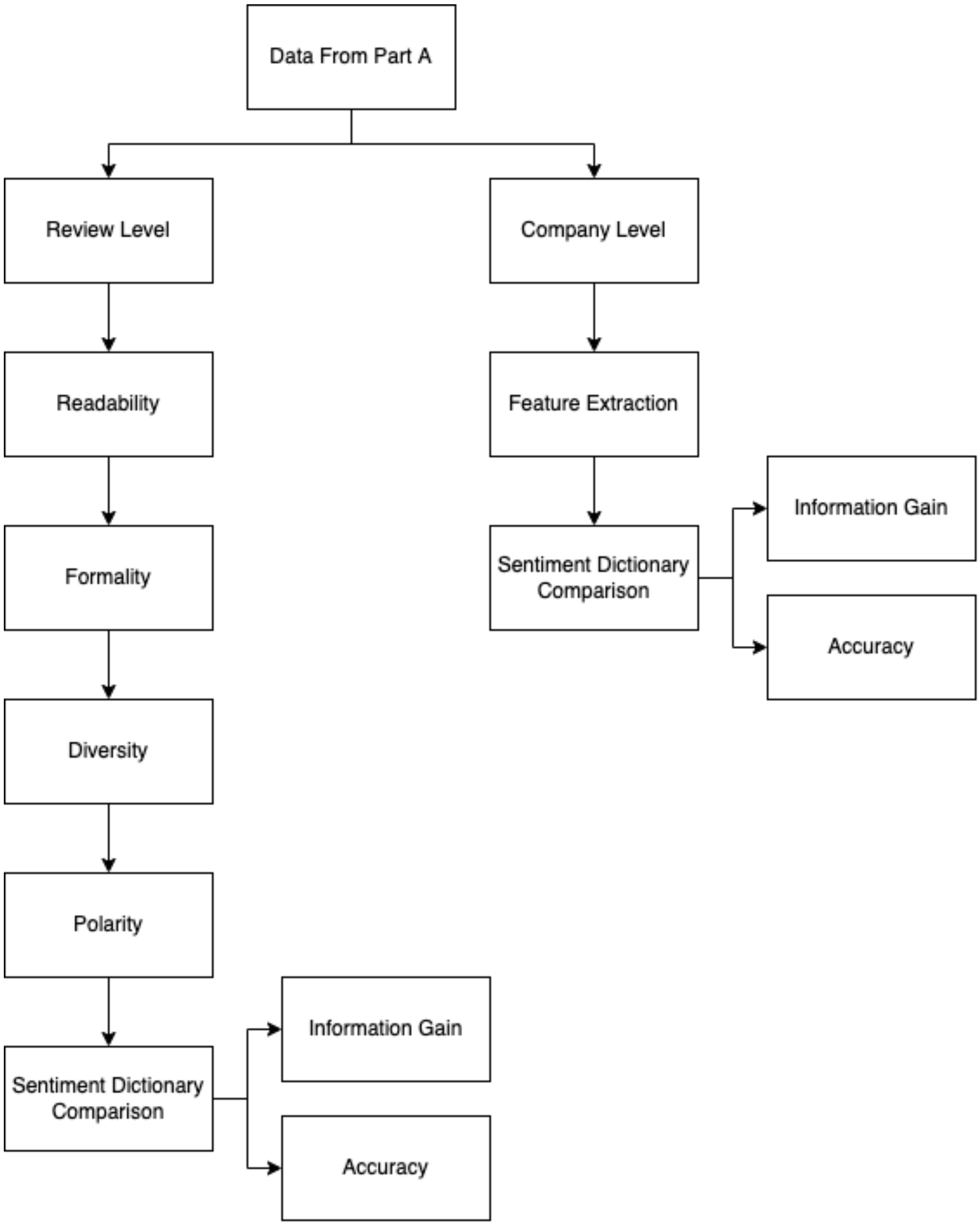
```
## 20 CA         pay                 10
## 21 GB         book                 1
## 22 GB         time                 2
## 23 GB         flight               3
## 24 GB         experience           4
## 25 GB         call                 5
## 26 GB         refund               6
## 27 GB         cancel               7
## 28 GB         moeny                8
## 29 GB         pay                  9
## 30 GB         stay                10
## 31 AU         book                 1
## 32 AU         time                 2
## 33 AU         flight               3
## 34 AU         experience           4
## 35 AU         call                 5
## 36 AU         refund               6
## 37 AU         cancel               7
## 38 AU         money                8
## 39 AU         ticket               9
## 40 AU         pay                 10
## 41 IN         book                 1
## 42 IN         flight               2
## 43 IN         call                 3
## 44 IN         time                 4
## 45 IN         refund               5
## 46 IN         cancel               6
## 47 IN         experience           7
## 48 IN         ticket               8
## 49 IN         money                9
## 50 IN         email               10
## 51 DE         book                 1
## 52 DE         flight               2
## 53 DE         time                 3
## 54 DE         refund               4
## 55 DE         cancel               5
## 56 DE         call                 6
## 57 DE         experience           7
## 58 DE         ticket               8
## 59 DE         money                9
## 60 DE         pay                 10
## 61 FR         book                 1
## 62 FR         flight               2
## 63 FR         time                 3
## 64 FR         refund               4
## 65 FR         experience           5
## 66 FR         cancel               6
## 67 FR         call                 7
## 68 FR         ticket               8
## 69 FR         money                9
```

```
## 70 FR         pay                10
## 71 ES         book                1
## 72 ES         flight              2
## 73 ES         time                3
## 74 ES         cancel              4
## 75 ES         call                5
## 76 ES         refund              6
## 77 ES         money               7
## 78 ES         pay                 8
## 79 ES         experience          9
## 80 ES         ticket             10
## 81 NL         book                1
## 82 NL         flight              2
## 83 NL         time                3
## 84 NL         cancel              4
## 85 NL         refund              5
## 86 NL         call                6
## 87 NL         experience          7
## 88 NL         money               8
## 89 NL         pay                 9
## 90 NL         bad                10
## 91 IT         book                1
## 92 IT         flight              2
## 93 IT         time                3
## 94 IT         experience          4
## 95 IT         cancel              5
## 96 IT         refund              6
## 97 IT         ticket              7
## 98 IT         money               8
## 99 IT         call                9
##100 IT         pay                10
```

Part B

Section Plan

In this section, various sentiment dictionaries are compared and sentiment analysis is performed on both review and company level. For further understanding, the pipeline below is attached.



Data Preparation

```
# Loading the sentiment dictionaries
travel_all <- readRDS("travel_all.rds")
tokenized_reviews_grouped_by_company <- readRDS("tokenized_reviews_grouped_by
_company.rds")
text_grouped_by_company <- readRDS("text_grouped_by_company.rds")
bing_dictionary <- tidytext::get_sentiments("bing")
afinn_dictionary <- tidytext::get_sentiments("afinn")
lm_dictionary <- tidytext::get_sentiments("loughran")
nrc_dictionary <- tidytext::get_sentiments("nrc")
senticnet_dictionary <- lexicon::hash_sentiment_senticnet
jockers_rinker_dictionary <- lexicon::hash_sentiment_jockers_rinker
```

- What variables can be extracted from the text that can be related with the rating score?

```
# Finding the possible variables that can be extracted

# To be used for adding the star ratings of the reviews
for_star <- travel_all %>%
  dplyr::select(agency_id, star)

# Since in data preparation the overall ratings were rounded for ease of visu
alisation, they are re-calculated through the average of the ratings for each
travel agency
ratings <- text_grouped_by_company %>%
  group_by(agency_id) %>%
  left_join(for_star) %>%
  summarise(overall_rating_unrounded = mean(star)) %>%
  ungroup()

# Finding the quantiles of the distribution of overall ratings
quantile(ratings$overall_rating_unrounded)

# Assigning the 50% to be used as a boundary for splitting the ratings into t
wo groups
ratings$rating_category <- ifelse(ratings$overall_rating_unrounded < 4.389, 1
, 2)

# Adding the rating grouping, removing the stop words, and finding the freque
ncies of the words for each rating grouping
ratings_tokens <- tokenized_reviews_grouped_by_company %>%
  left_join(ratings) %>%
  anti_join(stop_words) %>%
  group_by(rating_category, word) %>%
  summarise(total = sum(n))

# Listing the top 10 most frequent words from rating group 1
ratings_tokens %>% filter(rating_category == 1) %>% arrange(desc(total)) %>%
top_n(10)
```

```r
# Listing the top 10 most frequent words from rating group 2
ratings_tokens %>% filter(rating_category == 2) %>% arrange(desc(total)) %>%
top_n(10)
```

Feature extraction

```r
# To be used for adding the names of the travel agencies using agency ID
for_agency_name <- travel_all %>%
  dplyr::select(agency_id, agency_name, overall_rating)

# Creating a data set of reviews without removing syntactical features
syntactical_reviews_grouped <- text_grouped_by_company %>%
  left_join(for_agency_name) %>%
  mutate(review_length = nchar(text_grouped))

# Feature extraction for the reviews that contain "flight" and building the m
odel
for_flight_feature_grouped <- syntactical_reviews_grouped %>%
  group_by(agency_id) %>%
  mutate(text_grouped = tolower(text_grouped)) %>%
  summarise(flight_check = str_detect(text_grouped, "flight")) %>%
  mutate(flight = ifelse(flight_check == TRUE, "1", "0")) %>%
  dplyr::select(agency_id, flight)

for_flight_feature_companies <- for_flight_feature_grouped %>%
 left_join(syntactical_reviews_grouped) %>%
 dplyr::select(agency_id, overall_rating, flight)

flight_model_company <- polr(factor(overall_rating) ~ flight, data = for_flig
ht_feature_companies, Hess = TRUE, method = c("logistic"))

# Feature extraction for the reviews that contain "refund" and building the m
odel
for_refund_feature <- syntactical_reviews_grouped %>%
  group_by(agency_id) %>%
  mutate(text_grouped = tolower(text_grouped)) %>%
  summarise(refund_check = str_detect(text_grouped, "refund")) %>%
  mutate(refund = ifelse(refund_check == TRUE, "1", "0")) %>%
  dplyr::select(agency_id, refund)

for_refund_feature_companies <- for_refund_feature %>%
 inner_join(syntactical_reviews_grouped) %>%
 dplyr::select(agency_id, overall_rating, refund)

refund_model_company <- polr(factor(overall_rating) ~ refund, data = for_refu
nd_feature_companies, Hess = TRUE, method = c("logistic"))

# Feature extraction for the reviews that contain "customer" and building the
model
for_customer_feature <- syntactical_reviews_grouped %>%
```

```
  group_by(agency_id) %>%
  mutate(text_grouped = tolower(text_grouped)) %>%
  summarise(customer_check = str_detect(text_grouped, "customer")) %>%
  mutate(customer = ifelse(customer_check == TRUE, "1", "0")) %>%
  dplyr::select(agency_id, customer)

for_customer_feature_companies <- for_customer_feature %>%
 inner_join(syntactical_reviews_grouped) %>%
 dplyr::select(agency_id, overall_rating, customer)

customer_model_company <- polr(factor(overall_rating) ~ customer, data = for_
customer_feature_companies, Hess = TRUE, method = c("logistic"))

# Feature extraction for the reviews that contain "experience" and building t
he model
for_experience_feature <- syntactical_reviews_grouped %>%
  group_by(agency_id) %>%
  mutate(text_grouped = tolower(text_grouped)) %>%
  summarise(experience_check = str_detect(text_grouped, "experience")) %>%
  mutate(experience = ifelse(experience_check == TRUE, "1", "0")) %>%
  dplyr::select(agency_id, experience)

for_experience_feature_companies <- for_experience_feature %>%
 inner_join(syntactical_reviews_grouped) %>%
 dplyr::select(agency_id, overall_rating, experience)

experience_model_company <- polr(factor(overall_rating) ~ experience, data =
for_experience_feature_companies, Hess = TRUE, method = c("logistic"))

# Feature extraction for the reviews that contain "recommend" and building th
e model
for_recommend_feature <- syntactical_reviews_grouped %>%
  group_by(agency_id) %>%
  mutate(text_grouped = tolower(text_grouped)) %>%
  summarise(recommend_check = str_detect(text_grouped, "recommend")) %>%
  mutate(recommend = ifelse(recommend_check == TRUE, "1", "0")) %>%
  dplyr::select(agency_id, recommend)

for_recommend_feature_companies <- for_recommend_feature %>%
 inner_join(syntactical_reviews_grouped) %>%
 dplyr::select(agency_id, overall_rating, recommend)

recommend_model_company <- polr(factor(overall_rating) ~ recommend, data = fo
r_recommend_feature_companies, Hess = TRUE, method = c("logistic"))

# Feature extraction for the reviews that contain "plan" and building the mod
el
for_plan_feature <- syntactical_reviews_grouped %>%
  group_by(agency_id) %>%
```

```
  mutate(text_grouped = tolower(text_grouped)) %>%
  summarise(plan_check = str_detect(text_grouped, "plan")) %>%
  mutate(plan = ifelse(plan_check == TRUE, "1", "0")) %>%
  dplyr::select(agency_id, plan)

for_plan_feature_companies <- for_plan_feature %>%
 inner_join(syntactical_reviews_grouped) %>%
 dplyr::select(agency_id, overall_rating, plan)

plan_model_company <- polr(factor(overall_rating) ~ plan, data = for_plan_fea
ture_companies, Hess = TRUE, method = c("logistic"))

# Feature extraction for the reviews that contain the name of the agency name
and building the model
for_agency_name_feature <- syntactical_reviews_grouped %>%
  group_by(agency_id) %>%
  mutate(text_grouped = tolower(text_grouped)) %>%
  summarise(agency_name_check = str_detect(text_grouped, agency_name)) %>%
  mutate(mentions_agency_name = ifelse(agency_name_check == TRUE, "1", "0"))
%>%
  dplyr::select(agency_id, mentions_agency_name)

for_agency_name_feature_companies <- for_agency_name_feature %>%
 inner_join(syntactical_reviews_grouped) %>%
 dplyr::select(agency_id, overall_rating, mentions_agency_name)

agency_name_model_company <- polr(factor(overall_rating) ~ mentions_agency_na
me,
                                  data = for_agency_name_feature_companies, H
ess = TRUE, method = c("logistic"))

# Feature extraction for the appearance of exclamation marks in each review a
nd building the models
for_exclamation_mark_feature <- travel_all %>%
  group_by(agency_id) %>%
  mutate(summary_review = tolower(summary_review)) %>%
  summarise(exclamation_check = str_detect(summary_review, "!")) %>%
  mutate(exclamation = ifelse(exclamation_check == TRUE, "1", "0")) %>%
  dplyr::select(agency_id, exclamation)

for_exclamation_mark_feature_companies <- for_exclamation_mark_feature %>%
 inner_join(travel_all) %>%
 dplyr::select(agency_id, overall_rating, exclamation)

exclamation_model_company <- polr(factor(overall_rating) ~ exclamation,
                                  data = for_exclamation_mark_feature_compani
es, Hess = TRUE, method = c("logistic"))

# Feature extraction for the appearance of capital words in each review and b
```

```
uilding the models
for_capital_word_feature <- travel_all %>%
  group_by(agency_id) %>%
  summarise(capital_word_check = str_detect(summary_review, "\\b[A-Z]+\\b"))
%>%
  mutate(capital_word = ifelse(capital_word_check == TRUE, "1", "0")) %>%
  dplyr::select(agency_id, capital_word)

for_capital_word_feature_companies <- for_capital_word_feature %>%
 inner_join(travel_all) %>%
 dplyr::select(agency_id, overall_rating, capital_word)

capital_word_model_company <- polr(factor(overall_rating) ~ capital_word,
                                  data = for_capital_word_feature_companies,
Hess = TRUE, method = c("logistic"))


# Comparing the models
stargazer(flight_model_company, refund_model_company, customer_model_company,
experience_model_company, recommend_model_company, plan_model_company, agency
_name_model_company, exclamation_model_company, capital_word_model_company,
          add.lines = list(c("AIC"), round(AIC(flight_model_company), 1),
                                     round(AIC(refund_model_company), 1),
                                     round(AIC(customer_model_company), ),
                                     round(AIC(experience_model_company), 1),
                                     round(AIC(recommend_model_company), 1),
                                     round(AIC(plan_model_company), 1),
                                     round(AIC(agency_name_model_company), 1)
,
                                     round(AIC(exclamation_model_company), 1)
,
                                     round(AIC(capital_word_model_company), 1
)),
          type = "text", notes.label = "Significance Labels", single.row = TR
UE, align = TRUE, flip = TRUE)
```

