TripAdvisor Travel Reviews Analysis

Muhammad Maaz Khan

2025-08-16

First, we load the csv file into a dataframe (reviews) and look at it's structure:

```
reviews <- read.csv("C:/Users/maazk/OneDrive/Documents/tripadvisor_review.csv")
str(reviews)
```

```
## 'data.frame':    980 obs. of  11 variables:
##  $ User.ID    : chr  "User 1" "User 2" "User 3" "User 4" ...
##  $ Category.1 : num  0.93 1.02 1.22 0.45 0.51 0.99 0.9 0.74 1.12 0.7 ...
##  $ Category.2 : num  1.8 2.2 0.8 1.8 1.2 1.28 1.36 1.4 1.76 1.36 ...
##  $ Category.3 : num  2.29 2.66 0.54 0.29 1.18 0.72 0.26 0.22 1.04 0.22 ...
##  $ Category.4 : num  0.62 0.64 0.53 0.57 0.57 0.27 0.32 0.41 0.64 0.26 ...
##  $ Category.5 : num  0.8 1.42 0.24 0.46 1.54 0.74 0.86 0.82 0.82 1.5 ...
##  $ Category.6 : num  2.42 3.18 1.54 1.52 2.02 1.26 1.58 1.5 2.14 1.54 ...
##  $ Category.7 : num  3.19 3.21 3.18 3.18 3.18 3.17 3.17 3.17 3.18 3.17 ...
##  $ Category.8 : num  2.79 2.63 2.8 2.96 2.78 2.89 2.66 2.81 2.79 2.82 ...
##  $ Category.9 : num  1.82 1.86 1.31 1.57 1.18 1.66 1.22 1.54 1.41 2.24 ...
##  $ Category.10: num  2.42 2.32 2.5 2.86 2.54 3.66 3.22 2.88 2.54 3.12 ...
```

```
head(reviews)
```

```
##   User.ID Category.1 Category.2 Category.3 Category.4 Category.5 Category.6
## 1  User 1       0.93       1.80       2.29       0.62       0.80       2.42
## 2  User 2       1.02       2.20       2.66       0.64       1.42       3.18
## 3  User 3       1.22       0.80       0.54       0.53       0.24       1.54
## 4  User 4       0.45       1.80       0.29       0.57       0.46       1.52
## 5  User 5       0.51       1.20       1.18       0.57       1.54       2.02
## 6  User 6       0.99       1.28       0.72       0.27       0.74       1.26
##   Category.7 Category.8 Category.9 Category.10
## 1       3.19       2.79       1.82        2.42
## 2       3.21       2.63       1.86        2.32
## 3       3.18       2.80       1.31        2.50
## 4       3.18       2.96       1.57        2.86
## 5       3.18       2.78       1.18        2.54
## 6       3.17       2.89       1.66        3.66
```

By observing it's structure and first 6 rows, we can see that the data is in the wide table format. We will want to pivot longer to make our analysis easier, and we also want to rename the categories to what they actually represent.

We'll also want to check if there is any missing data, and impute the missing data if required.

Check for missing data:

```r
sum(is.na(reviews))
```

```
## [1] 0
```

The sum of missing values in the dataframe is 0, so there is no missing data.
We can move onto the next step of renaming the columns to their respective categories. The category types are defined in the webpage where this dataset was downloaded.

```r
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.4.2

## Warning: package 'tidyr' was built under R version 4.4.2

## Warning: package 'readr' was built under R version 4.4.2

## Warning: package 'purrr' was built under R version 4.4.2

## Warning: package 'dplyr' was built under R version 4.4.2

## Warning: package 'forcats' was built under R version 4.4.2

## Warning: package 'lubridate' was built under R version 4.4.2

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
review <- reviews |>
  rename(
    user_id = User.ID,
    art_gallery = Category.1,
    dance_clubs = Category.2,
    juice_bars = Category.3,
    restaurants = Category.4,
    museums = Category.5,
    resorts = Category.6,
    parks_picnic_spots = Category.7,
    beaches = Category.8,
    theatres = Category.9,
    religious_institutions = Category.10
  )

head(review)
```

```
##   user_id art_gallery dance_clubs juice_bars restaurants museums resorts
## 1  User 1        0.93        1.80       2.29        0.62    0.80    2.42
## 2  User 2        1.02        2.20       2.66        0.64    1.42    3.18
## 3  User 3        1.22        0.80       0.54        0.53    0.24    1.54
## 4  User 4        0.45        1.80       0.29        0.57    0.46    1.52
## 5  User 5        0.51        1.20       1.18        0.57    1.54    2.02
## 6  User 6        0.99        1.28       0.72        0.27    0.74    1.26
##   parks_picnic_spots beaches theatres religious_institutions
## 1               3.19    2.79     1.82                   2.42
## 2               3.21    2.63     1.86                   2.32
## 3               3.18    2.80     1.31                   2.50
## 4               3.18    2.96     1.57                   2.86
## 5               3.18    2.78     1.18                   2.54
## 6               3.17    2.89     1.66                   3.66
```

Now that we have renamed to columns to better represent their categories we will pivot long to make our analysis easier with the tidy format.

```
reviews <- review |>
  pivot_longer(
    cols = c(art_gallery, dance_clubs, juice_bars, restaurants,
             museums, resorts, parks_picnic_spots, beaches,
             theatres, religious_institutions),
    names_to = "Categories",
    values_to = "Rating"
  )

reviews
```

```
## # A tibble: 9,800 x 3
##    user_id Categories             Rating
##    <chr>   <chr>                   <dbl>
##  1 User 1  art_gallery              0.93
##  2 User 1  dance_clubs              1.8
##  3 User 1  juice_bars               2.29
##  4 User 1  restaurants              0.62
##  5 User 1  museums                  0.8
##  6 User 1  resorts                  2.42
##  7 User 1  parks_picnic_spots       3.19
##  8 User 1  beaches                  2.79
##  9 User 1  theatres                 1.82
## 10 User 1  religious_institutions   2.42
## # i 9,790 more rows
```

We can also check which ratings are higher than a 3

```
high_ratings = reviews %>%
  filter(Rating > 3)
head(high_ratings)
```

```
## # A tibble: 6 x 3
##   user_id Categories         Rating
##   <chr>   <chr>               <dbl>
## 1 User 1  parks_picnic_spots   3.19
## 2 User 2  resorts              3.18
## 3 User 2  parks_picnic_spots   3.21
## 4 User 3  parks_picnic_spots   3.18
## 5 User 4  parks_picnic_spots   3.18
## 6 User 5  parks_picnic_spots   3.18
```

