Case4

December 3, 2018

By Group B: Jiaren Ma, Tian Zhu, Huayou Tu

Data Preparation

For data preparation, we first set seed equal to 88 to generate random sequence for "review" data set. Then we subset 1000 random sample from it. In order to have a better analysis, we believe it's better to only choose "English" comments since we may don't understand other language.

```r
library(tidytext)
library(wordcloud)
library(ggplot2)
library(igraph)
library(qdap)
library(textcat)
library(widyr)
library(plotrix)
library(magrittr)
#review the data
review <- read.csv("c:/Users/zttzhu/My R Work/Case 4/case-4-airbnb-text-minin
g-b//Case 4/reviews.csv", stringsAsFactors = FALSE)

#set random seed
set.seed(88)
review.s <- review[sample(c(1:dim(review)[1]),1000),]
review.s$language <- textcat(review.s[,6])
review.eng <- subset(review.s, language == "english")
review.eng <- review.eng[,-7]
```

Data Cleaning and TDM

For data cleaning, we first make a corpus function. We include "replace abbreviation", "remove Punctuation", "remove Numbers", "removeWords" "tolower" and "delete whitespace" into our function. Then we apply this function to a corpus vector that we created. We then made a TDM.

The dimension for our TDM is 3677 rows and 891 columns. It should be noticed that since we only consider the “English” comments, so there are only 891 documents here.

Then we use “findFreqTerms” to find out some words that appear at least 100 times in all our sample comments. From these high frequency words, we choose to remove “boston", "definitely", "get", "really", "the", "this". We believe “boston” cannot provide very useful information here since we all know it’s Boston Airbnb, so it’s redundant for our information. For other words, they may become very useful in n-gram analysis, but we believe they do not carry much information here.

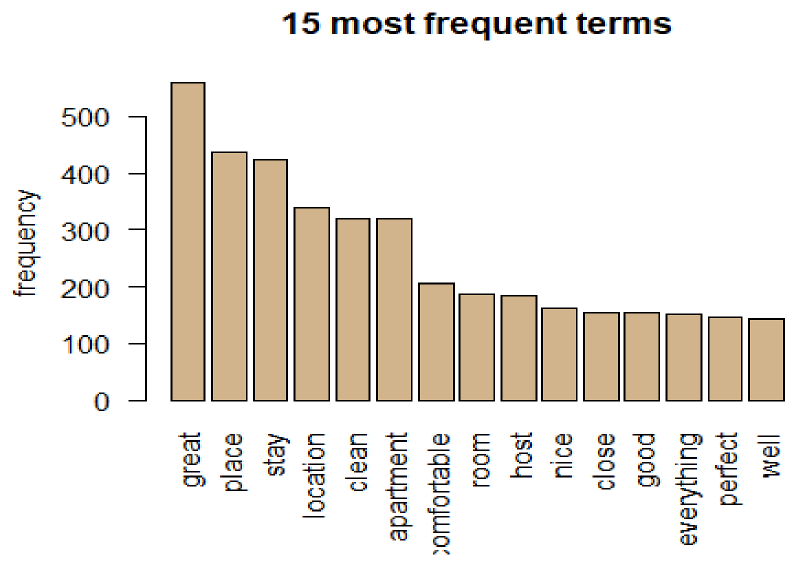
```r
#data cleaning
clean.corpus <- function(corpus){
  corpus <- tm_map(corpus, content_transformer(replace_abbreviation))
  corpus <- tm_map(corpus, removePunctuation)
  corpus <- tm_map(corpus, removeNumbers)
  corpus <- tm_map(corpus, removeWords, stopwords("en"))
  corpus <- tm_map(corpus, content_transformer(tolower))
  corpus <- tm_map(corpus, stripWhitespace)
}
re_vector <- VectorSource(review.eng$comments)
re_corpus <- VCorpus(re_vector)
re_clean <- clean.corpus(re_corpus)

# Create tf_tdm
re_tdm <- TermDocumentMatrix(re_clean, control=list(minWordLength=3))
re_tdm_m <- as.matrix(re_tdm)

# Dimensions of tdm matrix
dim(re_tdm_m)

#findFrequency
findFreqTerms(re_tdm, lowfreq = 100)

#Abandon words
re_clean2 <- tm_map(re_clean,removeWords, c("boston", "definitely", "get", "r
eally", "the", "this"))
```