- All of the features are significant, however since "experience" model has the highest AIC value, it is by far the most significant.

Readability

- Is readability of the summaries and reviews an important predictor of the review ratings?

```
# Storing the necessary variables from the main data set separately to use it
later on
all_reviews <- travel_all %>% dplyr::select(review_id, summary_review, star)
%>% unique()

# Creating an empty data frame for the readability
readability_df <- data.frame()
```

```r
# Checking the readability of every separate observation of summary and revie
w
for(i in 1:nrow(all_reviews)){

  readability_temp <- data.frame()

  text1 <- iconv(all_reviews$summary_review[i])
  text1 <- removeNumbers(text1)
  text1 <- removePunctuation(text1)

  tryCatch(readability_temp <- flesch_kincaid(text1), error = function(e){
    cat("Error parsing")
  })

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

     readability_temp <- readability_temp$Readability
     readability_temp$review_id <- all_reviews$review_id[i]
     readability_df <- bind_rows(readability_df, readability_temp)
   }

  print(i)
}

# Joining the reviews with their respective readabilities
all_reviews_read <- all_reviews %>%
  left_join(readability_df)

# saveRDS(all_reviews_read, "all_reviews_read.rds")
```

- Using the readability of the summaries and reviews, how does this relate with the star rating that each indiidual review has?

```r
all_reviews_read <- readRDS("all_reviews_read.rds")

# Building regression models
model1 <- lm(log(all_reviews_read$star) ~ log(all_reviews_read$word.count))
model2 <- lm(log(all_reviews_read$star) ~ log(all_reviews_read$syllable.count
))
model3 <- lm(log(all_reviews_read$star) ~ log(all_reviews_read$FK_grd.lvl))
model4 <- lm(log(all_reviews_read$star) ~ log(all_reviews_read$FK_read.ease))

# Comparing the regression models
stargazer::stargazer(model1, model2, model3, model4, type = "text", add.lines
= list(c("AIC", round(AIC(model1), 1), round(AIC(model2), 1), round(AIC(model
3), 1), round(AIC(model4), 1), single.row = TRUE, align = TRUE, flip = TRUE))
)

# It can clearly be seen from the regression models that all of the metrics a
```

```
re significant predictors of the review rating. AIC values show the fitness o
f the model to the data. Among the predictors, word count has the highest eff
ect which can be seen from the Adjusted R2 values.

all_reviews_read %>%
  dplyr::select(review_id, FK_grd.lvl, star) %>%
  distinct() %>%
  na.omit() %>%
  filter(FK_grd.lvl > 15) %>%
  ggplot(aes(FK_grd.lvl, star)) +
  geom_smooth(method = "lm") + geom_point() +
  ggtitle("Rating Scores vs. Readability of Reviews")

cor.test(all_reviews_read$star,
         all_reviews_read$FK_grd.lvl,
         method="pearson")
```

Formality

```
# Calculating the formality of the reviews
formality <- formality(all_reviews$summary_review, all_reviews$review_id)
formality_calculation <- formality$formality %>% dplyr::select(review_id, for
mality)
formality_calculation$review_id <- as.numeric(formality_calculation$review_id
)

# saveRDS(formality_calculation, "formality_calculation.rds")

formality_calculation <- readRDS("formality_calculation.rds")

# Joining the formalities with their respective reviews
all_reviews_formality <- all_reviews %>%
  left_join(formality_calculation)

# Visualising the relationship with review ratings
all_reviews_formality %>%
  dplyr::select(formality, star) %>%
  na.omit() %>%
  ggplot(aes(formality, star)) + geom_smooth(method = "lm") + geom_point()

# Building a regression model for the formality of the reviews and their star
ratings
model5 <- lm(log(all_reviews_formality$star) ~ log(all_reviews_formality$form
ality))
```

Diversity

```
# Calculating the diversity of the reviews
diversity <- diversity(all_reviews$summary_review, all_reviews$review_id)
```

```r
# Visualising the diversities
plot(diversity(all_reviews$summary_review, all_reviews$star)) + coord_flip()
```

Polarity

```r
# Calculating the polarities of the reviews
polarity <- polarity(all_reviews$summary_review, all_reviews$review_id)
polarity_calculation <- polarity$all %>% dplyr::select(review_id, polarity)
polarity_calculation$review_id <- as.integer(polarity_calculation$review_id)
# Joining the polarity values with each respective reviews
all_reviews_polarity <- all_reviews %>%
  left_join(polarity_calculation)

# saveRDS(polarity_calculation, "polarity_calculation.rds")

# Visualising the relationship of polarity values with the review ratings
all_reviews_polarity %>%
  dplyr::select(polarity, star) %>%
  na.omit() %>%
  ggplot(aes(polarity, star)) + geom_smooth(method = "lm") + geom_point() + y
lim(0, 5.3)

# Building the regression model for the polarity of the reviews and their sta
r ratings
model6 <- lm(log(all_reviews_polarity$star) ~ all_reviews_polarity$polarity)

# Visualising the relationship between the poalrity and the review ratings
plot(polarity(all_reviews$summary_review, all_reviews$star))

# Comparing the linear models with stargazer function
stargazer(model1, model2, model3, model4, model5, model6, type = "text", note
s.label = "Significance Labels")
```

- It is observed that while all metrics are significant in predicting the review rating, polarity has the highest impact.

Examining the Dictionaries

The Computation of the Sentiment On Review Level

```r
# Computing the sentiment using the tidytext inner join since we already have
the tokens_all dataframe
# Storing the review lengths in a separate variable to be used later on
review_id_review_length <- travel_all %>%
  dplyr::select(review_id, review_length)

# Using the previously created unigrams for reviews, the word counts are calc
ulated
review_id_word_count <- unigram %>%
  group_by(review_id) %>%
  summarise(word_count = n())
```

```r
# To be used for adding the individual star ratings
for_star <- travel_all %>%
  dplyr::select(., c(12, 6))

# Storing the unigram tokens of reviews separately to use further
tokenized_reviews <- unigram

# To be used for adding the agency ID
for_agency_id <- travel_all %>%
  dplyr::select(., c(1, 12)) %>%
  distinct()

# To be used for adding the overall rating of the travel agencies
for_overall <- travel_all %>%
  dplyr::select(agency_id, overall_rating) %>%
  distinct()
```

