Yelp_project

Yuanyuan Lin

11/25/2019

Introduction

The rise in E — commerce, has brought a significant rise in the importance of customer reviews. There are hundreds of review sites online and massive amounts of reviews for every product. Reviews could be scores, descriptions etc. Yelp is currently the most widely used restaurant across United States. In order to improve Yelp users' experience, there are some main methods to approach. The first one is sentiment analysis which is based on comments from customers. Also, we can use Latent Dirichlet allocation(LDA) for fitting a topic modeling. The other methods are using cluster analysis and Principle Component Analysis.The main goal of this project is to predict restaurant rating. Yelp rating prediction could help improve Yelp user's experience.

Data Cleaning

The dataset used here is from yelp open dataset website. Some of data are using API to get while others are downloaded from offical website. The data that are able to loaded using Yelp open dataset API has large limitations. Each time I only able to get 50 observations. I can only analysis for example some restaurants from Columbus,OH using API. As a result, I planned to download from official yelp open dataset website. This project mainly focused on review, users and business datasets from Yelp open data source and I mainly discuss the restaurants in OH.

```r
devtools::install_github("OmaymaS/yelpr")
```

```
## Skipping install of 'yelpr' from a github remote, the SHA1 (84734851) has not changed since last ins
##   Use `force = TRUE` to force installation
```

```r
library(yelpr)
api<-"GBORIlecmrZGAjvMJLmknx0F9dbCOoysGYU9LlhPSf4aSH54R76cwHBhM_d72a8j0p4iejBoCauyXDsDBbEO8zGsG6puTZXft
```

```r
#using yelp api to get some relevant data which has large limitations
location<-"Columbus,OH"
limit<-50
Yelp<-business_search(api_key = api,location = location, limit=limit)
```

```
## No encoding supplied: defaulting to UTF-8.
```

```r
Yelp1<-Yelp$business
```

```r
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
#library(tidytext)
#library(knitr)
#library(textdata)
#library(magrittr)
#library(wordcloud)
library(magrittr)
```

```
##
## Attaching package: 'magrittr'

## The following object is masked from 'package:purrr':
##
##     set_names

## The following object is masked from 'package:tidyr':
##
##     extract
```

```r
library(tidyverse)  # data manipulation
library(cluster)    # clustering algorithms
library(factoextra)
```

```
## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFCZ
```

```r
library(predkmeans)
library(SwarmSVM)
library(ClusterR)
```

```
## Loading required package: gtools
```

```r
#load relevant datasets
business<-read.csv("business.csv")
user<-read.csv("user.csv")
review<-read.csv("review.csv")
```

```r
#drop irrelevant columns
user<-user[,-c(1,9,10)]
```

```r
#omit NA values
user<-na.omit(user)
business_clean<-business[,-c(1,2,4:9,49:57)]
#library(tidyverse)
#library(knitr)
#library(magrittr)
```

Data Exploration(EDA)

1.Word Cloud Graph

```
library(wordcloud)
library(tidytext)
library(textdata)
library(dplyr)
#text analysis on customers review, Select 5000 random rows
comment_sample<-review[sample(nrow(review), 5000), ]
write.csv(comment_sample,file="comment_sample.csv")
comment_sample<-read.csv("comment_sample.csv")
#index<-sample(nrow(review),5000,replace=F)
#index<-as.data.frame(index)
#newdata<-data[index,]
#comment<-data.frame(review$text)
comment_sam<-data.frame(comment_sample$text)
comment_sam$comment_sam<-as.character(comment_sam$comment_sam)
#break review text into individual tokens and tranfrom into a tidy data structure
tidy_word_com <- comment_sam %>%
  unnest_tokens(word,comment_sam)
#remove stop words
tidy_word_com %>%
  anti_join(stop_words) %>%
#find the most common words in the reviews
  count(word) %>%
  with(wordcloud(word, n, max.words = 100,colors = brewer.pal(11, 'Dark2'), random.order = FALSE,rot.pe
```

By graphing the world cloud graph based on customer reviews, we found the most common words are “food”, “time” and “service”, “nice”, “love” and “service”. We would like to learn the importance of having high quality of service in restaurants.

2.Most Common Words in Review

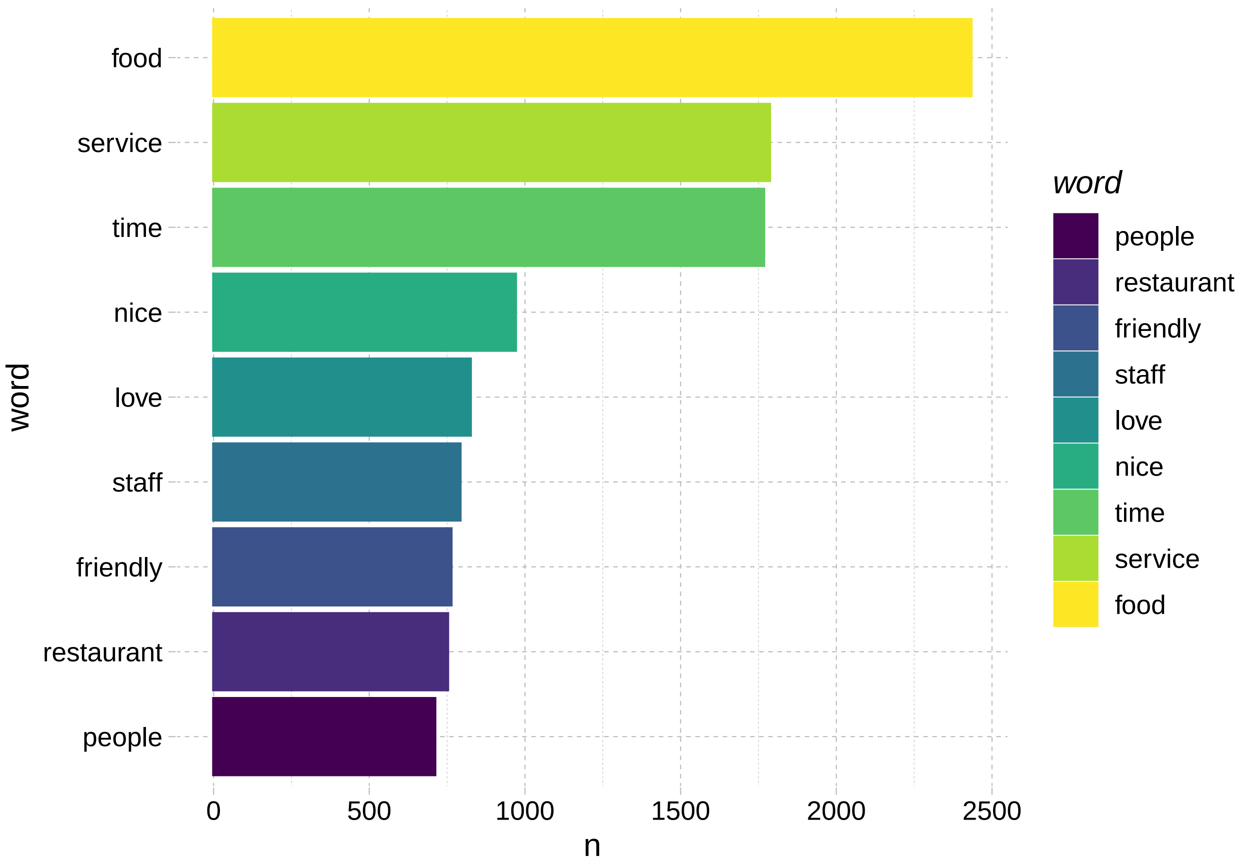
```
#most common words in the review by table
word_counts <-tidy_word_com %>% anti_join(stop_words, by="word")%>% count(word, sort = TRUE)
head(word_counts)
```

```
## # A tibble: 6 x 2
##   word        n
##   <chr>   <int>
## 1 food     2433
## 2 service  1786
## 3 time     1767
## 4 nice      970
## 5 love      825
## 6 staff     792
```

```
#change table into kable
library(magrittr)
library(knitr)
knitr::kable(head(word_counts))%>%kableExtra::kable_styling(bootstrap_options = c("striped", "hover"))
```

word	n
food	2433
service	1786
time	1767
nice	970
love	825
staff	792

```
library(ggthemes)
library(ggplot2)
#most common words in the review
tidy_word_com %>%
   count(word, sort = TRUE) %>% # count the number of words and sort them by frequency
  anti_join(stop_words, by="word")%>%
  filter(n > 700) %>% # filters the data to get only words that are used more than 80 times
  mutate(word = reorder(word, n)) %>% #Sentiment Analysis with inner join
  ggplot() + # plot function
  aes(x = word , y = n,fill=word,color=word) + # word on the x-axis, count (n) on the y-axis
  geom_col() + # we want to plot *col*umns
  coord_flip() +
scale_fill_viridis_d(option = "viridis") +
 scale_color_viridis_d(option = "viridis") +
 theme_pander()
```