15 most frequent terms

Here we calculated the sum of rows and use them to draw a bar chart. From this bar chart, the 15 most frequent terms are “great”, “place”, “stay”, “location”, “clean”, “apartment”, “comfortable”, “room”, “host”, “nice”, “close”, “good”, “everything”, “perfect”, “well”.

These words do make sense to us. From these 15 most frequent terms, we can get a very basic information about Airbnb such as “place”, “apartment”, “room”, “host”. Also, there are many positive words such as “great”, “good”, “perfect” appear many times. Especially “great” which appears more than 500 times. There are no bad words in our 15 most frequent terms. So overall, from our 891 samples, guests feel pretty satisfied with the quality of Boston Airbnb. Still, there are some words maybe more meaningful if we can do more analysis such as bigram on these words.

```r
re_tdm <- TermDocumentMatrix(re_clean2, control=list(minWordLength=3))
re_tdm_m <- as.matrix(re_tdm)
term_frequency <- rowSums(re_tdm_m)

# Sort term_frequency in descending order
term_frequency <- sort(term_frequency, decreasing = TRUE)

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



Bigram

Bigram analysis can help us to examine pairs of two consecutive words. We first do a very basic bigram analysis which only provide us some very common sentences such as “in the”, “to the”, “was very”, “and the”. We notice these so called “stop-words” cannot provide us any interesting information.

Next, we try to use “separate” function which can help us to remove these “stop-words”. This time, we find these short phases make much more sense to us and they can bring us more information comparing with single word analysis.

For example, they combine positive words with specific noun such as “perfect location”. For single word analysis, we cannot get such specific description. We need to guess what is perfect for. Is apartment perfect? Or is host perfect?

By combining positive words with some high frequency words which cannot bring very informative messages by just themselves such as “location”, “walk”, “transportation”, they can provide us much more useful information and give us a better understanding about all these comments.

We also use “ggraph to try to visualize short phase. The visualization graph gives us more details. For example, there is a short phase that set “wonderful” as the phase center and have two arrows to two different words which are “host” and “stay”. Through visualization, it is much easier to find the pattern within each single word and can help us to better analyze comments.

```
#5 bigram
re.bigram <-unnest_tokens(review.eng, bigram, comments, token = "ngrams", n =
 2)
re.bigram %>% count(bigram, sort = TRUE)
```

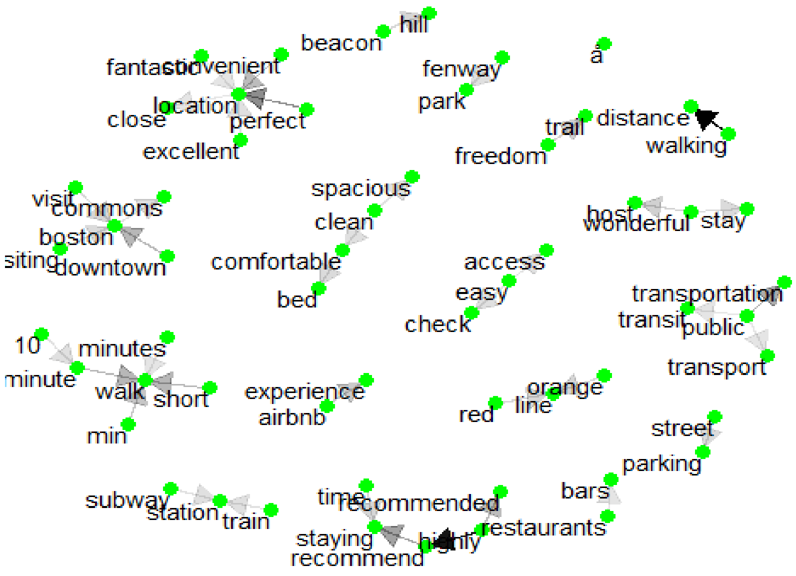
```
## # A tibble: 20,696 x 2
##    bigram             n
##    <chr>          <int>
##  1 in the           239
##  2 to the          218
##  3 a great         212
##  4 the apartment   175
##  5 was very        171
##  6 it was          161
##  7 and the         153
##  8 clean and       152
##  9 was a           144
## 10 is a            138
## # ... with 20,686 more rows
```

