Text Analytics

Renuka Ramachandran

1. INTRODUCTION - ABOUT THE PROJECT

This project aims at analyzing Yelp review data set to predict the sentiment towards a business using verbatim review text. This was done as an academic exercise on natural language techniques for deriving insights from text.

Due to computational constraints the analysis was carried out using a sample of 50,000 reviews. The sentiment scores have been derived using the AFINN dictionary (http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010 (http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010)). The Naive Bayes technique has been used to carry out the modelling task.

2. CREATING AN APPROPRIATE ENVIRONMENT

Loading the required packages and dataset into the current work directory.

```
rm(list=ls())
library(tm)
library(SnowballC)
library(wordcloud)
library(RColorBrewer)
library(dplyr)
library(tidytext)
library(stringr)
library(reshape2)
library(ggplot2)
library(e1071)
library(caret)
library(glmnet)


reviews <- read.csv("yelpRestaurantReviewSample_50K.csv",header=TRUE, sep =';')
attach(reviews)
```

3. DATA EXPLORATION PHASE

```
dim(reviews)
```

```
## [1] 50000    18
```

```
sapply(reviews,class)
```

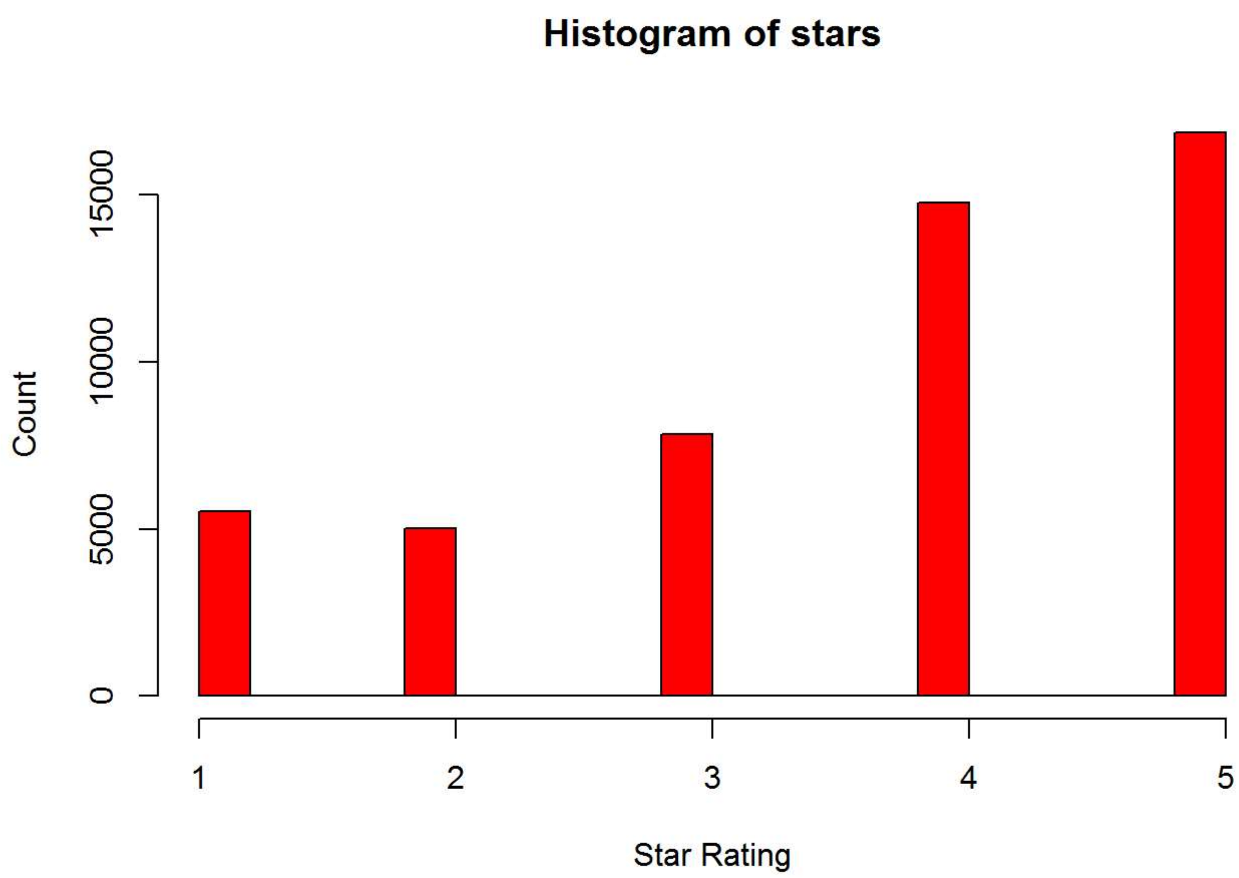
```
##    business_id          cool          date         funny     review_id
##       "factor"     "numeric"      "factor"     "numeric"      "factor"
##          stars          text          type        useful       user_id
##      "numeric"      "factor"      "factor"     "numeric"      "factor"
## categories.0. categories.1. categories.2.          name   postal_code
##       "factor"      "factor"      "factor"      "factor"      "factor"
##   review_count         state  businessType
##      "numeric"      "factor"      "factor"
```

```
names(reviews)
```

```
##  [1] "business_id"   "cool"          "date"          "funny"
##  [5] "review_id"     "stars"         "text"          "type"
##  [9] "useful"        "user_id"       "categories.0." "categories.1."
## [13] "categories.2." "name"          "postal_code"   "review_count"
## [17] "state"         "businessType"
```

3.1. How are star ratings distributed? How will you use the star ratings to obtain a label indicating ‘positive’ or ‘negative’?