Sentiment Dictionaries

```r
# Bing Liu - Sentiment Dictionary
bing_liu_sentiment_reviews <- tokenized_reviews %>%
  inner_join(bing_dictionary) %>%
  count(sentiment, review_id) %>%
  spread(sentiment, n) %>%
  left_join(for_star) %>%
  mutate(bing_liu_sentiment = positive - negative) %>%
  dplyr::select(review_id, bing_liu_sentiment, star) %>%
  left_join(review_id_word_count)

# NRC Dictionary
nrc_emotions_reviews <- tokenized_reviews %>%
  inner_join(nrc_dictionary) %>%
  count(sentiment, review_id) %>%
  spread(sentiment, n) %>%
  left_join(for_star) %>%
  left_join(review_id_word_count) %>%
  mutate(sentiment_nrc = positive - negative)

# Afinn Dictionary
afinn_sentiment_reviews <- tokenized_reviews %>%
  inner_join(afinn_dictionary) %>%
  group_by(review_id) %>%
  summarise(sentiment_afinn = sum(value)) %>%
  left_join(for_star) %>%
  left_join(review_id_word_count)

# Loughran Dictionary
loughran_sentiments_reviews <- tokenized_reviews %>%
  inner_join(lm_dictionary) %>%
```

```
  count(sentiment, review_id) %>%
  spread(sentiment, n) %>%
  left_join(for_star) %>%
  left_join(review_id_word_count) %>%
  mutate(sentiment_loughran = positive - negative) %>%
  dplyr::select(review_id, sentiment_loughran, star, word_count)

# Senticnet Dictionary
senticnet_dictionary <- senticnet_dictionary %>%
  rename(word = x,
         value = y)

senticnet_sentiment_reviews <- tokenized_reviews %>%
  inner_join(senticnet_dictionary) %>%
  group_by(review_id) %>%
  summarise(sentiment_senticnet = sum(value)) %>%
  left_join(for_star) %>%
  left_join(review_id_word_count)

# Jockers Rinker Dictionary
jockers_rinker_dictionary <- jockers_rinker_dictionary %>%
  rename(word = x,
         value = y)

jr_sentiment_reviews <- tokenized_reviews %>%
  inner_join(jockers_rinker_dictionary) %>%
  group_by(review_id) %>%
  summarise(sentiment_jr = sum(value)) %>%
  left_join(for_star) %>%
  left_join(review_id_word_count)

# Joining all the data from the sentiment dictionaries and removing the insta
nces where one dictionary does not contain the word
all_sentiments_together_reviews <- bing_liu_sentiment_reviews %>%
  left_join(nrc_emotions_reviews) %>%
  left_join(afinn_sentiment_reviews) %>%
  left_join(loughran_sentiments_reviews) %>%
  left_join(senticnet_sentiment_reviews) %>%
  left_join(jr_sentiment_reviews) %>%
  na.omit()

# Visualising the sign of the words (positive/negative) of each sentiment dic
tionary to understand the distribution of their contents of the tokenized rev
iews
bing_plot <- all_sentiments_together_reviews %>%
  mutate(index = row_number(), sign = sign(bing_liu_sentiment)) %>%
  ggplot(aes(x = index, y = bing_liu_sentiment, fill = sign)) +
  geom_bar(stat = "identity") +
  labs(y = "Bing Liu Sentiment")
nrc_plot <- all_sentiments_together_reviews %>%
```

```
  mutate(index = row_number(), sign = sign(sentiment_nrc)) %>%
  ggplot(aes(x = index, y = sentiment_nrc, fill = sign)) +
  geom_bar(stat = "identity") +
  labs(y = "NRC Sentiment")
afinn_plot <- all_sentiments_together_reviews %>%
  mutate(index = row_number(), sign = sign(sentiment_afinn)) %>%
  ggplot(aes(x = index, y = sentiment_afinn, fill = sign)) +
  geom_bar(stat = "identity") +
  labs(y = "Afinn Sentiment")
lm_plot <- all_sentiments_together_reviews %>%
  mutate(index = row_number(), sign = sign(sentiment_loughran)) %>%
  ggplot(aes(x = index, y = sentiment_loughran, fill = sign)) +
  geom_bar(stat = "identity") +
  labs(y = "Loughran Sentiment")
senticnet_plot <- all_sentiments_together_reviews %>%
  mutate(index = row_number(), sign = sign(sentiment_senticnet)) %>%
  ggplot(aes(x = index, y = sentiment_senticnet, fill = sign)) +
  geom_bar(stat="identity") +
  labs(y = "Senticnet Sentiment")
jr_plot <- all_sentiments_together_reviews %>%
  mutate(index = row_number(), sign = sign(sentiment_jr)) %>%
  ggplot(aes(x = index, y = sentiment_jr, fill = sign)) +
  geom_bar(stat="identity") +
  labs(y = "Jockers Rinker Sentiment")

ggpubr::ggarrange(bing_plot, nrc_plot, afinn_plot, lm_plot, senticnet_plot, j
r_plot, common.legend = TRUE, legend = "right")

# Visualising the polarities of the sentiment dictionaries
# Bing Liu Dictionary
bing_polarity <- tokenized_reviews %>%
  inner_join(bing_dictionary)
bing_polarity1 <- bing_polarity %>%
  group_by(review_id, sentiment) %>%
  left_join(review_id_word_count) %>%
  summarise(total_words_per_review = n(), word_count = mean(word_count)) %>%
  pivot_wider(names_from = sentiment, values_from = total_words_per_review, v
alues_fill = 0) %>%
  mutate(sentiment = (positive - negative) / (positive + negative))

bing_summary <- bing_polarity %>%
  filter(sentiment %in% c("positive", "negative")) %>%
  group_by(sentiment) %>%
  summarise(total_count = n())

# NRC Dictionary
nrc_polarity <- tokenized_reviews %>%
  inner_join(nrc_dictionary)
nrc_polarity1 <- nrc_polarity  %>%
  filter(sentiment %in% c("positive", "negative")) %>%
```

```
  group_by(review_id, sentiment) %>%
  summarise(total_sentiment = n()) %>%
  group_by(review_id, sentiment) %>%
  left_join(review_id_word_count) %>%
  summarise(total_words_per_review = n(), word_count = mean(word_count)) %>%
  pivot_wider(names_from = sentiment, values_from = total_words_per_review, v
alues_fill = 0) %>%
  mutate(sentiment = (positive - negative) / (positive + negative))

nrc_summary <- nrc_polarity %>%
  filter(sentiment %in% c("positive", "negative")) %>%
  group_by(sentiment) %>%
  summarise(total_count = n())

# Afinn Dictionary
afinn_polarity <- tokenized_reviews %>% inner_join(afinn_dictionary) %>%mutat
e(sentiment = ifelse(afinn_polarity$value > 0, "positive",
                            ifelse(afinn_polarity$value < 0, "negative", "neu
tral")))
afinn_polarity1 <- afinn_polarity%>%
  group_by(review_id, sentiment) %>%
  left_join(review_id_word_count) %>%
  summarise(total_words_per_review = n(), word_count = mean(word_count)) %>%
  pivot_wider(names_from = sentiment, values_from = total_words_per_review, v
alues_fill = 0) %>%
  mutate(sentiment = (positive - negative) / (positive + negative))

afinn_summary <- afinn_polarity %>%
  group_by(sentiment) %>%
  summarise(total_count = n())

# Loughran Dictionary
lm_polarity <- tokenized_reviews %>%
  inner_join(lm_dictionary)
lm_polarity1 <- lm_polarity %>%
  group_by(review_id, sentiment) %>%
  left_join(review_id_word_count) %>%
  summarise(total_words_per_review = n(), word_count = mean(word_count)) %>%
  pivot_wider(names_from = sentiment, values_from = total_words_per_review, v
alues_fill = 0) %>%
  mutate(sentiment = (positive - negative) / (positive + negative))

lm_summary <- lm_polarity %>%
  filter(sentiment %in% c("positive", "negative")) %>%
  group_by(sentiment) %>%
  summarise(total_count = n())

# Senticnet Dictionary
senticnet_polarity <- tokenized_reviews %>% inner_join(senticnet_dictionary)
```

```
%>% mutate(sentiment = ifelse(senticnet_polarity$value > 0, "positive",
                              ifelse(senticnet_polarity$value < 0, "negative",
"neutral")))
senticnet_polarity1 <- senticnet_polarity %>%
  group_by(review_id, sentiment) %>%
  left_join(review_id_word_count) %>%
  summarise(total_words_per_review = n(), word_count = mean(word_count)) %>%
  pivot_wider(names_from = sentiment, values_from = total_words_per_review, v
alues_fill = 0) %>%
  mutate(sentiment = (positive - negative) / (positive + negative))

senticnet_summary <- senticnet_polarity %>%
  filter(sentiment %in% c("positive", "negative")) %>%
  group_by(sentiment) %>%
  summarise(total_count = n())

# Jockers Rinker Dictionary
jockers_rinker_polarity <- tokenized_reviews %>%
  inner_join(jockers_rinker_dictionary)  %>% mutate(sentiment = ifelse(jocker
s_rinker_polarity$value > 0, "positive",
                              ifelse(jockers_rinker_polarity$value < 0, "negati
ve", "neutral")))
jockers_rinker_polarity1 <- jockers_rinker_polarity %>%
  group_by(review_id, sentiment) %>%
  left_join(review_id_word_count) %>%
  summarise(total_words_per_review = n(), word_count = mean(word_count)) %>%
  pivot_wider(names_from = sentiment, values_from = total_words_per_review, v
alues_fill = 0) %>%
  mutate(sentiment = (positive - negative) / (positive + negative))

jockers_rinker_summary <- jockers_rinker_polarity %>%
  filter(sentiment %in% c("positive", "negative")) %>%
  group_by(sentiment) %>%
  summarise(total_count = n())

# Comparing different sentiment dictionaries
polarity_comparison <- data.frame(polarity = c("Negative", "Positive"),
                                  bing_pol = bing_summary$total_count,
                                  nrc_pol = nrc_summary$total_count,
                                  afinn_pol = afinn_summary$total_count,
                                  lm_pol = lm_summary$total_count,
                                  senticnet_pol = senticnet_summary$total_cou
nt,
                                  jr_pol = jockers_rinker_summary$total_count
)
polarity_comparison <- polarity_comparison %>%
  pivot_longer(!polarity, names_to = "Dictionary", values_to = "Total")

polarity_comparison %>%
  ggplot(aes(Dictionary, Total)) + geom_col(aes(fill = polarity), position =
```

```
position_dodge2(preserve = "single")) + geom_text(aes(label = Total), positio
n = position_dodge2(preserve = "single"), hjust = 0.5, vjust = 0.5)
```

Information Gain on Review Level

```
dictionary_information_gain <- information.gain(star~., all_sentiments_togeth
er_reviews[c("star", "bing_liu_sentiment", "sentiment_nrc", "sentiment_afinn"
, "sentiment_loughran", "sentiment_senticnet", "sentiment_jr")])
print(dictionary_information_gain)
# It is observed that Bing Liu Dictionary has the highest information gain fo
r review level documents

all_sentiments_together_reviews <- all_sentiments_together_reviews %>%
  mutate(star = factor(star, levels = c('1','2','3','4','5'), ordered = TRUE)
)

set.seed(1)
split <- sample.split(all_sentiments_together_reviews$star, SplitRatio = 0.70
)
training <- subset(all_sentiments_together_reviews, split == TRUE)
test <- subset(all_sentiments_together_reviews, split == FALSE)

bing_model_reviews_training <- polr(star ~ bing_liu_sentiment, data = trainin
g)
nrc_model_reviews_training <- polr(star ~ sentiment_nrc, data = training)
afinn_model_reviews_training <- polr(star ~ sentiment_afinn, data = training)
lm_model_reviews_training <- polr(star ~ sentiment_loughran, data = training)
senticnet_model_reviews_training <- polr(star ~ sentiment_senticnet, data = t
raining)
jr_model_reviews_training <- polr(star ~ sentiment_jr, data = training)

summary(bing_model_reviews_training)
summary(nrc_model_reviews_training)
summary(afinn_model_reviews_training)
summary(lm_model_reviews_training)
summary(senticnet_model_reviews_training)
summary(jr_model_reviews_training)

stargazer(bing_model_reviews_training, nrc_model_reviews_training, afinn_mode
l_reviews_training, lm_model_reviews_training, senticnet_model_reviews_traini
ng, jr_model_reviews_training, type = "text")
# The model built from the sentiment scores calculated by the sentiment dicti
onaries, Afinn Dictionary has the smallest AIC value, hence this model is a b
etter choice for predicting on a review level.

test$bing_liu_sentiment_predict <- predict(bing_model_reviews_training, test)
test$sentiment_nrc_predict <- predict(nrc_model_reviews_training, test)
test$sentiment_afinn_predict <- predict(afinn_model_reviews_training, test)
test$sentiment_loughran_predict <- predict(lm_model_reviews_training, test)
test$sentiment_senticnet_predict <- predict(senticnet_model_reviews_training,
```

```r
test)
test$sentiment_jr_predict <- predict(jr_model_reviews_training, test)

# Calculating the accuracies of the dictionaries
accuracy <- vector()
accuracy$bing <- length(which(as.numeric(test$bing_liu_sentiment_predict) ==
as.numeric(test$star)))/nrow(test)
accuracy$nrc <- length(which(as.numeric(test$sentiment_nrc_predict) == as.num
eric(test$star)))/nrow(test)
accuracy$afinn <- length(which(as.numeric(test$sentiment_afinn_predict) == as
.numeric(test$star)))/nrow(test)
accuracy$lm <- length(which(as.numeric(test$sentiment_loughran_predict) == as
.numeric(test$star)))/nrow(test)
accuracy$senticnet <- length(which(as.numeric(test$sentiment_senticnet_predic
t) == as.numeric(test$star)))/nrow(test)
accuracy$jr <- length(which(as.numeric(test$sentiment_jr_predict) == as.numer
ic(test$star)))/nrow(test)
accuracy

# Even though some of the dictionaries such as Bing, Afinn, Lm, and Jockers R
inkers have better accuracy than the others, the accuracy rate is still simil
ar. This may be a reason due to using a small number of data, hence accuracy
check is not giving a good insight on the dictionaries.

all_sentiments_together_reviews$star <- as.numeric(all_sentiments_together_re
views$star)

# Bing Liu for review count and sentiments
bing_model_reviews <- lm(log(star) ~ bing_liu_sentiment,
                         data = all_sentiments_together_reviews)

bing_model_reviews_res <- resid(bing_model_reviews)

plot(all_sentiments_together_reviews$bing_liu_sentiment, bing_model_reviews_r
es, ylab = "Residuals", xlab = "Bing Liu Sentiments")

# NRC affection for review count and sentiments
nrc_model_reviews <- lm(log(star) ~ sentiment_nrc,
                        data = all_sentiments_together_reviews)

nrc_model_reviews_res <- resid(nrc_model_reviews)

plot(all_sentiments_together_reviews$sentiment_nrc, nrc_model_reviews_res, yl
ab = "Residuals", xlab = "NRC Sentiments")

#Afinn for review count and sentiments
afinn_model_reviews <- lm(log(star) ~ sentiment_afinn,
                          data = all_sentiments_together_reviews)
```

```r
afinn_model_reviews_res <- resid(afinn_model_reviews)

plot(all_sentiments_together_reviews$sentiment_afinn, afinn_model_reviews_res
, ylab = "Residuals", xlab = "Afinn Sentiments")

# Loughran for review count and sentiments
lm_model_reviews <- lm(log(star) ~ sentiment_loughran,
                       data = all_sentiments_together_reviews)

lm_model_reviews_res <- resid(lm_model_reviews)

plot(all_sentiments_together_reviews$sentiment_loughran, lm_model_reviews_res
, ylab = "Residuals", xlab = "Loughran Sentiments")

# Senticnet for review count and sentiments
senticnet_model_reviews <- lm(log(star) ~ sentiment_senticnet,
                              data = all_sentiments_together_reviews)

senticnet_model_reviews_res <- resid(senticnet_model_reviews)

plot(all_sentiments_together_reviews$sentiment_senticnet, senticnet_model_rev
iews_res, ylab = "Residuals", xlab = "Senticnet Sentiments")

# Jockers Rinker for review count and sentiments
jr_model_reviews <- lm(log(star) ~ sentiment_jr,
                       data = all_sentiments_together_reviews)

jr_model_reviews_res <- resid(jr_model_reviews)

plot(all_sentiments_together_reviews$sentiment_jr, jr_model_reviews_res, ylab
= "Residuals", xlab = "Jockers Rinker Sentiments")



stargazer(bing_model_reviews, nrc_model_reviews, afinn_model_reviews, lm_mode
l_reviews, senticnet_model_reviews, jr_model_reviews, type = "text", notes.la
bel = "Significance Labels")
# It is observed that the Afinn dictionary has the highest Adjusted R2 value,
hence it is the best choice for reviews.
```