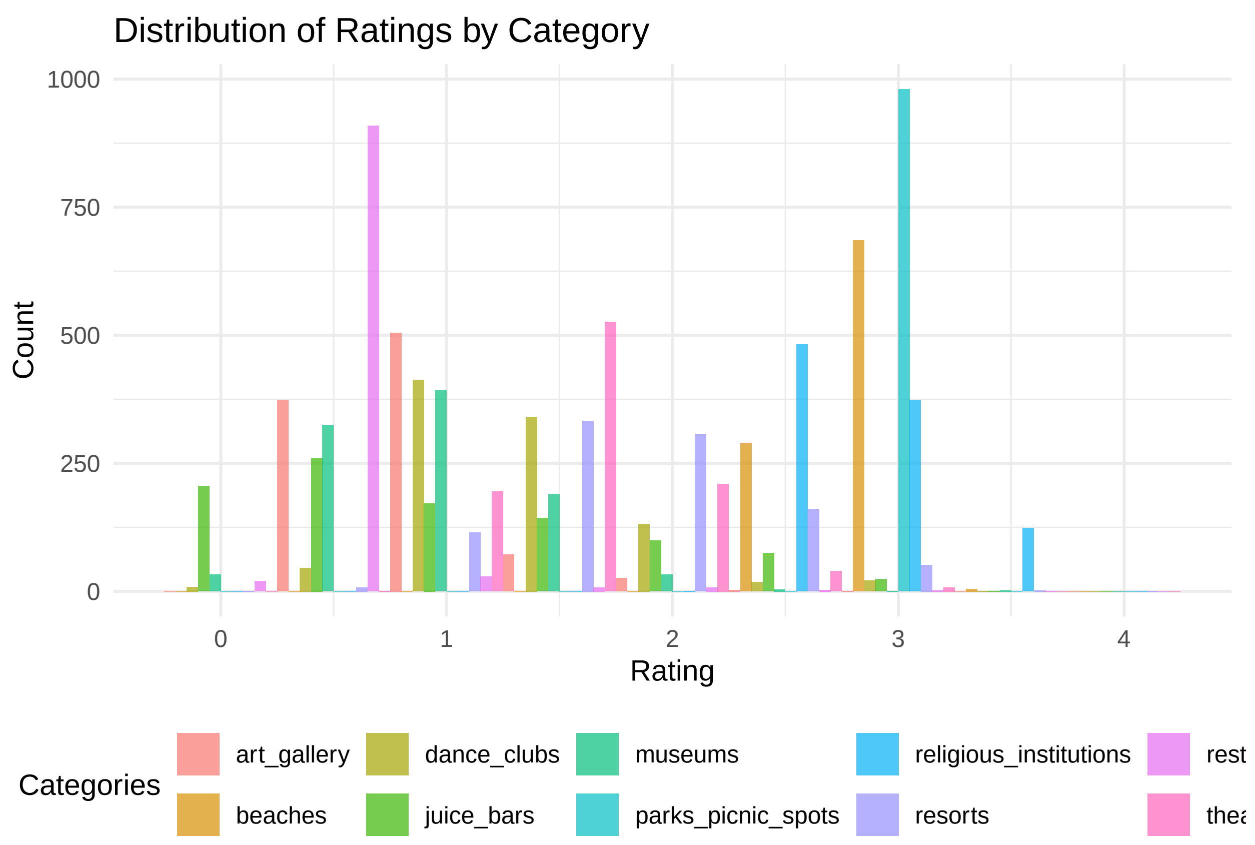
And which are lower than 1

```
low_ratings = reviews %>%
  filter(Rating < 1)
head(low_ratings)
```

```
## # A tibble: 6 x 3
##   user_id Categories  Rating
##   <chr>   <chr>        <dbl>
## 1 User 1  art_gallery   0.93
## 2 User 1  restaurants   0.62
## 3 User 1  museums       0.8
## 4 User 2  restaurants   0.64
## 5 User 3  dance_clubs   0.8
## 6 User 3  juice_bars    0.54
```

We can use a histogram to visualize the distribution of ratings across categories, where the X axis displays the ratings and the Y axis displays how many times each rating appears across all reviews.

```
ggplot(reviews, aes(x = Rating, fill = Categories)) +
  geom_histogram(binwidth = 0.5, alpha = 0.7, position = "dodge") +
  theme_minimal() +
  labs(title = "Distribution of Ratings by Category",
       x = "Rating", y = "Count") +
  theme(legend.position = "bottom")
```