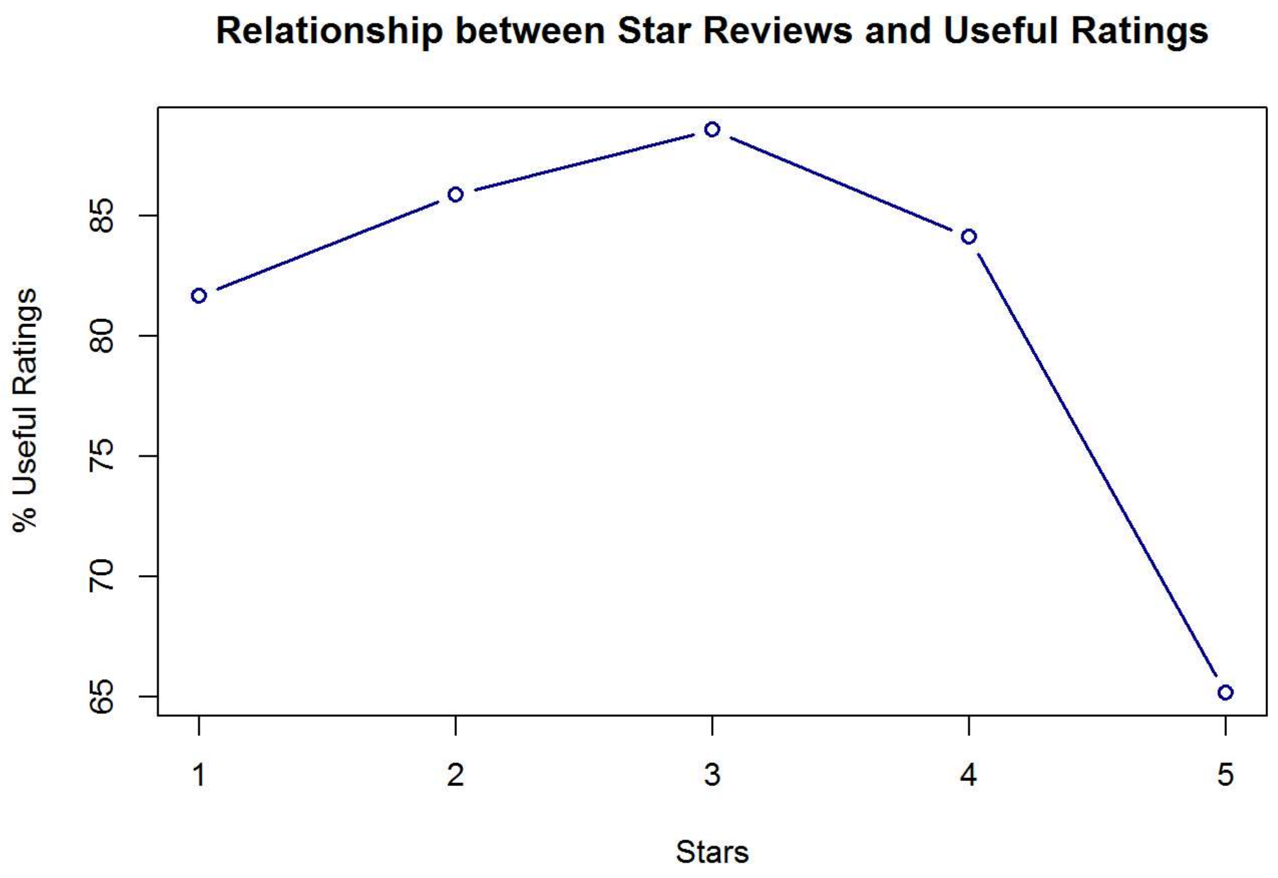
```
hist(stars, xlab = "Star Rating", ylab="Count",col = "red")
```



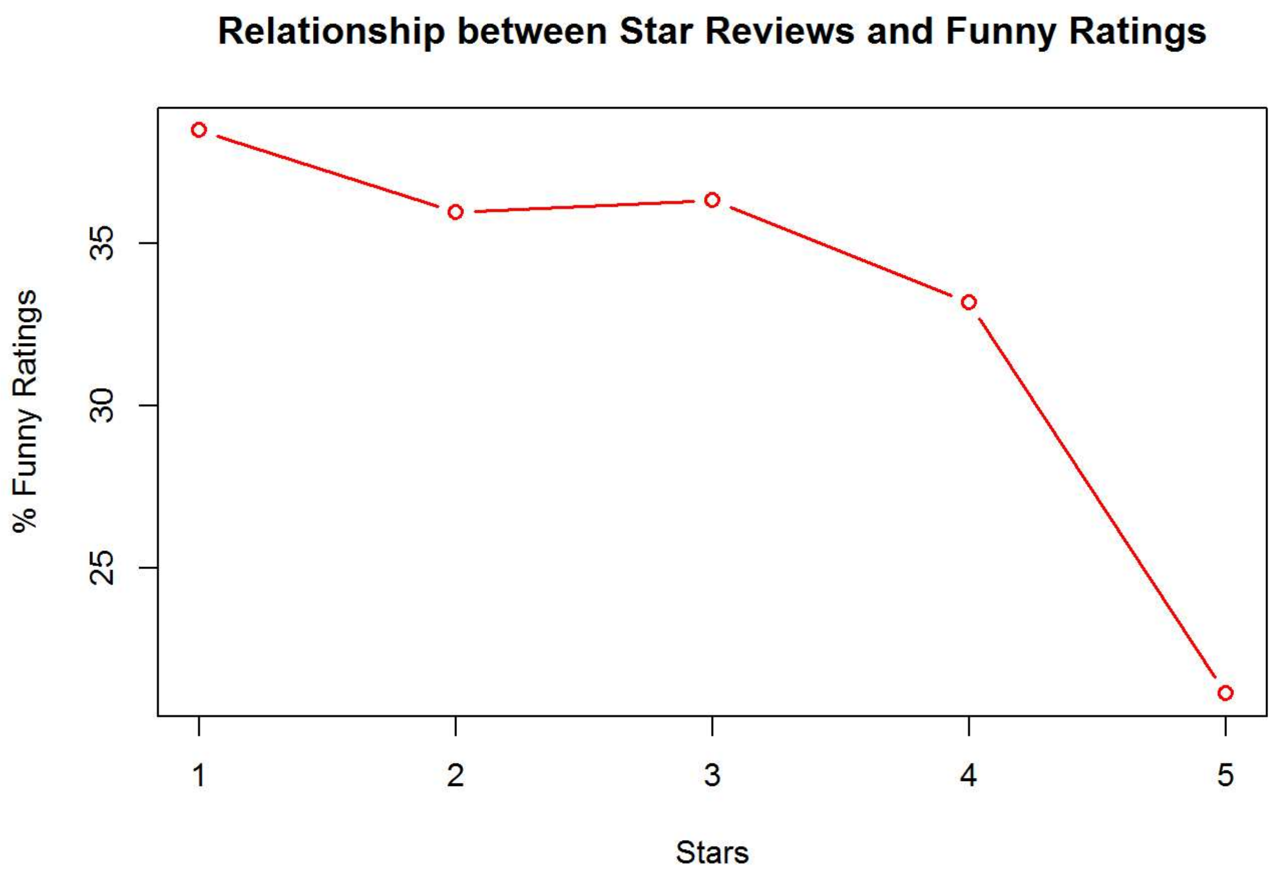
INFERENCE: Based on the distribution: - Ratings 1,2,3 can be considered negative - RatingS 4 and 5 can be considered positive