Sentiment on Company Level

```r
tokenized_reviews_company <- unigram_grouped %>%
  count(agency_id, word, sort = TRUE)

for_agency_id <- travel_all %>%
  dplyr::select(., c(1, 12)) %>%
  distinct()

for_overall <- travel_all %>%
```

```
  dplyr::select(agency_id, overall_rating) %>%
  distinct()

travel_all <- travel_all %>%
  group_by(agency_id) %>%
  mutate(review_count = n()) %>%
  ungroup()

agency_id_review_count <- travel_all %>%
  dplyr::select(agency_id, review_count)

# Bing Liu - Sentiment Dictionary
bing_liu_sentiment_companies <- tokenized_reviews_company %>%
  inner_join(bing_dictionary) %>%
  count(sentiment, agency_id) %>%
  spread(sentiment, n) %>%
  mutate(bing_liu_sentiment = positive - negative) %>%
  dplyr::select(agency_id, bing_liu_sentiment) %>%
  left_join(for_overall_ratings) %>%
  left_join(agency_id_review_count) %>%
  distinct()

hist(bing_liu_sentiment_companies$overall_rating)

# NRC Dictionary
nrc_emotions_companies <- tokenized_reviews_company %>%
  inner_join(nrc_dictionary) %>%
  count(sentiment, agency_id) %>%
  spread(sentiment, n) %>%
  left_join(for_overall_ratings) %>%
  left_join(agency_id_review_count) %>%
  mutate(sentiment_nrc = positive - negative) %>%
  distinct()

hist(nrc_emotions_companies$overall_rating)

# Afinn Dictionary
afinn_sentiment_companies <- tokenized_reviews_company %>%
  inner_join(afinn_dictionary) %>%
  group_by(agency_id) %>%
  summarise(sentiment_afinn = sum(value)) %>%
  left_join(for_overall_ratings) %>%
  left_join(agency_id_review_count) %>%
  distinct()

hist(afinn_sentiment_companies$overall_rating)

# Loughran Dictionary
loughran_sentiments_companies <- tokenized_reviews_company %>%
```

```
  inner_join(lm_dictionary) %>%
  count(sentiment, agency_id) %>%
  spread(sentiment, n) %>%
  left_join(for_overall_ratings) %>%
  left_join(agency_id_review_count) %>%
  mutate(sentiment_loughran = positive - negative) %>%
  dplyr::select(agency_id, review_count, sentiment_loughran, overall_rating)
%>%
  distinct()

hist(loughran_sentiments_companies$overall_rating)

# Senticnet Dictionary
senticnet_sentiment_companies <- tokenized_reviews_company %>%
  inner_join(senticnet_dictionary) %>%
  group_by(agency_id) %>%
  summarise(sentiment_senticnet = sum(value)) %>%
  left_join(for_overall_ratings) %>%
  left_join(agency_id_review_count) %>%
  distinct()

hist(senticnet_sentiment_companies$overall_rating)

# Jockers Rinker Dictionary
jr_sentiment_companies <- tokenized_reviews_company %>%
  inner_join(jockers_rinker_dictionary) %>%
  group_by(agency_id) %>%
  summarise(sentiment_jr = sum(value)) %>%
  left_join(for_overall_ratings) %>%
  left_join(agency_id_review_count) %>%
  distinct()

hist(jr_sentiment_companies$overall_rating)

# Joining all of the sentiments together
all_sentiments_together_companies <- bing_liu_sentiment_companies %>%
  left_join(nrc_emotions_companies) %>%
  left_join(afinn_sentiment_companies) %>%
  left_join(loughran_sentiments_companies) %>%
  left_join(senticnet_sentiment_companies) %>%
  left_join(jr_sentiment_companies) %>%
  na.omit()

# Extracting feelings from the NRC dictionary
nrc_feelings <- nrc_emotions_companies %>%
  dplyr::select(-c(negative, positive, overall_rating, review_count)) %>%
  pivot_longer(anger:sentiment_nrc, names_to = "feeling", values_to = "sentim
ent_nrc")
# Plotting the feelings extracted
```

```
nrc_feelings %>%
  ggplot(aes(x = agency_id, y = sentiment_nrc, fill = feeling))+
  geom_smooth()+
  facet_wrap(~feeling, scales = "free_y", ncol = 2)
```

Information Gain on Company Level

```
dictionary_information_gain_grouped <- information.gain(overall_rating~., all
_sentiments_together_companies[c("overall_rating", "bing_liu_sentiment", "sen
timent_nrc", "sentiment_afinn", "sentiment_loughran", "sentiment_senticnet",
"sentiment_jr")])
print(dictionary_information_gain_grouped)
# It is observed that similarly, Bing Liu Dictionary has the highest informat
ion gain for company level documents

all_sentiments_together_companies <- all_sentiments_together_companies %>%
  mutate(overall_rating = factor(overall_rating, levels = c('1','2','3','4','
5'), ordered = TRUE))

set.seed(1)
split <- sample.split(all_sentiments_together_companies$overall_rating, Split
Ratio = 0.70)
training <- subset(all_sentiments_together_companies, split == TRUE)
test <- subset(all_sentiments_together_companies, split == FALSE)

bing_model_company_training <- polr(overall_rating ~ bing_liu_sentiment, data
= training)
nrc_model_company_training <- polr(overall_rating ~ sentiment_nrc, data = tra
ining)
afinn_model_company_training <- polr(overall_rating ~ sentiment_afinn, data =
training)
lm_model_company_training <- polr(overall_rating ~ sentiment_loughran, data =
training)
senticnet_model_company_training <- polr(overall_rating ~ sentiment_senticnet
, data = training)
jr_model_company_training <- polr(overall_rating ~ sentiment_jr, data = train
ing)

stargazer(bing_model_company_training, nrc_model_company_training, afinn_mode
l_company_training, lm_model_company_training, senticnet_model_company_traini
ng, jr_model_company_training, type = "text")

summary(bing_model_company_training)
summary(nrc_model_company_training)
summary(afinn_model_company_training)
summary(lm_model_company_training)
summary(senticnet_model_company_training)
summary(jr_model_company_training)
# The model built from the sentiment scores calculated by the sentiment dicti
onaries and the Bing Dictionary has the lowest AIC value, hence this model is
```

```r
a better choice for predicting on a company level.


test$bing_liu_sentiment_predict <- predict(bing_model_company_training, test)
test$sentiment_nrc_predict <- predict(nrc_model_company_training, test)
test$sentiment_afinn_predict <- predict(afinn_model_company_training, test)
test$sentiment_loughran_predict <- predict(lm_model_company_training, test)
test$sentiment_senticnet_predict <- predict(senticnet_model_company_training,
test)
test$sentiment_jr_predict <- predict(jr_model_company_training, test)


accuracy <- vector()
accuracy$bing <- length(which(as.numeric(test$bing_liu_sentiment_predict) ==
as.numeric(test$overall_rating)))/nrow(test)
accuracy$nrc <- length(which(as.numeric(test$sentiment_nrc_predict) == as.num
eric(test$overall_rating)))/nrow(test)
accuracy$afinn <- length(which(as.numeric(test$sentiment_afinn_predict) == as
.numeric(test$overall_rating)))/nrow(test)
accuracy$lm <- length(which(as.numeric(test$sentiment_loughran_predict) == as
.numeric(test$overall_rating)))/nrow(test)
accuracy$senticnet <- length(which(as.numeric(test$sentiment_senticnet_predic
t) == as.numeric(test$overall_rating)))/nrow(test)
accuracy$jr <- length(which(as.numeric(test$sentiment_jr_predict) == as.numer
ic(test$overall_rating)))/nrow(test)
accuracy

# Similarly, even though some of the dictionaries such as Bing, NRC, Afinn, a
nd Jockers Rinkers have better accuracy than the others, the accuracy rate is
still similar. This may be a reason due to using a small number of data, henc
e accuracy check is not giving a good insight on the dictionaries for company
level either.

all_sentiments_together_companies$overall_rating <- as.numeric(all_sentiments
_together_companies$overall_rating)

# Bing Liu for review count and company sentiments
bing_model_companies <- lm(log(overall_rating) ~ bing_liu_sentiment,
                           data = all_sentiments_together_companies)

bing_model_companies_res <- resid(bing_model_companies)

plot(all_sentiments_together_companies$bing_liu_sentiment, bing_model_compani
es_res, ylab = "Residuals", xlab = "Bing Liu Sentiments")

# NRC affection for review count and company sentiments
nrc_model_companies <- lm(log(overall_rating) ~ sentiment_nrc,
                          data = all_sentiments_together_companies)
```