```
bigram_seperated <- re.bigram %>%
  separate(bigram, c("word1", "word2"), sep = " ")
bigram_filtered <- bigram_seperated %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word)
bigram_counts <- bigram_filtered %>% count(word1, word2, sort = TRUE)
bigram_counts
```

```
## # A tibble: 3,777 x 3
##    word1      word2              n
##    <chr>      <chr>          <int>
##  1 walking    distance          52
##  2 highly     recommend         48
##  3 perfect    location          25
##  4 recommend  staying           21
##  5 public     transportation    19
##  6 downtown   boston            17
##  7 minute     walk              17
##  8 short      walk              17
##  9 convenient location          16
## 10 highly     recommended       16
## # ... with 3,767 more rows
```

```
# filtering for relatively common combinations
bigram_graph <- bigram_counts %>%
  filter(n > 7) %>%
  graph_from_data_frame()

bigram_graph
```

```
## IGRAPH 12bab3f DN-- 57 40 --
## + attr: name (v/c), n (e/n)
## + edges from 12bab3f (vertex names):
##  [1] walking    ->distance       highly     ->recommend
##  [3] perfect    ->location       recommend  ->staying
##  [5] public     ->transportation downtown   ->boston
##  [7] minute     ->walk           short      ->walk
##  [9] convenient ->location       highly     ->recommended
## [11] min        ->walk           airbnb     ->experience
## [13] freedom    ->trail          wonderful  ->host
## [15] beacon     ->hill           orange     ->line

# plotting the bigram graph
set.seed(88)
ggraph(bigram_graph, layout = "fr") +
  geom_edge_link() +
  geom_node_point() +
  geom_node_text(aes(label = name), vjust = 1, hjust = 1)