After that, we will make a table for each category

```
art_gallery_rating = reviews %>%
  filter(Categories %in% "art_gallery") %>%
  select(c("user_id", "Rating"))
head(art_gallery_rating)
```

```
## # A tibble: 6 x 2
##   user_id Rating
##   <chr>    <dbl>
## 1 User 1    0.93
## 2 User 2    1.02
## 3 User 3    1.22
## 4 User 4    0.45
## 5 User 5    0.51
## 6 User 6    0.99
```

```
dance_clubs_rating = reviews %>%
  filter(Categories %in% "dance_clubs") %>%
  select(c("user_id", "Rating"))
head(dance_clubs_rating)
```

```
## # A tibble: 6 x 2
##   user_id Rating
##   <chr>    <dbl>
## 1 User 1    1.8
## 2 User 2    2.2
## 3 User 3    0.8
## 4 User 4    1.8
## 5 User 5    1.2
## 6 User 6    1.28
```

```r
juice_bars_rating = reviews %>%
  filter(Categories %in% "juice_bars") %>%
  select(c("user_id", "Rating"))
head(juice_bars_rating)
```

```
## # A tibble: 6 x 2
##   user_id Rating
##   <chr>    <dbl>
## 1 User 1    2.29
## 2 User 2    2.66
## 3 User 3    0.54
## 4 User 4    0.29
## 5 User 5    1.18
## 6 User 6    0.72
```

```r
restaurants_rating = reviews %>%
  filter(Categories %in% "restaurants") %>%
  select(c("user_id", "Rating"))
head(restaurants_rating)
```

```
## # A tibble: 6 x 2
##   user_id Rating
##   <chr>    <dbl>
## 1 User 1    0.62
## 2 User 2    0.64
## 3 User 3    0.53
## 4 User 4    0.57
## 5 User 5    0.57
## 6 User 6    0.27
```

```r
museums_rating = reviews %>%
  filter(Categories %in% "museums") %>%
  select(c("user_id", "Rating"))
head(museums_rating)
```

```
## # A tibble: 6 x 2
##   user_id Rating
##   <chr>    <dbl>
## 1 User 1    0.8
## 2 User 2    1.42
## 3 User 3    0.24
## 4 User 4    0.46
## 5 User 5    1.54
## 6 User 6    0.74
```

```r
resorts_rating = reviews %>%
  filter(Categories %in% "resorts") %>%
  select(c("user_id", "Rating"))
head(resorts_rating)
```

```
## # A tibble: 6 x 2
##   user_id Rating
##   <chr>    <dbl>
## 1 User 1    2.42
## 2 User 2    3.18
## 3 User 3    1.54
## 4 User 4    1.52
## 5 User 5    2.02
## 6 User 6    1.26
```

```r
park_picnic_spots_rating = reviews %>%
  filter(Categories %in% "park_picnic_spots") %>%
  select(c("user_id", "Rating"))
head(park_picnic_spots_rating)
```

```
## # A tibble: 0 x 2
## # i 2 variables: user_id <chr>, Rating <dbl>
```

```r
beaches_rating = reviews %>%
  filter(Categories %in% "beaches") %>%
  select(c("user_id", "Rating"))
head(beaches_rating)
```

```
## # A tibble: 6 x 2
##   user_id Rating
##   <chr>    <dbl>
## 1 User 1    2.79
## 2 User 2    2.63
## 3 User 3    2.8
## 4 User 4    2.96
## 5 User 5    2.78
## 6 User 6    2.89
```

```r
theatres_rating = reviews %>%
  filter(Categories %in% "theatres") %>%
  select(c("user_id", "Rating"))
head(theatres_rating)
```

```
## # A tibble: 6 x 2
##   user_id Rating
##   <chr>    <dbl>
## 1 User 1    1.82
## 2 User 2    1.86
## 3 User 3    1.31
## 4 User 4    1.57
## 5 User 5    1.18
## 6 User 6    1.66
```

```r
religious_institutions_rating = reviews %>%
  filter(Categories %in% "religious_institutions") %>%
  select(c("user_id", "Rating"))
head(religious_institutions_rating)
```

```
## # A tibble: 6 x 2
##   user_id Rating
##   <chr>    <dbl>
## 1 User 1    2.42
## 2 User 2    2.32
## 3 User 3    2.5
## 4 User 4    2.86
## 5 User 5    2.54
## 6 User 6    3.66
```

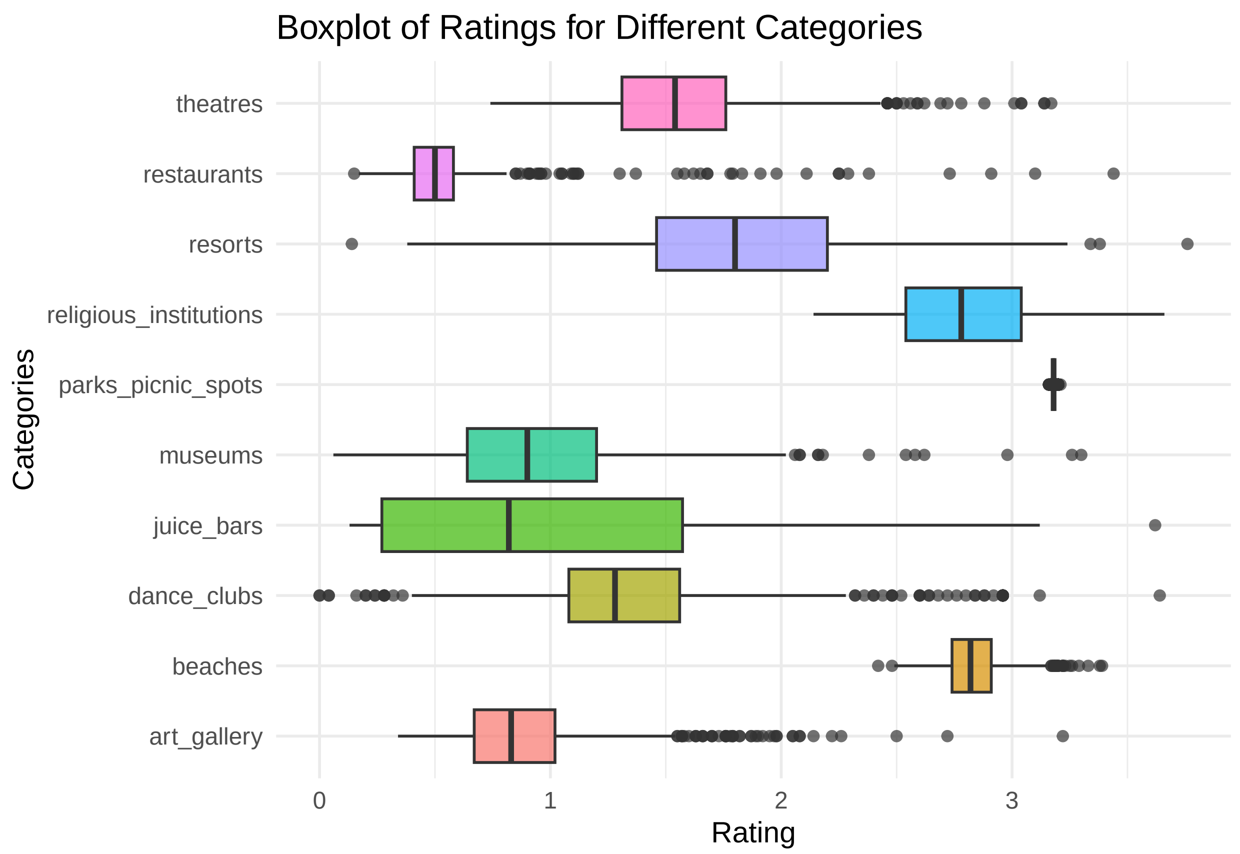
Calculating the summary statistics (mean, median, standard deviation, variance) for the ratings in each category and storing them in a summary table.

```r
calculating_Summary_Stat= function(X){
 m=mean(X)
 md= median(X)
 s_d=sd(X)
 Var=var(X)
 return(c(m,md,s_d,Var))
 }
categories = c("art_gallery", "dance_clubs", "juice_bars", "restaurants",
               "museums", "resorts", "parks_picnic_spots", "beaches", "theatres",
               "religious_institutions")

summary_table = data.frame(Category = character(0), Mean = numeric(0),
                           Median = numeric(0), SD = numeric(0), Variance = numeric(0))

for (category in categories) {
  category_rating = reviews %>%
    filter(Categories %in% category) %>%
    select(c("user_id", "Rating"))
  summary_stats = calculating_Summary_Stat(category_rating[[2]])

  summary_table = rbind(summary_table, data.frame(Category = category,
                                  Mean = summary_stats[1],
                                  Median = summary_stats[2],
                                  SD = summary_stats[3],
                                  Variance = summary_stats[4]))
}

summary_table
```

```
##                  Category      Mean Median          SD     Variance
## 1             art_gallery 0.8931939   0.83 0.326912231 1.068716e-01
## 2             dance_clubs 1.3526122   1.28 0.478280151 2.287519e-01
## 3              juice_bars 1.0133061   0.82 0.788606876 6.219008e-01
## 4             restaurants 0.5325000   0.50 0.279731330 7.824962e-02
## 5                 museums 0.9397347   0.90 0.437429966 1.913450e-01
## 6                 resorts 1.8428980   1.80 0.539538040 2.911013e-01
## 7      parks_picnic_spots 3.1809388   3.18 0.007824448 6.122199e-05
## 8                 beaches 2.8350612   2.82 0.137505488 1.890776e-02
## 9                theatres 1.5694388   1.54 0.364629454 1.329546e-01
## 10 religious_institutions 2.7992245   2.78 0.321379831 1.032850e-01
```