Sentiment Analysis on Customer Reviews

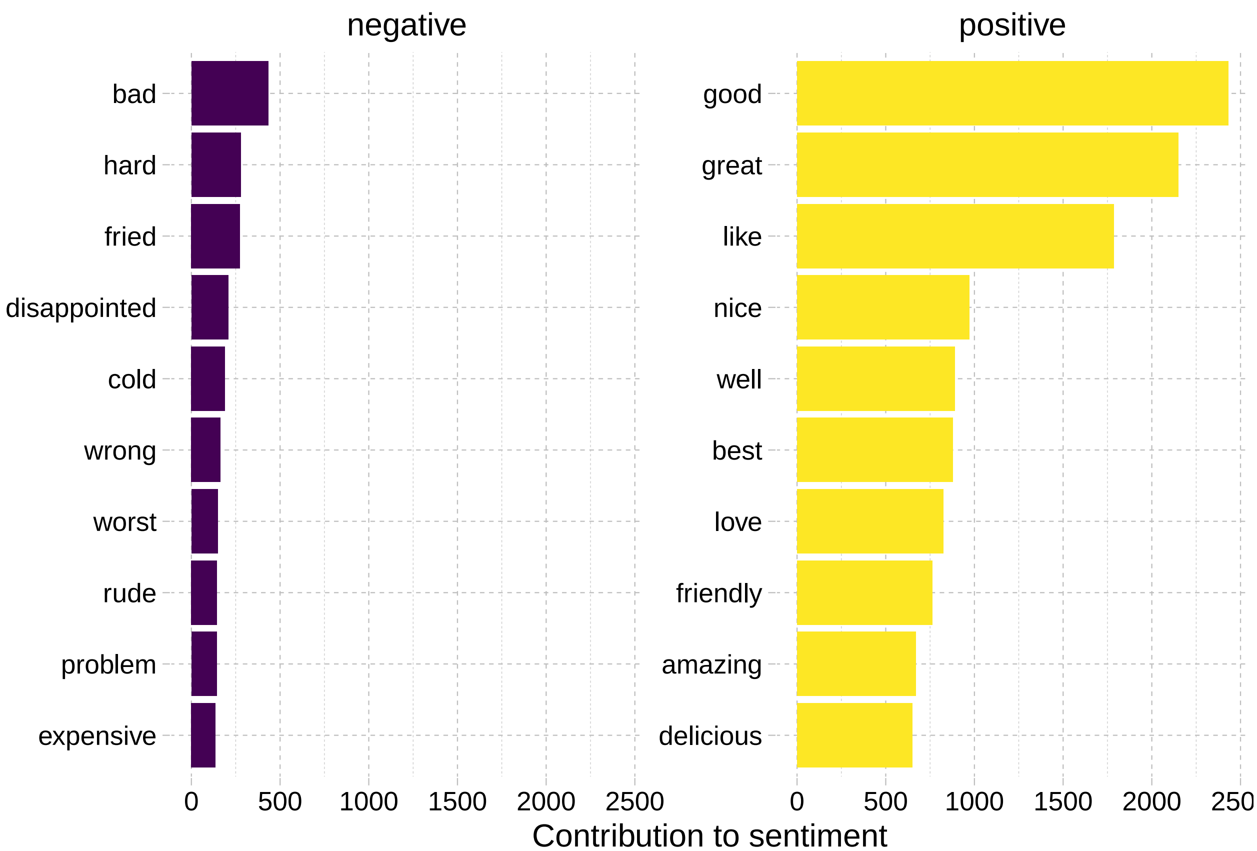
After process the opinion of restaurants computationally for identifying and categorizing, we are able to determine the attitude of the customers or say the customer towards the certain restaurtants is negative, positive or netural. I used bing to get sentiment count by doing single word and bigram analysis.

```
bing_word_counts <- tidy_word_com %>%
#find a sentiment score for each word using the Bing lexicon and inner_join()
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE)%>%
  ungroup()
knitr::kable(head(bing_word_counts))%>%kableExtra::kable_styling(bootstrap_options = c("striped", "hove
```

word	sentiment	n
good	positive	2430
great	positive	2149
like	positive	1785
nice	positive	970
well	positive	890
best	positive	878

```
#Most Common Positive and negative words
bing_word_counts %>%
#group by sentiment
```

```r
group_by(sentiment) %>%
#select top ten result
top_n(10) %>%
ungroup() %>%
#add a column to count how many number in each word
mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  labs(y = "Contribution to sentiment",
x = NULL) + coord_flip()+
scale_fill_viridis_d(option = "viridis") +
 scale_color_viridis_d(option = "viridis") +
 theme_pander()
```




```r
library(reshape2)
tidy_word_com %>%
#tag positive and negative words using inner join
inner_join(get_sentiments("bing")) %>%
#find the most positve and negative words
count(word, sentiment, sort = TRUE) %>%
#turn the data frame into a matrix with acast()
acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("gray20", "gray80"),
                   max.words = 100)
```


```r
#relationship between words
library(dplyr)
library(tidytext)
#examine pairs of two consecutive words
comment_bigrams<-comment_sam%>%
  unnest_tokens(bigram,comment_sam,token="ngrams",n= 2)
head(comment_bigrams)
```

```
##
## 1 -Smaller store. Very dark, maybe they don't ever turn the lights on since we live in the Valley of
## 2 -Smaller store. Very dark, maybe they don't ever turn the lights on since we live in the Valley of
## 3 -Smaller store. Very dark, maybe they don't ever turn the lights on since we live in the Valley of
## 4 -Smaller store. Very dark, maybe they don't ever turn the lights on since we live in the Valley of
## 5 -Smaller store. Very dark, maybe they don't ever turn the lights on since we live in the Valley of
## 6 -Smaller store. Very dark, maybe they don't ever turn the lights on since we live in the Valley of
##          bigram
## 1 smaller store
## 2    store very
## 3     very dark
## 4    dark maybe
## 5    maybe they
## 6    they don't
```

```r
#couting the most common bigrams
comment_bigrams%>% count(bigram, sort = TRUE)
```

```
## # A tibble: 201,463 x 2
```

```
##    bigram         n
##    <chr>      <int>
##  1 of the      1762
##  2 it was      1750
##  3 in the      1542
##  4 and the     1511
##  5 i was       1257
##  6 this place  1213
##  7 on the      1142
##  8 and i       1136
##  9 to the      1012
## 10 for the      990
## # ... with 201,453 more rows
```

```
library(tidyr)
#spilt a column into multiple columns based on delimiter,
bigrams_separated <- comment_bigrams %>%
      separate(bigram, c("word1", "word2"), sep = " ")
#remove cases where either is a stop word
bigrams_filtered <- bigrams_separated %>%
      filter(!word1 %in% stop_words$word) %>%
      filter(!word2 %in% stop_words$word)
# new bigram counts:
bigram_counts <- bigrams_filtered %>% count(word1, word2, sort = TRUE)
bigram_counts
```

```
## # A tibble: 46,666 x 3
##    word1    word2         n
##    <chr>    <chr>     <int>
##  1 customer service     235
##  2 highly   recommend   167
##  3 ice      cream       156
##  4 las      vegas       137
##  5 happy    hour         85
##  6 front    desk         84
##  7 5        stars        79
##  8 15       minutes      71
##  9 10       minutes      67
## 10 friendly staff        57
## # ... with 46,656 more rows
```

```
#use bigrams to provide context in sentiment analysis
bigrams_separated %>%
  filter(word1 == "not") %>%
  count(word1, word2, sort = TRUE)
```

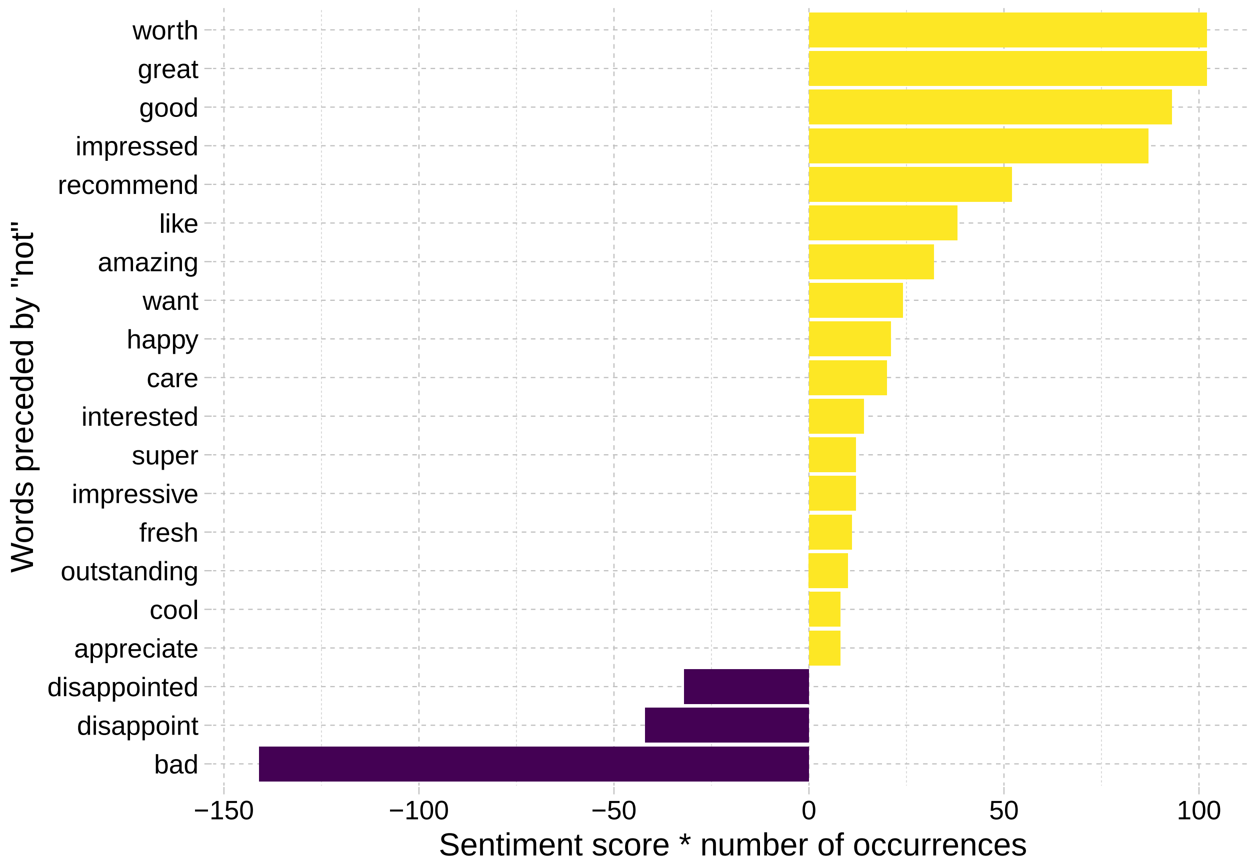
```
## # A tibble: 800 x 3
##    word1 word2     n
##    <chr> <chr> <int>
##  1 not   a       213
##  2 not   the     150
##  3 not   sure    116
##  4 not   only     98
```

```
##   5 not   to       88
##   6 not   be       86
##   7 not   too      84
##   8 not   have     65
##   9 not   so       65
##  10 not   even     59
## # ... with 790 more rows
```

```
#most frequent words that were preceded by "not" and were associated with a sentiment
AFINN <- get_sentiments("afinn")
not_words <- bigrams_separated %>%
  filter(word1 == "not") %>%
  inner_join(AFINN, by = c(word2 = "word")) %>%
  count(word2, value,sort = TRUE) %>%
  ungroup()
not_words
```

```
## # A tibble: 125 x 3
##    word2        value     n
##    <chr>        <dbl> <int>
##  1 worth            2    51
##  2 bad             -3    47
##  3 great            3    34
##  4 good             3    31
##  5 impressed        3    29
##  6 recommend        2    26
##  7 want             1    24
##  8 disappoint      -2    21
##  9 like             2    19
## 10 disappointed    -2    16
## # ... with 115 more rows
```

```
library(ggthemes)
#visualize most frequent words that were preceded by "not" and were associated with a sentiment
not_words %>%
#multiply their score by the number of times they appear
mutate(contribution = n * value) %>% arrange(desc(abs(contribution))) %>%
head(20) %>%
mutate(word2 = reorder(word2, contribution)) %>% ggplot(aes(word2, n * value, fill = n * value > 0)) +
  geom_col(show.legend = FALSE) +
xlab("Words preceded by \"not\"") +
ylab("Sentiment score * number of occurrences") + coord_flip()+
  scale_fill_viridis_d(option = "viridis") +
 scale_color_viridis_d(option = "viridis") +
 theme_pander()
```




```r
library(igraph)
#visualize a network of bigrams
bigram_counts
```