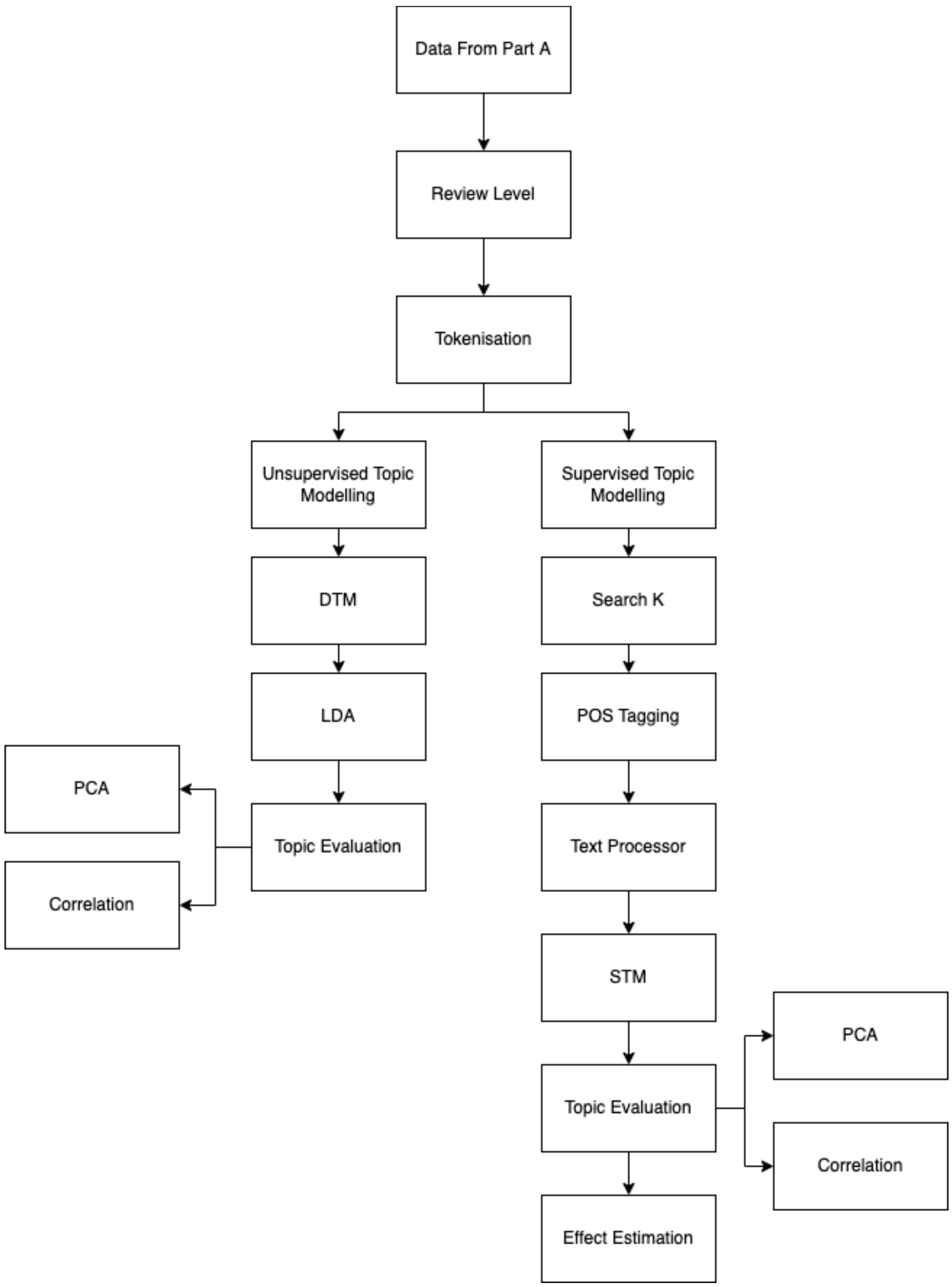
```r
nrc_model_companies_res <- resid(nrc_model_companies)

plot(all_sentiments_together_companies$sentiment_nrc, nrc_model_companies_res
, ylab = "Residuals", xlab = "NRC Sentiments")

#Afinn for review count and company sentiments
afinn_model_companies <- lm(log(overall_rating) ~ sentiment_afinn,
                            data = all_sentiments_together_companies)

afinn_model_companies_res <- resid(afinn_model_companies)

plot(all_sentiments_together_companies$sentiment_afinn, afinn_model_companies
_res, ylab = "Residuals", xlab = "Afinn Sentiments")

# Loughran for review count and company sentiments
lm_model_companies <- lm(log(overall_rating) ~ sentiment_loughran,
                         data = all_sentiments_together_companies)

lm_model_companies_res <- resid(lm_model_companies)

plot(all_sentiments_together_companies$sentiment_loughran, lm_model_companies
_res, ylab = "Residuals", xlab = "Loughran Sentiments")

# Senticnet for review count and company sentiments
senticnet_model_companies <- lm(log(overall_rating) ~ sentiment_senticnet,
                                data = all_sentiments_together_companies)

senticnet_model_companies_res <- resid(senticnet_model_companies)

plot(all_sentiments_together_companies$sentiment_senticnet, senticnet_model_c
ompanies_res, ylab = "Residuals", xlab = "Senticnet Sentiments")

# Jockers Rinker for review count and company sentiments
jr_model_companies <- lm(log(overall_rating) ~ sentiment_jr,
                         data = all_sentiments_together_companies)

jr_model_companies_res <- resid(jr_model_companies)

plot(all_sentiments_together_companies$sentiment_jr, jr_model_companies_res,
ylab = "Residuals", xlab = "Jockers Rinker Sentiments")


stargazer(bing_model_companies, nrc_model_companies, afinn_model_companies, l
m_model_companies, senticnet_model_companies, jr_model_companies, type = "tex
t", notes.label = "Significance Labels")
# It is observed that the Bing Liu dictionary has the highest Adjusted R2 val
ue, hence it is the best choice for company level sentiments.
```

```r
all_sentiments_together_reviews %>%
  pivot_longer(Input, )
  dplyr::select(bing_liu_sentiment, sentiment_nrc, sentiment_afinn, sentiment
_loughran, sentiment_senticnet, sentiment_jr, star) %>%
  na.omit() %>%
  ggplot(aes(x = bing_liu_sentiment, y = log(star))) +
  geom_point(size = 2, shape = 1, alpha = 0.1) +
  xlab("Bing-Liu Sentiments")

gr_nrc_1 = all_sentiments %>%
  dplyr::select(nrc_sentiment,review_scores_rating) %>%
  na.omit() %>%
  ggplot(aes(x=nrc_sentiment,y=log(review_scores_rating)))  + geom_smooth(met
hod="lm"
) + geom_point(size = 2, shape=1,alpha=0.1) + xlab("NRC")
gr_afinn_1 = all_sentiments %>%
  dplyr::select(afinn_sentiment,review_scores_rating) %>%
  na.omit() %>%
  ggplot(aes(x=afinn_sentiment,y=log(review_scores_rating)))  + geom_smooth(m
ethod="l
m") + geom_point(size = 2, shape=1,alpha=0.1) + xlab("afinn")
gr_loughran_1 = all_sentiments %>%
  dplyr::select(loughran_sentiment,review_scores_rating) %>%
  na.omit() %>%
  ggplot(aes(x=loughran_sentiment,y=log(review_scores_rating))) + geom_smooth
(method=
"lm") + geom_point(size = 2, shape=1,alpha=0.1) + xlab("Loughran")
grid.arrange(gr_bl_1, gr_nrc_1, gr_afinn_1, gr_loughran_1,
          ncol = 2, nrow = 2)
```

Part C

Section Plan

In this part, topic modelling was performed on the reviews. First, an unsupervised approach is followed through the usage of perplexity. However, then a supervised approach is followed where the number of topics was chosen according to the Held-Out-Likelihood, Semantic Coherence, and Residual value were taken into account. Lastly, the time series of topics throughout the years and the expected topic proportions are visualized in order to visualize their trends through the years, hence a plan to which aspects to focus can be created by the travel agencies.



Data Preparation

```
# Loading the partially cleaned data from Part A
travel_all <- readRDS("travel_all.rds")

#
travel_all <- travel_all %>%
  select(review_id, agency_id, summary_review, overall_rating, star, review_l
ength, year, month)
```

Unsupervised Topic Modelling on Company Level

Creating Review Text Summaries

```
# Tokenizing the reviews and their summaries
reviews_tokens <- unnest_tokens(travel_all, word, summary_review)
# Removing the stop words from tidyverse package
reviews_tokens <- reviews_tokens %>%
  anti_join(stop_words)

## Joining, by = "word"

# Lemmatizing the tokens
reviews_tokens$word <- lemmatize_words(reviews_tokens$word)
# After visualising the topics later on, the following stop words are added t
o a custom stop word dictionary - before that it was empty
custom_stop_words_c <- c("dot", "dollar", "time", "company", "customer", "hel
p", "time",
                         "day", "hour", "month", "tell", "egypt", "week", "sa
fari")
custom_stop_words_df_c <- data.frame(word = custom_stop_words_c, lexicon = re
p("custom", length(custom_stop_words_c)))
# Custom stop words are removed
reviews_tokens <- reviews_tokens %>%
  anti_join(custom_stop_words_df_c)

## Joining, by = "word"
```

Creating a DTM

```
# Casting DTM to find the topics
dtm_for_topics <- reviews_tokens %>%
  count(agency_id, word) %>%
  cast_dtm(agency_id, word, n)
inspect(dtm_for_topics)
```

```
## <<DocumentTermMatrix (documents: 548, terms: 32957)>>
## Non-/sparse entries: 266642/17793794
## Sparsity           : 99%
## Maximal term length: 39
## Weighting          : term frequency (tf)
## Sample             :
##      Terms
## Docs  book call experience flight hotel refund service tour travel trip
##   112  209  105         22      3    14    155     113    0      5   39
##   113   80   97         48      6    20     10      32    6     28   26
##   618   50   56         38      2     2     22      50    0     18   18
##   625   82   31         26     18    70     37      63   27     20   77
##   716  268   90         26    222    68    113      67    1     23   56
##   766   62   84         15      0   281     36      57    0      6    9
##   773   86   63         38    235    16     57     113    0     44   17
##   788   41   70         32    235    19     42      79    0     48   33
##   790   59   56         48    269    17     31     109    0     37   19
##   792  339   96         41      6   710    128     136    1     17   21

# Since the sparsity was 99%, some of the sparsity is removed
dtm_for_topics_sparse <- removeSparseTerms(dtm_for_topics, sparse = 0.8)
inspect(dtm_for_topics_sparse)

## <<DocumentTermMatrix (documents: 548, terms: 557)>>
## Non-/sparse entries: 113107/192129
## Sparsity           : 63%
## Maximal term length: 15
## Weighting          : term frequency (tf)
## Sample             :
##      Terms
## Docs  book call experience flight hotel refund service tour travel trip
##   112  209  105         22      3    14    155     113    0      5   39
##   113   80   97         48      6    20     10      32    6     28   26
##   637  154  120         50    160     5    148      85    0     54   29
##   699   93   84         26    198     0    232      94    1     39   25
##   716  268   90         26    222    68    113      67    1     23   56
##   748  193  114         17      4   177    123      53    0     15   11
##   773   86   63         38    235    16     57     113    0     44   17
##   788   41   70         32    235    19     42      79    0     48   33
##   790   59   56         48    269    17     31     109    0     37   19
##   792  339   96         41      6   710    128     136    1     17   21

# Trying 2 topics for LDA using DTM
lda_model <- LDA(dtm_for_topics, k = 2, method = "Gibbs", control = list(seed
= 1))
# Trying 2 topics for LDA using DTM that has been reduced in terms of sparsit
y
lda_model_sparse <- LDA(dtm_for_topics_sparse, k = 2, method = "Gibbs", contr
ol = list(seed = 1))
```

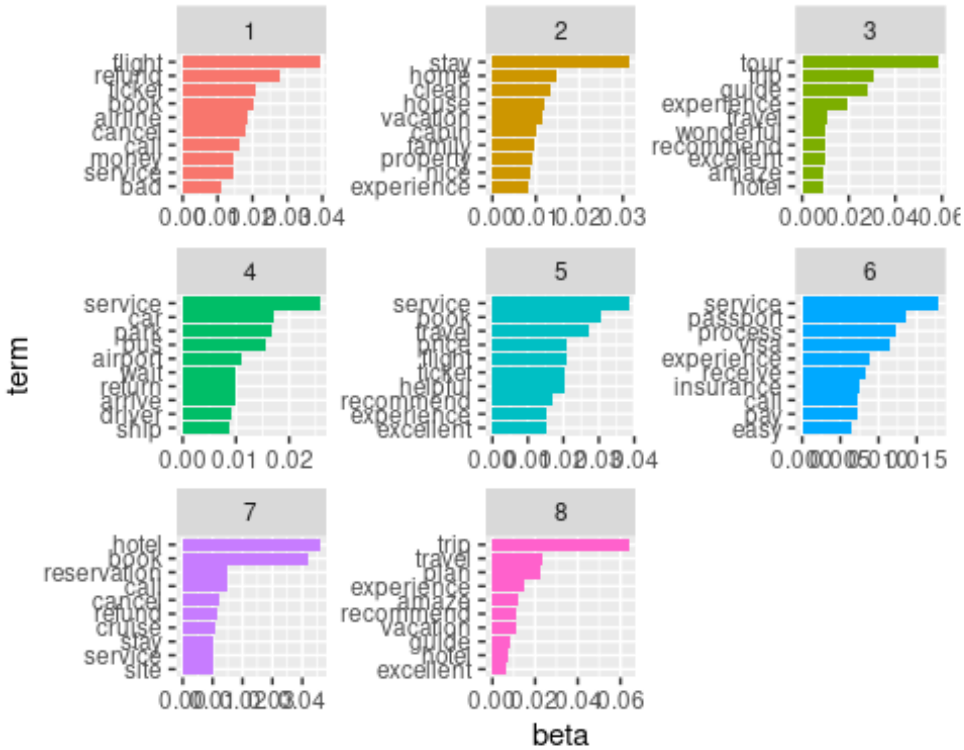
Perplexity Analysis for Evaluating K

```
set.seed(1)
perplexity_df <- data.frame(perplexity_value = numeric())
# Generating perplexity values for topics varying from 2 to 18
for (i in 2:18) {
  fitted <- LDA(dtm_for_topics, k = i, method = "Gibbs")
  perplexity_df[i, 1] <- perplexity(lda_model, dtm_for_topics)
}

# Visualising the perplexity values
ggplot(perplexity_df, aes(x = as.numeric(row.names(perplexity_df)))) +
  labs(x = "Number of Topics", y = "Perplexity", title = "Perplexity") +
  geom_line(aes(y = perplexity_value), color = "black") +
  geom_point(aes(y = perplexity_value), size = 2.5) +
  theme_bw() +
  theme(panel.grid.minor = element_blank())
```

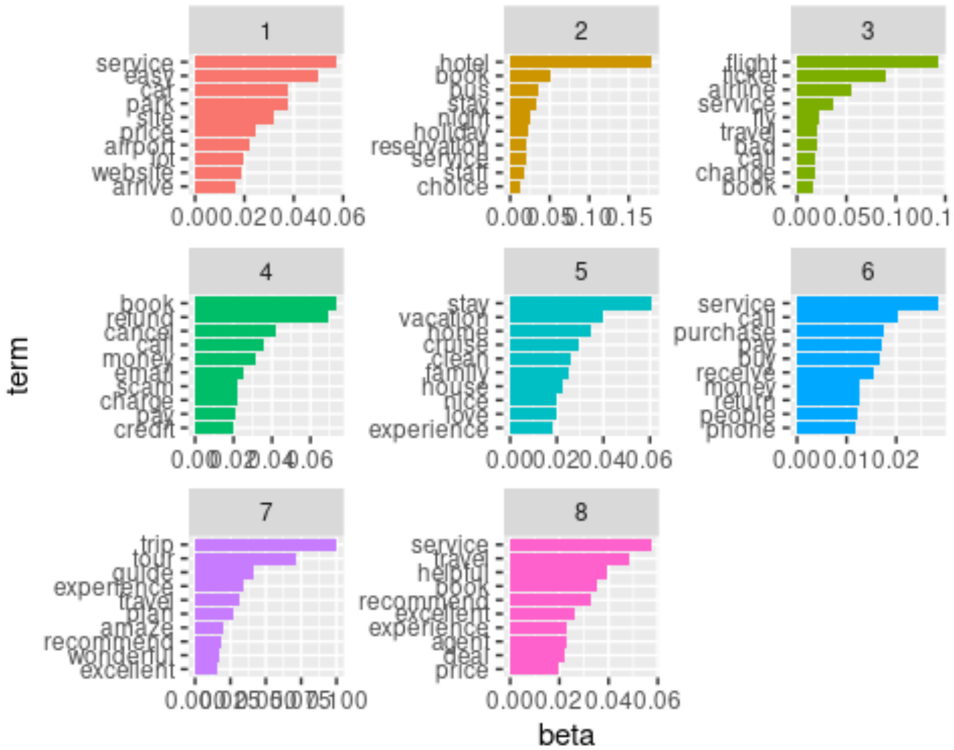
* According to the perplexity plot, 8 is selected as the optimal topic number since it has the lowest perplexity except for the end points in the graph