This can be visualized using box plots, with outliers.

```r
ggplot(reviews, aes(x = Categories, y = Rating, fill = Categories)) +
  geom_boxplot(alpha = 0.7) +
  theme_minimal() +
  coord_flip() +
  labs(title = "Boxplot of Ratings for Different Categories",
       x = "Categories", y = "Rating") +
  theme(legend.position = "none")
```



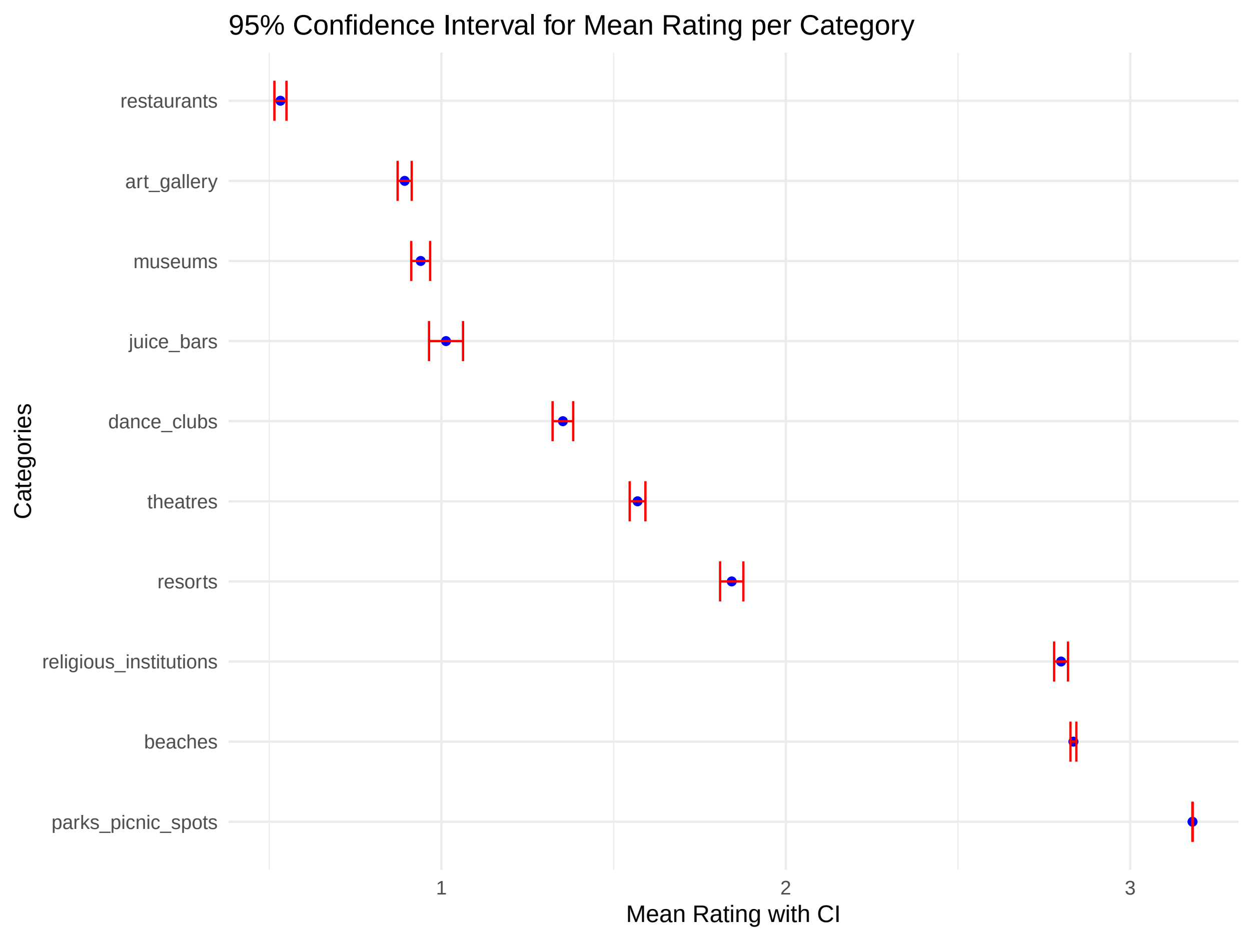
Now calculating the 95% confidence interval for the mean rating of each category and stores the results in a data frame.

```r
calculating_CI=function(X){
 m=mean(X)
 s_d=sd(X)
 l=length(X)
 from=m-1.96*s_d/sqrt(l)
 to=m+1.96*s_d/sqrt(l)
 return(c(from,to))
}

CI_95 = data.frame(Category = character(0), CI1 = numeric(0),
                          CI2 = numeric(0))

for (category in categories) {
  category_rating = reviews %>%
    filter(Categories %in% category) %>%
    select(c("user_id", "Rating"))
  calculating_interval = calculating_CI(category_rating[[2]])

  CI_95 = rbind(CI_95, data.frame(Category = category,
                                  CI1 =  calculating_interval[1],CI2 =  calculating_interval[2]))
}

CI_95
```

```
##                  Category       CI1       CI2
## 1             art_gallery 0.8727259 0.9136618
## 2             dance_clubs 1.3226672 1.3825573
## 3              juice_bars 0.9639315 1.0626807
## 4             restaurants 0.5149860 0.5500140
## 5                 museums 0.9123472 0.9671221
## 6                 resorts 1.8091175 1.8766784
## 7       parks_picnic_spots 3.1804489 3.1814287
## 8                 beaches 2.8264520 2.8436704
## 9                theatres 1.5466094 1.5922682
## 10 religious_institutions 2.7791029 2.8193460
```