3.2. Does star ratings have any relation to ‘funny’, ‘cool’, ‘useful’?

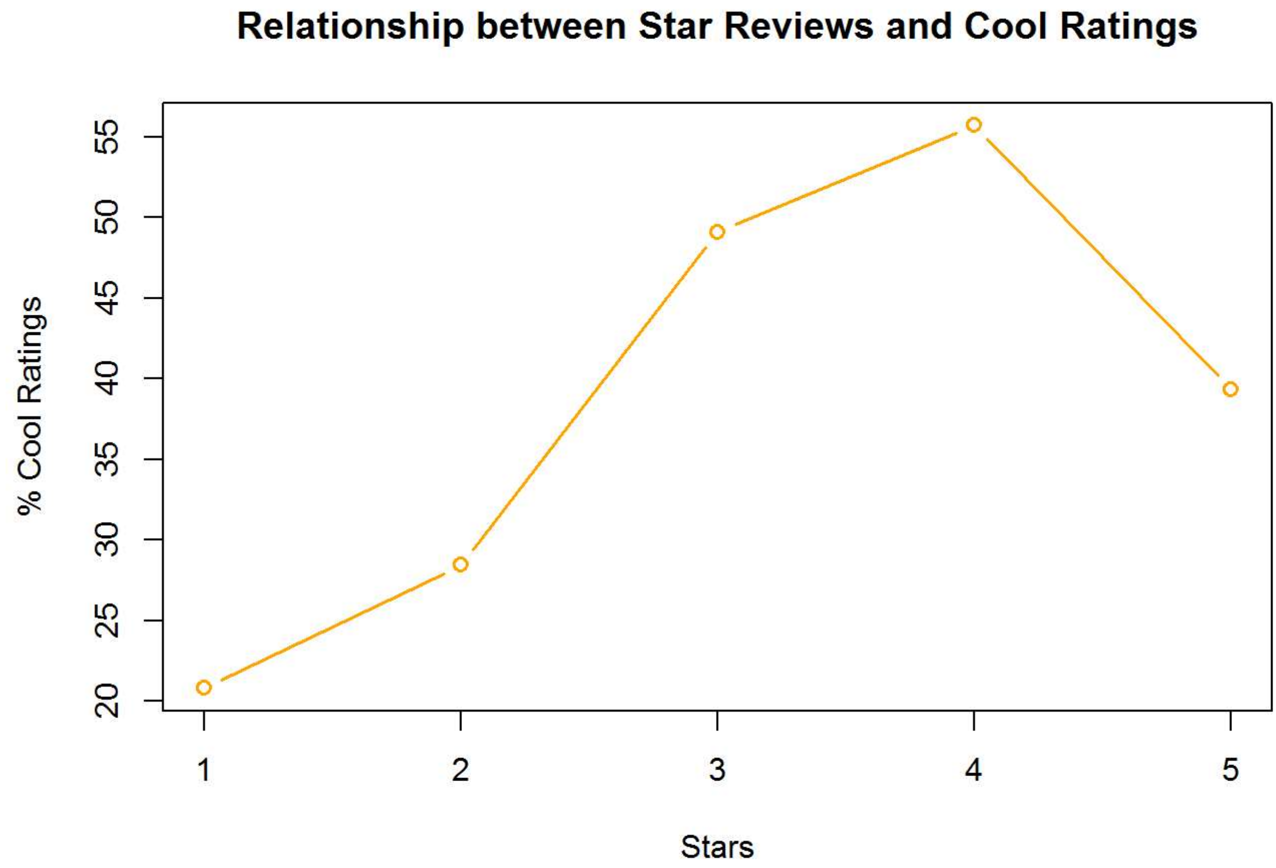
```
rating <- aggregate(stars,list(stars),length)
names(rating)<-c("Stars","Total")

useful <- aggregate(useful,list(stars),sum)
names(useful)<-c("Stars","Useful Reviews")

funny <- aggregate(funny,list(stars),sum)
names(funny)<-c("Stars","Funny Reviews")

cool <- aggregate(cool,list(stars),sum)
names(cool)<-c("Stars","Cool Reviews")

review_rating_type <- merge(rating,useful,by= "Stars")
review_rating_type <- merge(review_rating_type,funny,by= "Stars")
review_rating_type <- merge(review_rating_type,cool,by= "Stars")

#Percentage
#par(mfrow=c(1,3))
plot(review_rating_type$Stars,(review_rating_type$`Useful Reviews`/review_rating_type$Total)
*100, type = "b", xlab = "Stars", ylab ="% Useful Ratings", main = "Relationship between Star
 Reviews and Useful Ratings", col ="darkblue",lwd=1.5)
```




```
plot(review_rating_type$Stars,(review_rating_type$`Funny Reviews`/review_rating_type$Total)*1
00, type = "b", xlab = "Stars", ylab ="% Funny Ratings", main = "Relationship between Star Re
views and Funny Ratings", col ="red",lwd=1.5)
```




```
plot(review_rating_type$Stars,(review_rating_type$`Cool Reviews`/review_rating_type$Total)*10
0, type = "b", xlab = "Stars", ylab ="% Cool Ratings", main = "Relationship between Star Revi
ews and Cool Ratings", col ="orange",lwd=1.5)
```



