VIDHI AMIN
DATA 601 INTRODUCTION TO DATA SCIENCE
PROJECT: ZOMATO

1. Introduction

Zomato is an Indian restaurant search and discovery service founded in 2008 by Deepinder Goyal and Pankaj Chaddah. It currently operates in 23 countries, including Australia and United States. It provides information and reviews on restaurants, including images of menus where the restaurant does not have its own website. Refrence:https://en.wikipedia.org/wiki/Zomato (https://en.wikipedia.org/wiki/Zomato)

There are many people who get fascinated by excellent quality of food being served in restaurants and Zomato is doing an excellent job in helping people to find best restaurants in their area. Zomato Analysis is one of the most useful analysis for foodies who want to taste the best cuisines of every part of the world which lies in their budget. This analysis is also for those who want to find the value for money restaurants in various parts of the country for the cuisines.From this analysis, we can find how many restaurants fall in which category like excellent, average, not rated etc.

2. Data Description

In this dataset, there are 9551 observations with 21 columns describing (almost) every aspect of restaurants in India, Philippines, Lowa, etc. Among explanatory variables, there are 6 integer variables such as price. range, aggregate. rating, etc., there are 2 double variables such as longitude, altitude, rest of are factors such as Address, Restaurant.Name etc.

Data Source: I got Zomato dataset from :https://www.kaggle.com/tanmaydisoriya/zomato-eda/data (https://www.kaggle.com/tanmaydisoriya/zomato-eda/data)

Here, I will import the data set into R.

```
zomatofinal <- read.csv("C:/Users/Dhvani/Desktop/zomato.csv/zomatofinal.csv")
```

I added all columns of this dataset and description of this columns.

Restaurant.Id = It gives unique Id to each restaurant.

Restaurant.Name = Name of the restaurant.

Country.code = It gives code of country where restaurant is located

City = Name of the city where restaurant is located.

Address = It gives the location of restaurant.

Locality = Local address of restaurant.

Locality. Verbose = It does not give exact address, but it gives verbose of the restaurant.

Longitude = It represent Longitude

Latitude = It represent Latitude.

Cuisines = It represent which type of cuisines restaurant offer.

Average.rate.for.two = It gives estimation of cost for two people.

Currency = Which type of currency restaurant offer.

Has.Table.booking = It gives whether restaurant allow table booking or not?

Has.Online.delivery = It gives whether restaurant provides online delivery or not?

Is.delivery.now = It gives whether restaurant provides delivery now?

Switch.to.order.menu = It gives whether it allows to switch to menu or not?

Price.range = It gives value from 1 to 4.

Aggregate.rating = It gives the average rating of restaurant from 1 to 5.

Rating.color = it gives colour according to rating of restaurant.

Rating.Text = It provides text which describes the rating of restaurant.

Votes = it gives the vote.

Library which I have used in this project.

```
library(ggplot2)
library(magrittr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
```

```
##
## Attaching package: 'tidyr'
```

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

```
library(VIM)
```

```
## Loading required package: colorspace
```

```
## Loading required package: grid
```

```
## Loading required package: data.table
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## VIM is ready to use.
##  Since version 4.0.0 the GUI is in its own package VIMGUI.
##
##           Please use the package to use the new (and old) GUI.
```

```
## Suggestions and bug-reports can be submitted at: https://github.com/alexkowa/VIM/is
sues
```

```
##
## Attaching package: 'VIM'
```

```
## The following object is masked from 'package:datasets':
##
##     sleep
```

```
library(corrgram)
library(plyr)
```

```
## ------------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## ------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'
```

```
## The following object is masked from 'package:corrgram':
##
##     baseball
```

```
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
library("tm")
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

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

```
library("SnowballC")
library("wordcloud")
```

```
## Loading required package: RColorBrewer
```

```
library("RColorBrewer")
library("ggmap")
```

```
##
## Attaching package: 'ggmap'
```

```
## The following object is masked from 'package:magrittr':
##
##     inset
```

```
library(maptools)
```

```
## Loading required package: sp
```

```
## Checking rgeos availability: FALSE
##      Note: when rgeos is not available, polygon geometry     computations in maptoo
ls depend on gpclib,
##      which has a restricted licence. It is disabled by default;
##      to enable gpclib, type gpclibPermit()
```

```
library(maps)
```

```
##
## Attaching package: 'maps'
```

```
## The following object is masked from 'package:plyr':
##
##     ozone
```

```
library(rpart)
library(rpart.plot)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

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

```
library(rworldmap)
```

```
## ### Welcome to rworldmap ###
```

```
## For a short introduction type :   vignette('rworldmap')
```

```
library(randomForest)
```

```
## randomForest 4.6-14
```

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:gridExtra':
##
##     combine
```

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

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

```
library(rpart)
library(rpart.plot)
library(ROCR)
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:wordcloud':
##
##     textplot
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.2.1 --
```

```
## v tibble  1.4.2     v purrr   0.2.4
## v readr   1.1.1     v stringr 1.3.0
## v tibble  1.4.2     v forcats 0.3.0
```

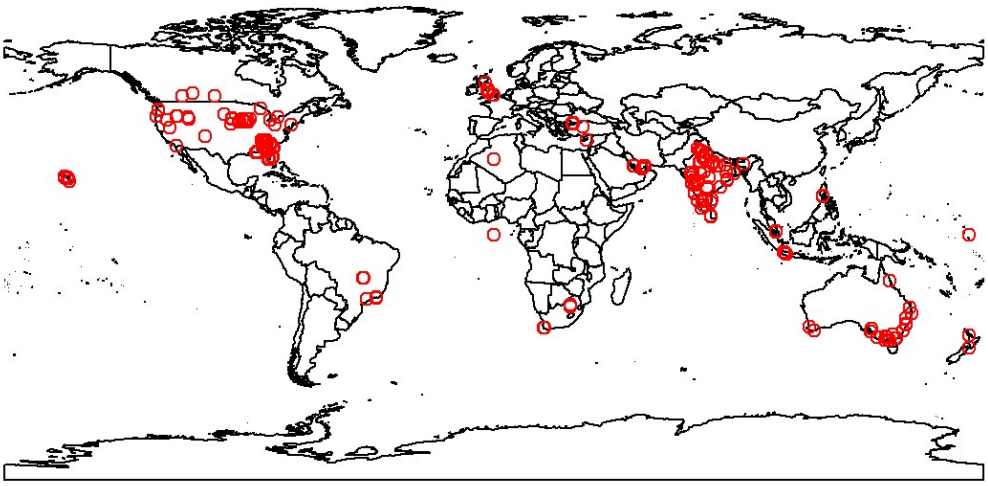
```
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x NLP::annotate()         masks ggplot2::annotate()
## x plyr::arrange()         masks dplyr::arrange()
## x data.table::between()   masks dplyr::between()
## x randomForest::combine() masks gridExtra::combine(), dplyr::combine()
## x purrr::compact()        masks plyr::compact()
## x plyr::count()           masks dplyr::count()
## x tidyr::extract()        masks magrittr::extract()
## x plyr::failwith()        masks dplyr::failwith()
## x dplyr::filter()         masks stats::filter()
## x data.table::first()     masks dplyr::first()
## x plyr::id()              masks dplyr::id()
## x ggmap::inset()          masks magrittr::inset()
## x dplyr::lag()            masks stats::lag()
## x data.table::last()      masks dplyr::last()
## x purrr::map()            masks maps::map()
## x randomForest::margin()  masks ggplot2::margin()
## x plyr::mutate()          masks dplyr::mutate()
## x plyr::rename()          masks dplyr::rename()
## x purrr::set_names()      masks magrittr::set_names()
## x plyr::summarise()       masks dplyr::summarise()
## x plyr::summarize()       masks dplyr::summarize()
## x purrr::transpose()      masks data.table::transpose()
```

```
library("ggmap")
library(maptools)
library(maps)
```

3. Data Assessment and Visualization:

Now we will identify where are the maximum restaurants. Refrence: http://www.milanor.net/blog/maps-in-r-plotting-data-points-on-a-map/ (http://www.milanor.net/blog/maps-in-r-plotting-data-points-on-a-map/)

```
library(rworldmap)
newmap <- getMap(resolution = "low")
plot(newmap , asp = 1)
points(zomatofinal$Longitude, zomatofinal$Latitude, col = "red", cex = 1)
```



From this graph we can say that there are maximum restaurants presents in India, United States, Canada and Australia.There are few restaurants presents in other countries.

First we will summary the dataset.

```
summary(zomatofinal)
```

```
##  Restaurant.ID             Restaurant.Name  Country.Code
##  Min.   :      53   Cafe Coffee Day :  83   Min.   :  1.00
##  1st Qu.:  301962   Domino's Pizza  :  79   1st Qu.:  1.00
##  Median : 6004089   Subway          :  63   Median :  1.00
##  Mean   : 9051128   Green Chick Chop:  51   Mean   : 18.37
##  3rd Qu.:18352292   McDonald's      :  48   3rd Qu.:  1.00
##  Max.   :18500652   Keventers       :  34   Max.   :216.00
##                     (Other)         :9193
##        City
##  New Delhi:5473
##  Gurgaon  :1118
##  Noida    :1080
##  Faridabad: 251
##  Ghaziabad:  25
##  Ahmedabad:  21
##  (Other)  :1583
##                                                                    Address
##  Dilli Haat, INA, New Delhi                                            :  11
##  Sector 41, Noida                                                      :  11
##  Greater Kailash (GK) 1, New Delhi                                     :  10
##  The Imperial, Janpath, New Delhi                                      :   9
##  Cyber Hub, DLF Cyber City, Gurgaon                                    :   8
##  Food Court, 3rd Floor, Logix City Centre, Sector 32, Near Sector 34, Noida:   8
##  (Other)                                                               :9494
##            Locality                      Locality.Verbose
##  Connaught Place: 122   Connaught Place, New Delhi: 122
##  Rajouri Garden :  99   Rajouri Garden, New Delhi :  99
##  Shahdara       :  87   Shahdara, New Delhi       :  87
##  Defence Colony :  86   Defence Colony, New Delhi :  86
##  Malviya Nagar  :  85   Pitampura, New Delhi      :  85
##  Pitampura      :  85   Malviya Nagar, New Delhi  :  84
##  (Other)        :8987   (Other)                   :8988
##    Longitude          Latitude                      Cuisines
##  Min.   :-157.95   Min.   :-41.33   North Indian         : 936
##  1st Qu.:  77.08   1st Qu.: 28.48   North Indian, Chinese: 511
##  Median :  77.19   Median : 28.57   Chinese              : 354
##  Mean   :  64.13   Mean   : 25.85   Fast Food            : 354
##  3rd Qu.:  77.28   3rd Qu.: 28.64   North Indian, Mughlai: 334
##  Max.   : 174.83   Max.   : 55.98   Cafe                 : 299
##                                     (Other)              :6763
##  Average.Cost.for.two               Currency    Has.Table.booking
##  Min.   :     0       Indian Rupees(Rs.):8652   No :8393
##  1st Qu.:   250       Dollar($)         : 482   Yes:1158
##  Median :   400       Pounds(£)         :  80
##  Mean   :  1199       Brazilian Real(R$):  60
##  3rd Qu.:   700       Emirati Diram(AED):  60
##  Max.   :800000       Rand(R)           :  60
##                       (Other)           : 157
```

```
##  Has.Online.delivery Is.delivering.now Switch.to.order.menu
##  No :7100            No :9517          No:9551
##  Yes:2451            Yes:  34
##
##
##
##
##
##   Price.range    Aggregate.rating     Rating.color     Rating.text
##  Min.   :1.000   Min.   :0.000    Dark Green: 301   Average  :3737
##  1st Qu.:1.000   1st Qu.:2.500    Green     :1079   Excellent: 301
##  Median :2.000   Median :3.200    Orange    :3737   Good     :2100
##  Mean   :1.805   Mean   :2.666    Red       : 186   Not rated:2148
##  3rd Qu.:2.000   3rd Qu.:3.700    White     :2148   Poor     : 186
##  Max.   :4.000   Max.   :4.900    Yellow    :2100   Very Good:1079
##
##      Votes
##  Min.   :    0.0
##  1st Qu.:    5.0
##  Median :   31.0
##  Mean   :  156.9
##  3rd Qu.:  131.0
##  Max.   :10934.0
##
```

From these summary we can get idea that there are more restaurants located in New Delhi, Noida and Gurgaon. From all restaurants, number of franchising restaurants are more such as cafe coffee day, subway, Dominos pizza, Mcdonalads etc.Most of resturants provide north indian and chinese food. Most of restuarants do not provide table booking, online delivery and switch to order menu facilities.

Data Validation and wrangling:

Now we will find unique value of avg cost for two.

```
sort(unique(zomatofinal$Average.Cost.for.two))
```

```
##   [1]      0      7     10     15     20     25     30     35     40     45
##  [11]     50     55     60     65     70     75     80     85     90     95
##  [21]    100    105    110    120    125    130    140    150    160    170
##  [31]    180    190    200    220    230    240    250    260    270    280
##  [41]    285    290    294    300    315    320    330    350    360    390
##  [51]    400    410    430    445    450    500    515    535    545    550
##  [61]    570    600    650    700    720    750    800    850    900    950
##  [71]    955   1000   1050   1100   1150   1200   1250   1300   1350   1400
##  [81]   1450   1500   1540   1550   1600   1650   1700   1750   1800   1850
##  [91]   1900   1950   2000   2100   2200   2300   2350   2400   2500   2600
## [101]   2650   2700   2800   2900   3000   3200   3210   3300   3500   3600
## [111]   3650   3700   3800   4000   4100   4200   4300   4400   4500   4700
## [121]   4800   5000   5100   5500   6000   6500   7000   8000  70000 100000
## [131] 120000 150000 165000 200000 250000 300000 350000 450000 500000 800000
```

We know that avg cost for two person never be 0 so we have to remove this rows which have avg cost for two is 0.

Now we will find which rows contain 0 value for avg cost for two.

```
which(zomatofinal$Average.Cost.for.two==0)
```

```
##  [1]   85   86   88  202  241  278  347  398  408  635  638  640  678  852
## [15] 2365 2369 9243 9255
```

These all rows contain 0 value for avg cost for two people.

First, I will check is any restaurant id has any duplicate ID?

```
#To find the position of duplicate elements in Restaurant ID, use this:
#duplicated(zomatofinal$Restaurant.ID)


#Extract duplicate elements:
zomatofinal$Restaurant.ID[duplicated(zomatofinal$Restaurant.ID)]
```

```
## integer(0)
```

```
#If you want to remove duplicated elements, use !duplicated(), where ! is a logical ne
gation:
#zomatofinal$Restaurant.ID[!duplicated(zomatofinal$Restaurant.ID)]
```

From this we can conclude that there is no duplicate ID in the Restaurant Id.

second, there are many restaurants that has same name. For example, In these dataset there are many Dominos, mcDonalds, subway restaurants in the same state but in different location.So we have to figure out that whether same restaunt name has different unique id or not)

```
library(magrittr)
library(dplyr)
#first I only selected two columns to verify.
r<- zomatofinal %>% select(Restaurant.ID, Restaurant.Name)

#I converted into dataframe.
g<- as.data.frame(r)


#I used duplicated function to find dupication in Restaurant name and id both
y<- duplicated(g)

#There is no Duplication in Restaurant name and Id. So I only print 6 rows of this out
put.
head(y)
```

```
## [1] FALSE FALSE FALSE FALSE FALSE FALSE
```

From these output we can conclude that same restaunt name has different unique id because it is located in different location.

Third, I have to check is there NA in this dataset. If NA is present, then I have to drop these rows.

```
library(tidyr)
#it will drop rows which has NA.
head(drop_na(zomatofinal))
```

```
##   Restaurant.ID        Restaurant.Name Country.Code             City
## 1       6317637       Le Petit Souffle          162      Makati City
## 2       6304287       Izakaya Kikufuji          162      Makati City
## 3       6300002 Heat - Edsa Shangri-La          162 Mandaluyong City
## 4       6318506                   Ooma          162 Mandaluyong City
## 5       6314302            Sambo Kojin          162 Mandaluyong City
## 6      18189371           Din Tai Fung          162 Mandaluyong City
##                                                                 Address
## 1 Third Floor, Century City Mall, Kalayaan Avenue, Poblacion, Makati City
## 2     Little Tokyo, 2277 Chino Roces Avenue, Legaspi Village, Makati City
## 3                Edsa Shangri-La, 1 Garden Way, Ortigas, Mandaluyong City
## 4  Third Floor, Mega Fashion Hall, SM Megamall, Ortigas, Mandaluyong City
## 5        Third Floor, Mega Atrium, SM Megamall, Ortigas, Mandaluyong City
## 6 Ground Floor, Mega Fashion Hall, SM Megamall, Ortigas, Mandaluyong City
##                                  Locality
## 1  Century City Mall, Poblacion, Makati City
## 2 Little Tokyo, Legaspi Village, Makati City
## 3 Edsa Shangri-La, Ortigas, Mandaluyong City
## 4     SM Megamall, Ortigas, Mandaluyong City
## 5     SM Megamall, Ortigas, Mandaluyong City
## 6     SM Megamall, Ortigas, Mandaluyong City
##                                           Locality.Verbose Longitude
## 1       Century City Mall, Poblacion, Makati City, Makati City  121.0275
## 2      Little Tokyo, Legaspi Village, Makati City, Makati City  121.0141
## 3 Edsa Shangri-La, Ortigas, Mandaluyong City, Mandaluyong City  121.0568
## 4     SM Megamall, Ortigas, Mandaluyong City, Mandaluyong City  121.0565
## 5     SM Megamall, Ortigas, Mandaluyong City, Mandaluyong City  121.0575
## 6     SM Megamall, Ortigas, Mandaluyong City, Mandaluyong City  121.0563
##   Latitude                          Cuisines Average.Cost.for.two
## 1 14.56544       French, Japanese, Desserts                 1100
## 2 14.55371                          Japanese                 1200
## 3 14.58140 Seafood, Asian, Filipino, Indian                 4000
## 4 14.58532                  Japanese, Sushi                 1500
## 5 14.58445                 Japanese, Korean                 1500
## 6 14.58376                          Chinese                 1000
##           Currency Has.Table.booking Has.Online.delivery Is.delivering.now
## 1 Botswana Pula(P)               Yes                  No                No
## 2 Botswana Pula(P)               Yes                  No                No
## 3 Botswana Pula(P)               Yes                  No                No
## 4 Botswana Pula(P)                No                  No                No
## 5 Botswana Pula(P)               Yes                  No                No
## 6 Botswana Pula(P)                No                  No                No
##   Switch.to.order.menu Price.range Aggregate.rating Rating.color
## 1                   No           3              4.8   Dark Green
## 2                   No           3              4.5   Dark Green
## 3                   No           4              4.4        Green
## 4                   No           4              4.9   Dark Green
## 5                   No           4              4.8   Dark Green
```

```
## 6                   No           3              4.4        Green
##   Rating.text Votes
## 1   Excellent   314
## 2   Excellent   591
## 3   Very Good   270
## 4   Excellent   365
## 5   Excellent   229
## 6   Very Good   336
```

From these output, we can conclude that there is no NA in these dataset because after appling NA in these dataset number of rows are same as orignal dataset.

From these dataset, we can see that there are some restaurant has 0 value for average cost for two people.So we have to remove these type of rows because each restaurant has some value for average cost for two people.

