Case 4-F

Question 1

Due to the huge size of the data set, we decided to take a sample set of around 1,200 observations. First, we randomly take 1,400 samples. Then we filter the languages of the comments so we end up with 1,271 observations, all in English.

```
library(tidyverse)
library(tm)
library(igraph)
library(ggraph)
library(ggthemes)
library(wordcloud)
library(textcat)
library(scales)
library(tidytext)
library(micEcon)
library(dplyr)
library(tidyr)
library(ggplot2)
library(plotrix)

rev.df <- read.csv("~/Documents/Spring 2018/Big Data 2/Assignments/4:19
 team/reviews.csv", encoding="latin-1", stringsAsFactors=FALSE)
set.seed(123)
rev <- rev.df[sample(c(1:dim(rev.df)[1]), 1400),]
rev$language<-textcat(rev[,6])
rev<-subset(rev,language=="english")
review <- as.character(rev$comments)
```

Question2

First we generate a corpus of the sample set. Then we create a function to clean the corpus. To decide on the stopwords, we first use the default stopwords to draw the barplot of word frequency. Then we look at the most frequent words, finding the words that contain little information or express little attitude, such as "boston" or "airbnb". The set of stopwords is tried and adjusted throughout the whole process of our case analysis. Finally we come up with a new stopwords set with the default stopwords excluding "not" and all the "n't"s, and other words such as "place","really","everything","home", and "definitely".

```
review_source<-VectorSource(review)
review_corpus<-VCorpus(review_source)

head(review_corpus)
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 6
```

```r
# Alter the function code to match the instructions
# Get Customized Stopwords
exceptions <- grep(pattern = "not|n't", x = stopwords(), value = TRUE)
my_stopwords <- setdiff(stopwords("en"), exceptions)

clean_corpus <- function(corpus){
  corpus <- tm_map(corpus, removePunctuation)
  corpus <- tm_map(corpus, stripWhitespace)
  corpus <- tm_map(corpus, removeNumbers)
  corpus <- tm_map(corpus, content_transformer(tolower))
  corpus <- tm_map(corpus, removeWords,
                   c(my_stopwords, "boston","airbnb","stay","place","re
ally","everything","home","definitely","also","just","made","back","get
","one"))
  return(corpus)
}
clean_corp<-clean_corpus(review_corpus)
```

Question3

The TermDocument Matrix has 4636 rows and 1271 columns.

```r
# Create the tdm from the corpus: review_tdm
review_tdm <- TermDocumentMatrix(clean_corp)
# Print out review_tdm data
print(review_tdm)
```

```
## <<TermDocumentMatrix (terms: 4636, documents: 1271)>>
## Non-/sparse entries: 30678/5861678
## Sparsity           : 99%
## Maximal term length: 73
## Weighting          : term frequency (tf)
```

```r
# Convert coffee_dtm to a matrix: review_m
review_m <- as.matrix(review_tdm)
# Print the dimensions of review_m
dim(review_m)
```

```
## [1] 4636 1271
```