INFERENCE: Based on the distribution: 1. Reviews that are rated low are considered to be more 'useful' 2. Lower ratings seem to have been voted as 'funny' compared to reviews with higher star rating 3. Rating 3 and 4 have been voted as 'cool'

4. MAKING DATA SUITABLE FOR MODEL BUILDING

Steps Involved:

- We remove weird symbols in-order to facilitate text mining operations
- The data-set is stored in a ***corpus***
- The characters in the data-set is ***converted to lower-case***
- The punctuations, numerical values and stop-words (words such as the, this, that etc occurs several times in a data-set which adds no value in our text mining purposes and hence these words are to be removed in-order to fetch better results) are removed.

4.1 Building a corpus and cleaning the data

```
review_Corpus <- Corpus(VectorSource(reviews$text))
review_Corpus<-sample(review_Corpus, 50000)

#Cleaning reviews text

revs<-tm_map(review_Corpus,tolower)

#removing punctuation marks
revs<-tm_map(revs,removePunctuation)

#removing numbers
revs<-tm_map(revs,removeNumbers)

#removing stop words
revs<-tm_map(revs,removeWords, stopwords("english"))

#removing whitespaces
revs<-tm_map(revs,stripWhitespace)
```

4.2 Building a Document Term Matrix

- Transforming the data-set into a document term matrix enables the data to be stored as : (rows, columns) = (document, term)
- The document term matrix is further converted into a matrix in-order to make computations pretty efficiently
- The number of terms and their corresponding frequencies shall be obtained by computing the column sum ansd its further sorted in descending order (based on frequencies)

```
dtm<-DocumentTermMatrix(revs)
show(dtm)
```

```
## <<DocumentTermMatrix (documents: 50000, terms: 111807)>>
## Non-/sparse entries: 2254247/5588095753
## Sparsity           : 100%
## Maximal term length: 129
## Weighting          : term frequency (tf)
```

4.3. Removing Sparse Terms

```
dtm <- removeSparseTerms(dtm, 0.99)
dtm.matrix <- as.matrix(dtm)
```

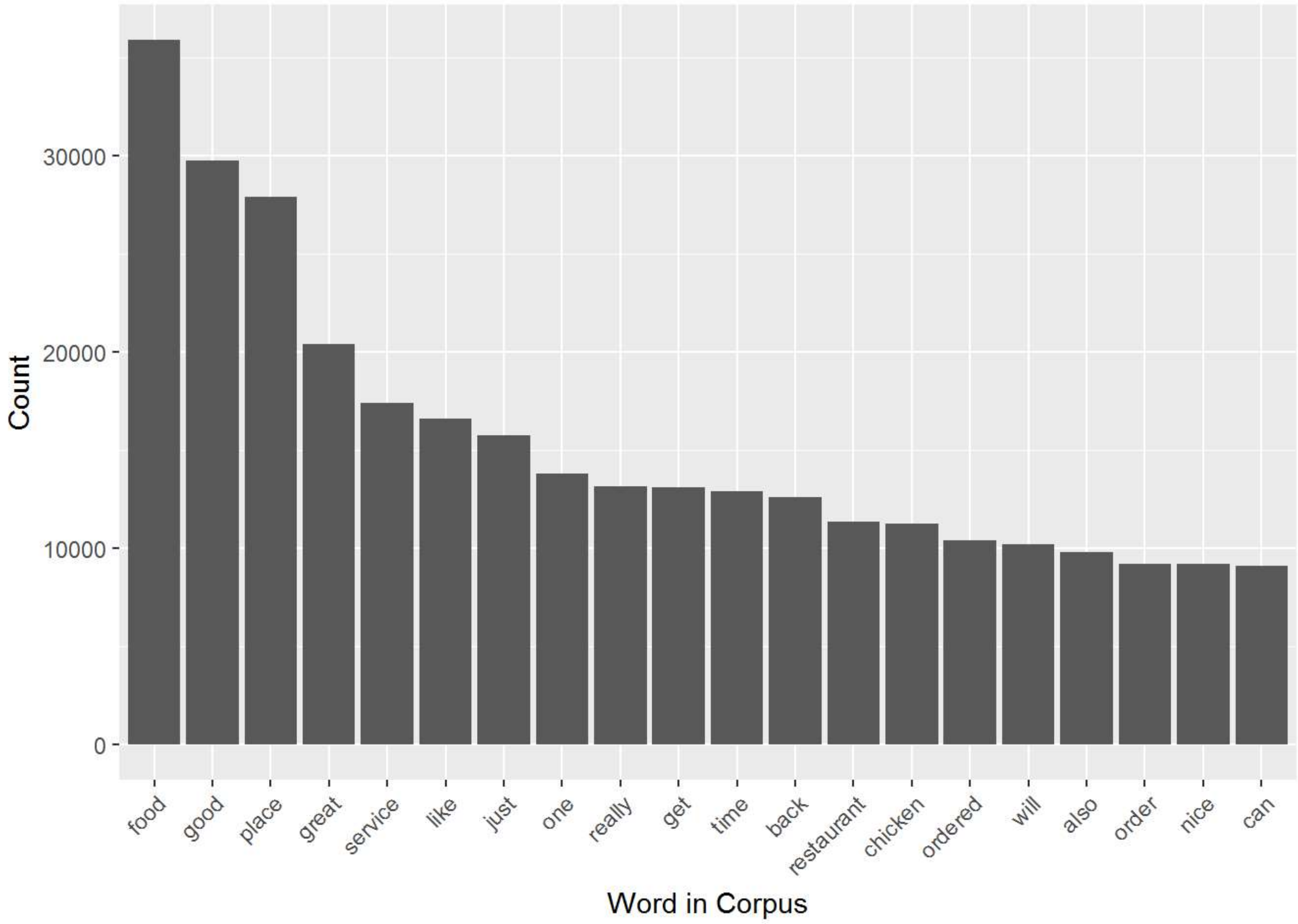
4.4. Converting DTM to Dataframe

```
dtm_df <- tidy(dtm)
reviews$document <- seq(1:50000)
reviews_df <- merge(subset(reviews, select=c("review_id","stars","document","business_id")),d
tm_df,by="document")
```