```
nrow(zomatofinal)
```

```
## [1] 9551
```

```
zomatofinal = zomatofinal[!zomatofinal$Average.Cost.for.two== 0,]
nrow(zomatofinal)
```

```
## [1] 9533
```

Original dataset has 9551 rows.There are 18 rows which have average cost for two's value is 0 which is invalid.So after removing these rows dataset has 9533 rows.

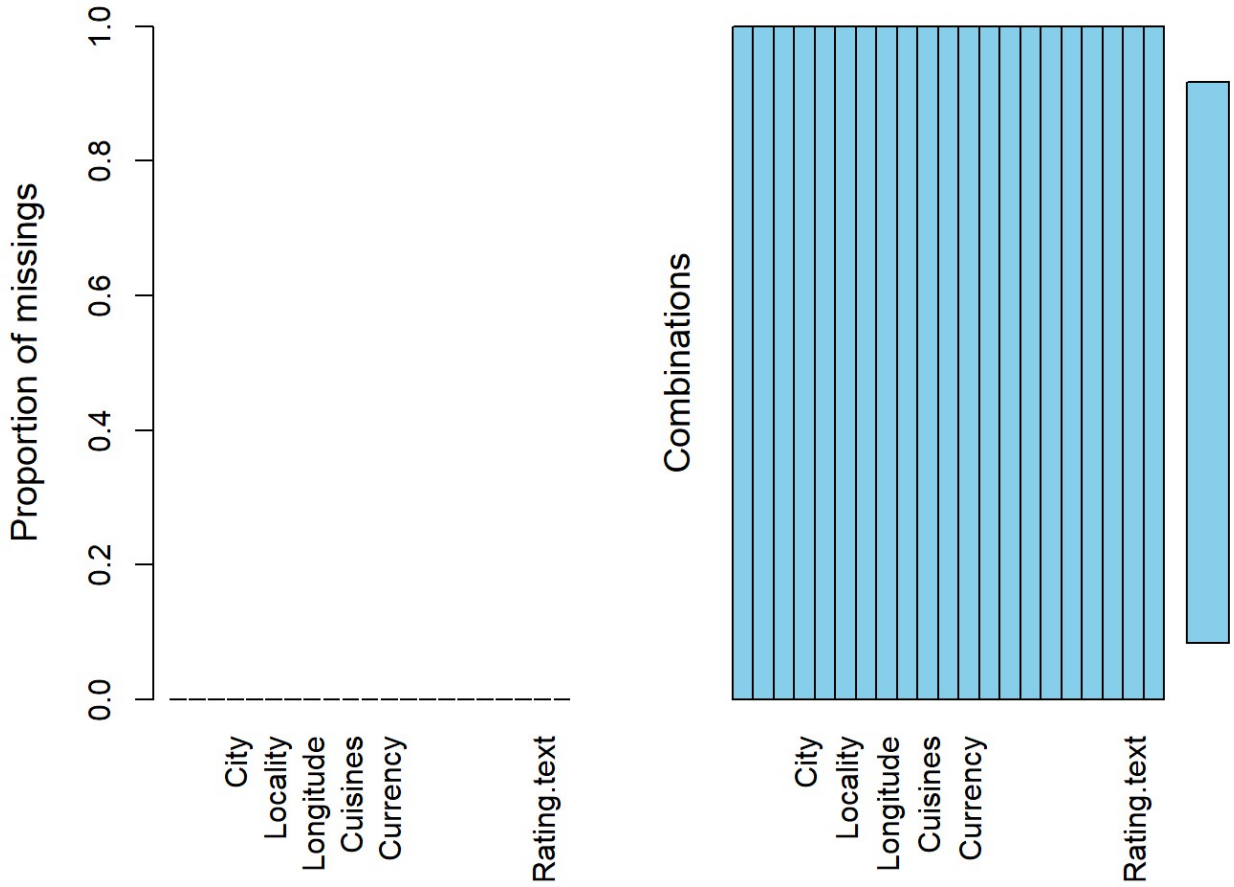
There are many restaurants in which cuisines are not specified so we have to remove these rows from dataset.

```
zomatofinal<-zomatofinal[!(zomatofinal$Cuisines==""), ]
```

There are 6 rows in which there are no cuisines satisfied. So I will remove these rows from this dataset. After applying this, we get 9527 rows.

Now we will visualize is there any missing data in zomato data set? Refrence: https://cran.r-project.org/web/packages/VIMGUI/vignettes/VIM-Imputation.pdf (https://cran.r-project.org/web/packages/VIMGUI/vignettes/VIM-Imputation.pdf)

```
# VIM library for using 'aggr'
library(VIM)

# 'aggr' plots the amount of missing/imputed values in each column
aggr(zomatofinal)
```



From this output we can say that there is no missing value in this dat set.

Now we will convert Has.Table.booking, Has.Online.booking, Is.delivering.now and Switch.to.order.menu columns into 0 and 1. 0 means No and 1 is for Yes.

```
library(plyr)
zomatofinal$Has.Table.booking <- revalue(zomatofinal$Has.Table.booking, c("Yes"=1))
zomatofinal$Has.Table.booking <- revalue(zomatofinal$Has.Table.booking, c("No"=0))
zomatofinal$Has.Online.delivery <- revalue(zomatofinal$Has.Online.delivery, c("Yes"=
1))
zomatofinal$Has.Online.delivery <- revalue(zomatofinal$Has.Online.delivery, c("No"=0))
zomatofinal$Is.delivering.now <- revalue(zomatofinal$Is.delivering.now, c("Yes"= 1))
zomatofinal$Is.delivering.now <- revalue(zomatofinal$Is.delivering.now, c("No"= 0))
zomatofinal$Switch.to.order.menu <- revalue(zomatofinal$Switch.to.order.menu, c("Ye
s"= 1))
```

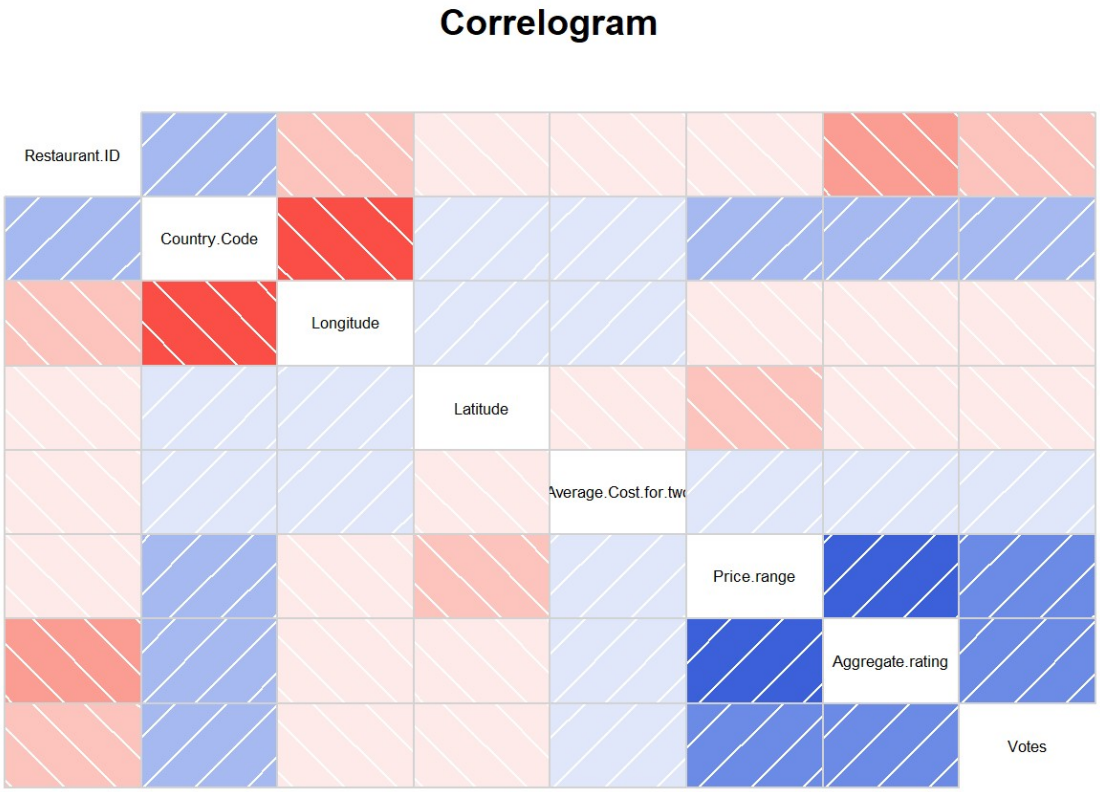
```
## The following `from` values were not present in `x`: Yes
```

```
zomatofinal$Switch.to.order.menu <- revalue(zomatofinal$Switch.to.order.menu, c("No"=
0))
```

Correlation between parameters:

The correlogram which is used to test the level of co-relation among continuous pairs of the variables in the data set. The cells of the matrix can be shaded or colored to show the co-relation value. The darker the color, the higher the co-relation between variables. Positive co-relations are displayed in blue and negative correlations in red color. Color intensity is proportional to the co-relation value. Here, I used correlogram to see the correlation between different columns.

```
library(corrgram)
corrgram(zomatofinal, order=NULL, panel=panel.shade,text.panel=panel.txt, main="Correl
ogram")
```



1)Aggregate rating and Price range are positively highly correlated.

```
cor(zomatofinal$Price.range , zomatofinal$Aggregate.rating)
```

```
## [1] 0.4399385
```

2)aggregate rating and votes are positively medium correlated.

```
cor(zomatofinal$Aggregate.rating , zomatofinal$Votes)
```

```
## [1] 0.3138192
```

3. price range and votes are positively medium correlated.

```
cor(zomatofinal$Price.range , zomatofinal$Votes)
```

```
## [1] 0.3094599
```

4)average cost for two and votes are positively low correlated with each other.

```
cor(zomatofinal$Average.Cost.for.two , zomatofinal$Votes)
```

```
## [1] 0.06781712
```

5. country code and longitude are negatively high correlated with each other.

```
cor(zomatofinal$Country.Code , zomatofinal$Longitude)
```

```
## [1] -0.6923015
```

6. restaurant id and votes are negatively medium correlated with each other.

```
cor(zomatofinal$Restaurant.ID , zomatofinal$Votes)
```

```
## [1] -0.1474492
```

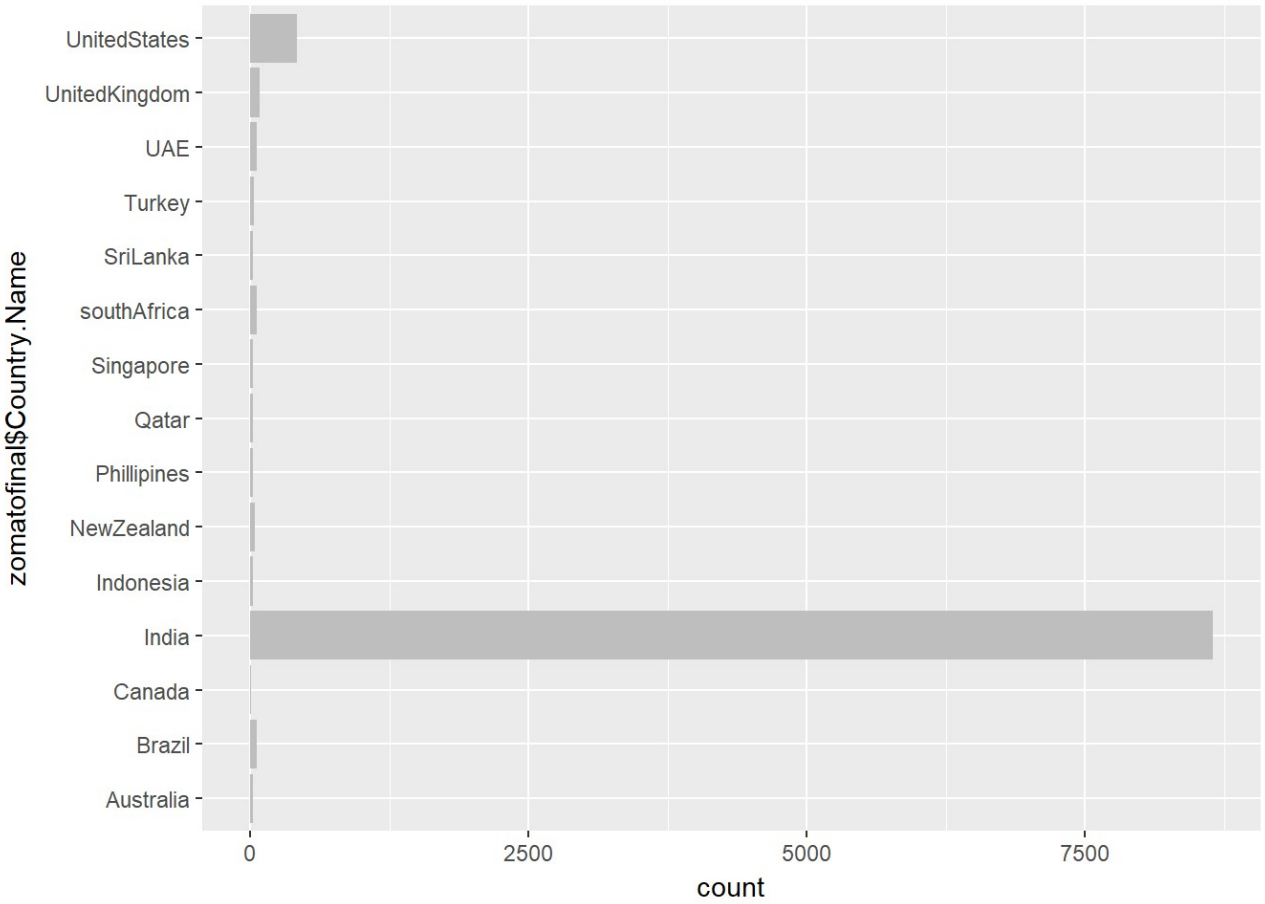
Here, I assign country Name to country code.

