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.

```
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
```

```
library(yelpr)
api<-"GBORIlecmrZGAjvMJLmknx0F9dbCOoysGYU9LlhPSf4aSH54R76cwHBhM_d72a8jOp4iejBoCauyXDsDBbEO8zGsG6puTZXft
```

```
#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.
```

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

```
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()
```

```
#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
```

```
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
```

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

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

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

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

```
#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

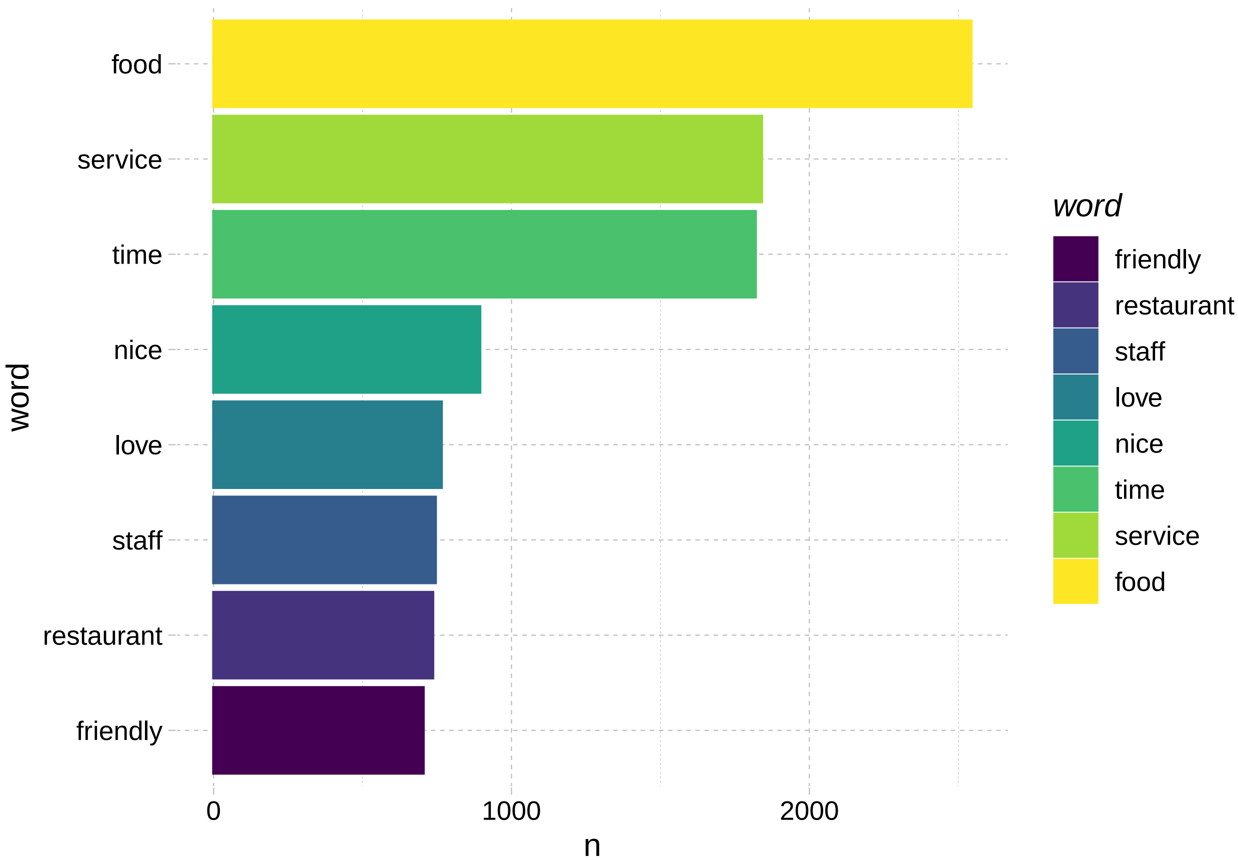
```
#most common words in the review by table
word_counts <-tidy_word_com %>% anti_join(stop_words, by="word")%>% count(word, sort = TRUE)
head(word_counts)
```

```
## # A tibble: 6 x 2
##   word        n
##   <chr>   <int>
## 1 food     2543
## 2 service  1840
## 3 time     1819
## 4 nice      894
## 5 love      765
## 6 staff     745
```

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

word	n
food	2543
service	1840
time	1819
nice	894
love	765
staff	745

```
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

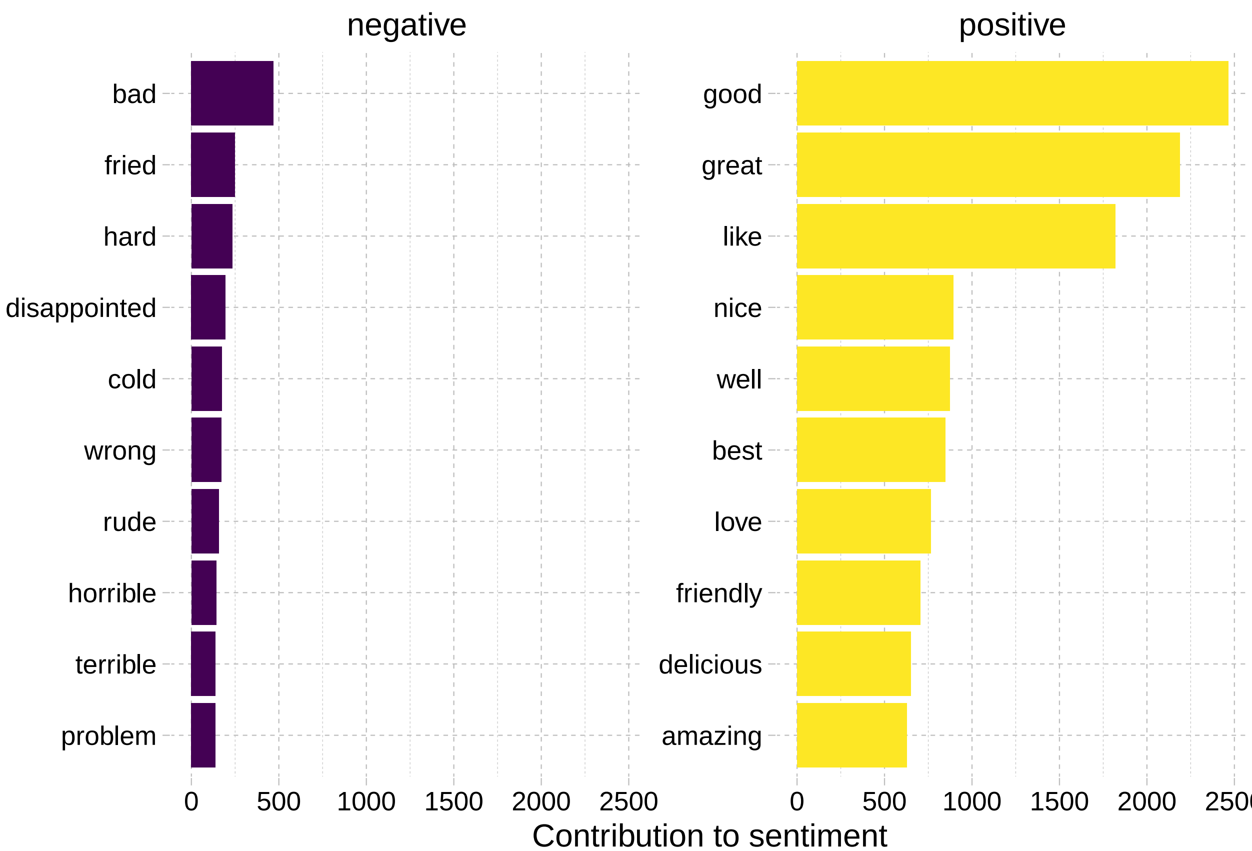
After process the opinion of restaurants computationally for identifying and categorizing, we are able to determine the attitude of the customers or say the customer towards the certain restaurtants is negative, positive or netural. I used bing to get sentiment count by doing single word and bigram analysis.

```
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	2464
great	positive	2187
like	positive	1818
nice	positive	894
well	positive	873
best	positive	847