5. EXPLORATORY DATA ANALYSIS

5.1. Top 20 Words and their frequency

```
wordcount <- colSums(dtm.matrix)
top <- head(sort(wordcount, decreasing=TRUE), 20)

dfplot <- as.data.frame(melt(top))
dfplot$word <- dimnames(dfplot)[[1]]
dfplot$word <- factor(dfplot$word,
                      levels=dfplot$word[order(dfplot$value,
                                                decreasing=TRUE)])

fig <- ggplot(dfplot, aes(x=word, y=value)) + geom_bar(stat="identity")
fig <- fig + xlab("Word in Corpus")
fig <- fig + ylab("Count")
fig <- fig + theme(axis.text.x = element_text(angle = 45, hjust = 1))
print(fig)
```



5.2. Generating word cloud for top 100 frequent words

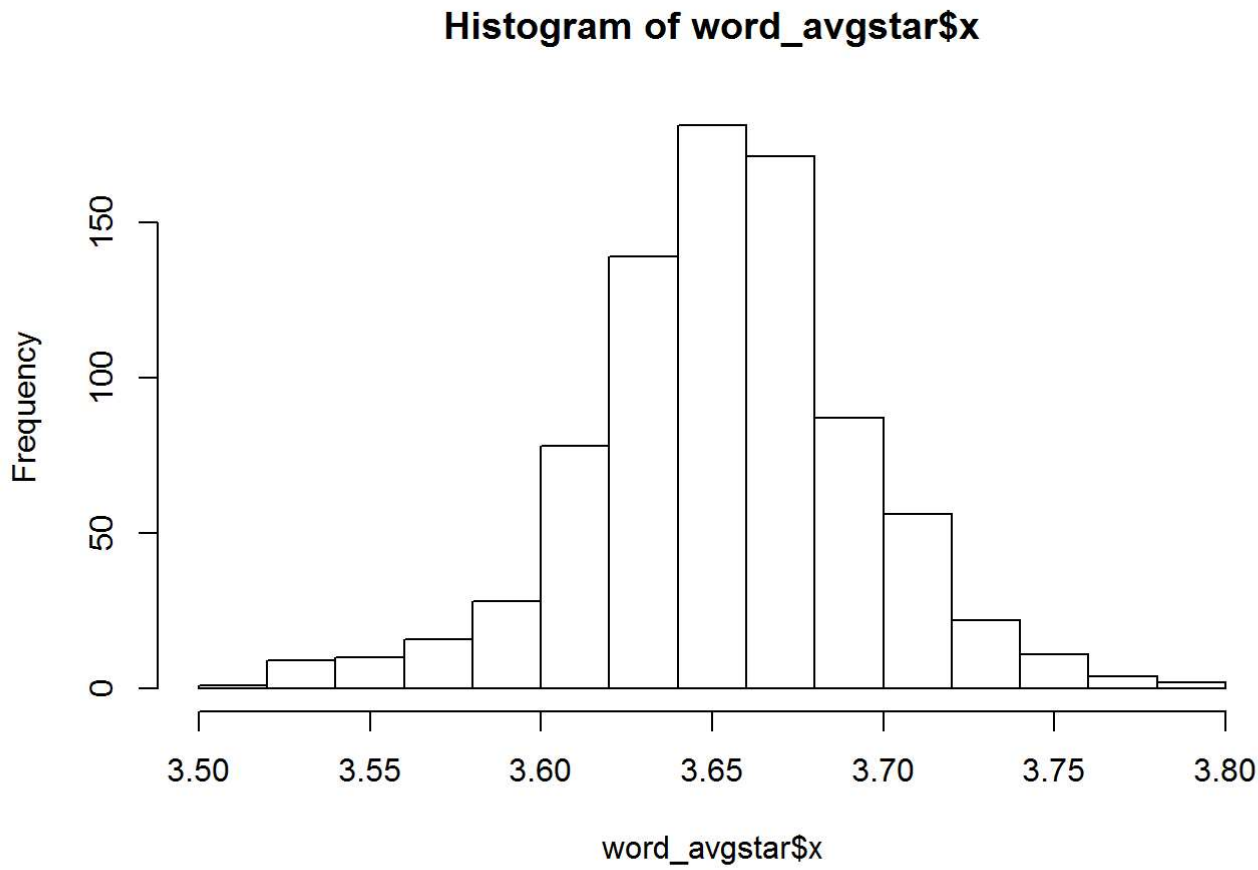
```
wordcloud(revs, max.words = 100, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2"))
```