Then we visualize the size of the intervals using error bars

```
#fig.width and fig.height are used to make the area R give the plot larger, the data is harder to see w
ggplot(CI_95, aes(x = reorder(Category, -CI1), y = (CI1+CI2)/2)) +
  geom_point(color = "blue", size = 1.5) +
  geom_errorbar(aes(ymin = CI1, ymax = CI2), width = 0.5, color = "red") +
  theme_minimal() +
  coord_flip() +
  labs(title = "95% Confidence Interval for Mean Rating per Category",
       x = "Categories", y = "Mean Rating with CI")
```



Calculating the Pearson correlation matrix for all columns in the 'review' data frame

```
cor(review[,-1],method = "pearson")
```

```
##                         art_gallery dance_clubs   juice_bars restaurants
## art_gallery             1.000000000 -0.18769237  0.008612625  0.07345561
## dance_clubs            -0.187692365  1.00000000  0.043586045  0.13162356
## juice_bars              0.008612625  0.04358605  1.000000000  0.06112782
## restaurants             0.073455614  0.13162356  0.061127822  1.00000000
## museums                -0.100482517  0.11963095  0.281668111  0.10187761
## resorts                 0.094141522  0.14840356  0.356434838  0.21586633
## parks_picnic_spots     -0.012474431  0.11005208  0.750650935  0.22834826
## beaches                 0.020029399 -0.15864173 -0.172953071 -0.10358278
## theatres               -0.047312854  0.07334233 -0.085435390  0.02667007
## religious_institutions  0.050699941 -0.06576225 -0.440542734 -0.35290940
##                           museums      resorts parks_picnic_spots
## art_gallery            -0.10048252 9.414152e-02        -0.01247443
## dance_clubs             0.11963095 1.484036e-01         0.11005208
## juice_bars              0.28166811 3.564348e-01         0.75065094
## restaurants             0.10187761 2.158663e-01         0.22834826
## museums                 1.00000000 5.813057e-01         0.23231780
## resorts                 0.58130573 1.000000e+00         0.43074335
## parks_picnic_spots      0.23231780 4.307434e-01         1.00000000
## beaches                -0.02095721 7.057716e-05        -0.07249175
## theatres                0.04169350  9.640958e-02          0.08496488
## religious_institutions -0.24746972 -4.381035e-01         -0.71073094
##                             beaches    theatres religious_institutions
## art_gallery             2.002940e-02 -0.04731285             0.05069994
## dance_clubs            -1.586417e-01  0.07334233            -0.06576225
## juice_bars             -1.729531e-01 -0.08543539            -0.44054273
## restaurants            -1.035828e-01  0.02667007            -0.35290940
## museums                -2.095721e-02  0.04169350            -0.24746972
## resorts                 7.057716e-05  0.09640958            -0.43810350
## parks_picnic_spots     -7.249175e-02  0.08496488            -0.71073094
## beaches                 1.000000e+00  0.16969491             0.11470062
## theatres                1.696949e-01  1.00000000            -0.04568217
## religious_institutions  1.147006e-01 -0.04568217             1.00000000
```

The Strongest Positive Correlation is between parks_picnic_spots and juice_bars which is 0.7506509

The Strongest Negative Correlation is between religious_institutions and parks_picnic_spots which is -0.71073094

Calculating the Spearman correlation matrix for all columns in the 'review' data frame

```
cor(review[,-1],method = "spearman")
```

```
##                          art_gallery  dance_clubs   juice_bars restaurants
## art_gallery             1.0000000000 -0.194209244  0.075176097  0.01816295
## dance_clubs            -0.1942092442  1.000000000  0.005386198  0.04326230
## juice_bars              0.0751760971  0.005386198  1.000000000  0.20515327
## restaurants             0.0181629500  0.043262305  0.205153274  1.00000000
## museums                -0.0685588532  0.115838003  0.323479069  0.19452955
## resorts                 0.1542965622  0.101428221  0.368726682  0.41026431
## parks_picnic_spots     -0.0050537788  0.075169668  0.694094982  0.46250294
## beaches                -0.0003791795 -0.157100827 -0.148667824 -0.04934650
## theatres               -0.0056378868  0.076487160 -0.061770539  0.11680985
## religious_institutions  0.0247113108 -0.027615815 -0.425543389 -0.58006325
##                            museums     resorts parks_picnic_spots       beaches
## art_gallery            -0.06855885  0.15429656       -0.005053779 -0.0003791795
## dance_clubs             0.11583800  0.10142822        0.075169668 -0.1571008269
## juice_bars              0.32347907  0.36872668        0.694094982 -0.1486678236
## restaurants             0.19452955  0.41026431        0.462502938 -0.0493464975
## museums                 1.00000000  0.59525554        0.260784175 -0.0138257023
## resorts                 0.59525554  1.00000000        0.445913093 -0.0107808448
## parks_picnic_spots      0.26078417  0.44591309        1.000000000 -0.0900615895
## beaches                -0.01382570 -0.01078084       -0.090061590  1.0000000000
## theatres                0.03001192  0.12054329        0.108730551  0.1657627375
## religious_institutions -0.26032681 -0.45154572       -0.728365538  0.1403519492
##                            theatres religious_institutions
## art_gallery            -0.005637887             0.02471131
## dance_clubs             0.076487160            -0.02761582
## juice_bars             -0.061770539            -0.42554339
## restaurants             0.116809855            -0.58006325
## museums                 0.030011919            -0.26032681
## resorts                 0.120543289            -0.45154572
## parks_picnic_spots      0.108730551            -0.72836554
## beaches                 0.165762738             0.14035195
## theatres                1.000000000           -0.05097580
## religious_institutions -0.050975801            1.00000000
```