Topic Modelling

```
# LDA model is re-built using 8 topics
lda_model <- LDA(dtm_for_topics, k = 8, method = "Gibbs", control = list(seed
= 1))
# Using DTM that was reduced in sparsity level is used with 8 topics for a se
parate LDA model
lda_model_sparse <- LDA(dtm_for_topics_sparse, k = 8, method = "Gibbs", contr
ol = list(seed = 1))

# Tidying the untidy models for further analysis according to the beta values
review_topics <- tidy(lda_model, matrix = "beta")
review_topics_sparse <- tidy(lda_model_sparse, matrix = "beta")

# Finding the top 10 terms for each topic having the highest beta values and
plotting them
(review_top_terms <- review_topics %>%
   group_by(topic) %>%
   slice_max(beta, n = 10) %>%
   ungroup() %>%
   arrange(topic, desc(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())
```

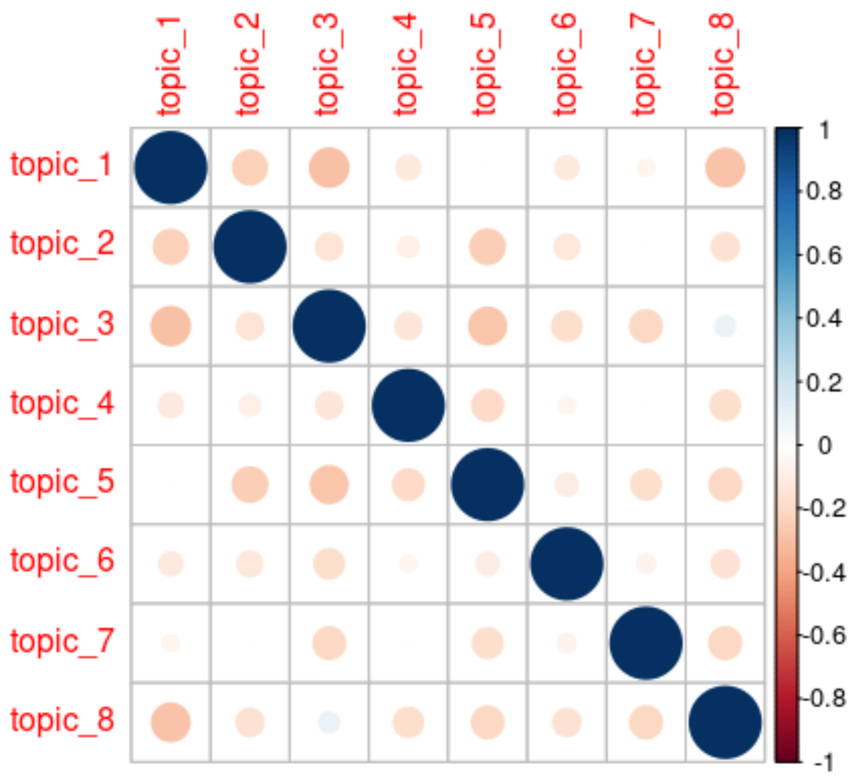


```
# Performing the same thing with the tidied topic model that has been reduced
(review_top_terms_sparse <- review_topics_sparse %>%
    group_by(topic) %>%
    slice_max(beta, n = 10) %>%
    ungroup() %>%
    arrange(topic, desc(beta)) %>%
    mutate(term = reorder_within(term, beta, topic)) %>%
    ggplot(aes(beta, term, fill = factor(topic))) +
    geom_col(show.legend = FALSE) +
    facet_wrap(~ topic, scales = "free") +
    scale_y_reordered())
```

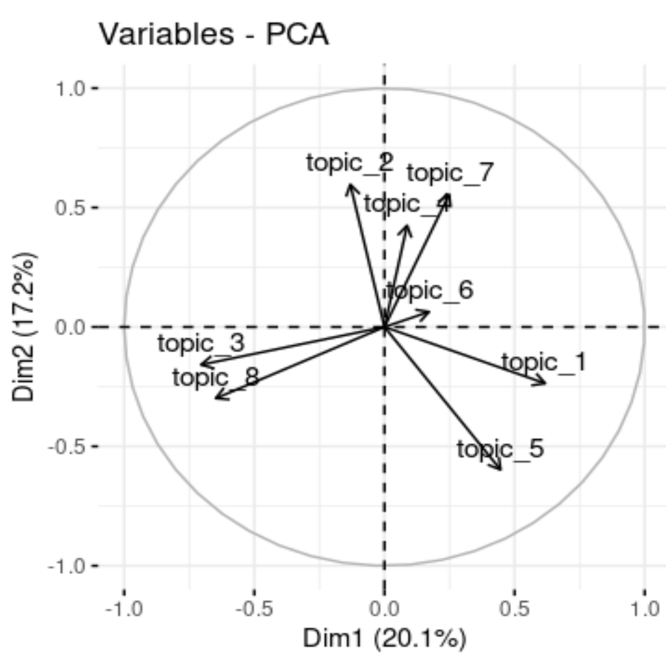


Evaluating the Topics

```
# Renaming the topics after inspecting them
topic_labels <- c("topic_1", "topic_2", "topic_3", "topic_4", "topic_5", "top
ic_6", "topic_7", "topic_8") # Change the topic names !!!!!
# Building the gamma matrix using the LDA model and tidying it
gamma_topics <- tidy(lda_model, matrix = "gamma")
# Transforming the data frame to remove the repeating rows
gamma_topics <- gamma_topics %>%
  pivot_wider(names_from = topic, values_from = gamma)
# Renaming the columns
colnames(gamma_topics) <- c("review", topic_labels)
# Renaming the rows with review numbers
rownames(gamma_topics) <- gamma_topics$review

## Warning: Setting row names on a tibble is deprecated.

# Since the rows are named, the review column is removed
gamma_topics$review <- NULL

# Plotting the correlation among the topics
corrplot(cor(gamma_topics))
```




```
# Performing Factor Component Analysis to visualise the directions of varianc
e of each topic
pcah <- FactoMineR::PCA(gamma_topics, graph = FALSE)
fviz_pca_var(pcah)
```



The PCA is able to cover 37.3% of the variance in the dataset

Supervised Topic Modelling on Company Level

Processing the Text

```
# Loading the English language model
language <- udpipe_download_model(language = "english", overwrite = F)

ud_model <- udpipe_load_model(language$file_model)

# Creating an empty data frame to be used later on
annotated_reviews <- data.frame()

# Since the data set is not very small but not huge either, split size is cho
sen to be 1000
split_size <- 1000

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

POS Tagging

```
# Performing Parts-of-Speech Tagging on the splitted data set
for(i in 1:length(for_pos_list)){annotated_reviews_temp <-
udpipe_annotate(for_pos_list[[i]]$summary_review,
                doc_id = for_pos_list[[i]]$review_id,
                object = ud_model) %>%
      as.data.frame() %>%
      filter(upos %in% c("NOUN","ADJ","ADV")) %>%
      select(doc_id, lemma) %>%
      group_by(doc_id) %>%
      rename(review_id = doc_id) %>%
      summarise(annotated_reviews_collapsed = paste(lemma, collapse =
" "))

  print(paste(i, "from", length(for_pos_list)))
  annotated_reviews <- bind_rows(annotated_reviews,
annotated_reviews_temp)
}
```

```
# saveRDS(annotated_reviews, file = "annotated_reviews.rds")
# annotated_reviews <- readRDS("annotated_reviews.rds")

# Preparing the metadata for the STM model
annotated_reviews$review_id <- as.integer(annotated_reviews$review_id)
```

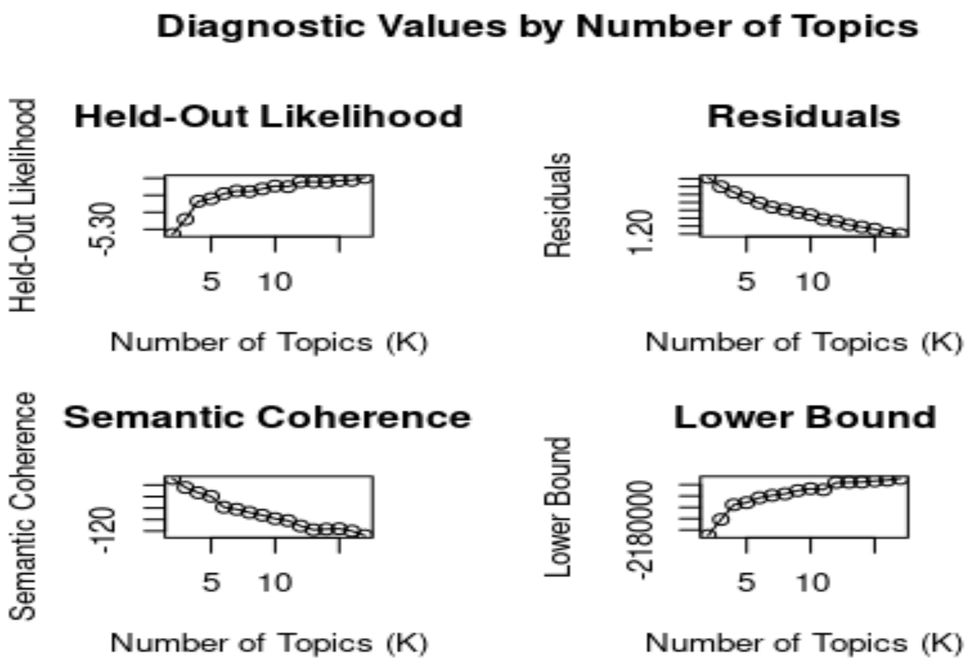
```r
for_stm <- annotated_reviews %>%
  left_join(travel_all)

## Joining, by = "review_id"

for_stm$star <- as.integer(for_stm$star)

# Performing automatic cleaning using the textProcessor function
processed <- textProcessor(for_stm$annotated_reviews_collapsed,
                           metadata = for_stm,
                           customstopwords = c("people", "also", "even", "rea
lly", "just", "dot", "com", "able", "ever", "thanks", "much",
                                               "still", "review", "one", "bac
k", "bag", "way"),
                           #                     "dollar","irish", "diamond",
"inn", "month", "vrbo", "cuba", "crotia", "still", "march", "swiss", "qatar",
"itally", "brian", "parker",
                           #                     "safari", "africa", "costa",
"adventur", "african", "croatia", "safaris", "tanzania", "galapago",
                           #                     "singapore", "cheapoair", "v
ega", "israel", "egypt", "rome", "rica", "costa", "greece", "machu",
                           #                     "picchu", "bora", "indion"),
                           stem = F)

# Keeping only the words that appear on the 1% of the corpus since they have
a higher frequency
threshold <- round(1/100 * length(processed$documents), 0)

out <- prepDocuments(processed$documents,
                     processed$vocab,
                     processed$meta,
                     lower.thresh = threshold)

## Removing 19593 of 19980 terms (185968 of 516938 tokens) due to frequency
## Removing 4 Documents with No Words
## Your corpus now has 25737 documents, 387 terms and 330970 tokens.
```

Evaluating Kappa

Running searchK function to select the optimal number of topics for using STM

```r
topic_num <- searchK(out$documents, out$vocab, K = seq(from = 2, to = 17,
by = 1)) # 2 to 17
```

```r
# Visualising the different topics in terms of Held-Out-Likelihood, Residuals
, Semantic Coherence, and Lower Bound
plot(topic_num)
```



- It is observed that 9 topics gives high enough Held-Out-Likelihood and Semantic Coherence, while the Residuals are on the lower side.

STM Topic Modelling

```
# Executing an STM model using the prevalence function prevalence = ~ star wi
th K = 9 for performing supervised topic modelling
travel_agencies_fit <- stm(documents = out$documents,
                           vocab = out$vocab,
                           K = 9,
                           prevalence = ~ star,
                           max.em.its = 75,
                           data = out$meta,
                           reportevery = 3,
                           sigma.prior = 0,
                           init.type = "LDA")



saveRDS(travel_agencies_fit, "travel_agencies_fit.rds")
summary(travel_agencies_fit)

## A topic model with 9 topics, 25737 documents and a 387 word dictionary.

## Topic 1 Top Words:
##       Highest Prob: family, staff, vacation, place, home, nice, stay
##       FREX: clean, house, location, cabin, place, staff, home
##       Lift: large, cabin, clean, beach, hot, house, pool
```