5.3. Identifying Positive and Negative Terms

Identifying positive and negative words using star ratings as an indicator (average of ratings wherever the word occurs in the review)

Approach[1]

```
word_avgstar <- aggregate(reviews_df$stars,list(reviews_df$term),mean)
hist(word_avgstar$x)
```




```
word_avgstar <- word_avgstar %>%arrange(x)

#Negative Words
head(word_avgstar,10)
```

```
##        Group.1        x
## 1  outstanding 3.514241
## 2        curry 3.520468
## 3         rude 3.522409
## 4         game 3.529880
## 5         glad 3.530909
## 6       entire 3.533875
## 7         ribs 3.534921
## 8       mostly 3.535849
## 9      phoenix 3.535948
## 10         yum 3.536134
```

```
#Positive Words
tail(word_avgstar,10)
```

```
##       Group.1        x
## 806 employees 3.749562
## 807       fãr 3.755179
## 808      show 3.758261
## 809       mix 3.759599
## 810   excited 3.761006
## 811   takeout 3.761484
## 812    taking 3.763085
## 813      tell 3.779635
## 814     split 3.780822
## 815     spice 3.788091
```

OBSERVATION: This method of finding top positive and negative words may not be accurate since it does not consider important words, i.e. words that not only occur frequently within a document but also occurs across documents. For this we approach using another method. This is carried out below.

Approach[2] - Alternate Approach

```
review_words <- reviews_df %>%
  mutate_all(as.character)

review_words_counted <- review_words %>%
  count(review_id, business_id, stars, term) %>%
  ungroup()

#review_words_counted


word_summaries <- review_words_counted %>%
  group_by(term) %>%
  summarize(businesses = n_distinct(business_id),
            reviews = n(),
            uses = sum(n),
            average_stars = mean(as.numeric(stars))) %>%
  ungroup()
#word_summaries

#Words that are present in atleast 100 documents and in more than 5 businesses
word_summaries_filtered <- word_summaries %>%
  filter(reviews >= 100, businesses >= 5)
#word_summaries_filtered

#Positive words -  from the filtered list
word_summaries_filtered %>%
  arrange(desc(average_stars))
```

```
## # A tibble: 815 x 5
##         term businesses reviews  uses average_stars
##        <chr>      <int>   <int> <int>         <dbl>
##  1     spice        471     571   571      3.788091
##  2     split        430     511   511      3.780822
##  3      tell        901    1316  1316      3.779635
##  4    taking        546     726   726      3.763085
##  5   takeout        462     566   566      3.761484
##  6   excited        503     636   636      3.761006
##  7       mix        472     599   599      3.759599
##  8      show        458     575   575      3.758261
##  9       fãr        427     531   531      3.755179
## 10 employees        455     571   571      3.749562
## # ... with 805 more rows
```

```
#Negative words - from filtered list
word_summaries_filtered %>%
  arrange(average_stars)
```

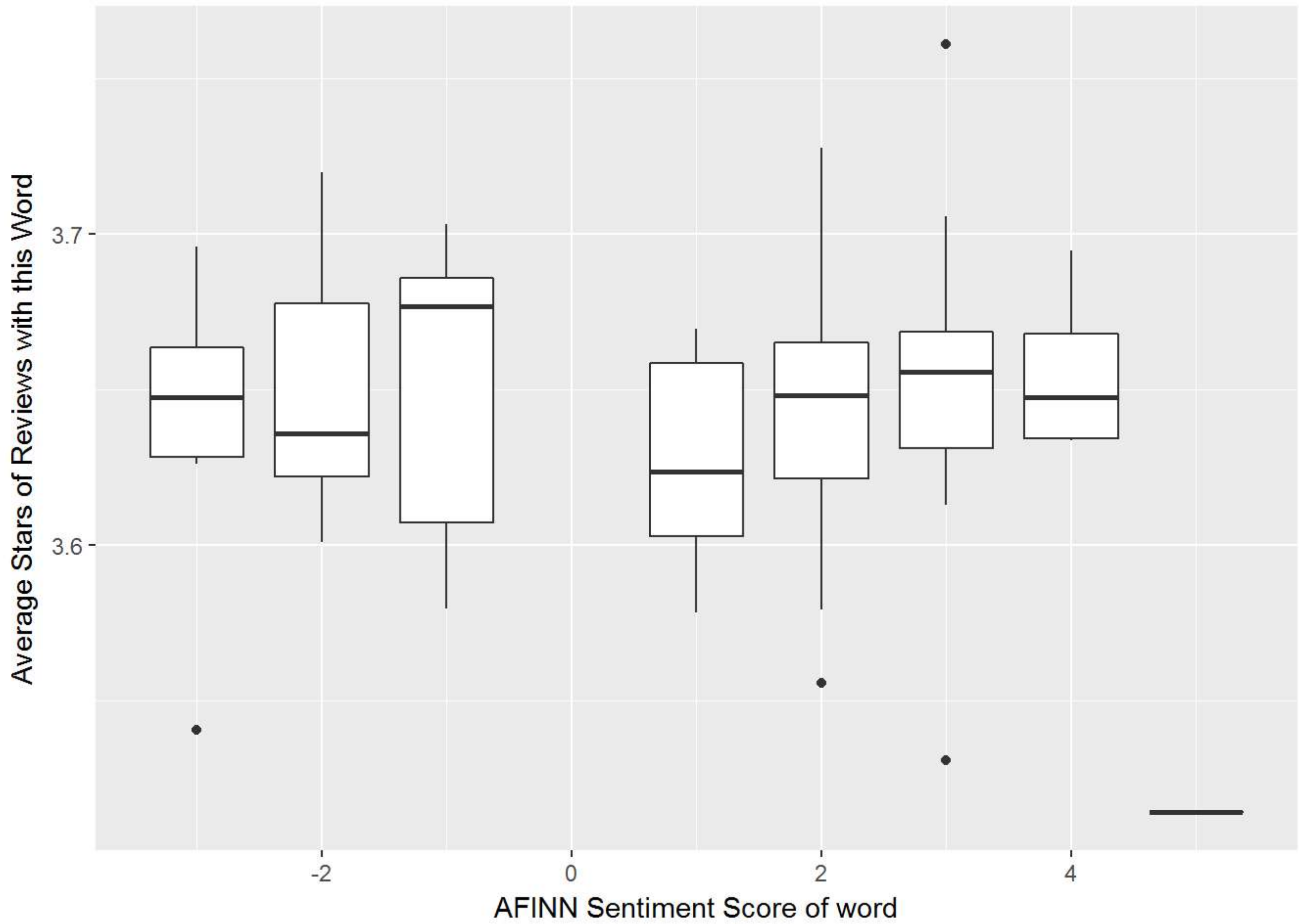
```
## # A tibble: 815 x 5
##           term businesses reviews  uses average_stars
##          <chr>      <int>   <int> <int>         <dbl>
##  1 outstanding        516     632   632      3.514241
##  2       curry        655     855   855      3.520468
##  3        rude        571     714   714      3.522409
##  4        game        407     502   502      3.529880
##  5        glad        634     825   825      3.530909
##  6      entire        564     738   738      3.533875
##  7        ribs        519     630   630      3.534921
##  8      mostly        431     530   530      3.535849
##  9     phoenix        494     612   612      3.535948
## 10         yum        474     595   595      3.536134
## # ... with 805 more rows
```

```
#Positive and Negative words as per AFINN
AFINN <- sentiments %>%
  filter(lexicon == "AFINN") %>%
  select(term = word, afinn_score = score)

words_afinn <- word_summaries_filtered %>%
  inner_join(AFINN)

words_afinn
```