Question4

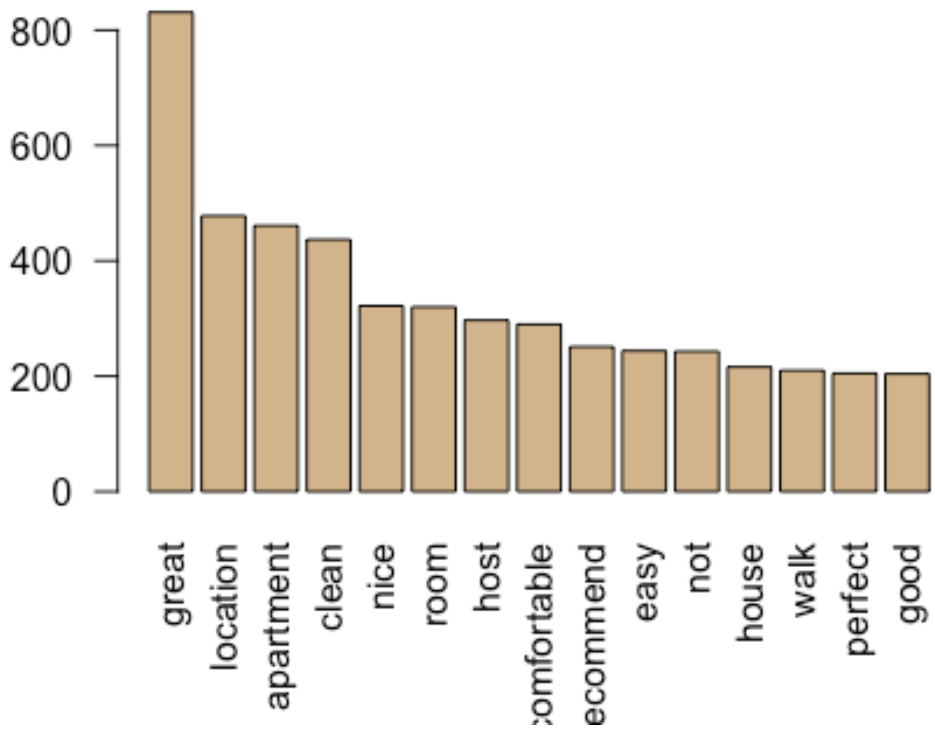
The 15 most frequent terms and their frequency are shown below. All the adjectives in the most frequent terms are positive, such as "great", "perfect","clean" and "easy", and the verb "recommend" shows positive attitude as well. This seems to us not as expectation because in the real world there must be negative reviews. However, the frequency of "not" catches our attention. What if it was "not recommend" or "not easy" instead of "recommend" and "easy"? This observation encourages us to dig deeper into the bigrams in the research later in this case.

As for the nouns, the most frequent ones are "location", "apartment", "room", "host", "house" and "walk". All of them seem like the things people consider when assessing an airbnb stay. So the nouns make a lot of sense in our opinion.

```
# Calculate the rowSums: term_frequency
term_frequency <- rowSums(review_m)
# Sort term_frequency in descending order
term_frequency <- sort(term_frequency,decreasing=TRUE)
# View the top 15 most common words
head(term_frequency,15)
```

```
##       great    location   apartment       clean        nice           r
oom
##         831         478         461         437         322            
320
##        host comfortable   recommend        easy         not          ho
use
##         297         290         251         244         243            
216
##        walk     perfect        good 
##         210         205         204
```

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



Question5

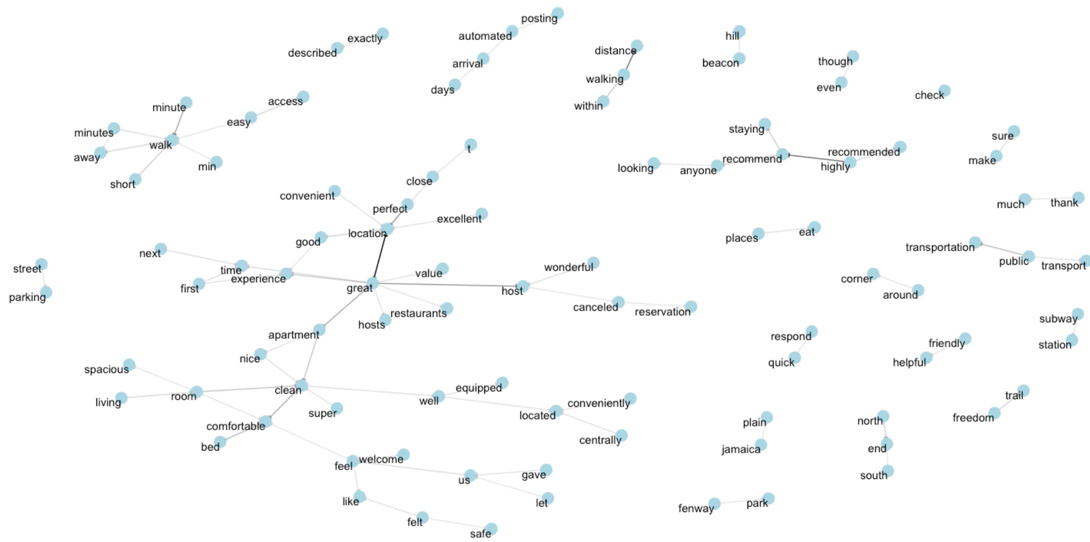
The most frequent bigrams and the plot are shown below. The 10 most frequent bigrams are "great location", "walking distance" "highly recommended", "great host", "minute walk", "public transportaion", "apartment great", "room clean", "apartment clean", "clean comfortable".