```
#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 -1 star for the 2013 Ford Escape and -1 star for the lack of quality control after a major repair.
## 2 -1 star for the 2013 Ford Escape and -1 star for the lack of quality control after a major repair.
## 3 -1 star for the 2013 Ford Escape and -1 star for the lack of quality control after a major repair.
## 4 -1 star for the 2013 Ford Escape and -1 star for the lack of quality control after a major repair.
## 5 -1 star for the 2013 Ford Escape and -1 star for the lack of quality control after a major repair.
## 6 -1 star for the 2013 Ford Escape and -1 star for the lack of quality control after a major repair.
##        bigram
## 1      1 star
## 2    star for
## 3     for the
## 4    the 2013
## 5   2013 ford
## 6 ford escape
```

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

```
## # A tibble: 198,581 x 2
```

```
##    bigram         n
##    <chr>      <int>
##  1 of the      1764
##  2 it was      1713
##  3 and the     1554
##  4 in the      1544
##  5 i was       1259
##  6 this place  1254
##  7 on the      1138
##  8 and i       1109
##  9 for the      938
## 10 the food     914
## # ... with 198,571 more rows
```

```r
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: 46,120 x 3
##    word1    word2         n
##    <chr>    <chr>     <int>
##  1 customer service     259
##  2 highly   recommend   174
##  3 ice      cream       159
##  4 las      vegas       129
##  5 5        stars        80
##  6 happy    hour         73
##  7 10       minutes      62
##  8 15       minutes      60
##  9 20       minutes      58
## 10 front    desk         55
## # ... with 46,110 more rows
```

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

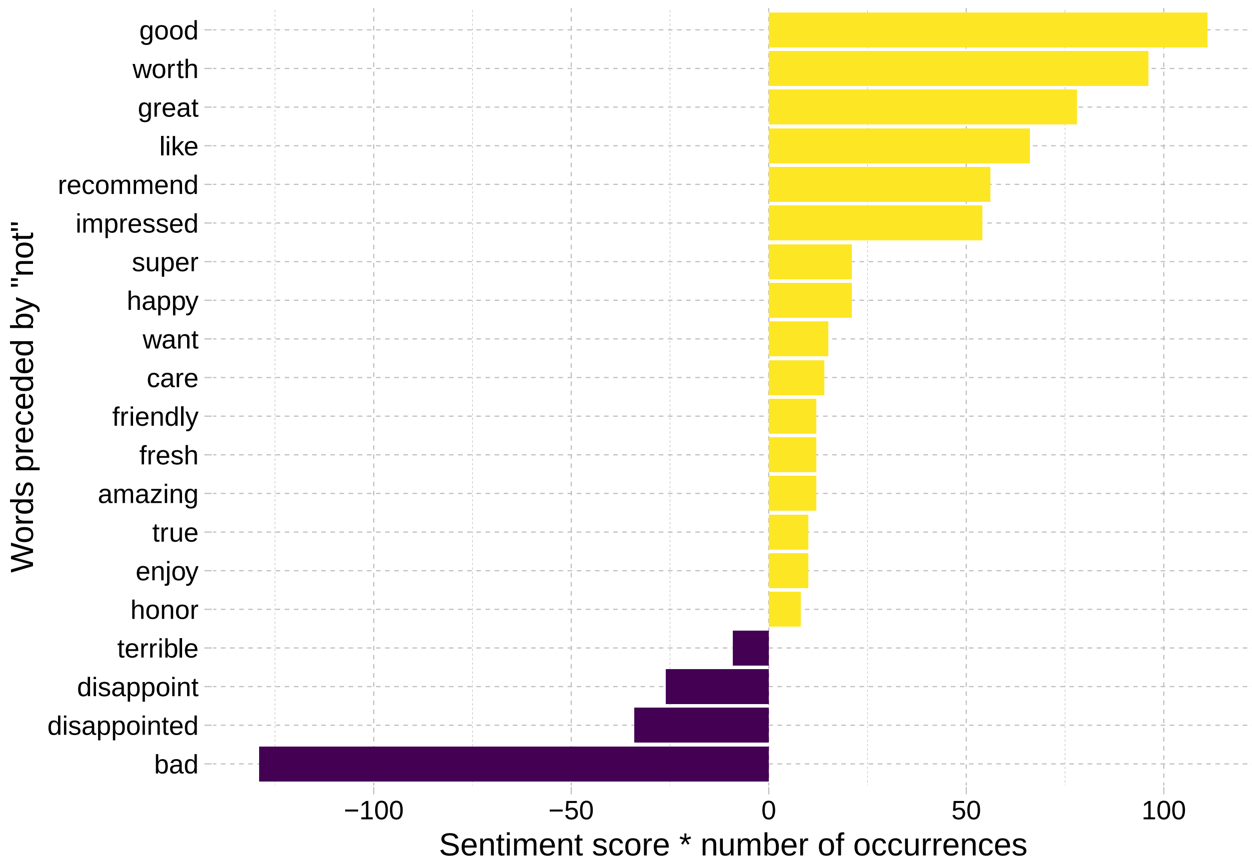
```
## # A tibble: 785 x 3
##    word1 word2     n
##    <chr> <chr> <int>
##  1 not   a       196
##  2 not   the     139
##  3 not   be      102
##  4 not   sure    100
```

```
##   5 not   too      78
##   6 not   to       74
##   7 not   even     69
##   8 not   only     69
##   9 not   have     62
##  10 not   worth    48
## # ... with 775 more rows
```

```r
#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: 135 x 3
##    word2        value     n
##    <chr>        <dbl> <int>
##  1 worth            2    48
##  2 bad             -3    43
##  3 good             3    37
##  4 like             2    33
##  5 recommend        2    28
##  6 great            3    26
##  7 impressed        3    18
##  8 disappointed    -2    17
##  9 want             1    15
## 10 disappoint      -2    13
## # ... with 125 more rows
```

