WARWICK BUSINESS SCHOOL

Masters Programmes

Assignment Cover Sheet

Submitted by:	**2215107**
Date Sent:	**18 June 2023**
Module Title:	**Text Analytics**
Module Code:	**IB9CW0**
Date/Year of Module:	**Term 3 2022/2023**
Submission Deadline:	**20 June 2023**
Word Count:	**1451**
Number of Pages:	**39**

Question: For this assignment you are going to work with product reviews from Amazon. You can download a dataset of Amazon reviews

"I declare that this work is entirely my own in accordance with the University's Regulation 11 and the WBS guidelines on plagiarism and collusion. All external references and sources are clearly acknowledged and identified within the contents.

No substantial part(s) of the work submitted here has also been submitted by me in other assessments for accredited courses of study, and I acknowledge that if this has been done it may result in me being reported for self-plagiarism and an appropriate reduction in marks may be made when marking this piece of work."

Table of Contents

1 Text cleaning and normalisation

1.1 Importing libraries and data

```
knitr::opts_chunk$set
```

```
## function (...)
## {
##     set2(resolve(...))
## }
## <bytecode: 0x000001f385d58e08>
## <environment: 0x000001f385d5bc38>
```

```
library(tokenizers)
library(tidyverse)
library(tidytext)
library(topicmodels)
library(textstem)
library(tm)
library(dplyr)
library(qdap)
library(qdapRegex)
library(hcandersenr)
library(hunspell)
library(ggplot2)
library(ggrepel)
library(word2vec)
library(uwot)
library(sentimentr)
library(lexicon)
library(jsonlite)
library(SnowballC)
library(wordcloud)
library(RColorBrewer)
library(Hmisc)
```

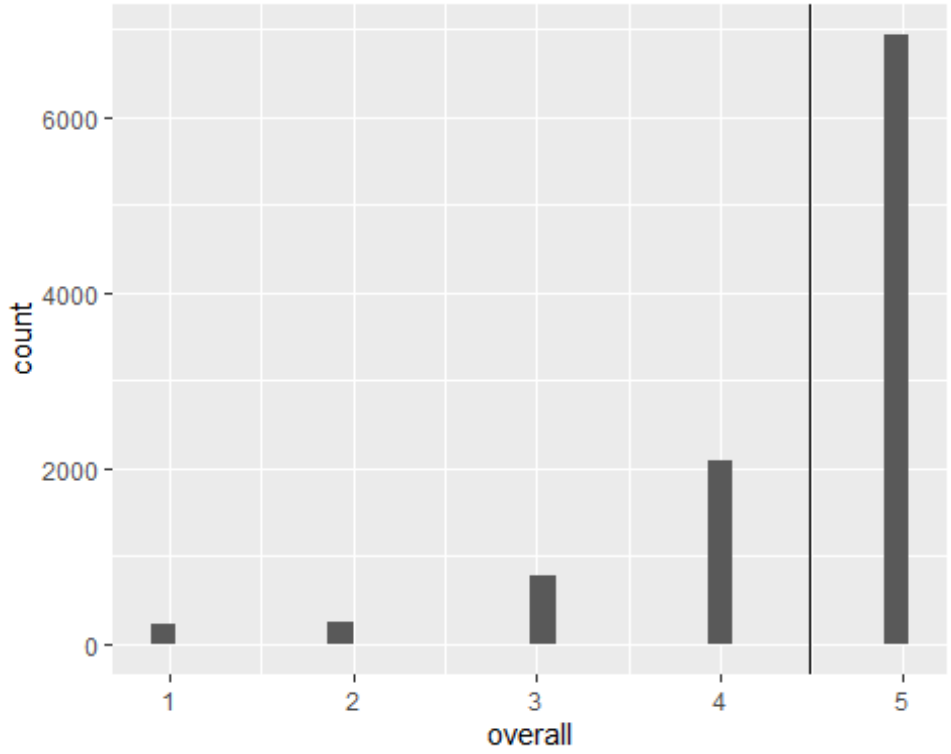
As per assignment question, Musical Instrument reviews, containing 10,254 records, have been chosen.

```
# read JSON file and convert to data frame
json_file <- "reviews_Musical_Instruments_5.json.gz"
json_lines <- readLines(json_file)

# wrap the JSON data in square brackets
json_data <- paste0("[", paste(json_lines, collapse=","), "]")

# convert the JSON data to a data frame
review <- fromJSON(json_data, simplifyDataFrame = TRUE)
```

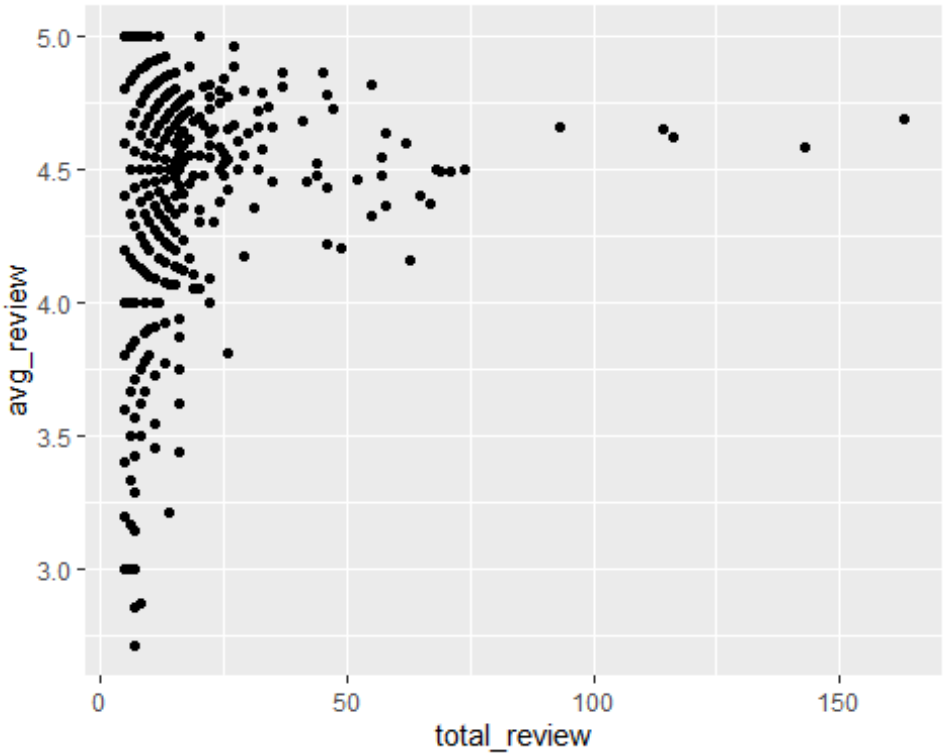
1.3 Overview of the dataset

Before we start text processing, we need to investigate overall structure of the dataset.

```
# Check Descriptive Statistic
# review %>%
#   select(- helpful)%>%
#   Hmisc::describe()

# See distrubution of review score
review%>%
  ggplot(aes(overall))+
    geom_histogram()+
    geom_vline(aes(xintercept = mean(overall)))
```




```
# Plot scatter plot to see pattern of number of review and review score
review%>%
  group_by(asin)%>%
  summarise(avg_review = mean(overall), total_review =
n_distinct(reviewerID))%>%
  arrange(desc(total_review))%>%
  ggplot()+
    geom_point(mapping = aes(x = total_review, y = avg_review))
```




```
# Check total review count, distinct reviewer, and product
review %>%
  summarise(count = n(),
            count_distinct_ID = n_distinct(reviewerID),
            count_distinct_ID_asin = n_distinct(reviewerID, asin) )

##   count count_distinct_ID count_distinct_ID_asin
## 1 10261              1429                  10261
```

1.4 Text cleaning

1. Removing any review which contains no text comment, since it will not give us any insight on the review

```
review <- review%>%
  filter(reviewText != "")
```

2. After texts have been inspected, some of reviews contain emoticon. "qdap" has been chosen to pre-process text by converting emoticon into proper words.

```
# Replace emoticons and internet slang
review$reviewText1 <- replace_emoticon(review$reviewText)
# review$reviewText1 <- replace_internet_slang(review$reviewText1)
```