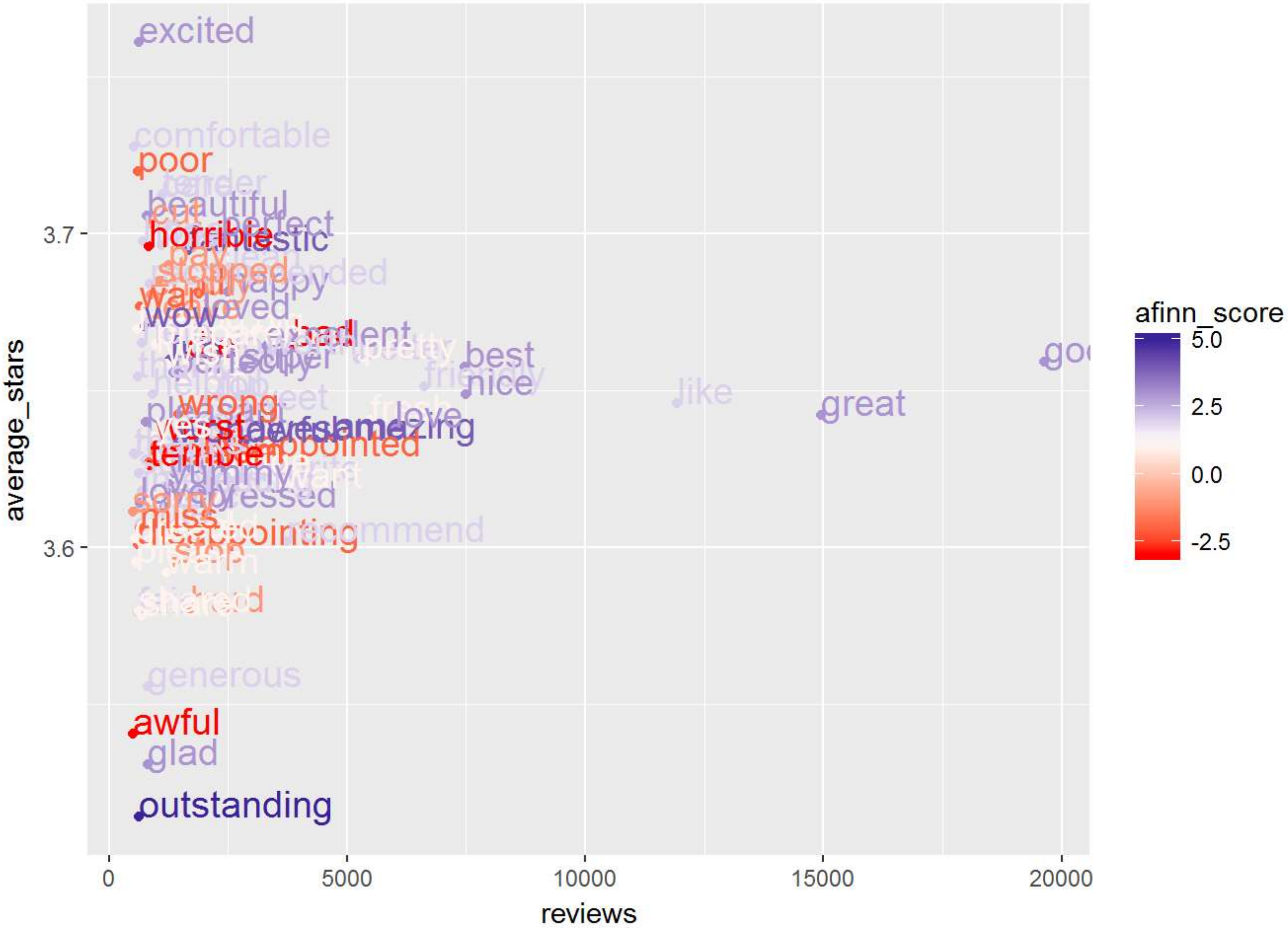
```
## # A tibble: 94 x 6
##         term businesses reviews  uses average_stars afinn_score
##        <chr>      <int>   <int> <int>         <dbl>       <int>
##  1   amazing       2326    4733  4733      3.635115           4
##  2   awesome       1621    2887  2887      3.634222           4
##  3     awful        417     505   505      3.540594          -3
##  4       bad       1974    3860  3860      3.664508          -3
##  5 beautiful        596     792   792      3.705808           3
##  6      best       2982    7471  7471      3.657743           3
##  7    better       2402    5238  5238      3.660176           2
##  8       big       1696    3048  3048      3.668635           1
##  9      care        772    1101  1101      3.712080           2
## 10    chance        506     656   656      3.612805           2
## # ... with 84 more rows
```