```
library(dplyr)

countryNames <- data.frame(Country.Code=c(1,14,30,37,94,148,162,166,184,189,191,208,21
4,215,216),
                           Country.Name= c("India","Australia","Brazil","Canada","Indo
nesia","NewZealand",
                                           "Phillipines","Qatar","Singapore","southAfr
ica","SriLanka","Turkey","UAE","UnitedKingdom","UnitedStates"),
     stringsAsFactors=FALSE)

# use dplyr::inner_join() to join country names
zomatofinal<- zomatofinal %>% inner_join(countryNames)
```

```
## Joining, by = "Country.Code"
```

Now we will see, how many restaurants are presents in each country.

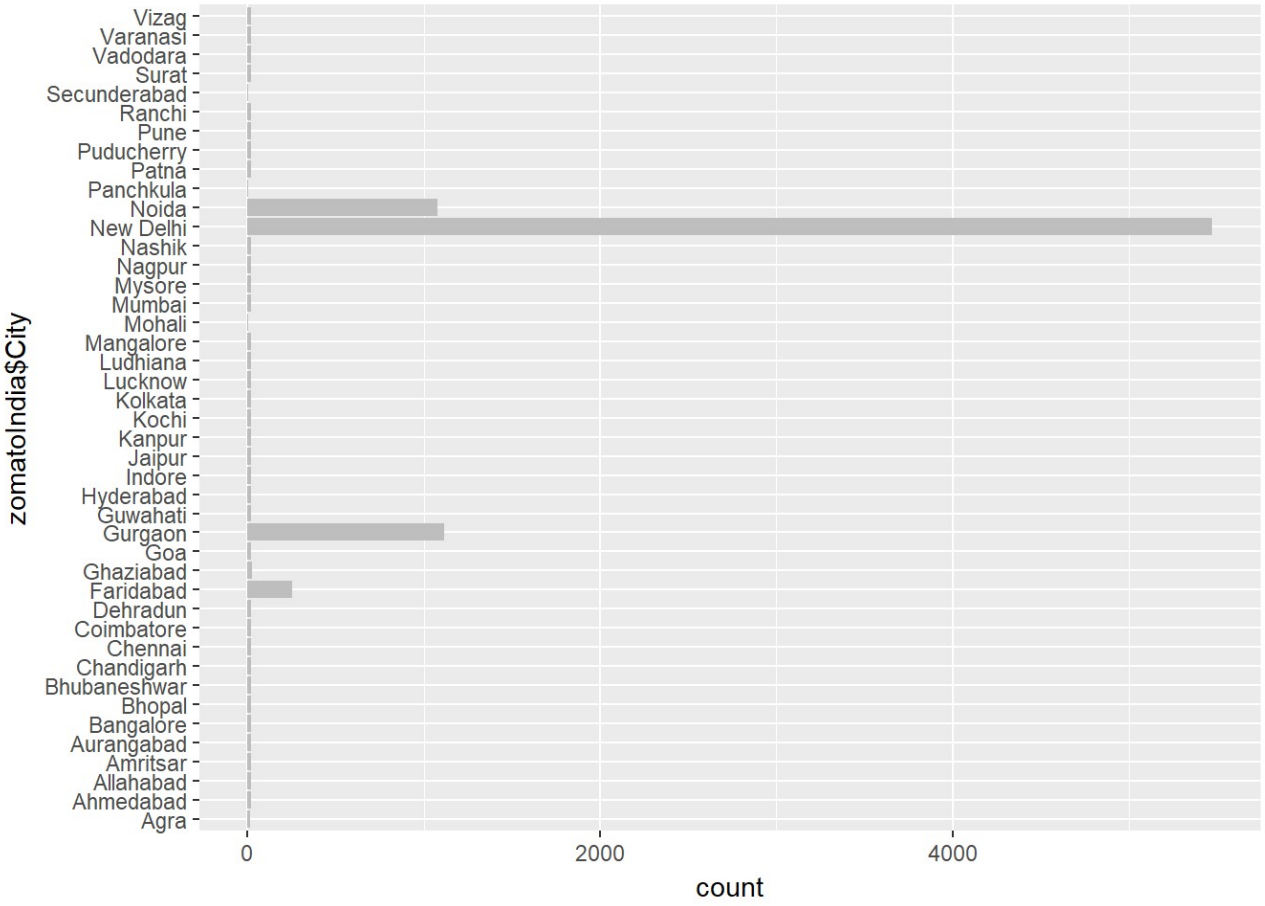
```
library(ggplot2)
ggplot(zomatofinal) +
  geom_bar(aes(x = zomatofinal$Country.Name), fill = "gray")+coord_flip()
```



From this graph, we can say that majority of restaurants are presents in India. There are only few restaurants presents in other countries.

Now we will find in India,in which cities restaurants are presents which are registered in zomato

```
library(magrittr)
library(dplyr)
zomatoIndia <- zomatofinal %>% filter( Country.Name == "India")


# Now we will find how many restaurants are presents in each city in India.

ggplot(zomatoIndia) +
  geom_bar(aes(x = zomatoIndia$City), fill = "gray") +coord_flip()
```



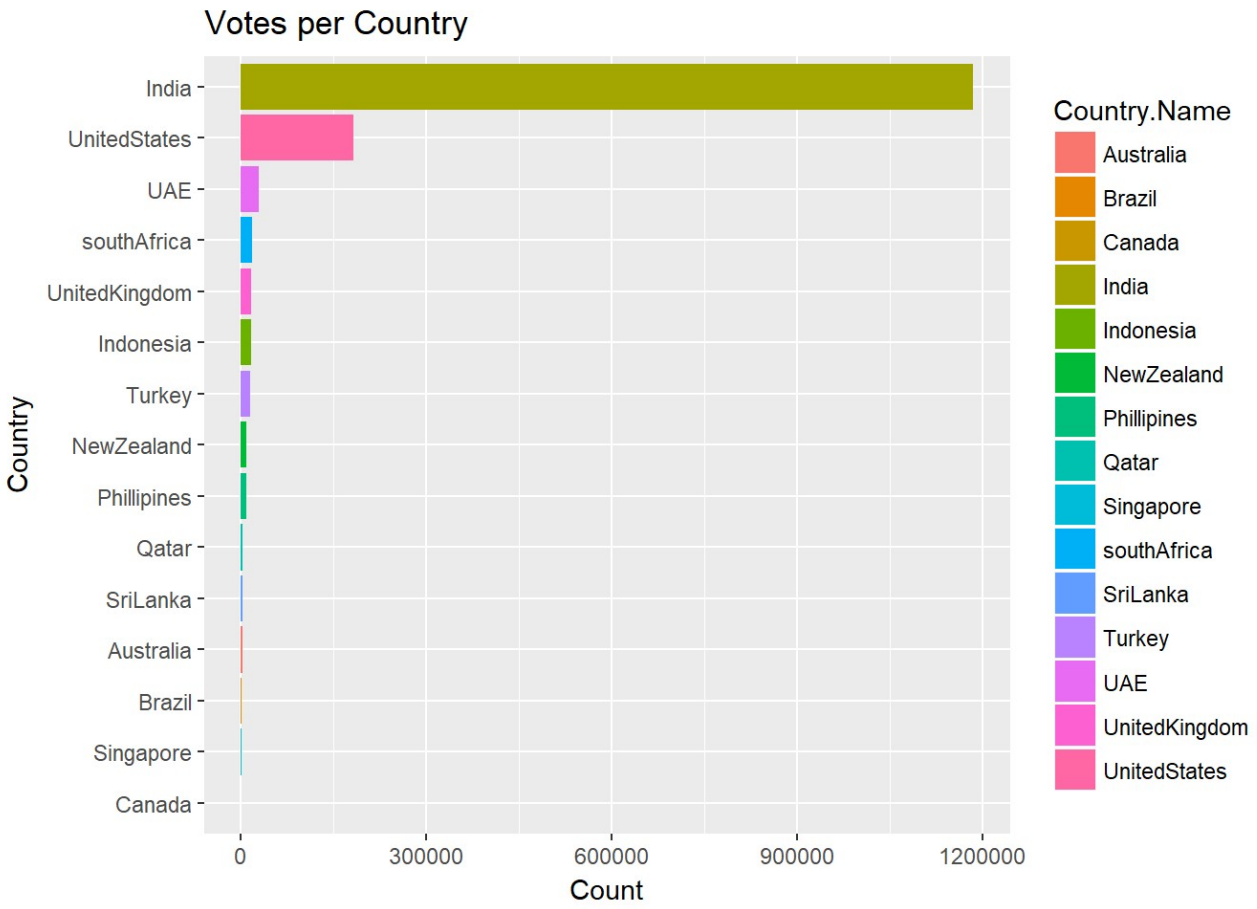
From this graph we can say that maximum restaurants are presents in New Delhi. Then, Gurgaon, Noida and Faridabad. There are only few restaurants presents in other cities.

Now we will see the relationship between number of votes and country.

```
library(dplyr)
detach("package:plyr", unload=TRUE)
```

```
## Warning: 'plyr' namespace cannot be unloaded:
##   namespace 'plyr' is imported by 'ggplot2', 'scales', 'reshape2', 'ggmap', 'broo
m' so cannot be unloaded
```

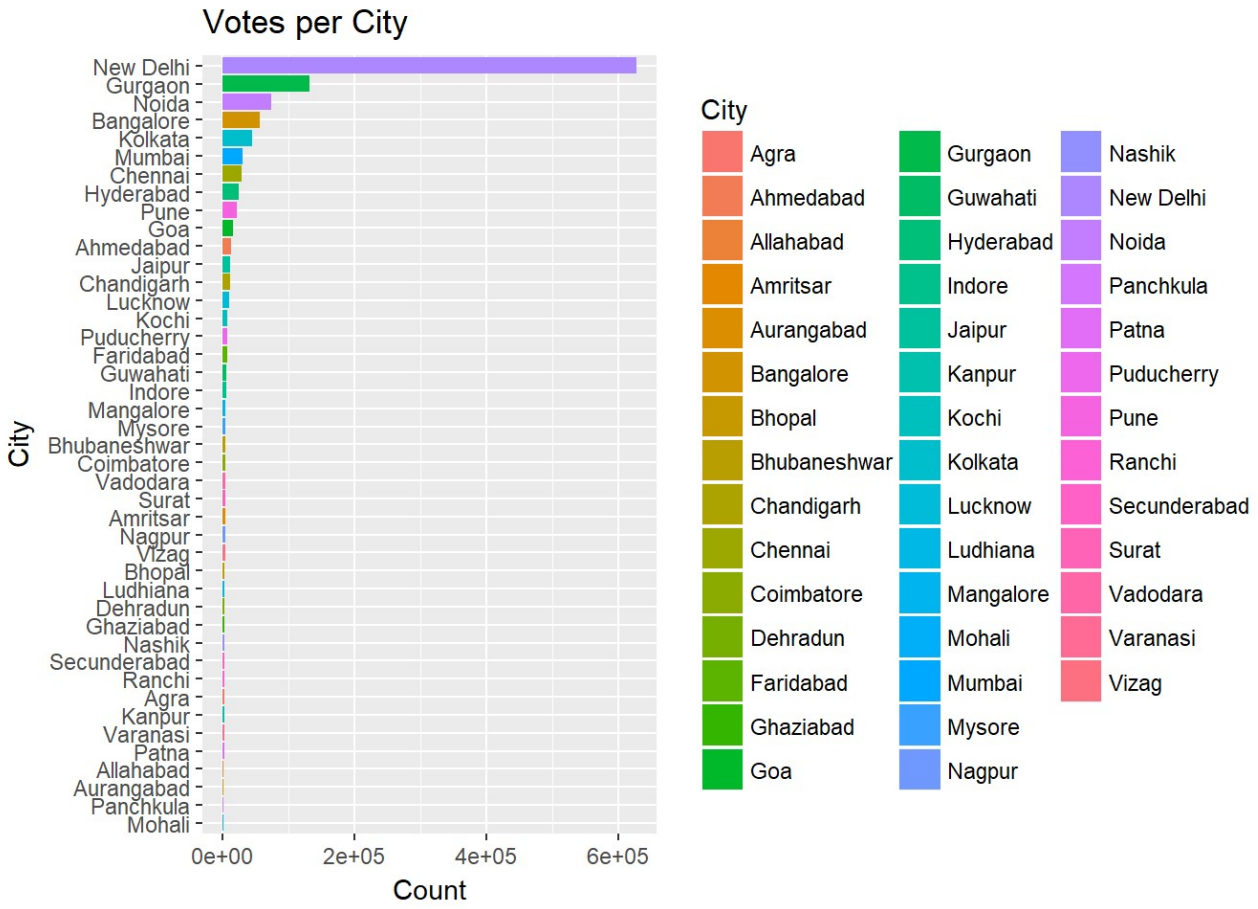
```
#if it gives the error "Error: This function should not be called directly" then detac
h the plyr package.
zomatofinal %>%
    filter(Rating.text != 'Not rated') %>%   #we did not consider Not rated rate for a
nalsis.
    group_by(Country.Name) %>%
    summarize( vote.count = sum(Votes), restaurant.count = n()) %>%
    ungroup() -> country.con



p2 <- country.con %>%
        ggplot(aes(x = reorder(Country.Name, vote.count), y = vote.count , fill = Coun
try.Name)) +
        geom_col() +
        coord_flip() +
        labs(title = 'Votes per Country', x= 'Country', y = 'Count')

p2
```



From this graph, we can say that there are more number of votes for restaurants which are located in India. We can also say that there are more restaurants presents in India. If there are more restaurants then they have more votes.

We will look, votes per city in India.

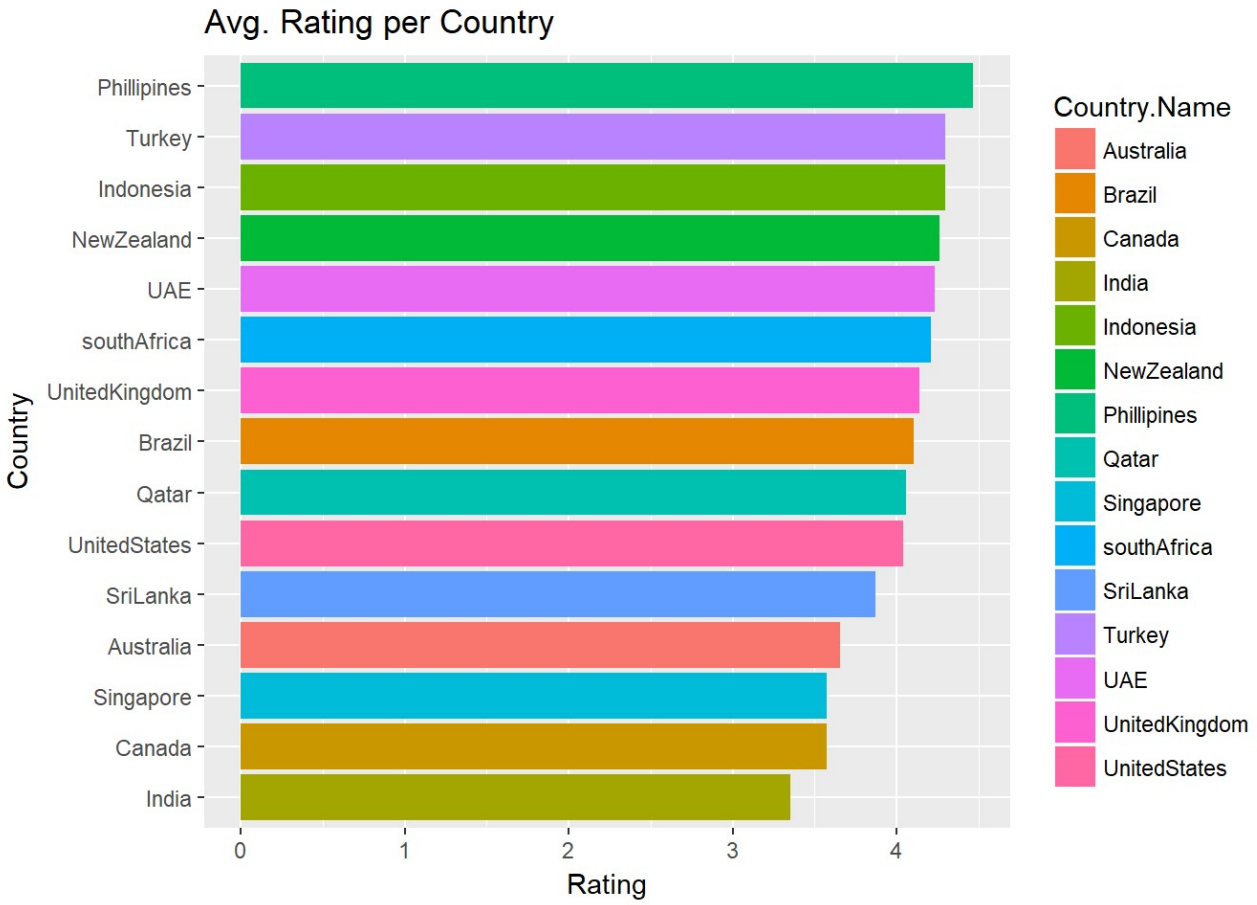
```
library(dplyr)
#if it gives the error "Error: This function should not be called directly" then detac
h the plyr package.
zomatoIndia %>%
    filter(Rating.text != 'Not rated') %>%   #we did not consider Not rated rate for a
nalsis.
    group_by(City) %>%
    summarize( vote.count = sum(Votes), restaurant.count = n()) %>%
    ungroup() -> country.con



p2 <- country.con %>%
        ggplot(aes(x = reorder(City, vote.count), y = vote.count , fill = City)) +
        geom_col() +
        coord_flip() +
        labs(title = 'Votes per City', x= 'City', y = 'Count')

p2
```



From this graph, we can say that New delhi's restaurants have the highest votes while Mohali's restaurants have the lowest votes.

Now we will see is there any relationship between aggregate rating and country

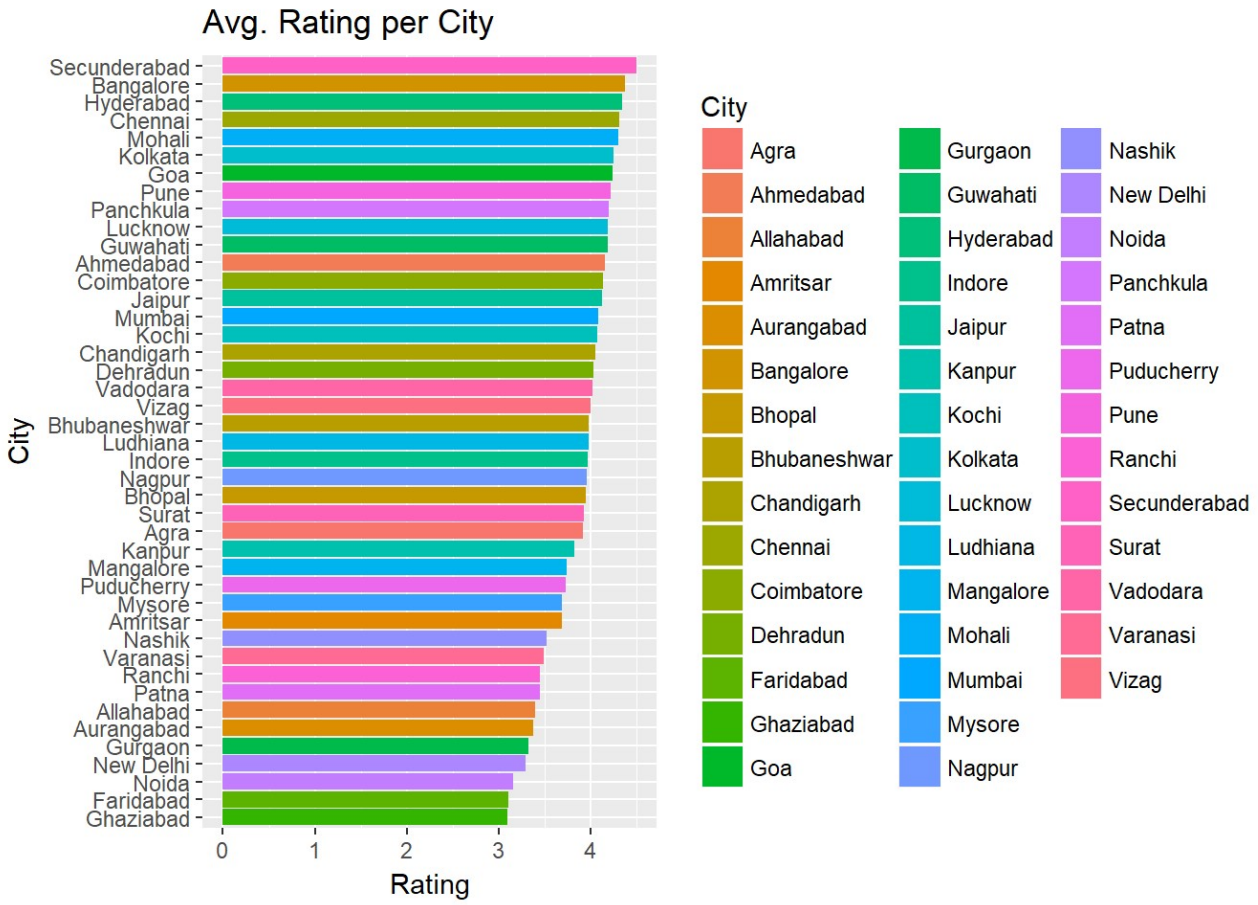
```
p3 <- zomatofinal %>%
        filter(Rating.text != 'Not rated') %>%
        group_by(Country.Name) %>%
        summarize(avg.rating = mean(Aggregate.rating)) %>%
        ungroup() %>%
        ggplot(aes(x = reorder(Country.Name, avg.rating), y = avg.rating, fill = Count
ry.Name)) +
        geom_col() +
        coord_flip() +
        labs(title = 'Avg. Rating per Country', x= 'Country', y = 'Rating')