# bigram graph with word sequences
set.seed(88)
a <- grid::arrow(type = "closed", length = unit(.15, "inches"))
ggraph(bigram_graph, layout = "fr") +
  geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
                 arrow = a, end_cap = circle(.03, 'inches')) +
  geom_node_point(color = "green", size = 3) +
  geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
  theme_void()
```

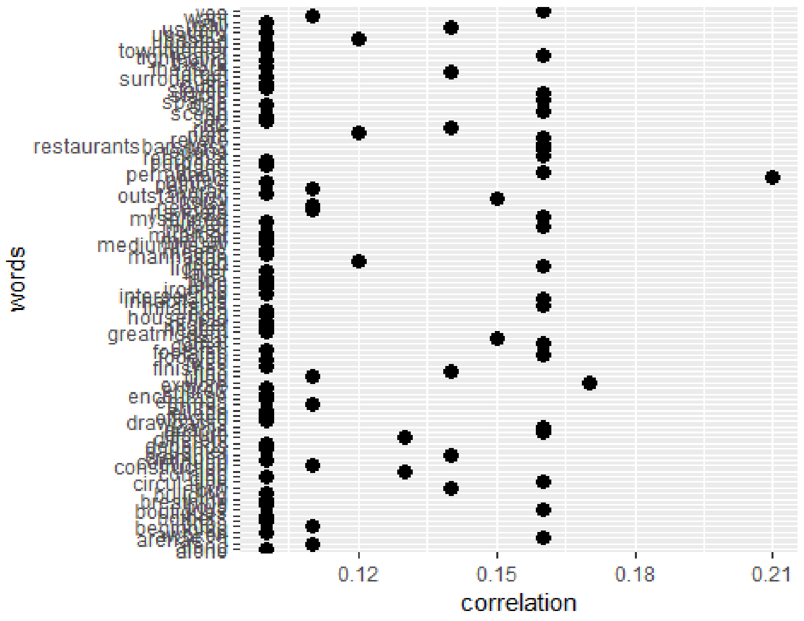


Location Association

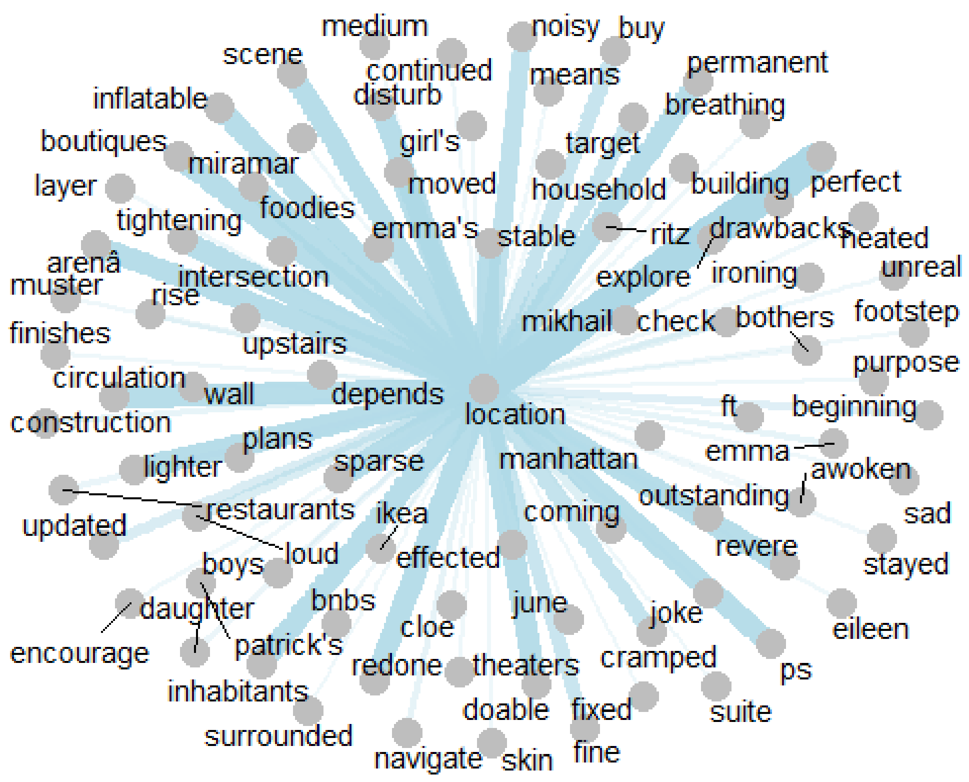
We used “findAssocs” method to find the top 15 words that have the highest correlation with the word “location”. As we can see from the result below, “perfect”, “explore ", and “arena” are the top three frequent words that always show up with “location”.

```
#6 Word association
re_associations <- findAssocs(re_tdm, c("location"), c(0.1))
re_associations
```

```
# Making the result into a data frame
re_associations.df <- as.data.frame(re_associations)
names <- rownames(re_associations.df)
rownames(re_associations.df) <- NULL
re_associations.df <- cbind(names, re_associations.df)
colnames(re_associations.df) <- c("words","correlation")
re_associations.df

# Plotting the values of associations.df
set.seed(88)
ggplot(re_associations.df, aes(y = words)) +
  geom_point(aes(x = correlation),
             data = re_associations.df, size = 3, sort = TRUE)
```