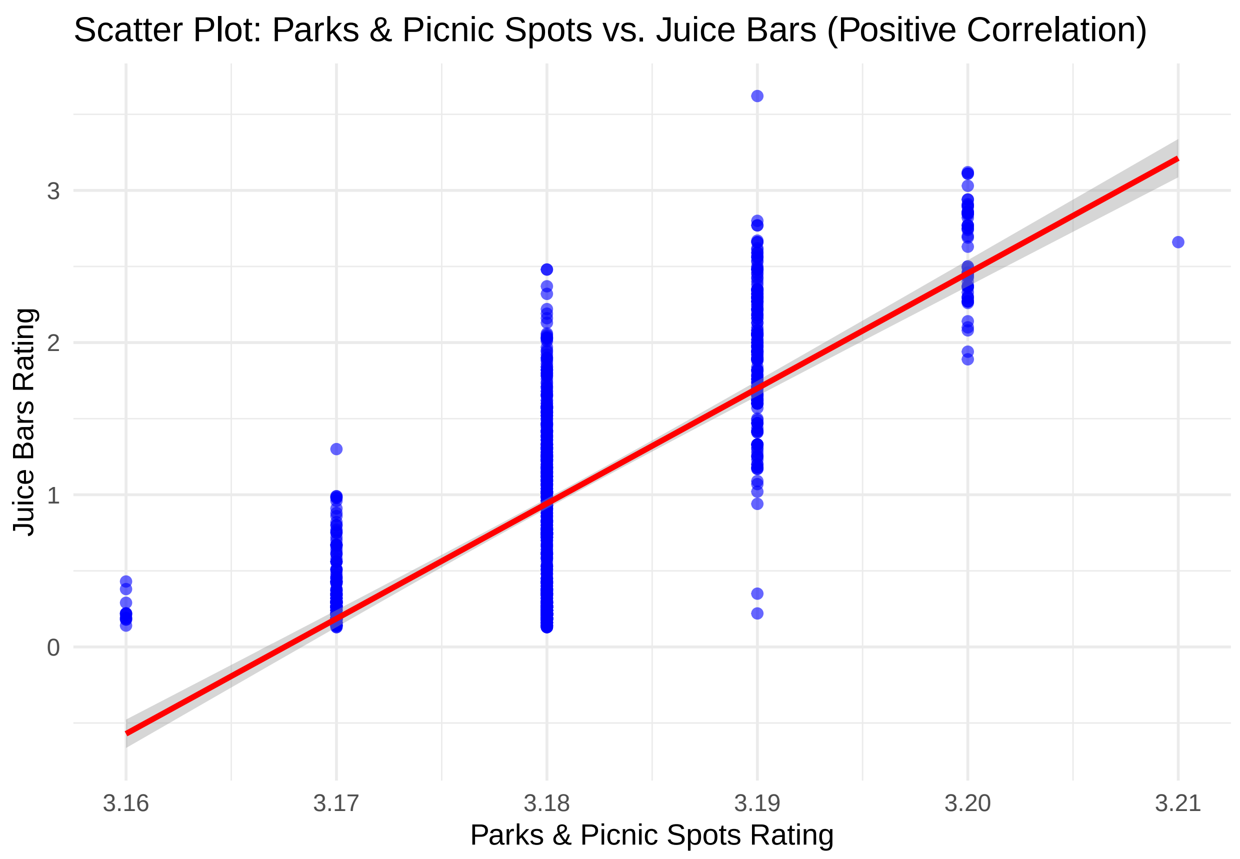
The Strongest Positive Correlation is between parks_picnic_spots and juice_bars which is 0.694094982

The Strongest Negative Correlation is between religious_institutions and parks_picnic_spots which is -0.728365538

The strongest positive and negative correlation can be visualized using a scatter plot

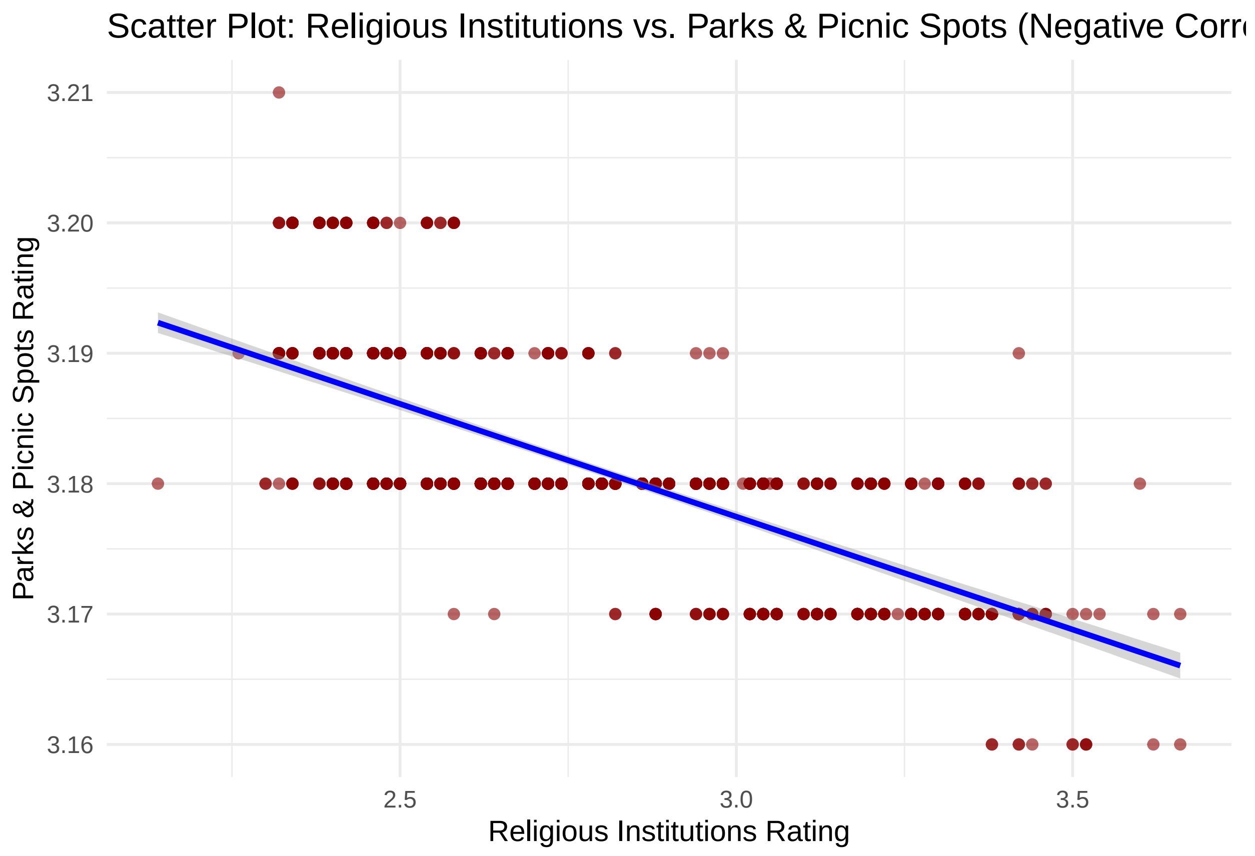
```
ggplot(review, aes(x = parks_picnic_spots, y = juice_bars)) +
  geom_point(alpha = 0.6, color = "blue") +
  geom_smooth(method = "lm", col = "red") +
  theme_minimal() +
  labs(title = "Scatter Plot: Parks & Picnic Spots vs. Juice Bars (Positive Correlation)",
       x = "Parks & Picnic Spots Rating", y = "Juice Bars Rating")
```

```
## 'geom_smooth()' using formula = 'y ~ x'
```



```
ggplot(review, aes(x = religious_institutions, y = parks_picnic_spots)) +
  geom_point(alpha = 0.6, color = "darkred") +
  geom_smooth(method = "lm", col = "blue") +
  theme_minimal() +
  labs(title = "Scatter Plot: Religious Institutions vs. Parks & Picnic Spots (Negative Correlation)",
       x = "Religious Institutions Rating", y = "Parks & Picnic Spots Rating")
```

```
## 'geom_smooth()' using formula = 'y ~ x'
```