```r
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
```

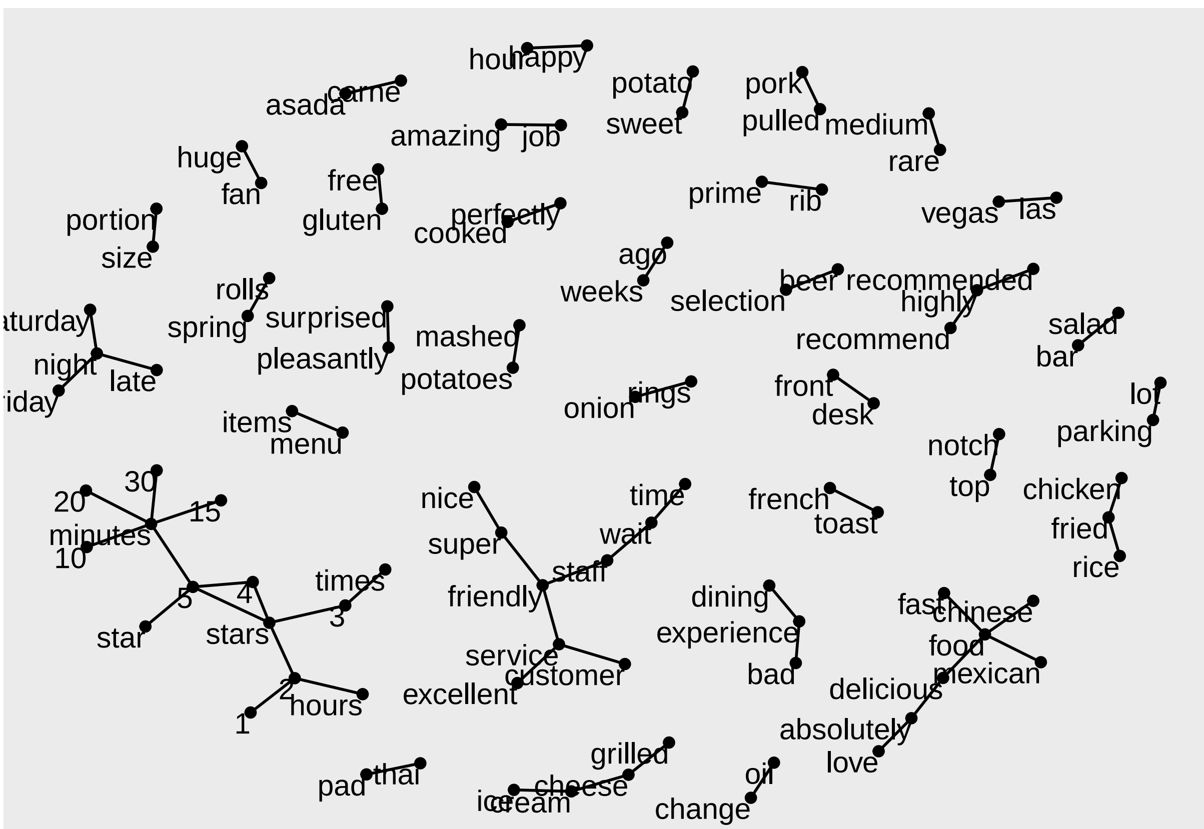
```
## # A tibble: 46,120 x 3
##    word1    word2         n
##    <chr>    <chr>     <int>
##  1 customer service     259
##  2 highly   recommend   174
##  3 ice      cream       159
##  4 las      vegas       129
##  5 5        stars        80
##  6 happy    hour         73
##  7 10       minutes      62
##  8 15       minutes      60
##  9 20       minutes      58
## 10 front    desk         55
## # ... with 46,110 more rows
```

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

```
## IGRAPH 1711523 DN-- 99 66 --
```

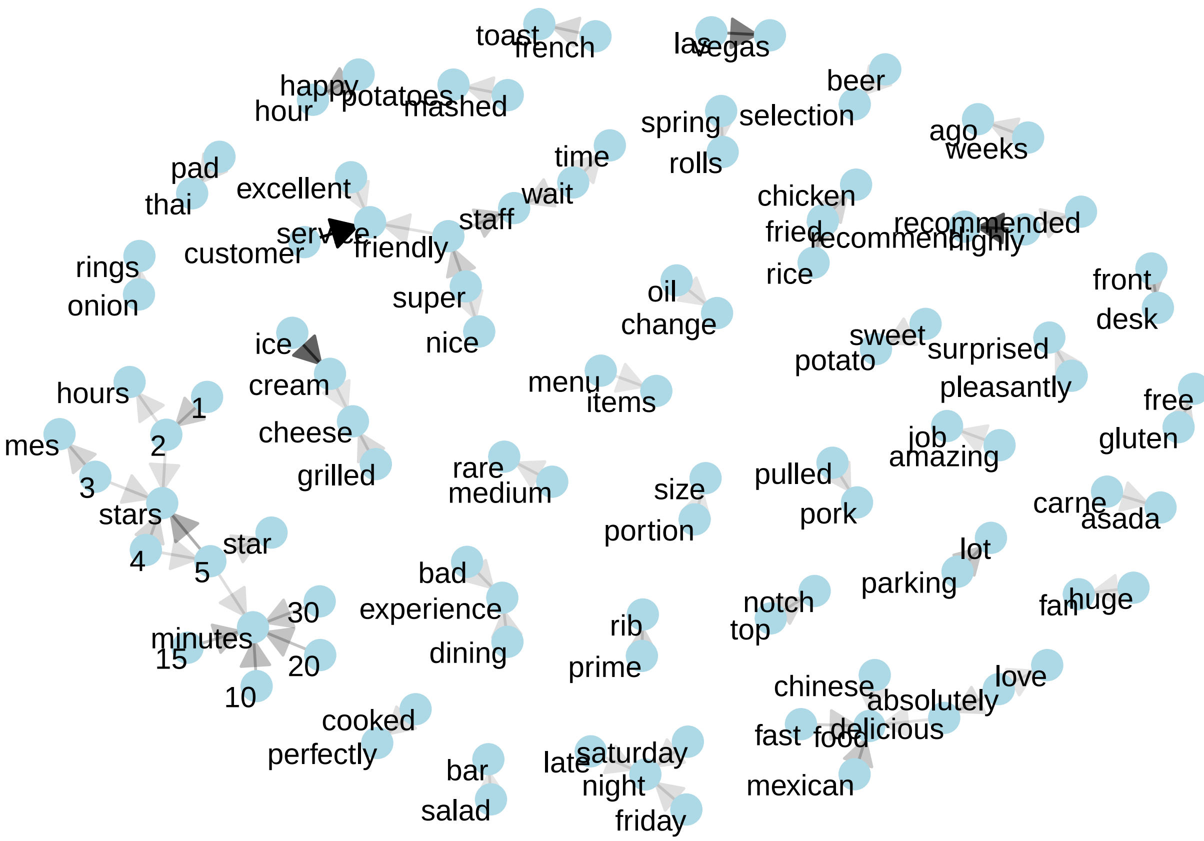
```
## + attr: name (v/c), n (e/n)
## + edges from 1711523 (vertex names):
##  [1] customer->service   highly  ->recommend ice     ->cream
##  [4] las     ->vegas     5       ->stars     happy   ->hour
##  [7] 10      ->minutes   15      ->minutes   20      ->minutes
## [10] front   ->desk      parking ->lot       mexican ->food
## [13] fried   ->rice      friendly->staff     1       ->2
## [16] 30      ->minutes   super   ->friendly  fast    ->food
## [19] 3       ->times     4       ->stars     fried   ->chicken
## [22] prime   ->rib       top     ->notch     french  ->toast
## + ... omitted several edges
```