```
##       Score: house, clean, place, cabin, vacation, stay, nice
## Topic 2 Top Words:
##       Highest Prob: service, customer, never, phone, call, star, worst
##       FREX: bad, worst, terrible, horrible, car, order, poor
##       Lift: poor, rude, bad, order, terrible, disappointed, worst
##       Score: customer, service, worst, never, terrible, phone, horrible
## Topic 3 Top Words:
##       Highest Prob: great, service, excellent, helpful, easy, good, defini
tely
##       FREX: easy, helpful, great, professional, quick, excellent, patient
##       Lift: efficient, courteous, fast, quick, patient, easy, helpful
##       Score: great, excellent, helpful, service, easy, professional, highl
y
## Topic 4 Top Words:
##       Highest Prob: experience, best, travel, always, company, friend, tea
m
##       FREX: best, always, team, job, experience, care, need
##       Lift: team, pleasure, always, best, job, need, world
##       Score: best, experience, travel, always, awesome, team, job
## Topic 5 Top Words:
##       Highest Prob: flight, ticket, airline, hour, price, website, busines
s
##       FREX: flight, airline, ticket, seat, hour, online, return
##       Lift: flight, seat, airline, ticket, delay, return, luggage
##       Score: flight, ticket, airline, hour, price, website, business
## Topic 6 Top Words:
##       Highest Prob: trip, tour, guide, amazing, wonderful, well, knowledge
able
##       FREX: tour, guide, driver, group, adventure, wonderful, knowledgeabl
e
##       Lift: guide, tour, adventure, drivers, excursion, history, interesti
ng
##       Score: tour, guide, trip, amazing, wonderful, knowledgeable, fantast
ic
## Topic 7 Top Words:
##       Highest Prob: hotel, booking, room, site, reservation, last, minute
##       FREX: site, reservation, minute, point, hotel, last, booking
##       Lift: point, site, desk, minute, fine, reservation, party
##       Score: hotel, room, reservation, booking, night, site, minute
## Topic 8 Top Words:
##       Highest Prob: time, day, first, year, good, many, next
##       FREX: first, cruise, time, many, second, year, next
##       Lift: cruise, package, mile, second, surprise, first, mind
##       Score: time, first, cruise, year, day, many, good
## Topic 9 Top Words:
##       Highest Prob: company, refund, money, dollar, email, now, month
##       FREX: refund, credit, scam, card, dollar, money, fee
##       Lift: card, credit, scam, bank, refund, account, insurance
##       Score: refund, dollar, money, scam, card, credit, email
```

Evaluating the Topics

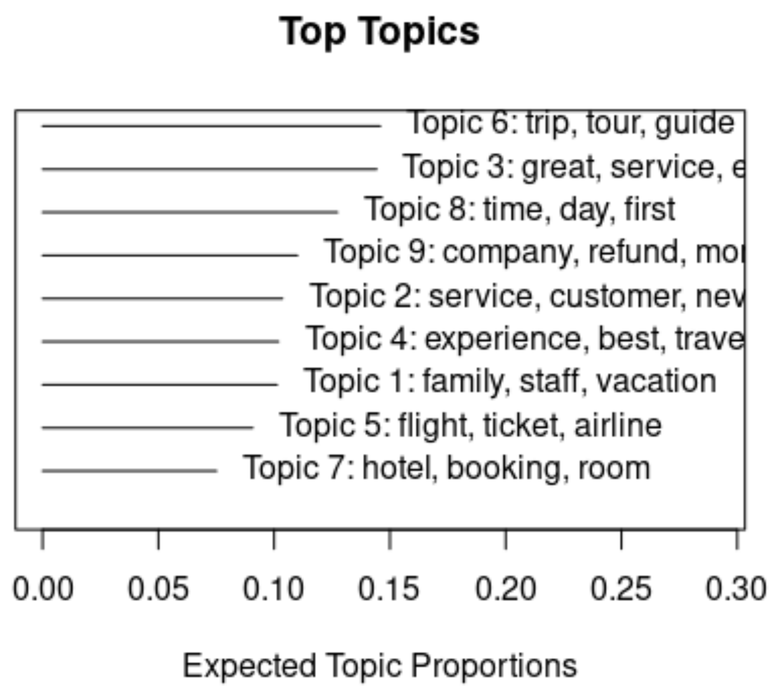
# Calculating the corpus-level theta and the top words for each topic for presenting the topics
topic_summary <- summary(travel_agencies_fit)

```
## A topic model with 9 topics, 25737 documents and a 387 word dictionary.
```

```
topic_proportions <- colMeans(travel_agencies_fit$theta)
# Renaming the topics according to the high probability words that may appear
topic_labels <- c("Service Quality", "Flight Tickets", "Baggage Checking", "T
our Guide", "Vacation Experience", "Lodging", "Staff Behaviours", "Online Vis
a Issuance", "Refunds")

# Creating an empty data frame for labeling the topics and summarising the wo
rd components
table_to_write_labels <- data.frame()

for(i in 1:length(topic_summary$topicnums)){

   row_temp <- tibble(topicnum= topic_summary$topicnums[i],
                      topic_label = topic_labels[i],
                      proportion = 100 * round(topic_proportions[i], 4),
                      frex_words = paste(topic_summary$frex[i, 1:7],
                                         collapse = ", "))
   table_to_write_labels <- rbind(row_temp, table_to_write_labels)
}

(table_to_write_labels %>% arrange(topicnum))
```

```
## # A tibble: 9 × 4
##   topicnum topic_label         proportion frex_words
##      <int> <chr>                    <dbl> <chr>
## 1        1 Service Quality          10.1  clean, house, location, cabin,
place…
## 2        2 Flight Tickets           10.4  bad, worst, terrible, horrible,
car,…
## 3        3 Baggage Checking         14.4  easy, helpful, great, professio
nal, …
## 4        4 Tour Guide               10.2  best, always, team, job, experi
ence,…
## 5        5 Vacation Experience       9.07 flight, airline, ticket, seat,
hour,…
## 6        6 Lodging                  14.6  tour, guide, driver, group, adv
entur…
## 7        7 Staff Behaviours          7.51 site, reservation, minute, poin
t, ho…
## 8        8 Online Visa Issuance     12.7  first, cruise, time, many, seco
nd, y…
```

```
## 9        9 Refunds                    11    refund, credit, scam, card, dol
lar, …

# Visualising the model
plot(travel_agencies_fit)
```



```r
# Forming word clouds for each of the topics
stm::cloud(travel_agencies_fit, topic = 1, max.words = 46,
           colors = c("forestgreen", "violetred", "darkgoldenrod3"))
```

```r
stm::cloud(travel_agencies_fit, topic = 2, max.words = 46,
           colors = c("forestgreen", "violetred", "darkgoldenrod3"))
```


```r
stm::cloud(travel_agencies_fit, topic = 3, max.words = 46,
           colors = c("forestgreen", "violetred", "darkgoldenrod3"))
```

```r
stm::cloud(travel_agencies_fit, topic = 4, max.words = 46,
           colors = c("forestgreen", "violetred", "darkgoldenrod3"))
```

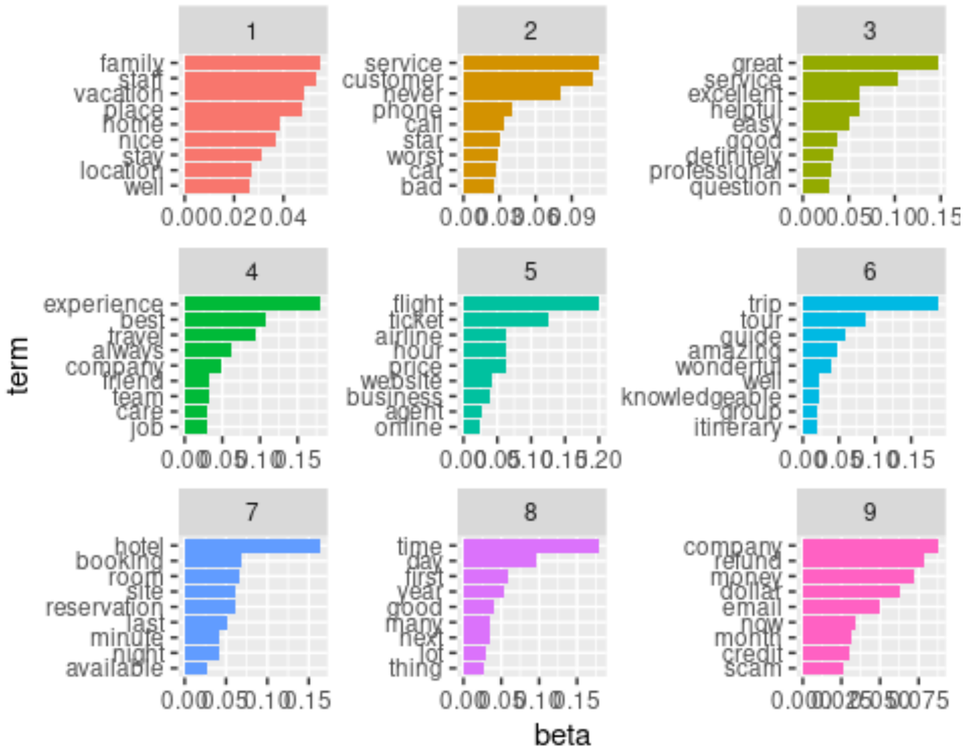

```r
stm::cloud(travel_agencies_fit, topic = 5, max.words = 46,
           colors = c("forestgreen", "violetred", "darkgoldenrod3"))
```

```r
stm::cloud(travel_agencies_fit, topic = 6, max.words = 46,
           colors = c("forestgreen", "violetred", "darkgoldenrod3"))
```

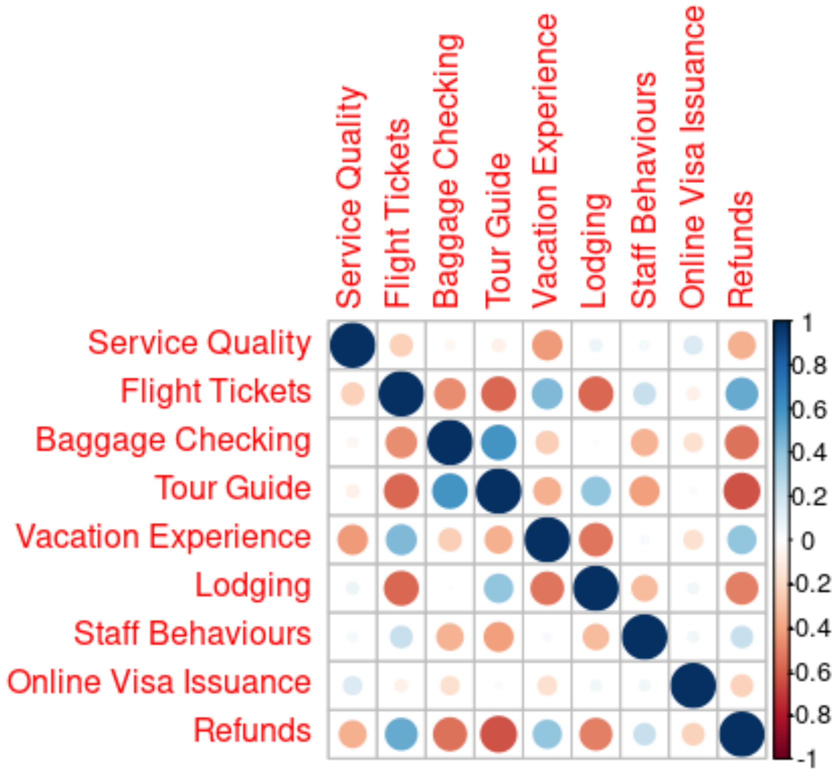

```r
stm::cloud(travel_agencies_fit, topic = 7, max.words = 46,
           colors = c("forestgreen", "violetred", "darkgoldenrod3"))
```

```
stm::cloud(travel_agencies_fit, topic = 8, max.words = 46,
           colors = c("forestgreen", "violetred", "darkgoldenrod3"))
```

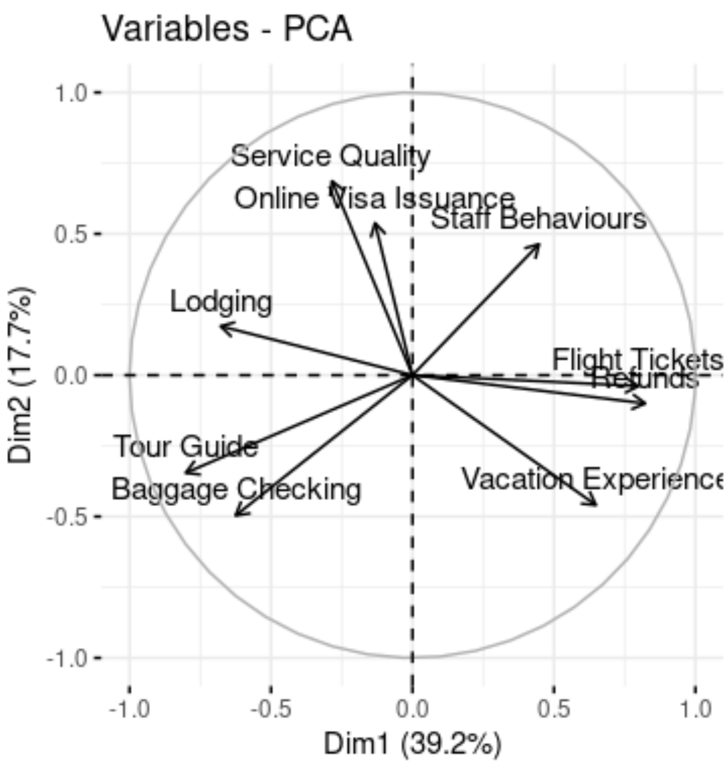