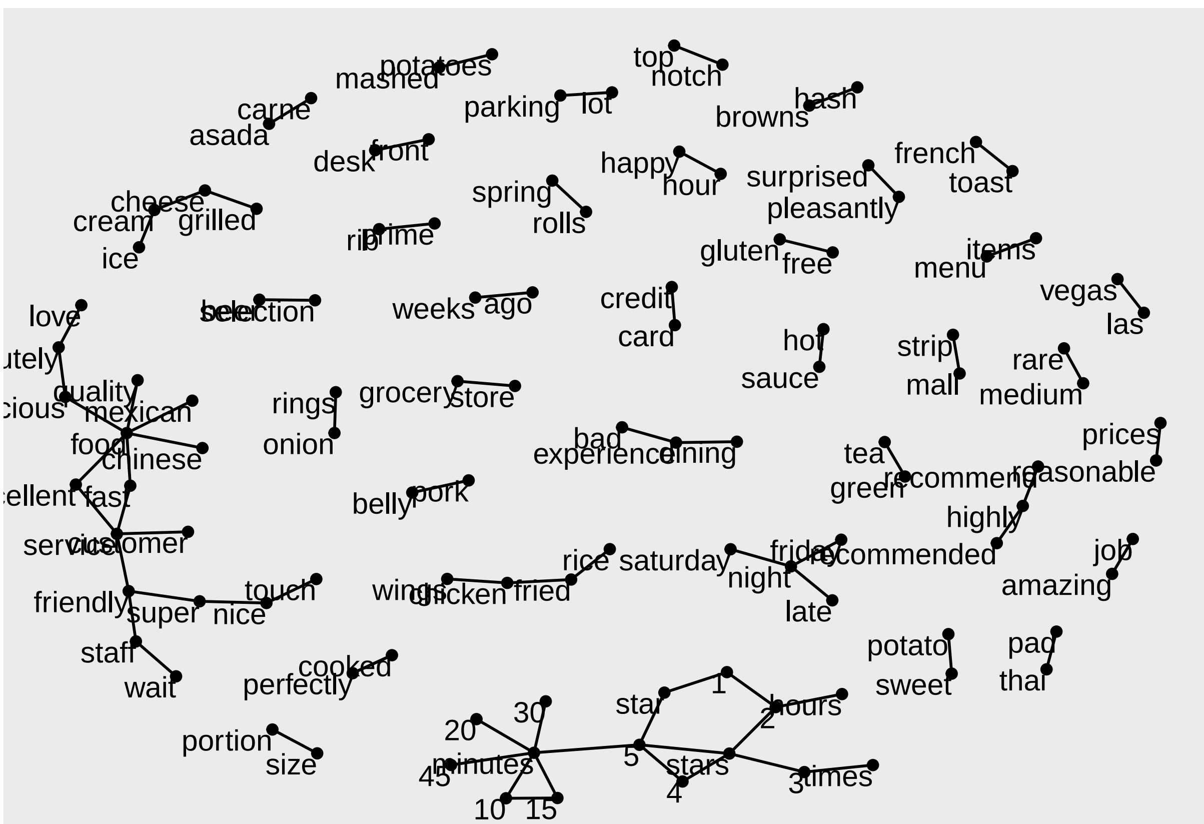
```
## # A tibble: 46,666 x 3
##    word1    word2         n
##    <chr>    <chr>     <int>
##  1 customer service     235
##  2 highly   recommend   167
##  3 ice      cream       156
##  4 las      vegas       137
##  5 happy    hour         85
##  6 front    desk         84
##  7 5        stars        79
##  8 15       minutes      71
##  9 10       minutes      67
## 10 friendly staff        57
## # ... with 46,656 more rows
```

```r
# filter for only relatively common combinations
bigram_graph <- bigram_counts %>%
  filter(n > 20) %>%
  graph_from_data_frame()
bigram_graph
```

```
## IGRAPH 5f7105c DN-- 110 79 --
```

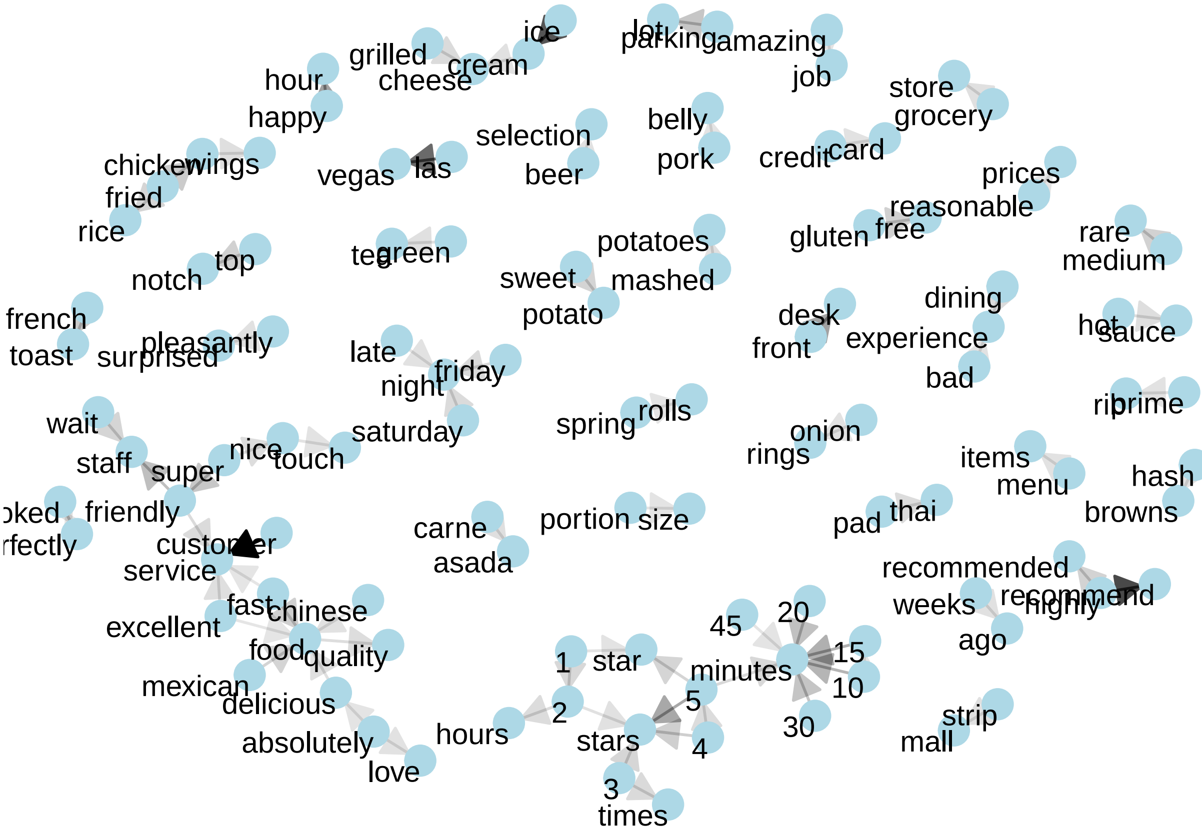
```
## + attr: name (v/c), n (e/n)
## + edges from 5f7105c (vertex names):
##  [1] customer->service     highly  ->recommend   ice     ->cream
##  [4] las     ->vegas       happy   ->hour        front   ->desk
##  [7] 5       ->stars       15      ->minutes     10      ->minutes
## [10] friendly->staff       20      ->minutes     30      ->minutes
## [13] parking ->lot         super   ->friendly    french  ->toast
## [16] fried   ->chicken     4       ->stars       fast    ->food
## [19] gluten  ->free        top     ->notch       medium  ->rare
## [22] highly  ->recommended mexican ->food        sweet   ->potato
## + ... omitted several edges
```

```
#convert igraph object into a ggraph
library(ggraph)
    set.seed(201)
#common bigrams in review, showing those that occcured more than 20 times and where neither word was a
    ggraph(bigram_graph, layout = "fr") +
      geom_edge_link() +
      geom_node_point() +
      geom_node_text(aes(label = name), vjust = 1, hjust = 1)
```




```
set.seed(2016)
#add directionality with an arrow
    a <- grid::arrow(type = "closed", length = unit(.15, "inches"))
#add edge_alpha aesthetic to link layer to make links transparent
ggraph(bigram_graph, layout = "fr") + geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
```