Machine Learning Analysis on TripAdvisor Reviews Dataset

For Machine Learning Analusis We will do the following tasks: - Perform **K-Means Clustering** to segment users based on their rating behavior. - Use **Random Forest Regression** to predict ratings for a specific category. - Later,use **Simple Regression** as well to see which model best suit for this data set.

Now performing K-Means Clustering to segment users based on rating patterns.

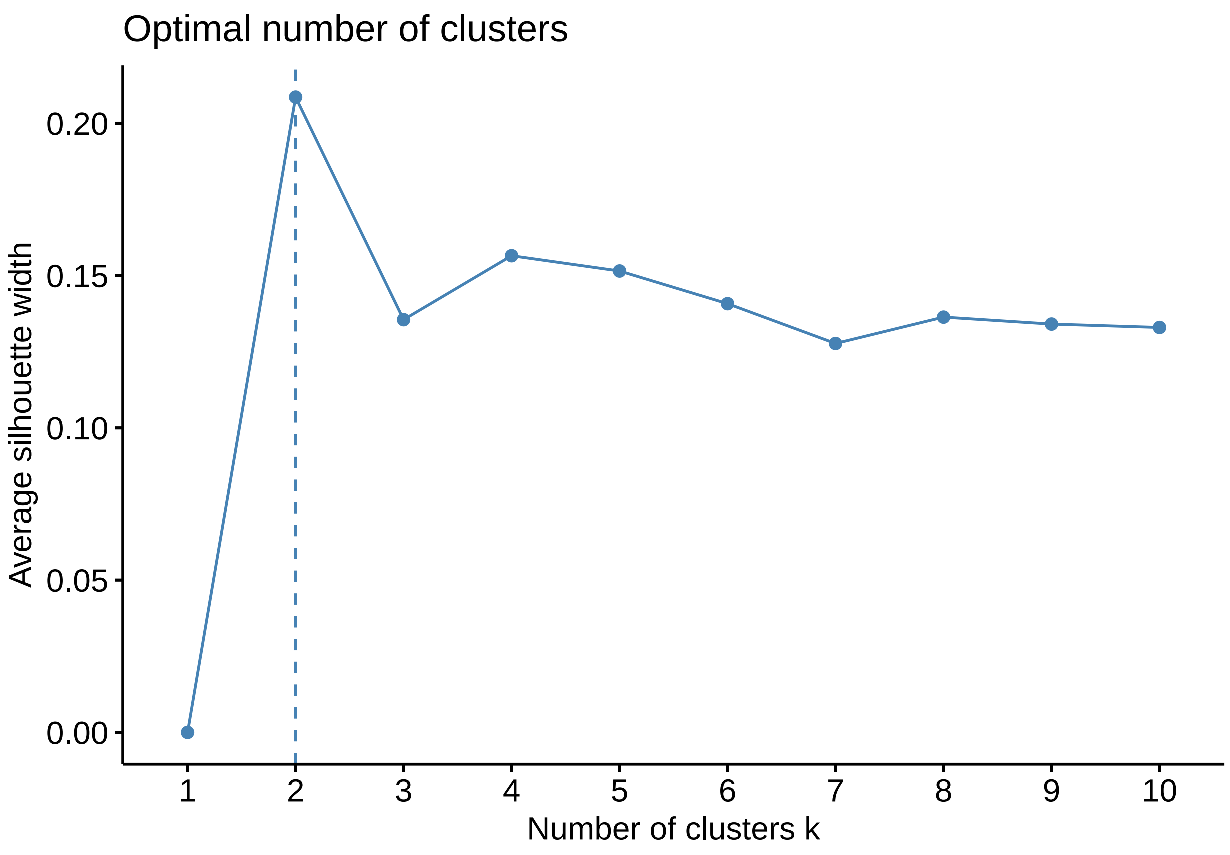
Removing user_id and normalizing data for better clustering results and then using the 'fviz_nbclust()' function with the "silhouette" method that will help us to find the best number of clusters by evaluating their separation.

```r
library(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 4.4.3
```