```
# Finding words that are highly correlated to "location"
re_words <- unnest_tokens(review.eng, word, comments)
re_words.df <- data.frame(group = re_words$reviewer_id, word = re_words$word)
re_word_pairs <- re_words.df %>%
  pairwise_cor(word, group, sort = TRUE) %>%
  filter(item1 == "location"|item2 == "location") %>%
  filter(!item1 %in% stop_words$word) %>%
  filter(!item2 %in% stop_words$word)

re_word_pairs <- re_word_pairs %>%
  filter(correlation > 0.1)
# Word network plot
set.seed(88)
re_word_pairs %>%
  graph_from_data_frame() %>%
  ggraph(layout = "fr") +
  geom_edge_link(aes(edge_alpha = correlation, edge_width = correlation*10),
                       edge_colour = "lightblue", show.legend = FALSE) +
  geom_node_point(color = "grey", size = 5) +
  geom_node_text(aes(label = name), repel = TRUE) + theme_void()
```



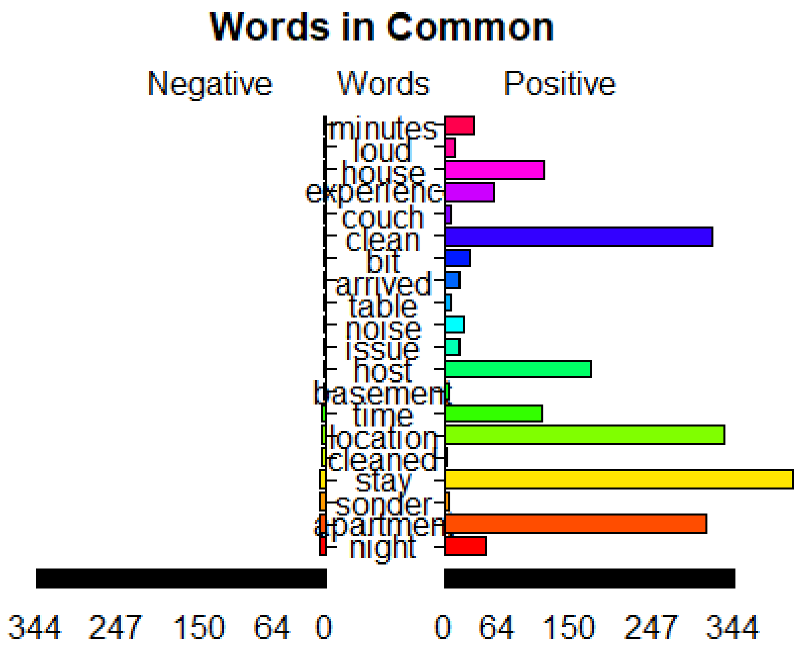
Bing

In this section, we analyze positiveness or negative by using Bing lexicon. We first subtract total negative terms from total positive terms in each comment. Then we put comments with total sentiment above zero to the positive group and below zero to the negative group.

After counting the common terms in each type of comment, we find that number of negative terms are much less than positive terms. Generally, negative comments overlap with positive ones. We choose to display the top 20 negative comments in the pyramid plot.

As can be seen from the plot, negative terms include “night”, “apartment”, “sonder”, “stay”. Generally, terms displayed are much lower in negative comments than in positive terms, with only a few words like “cleaned”, “basement” as exception.

```r
#7 Bing
# Tokenize the comments
set.seed(88)
re_review.words <- unnest_tokens(review.eng, word, comments)
# Inner_join bing with reviews terms
bing <- get_sentiments("bing")
reviews_bing_words <- inner_join(re_review.words, bing, by = c("word"))

# Count, Spread and Calculate sentiment for each term
reviews_bing_count <- reviews_bing_words %>%
  count(id, word, sentiment)
reviews_bing_spread <- reviews_bing_count %>%
  spread(sentiment,n,fill = 0)
reviews_bing_spread$sentiment <- reviews_bing_spread$positive - reviews_bing_
spread$negative