p3
```



From this graph, we can say that Philipines has highest avg Rating while India has the lowest avg rating.We already saw above that there are less number of restaurants in Philipines so there are more chances that avg rating of philipies' restaurants are higher compared to maximum restaurants in any others countries.

There are maximum restaurants presents in India, So we will find avg rating per city in India.

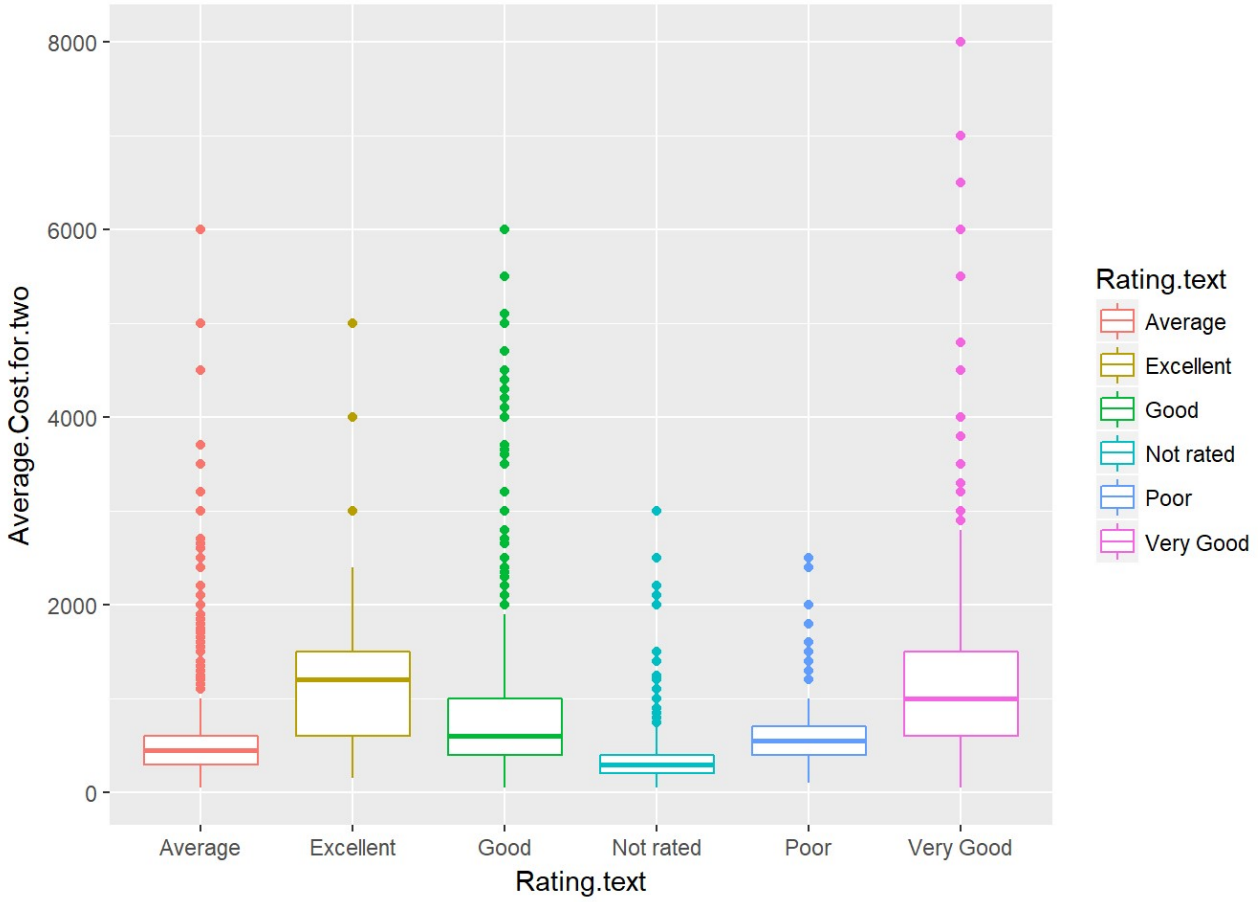
```
p3 <- zomatoIndia %>%
        filter(Rating.text != 'Not rated') %>%
        group_by(City) %>%
        summarize(avg.rating = mean(Aggregate.rating)) %>%
        ungroup() %>%
        ggplot(aes(x = reorder(City, avg.rating), y = avg.rating, fill = City)) +
        geom_col() +
        coord_flip() +
        labs(title = 'Avg. Rating per City', x= 'City', y = 'Rating')

p3
```



From this plot, we can say that Secunderabad has the highest avg rating in India.while in New delhi, Gurgaon, Noida, and Faridabad have the highest restaurants but they have lowest avg rating per city.

Generally people believe that, costly restaurants provide good service.

```
library(ggplot2)
ggplot(zomatoIndia, aes(x = Rating.text, y = Average.Cost.for.two, color= Rating.tex
t)) +
  geom_boxplot()+
    scale_fill_manual(values=c("red", "orange", "yellow","green","blue","purple"))
```

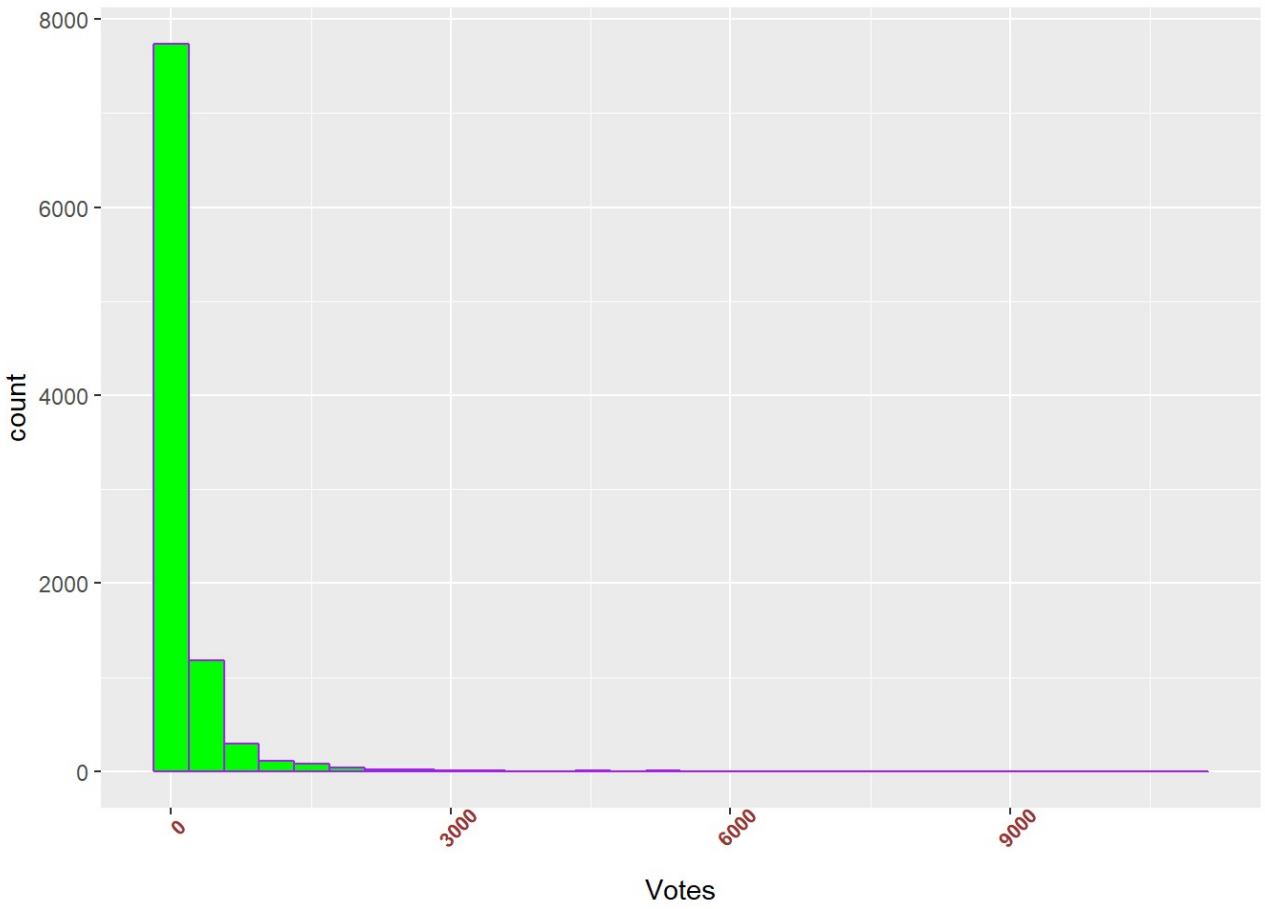


From this boxplot, we can say that which restaurant has higher price for two person these restaurants provides good service. Median value of “Avg cost for two” is increases with the imporovement of Rating.text

Now we will discover the distrubution of votes.

```
library(ggplot2)
ggplot(zomatofinal) +
   geom_histogram(aes(x=Votes), color="purple", fill="green") +
   theme(axis.text.x = element_text(face="bold", color="#993333", size=8, angle=45))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Here, from this histogram we can see that the number of voters are less as count of votes increase.

Is there any relationship between price range and aggregate rating?

```
library(ggplot2)
a<- ggplot(zomatofinal, aes(x=Price.range, y=Aggregate.rating)) +
stat_smooth(method = "loess", colour = "red", size = 1) +
   xlab("Price Range") + ylab("Aggregate Rating")
a
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at 0.985
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 1.015
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 5.3665e-028
```

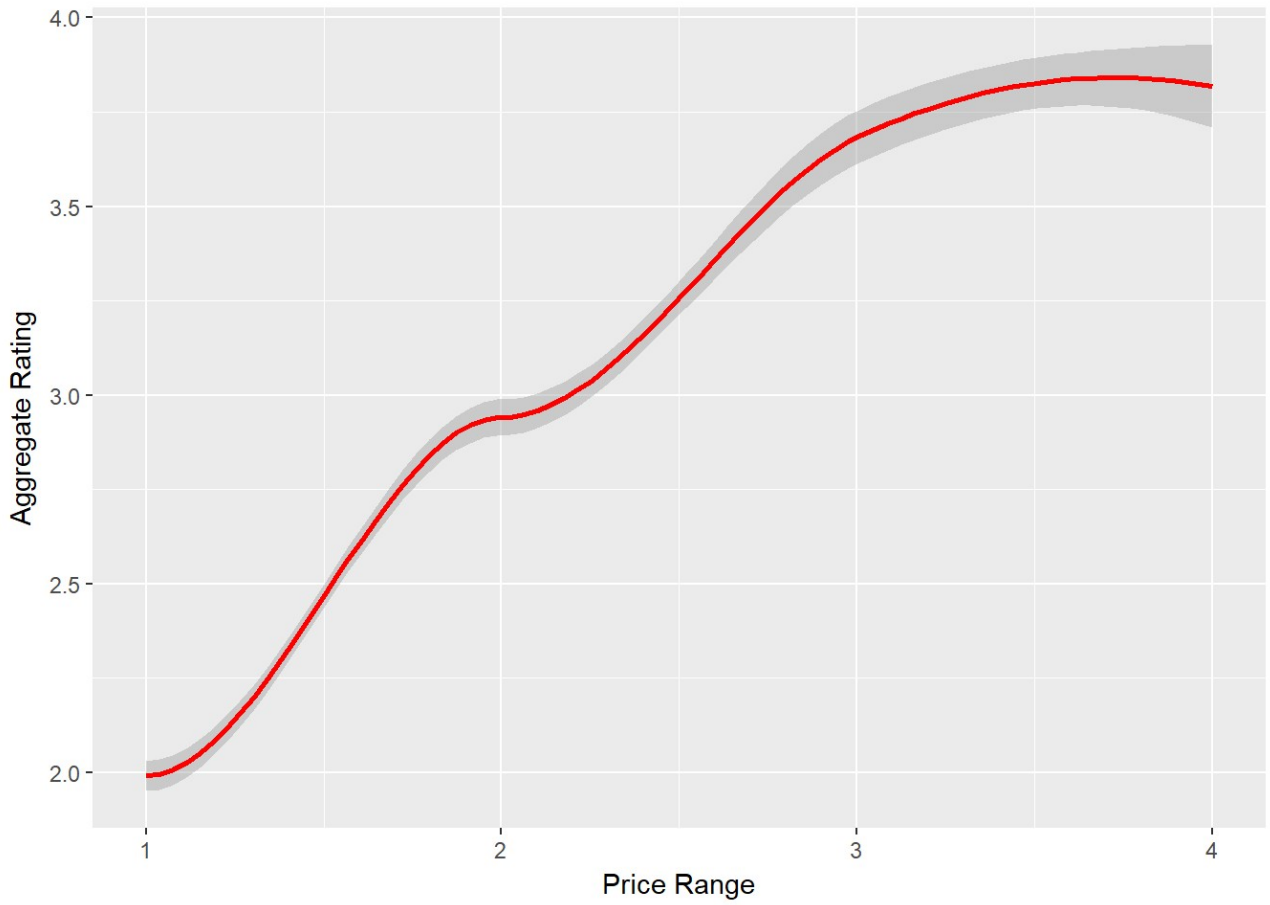
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : There are other near singularities as well. 1
```

```
## Warning in predLoess(object$y, object$x, newx = if
## (is.null(newdata)) object$x else if (is.data.frame(newdata))
## as.matrix(model.frame(delete.response(terms(object)), : pseudoinverse used
## at 0.985
```

```
## Warning in predLoess(object$y, object$x, newx = if
## (is.null(newdata)) object$x else if (is.data.frame(newdata))
## as.matrix(model.frame(delete.response(terms(object)), : neighborhood radius
## 1.015
```

```
## Warning in predLoess(object$y, object$x, newx = if
## (is.null(newdata)) object$x else if (is.data.frame(newdata))
## as.matrix(model.frame(delete.response(terms(object)), : reciprocal
## condition number 5.3665e-028
```

```
## Warning in predLoess(object$y, object$x, newx = if
## (is.null(newdata)) object$x else if (is.data.frame(newdata))
## as.matrix(model.frame(delete.response(terms(object)), : There are other
## near singularities as well. 1
```



From this plot we can conclude that which restaurant has more average rating they have higher price.As price range is increasing then average rating is also increasing.

Is there any relationship between price range and votes?