3. We then move on to processing text by using regular expression (regex).
   - Firstly, we removes HTML tags and non-alphanumeric characters from the variable **review$reviewText1**, replaces them with a space and removes

consecutive whitespace characters by using `"[^[:alnum:][:punct:]]"` and `"\\s{2,}"`

- Then we remove all special characters, excluding apostrophe which will be processed in the next step, using regular expressions `"[^'[:alpha:][:space:]]"` and replacing them with a space
- **`str_squish`** and **`tolower`** function are then used to removes extra spaces and converting text to lowercase

```
# Replace html tags and non-alphanumeric characters
review$text1 <- str_remove_all(review$reviewText1, "<.*?>") %>%
  str_replace_all("[^[:alnum:][:punct:]]", " ") %>%
   str_replace_all("\\s{2,}", " ")

# Remove all special characters, excluding apostrophe, using regex
review$text1 <- review$text1 %>%
  gsub("[^'[:alpha:][:space:]]", " ",.)

# remove extra spaces
review$text1 <- str_squish(review$text1)

# lowercase
review$text1 <- tolower(review$text1)
```

4. Since there is no single primary key in this dataset, row numbers are assigned to each review as its unique `review_id`

```
# create review_id represent the unique identifier of each review
review <- review%>%
  #select(- c(helpful,unixReviewTime,reviewTime))%>%
  mutate(review_id = row_number())
```

1.5 Tokenize, Stop words and Lemmatization

The regular expression `"[[:alnum:]]+(?:[-'][[:alnum:]]+)*"` has been used to matches words or phrases that consist of alphanumeric characters (letters or digits) and optional hyphens or apostrophes.

```r
# Define a regular expression pattern to match hyphenated words
hyphen_pattern <- "[[:alnum:]]+(?:[-'][[:alnum:]]+)*"

# Define a custom tokenization function using predefined pattern
custom_tokenize <- function(x) {
  str_extract_all(x, hyphen_pattern)
}

# Check number of term and unique terms
review%>%
  unnest_tokens(text1,output=word_token,token=custom_tokenize)%>%
  summarise(count = n(),
            count_distinct_words = n_distinct(word_token))

##    count count_distinct_words
## 1 939294                19628
```

For stop words, there are 3 common lexicons that can be used to identify stop words. However, we do not want to remove too many words, as certain stop words such as “good” and “interesting” are important for conducting topic modelling and sentiment analysis. As a result, we set the criteria that stop words are only valid, if it’s existing in 2 out of 3 lexicons.

Once stop words has been defined, we proceed to lemmetize then removing stop wards respectively. As a result, the number of tokens has been reduced by half and ready to be used for further analysis.

```r
# Decide to use stop words that only in 2 out of 3 lexicons
stop_words_use <- stop_words%>%
  group_by(word)%>%
  summarise(count = n())%>%
  filter(count > 0)

# Define custom list of words to be removed

custom_remove <-
as.data.frame(as.matrix(c("guitar","string","pedal","buy","stick","amp","pick
","tongue", "cable", "mic", "tuner", "strap","sound","price","play","time")))


# Remove stop words, lemmetized and save data in token level
token <- review%>%
  unnest_tokens(text1,output=word_token,token=custom_tokenize)%>%
  mutate(word_lemma = lemmatize_words(word_token)) %>%
  anti_join(stop_words_use,by=c("word_lemma"="word"))%>%
  anti_join(custom_remove,by=c("word_lemma"="V1"))%>%
  unnest(word_lemma)


# Check number of term and unique terms post-processing
token %>%
  summarise(count = n(),
            count_distinct_words = n_distinct(word_lemma))

## # A tibble: 1 × 2
##    count count_distinct_words
##    <int>                <int>
## 1 272262                14399
```

2 Bag-of-words analysis

2.1 Calculate TFIDF

Firstly, we summarise frequency for each document, in this case each review, and calculate TF-IDF, which is a document weighted frequency.

```r
# Create frequency for each review
y_counts <- token %>%
  count(overall,review_id, word_lemma, sort = TRUE) %>%
  ungroup() %>%
  rename(count=n) %>%
  arrange(desc(count))

# Calculate TF-IDF
y_tfidf <- y_counts %>%
bind_tf_idf(review_id, word_lemma, count)

# Checking result
head(y_tfidf)
```

```
## # A tibble: 6 × 7
##   overall review_id word_lemma count     tf   idf tf_idf
##     <dbl>     <int> <chr>      <int>  <dbl> <dbl>  <dbl>
## 1       4      6384 light         39 0.0385  4.93 0.190
## 2       5      1885 sm            32 0.113   4.20 0.477
## 3       3      9349 driver        28 0.203   3.90 0.791
## 4       5       195 sm            23 0.0816  5.18 0.423
## 5       5      3613 record        23 0.0241  3.69 0.0890
## 6       5     10178 fret          23 0.0601  4.51 0.271
```

2.2 Create DTM

Once we have count and TF-IDF figure, we can proceed to create DTM. For this project, TF-IDF is chosen as a metric for creating DTM to help make important words more relevant as well as deprioritise rare words.

Once the initial DTM has been created, there are 14,399 terms in the document which mostly are not important (Sparsity ~100%). Hence, we proceed to remove non-significant term by using `removeSparseTerms` function, which set sparsity threshold to be no more than 98%. As a result, 228 terms remain in the final DTM.

```
# Create DTM
dtm <- y_tfidf %>%
  select(-overall)%>%
  cast_dtm(review_id,word_lemma,count)
dtm
```

```
## <<DocumentTermMatrix (documents: 10232, terms: 14399)>>
## Non-/sparse entries: 225068/147105500
## Sparsity           : 100%
## Maximal term length: 55
## Weighting          : term frequency (tf)
```

```
# Set sparsity threshold
spr = 0.98

# Store DTM with sparse term removed separately
dtm_sp <- removeSparseTerms(dtm,spr)
dtm_sp
```

```
## <<DocumentTermMatrix (documents: 10232, terms: 226)>>
## Non-/sparse entries: 90359/2222073
## Sparsity           : 96%
## Maximal term length: 11
## Weighting          : term frequency (tf)
```

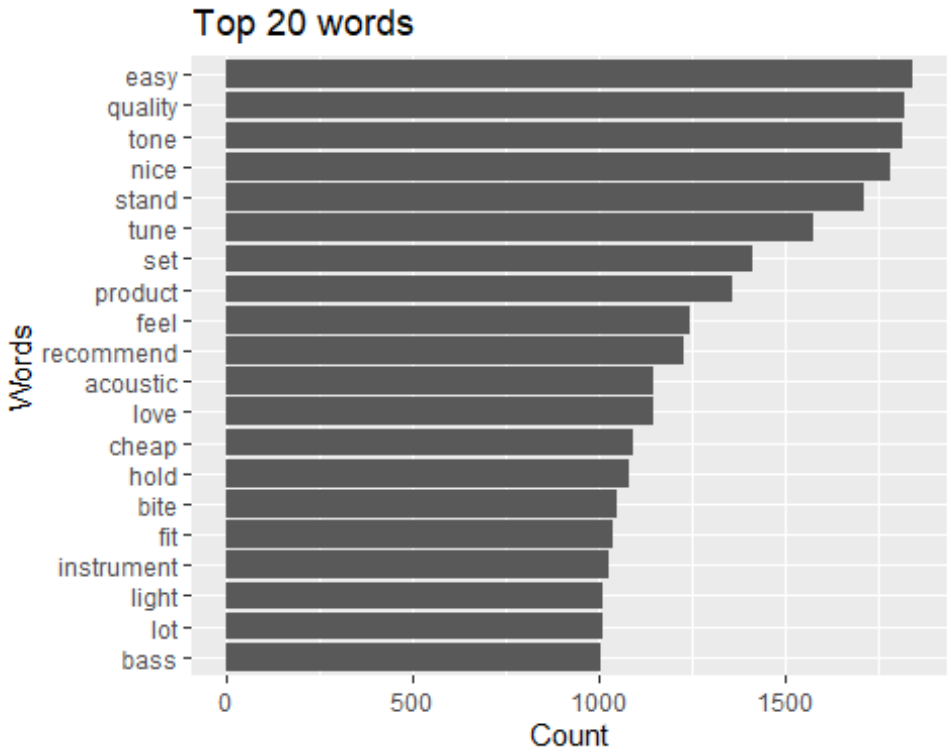
```
# Extract review id which does not contains any important term
remove_doc <-
as.numeric(rownames(as.matrix(dtm_sp)[which(rowSums(as.matrix(dtm_sp)) == 0),
]))

# Create new DTM without 0 term Docs
dtm_clean <- y_tfidf%>%
  select(-overall)%>%
  filter(!review_id %in% remove_doc )%>%
  cast_dtm(review_id,word_lemma,count)%>%
  removeSparseTerms(spr)
dtm_clean
```

```
## <<DocumentTermMatrix (documents: 10128, terms: 228)>>
## Non-/sparse entries: 90766/2218418
## Sparsity           : 96%
## Maximal term length: 11
## Weighting          : term frequency (tf)
```