# Sum total_sentiment for each comment
reviews_bing_sentiment <- reviews_bing_spread %>%
  group_by(id) %>%
  summarize(total_sentiment = sum(sentiment))

# Filter positive and negative comments by sentiment
reviews_bing_with_sentiment <- inner_join(review.eng, reviews_bing_sentiment,
 by = c("id"))
pos_bing_comments <- reviews_bing_with_sentiment %>%
  filter(total_sentiment>0)
neg_bing_comments <- reviews_bing_with_sentiment %>%
  filter(total_sentiment<0)

# Count word freqency in both types of comments
pos_comments_bing_count <- pos_bing_comments %>%
  unnest_tokens(word, comments) %>%
  filter(!word %in% c(stop_words$word,"airbnb","boston")) %>%
  count(word)

neg_comments_bing_count <- neg_bing_comments %>%
  unnest_tokens(word, comments) %>%
  filter(!word %in% c(stop_words$word,"airbnb","boston")) %>%
  count(word)

# Find most common terms by inner_join pos and neg comments word count
common_words_bing <- inner_join(neg_comments_bing_count, pos_comments_bing_co
unt, by = c("word"))

# Sort by negative comments word frequency and create pyramid plot
common_words_bing <- common_words_bing[order(common_words_bing$n.x, decreasin
g = TRUE), ]
pyramid.plot(common_words_bing$n.x[1:20], common_words_bing$n.y[1:20], labels
 = common_words_bing$word[1:20], gap = 70, top.labels = c("Negative", "Words"
, "Positive"), main = "Words in Common", laxlab = NULL, raxlab = NULL, unit =
 NULL)
```

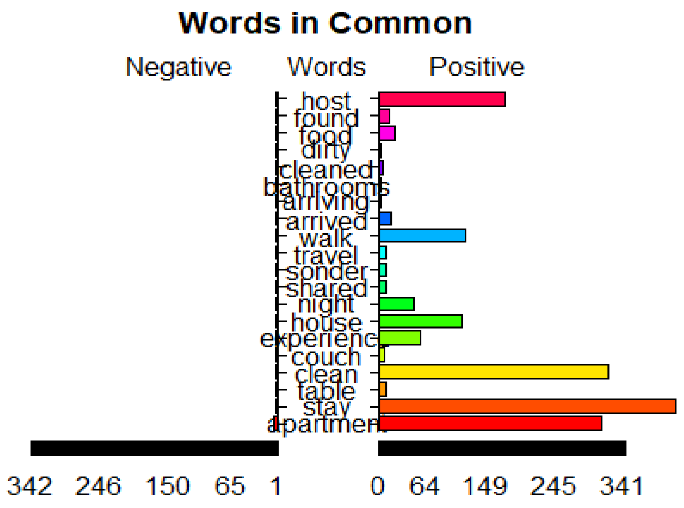


Afinn

In this section, we analyze each common by using Afinn. In all comments, there are positive comments and negative comments. Afinn could provide a sentiment score on each term, so we calculate the total score of each comment, and we put reviews with a total score of more than 0 in the positive comments group, and reviews with a total score of less than 0 in the negative comments group. We found out that the number of negative comments is still lower than 1% of the number of positive comments. As shown in the plot the number of positive comments is way higher than the negative due to the small portion of total negative reviews.

```
#8 Afinn
set.seed(88)
reviews_afinn <- get_sentiments("afinn")
reviews_afinn_words <- inner_join(re_review.words, reviews_afinn, by = c("wor
d"))
```