```
library(ggplot2)
a<- ggplot(zomatofinal, aes(x=Price.range, y=Votes)) +
stat_smooth(method = "loess", colour = "blue", size = 1) +
   xlab("Price Range") + ylab("votes")
a
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at 0.985
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 1.015
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 5.3665e-028
```

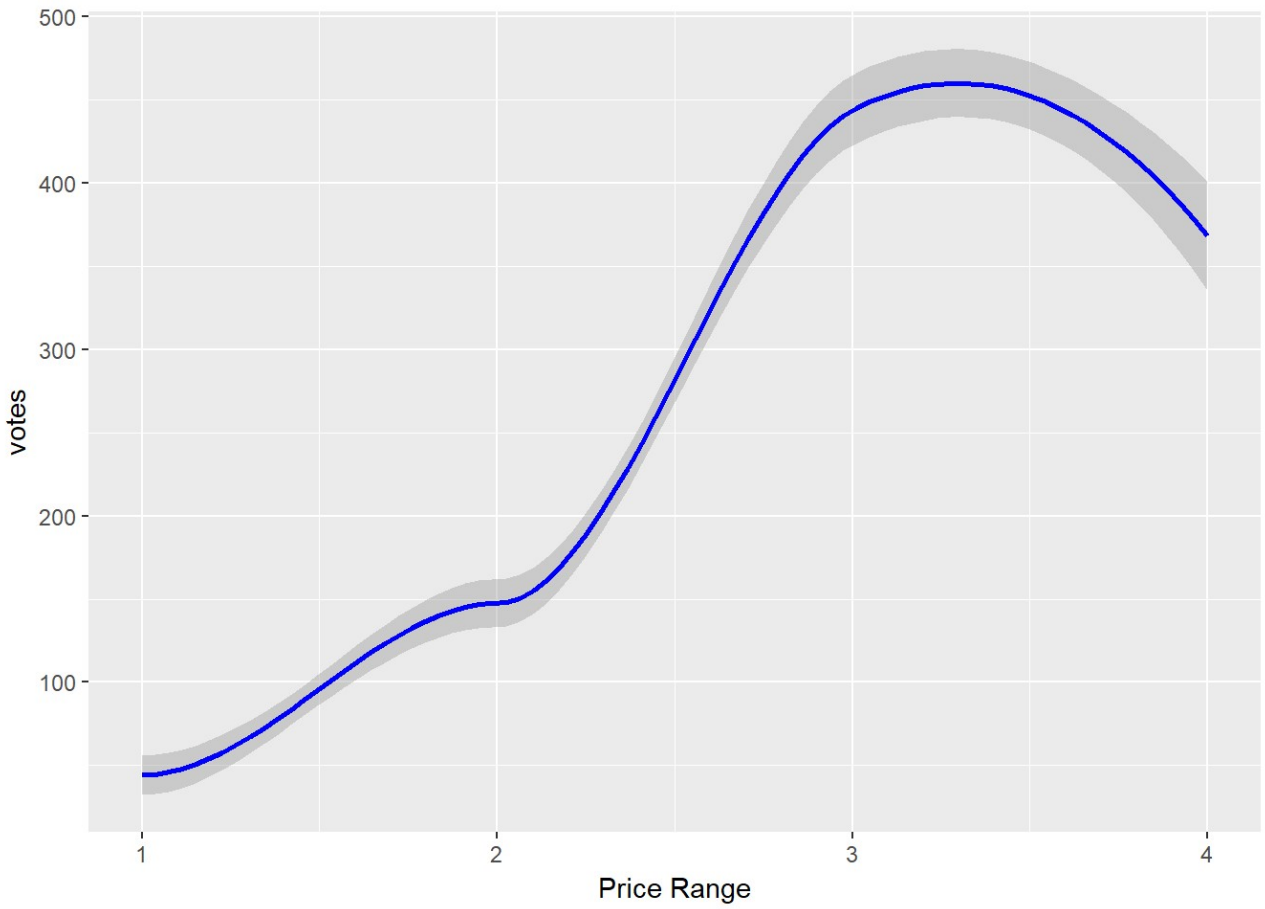
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : There are other near singularities as well. 1
```

```
## Warning in predLoess(object$y, object$x, newx = if
## (is.null(newdata)) object$x else if (is.data.frame(newdata))
## as.matrix(model.frame(delete.response(terms(object)), : pseudoinverse used
## at 0.985
```

```
## Warning in predLoess(object$y, object$x, newx = if
## (is.null(newdata)) object$x else if (is.data.frame(newdata))
## as.matrix(model.frame(delete.response(terms(object)), : neighborhood radius
## 1.015
```

```
## Warning in predLoess(object$y, object$x, newx = if
## (is.null(newdata)) object$x else if (is.data.frame(newdata))
## as.matrix(model.frame(delete.response(terms(object)), : reciprocal
## condition number 5.3665e-028
```

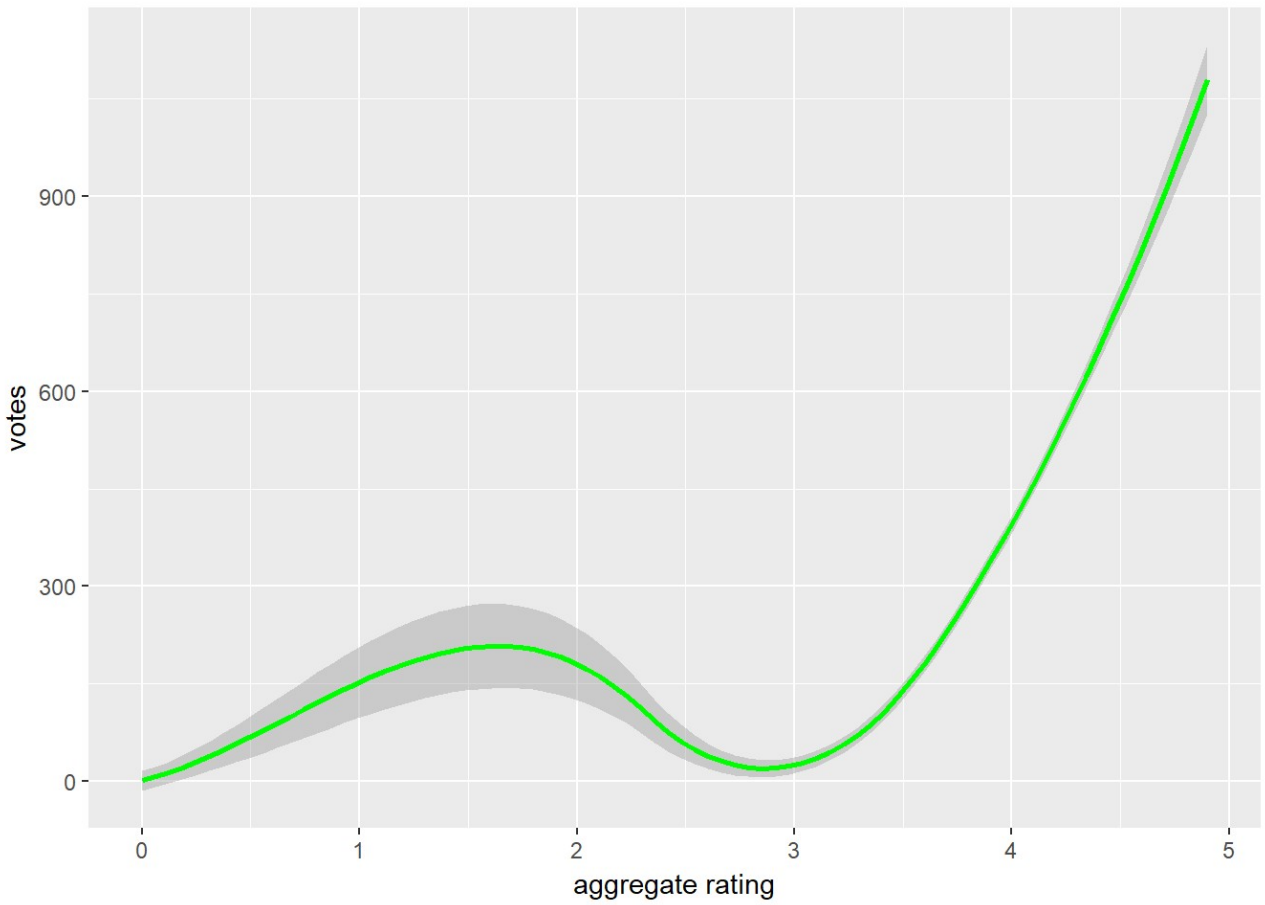
```
## Warning in predLoess(object$y, object$x, newx = if
## (is.null(newdata)) object$x else if (is.data.frame(newdata))
## as.matrix(model.frame(delete.response(terms(object)), : There are other
## near singularities as well. 1
```



From this graph we can see that as price range increases, number of votes also increases. But, number of votes are decreasing after price range is at 3.

Is there any relationship between number of votes and aggregate rating?

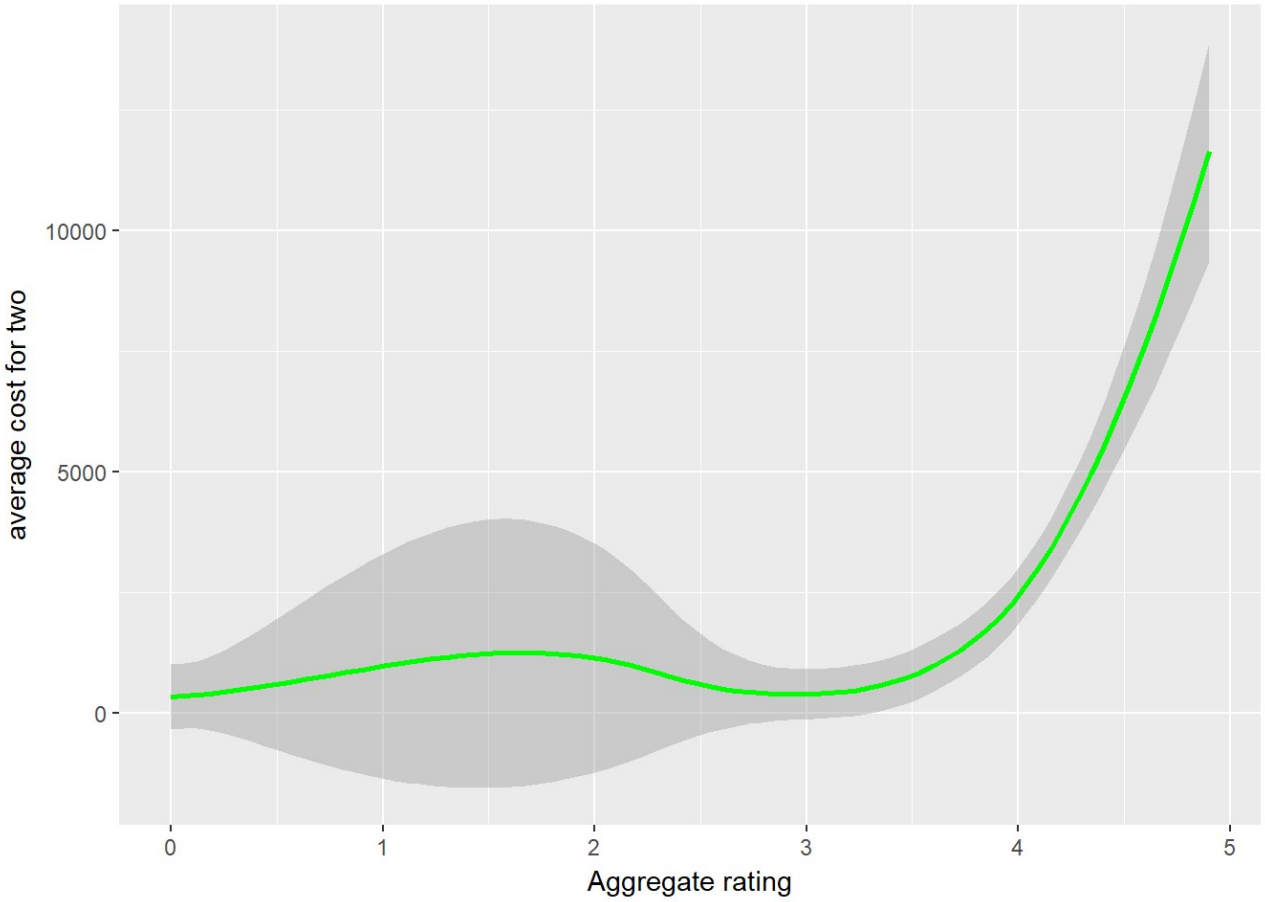
```
library(ggplot2)
a<- ggplot(zomatofinal, aes(x=zomatofinal$Aggregate.rating, y=zomatofinal$Votes)) +
stat_smooth(method = "loess", colour = "green", size = 1) +
   xlab("aggregate rating") + ylab("votes")
a
```



From this graph we can say that there is a relationship between aggreagte rating and number of votes. As the aggregate rating increases, number of votes also increases. But, from aggregate rating at 1.5 to 3, the number of votes are decreasing. After aggregate rating at 3 , the number of voter increases.

Is there any relationship between average cost for two and aggregate rating?

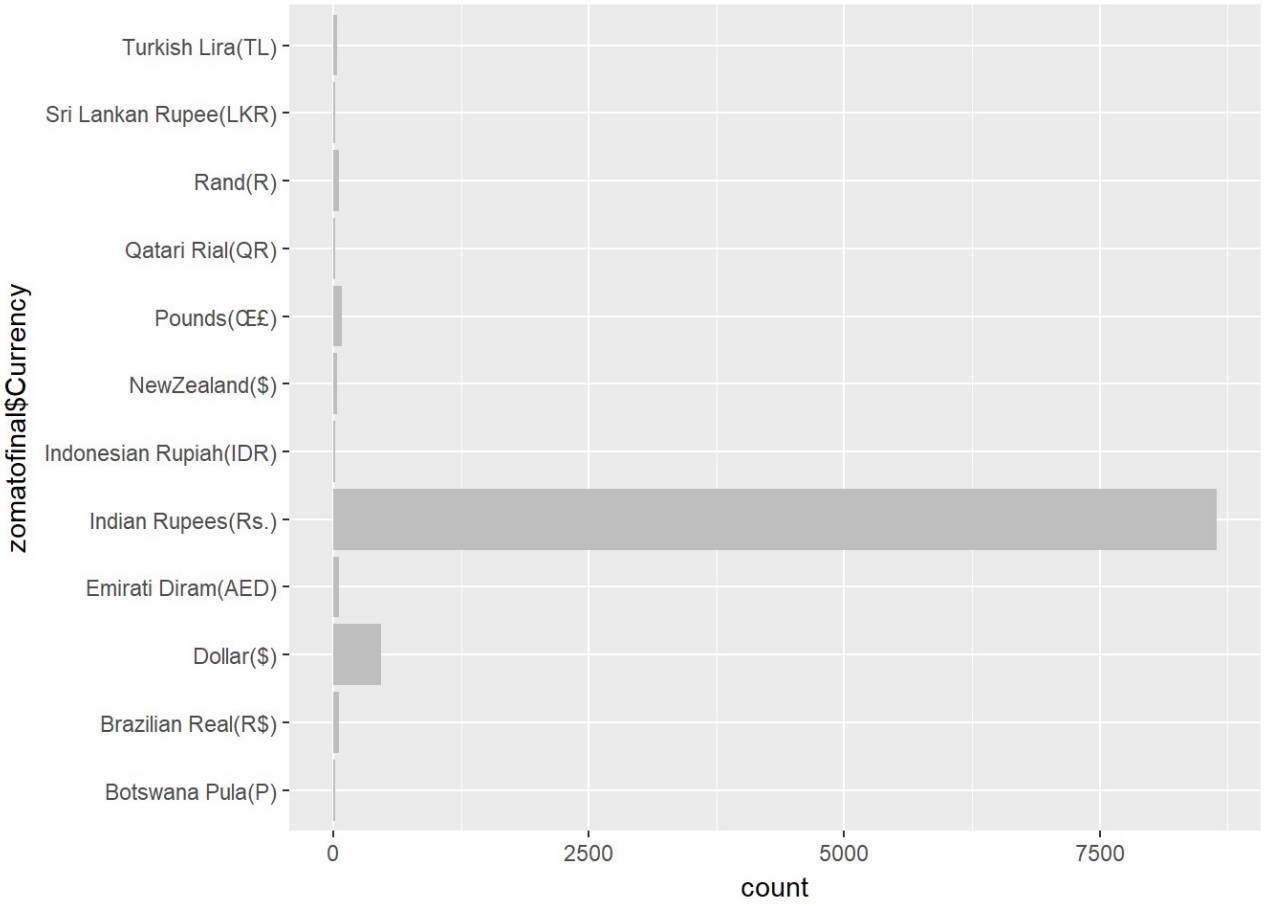
```
library(ggplot2)
a<- ggplot(zomatofinal, aes(y=zomatofinal$Average.Cost.for.two, x=zomatofinal$Aggregat
e.rating)) +
stat_smooth(method = "loess", colour = "green", size = 1) +
   xlab("Aggregate rating") + ylab("average cost for two")
a
```



From this graph, we can say that as aggregate rating increases, average cost for two is also increasing.So, we conclude that Which restaurants have higher aggregate rating, they are expensive which means cost for two people is more.

Here, we will determine the most accepted currency by restauants.