2.3 Exploring word token

We then look at what is the popular terms across the corpus. The below charts show that some of the top words are “easy”, “quality”, “tone”, and “nice”. As a result of previous data cleaning, these words could be useful as representation for topics and sentiment.

```r
# Bar chart for top 20 words
y_tfidf%>%
  group_by(word_lemma) %>%
  summarise(freq = sum(count)) %>%
  arrange(desc(freq))%>%
  head(20) %>%
  ggplot(.,aes(y=reorder(word_lemma,freq),x=freq))+
    geom_bar(stat='identity')+
    labs(x = "Count", y = "Words", title = "Top 20 words")
```

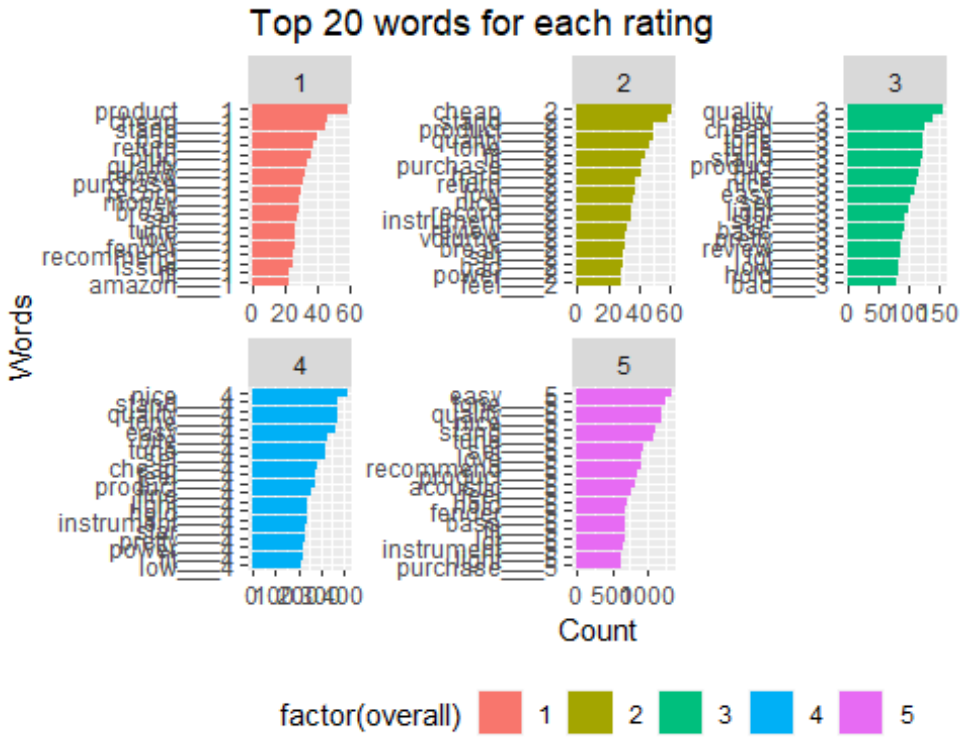


Next, we are drilling down into bag-of-words for each rating categories to observe whether there is any trends and common words for each group.

- For lower rating group (1-2 stars), it seems that negative words such as “cheap”, “bad”, “low” are dominant as well as certain action words that related to bad product, such as “return” and “screw”
- For 3 stars and above, positive words, such as “good”, “easy”, “nice” as well as mentioning about “quality” are dominant, which could be suggesting that customer satisfied with the product

With this result, we can confirm that there is variability of words among each rating groups, which is essential for further analysis by Topic Modelling and Sentiment Analysis.

```r
# Bar chart for top 20 words for each rating group
y_tfidf%>%
  group_by(word_lemma,overall) %>%
  summarise(freq = sum(count)) %>%
  arrange(desc(freq))%>%
  group_by(overall)%>%
  slice(1:20) %>%
  ggplot()+
    geom_bar(stat = 'identity', aes(x = freq, y =
reorder_within(word_lemma,freq,overall), fill = factor(overall)))+
    facet_wrap(~factor(overall), scale = "free")+
    labs(x = "Count", y = "Words", title = "Top 20 words for each rating")+
    theme(legend.position = "bottom")
```



```
# Word clouds for top 20 words for each rating group
for (n in 1:5){
set.seed(1234)

wc <- token%>%
  group_by(word_lemma,overall) %>%
  summarise(freq = n()) %>%
  arrange(desc(freq))%>%
  group_by(overall)%>%
  filter(overall == n)

wordcloud(words = wc$word_lemma, freq = wc$freq, min.freq = 10,
          max.words=200, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2"))
}
```

Word cloud – 1-star rating

Word cloud - 2-star rating

Word cloud - 3-star rating

Word cloud - 4-star rating

Word cloud - 5-star rating

3 Sentiment Analysis

Sentiment score calculation has been performed based on 2 approaches, text-based and token-based, then we will compare the result.

3.1 Text-based sentiment

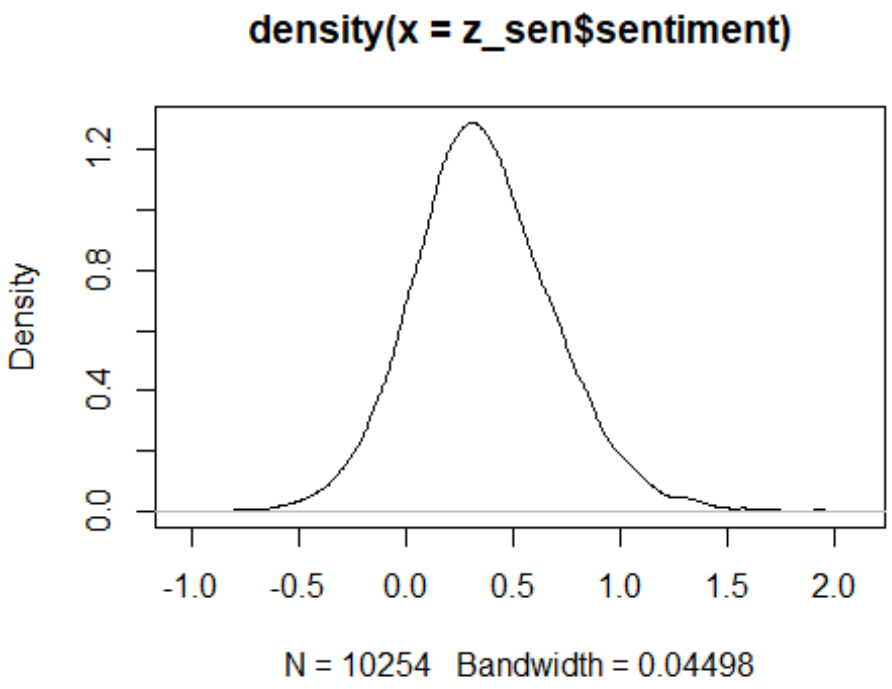
Firstly, we perform a text-based sentiment score calculation by using `sentiment_by` function for each sentence, which can be extracted by using `get_sentence`.

```
# Calculate sentiment by sentence for each review
z = sentiment_by(get_sentences(review$text1))
# highlight(z) # To check sentiment for each review

# Save the result
z_sen <- review
z_sen$sentiment <- z$ave_sentiment
z_sen <- z_sen%>%
  select(overall,sentiment)
```

After sentiment score has been obtained, we then explore and analyse the distribution of sentiment score for each rating groups.