```
reviews_afinn_sentiment <- reviews_afinn_words %>%
  group_by(id) %>%
  summarize(total_score = sum(score))
reviews_afinn_with_sentiment <- inner_join(review.eng, reviews_afinn_sentimen
t, by = c("id"))
pos_comments_afinn <- reviews_afinn_with_sentiment %>%
  filter(total_score>0)
neg_comments_afinn <- reviews_afinn_with_sentiment %>%
  filter(total_score<0)

pos_comments_afinn_count <- pos_comments_afinn %>%
  unnest_tokens(word, comments) %>%
  filter(!word %in% c(stop_words$word,"airbnb","boston", "1")) %>%
  count(word)

neg_comments_afinn_count <- neg_comments_afinn %>%
  unnest_tokens(word, comments) %>%
  filter(!word %in% c(stop_words$word,"airbnb","boston")) %>%
  count(word)

common_words_afinn <- inner_join(neg_comments_afinn_count, pos_comments_afinn
_count, by = c("word"))
common_words_afinn <- inner_join(neg_comments_afinn_count, pos_comments_afinn
_count, by = c("word"))
common_words_afinn <- common_words_afinn[order(common_words_afinn$n.x, decrea
sing = TRUE), ]
pyramid.plot(common_words_afinn$n.x[1:20], common_words_afinn$n.y[1:20], labe
ls = common_words_afinn$word[1:20], gap = 70, top.labels = c("Negative", "Wor
ds", "Positive"), main = "Words in Common", laxlab = NULL, raxlab = NULL, uni
t = NULL)
```



Comparison Cloud

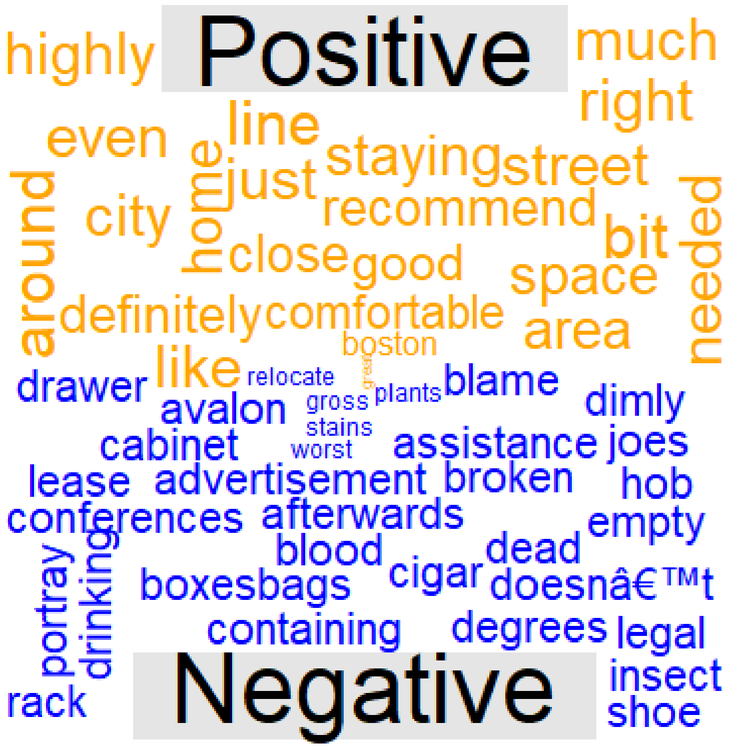
We think both plots have good capture of the positive or negative of words in comments. However, we prefer and choose Afinn because it measures from a [-5, 5] scale instead of dividing it binarily as negative or positive. This could help us compare:

1; two positive words, which one is more positive

2; two negative words, which one is more negative.

We then use Afinn to construct a comparison cloud by pasting and concatenating the terms and transforming them into TDM. Judging from the result, the very positive terms include “good”, “wonderful” and “recommended”, and the negative terms include “broken”, “blame” and “dead”, etc. This result is aligned with our common sense regarding to Airbnb reviews.