Here we find out that "walk" in the airbnb reviews is usually used to discribe distance and location. Just like the case of "walk", the bigrams make sense of a lot of single words by joining them together. For example, the places of interests such as "beacon hill" and "jamaica plain", we may have no clue when we look at "hill" or "beacon" as single words, but the 2 words connected we can understand immediately. The bigrams also show directions of single words. Through bigrams we see that the most frequent single word "great" represents "great location","great host" and "great apartment", which gives us a lot more information to analysis.

```
#Transform clean_corp into a data frame
rev_df_new<-data.frame(text = sapply(clean_corp, as.character), strings
AsFactors = FALSE)
rev_df_new$document<-c(1:nrow(rev_df_new))
# Bigrams
text<-as.character(rev_df_new$text)
bigrams <- unnest_tokens(rev_df_new, bigram, text, token="ngrams", n=2)
```

```
bigrams %>%
  count(bigram, sort = TRUE)

## # A tibble: 25,027 x 2
##    bigram                     n
##    <chr>                  <int>
##  1 great location           141
##  2 walking distance          91
##  3 highly recommend          90
##  4 great host                65
##  5 minute walk               57
##  6 public transportation     42
##  7 apartment great           40
##  8 room clean                37
##  9 apartment clean           36
## 10 clean comfortable         36
## # ... with 25,017 more rows

bigrams_separated <- bigrams %>%
  separate(bigram, c("word1", "word2"), sep = " ")
bigram_counts <- bigrams_separated %>%
  count(word1, word2, sort = TRUE)
head(bigram_counts, 10)

## # A tibble: 10 x 3
##    word1     word2              n
##    <chr>     <chr>          <int>
##  1 great     location         141
##  2 walking   distance          91
##  3 highly    recommend         90
##  4 great     host              65
##  5 minute    walk              57
##  6 public    transportation    42
##  7 apartment great             40
##  8 room      clean             37
##  9 apartment clean             36
## 10 clean     comfortable       36

bigram_graph <- bigram_counts %>%
  filter(n > 12) %>%
  graph_from_data_frame()
bigram_graph

## IGRAPH 9e669c8 DN-- 97 87 --
## + attr: name (v/c), n (e/n)
## + edges from 9e669c8 (vertex names):
##  [1] great    ->location       walking  ->distance
##  [3] highly   ->recommend      great    ->host
##  [5] minute   ->walk           public   ->transportation
##  [7] apartment->great          room     ->clean
##  [9] apartment->clean          clean    ->comfortable
```

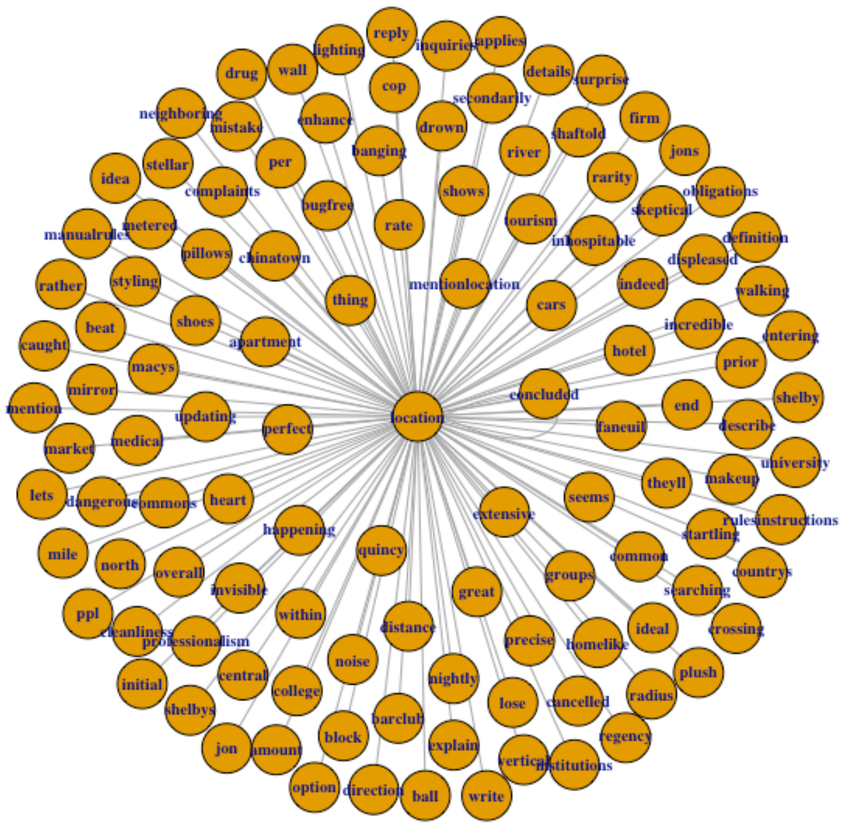
```
## [11] within   ->walking        recommend->staying
## [13] location ->great          bed      ->comfortable
## [15] great    ->experience     north    ->end
## + ... omitted several edges

# Plot bigrams
set.seed(1995)
a <- grid::arrow(type = "closed", length = unit(.05, "inches"))
ggraph(bigram_graph, layout = "fr") +
  geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
                 arrow = a, end_cap = circle(.05, 'inches')) +
  geom_node_point(color = "lightblue", size = 5) +
  geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
  theme_void()
```