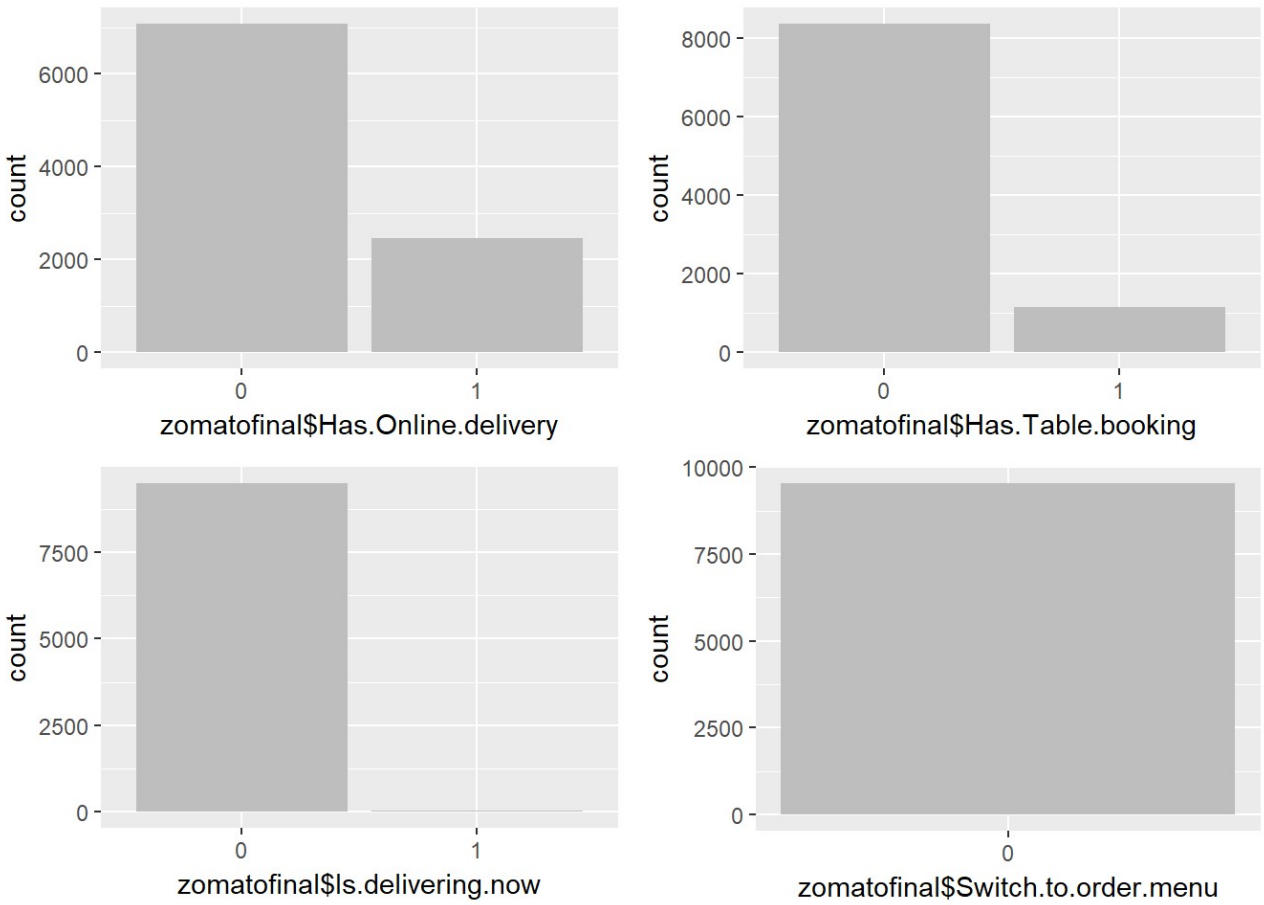
```
library(ggplot2)
ggplot(zomatofinal) +
  geom_bar(aes(x = zomatofinal$Currency), fill = "gray")+coord_flip()
```



From the above graph, we can say that the Indian Rupees are most accepted by restaurants. So, we can also say that most of the restaurants that are registered in Zomato, are from India.

Now, we will determine some plots regarding restaurant's Service Feature.

```
library(ggplot2)
library(gridExtra)
a<-ggplot(zomatofinal) +
  geom_bar(aes(x = zomatofinal$Has.Online.delivery), fill = "gray")

b<-ggplot(zomatofinal) +
  geom_bar(aes(x = zomatofinal$Has.Table.booking), fill = "gray")

c<-ggplot(zomatofinal) +
  geom_bar(aes(x = zomatofinal$Is.delivering.now), fill = "gray")

d<-ggplot(zomatofinal) +
  geom_bar(aes(x = zomatofinal$Switch.to.order.menu), fill = "gray")

grid.arrange(a,b,c,d ,widths= c(2,2))
```



From the above graph, we can say that most of the restaurants do not provide online delievery facility. Only few restaurants provide online delivery facility.

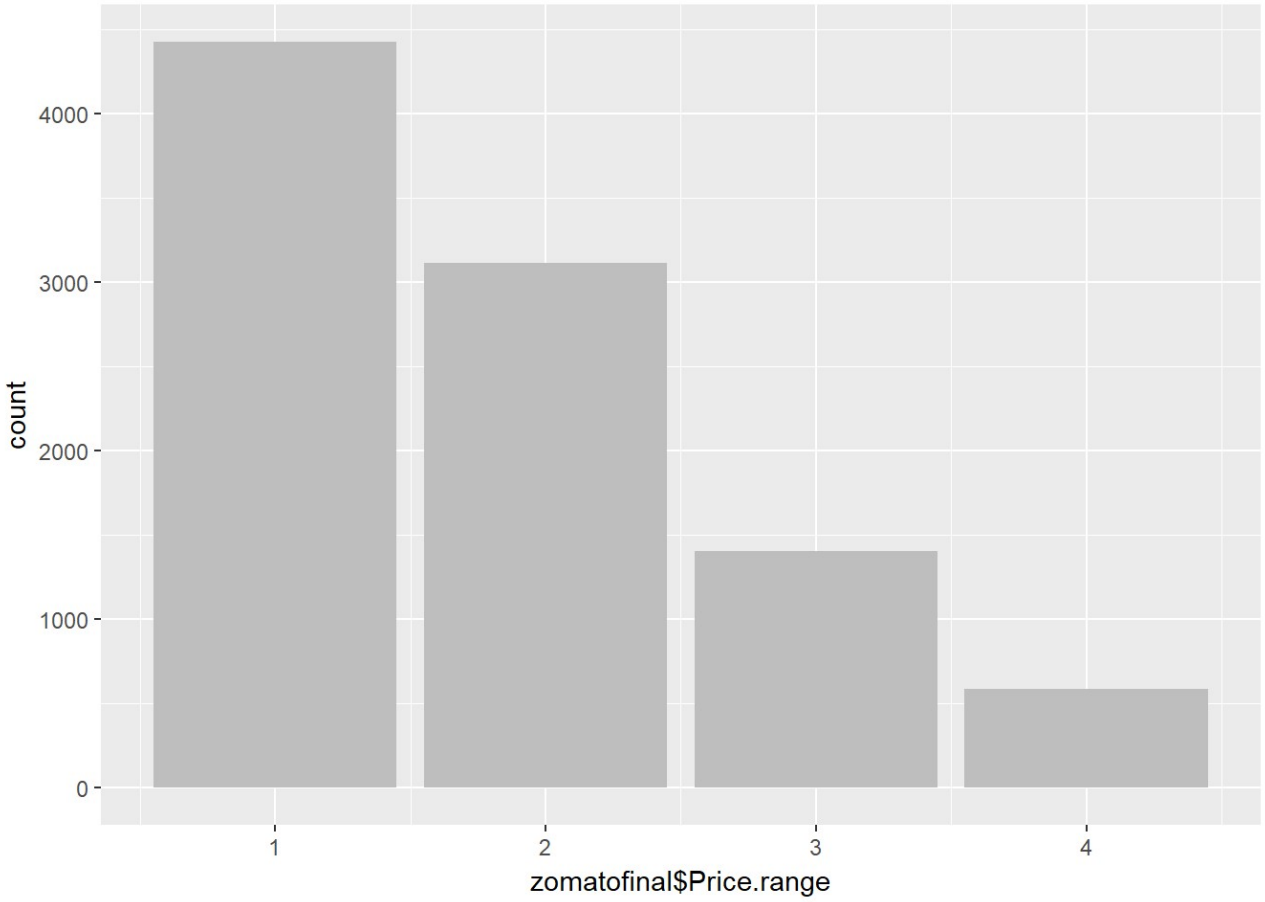
From the above graph, we can say that most of the restaurants do not provide a table booking facility. Only few of the restaurants provide this facility.

From the above graph, we can say that almost all the restaurants do not provide home delievery facility. Almost 99% of the restaurants do not provide home delievery facility.

From the abpve graph, we can say that no restaurants provide switch to order menu facility.

Now we will determine the price range of restaurant.what are the price range of restaurant?

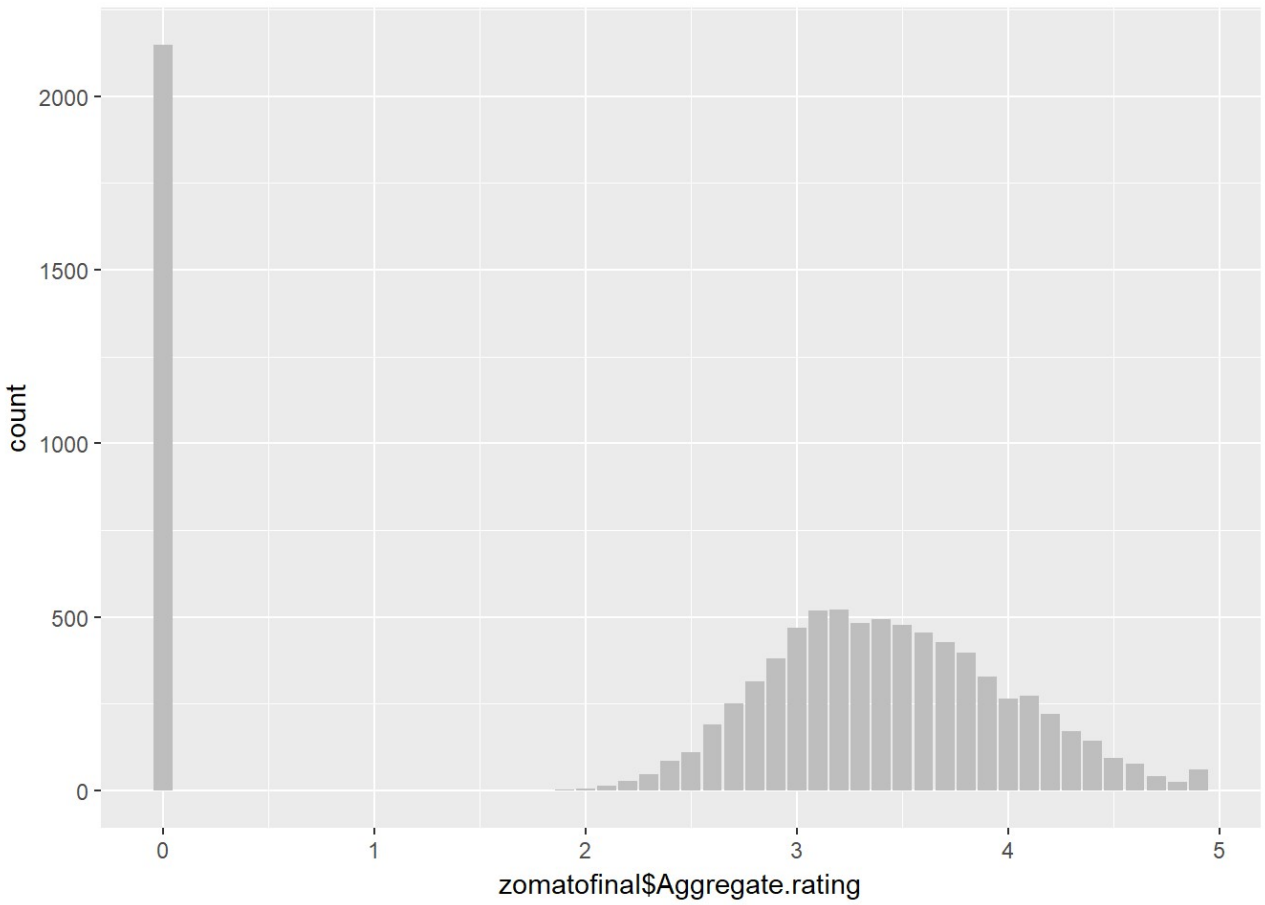
```
library(ggplot2)
ggplot(zomatofinal) +
  geom_bar(aes(x = zomatofinal$Price.range), fill = "gray")
```



From this graph, we can see that most of the restaurant have price range 1, as the price range increase, number of restaurant decreases. So, number of restaurants are more, which have lower price range.

We will see the distrubution of aggregate rating of restaurant.

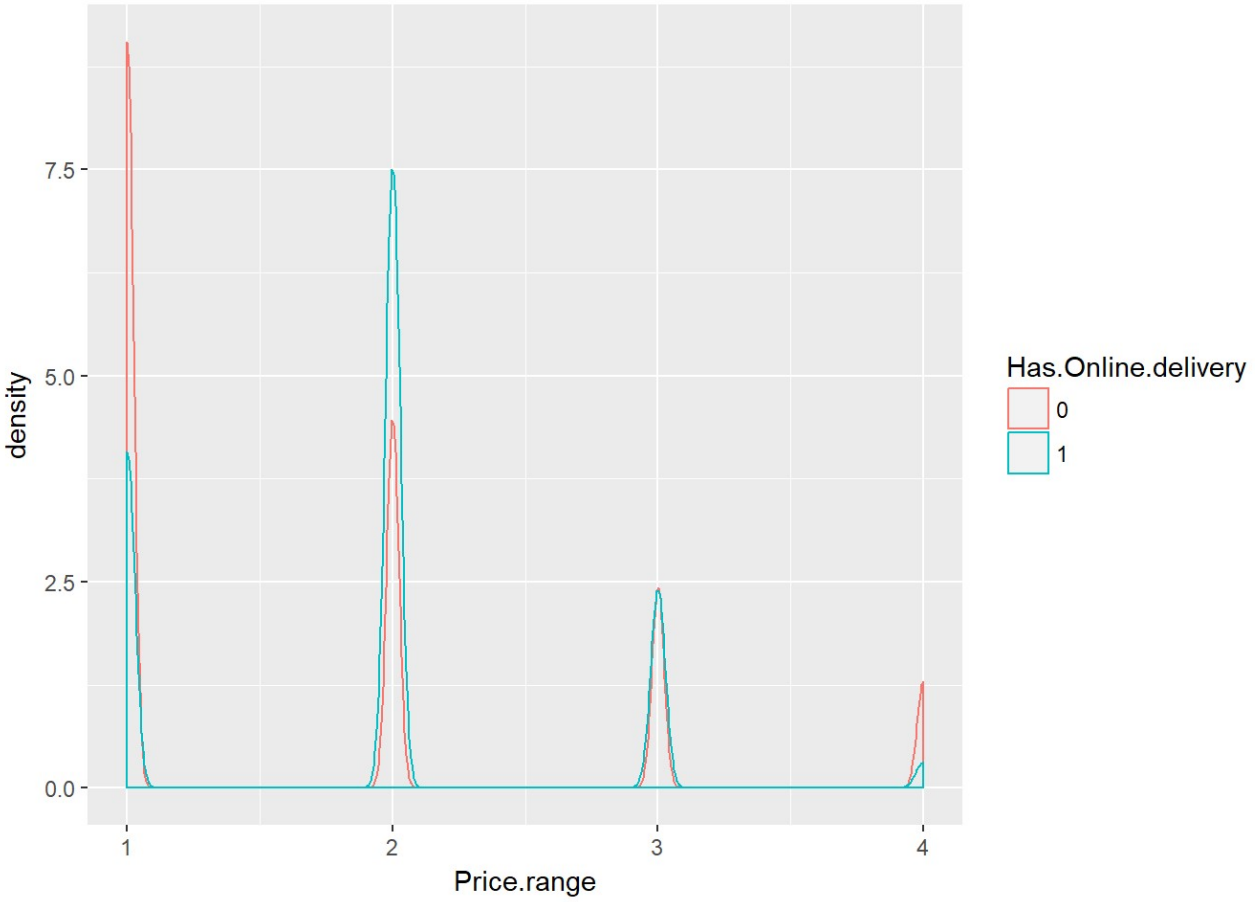
```
library(ggplot2)
ggplot(zomatofinal) +
  geom_bar(aes(x = zomatofinal$Aggregate.rating), fill = "gray")
```



From this graph we can see that majority of restaurants have 0 aggregate rating and there is no aggregate rating between 0 to 2. From 2 to 5, distrubution of aggregate rating looks like a normal distrubution.

Now, we will check if there is any relationship between the price range of a restaurant and the delievery.

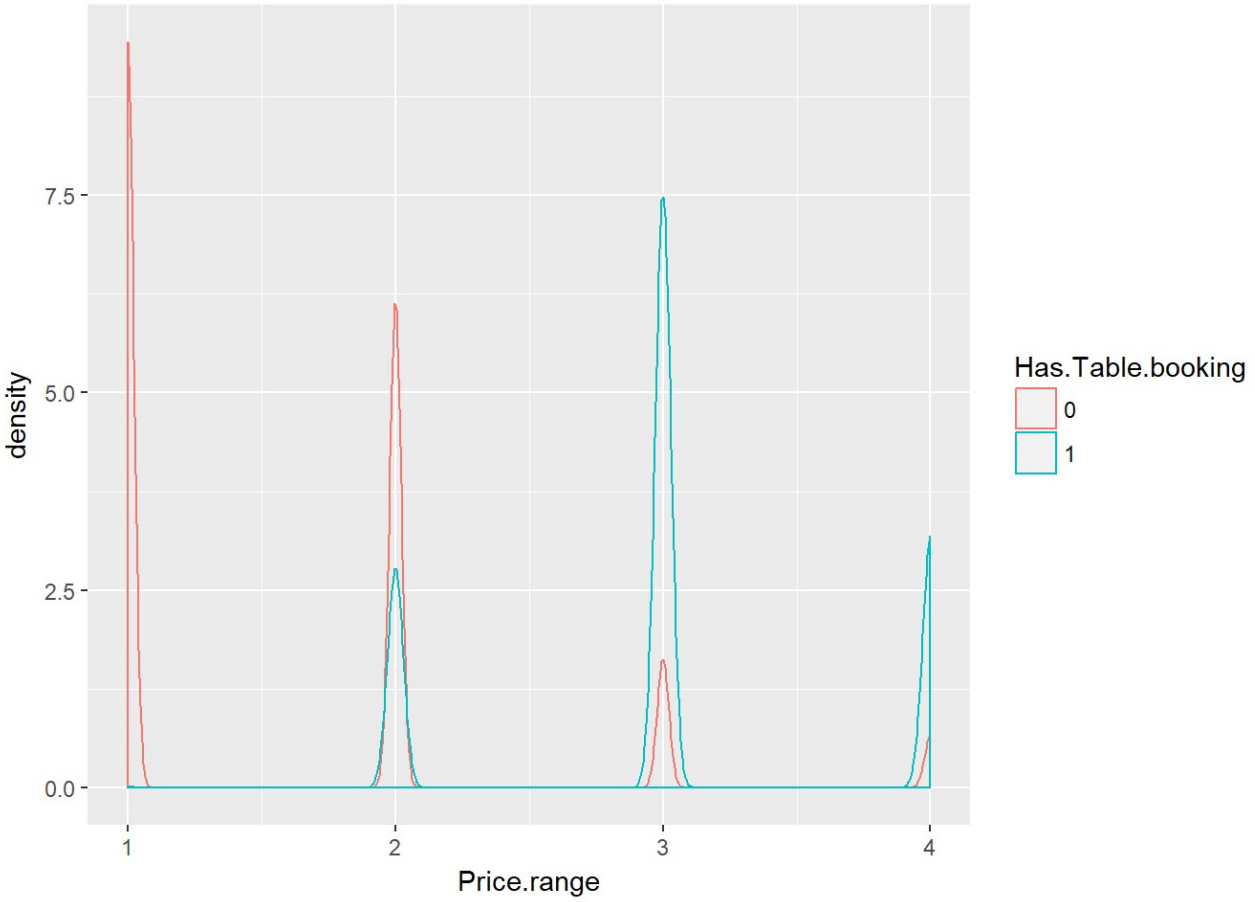
```
library(ggplot2)
ggplot(zomatofinal,aes(Price.range,color=Has.Online.delivery))+ geom_density(adjust=1/
5)
```



From the above plot, we can say that the restauraants which have the lowest and the higheat price range, do not provide delievery and the restaurants which have medium price range, provide delievery.