- Pair-wise correlation matrix show that there are minor correlations between rating score and sentiment score, which confirmed that we can explore the relationship between two variables later
- The distribution of sentiment score has been plotted in the density chart which reveal that sentiment score is normally distributed with the mean above zero. This indicates that, generally, reviews are mostly positive which can be explained by concentration of good reviews (4-5 stars) as mentioned in the first section
- In term of distribution among rating groups, there is a pattern that average sentiment (represented by vertical line) is higher when star rating is high, and vice versa.

```
# Check correlation
rcorr(as.matrix(z_sen))

##           overall sentiment
## overall       1.0       0.3
## sentiment     0.3       1.0
##
## n= 10254
##
##
## P
##           overall sentiment
## overall            0
## sentiment  0

# Check distribution of sentiment
plot(density(z_sen$sentiment))
```



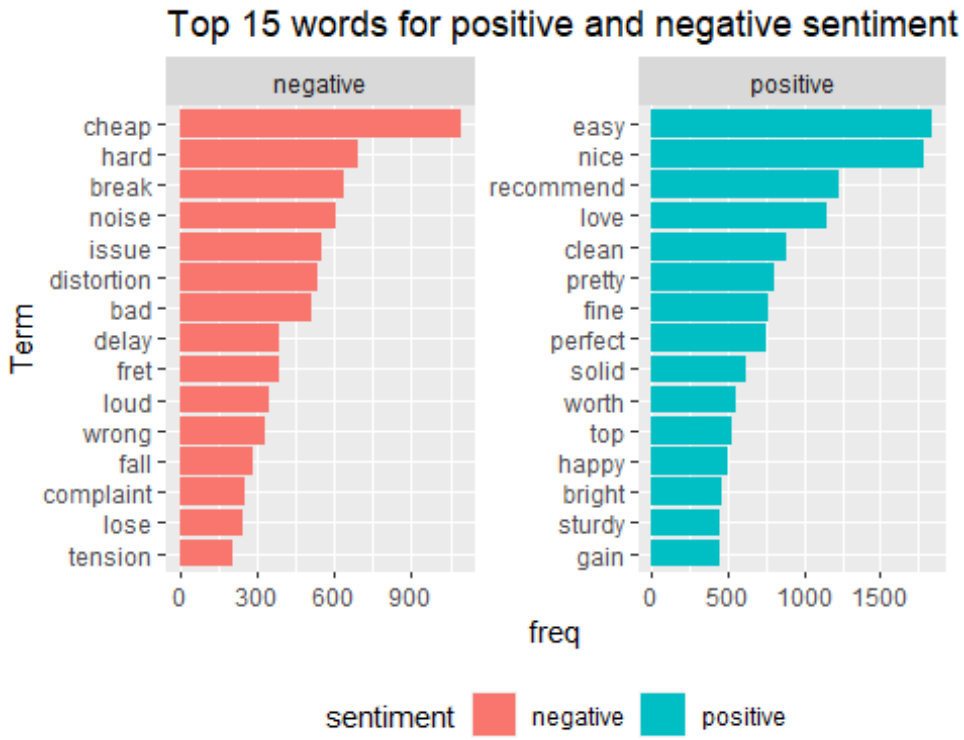
```r
# Calculate average sentiment for each rating
avg_sen <- z_sen%>%
  group_by(overall)%>%
  summarise(avg = mean(sentiment))

# Check trends

z_sen%>%
  mutate(overall = as.factor(overall))%>%
  ggplot()+
  geom_histogram( aes(sentiment, color = overall, fill = overall), bins =
100)+
  facet_grid(overall~.)+
  # scale_y_log10()+
  geom_vline(data = avg_sen, aes(xintercept = avg))+
  labs(title = "Sentiment breakdown by Overall rating")
```

3.2 Token-based sentiment

Bing dictionary was chosen as it gives a nice general positive or negative tag for each word. We transformed sentiment into value with 1 represented “positive”, 0 represented “neutral” and -1 represented “negative”.

```r
# Load bing dic
bing_dictionary <- tidytext::get_sentiments("bing")

# Edit bing dic to have numeric value
bing_dictionary_v <- bing_dictionary%>%mutate(value = ifelse(sentiment ==
'positive',1,-1))

# Load afinn
afinn_dictionary <- tidytext::get_sentiments("afinn")

# Pick dictionary for using in the sentiment analysis
use_dict <- bing_dictionary_v
```

```r
# Attached sentiment for each token
token_sen <- token%>%
  left_join( use_dict, by = c("word_lemma" = "word"))

# Observed top word for each sentiment types
token_sen %>%
  group_by(word_lemma,sentiment)%>%
  filter(!is.na(sentiment))%>%
  summarise(freq = n())%>%
  arrange(desc(freq))%>%
  group_by(sentiment)%>%
  slice(1:15)%>%
  ggplot()+
    geom_bar(stat = 'identity', aes(x = freq, y = reorder(word_lemma,freq),
fill = sentiment))+
    facet_wrap(~sentiment, scale = "free")+
    labs(y = "Term", title = "Top 15 words for positive and negative
sentiment (Bing)")+
    theme(legend.position = "bottom")
```

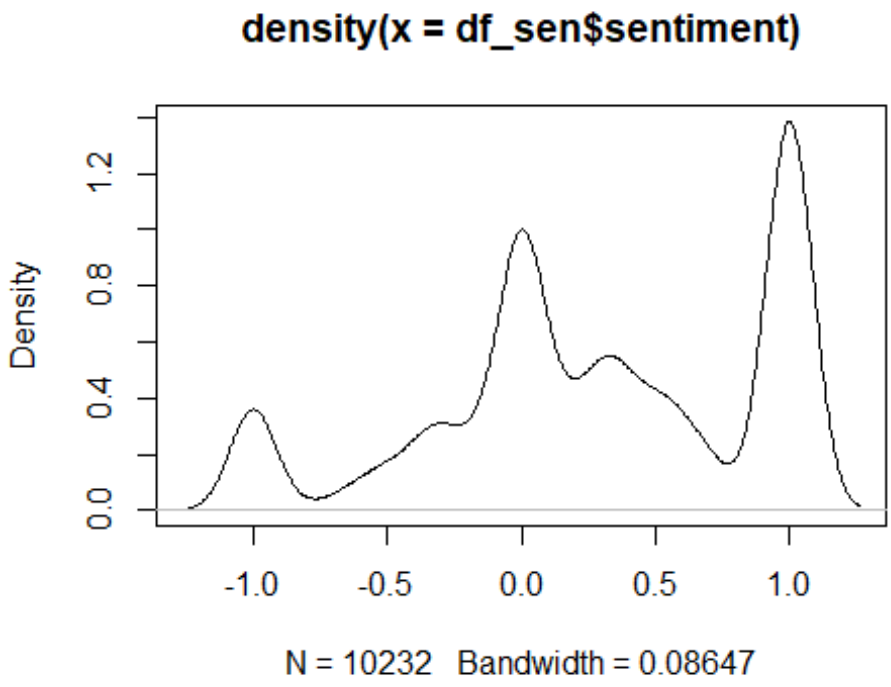


To represent sentiment on the review level, each review is then summarising by averaging sentiment of token that belong to that review, excluding any token with neutral sentiment. This method is chosen to avoid bias arising from review's length and represented average sentiment for each review.