Question 6

```
# Word Network
associations <- findAssocs(review_tdm,"location",0.1)
g1<-data.frame(associations$location)
colnames(g1)<-"distance"
g2<-data.frame(0)
rownames(g2)<-"location"
colnames(g2)<-"distance"
g3<-data.frame(rbind(g2,g1))
g3[,2]<-g3[,1]
g3[,3]<-g3[,2]
g3[,3]<-1/g3[,3]
g3[1,3]<-0
g3[,1]<-"location"
g3[,2]<-rownames(g3)
rownames(g3)<-c(1:118)
g4<-graph_from_data_frame(g3, directed = F, vertices = g3[,2])
coords <- layout.fruchterman.reingold(g4, niter=5000)
```

```r
plot(g4, layout = coords, vertex.label.font=2, vertex.size=13, vertex.l
abel.cex=0.65)
```

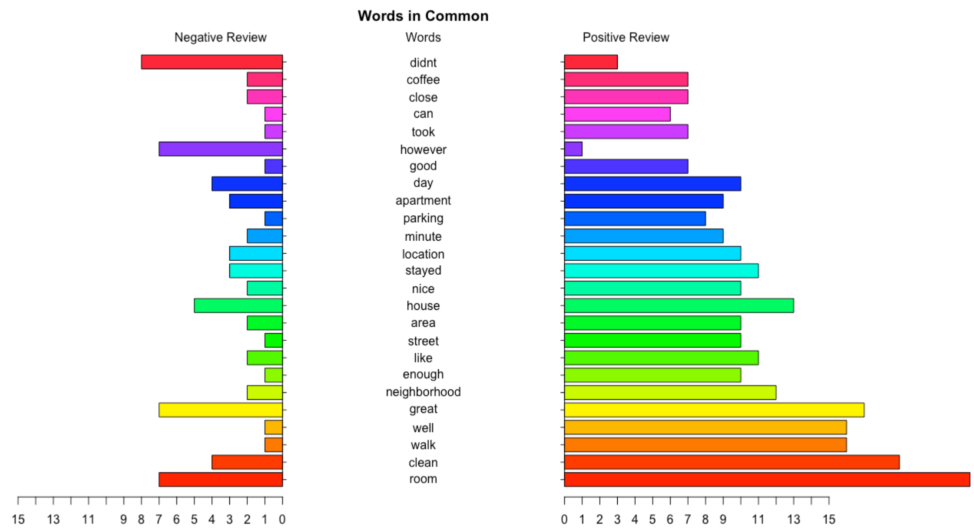


The "location" term network diagram reveals relationships between "location" and other terms from closeness to farness. The closer the distance is, the more frequently two terms are used together. For example, "Chinatown", "institution" and "university", "market" are close to "location". That is, when reviewers like the location of Airbnb's booking, they may frequently explain their reasons by mentioning convenience infrastructures and builds nearby to introduce the surroundings in detail. The adjective term such as "noise", "common", "inhospitable" are also close to "location", which express the reviewers' negative feelings towards booking location. Therefore, when reviews dislike the location of Airbnb's booking, they may frequently express the self-feeling directly. Some adjective terms such as "perfect", "precise", "incredible" are far away from location, which further verifies our guess. That is reviewers who leave positive reviews are more likely to describe the surroundings in details instead of expressing overall self-feelings directly, and reviewers who leave negative reviews are more likely to express overall self-feelings directly.