5.4 Visualizing the relationships and distribution

```
ggplot(words_afinn, aes(afinn_score, average_stars, group = afinn_score)) +
  geom_boxplot() +
  xlab("AFINN Sentiment Score of word") +
  ylab("Average Stars of Reviews with this Word")
```




```
mid<-mean(words_afinn$afinn_score)
ggplot(words_afinn, aes(x=reviews, y=average_stars,color=afinn_score)) +
  geom_point()+
  scale_color_gradient2(midpoint=mid, low="red",high="dark blue", space ="Lab" )+
  geom_text(aes(label=term ,hjust=0, vjust=0),size=5)
```



OBSERVATION - The graph illustrates different kind of words (positive or negative) that occur frequently in reviews rated low to high - It can be observed that the density of positive words begin to increase for reviews higher than rating of 3 stars.

5.5. Sentiment Score Analysis

Using the dictionary based positive and negative terms to predict sentiment (positive or negative based on star rating) of a restaurant. For (AFINN) dictionary, obtain an aggregated positiveScore and a negativeScore for each review; for the AFINN dictionary, an aggregate positivity score can be obtained for each review.

Based on the scatter plot created earlier average score of 3 has been considered as the threshold to classify reviews as positive and negative in the training set (since density of positive words begin to increase for reviews having rating higher than 3 stars). Reviews having sentiment score >= 3 is considered positive and reviews having sentiment score < 3 is considered negative.

```
reviews_sentiment <- review_words %>%
  inner_join(AFINN, by = "term")

#reviews_sentiment

#Average Sentiment score for each review using AFINN dictionary
reviews_score <- reviews_sentiment%>%
  group_by(review_id, stars) %>%
  summarize(sentiment = mean(afinn_score))

#reviews_score

reviews_score <- reviews_score %>%
  right_join(subset(reviews, select=c("review_id","text","document")), by = "review_id")


#Classifying review as positive and negative based on sentiment score
reviews_score <- mutate(reviews_score,
                        sentiment_cat = ifelse(sentiment >= 3,"Pos","Neg"))

#reviews_score
```

PREDICTIVE MODELLING

Predict review sentiment based on these aggregated sentiment scores and understand model performance

6.1 CREATING DATASETS

- Creating test and train corpus
- Creating test and train dataset
- 70:30 ratio

```
#Creating test and train corpus
revs.train <- revs[1:35000]
revs.test <- revs[35001:50000]

#Creating test and train dataset
df.train <- reviews_score[1:35000,]
df.test <- reviews_score[35001:50000,]
```

6.2 THE FREQUENT FIVE

- Keeping words that occur in atleast 5 reviews
- Use most frequent words (fivefreq) to build the train and test DTM

```
fivefreq <- findFreqTerms(dtm, 5)
length((fivefreq))
```

```
## [1] 815
```

```
dtm.train <- DocumentTermMatrix(revs.train, control=list(dictionary = fivefreq))
dim(dtm.train)
```

```
## [1] 35000   815
```

```
dtm.test <- DocumentTermMatrix(revs.test, control=list(dictionary = fivefreq))
dim(dtm.test)
```

```
## [1] 15000   815
```

6.3 MODEL BUILDING - Naive Bayes

Training and testing using Naive Bayes Classifier

```
convert_count_to_boolean <- function(x) {
  y <- ifelse(x > 0, 1,0)
  y <- factor(y, levels=c(0,1), labels=c("No", "Yes"))
  y
}

trainNB <- apply(dtm.train, 2, convert_count_to_boolean)
testNB <- apply(dtm.test, 2, convert_count_to_boolean)


classifier <- naiveBayes(trainNB, as.factor(df.train$sentiment_cat), laplace = 0)

pred <- predict(classifier, newdata=testNB)
table("Predictions"= pred,  "Actual" = as.factor(df.test$sentiment_cat))
```

```
##            Actual
## Predictions  Neg  Pos
##         Neg 9029 1047
##         Pos 2851 1391
```

6.4 Confusion Matrix

Model Performance

```
conf.mat <- confusionMatrix(pred, df.test$sentiment_cat)
conf.mat
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction  Neg  Pos
##        Neg 9029 1047
##        Pos 2851 1391
##                                           
##                Accuracy : 0.7278          
##                  95% CI : (0.7204, 0.735) 
##     No Information Rate : 0.8297          
##     P-Value [Acc > NIR] : 1               
##                                           
##                   Kappa : 0.2555          
##  Mcnemar's Test P-Value : <2e-16          
##                                           
##             Sensitivity : 0.7600          
##             Specificity : 0.5705          
##          Pos Pred Value : 0.8961          
##          Neg Pred Value : 0.3279          
##              Prevalence : 0.8297          
##          Detection Rate : 0.6306          
##    Detection Prevalence : 0.7037          
##       Balanced Accuracy : 0.6653          
##                                           
##        'Positive' Class : Neg             
## 
```