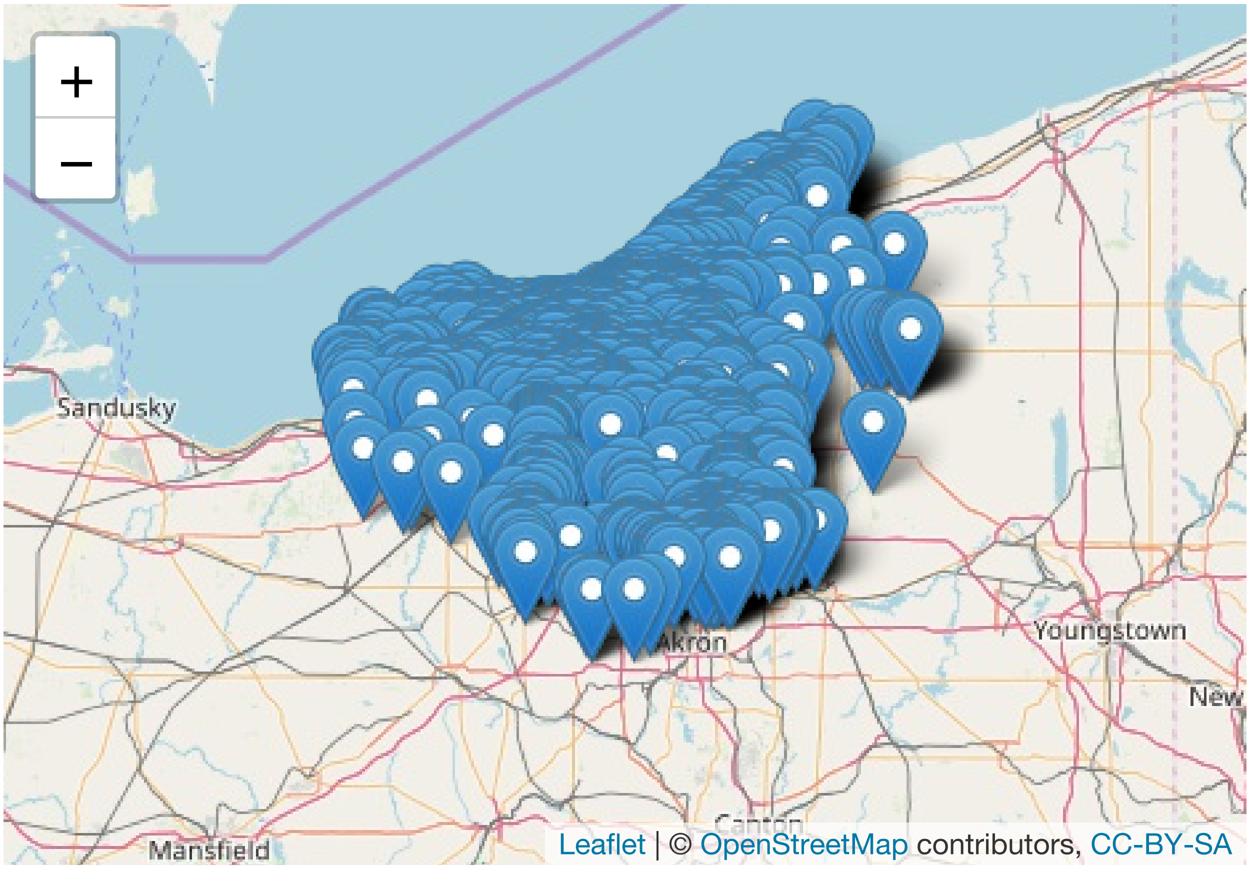
```
                        arrow = a, end_cap = circle(.07, 'inches')) +
      geom_node_point(color = "lightblue", size = 5) +
      geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
# add theme that is useful for plotting networks
      theme_void()
```



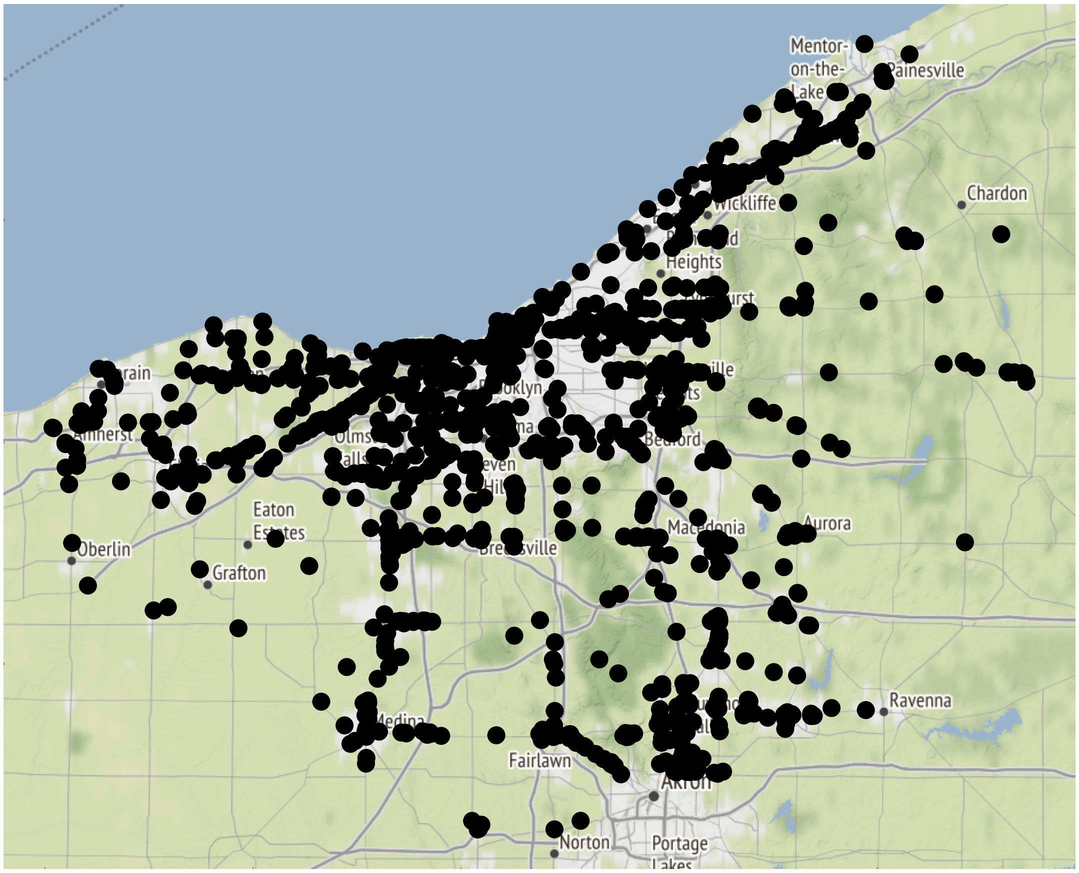
Mapping for yelp restaurants

The result of the distribution of yelp restaurants in OH does not looks good in leaflet since the pop-up logo is too large on the graph. But using ggmap, it gets more clear on the graph. We are able to notice that most restaurats are centered around the capital of Ohio, which is Columbus.

```
#library(dplyr)
#distribution of restaurants in OH
OH<-filter(business,state=="OH")
Latitude<-OH[,8]
Latitude<-data.frame(Latitude)
Latitude$long<-OH[,9]
library(leaflet)
Latitude %>%
  leaflet() %>%
  addTiles() %>%
  addMarkers(popup="sites")
```



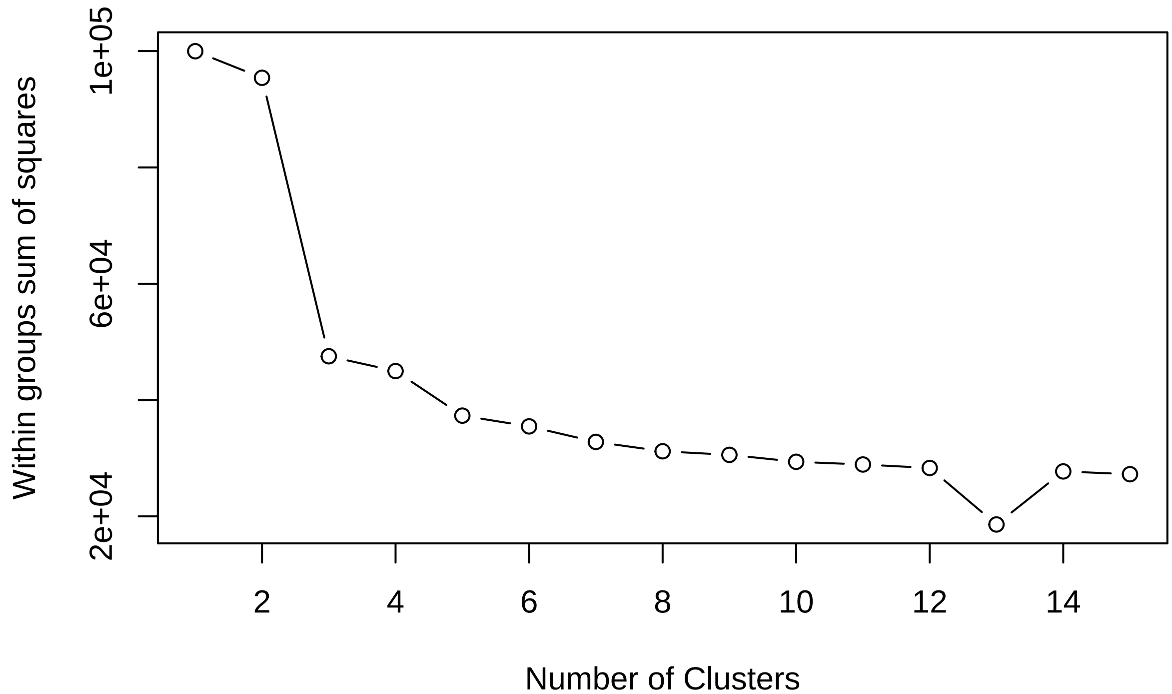

```
#distribution of restaurants in OH using ggmap
library(ggmap)
m6<-qmplot(long, Latitude, data = Latitude,
          color = I("black"), size = I(2.5))
m6
```