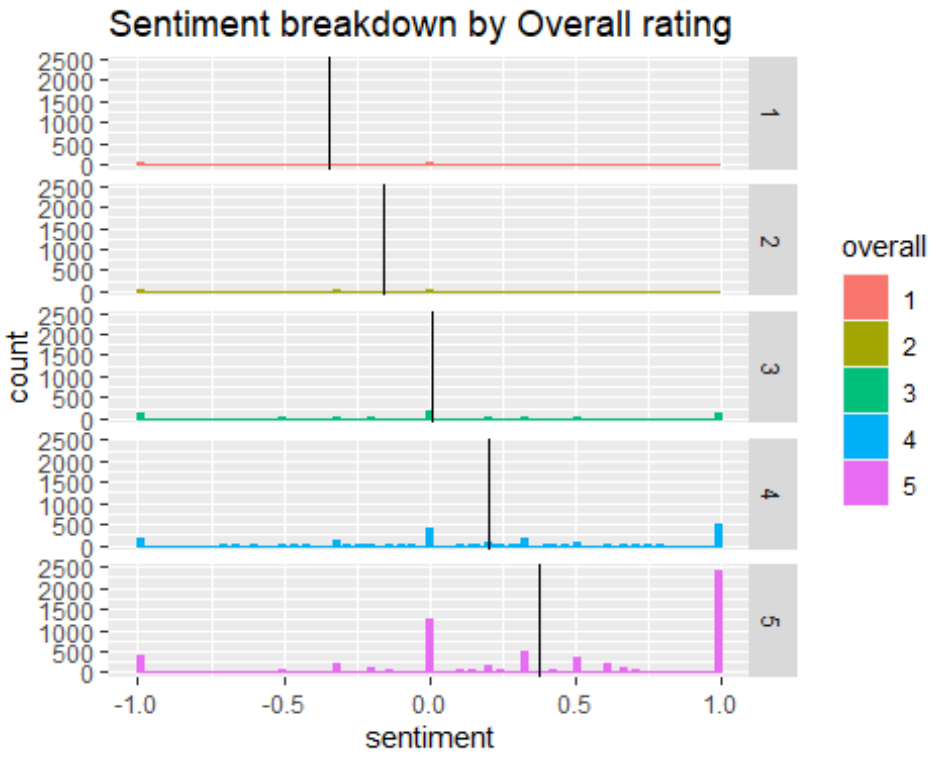
```
# Calculate average sentiment for each review (excluding neutral words)
df_sen <- token_sen%>%
  group_by(review_id,overall)%>%
  summarise(sentiment = round(mean(value, na.rm=TRUE),3))%>%
  dplyr::mutate(sentiment = replace_na(sentiment,0))%>%
  ungroup()%>%
  select(-review_id)
```

Below chart illustrated the distribution of review's sentiment. The distribution is biasing towards positive sentiment, which corresponded to the imbalance of positive words as illustrated in previous chart. The density plot exhibited a 3 peaks structure at -1, 0, and +1, which is due to nature of average where value range will be in between -1 and +1 and concentrated in these 3 values.

```r
# Check distribution of sentiment
plot(density(df_sen$sentiment))
```




```r
# Calculate average sentiment for each rating
avg_sen_t <- df_sen%>%
  group_by(overall)%>%
  summarise(avg = mean(sentiment))

# Check trends

df_sen%>%
  mutate(overall = as.factor(overall))%>%
  ggplot()+
  geom_histogram( aes(sentiment, color = overall, fill = overall), bins =
100)+
  facet_grid(overall~.)+
  # scale_y_log10()+
  geom_vline(data = avg_sen_t, aes(xintercept = avg))+
  labs(title = "Sentiment breakdown by Overall rating")
```



3.3 Regression analysis

We further investigate the relationship between sentiment and overall rating by running a linear regression analysis. The result show that there is a statistically significant relationship between sentiment score and overall rating, in which a plus one sentiment is likely to have 0.54 higher overall score rating.

While the test is statistically significant, the model is not performing well in term of explanatory power as adjusted R-squared is just 9% for both methods. This may cause by the nature of text sentiment, as majority of words are having neutral sentiment, causing less variation to be explained by the model.

```r
# Linear regression for text-based

z_sen_clean <- z_sen%>%
  filter(sentiment != 0)

m.lm.sen <- lm(overall~.,data = z_sen_clean)
summary(m.lm.sen)

##
## Call:
## lm(formula = overall ~ ., data = z_sen_clean)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.8801 -0.3711  0.3179  0.5632  1.5499
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.19197    0.01254  334.28   <2e-16 ***
## sentiment    0.81614    0.02550   32.01   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8525 on 10172 degrees of freedom
## Multiple R-squared:  0.09149,    Adjusted R-squared:  0.0914
## F-statistic:  1024 on 1 and 10172 DF,  p-value: < 2.2e-16
```

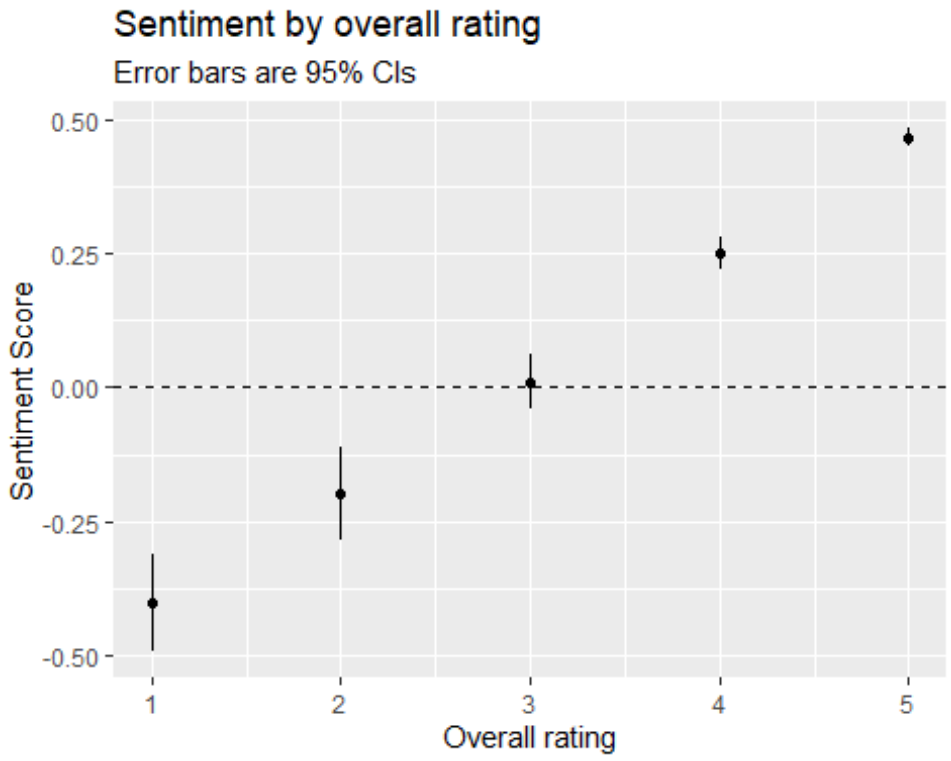
```r
# Linear regression for token-based

df_sen_clean <- df_sen%>%
  filter(sentiment != 0)

m.lm.sen.t <- lm(overall~.,data = df_sen_clean)
summary(m.lm.sen.t)
```

```
##
## Call:
## lm(formula = overall ~ ., data = df_sen_clean)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.7569 -0.3842  0.2431  0.5166  1.0631
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.34687    0.01067  407.50   <2e-16 ***
## sentiment    0.41001    0.01427   28.72   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8563 on 8308 degrees of freedom
## Multiple R-squared:  0.09033,    Adjusted R-squared:  0.09022
## F-statistic:   825 on 1 and 8308 DF,  p-value: < 2.2e-16
```

To further investigate the relationship, we estimated mean sentiment score of each rating group and 95% CI. The result from token-based sentiment shows that the average sentiment score for rating five group is 0.42 95%CI(0.41,0.43).