Now, we will see if there is any relationship between table booking and price range.

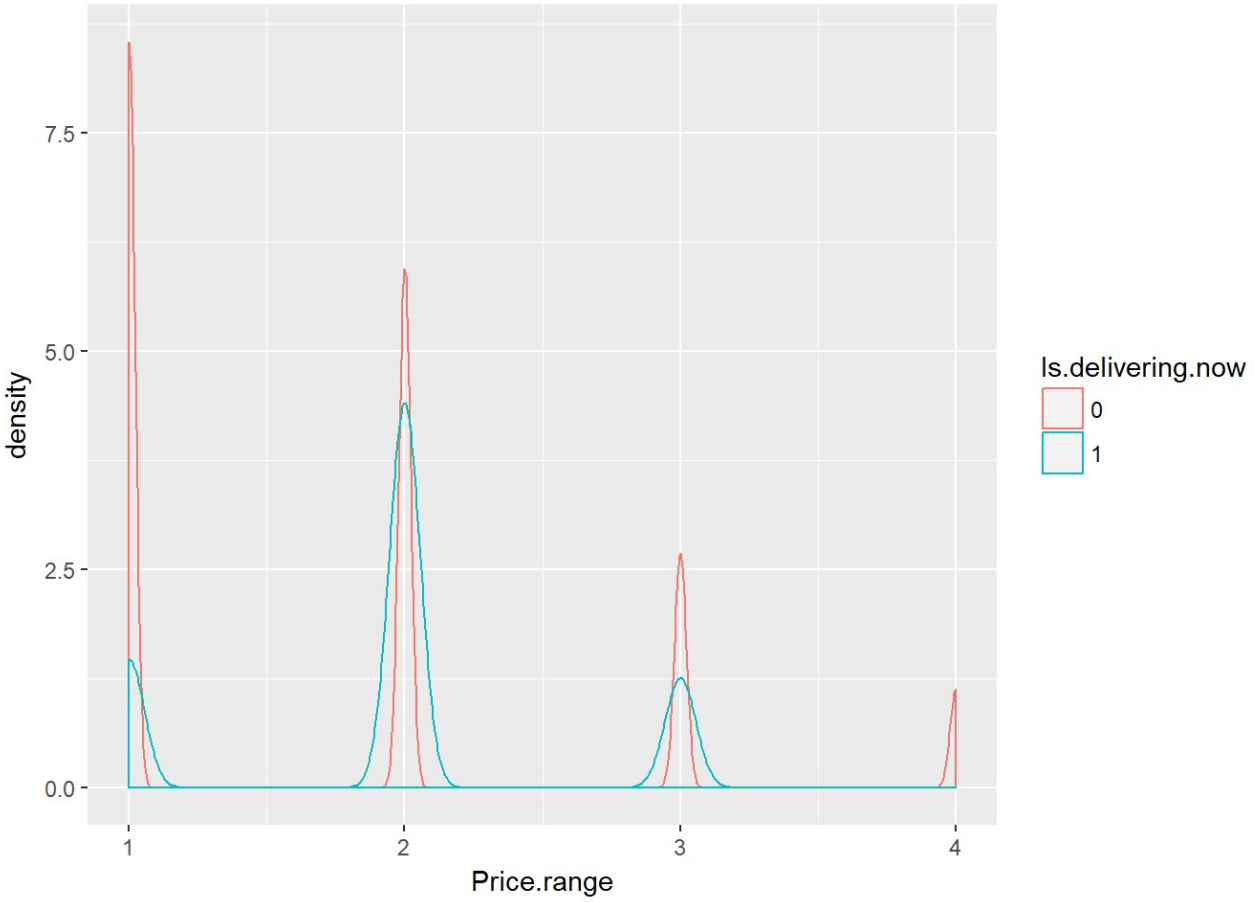
```
library(ggplot2)
ggplot(zomatofinal,aes(Price.range,color=Has.Table.booking))+ geom_density(adjust=1/5)
```



From this plot, we can say that restaurants which have the lower price range, do not provide table booking facility while higher price range restaurants provide table booking facility.

Now, we will see if there is any relationship between delivery facility and price range.

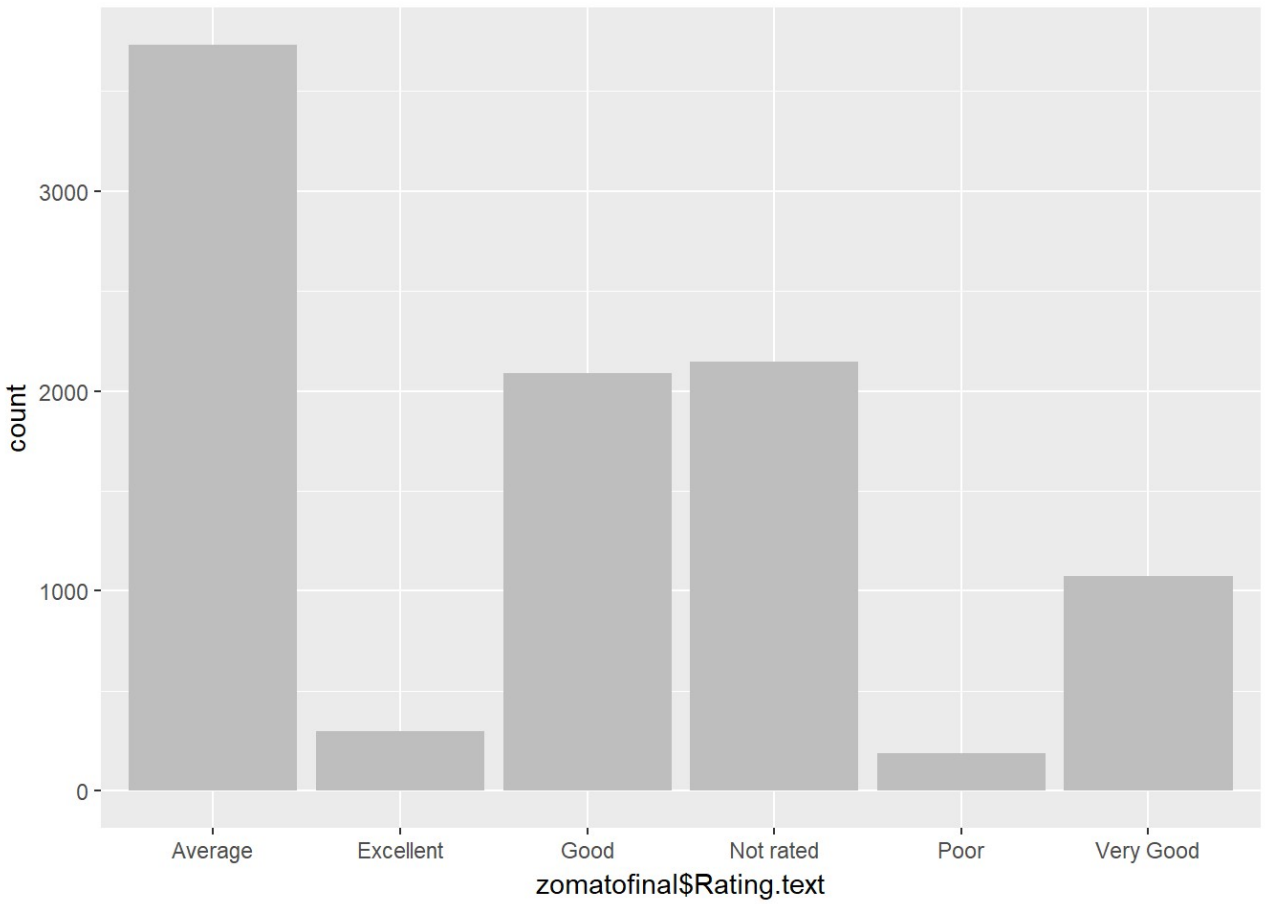
```
library(ggplot2)
ggplot(zomatofinal,aes(Price.range,color=Is.delivering.now))+ geom_density(adjust=1/5)
```



from this plot, we can say that which restaurants have the higher price range they do not provide delivery facility.We can say that restaurants which have medium and lowest price range,provide delivery facility.

Now, we will see the text rating of restaurant.

```
library(ggplot2)
ggplot(zomatofinal) +
  geom_bar(aes(x = zomatofinal$Rating.text), fill = "gray")
```



From this graph, we can see that there are only few restaurants which have excellent and poor ratings. Most of restaurants have average rating.The ratings for good and not rated restaurants are almost equal.

Now we will see is there any relationship between text rating and aggregate rating?
Refrence:https://www.r-bloggers.com/aggregate-a-powerful-tool-for-data-frame-in-r/ (https://www.r-bloggers.com/aggregate-a-powerful-tool-for-data-frame-in-r/)

```
Textrating <- aggregate(zomatofinal$Aggregate.rating, by=list(Rating=zomatofinal$Ratin
g.text), FUN=mean)
Textrating <- as.data.frame(Textrating)
Textrating[order(Textrating$x, -Textrating$Rating, decreasing = TRUE), ]
```

```
## Warning in Ops.factor(Textrating$Rating): '-' not meaningful for factors
```

```
##      Rating        x
## 2 Excellent 4.658863
## 6 Very Good 4.167814
## 3      Good 3.682990
## 1   Average 3.051179
## 5      Poor 2.297849
## 4 Not rated 0.000000
```

From this table we can conclude that, text rating and aggregate rating indicate the same thing.

We will see is there any relationshio between rating color and aggregate rating? Refrence: https://www.r-bloggers.com/aggregate-a-powerful-tool-for-data-frame-in-r/ (https://www.r-bloggers.com/aggregate-a-powerful-tool-for-data-frame-in-r/)

```
Rating_Color <- aggregate(zomatofinal$Aggregate.rating, by=list(Rating=zomatofinal$Rat
ing.color), FUN=mean)
Rating_Color [order(Rating_Color$x, -Rating_Color$Rating, decreasing = TRUE), ]
```

```
## Warning in Ops.factor(Rating_Color$Rating): '-' not meaningful for factors
```

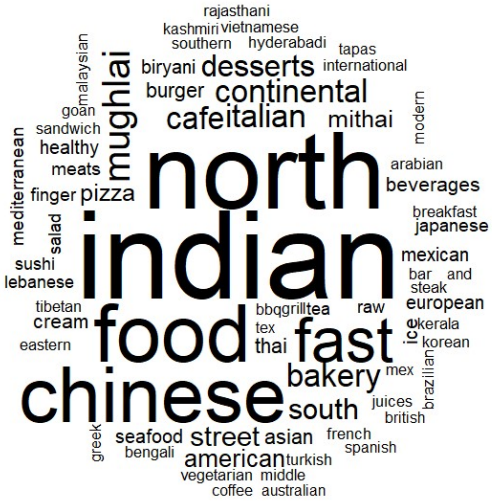
```
##       Rating        x
## 1 Dark Green 4.658863
## 2      Green 4.167814
## 6     Yellow 3.682990
## 3     Orange 3.051179
## 4        Red 2.297849
## 5      White 0.000000
```

From this table we can say that dark green color indicate excellent rating while white color indicate poor rating.

Word Cloud: Word clouds are a method for visually presenting text data. They are popular for text analysis because they make it easy to spot word frequencies. The more frequent the work is used, the larger and bolder it is displayed.

Now we will see which cusinies are popular among restaurants.We will build wordcloud to identify it. Refrence:http://www.sthda.com/english/wiki/text-mining-and-word-cloud-fundamentals-in-r-5-simple-steps-you-should-know (http://www.sthda.com/english/wiki/text-mining-and-word-cloud-fundamentals-in-r-5-simple-steps-you-should-know)

```
library("tm")
library("SnowballC")
library("wordcloud")
library("RColorBrewer")
docs1 <- Corpus(VectorSource(zomatofinal$Cuisines))
wordcloud(docs1, min.freq = 15, max.words = 100,  random.order = FALSE)
```



From this wordcloud we can conclude that north indian, chinese, fast food available in most of restaurants.

Now we will extract the words from wordcloud and try to identify the frequency of words. Refrence: <http://www.sthda.com/english/wiki/text-mining-and-word-cloud-fundamentals-in-r-5-simple-steps-you-should-know (http://www.sthda.com/english/wiki/text-mining-and-word-cloud-fundamentals-in-r-5-simple-steps-you-should-know) >

```
dtm <- TermDocumentMatrix(docs1)
m <- as.matrix(dtm)
v <- sort(rowSums(m),decreasing=TRUE)
d <- data.frame(word = names(v),freq=v)
d
```

```
##                        word freq
## indian                indian 4675
## north                  north 3963
## food                    food 2852
## chinese              chinese 2729
## fast                    fast 1986
## mughlai              mughlai  995
## italian              italian  760
## bakery                bakery  745
## continental     continental  732
## cafe                    cafe  704
## desserts            desserts  651
## south                  south  644
## street                street  561
## american            american  405
## pizza                  pizza  386
## mithai                mithai  380
## burger                burger  251
## asian                  asian  235
## thai                    thai  234
## beverages          beverages  228
## cream                  cream  224
## ice                      ice  224
## mexican              mexican  177
## biryani              biryani  177
## seafood              seafood  174
## healthy              healthy  150
## european            european  148
## japanese            japanese  135
## finger                finger  114
## meats                  meats  114
## raw                      raw  114
## mediterranean mediterranean  112
## salad                  salad   93
## sushi                  sushi   75
## lebanese            lebanese   69
## tea                      tea   68
## steak                  steak   62
## sandwich            sandwich   53
## tibetan              tibetan   44
## breakfast          breakfast   41
## bar                      bar   38
## bbq                      bbq   33
## eastern              eastern   31
## french                french   29
## juices                juices   29
## bengali              bengali   29
## brazilian          brazilian   28
```

```
## arabian             arabian   28
## modern               modern   27
## hyderabadi       hyderabadi   26
## grill                 grill   25
## southern           southern   24
## vegetarian       vegetarian   23
## turkish             turkish   23
## kerala               kerala   23
## middle               middle   22
## malaysian         malaysian   22
## korean               korean   21
## international international   21
## vietnamese       vietnamese   21
## rajasthani       rajasthani   21
## and                     and   20
## goan                   goan   20
## kashmiri           kashmiri   20
## coffee               coffee   19
## tapas                 tapas   19
## mex                     mex   19
## tex                     tex   19
## spanish             spanish   16
## australian       australian   16
## british             british   16
## greek                 greek   15
## indonesian       indonesian   14
## afghani             afghani   14
## african             african   14
## lucknowi           lucknowi   13
## pakistani         pakistani   12
## latin                 latin   11
## gujarati           gujarati   11
## chettinad         chettinad   11
## awadhi               awadhi   11
## filipino           filipino   10
## german               german   10
## cajun                 cajun   10
## western             western   10
## maharashtrian maharashtrian   10
## burmese             burmese   10
## andhra               andhra   10
## kebab                 kebab   10
## nepalese           nepalese    9
## contemporary   contemporary    9
## hawaiian           hawaiian    8
## parsi                 parsi    8
## naga                   naga    8
## southwestern   southwestern    7
## caribbean         caribbean    7
```