Question 7

```
# Using Bing lexicon
rev_tidy<-tidy(review_tdm)
bing <- get_sentiments("bing")
#inner join review dataset with bing lexicon
moby_sents <- inner_join(rev_tidy, bing, by = c("term" = "word"))
# calculate the polarity of each review statement
moby_tidy_sentiment <- moby_sents %>%
  count(document, sentiment, wt = count) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(polarity = positive - negative)
# filter the positive review with polarity more than 15, and negative r
eview with polarity less than 0
moby_tidy_small <- moby_tidy_sentiment %>%
  filter(polarity >= 15 |polarity <=0 )
# add label as positive and negative to reviews
moby_tidy_pol <- moby_tidy_small %>%
  mutate(
    pol = ifelse(polarity>0, "positive", "negative"))
# seperate postive and negative reviews into two datasets
pos<-subset(moby_tidy_pol,pol=="positive")
neg<-subset(moby_tidy_pol,pol=="negative")
# insert review context into two datasets
pos<-rev_df_new[rev_df_new$document %in% pos$document,]
neg<-rev_df_new[rev_df_new$document %in% neg$document,]
# combine two datasets
all_pos <- paste(pos$text, collapse = " ")
all_neg <- paste(neg$text, collapse = " ")
all <- c(all_pos, all_neg)
# transform into tdm
all_source <- VectorSource(all)
all_corpus <- VCorpus(all_source)
all_tdm <- TermDocumentMatrix(all_corpus)
colnames(all_tdm) <- c("positive review", "negative review")
# transform tdm into matrix
all_m <- as.matrix(all_tdm)
# find common words from negative and positive reviews
common_words <- subset(all_m, all_m[, 1] > 0 & all_m[, 2] > 0)
difference <- abs(common_words[, 1] - common_words[, 2])
common_words <- cbind(common_words, difference)
common_words <- common_words[order(common_words[, 3], decreasing = TRU
E), ]
# select 25 most common words
top25_df <- data.frame(x = common_words[1:25, 2],
                       y = common_words[1:25, 1],
                       labels = rownames(common_words[1:25, ]))
# Create the pyramid plot
pyramid.plot(top25_df$x, top25_df$y, labels = top25_df$labels,
             gap = 8, top.labels = c("Negative Review", "Words", "Posit
```

```
ive Review"),
             main = "Words in Common", laxlab = NULL,
             raxlab = NULL, unit = NULL)
```




```
## [1] 5.1 4.1 4.1 2.1
```

First, we assign sentiments to terms and calculate the total polarity of each review statement. Then we separate total reviews into negative and postive reviews in order to get common words.After drawing the pyramid plot, we find that the 25 most common words in negative and positive reviews are: room, clean, walk, well, great, neighborhood, enough, like, street, area, house, nice, stayed, location, minute, parking, apartment, day, good, however, took, can, close, coffee and didnt. The result makes sense. Didn't and however appears more times in negative reviews than in positive reviews, because the words or sentences following by these two words usually have negative meanings. For example, this house didn't provide heat. Or, the building is great, however, the room is not clean. And other words such as nice, close, clean and great appears much more times in postive reviews because people always mention these to illustrate how good the place is. Besides, we can see that many nouns appear many times both in negative and positive reviews like apartment, location, house, and room. We think this is because all the reviews focus on these aspects no matter customers appreciate the house or not.

Question 8

```
#Using FINN lexicon
afinn <- get_sentiments("afinn")
rev_afinn <- rev_tidy %>%
# Inner Join to AFINN lexicon
inner_join(afinn, by = c("term" = "word"))%>%
# Count by score and term
count(score, document)
head(rev_afinn)
```

```
## # A tibble: 6 x 3
##   score document     n
##   <int> <chr>    <int>
## 1    -3 1083         1
## 2    -3 11           1
## 3    -3 1168         1
## 4    -3 1170         1
## 5    -3 1192         2
## 6    -3 1255         1
```

```
# calculate total score of each review
rev_afinn$t_score<-rev_afinn$score*rev_afinn$n
rev_afinn[,1]<-NULL
rev_afinn[,2]<-NULL
rev_afinn_agg<-aggregate(rev_afinn$t_score, by=list(rev_afinn$document),
 FUN=sum)
colnames(rev_afinn_agg)<-c("document","total_score")
head(rev_afinn_agg)
```