```
# Since the model is not in tidy format, it is tidied according to the beta v
alues
stm_tidy <- tidy(travel_agencies_fit, matrix = "beta")

# Visualising the top 9 words of each topic according to their beta values
stm_tidy %>%
   group_by(topic) %>%
   slice_max(beta, n = 9) %>%
   ungroup() %>%
   arrange(topic, desc(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()
```




```
# Extracting the theta matrix from the fitted STM model
gamma_topics <- cbind(out$meta, travel_agencies_fit$theta)
gamma_topics$doc_id <- 1:nrow(gamma_topics)
gamma_topics <- gamma_topics %>%
  select(10:19)

# Renaming the columns according to the topic labels
colnames(gamma_topics) <- c(topic_labels, "doc_id")
# Renaming the rows according to the doc IDs
rownames(gamma_topics) <- gamma_topics$doc_id
gamma_topics$doc_id <- NULL

# Visualising the correlation among the 9 topics that were extracted from the
supervised topic model
corrplot::corrplot(cor(gamma_topics))
```

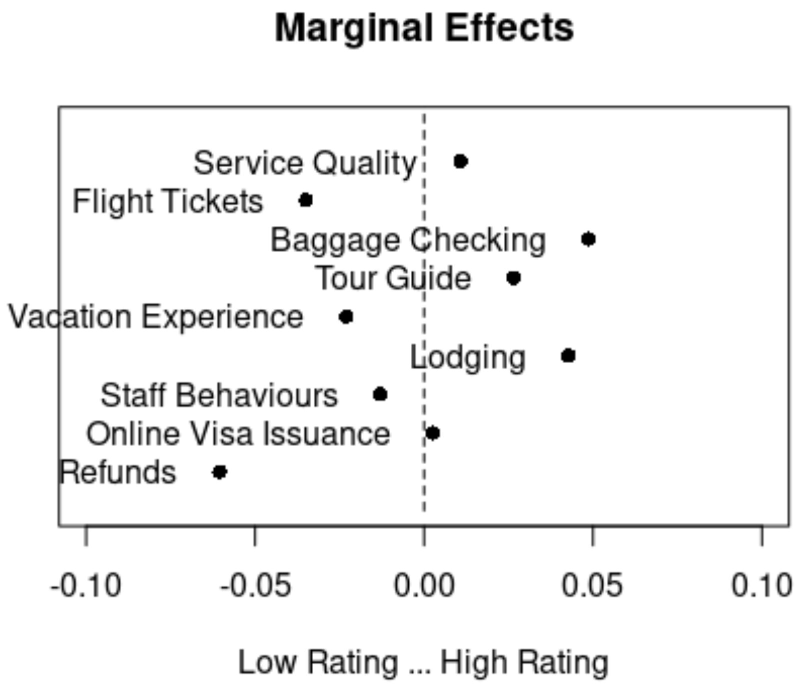



```
# Performing Factor Component Analysis to visualise the directions of varianc
e of each topic
pcah <- FactoMineR::PCA(gamma_topics, graph = FALSE)
fviz_pca_var(pcah)
```

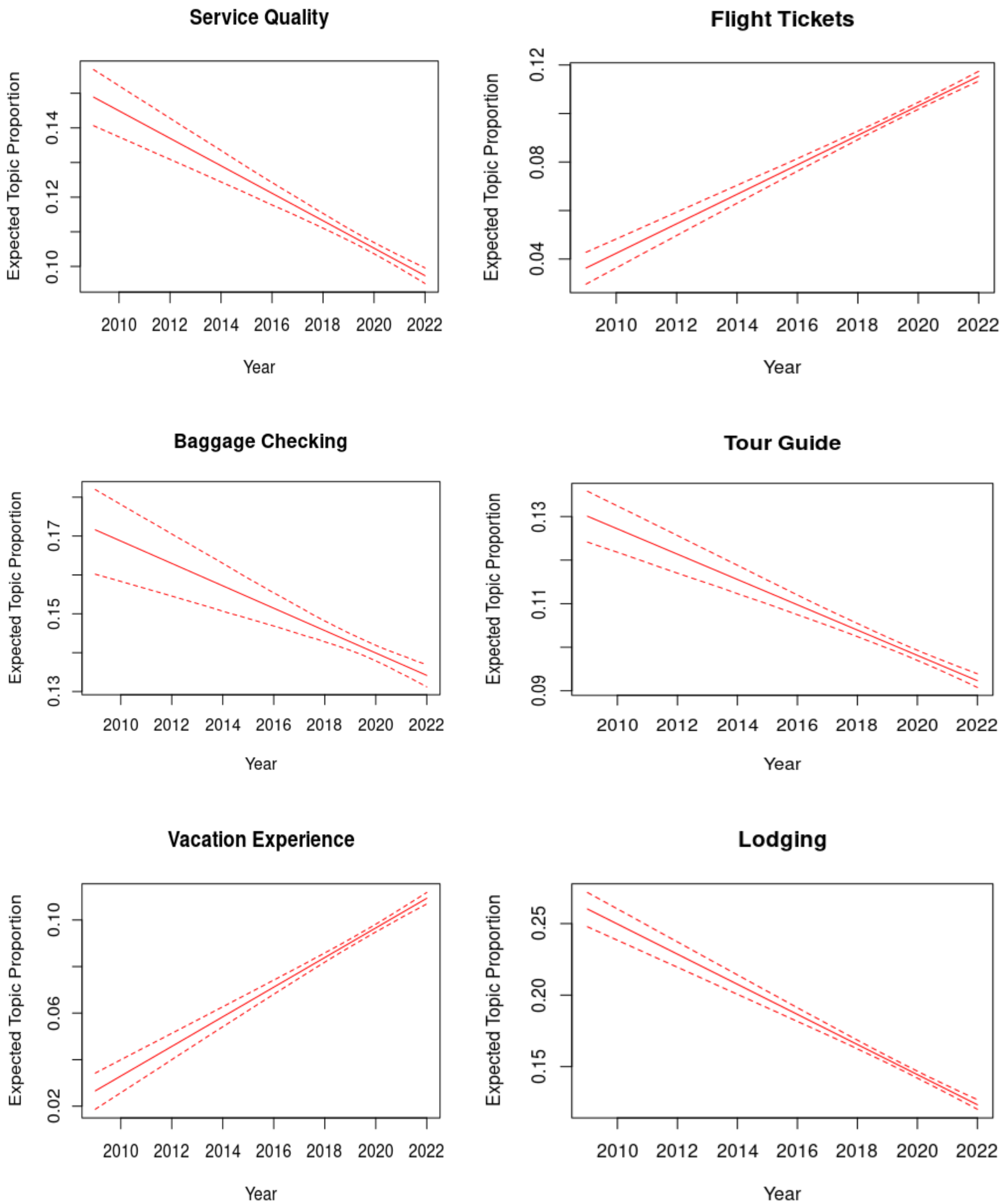


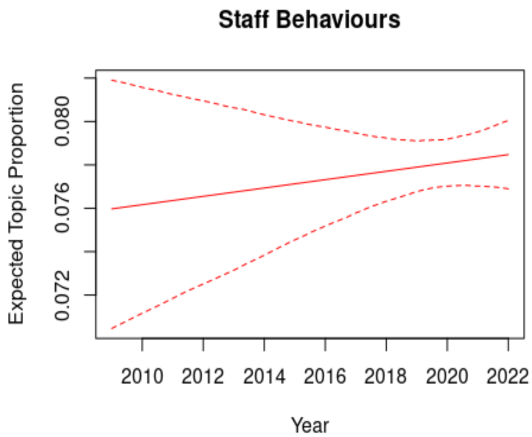
Effect Estimations

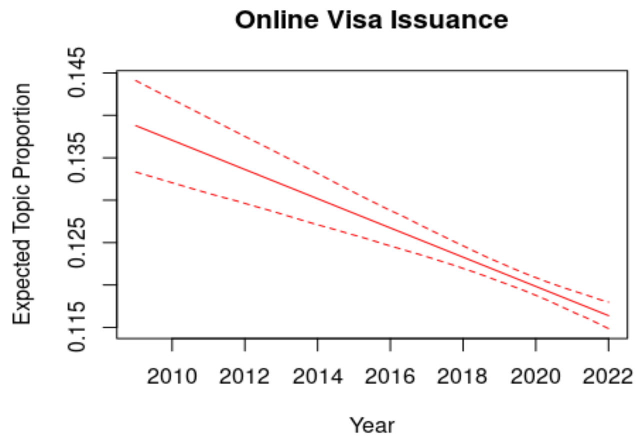
```
# Estimating the effect of review ratings on the topics
effects <- estimateEffect(~ star,
                          stmobj = travel_agencies_fit,
                          metadata = out$meta)

# Visualising the effect
plot(effects, covariate = "star",
     topics = c(1:9),
     model = travel_agencies_fit, method = "difference",
     cov.value1 = "5", cov.value2 = "1",
     xlab = "Low Rating ... High Rating",
     xlim = c(-0.1, 0.1),
     main = "Marginal Effects",
     custom.labels = topic_labels,
     labeltype = "custom")
```

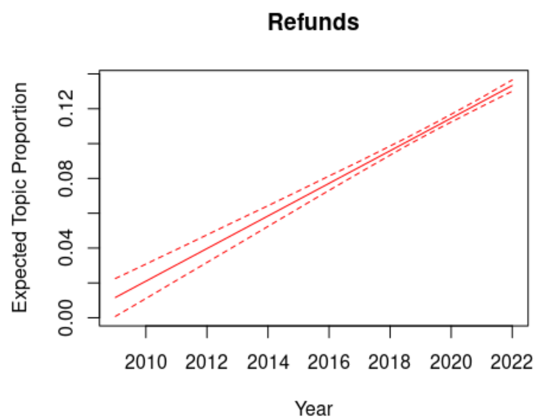



```
# Transforming the data type of "year" into "integer"
out$meta$year <- out$meta$year %>% as.integer()

# Estimating the effect of year on the topics
effects_year <- estimateEffect(~ year,
                          stmobj = travel_agencies_fit,
                          metadata = out$meta)

# Visualising the effect
for(i in 1:length(topic_labels)){
  plot(effects_year, covariate = "year",
       topics = i,
       model = travel_agencies_fit, method = "continuous",
       # For this plotting we get the uper quantile and low quantile of the p
rice
       xlab = "Year",
       # xlim = c(150, 2000),
       main = topic_labels[i],
       printlegend = FALSE,
       custom.labels = topic_labels[i],
       labeltype = "custom")
}
```


Service Quality
Expected Topic Proportion
0.10
0.12
0.14
2010 2012 2014 2016 2018 2020 2022
Year
Flight Tickets
Expected Topic Proportion
0.04
0.08
0.12
2010 2012 2014 2016 2018 2020 2022
Year
Baggage Checking
Expected Topic Proportion
0.13
0.15
0.17
2010 2012 2014 2016 2018 2020 2022
Year
Tour Guide
Expected Topic Proportion
0.09
0.11
0.13
2010 2012 2014 2016 2018 2020 2022
Year
Vacation Experience
Expected Topic Proportion
0.02
0.06
0.10
2010 2012 2014 2016 2018 2020 2022
Year
Lodging
Expected Topic Proportion
0.15
0.20
0.25
2010 2012 2014 2016 2018 2020 2022
Year


Staff Behaviours
Expected Topic Proportion
0.072
0.076
0.080
2010
2012
2014
2016
2018
2020
2022
Year


Online Visa Issuance
Expected Topic Proportion
0.115
0.125
0.135
0.145
2010
2012
2014
2016
2018
2020
2022
Year


Refunds
Expected Topic Proportion
0.00
0.04
0.08
0.12
2010
2012
2014
2016
2018
2020
2022
Year

Conclusion

After analysing the reviews of the travel agencies, it is discovered that refunds are gaining more importance in the customers' lives which may be an indicator of the volatility of their situation and at any moment they can be tested positive for Covid-19 or there may be a lockdown. Hence, in order to adapt to these volatile and unstable times, travel agencies should make a bigger effort to ensure the customers that under unforeseen circumstances, they will get refund. By this way, the customers can be more comfortable and at ease and may pick the travel agency again in the future.