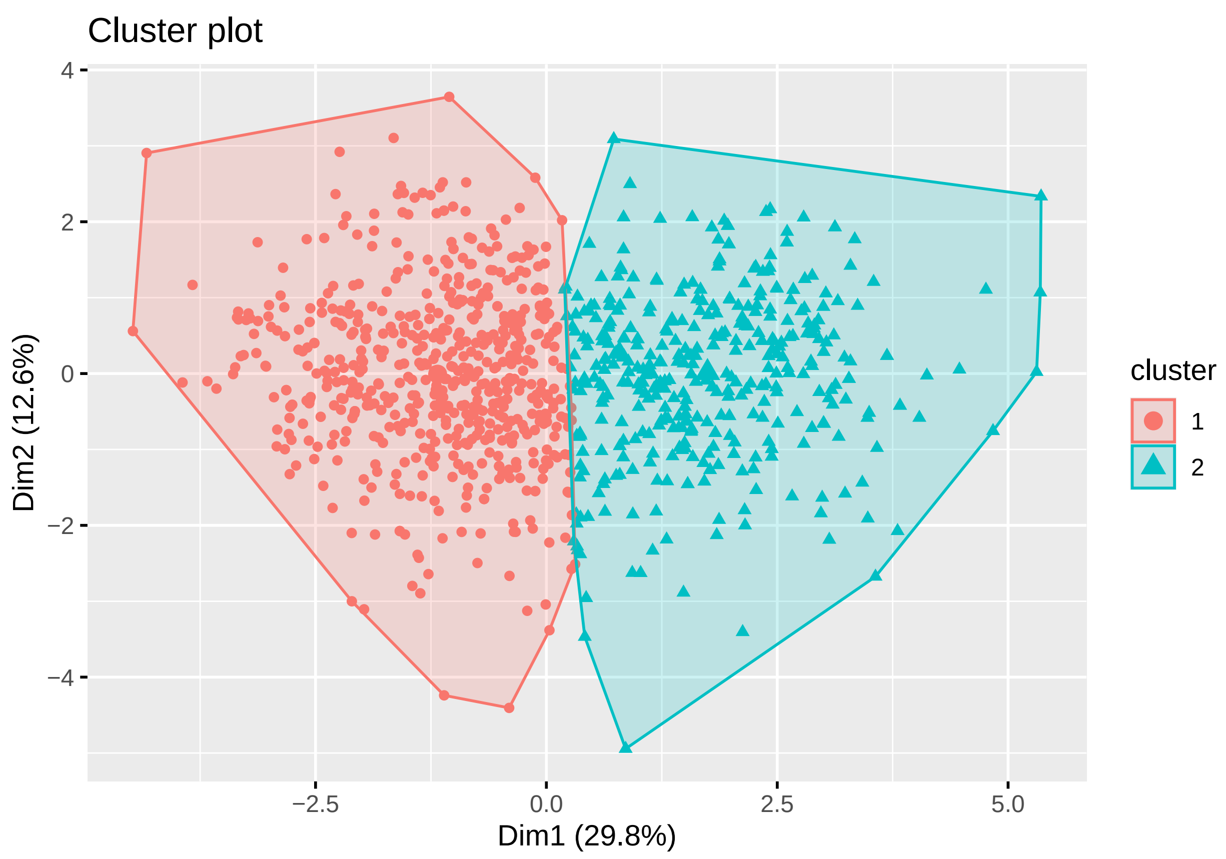
```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
scaled_data = scale(review[,-1])
fviz_nbclust(scaled_data, kmeans,method = "silhouette")
```



As we can see, the optimal K is 2. so we will keep centers = 2 and find k means and then visualizing it.

```
results = kmeans(scaled_data, centers = 2)
fviz_cluster(results, data = scaled_data, geom = "point")
```



```
review$cluster = as.factor(results$cluster)

head(review)
```

```
##   user_id art_gallery dance_clubs juice_bars restaurants museums resorts
## 1  User 1        0.93        1.80       2.29        0.62    0.80    2.42
## 2  User 2        1.02        2.20       2.66        0.64    1.42    3.18
## 3  User 3        1.22        0.80       0.54        0.53    0.24    1.54
## 4  User 4        0.45        1.80       0.29        0.57    0.46    1.52
## 5  User 5        0.51        1.20       1.18        0.57    1.54    2.02
## 6  User 6        0.99        1.28       0.72        0.27    0.74    1.26
##   parks_picnic_spots beaches theatres religious_institutions cluster
## 1               3.19    2.79     1.82                   2.42       2
## 2               3.21    2.63     1.86                   2.32       2
## 3               3.18    2.80     1.31                   2.50       1
## 4               3.18    2.96     1.57                   2.86       1
## 5               3.18    2.78     1.18                   2.54       2
## 6               3.17    2.89     1.66                   3.66       1
```

```
aggregate(review[, -1], by = list(review$cluster), mean)
```

```
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA
##   Group.1 art_gallery dance_clubs juice_bars restaurants   museums  resorts
## 1       1   0.8858632    1.267761  0.5901026   0.4564444 0.7543248 1.569641
## 2       2   0.9040506    1.478278  1.6400759   0.6451392 1.2143291 2.247595
##   parks_picnic_spots  beaches theatres religious_institutions cluster
## 1           3.176803 2.852239 1.565829               2.967043      NA
## 2           3.187063 2.809620 1.574785               2.550684      NA
```

Now applying Random Forest Regression to predict user ratings

for a specific category based on other category ratings

First we are the dataset into training (80%) and testing (20%) sets

```r
set.seed(123)
trainIndex <-  sample(1:nrow(review), 2/3*nrow(review))
trainData <- review[trainIndex, ]
testData <- review[-trainIndex, ]
```

Training the Random Forest model to predict "art_gallery" ratings using ratings from other categories as predictors and then Making predictions on the test set using the trained Random Forest model

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.4.3

## randomForest 4.7-1.2

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
rf = randomForest(art_gallery~., data=trainData)
rf
```

```
##
## Call:
##  randomForest(formula = art_gallery ~ ., data = trainData)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 0.07927264
##                     % Var explained: 27.78
```

Making predictions on the test set using the trained Random Forest model. Evaluating model performance by calculating RMSE

```
actuals = testData$art_gallery
preds = predict(rf, testData)

rmse_value = sqrt(mean((actuals - preds)^2))

rmse_value
```

```
## [1] 0.2868949
```

The RMSE value of 0.286 suggests that, on average, the model's predictions for the art_gallery category are relatively close to the actual ratings, with an error margin of about 0.29. Given the scale of ratings, this indicates the model performs reasonably well

Random Forest VS Simple Regression

We will perform a simple regression test to verify that using the random forests gives a more accurate test result

```
fit = lm(
  art_gallery ~ dance_clubs + juice_bars + restaurants + museums + resorts + parks_picnic_spots +
    beaches + theatres + religious_institutions,
  data = trainData
)
fit
```

```
##
## Call:
## lm(formula = art_gallery ~ dance_clubs + juice_bars + restaurants +
##     museums + resorts + parks_picnic_spots + beaches + theatres +
##     religious_institutions, data = trainData)
##
## Coefficients:
##            (Intercept)             dance_clubs              juice_bars
##              -1.955513               -0.169106                0.014535
##            restaurants                 museums                 resorts
##               0.151422               -0.198449                0.191601
##     parks_picnic_spots                 beaches                theatres
##               0.734645               -0.002825               -0.023044
## religious_institutions
##               0.185917
```

```
pred_vals = predict(fit, newdata = testData)

target = testData$art_gallery
rmse = sqrt(mean((target - pred_vals) ^ 2))

rmse
```

```
## [1] 0.3113744
```

The RMSE value of the simple regression method is 0.311. Since the RMSE value of simple regression is greater than the rmse for the random forest regression method (0.286), that means that random forest is a better fit for predicting art gallery ratings using other categories as predictors