Furthermore, we could observe that estimate sentiment means for three-star rating is approximately zero, which indicates that low rating (1-2 star) is associated with negative sentiment while high rating (4-5) is associated with positive sentiment.

```
# Use ANOVA to check if the difference in sentiment of each rating group are
different
summary(aov(overall~.,data = z_sen_clean))

##                Df Sum Sq Mean Sq F value Pr(>F)
## sentiment       1    744   744.4    1024 <2e-16 ***
## Residuals   10172   7392     0.7
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

summary(aov(overall~.,data = df_sen_clean))

##                Df Sum Sq Mean Sq F value Pr(>F)
## sentiment       1    605   605.0     825 <2e-16 ***
## Residuals    8308   6092     0.7
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# Estimate mean with emmeans package for token-based
library(emmeans)
emm.sen.t <- lm(sentiment~factor(overall), data = df_sen_clean)
(m.lm.sen.t.emm <- emmeans(emm.sen.t, ~overall))

##  overall  emmean      SE   df lower.CL upper.CL
##        1 -0.4010 0.04603 8305  -0.4913  -0.3108
##        2 -0.1973 0.04450 8305  -0.2846  -0.1101
##        3  0.0109 0.02569 8305  -0.0395   0.0612
##        4  0.2505 0.01529 8305   0.2206   0.2805
##        5  0.4657 0.00836 8305   0.4493   0.4821
##
## Confidence level used: 0.95

# Estimate mean with emmeans package for sentence-based
emm.sen <- lm(sentiment~factor(overall), data = z_sen_clean)
(m.lm.sen.emm <- emmeans(emm.sen, ~overall))

##  overall  emmean      SE    df lower.CL upper.CL
##        1 -0.0270 0.02149 10169  -0.0691   0.0151
##        2  0.0848 0.02018 10169   0.0452   0.1243
##        3  0.1732 0.01141 10169   0.1508   0.1955
##        4  0.3253 0.00694 10169   0.3117   0.3389
##        5  0.4182 0.00381 10169   0.4107   0.4256
##
## Confidence level used: 0.95

# Plot emmeans result - Token-based method
ggplot(summary(m.lm.sen.t.emm), aes(x = overall, y = emmean, ymin=lower.CL,
ymax=upper.CL)) +
    geom_point() + geom_linerange() + geom_hline(yintercept = 0, lty = 2)+
    labs(y="Sentiment Score", x="Overall rating", subtitle="Error bars are
95% CIs", title="Sentiment by overall rating")
```



4 Topic modelling

For this last section, the aim is to perform Topic modelling on the review data to observe different themes or aspect that exist in reviews of Musical Instruments.

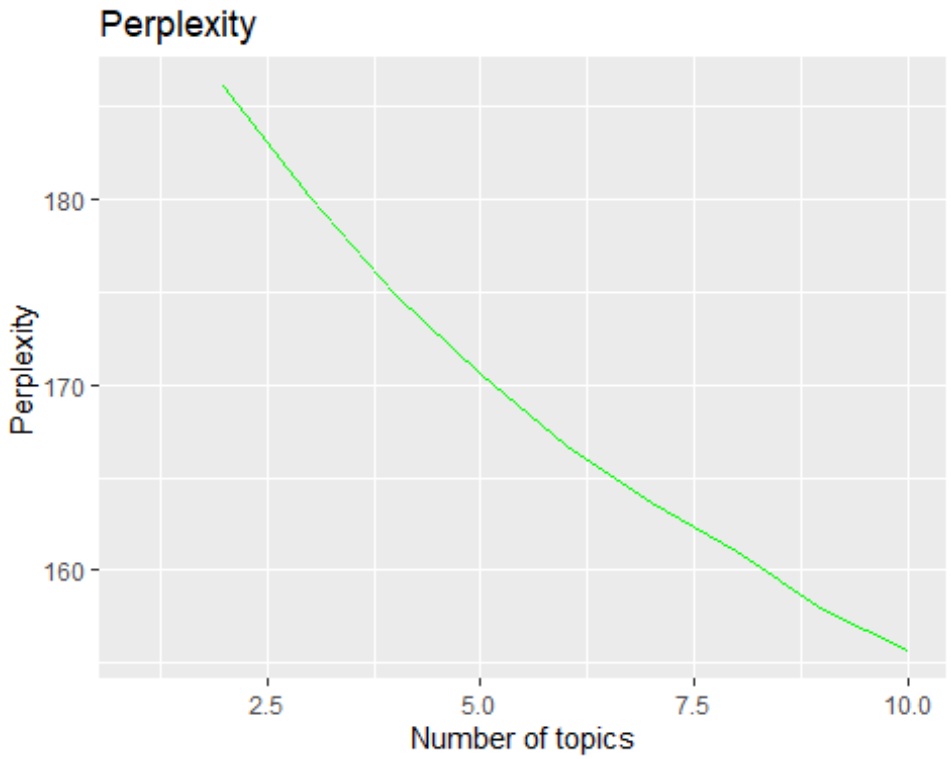
4.1 LDA and Topics

To determine number of topics, we first calculate perplexity score for 2 to 10 topics, as we don't want too few and too many topics since it will be difficult to analyse.

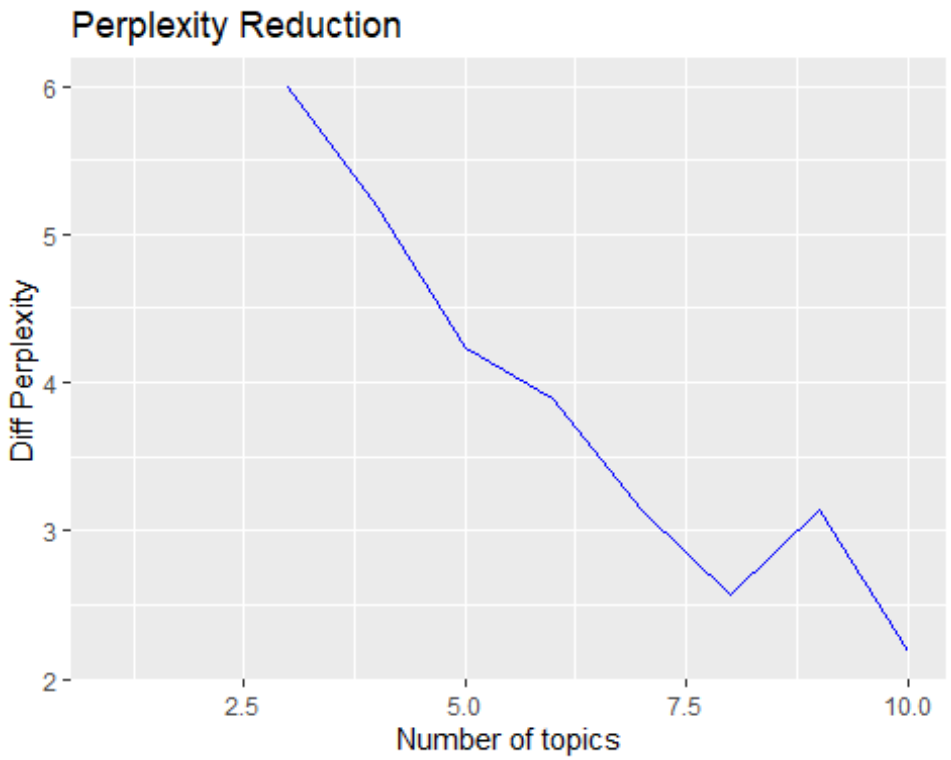
```
# Loop to calculate perplexity for 2-10 topics
set.seed(12345)
k_topics <- c(2:10)
perplexity_df <- data.frame(perp_value=numeric())
for (i in k_topics){
  fitted <- LDA(dtm_clean, k = i, method = "Gibbs")
  perplexity_df[i,1] <- perplexity(fitted,dtm_clean)
}

# Calculate difference in perplexity score
perplexity_df$diff <- c(NA, -diff(perplexity_df$perp_value))
```

Below chart illustrate perplexity score for each number of topics solution as well as difference from the previous solution.5 topics is when perplexity reduction become slower, hence, we start this analysis by exploring 5 topics.

```
ggplot(data=perplexity_df, aes(x= as.numeric(row.names(perplexity_df))))+
  labs(y="Perplexity",x="Number of topics") + ggtitle("Perplexity")+
  geom_line(aes(y=perp_value), colour="green")
```



```
ggplot(data=perplexity_df, aes(x= as.numeric(row.names(perplexity_df))))+
  labs(y="Diff Perplexity",x="Number of topics") + ggtitle("Perplexity
Reduction")+
  geom_line(aes(y=diff), colour="blue")
```