Cluster Analysis

I applied the k-means cluster analysis on the transformed data. Considering k-means is an efficient model for classifying observations based on their own characterstics, it would be available to have a general prediction for restaurants rating.

```
# Prepare Data
user<- na.omit(user) # listwise deletion of missing
user$user_id<-as.numeric(user$user_id)
user$name<-as.numeric(user$name)
user$yelping_since<-as.numeric(user$yelping_since)
user_cluster<-user[sample(nrow(user), 5000), ]
set.seed(123456789) ## to fix the random starting clusters
user_cluster <- scale(user_cluster) # standardize variables
# Determine number of clusters
wss <- (nrow(user_cluster)-1)*sum(apply(user_cluster,2,var))
for (i in 2:15) wss[i] <- sum(kmeans(user_cluster,
   centers=i)$withinss)
plot(1:15, wss, type="b", xlab="Number of Clusters",
  ylab="Within groups sum of squares")
```



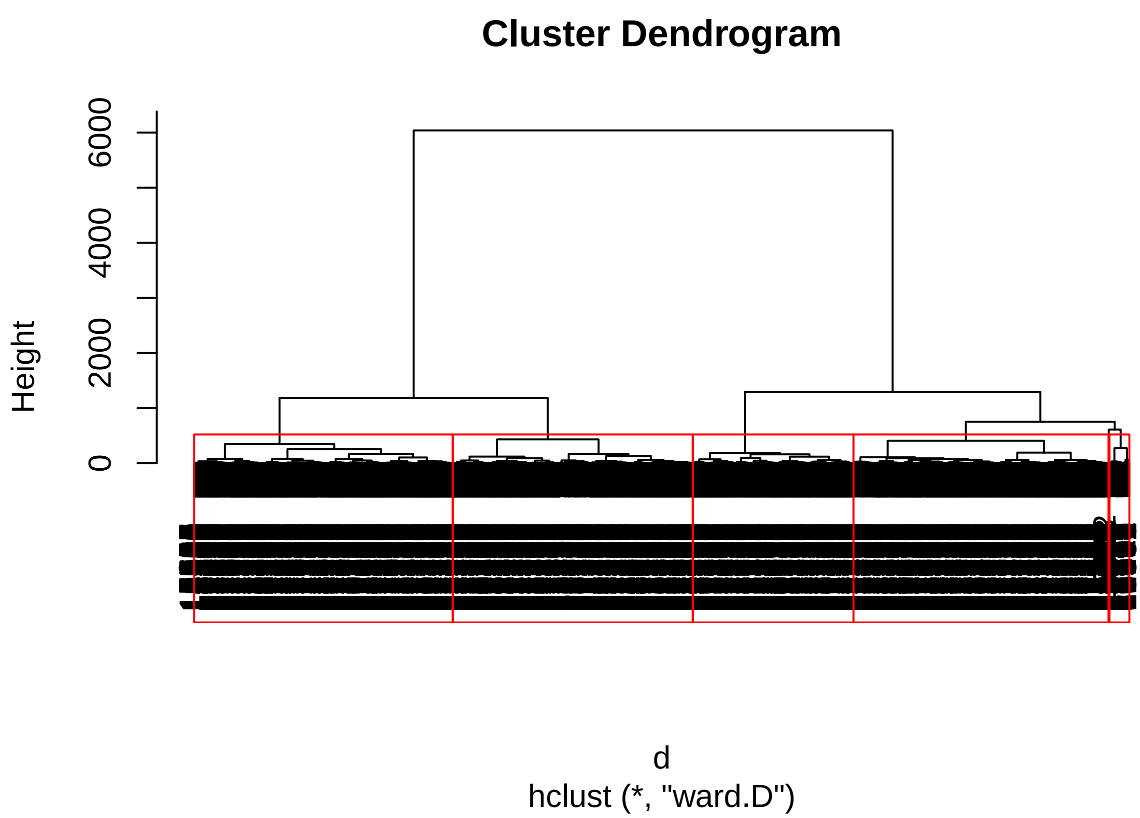

```r
# K-Means Cluster Analysis
fit <- kmeans(user_cluster, 6) # 5 cluster solution
# get cluster means
aggregate(user_cluster,by=list(fit$cluster),FUN=mean)
```

```
##   Group.1     user_id         name review_count yelping_since      useful
## 1       1 -0.91906672  0.256585450  0.168798694    -0.5769921  0.01047069
## 2       2 -0.11866911 -0.004004085 -0.291285070     0.5727427 -0.12435138
## 3       3 -0.05518235 -0.422717900  6.899720284    -1.3361428  5.80511827
## 4       4 -0.57919867  0.425162945 13.296616332    -1.0744859 26.49810896
## 5       5  0.21426983 -0.908444570 -0.190794190     0.4434717 -0.10716622
## 6       6  0.89644545  0.734856930  0.006919062    -0.1906257 -0.03443643
##         funny        cool        fans average_stars compliment_hot
## 1 -0.01557252 -0.01559568  0.03261909    0.26507010    -0.02897514
## 2 -0.08669724 -0.08677390 -0.16519033   -1.67528324    -0.04709605
## 3  4.96145675  4.94718203  8.20343084    0.06214809     2.53381724
## 4 27.35582801 27.33815134 19.97678853    0.06941237    28.67687749
## 5 -0.08044244 -0.07625464 -0.12883826    0.49372497    -0.04545744
## 6 -0.04218455 -0.04638486 -0.04601132    0.28616722    -0.03741816
##   compliment_more compliment_profile compliment_cute compliment_list
## 1    -0.009541035        -0.01902651     -0.02717443     -0.03255071
## 2    -0.071150913        -0.03443164     -0.04384500     -0.04332579
## 3     3.202551553         1.29889529      2.69603867      2.84429032
## 4    26.678780169        23.99756298     23.88405363     24.27521459
## 5    -0.062096065        -0.03268106     -0.04276400     -0.04120657
## 6    -0.034741841        -0.02644366     -0.03347191     -0.03431251
##   compliment_note compliment_plain compliment_cool compliment_funny
## 1     -0.01929018      -0.02558349     -0.02604140      -0.02604140
## 2     -0.06878202      -0.05937210     -0.06322169      -0.06322169
## 3      3.29265162       2.89394708      3.52317721       3.52317721
## 4     31.37795179      31.70201421     31.33411735      31.33411735
```

```
## 5     -0.06434422      -0.05643389     -0.06023546      -0.06023546
## 6     -0.03984558      -0.03912610     -0.04605437      -0.04605437
##   compliment_writer compliment_photos
## 1        -0.02135720       -0.02258673
## 2        -0.06201345       -0.03186629
## 3         2.75200122        1.02063714
## 4        31.85892220       25.85383559
## 5        -0.05795335       -0.03019616
## 6        -0.03746065       -0.02594663
```

```
# append cluster assignment
user_cluster <- data.frame(user_cluster, fit$cluster)
```

```
# Ward Hierarchical Clustering
d <- dist(user_cluster, method = "euclidean") # distance matrix
fit <- hclust(d, method="ward")
plot(fit) # display dendogram
groups <- cutree(fit, k=6) # cut tree into 5 clusters
# draw dendogram with red borders around the 5 clusters
rect.hclust(fit, k=6, border="red")
```

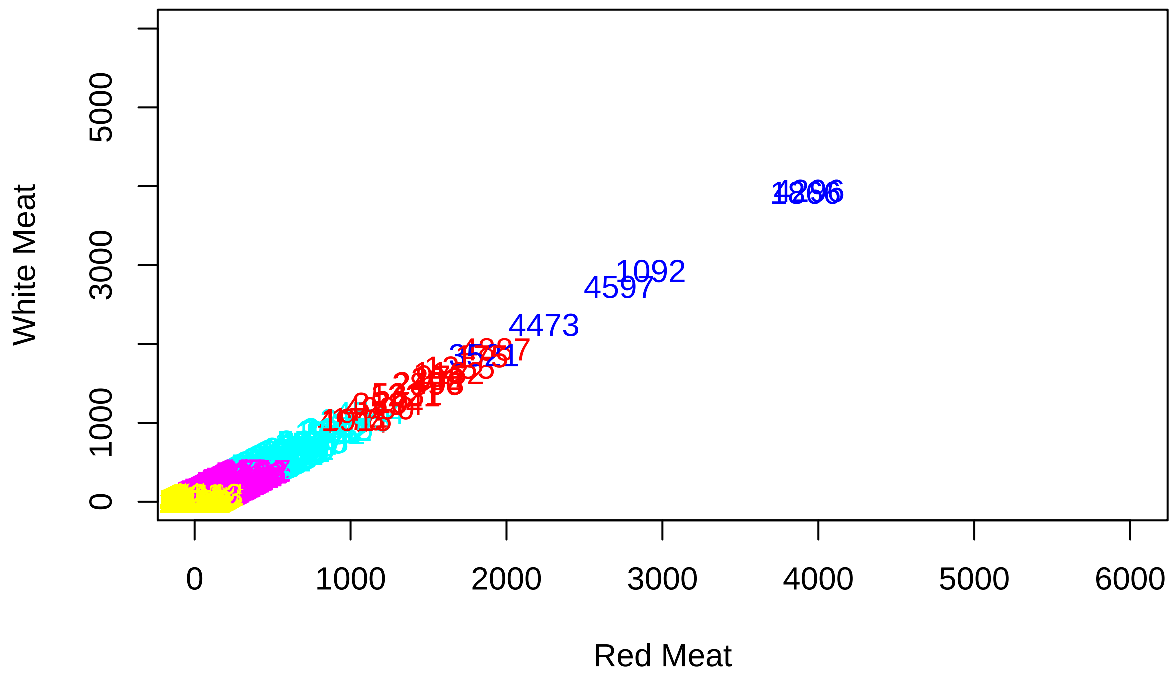



```
## list of cluster assignments
o=order(grpMeat$cluster)
head(data.frame(user_cluster_num$name[o],grpMeat$cluster[o]))
```

```
##       user_cluster_num.name.o. grpMeat.cluster.o.
## 11405                     4887                  1
```