```r
#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)
```




```r
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,
```

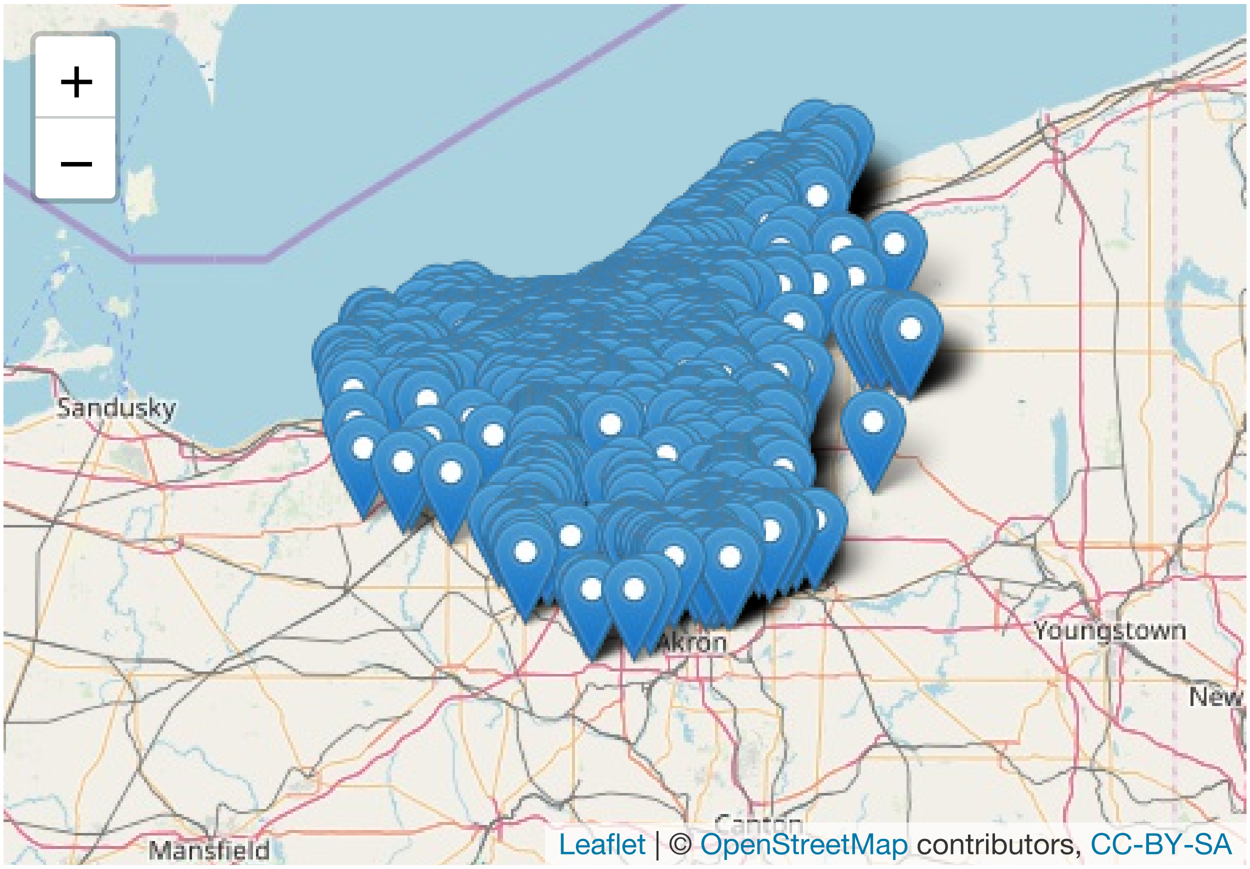
```
                          arrow = a, end_cap = circle(.07, 'inches')) +
      geom_node_point(color = "lightblue", size = 5) +
      geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
# add theme that is useful for plotting networks
      theme_void()
```



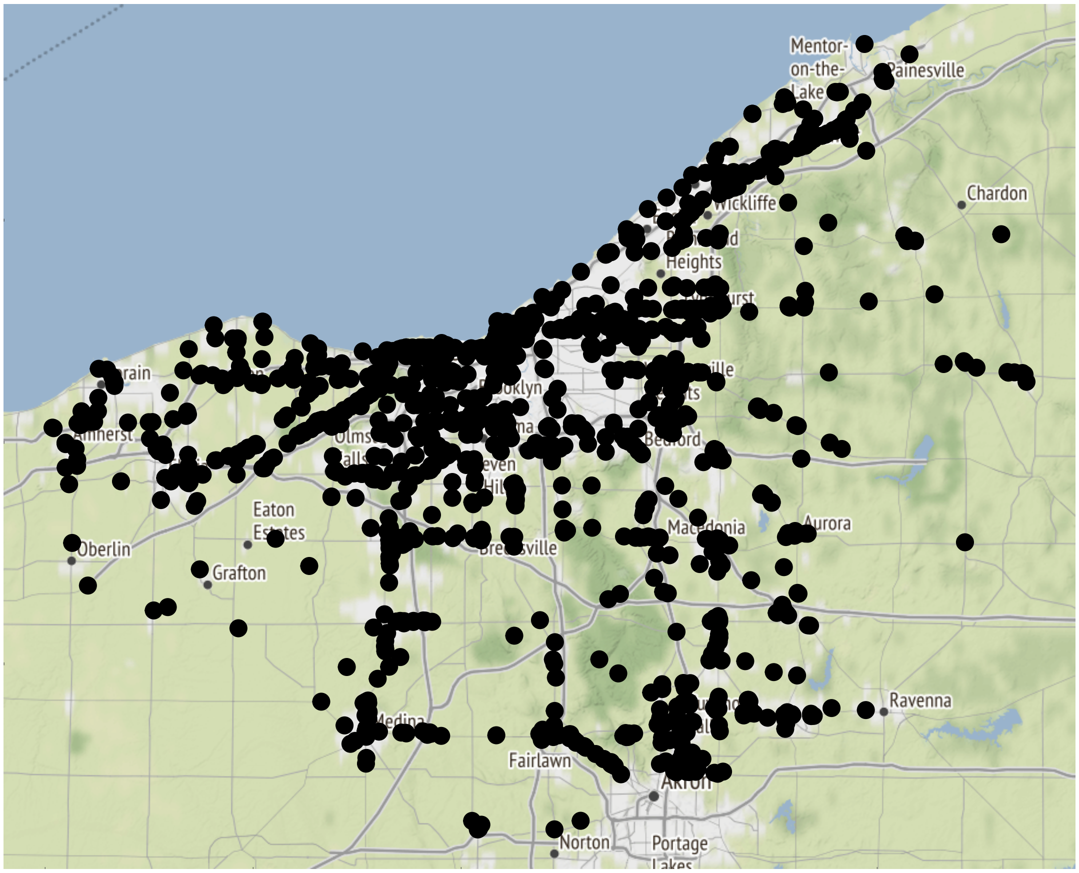
Mapping for yelp restaurants

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")
```