Set-up LDA model

We first setting up LDA with 5 topics solution, then store beta into a tidy format.

```
# Setting up LDA to generate topic and associated beta
set.seed(12345)
my_topic_model <- LDA(dtm_clean,k = 5,method = "Gibbs")

# Store beta information
topics <- tidy(my_topic_model, matrix = "beta")
```

Inspect Topics and Probabilities

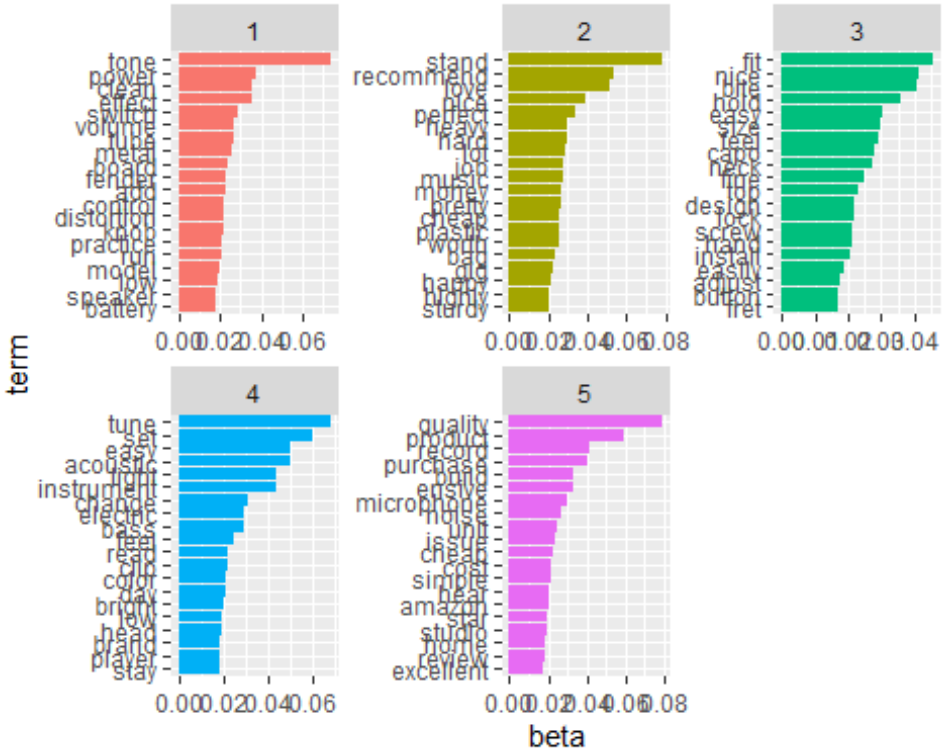
Inspect the topics and their probabilities for each document using the posterior function from the topicmodels package.

```r
# Generate posterior probability for each review
topics_prob <- posterior(my_topic_model)$topics

# Store all terms and top 5 terms
terms <- terms(my_topic_model, 5)
terms_all <- terms(my_topic_model)

# Tag each topic by its associated terms
colnames(topics_prob) <- apply(terms, 2, paste, collapse = ",")
head(topics_prob)
```

```
##       tone,power,clean,effect,switch stand,recommend,love,nice,perfect
## 6384                       0.1878788                         0.13333333
## 1885                       0.2452830                         0.15094340
## 9349                       0.2614679                         0.10091743
## 195                        0.2258065                         0.15053763
## 3613                       0.1637931                         0.09913793
## 10178                      0.1063830                         0.12765957
##       fit,nice,bite,hold,easy tune,set,easy,acoustic,light
## 6384                0.1151515                    0.4484848
## 1885                0.1383648                    0.1257862
## 9349                0.1330275                    0.1238532
## 195                 0.2258065                    0.1827957
## 3613                0.1120690                    0.1465517
## 10178               0.3563830                    0.3244681
##       quality,product,record,purchase,build
## 6384                              0.11515152
## 1885                              0.33962264
## 9349                              0.38073394
## 195                               0.21505376
## 3613                              0.47844828
## 10178                             0.08510638
```

Plot chart based on beta of term on each topic

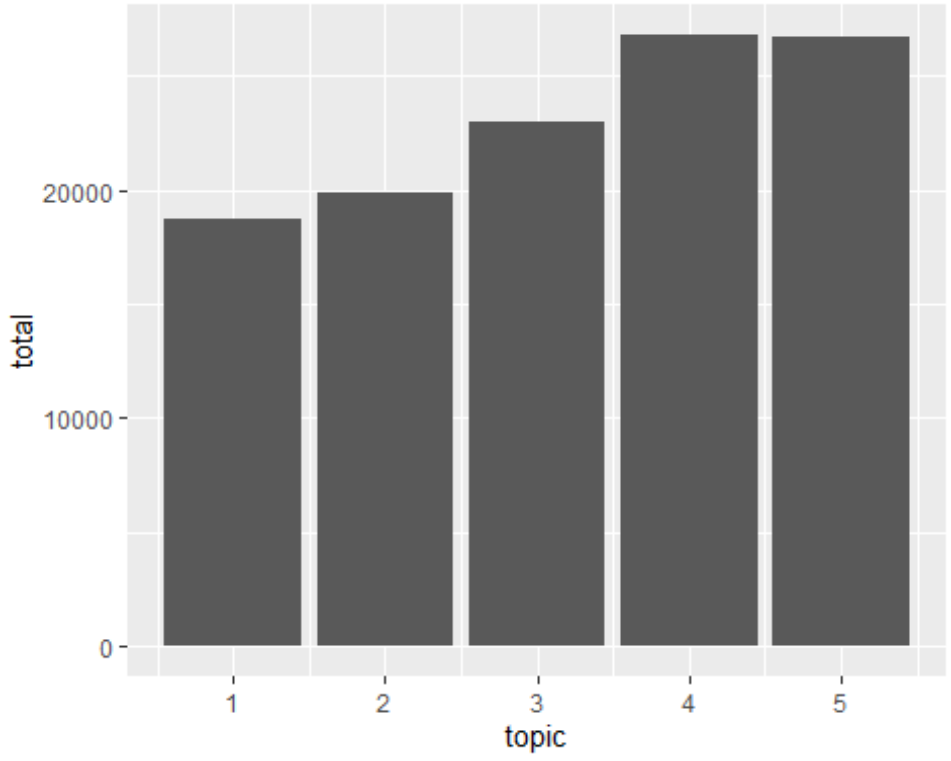
```
# Store top 20 terms for each topic
top_terms <- topics %>%
  group_by(topic) %>%
  slice_max(beta, n = 20) %>%
  ungroup() %>%
  arrange(topic, desc(beta))

# Plot bar chart to explore top terms for each topic
top_terms %>%
  mutate(term = reorder_within(term, beta, topic)) %>%
  ggplot(aes(beta, term, fill = factor(topic))) +
    geom_col(show.legend = FALSE) +
    facet_wrap(~ topic, scales = "free") +
    scale_y_reordered()
```



Assign each term to a topic

Before we finalise the topic naming, we can use `augment` to help pick the topic that has the highest probability to associate with each term. This needed to be done since output of LDA is not exclusive.

```
#Assign each term to each topics
assignments <- augment(my_topic_model, data = dtm_clean)

# Store mapping between term and topics
map_top_terms <- assignments%>%
  group_by(term)%>%
  summarise(topic = max(.topic), freq = sum(count))

# Check distribution of the topics
map_top_terms%>%
  group_by(topic)%>%
  summarise(total = sum(freq))%>%
  ggplot()+
    geom_col(mapping = aes(x = topic, y = total))
```



Name the Topics

Firstly, we extract top terms of each topic to help with the naming by using GPT to summarise potential topics.

```
k = max(top_terms$topic)
for (n in 1:k){
topic_name <- paste0("t",n)
result <- assign(topic_name,top_terms%>%
  filter(topic == n)%>%
  select(term))

my_string <- paste0(result$term,collapse = ",")

print(my_string)
}
```

```
## [1]
"tone,power,clean,effect,switch,volume,tube,metal,board,fender,add,control,di
stortion,knob,practice,run,model,low,speaker,battery"
## [1]
"stand,recommend,love,nice,perfect,heavy,hard,lot,job,music,money,pretty,chea
p,plastic,worth,bag,gig,happy,highly,sturdy"
## [1]
"fit,nice,bite,hold,easy,size,feel,capo,neck,fine,top,design,lock,screw,hand,
install,easily,adjust,button,fret"
## [1]
"tune,set,easy,acoustic,light,instrument,change,electric,bass,feel,read,clip,
color,day,bright,low,head,brand,player,stay"
## [1]
"quality,product,record,purchase,build,ensive,microphone,noise,unit,issue,che
ap,cost,simple,hear,amazon,star,studio,home,review,excellent"
```

In addition, we create word clouds based on frequency of words to help adjust the final naming of the topics.

```
# Word clouds
set.seed(1234)

for (n in 1:k){
wc <- map_top_terms%>%filter(topic == n)
wordcloud(words = wc$term, freq = wc$freq, min.freq = 1,
          max.words=200, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2"))
}
```

Word cloud – Topic 1

Word cloud – Topic 2

Word cloud – Topic 3

Word cloud – Topic 4

Word cloud – Topic 5

Based on abovementioned illustration, we can briefly name each topic as per below aspect.