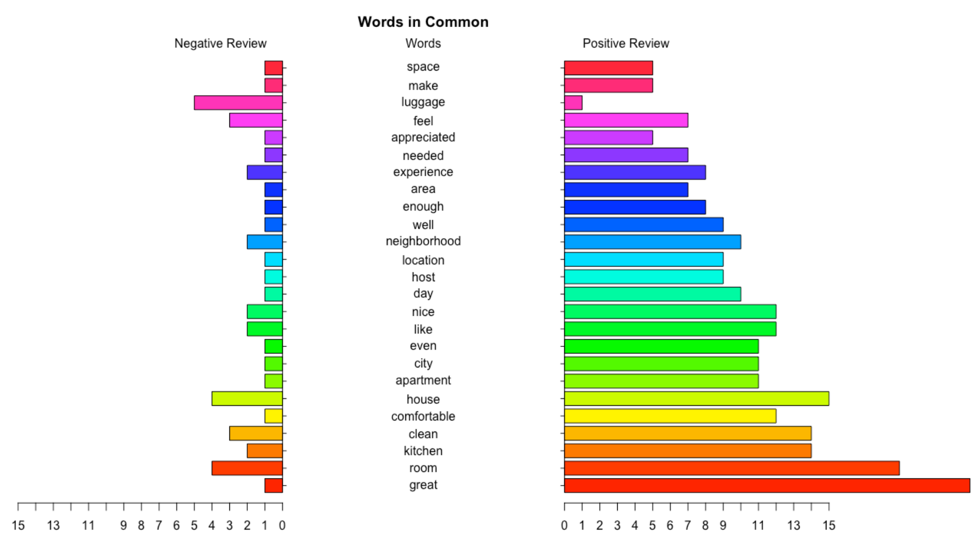
```
##   document total_score
## 1        1           5
## 2       10           3
## 3      100           3
## 4     1000          14
## 5     1001           3
## 6     1002           6
```

```
# filter the positive review with total score more than 30, and negativ
e review with total score less than 0
moby_tidy_small2 <- rev_afinn_agg %>%
  filter(total_score >= 25 |total_score <=0 )
# add label as positive and negative to reviews
moby_tidy_pol2 <- moby_tidy_small2 %>%
  mutate(
    pol = ifelse(total_score>0, "positive", "negative"))
head(moby_tidy_pol2)
```

```
##   document total_score      pol
## 1     1026          25 positive
## 2     1056          30 positive
## 3     1085          -1 negative
## 4     1112           0 negative
## 5     1139          28 positive
## 6     1160          -1 negative
```

```
# seperate postive and negative reviews into two datasets
pos2<-subset(moby_tidy_pol2,pol=="positive")
neg2<-subset(moby_tidy_pol2,pol=="negative")
# insert review context into two datasets
pos2<-rev_df_new[rev_df_new$document %in% pos2$document,]
neg2<-rev_df_new[rev_df_new$document %in% neg2$document,]
```

```r
# combine two datasets
all_pos2 <- paste(pos2$text, collapse = " ")
all_neg2 <- paste(neg2$text, collapse = " ")
all2 <- c(all_pos2, all_neg2)
# transform into tdm
all_source2 <- VectorSource(all2)
all_corpus2 <- VCorpus(all_source2)
all_tdm2 <- TermDocumentMatrix(all_corpus2)
colnames(all_tdm2) <- c("positive review", "negative review")
# transform tdm into matrix
all_m2 <- as.matrix(all_tdm2)
# find common words from negative and positive reviews
common_words2 <- subset(all_m2, all_m2[, 1] > 0 & all_m2[, 2] > 0)
difference2 <- abs(common_words2[, 1] - common_words2[, 2])
common_words2 <- cbind(common_words2, difference2)
common_words2 <- common_words2[order(common_words2[, 3], decreasing = T
RUE), ]
# select top 25 common words
top25_df2 <- data.frame(x = common_words2[1:25, 2],
                        y = common_words2[1:25, 1],
                        labels = rownames(common_words2[1:25, ]))
# Create the pyramid plot
pyramid.plot(top25_df2$x, top25_df2$y, labels = top25_df2$labels,
             gap = 8, top.labels = c("Negative Review", "Words", "Posit
ive Review"),
             main = "Words in Common", laxlab = NULL,
             raxlab = NULL, unit = NULL)
```




```
## [1] 5.1 4.1 4.1 2.1
```