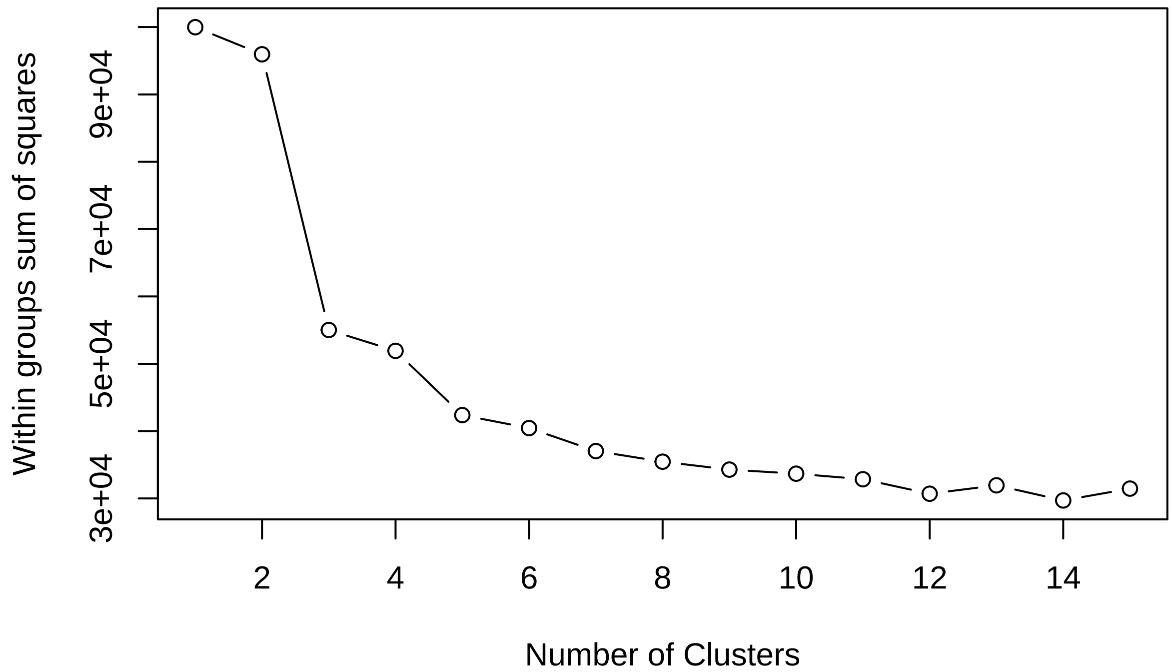

```r
#distribution of restaurants in OH using ggmap
library(ggmap)
m6<-qmplot(long, Latitude, data = Latitude,
           color = I("black"), size = I(2.5))
m6
```



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")
```



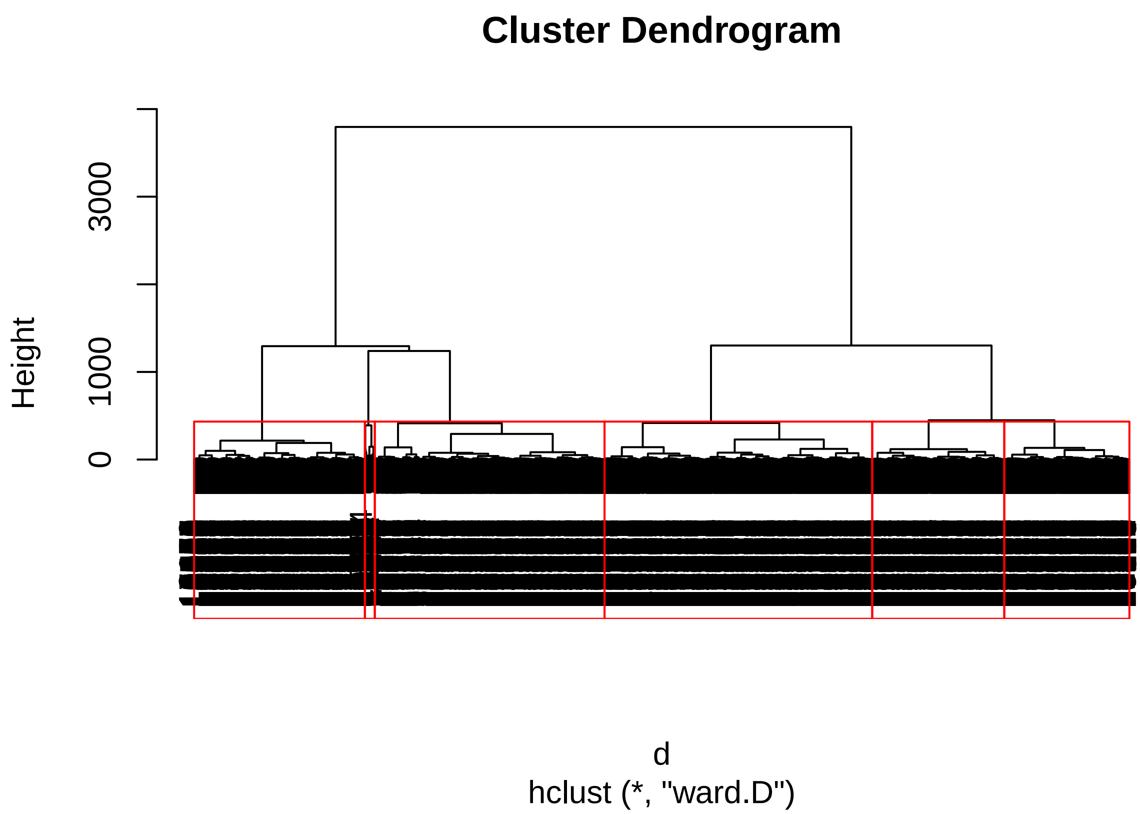

```r
# K-Means Cluster Analysis
fit <- kmeans(user_cluster, 6) # 5 cluster solution
# get cluster means
aggregate(user_cluster,by=list(fit$cluster),FUN=mean)
```

```
##   Group.1      user_id        name review_count yelping_since     useful
## 1       1 -0.027908891  1.05742333   -0.1824782     0.1033115 -0.1475937
## 2       2 -0.065354264 -0.79334628   -0.2175776     0.6091781 -0.1671749
## 3       3 -0.113269457  0.11982648    6.2154607    -1.3351940  5.7597378
## 4       4  0.108095084 -0.36169101    0.4058489    -1.1209197  0.1805829
## 5       5 -0.001639103  0.01642188   -0.2757653     0.4998499 -0.1716613
## 6       6  0.286631846  0.60551204    7.7745895    -1.0868555 16.7646823
##         funny        cool       fans average_stars compliment_hot
## 1 -0.12318949 -0.11348994 -0.1334720    0.44283905   -0.076125475
## 2 -0.13025862 -0.12087506 -0.1448229    0.55371377   -0.077782035
## 3  5.48329524  5.22638116  6.3942069    0.11560070    3.719746371
## 4  0.09016764  0.06789941  0.1294029    0.07534029   -0.007556152
## 5 -0.13036256 -0.13015286 -0.1608363   -1.63377856   -0.079988774
## 6 16.62368470 17.78816437 14.2607810    0.08621806   17.026257777
##   compliment_more compliment_profile compliment_cute compliment_list
## 1      -0.1274648        -0.08972229     -0.08171985     -0.05525578
## 2      -0.1362678        -0.09032407     -0.08918378     -0.05638890
## 3       5.3557015         5.17652089      3.57496596      2.85196883
## 4       0.1320351         0.01093560      0.03853847     -0.02350800
## 5      -0.1451579        -0.09421439     -0.09004943     -0.05638890
## 6      14.5459881        13.70486341     14.99759726     13.98682417
##   compliment_note compliment_plain compliment_cool compliment_funny
## 1     -0.09155264      -0.06193990    -0.089372343     -0.089372343
## 2     -0.10015364      -0.06557496    -0.092470949     -0.092470949
## 3      3.38887014       2.91161186     4.419741540      4.419741540
## 4      0.05239708      -0.00496796     0.002321849      0.002321849
## 5     -0.10003570      -0.06714864    -0.094738670     -0.094738670
```