```r
# 9 Comparison Cloud
# Paste and collapse the positive comments
pos_terms <- paste(pos_comments_afinn$comments, collapse = " ")
# Paste and collapse the negative comments
neg_terms <- paste(neg_comments_afinn$comments, collapse = " ")
# Concatenate the terms
re_terms <- c(pos_terms, neg_terms)
# Pipe a VectorSource Corpus
re_corpus <- re_terms %>%
  VectorSource() %>%
  VCorpus()
# Create TFIDF TDM matrix
re_tdm_m2 <- TermDocumentMatrix( re_corpus, control = list( weighting = weigh
tTfIdf, removePunctuation = TRUE, removeNumbers = TRUE, stopwords = stopwords
(kind = "en") ) )

colnames(re_tdm_m2) <- c("Positive","Negative")
re_term_m <- as.matrix(re_tdm_m2)
# Create comparison cloud
comparison.cloud(re_term_m, colors = c("orange", "blue"), max.words = 400, sc
ale = c(0.5, 2.0))
```

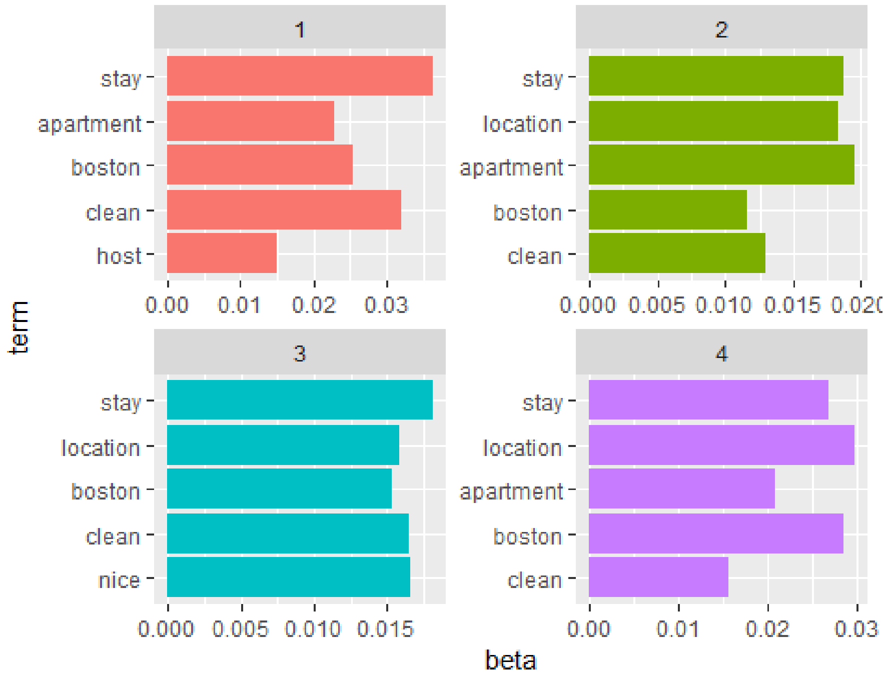


Most common words for four topics

We use LDA() function to construct the figure below. It seems that the most common terms include “stay”, “clean”, “boston”, “location”, “apartment”.

We find that the topics of reviews are somewhat homogeneous. In other words, they overlap with each other. The second and fourth topic are same, saying the apartments have good location and are clean. The first topic mentions about the host(positive or negative); the third topic generally gives good feedbacks for the staying.

```
#10 Most common words for four topics
re_words.count <- re_words %>% anti_join(stop_words) %>% count(reviewer_name,
 word, sort = TRUE) %>% ungroup

review_tdm <- re_words.count %>%
  cast_tdm(reviewer_name, word, n)

review_lda <- LDA(review_tdm, k = 4, control = list(seed = 88))
review_topics <- tidy(review_lda, matrix = "beta")


top_terms <- review_topics %>%
  group_by(topic) %>%
  top_n(5, beta) %>%
  ungroup() %>%
  arrange(topic, -beta)

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



Conclusion

After running all the analysis above, we found large differences between positive and negative comments. In all, positive comments take up more than 99% of the total comments. We can conclude that customers of Boston Airbnb generally have good experience, if they all honestly give their feedbacks.

Besides, despite the limited number of negative comments, the intensity of sentiment in negative comments is much lower than the intensity of positive comments. The highest positive Afinn score and number of high Afinn scores are much more than their negative counterparts.