1. Features and specifications
2. User Experience/Opinions
3. Instrument Fittings/Accessories
4. Instrument Tuning/Setup
5. Product Quality and Performance

```
# Name each aspect based on GPT and Word clouds
map_top_terms <- map_top_terms%>%
  mutate(aspect = case_when(
    topic == 1 ~ "Features and specifications",
    topic == 2 ~ "User Experience/Opinions",
    topic == 3 ~ "Instrument Fittings/Accessories",
    topic == 4 ~ "Instrument Tuning/Setup",
    topic == 5 ~ "Product Quality and Performance",
  ))
```

As illustrated by result below, customers who are unhappy with the product (rating 1-2 stars) tends to mention about “Product Quality and Performance” more than other topics.

On the other hand, better reviews (3-5 stars) seem to have more even distribution among topics with relatively higher concentration of “Features and specifications” as well as “User Experience/Opinions”.

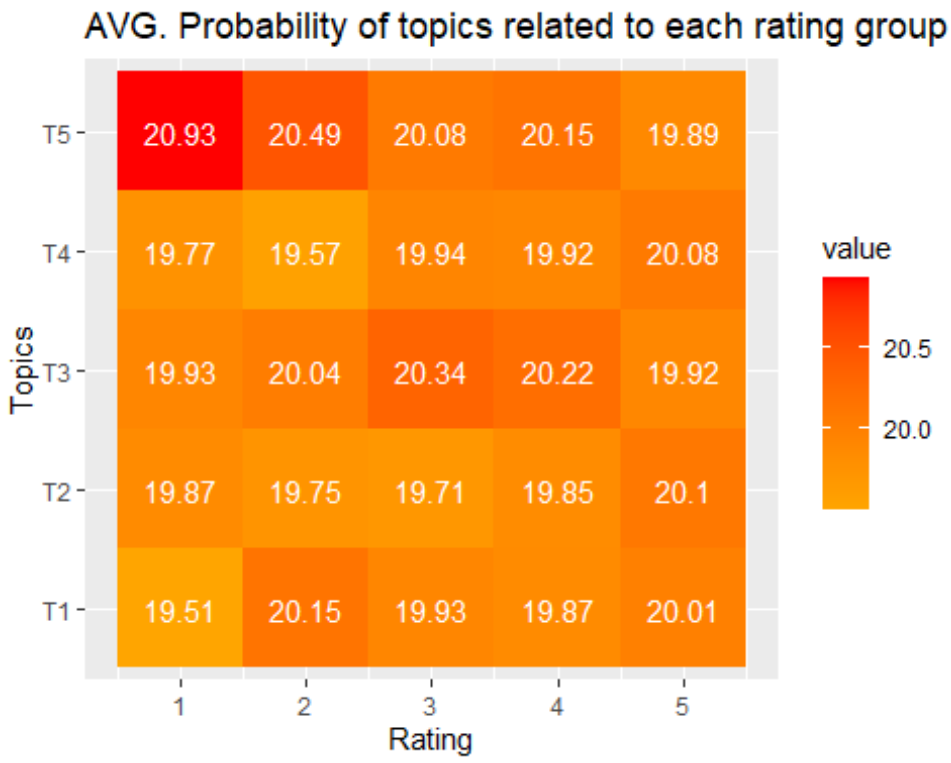
```r
# Store LDA result in tidy format
X <- tidy(my_topic_model, matrix = "gamma")

# Summarise data into per document and topic level
X1 <- X %>%
  group_by(document,topic) %>%
  dplyr::summarise(Prob = sum(gamma)) %>%
  ungroup()
X1$topic = paste0("T",X1$topic)
head(X1)
```

```
## # A tibble: 6 × 3
##   document topic  Prob
##   <chr>    <chr> <dbl>
## 1 1        T1    0.208
## 2 1        T2    0.189
## 3 1        T3    0.208
## 4 1        T4    0.189
## 5 1        T5    0.208
## 6 10       T1    0.204
```

```r
# Transform date from long to wide format
X11 <- X1 %>%
  pivot_wider(names_from = topic, values_from = Prob)
X11 <- as.data.frame(X11)
X11$document = as.integer(X11$document)

# Joing with review data to obtain rating stars
X12 <- review%>%
  select(review_id,overall)%>%
  inner_join(X11,by=c("review_id" = "document"))%>%
  select(-review_id)
head(X12)
```

```
##   overall        T1        T2        T3        T4        T5
## 1       5 0.2075472 0.1886792 0.2075472 0.1886792 0.2075472
## 2       5 0.1929825 0.1929825 0.1754386 0.1754386 0.2631579
## 3       5 0.2000000 0.2000000 0.2333333 0.1833333 0.1833333
## 4       5 0.1818182 0.1818182 0.2727273 0.1818182 0.1818182
## 5       5 0.1886792 0.1886792 0.1886792 0.1886792 0.2452830
## 6       5 0.1964286 0.2142857 0.1785714 0.2142857 0.1964286
```

```r
# Aggregate to rating group level by averaging probability
X12 %>%
  group_by(overall)%>%
  summarise(across(everything(), ~ round(mean(.x, na.rm = TRUE),4)*100))%>%
  as.data.frame(as.matrix())%>%
  column_to_rownames(., var = 'overall')%>%
  as.matrix()%>%
  reshape::melt()%>%
  ggplot(., aes(X1, X2))+
```


AVG. Probability of topics related to each rating group
Topics
T5
T4
T3
T2
T1
20.93
20.49
20.08
20.15
19.89
19.77
19.57
19.94
19.92
20.08
19.93
20.04
20.34
20.22
19.92
19.87
19.75
19.71
19.85
20.1
19.51
20.15
19.93
19.87
20.01
1
2
3
4
5
Rating
value
20.5
20.0

5 Appendix

5.1 Assignment questions

1500-word essay:

The goal of this assignment is to familiarize you with the complete process of extracting, refining, and delivering insights of business value deriving from unstructured data. This is an individual assignment where you will work alone in order to build a dataset to address a particular business problem.

The report should be written from the perspective of an analyst involving text mining methods in constructing a well written piece of work. This should be both academic as well as practical and consider possible application scenarios where text mining can be used in a prescriptive analytics manner.

For each category the reviews are aggregated as follows:

You should download the 5-core data files for one category of your choice. The 5-core means that it only includes products that have at least five reviews. You are required to conduct the following analyses:

1. A thorough explanation of the text cleaning and normalization process that you have followed

2. A bag-of-words analysis of the review text such as what are the dominant words per star rating category?

3. Using polarity and sentiment demonstrate how text derived features connect with the review score. Justify your use of the appropriate sentiment lexicon. A regression model should be used to evaluate the predictability of product ratings against the variables obtained from the review text.

4. Using a topic model, you should conduct an analysis of the topics that become dominant for your selected category. Your analysis should focus on finding which topics become important for satisfied and dissatisfied customers as well as how particular characteristics of the products are more likely to influence a dominance of different themes in a customer review.