```
## 6     18.16057467      14.74963515    18.489242483     18.489242483
##   compliment_writer compliment_photos
## 1       -0.10383311       -0.07009072
## 2       -0.11082466       -0.07953300
## 3        4.78885243        3.44861509
## 4        0.04541138        0.01178702
## 5       -0.11442680       -0.08487371
## 6       18.09445643       15.35823256
```

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

```r
# 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")
```

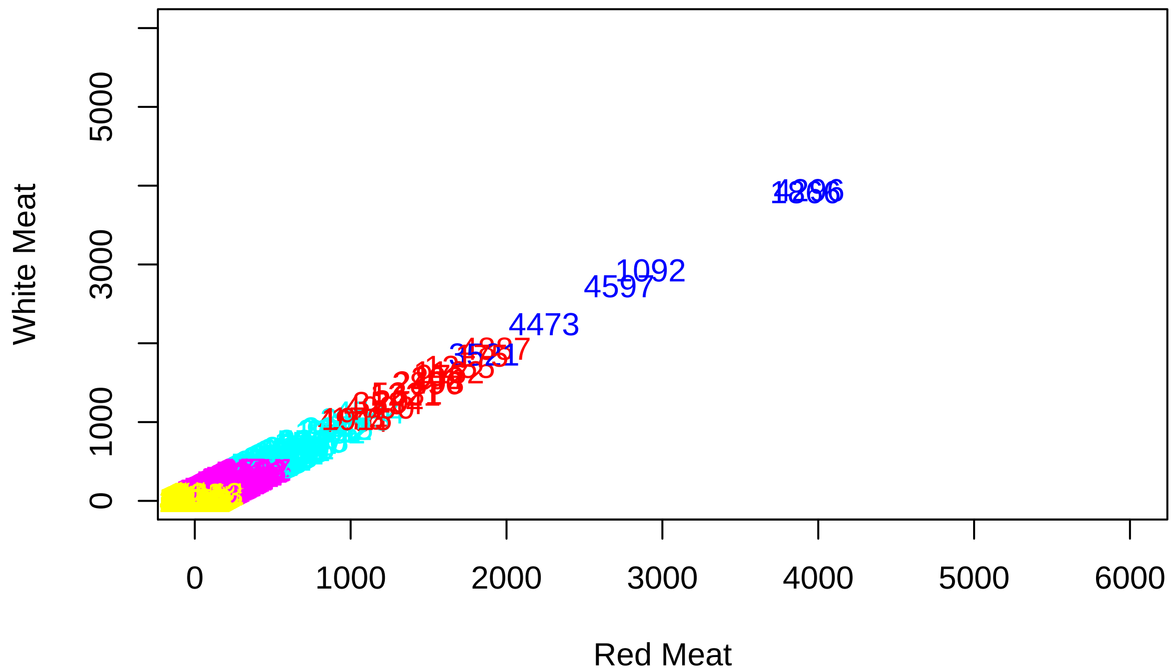