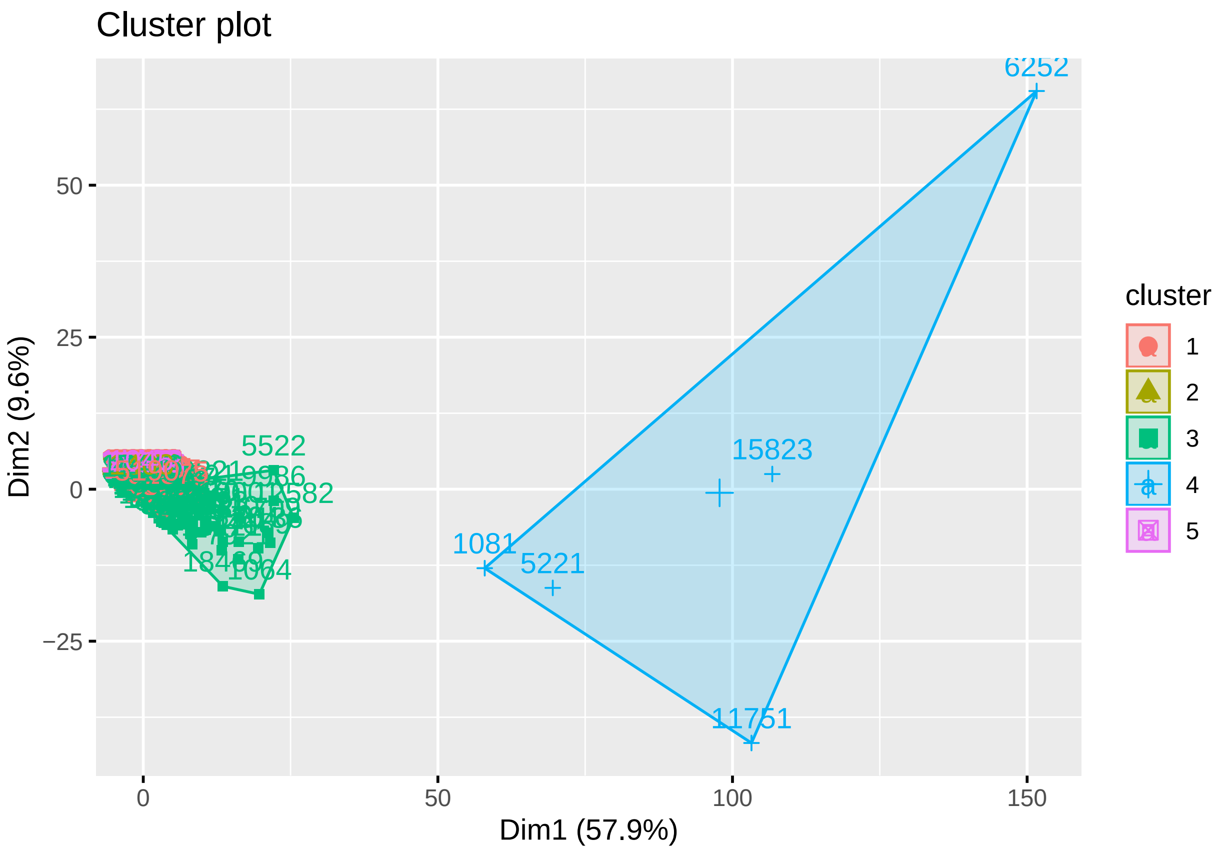
```
## 13243                       4964                 1
## 713                         5331                 1
## 11204                       1255                 1
## 14178                       4990                 1
## 8743                         176                 1
```

```r
plot(user_cluster_num$useful, user_cluster_num$funny, type="n", xlim=c(3,6000),ylim=c(3,6000), xlab="Rec
text(x=user_cluster_num$useful, y=user_cluster_num$useful, labels=user_cluster_num$name,col=grpMeat$clus
```

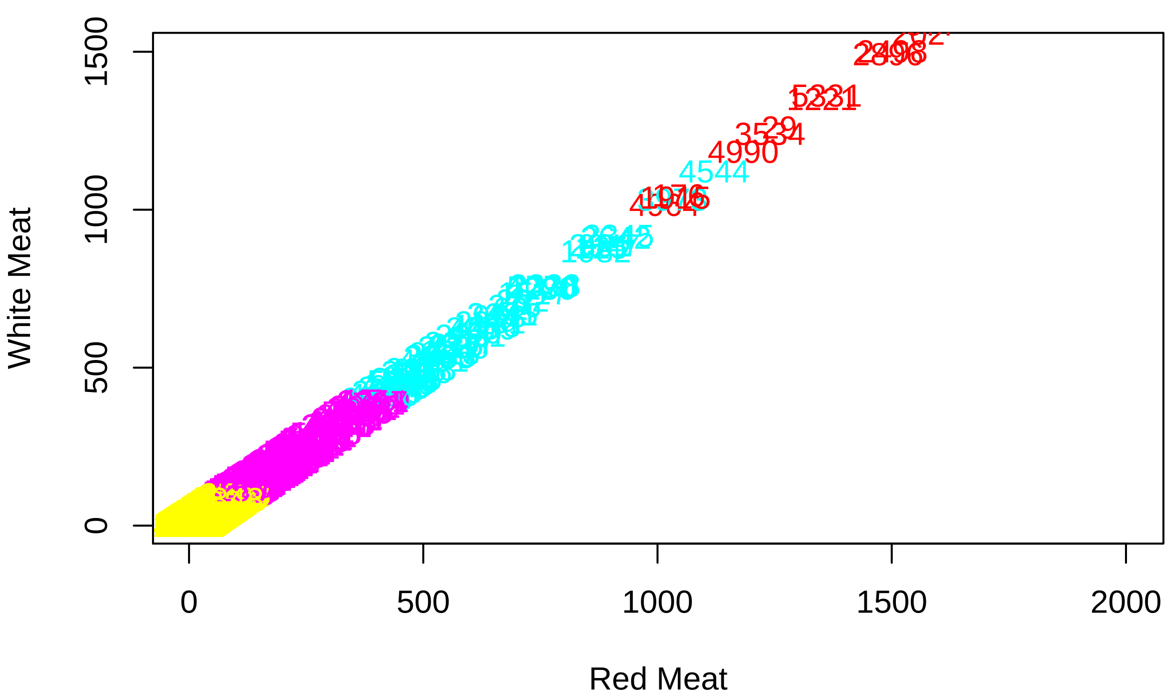



```r
#compute k-means clustering with k=5
fit.new<-kmeans(user_cluster,5)
```

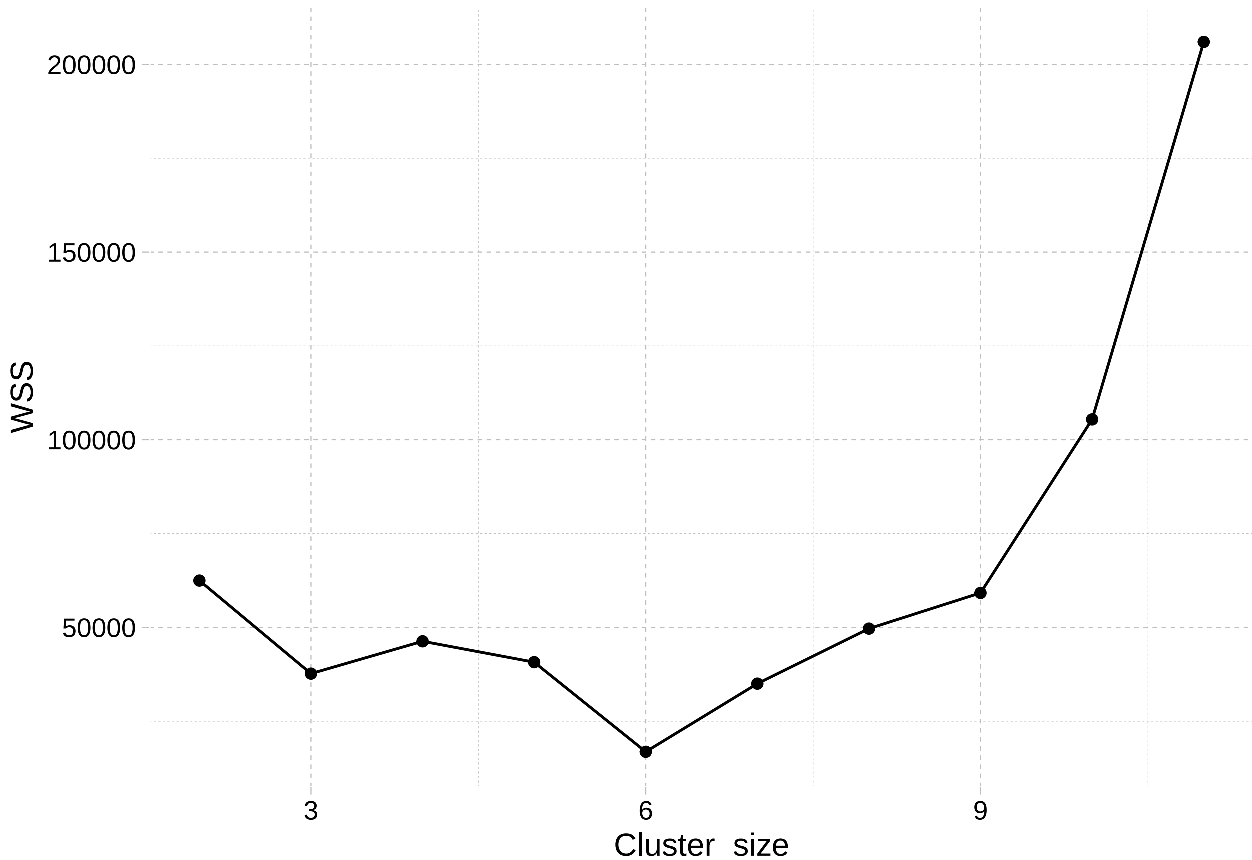
```r
#visual the results using fvia_cluster
fviz_cluster(fit.new,data=user_cluster)
```




```
plot(user_cluster_num$useful, user_cluster_num$funny, type="n", xlim=c(3,2000),ylim=c(3,1500), xlab="Rec
text(x=user_cluster_num$useful, y=user_cluster_num$useful, labels=user_cluster_num$name,col=grpMeat$clus
```