We repeat the same process by using FINN lexicon. And get 25 most common words in negative and positive reviews are: great, room, kitchen, clean, comfortable, house, apartment, city, even, like, nice, day, host, location, neighborhood, well, enough, area, experience, needed, appreciated, feel, luggage, make and space. In these words, only luggage appears much more times in negative reviews. Other common words shows more in positive reviews. Similar to using BING lexicon, room, kitchen and house appears many times both in positive and negative reviews. And compared to BING, FINN shows some new common words like kitchen, appreciated and city. The reason is that FINN considered scores of these words, and these words have high positive scores.

Question 9

```r
#Comparison Cloud
moby_tidy_small3 <- moby_tidy_sentiment %>%
  filter(polarity >= 0 |polarity <=0 )
# add label as positive and negative to reviews
moby_tidy_pol3 <- moby_tidy_small3 %>%
  mutate(
    pol = ifelse(polarity>0, "positive", "negative"))
# seperate postive and negative reviews into two datasets
pos3<-subset(moby_tidy_pol3,pol=="positive")
neg3<-subset(moby_tidy_pol3,pol=="negative")
# insert review context into two datasets
pos3<-rev_df_new[rev_df_new$document %in% pos3$document,]
neg3<-rev_df_new[rev_df_new$document %in% neg3$document,]
# combine two datasets
all_pos3 <- paste(pos3$text, collapse = " ")
all_neg3 <- paste(neg3$text, collapse = " ")
all3 <- c(all_pos3, all_neg3)
# transform into tdm
all_source3 <- VectorSource(all3)
all_corpus3 <- VCorpus(all_source3)
all_tdm3 <- TermDocumentMatrix(all_corpus3)
colnames(all_tdm3) <- c("positive review", "negative review")
# transform tdm into matrix
all_m3 <- as.matrix(all_tdm3)
comparison.cloud(all_m3,
                 max.words = 200,
                 colors = c("darkgreen", "darkred")
)
```

Both BING and FINN lexicon contain many English words with different sentiments assigned to them. The BING lexicon qualitatively defines words' characteristics as positive or negative while the FINN lexicon assigns these words with numbers between -5 to 5 (negative numbers refer to negative sentiments and positive numbers refer to positive sentiments). Compared to BING lexicon, FINN provides more quantitative base for analysis, and it can be widely used to calculate contributions of words by defining 'contribution=n*score'. In this case, we prefer to use BING lexicon to present common words analysis. From above two graphs, we can see that the BING plot shows more words frequency in negative review among the top 25 lists. One possible reason is that by calculating the characteristic of whole sentences, we may lose some negative sentences under FINN method (similar to weighted-average calculation) because the algorithm will reckon them as neutral or positive sentences. By using FINN lexicon, we get a more skewed plot. We can also see that the BING lexicon result are more explainable because word like 'however' appears much more frequently in negative review than in positive review, which means that this word is an important factor to justify the characteristic of the sentence as it does in real world.

Question 10

```
# Positive and negative words analysis
# Count review words frequency
count(pos3)
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1  1235

count(neg3)

## # A tibble: 1 x 1
##       n
##   <int>
## 1    19

# Calculate review words intensity
moby_tidy_pol3 %>% filter(polarity<0) %>% summarise(mean=mean(polarity))

## # A tibble: 1 x 1
##    mean
##   <dbl>
## 1 -2.09

moby_tidy_pol3 %>% filter(polarity>0) %>% summarise(mean=mean(polarity))

## # A tibble: 1 x 1
##    mean
##   <dbl>
## 1  5.17
```

Based on our previous analysis of Airbnb review, we find that the Airbnb in Boston Area enjoy a relative high reputation with usually more positive than negative feedbacks from customers. From our sample, we can see that positive reviews are dominant, and the number of positive reviews is about 6 times that of negative reviews. By splitting positive and negative reviews and calculating the polarity of sentence separately, we find that positive reviews have average score of 5.172 while negative reviews have average score of -2.091. These two numbers show that positive reviews are much more intense than negative reviews.