```r
## list of cluster assignments
o=order(grpMeat$cluster)
head(data.frame(user_cluster_num$name[o],grpMeat$cluster[o]))
```

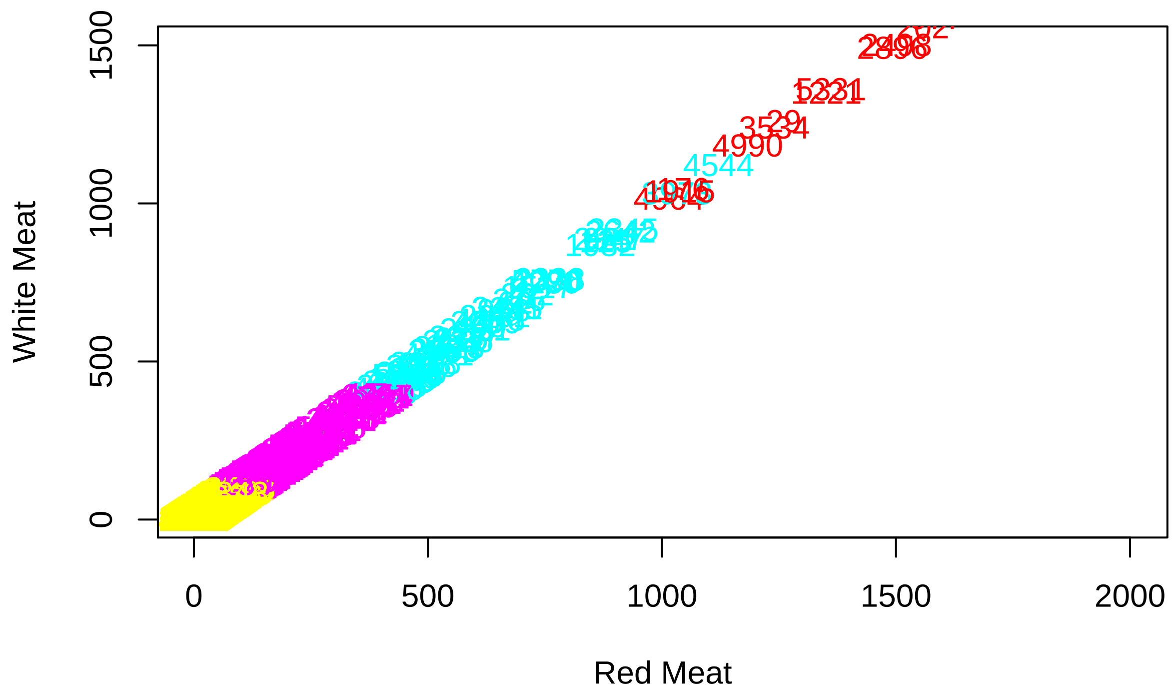
```
##       user_cluster_num.name.o. grpMeat.cluster.o.
## 11405                     4887                  1
## 13243                     4964                  1
```

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

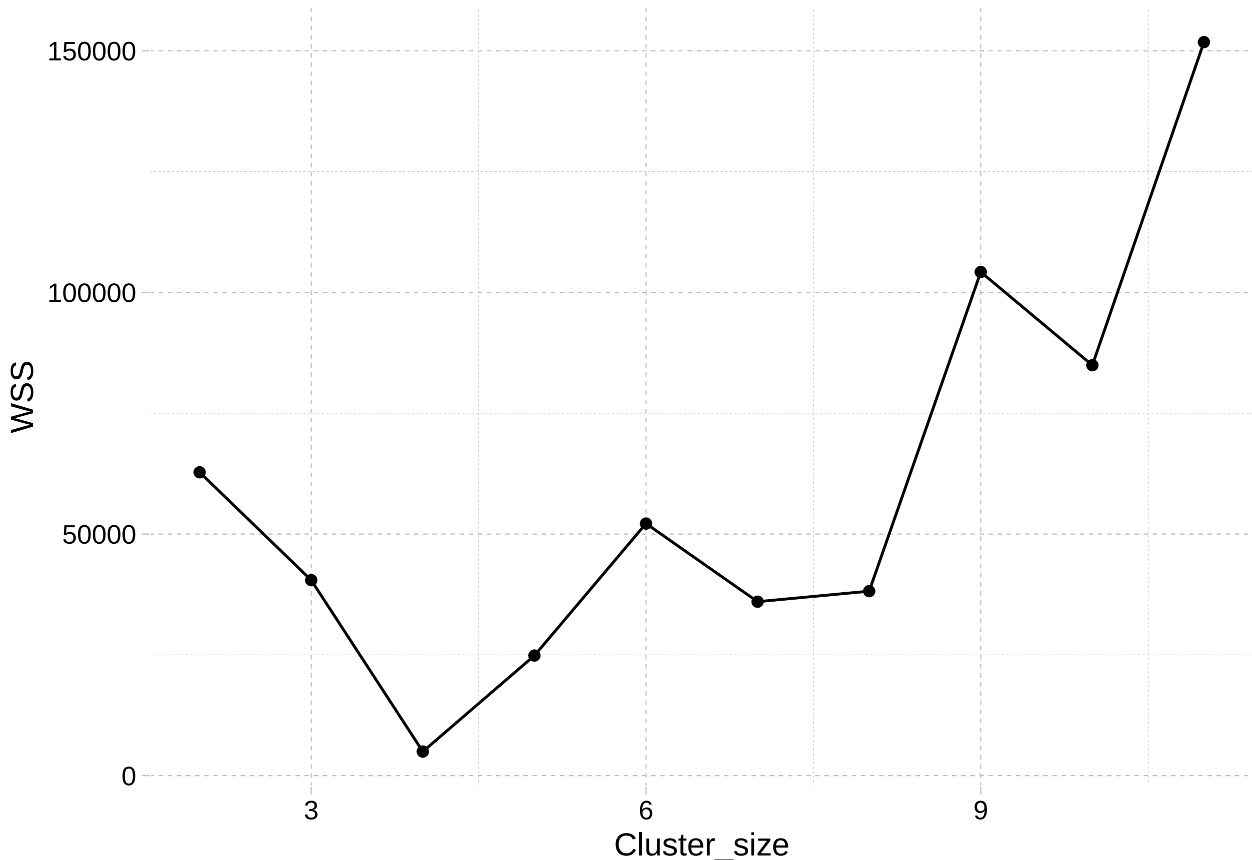
```
plot(user_cluster_num$useful, user_cluster_num$funny, type="n", xlim=c(3,6000),ylim=c(3,6000), xlab="Re
text(x=user_cluster_num$useful, y=user_cluster_num$useful, labels=user_cluster_num$name,col=grpMeat$clu
```




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

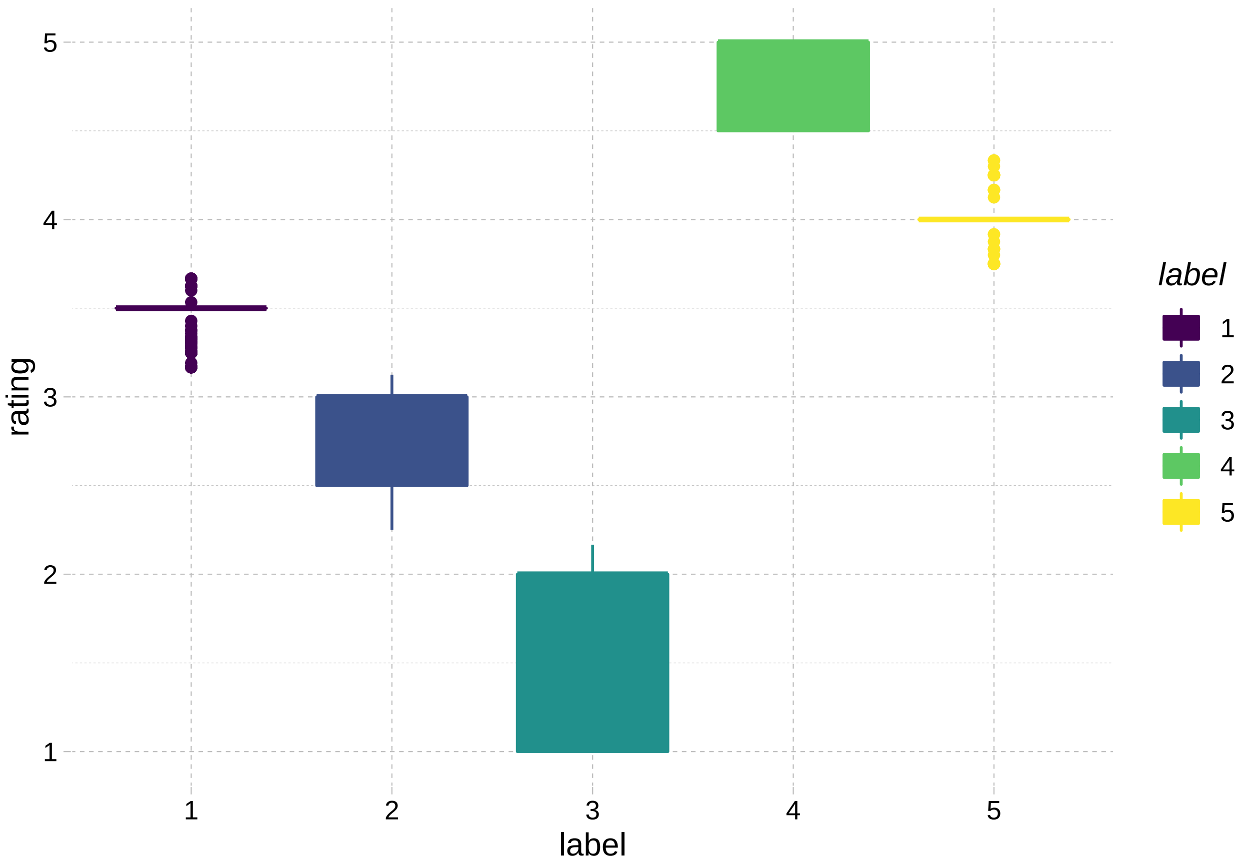