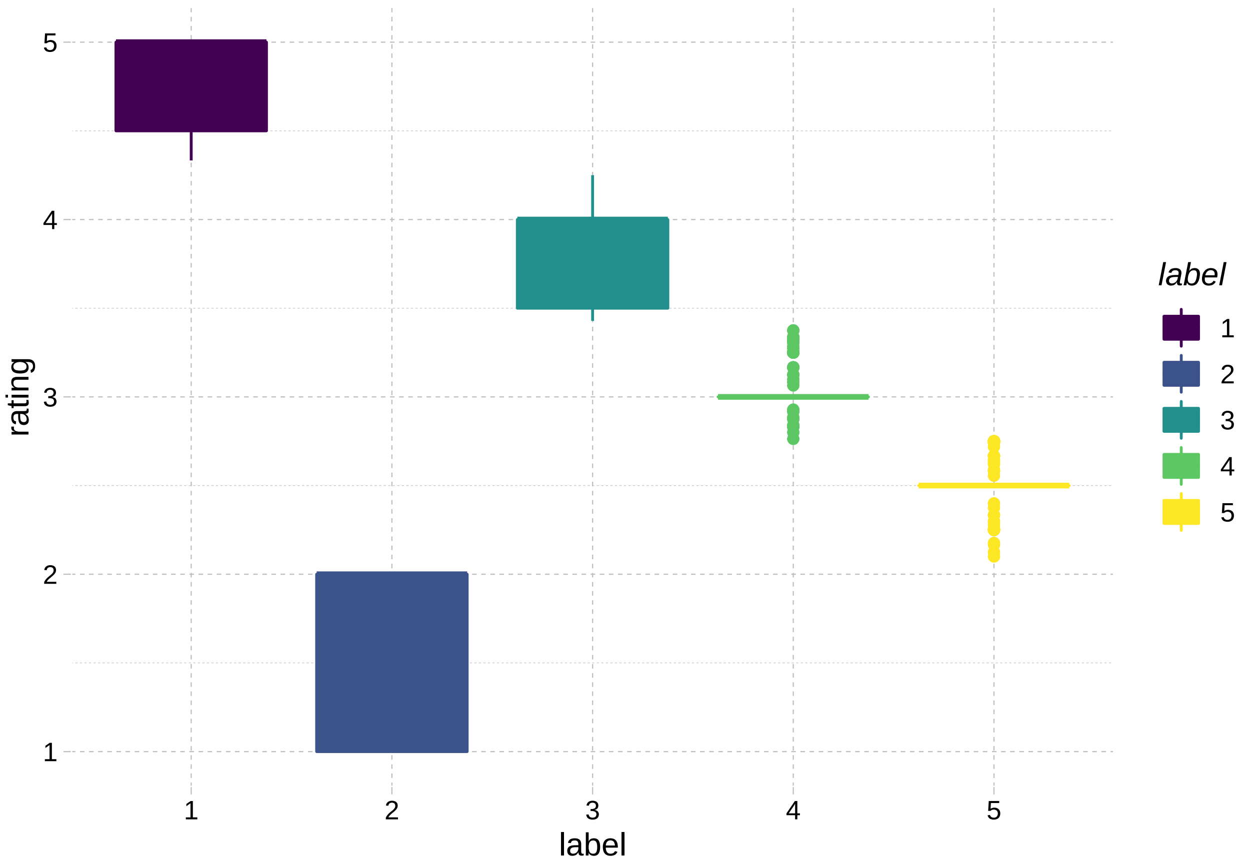


```r
business<-read.csv("business.csv")
star<-aggregate(business$stars, list(business$name), mean)
colnames(star)<-c("Name","Rating")
index<-sample(nrow(star),nrow(star)/2,replace = F)
#spilt data into train and test set
train<-data.frame(star$Rating)[index,]
test<-data.frame(star$Rating)[-index,]
#create empty vectors with 10 zeros
data<-double(10)
#try cluster sizes from 2 to 11
for (i in 2:11){
  kc <- kmeans(train,i, nstart = 50)
  centers <- kc$centers
#get prediction from test set
  assignments <- predict_KMeans(as.data.frame(test),kc$centers)
#calculate wss from test set
  wss=0
  k=1

  while (k<=i){
#perform 2 fold cross validation
    a<-assignments[assignments==k]
    wss <-wss+sum((a-centers[k])^2)
    k=k+1
  }
  data[i-1]<-wss
}

cluster<-seq(2,11)
data<-cbind(data,cluster)
data<-data.frame(data)
colnames(data)<-c("WSS","Cluster_size")
ggplot(data,aes(Cluster_size,WSS))+
  geom_line()+
  geom_point()+scale_fill_viridis_d(option = "viridis") +
 scale_color_viridis_d(option = "viridis") +
 theme_pander()
```




```r
# 5 clusters is the best

# predict
kc <- kmeans(train,5, nstart = 50)
centers <- kc$centers
assignments <- predict_KMeans(as.data.frame(test),kc$centers)

# get clusters
# first one
fir<-assignments[assignments==1]

# 5 clusters is the best

# predict
test<-data.frame(test)
k5 <- kmeans(train,5, nstart = 50)
centers <- k5$centers
predicted <- predict_KMeans(as.data.frame(test),centers)
predicted<-as.numeric(predicted)
test$label<-as.factor(predicted)
colnames(test)<-c('rating','label')
#get boxplot for each rating prediction
p<-ggplot(test, aes(x=label, y=rating, color=label,fill=label)) +
  geom_boxplot()+scale_fill_viridis_d(option = "viridis") +
 scale_color_viridis_d(option = "viridis") +
 theme_pander()
```

```
p
```



Topic Modeling

I create document term matrix for topic modeling for further LDA analysis. Latent Dirichlet Allocation helps to discover latent themes in the restaurant texts descriptions. We can see how the topics have been formed as a mixture of similar words from the same domain.

```
library(dplyr)
library(tidytext)
#convert text into document term matrix
text<-review$text
text<-as.data.frame(text)
text$text<-as.character(text$text)
#text_sample<-text[sample(nrow(text), 500), ]
#text_sample<-as.data.frame(text_sample)
#text_sample$text_sample<-as.character(text_sample$text_sample)
#tidy_text<-text%>%
 # unnest_tokens(word, text) %>%
 #  anti_join(stop_words) %>%
 # count(word)

#tidy_text[tidy_text==0] <- 0.001
#tidy_text<-tidy_text[-c(1:160),]
#tidy_text<-tidy_text%>%
```

```r
 # count(word)%>%
 #cast_sparse(word, n)
```

```r
tidy_text <-text %>%
  #break review text into individual tokens and tranfrom into a tidy data structure
  unnest_tokens(word, text)%>%
      anti_join(stop_words)%>%
  count(word)
tidy_text$id<-1
tidy_text<-tidy_text[-c(1:1543),]
```

```r
#convert to a DocumentTermMatrix object from tm
tidy_text<-tidy_text%>%
  count(word,id)%>%
  cast_dtm(id,word,n)
```

```r
#set a seed so that the output of the model is predictable
library(topicmodels)
lda<-LDA(tidy_text,k=2,control=list(seed=1234))
lda
```

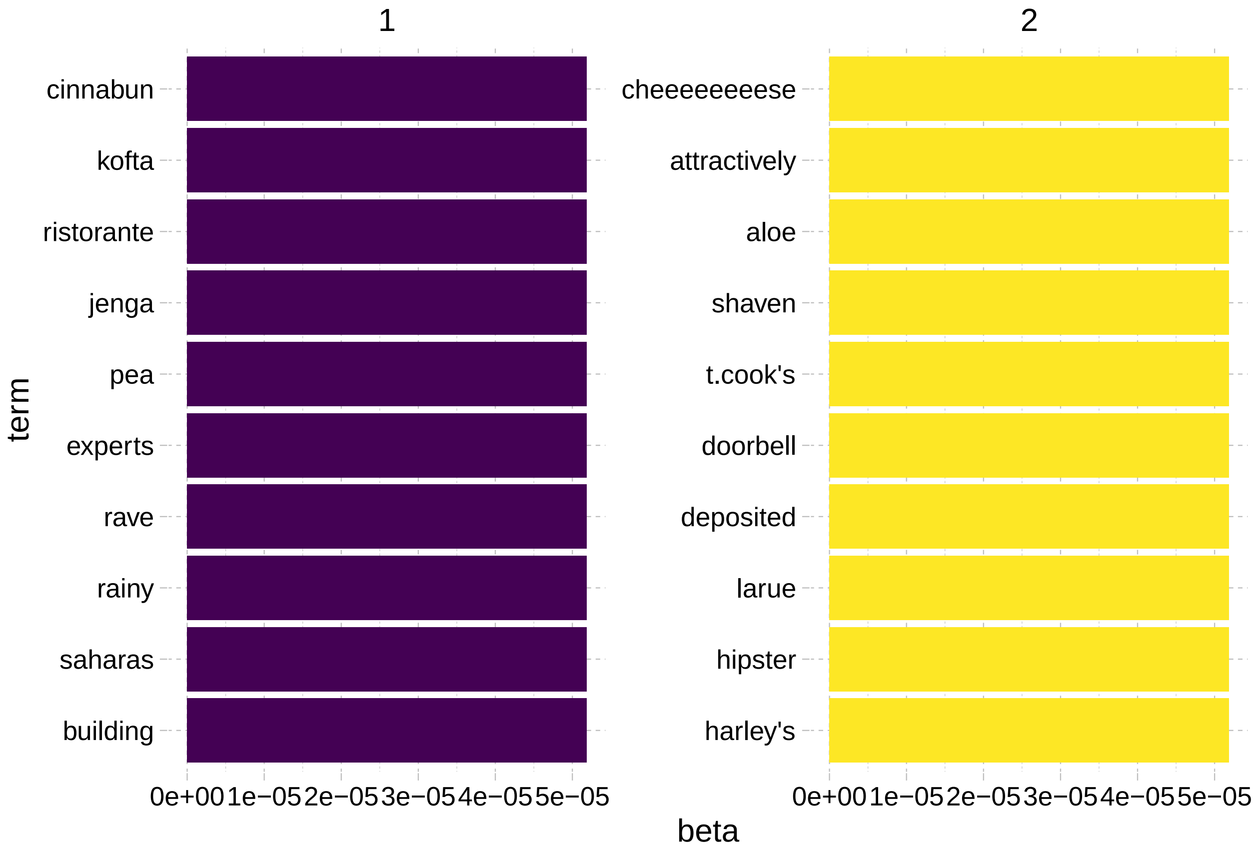
```
## A LDA_VEM topic model with 2 topics.
```

```r
#extracting the per-topic-per-word probabilities
library(tidytext)
ap_topics <- tidy(lda, matrix = "beta")
ap_topics
```

```
## # A tibble: 77,078 x 3
##    topic term             beta
##    <int> <chr>           <dbl>
##  1     1 aback      0.0000388
##  2     2 aback      0.0000131
##  3     1 abalone    0.0000322
##  4     2 abalone    0.0000197
##  5     1 abandoned  0.0000199
##  6     2 abandoned  0.0000320
##  7     1 abandoning 0.0000224
##  8     2 abandoning 0.0000295
##  9     1 abandonné  0.0000263
## 10     2 abandonné  0.0000256
## # ... with 77,068 more rows
```

```r
library(ggplot2)
library(dplyr)
library(ggthemes)
#find the 10 terms that are most common within each topic
ap_top_terms <- ap_topics %>%
      group_by(topic) %>%
      top_n(10, beta) %>%
      ungroup() %>%
```