```r
business<-read.csv("business.csv")
star<-aggregate(business$stars, list(business$name), mean)
colnames(star)<-c("Name","Rating")
index<-sample(nrow(star),nrow(star)/2,replace = F)
#spilt data into train and test set
train<-data.frame(star$Rating)[index,]
test<-data.frame(star$Rating)[-index,]
#create 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
```

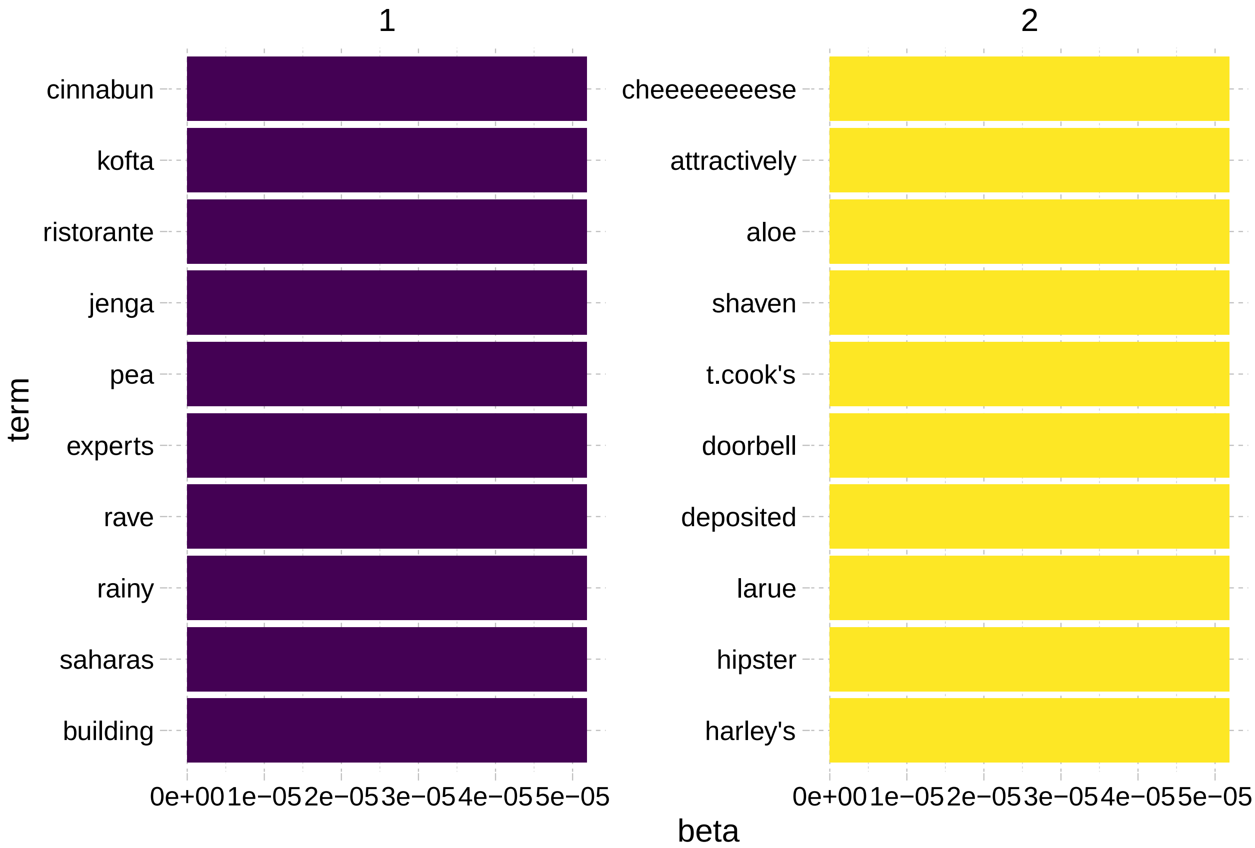
```
## A LDA_VEM topic model with 2 topics.
```

```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() %>%
```

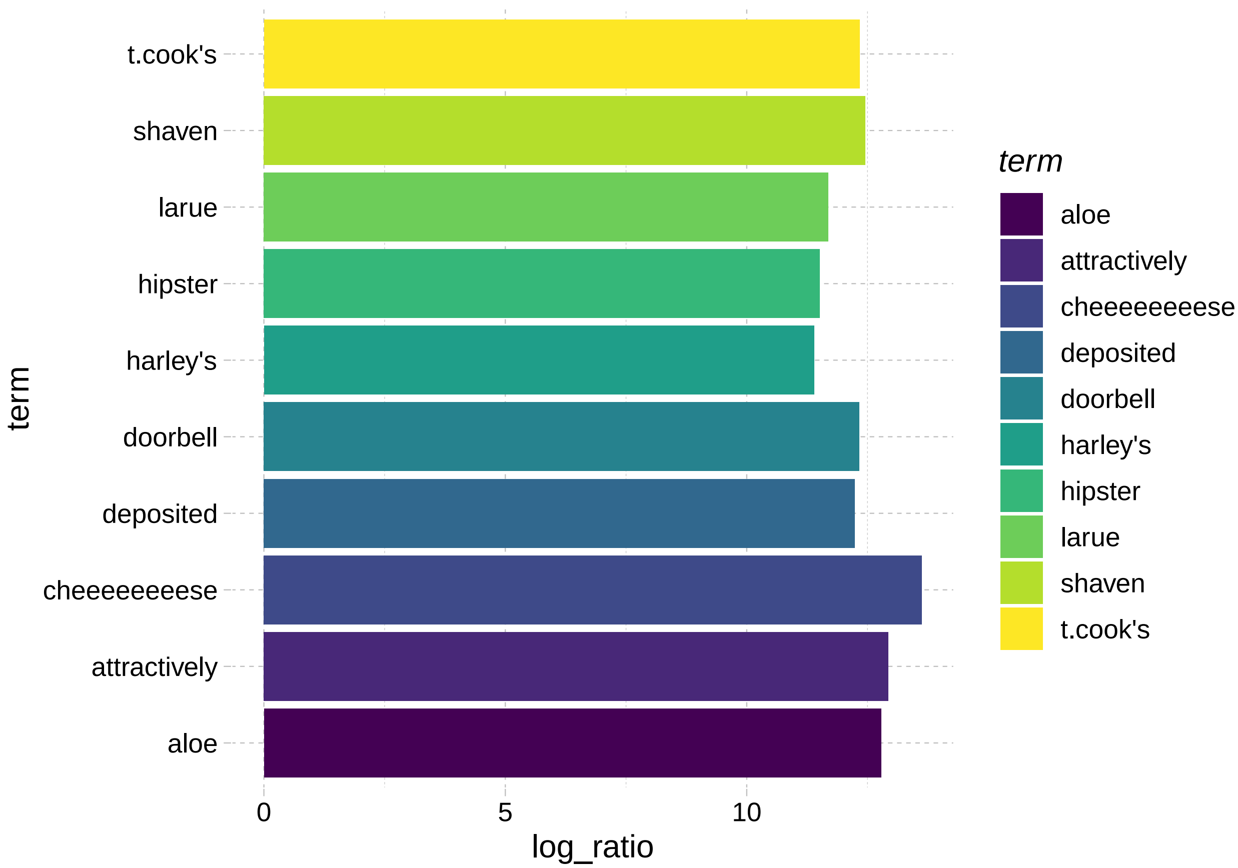
```r
      arrange(topic, -beta)
ap_top_terms %>%
mutate(term = reorder(term, beta)) %>% ggplot(aes(term, beta, fill = factor(topic))) + geom_col(show.leg
facet_wrap(~ topic, scales = "free") + coord_flip()+
  scale_fill_viridis_d(option = "viridis") +
 scale_color_viridis_d(option = "viridis") +
  theme_pander()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




```r
#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
```

```
#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.

```
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
```

```
# 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
```

```
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.