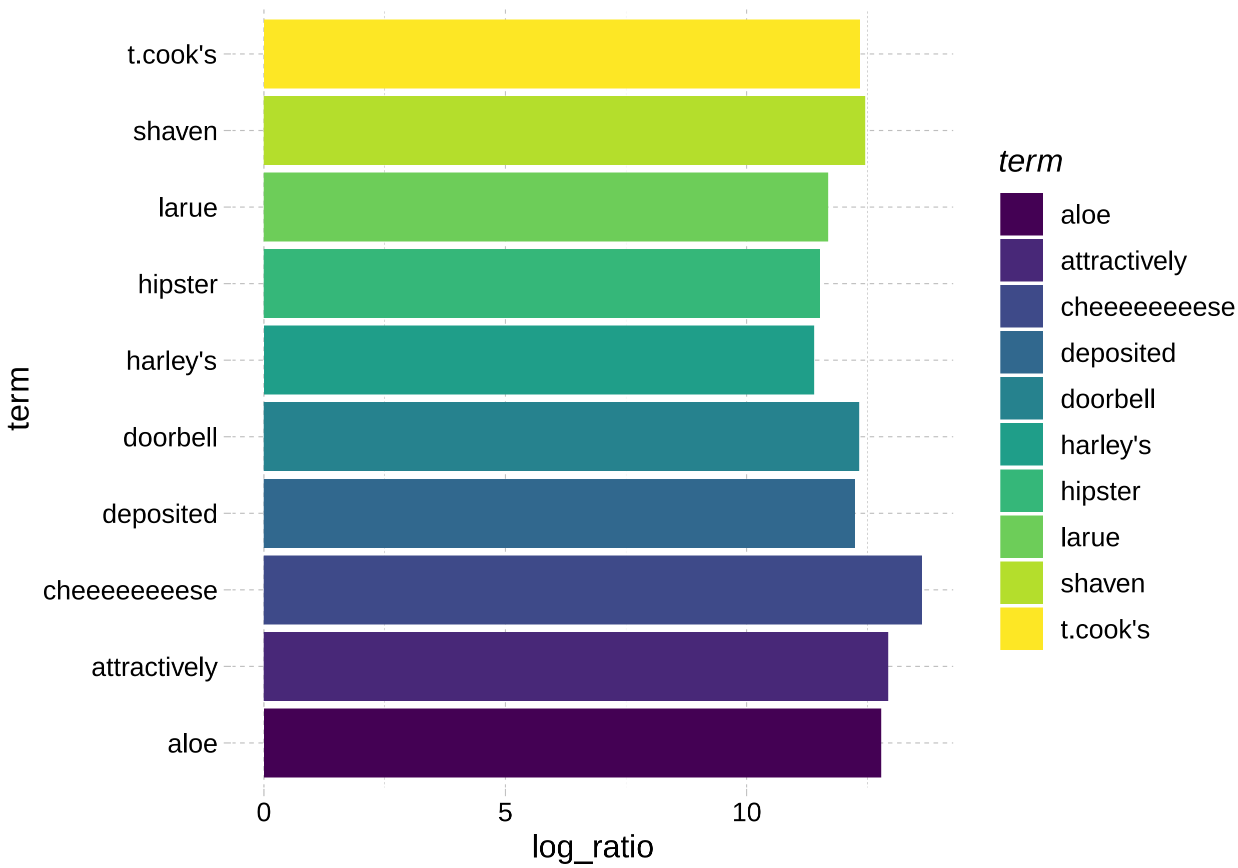
```
      arrange(topic, -beta)
ap_top_terms %>%
mutate(term = reorder(term, beta)) %>% ggplot(aes(term, beta, fill = factor(topic))) + geom_col(show.leg
facet_wrap(~ topic, scales = "free") + coord_flip()+
  scale_fill_viridis_d(option = "viridis") +
 scale_color_viridis_d(option = "viridis") +
  theme_pander()
```




```
#consider difference among topics
library(tidyr)
beta_spread <- ap_topics %>%
  mutate(topic = paste0("topic", topic)) %>%
  spread(topic, beta) %>%
  filter(topic1 > .00001 | topic2 > .00001) %>%
  mutate(log_ratio = log2(topic2 / topic1))
head(beta_spread)
```

```
## # A tibble: 6 x 4
##   term           topic1    topic2 log_ratio
##   <chr>           <dbl>     <dbl>     <dbl>
## 1 aback      0.0000388 0.0000131   -1.56
## 2 abalone    0.0000322 0.0000197   -0.704
## 3 abandoned  0.0000199 0.0000320    0.691
## 4 abandoning 0.0000224 0.0000295    0.398
## 5 abandonné  0.0000263 0.0000256   -0.0362
## 6 abay       0.0000246 0.0000273    0.152
```

```r
#filter for relatively common words that have a beta great than 0.00001
library(magrittr)
beta_spread %>%top_n(10, log_ratio) %>%ggplot(aes(term, log_ratio, fill = term)) + geom_col(show.legend
 coord_flip()+
  scale_fill_viridis_d(option = "viridis") +
 scale_color_viridis_d(option = "viridis") +
  theme_pander()
```



PCA

Principal Component Analysis is used to improve the prediction of star rating.

```r
user<-read.csv("user.csv")
user$X<-NULL
user$user_id<-NULL
user$name<-NULL
user$yelping_since<-NULL
user$friends<-NULL
user$elite<-NULL
user<-na.omit(user)
mod1<-princomp(na.omit(user),cor = T)
# take a look a out cumulative proporation
summary(mod1)
```

```
## Importance of components:
##                           Comp.1     Comp.2     Comp.3     Comp.4     Comp.5
## Standard deviation     3.492207 1.22037489 0.99997364 0.85009812 0.65830153
## Proportion of Variance 0.717383 0.08760676 0.05882043 0.04250981 0.02549182
## Cumulative Proportion  0.717383 0.80498974 0.86381017 0.90631998 0.93181180
##                            Comp.6    Comp.7     Comp.8      Comp.9     Comp.10
## Standard deviation     0.52517164 0.4832220 0.45705925 0.364621106 0.322951278
## Proportion of Variance 0.01622384 0.0137355 0.01228842 0.007820503 0.006135149
## Cumulative Proportion  0.94803564 0.9617711 0.97405956 0.981880067 0.988015215
##                            Comp.11     Comp.12     Comp.13    Comp.14
## Standard deviation     0.261897329 0.226054235 0.209964452 0.13748131
## Proportion of Variance 0.004034718 0.003005913 0.002593239 0.00111183
## Cumulative Proportion  0.992049934 0.995055846 0.997649086 0.99876092
##                             Comp.15      Comp.16      Comp.17
## Standard deviation     0.1214069391 0.0795285158 1.862645e-09
## Proportion of Variance 0.0008670379 0.0003720462 2.040851e-19
## Cumulative Proportion  0.9996279538 1.0000000000 1.000000e+00
```

```r
# take first sixth pcs
predictors<-mod1$scores[,3]
data<-data.frame(predictors)
data$star<-user$average_stars
mo1<-lm(star~.,data=data)
# use loocv
summary(mo1)
```

```
##
## Call:
## lm(formula = star ~ ., data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.4761 -0.0031 -0.0013  0.0006  2.2748
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.6775770  0.0003466   10611   <2e-16 ***
## predictors  1.1476862  0.0003466    3311   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04901 on 19998 degrees of freedom
## Multiple R-squared:  0.9982, Adjusted R-squared:  0.9982
## F-statistic: 1.096e+07 on 1 and 19998 DF,  p-value: < 2.2e-16
```

```r
mod2<-lm(user$average_stars~.,data=user)
```

Conclusion

By working with data analysis, I found that our data has large limitations using yelp API. Even though I used the API to extract the data from yelp official website, there are not a lot available data that I can use. As a result, I just downloaded the data from yelp website. By implementing the method of cluster analysis(k-means) and PCA. It is possible to be able to predict a restaurant's rating result based on certain relevant variables that are already presented in the cluster analysis and Principle Component Analysis.