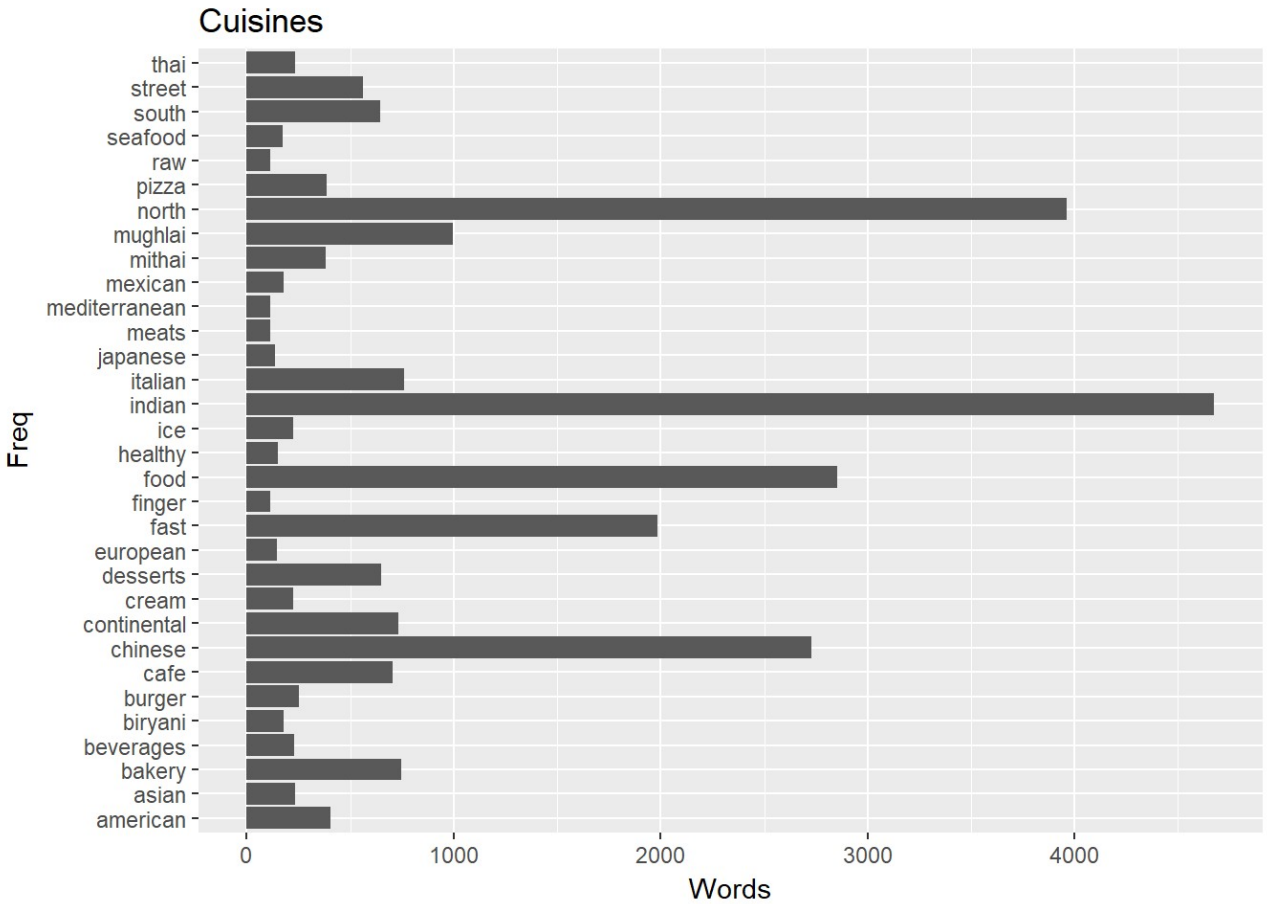
```
## portuguese       portuguese    7
## diner                 diner    6
## fusion               fusion    6
## bihari               bihari    6
## kiwi                   kiwi    6
## curry                 curry    6
## cuisine             cuisine    5
## moroccan           moroccan    5
## lankan               lankan    5
## sri                     sri    5
## singaporean     singaporean    4
## charcoal           charcoal    4
## assamese           assamese    4
## mangalorean     mangalorean    4
## patisserie       patisserie    4
## world                 world    4
## restaurant       restaurant    4
## dim                     dim    3
## sum                     sum    3
## deli                   deli    3
## armenian           armenian    3
## iranian             iranian    3
## sunda                 sunda    3
## scottish           scottish    3
## argentine         argentine    2
## teriyaki           teriyaki    2
## cuban                 cuban    2
## new                     new    2
## pub                     pub    2
## persian             persian    2
## cantonese         cantonese    2
## belgian             belgian    2
## drinks               drinks    2
## only                   only    2
## oriya                 oriya    2
## ramen                 ramen    2
## taiwanese         taiwanese    2
## izgara               izgara    2
## peruvian           peruvian    1
## gourmet             gourmet    1
## mineira             mineira    1
## canadian           canadian    1
## bubble               bubble    1
## irish                 irish    1
## soul                   soul    1
## malay                 malay    1
## malwani             malwani    1
## varies               varies    1
## peranakan         peranakan    1
```

```
## chips                chips    1
## fish                  fish    1
## durban              durban    1
## ner                    ner    1
## rek                    rek    1
```

From this output, we can say that indian, chinese and fast food are more popular.

Now we will plot the above output.

```
library(ggplot2)
library(tidyverse)
d <- subset(d, freq>=110)

d%>%
  mutate(name = fct_reorder(word, freq)) %>%
  ggplot( aes(x=word, y=freq)) +
  geom_bar(stat="identity") + xlab("Freq") + ylab("Words") + labs(title= "Cuisines") +
  coord_flip()
```



From this above plot, we can say that north indian, fast food and chinese food are more available in restaurants.

From the summary we know that there are more resturants located in New Delhi, Gurgaon and Noida.Now we will build wordcloud to identify in which city there are maximum resturants.From this wordclod we can conclude that new delhi, gurgaon and noida have maximum restaurants. Refrence:http://www.sthda.com/english/wiki/text-mining-and-word-cloud-fundamentals-in-r-5-simple-steps-you-should-know (http://www.sthda.com/english/wiki/text-mining-and-word-cloud-fundamentals-in-r-5-simple-steps-you-should-know)

```
docs <- Corpus(VectorSource(zomatofinal$City))
wordcloud(docs, min.freq = 15, colors=brewer.pal(8, "Dark2"), random.order = FALSE)
```

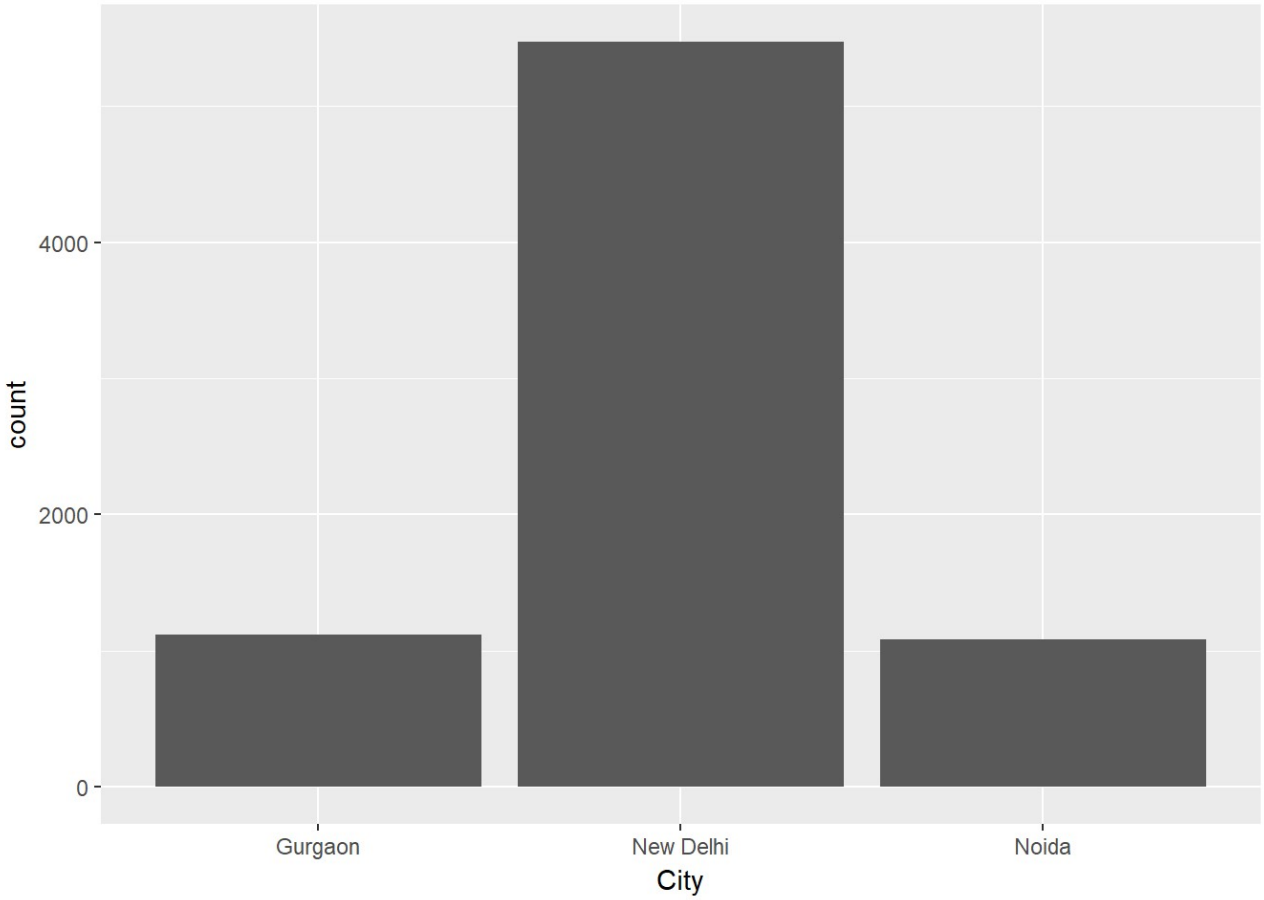
From this wordcloud we can get idea that most of restaurants are located in New Delhi, Noida, Gurgaon and faridabad. There are only few restaurants are there in other cities.

In zomato dataset, Over 80% of the data is for restaurants in Delhi, Gurugram & Noida which are in the India’s NCT. So we now only consider three cities and try to find which restaurant is more popular in India.

```
library(magrittr)
library(dplyr)
zomatoNCT <- zomatofinal %>% filter(zomatofinal$City=="New Delhi" | zomatofinal$City=
="Gurgaon" | zomatofinal$City=="Noida")
```

Now we will find how many restaurants are located in each city.

```
library(ggplot2)
ggplot(zomatoNCT) + geom_bar(aes(City))
```



From this plot, we can say that most of restaurants are located in New Delhi and number of restaurants in Gurgaon and Noida are same.

Now we will only Analysis, restaurants are located in New Delhi because number of restaurants are more in new delhi.

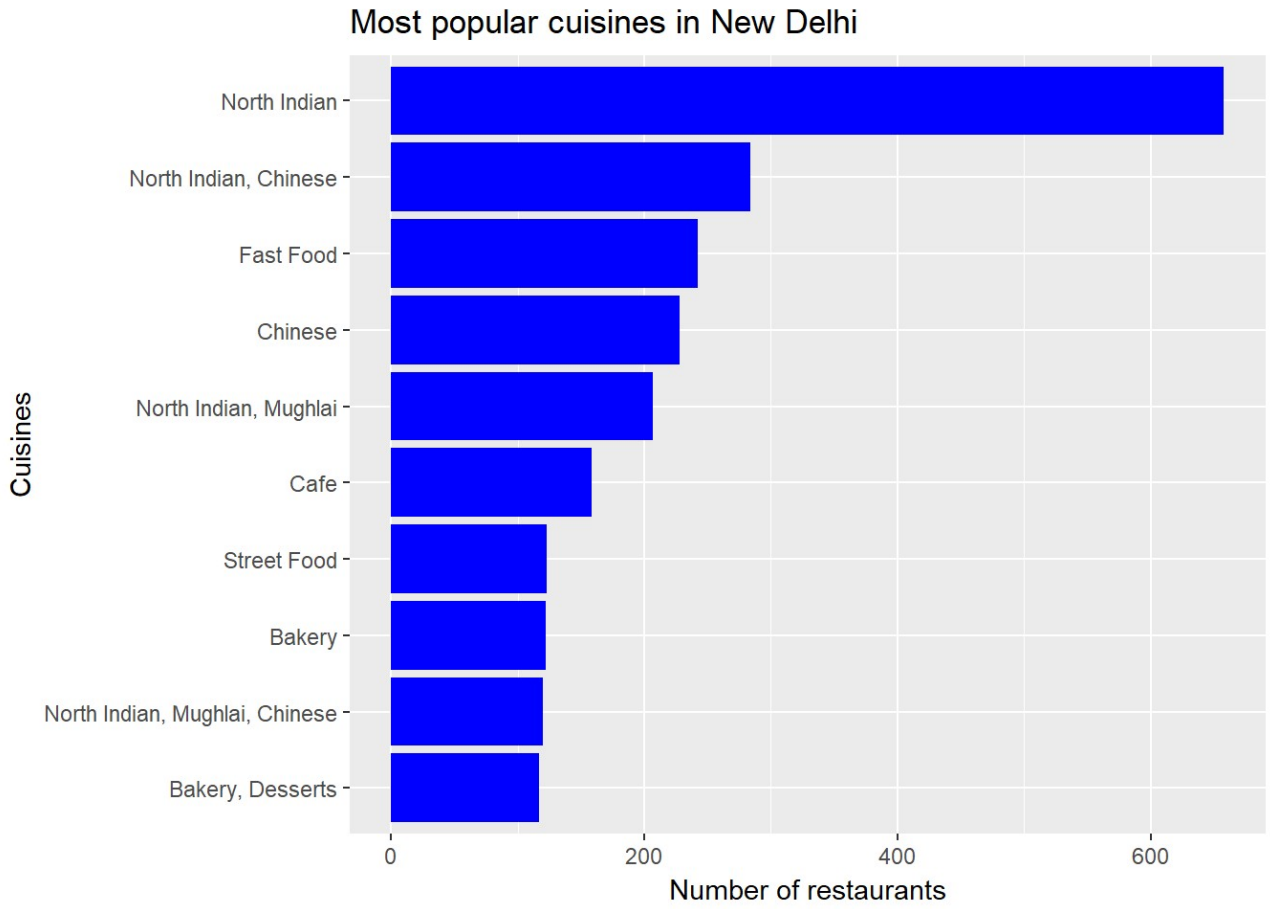
```
library(dplyr)
zomatoDelhi <- zomatoNCT %>% filter(City=="New Delhi")
head(zomatoDelhi)
```

```
##   Restaurant.ID Restaurant.Name Country.Code      City
## 1      18287358      Food Cloud            1 New Delhi
## 2      18216944       Burger.in            1 New Delhi
## 3        313333 Days of the Raj            1 New Delhi
## 4      18384127  Dilli Ka Dhaba            1 New Delhi
## 5           582       Govardhan            1 New Delhi
## 6      18414465  Mezbaan Grills            1 New Delhi
##                                                                                Address
## 1                                                          Aaya Nagar, New Delhi
## 2                                      84, Near Honda Showroom, Adchini, New Delhi
## 3                            81/3, 1st Floor, Qutub Residency, Adchini, New Delhi
## 4                        66 A, Ground Floor, Sri Aurobindo Marg, Adchini, New Delhi
## 5 84, Adjacent Hero Motor Bike Showroom, Main Mehrauli Road, Adchini, New Delhi
## 6                                    A- 96, Shri Aurbindo Marg, Adchini, New Delhi
##     Locality      Locality.Verbose Longitude Latitude
## 1 Aaya Nagar Aaya Nagar, New Delhi   0.00000  0.00000
## 2    Adchini    Adchini, New Delhi  77.19692 28.53538
## 3    Adchini    Adchini, New Delhi  77.19747 28.53549
## 4    Adchini    Adchini, New Delhi  77.19803 28.53755
## 5    Adchini    Adchini, New Delhi  77.19692 28.53552
## 6    Adchini    Adchini, New Delhi  77.19812 28.53813
##                               Cuisines Average.Cost.for.two
## 1                        Cuisine Varies                  500
## 2                             Fast Food                  350
## 3  North Indian, Seafood, Continental                 1500
## 4           South Indian, North Indian                  500
## 5 South Indian, North Indian, Chinese                  500
## 6                              Mughlai                  400
##              Currency Has.Table.booking Has.Online.delivery
## 1 Indian Rupees(Rs.)                 0                   0
## 2 Indian Rupees(Rs.)                 0                   1
## 3 Indian Rupees(Rs.)                 1                   1
## 4 Indian Rupees(Rs.)                 0                   0
## 5 Indian Rupees(Rs.)                 0                   1
## 6 Indian Rupees(Rs.)                 0                   0
##   Is.delivering.now Switch.to.order.menu Price.range Aggregate.rating
## 1                 0                    0           2              0.0
## 2                 0                    0           1              3.2
## 3                 0                    0           3              3.4
## 4                 0                    0           2              2.6
## 5                 0                    0           2              3.4
## 6                 0                    0           1              3.1
##   Rating.color Rating.text Votes Country.Name
## 1        White   Not rated     2        India
## 2       Orange     Average    46        India
## 3       Orange     Average    45        India
## 4       Orange     Average    11        India
```

```
## 5       Orange     Average   238        India
## 6       Orange     Average     8        India
```

Now we will find which cuisines are popular in New Delhi.

```
library(ggplot2)
library(magrittr)

#if code is not working and it gives the error "Error: This function should not be cal
led directly " then try to detach the plyr package.
a<-zomatoDelhi %>%
  group_by(Cuisines) %>%
  summarize(restaurants = n()) %>% top_n(n = 10, wt = restaurants) %>%
  arrange(-restaurants) %>%
  ungroup() -> newDelhi



g1 <- ggplot(newDelhi,aes(x=reorder(Cuisines,restaurants), y=restaurants)) +
 geom_bar(stat='identity', fill ="blue") +
labs(title="Most popular cuisines in New Delhi", x="Cuisines", y="Number of restaurant
s") +
 coord_flip()
g1
```



From this plot, we can say that North Indian offers in most of all restaurants in Delhi.

Classification with rpart:

Now we will classify which restaurants provides online delivery facility.
Refrence:https://blackboard.umbc.edu/webapps/blackboard/execute/content/file?cmd=view&content_id=_2962653_1&course_id=_37782_1&framesetWrapped=true (https://blackboard.umbc.edu/webapps/blackboard/execute/content/file?cmd=view&content_id=_2962653_1&course_id=_37782_1&framesetWrapped=true)

```
zomatoDelhi<- as.data.frame(zomatoDelhi)
table(zomatoDelhi$Has.Online.delivery)
```

```
##
##    0    1
## 3984 1489
```

from this we can see that only ~27% of restaurants provides online delivery.

Next, we run classification tree.

```
library(rpart)
library(rpart.plot)
fit1 <- rpart(Has.Online.delivery ~ Price.range + Aggregate.rating + Is.delivering.no
w + Has.Table.booking+Switch.to.order.menu, method = "class",  parms = list(split = "g
ini"), data = zomatoDelhi)
```

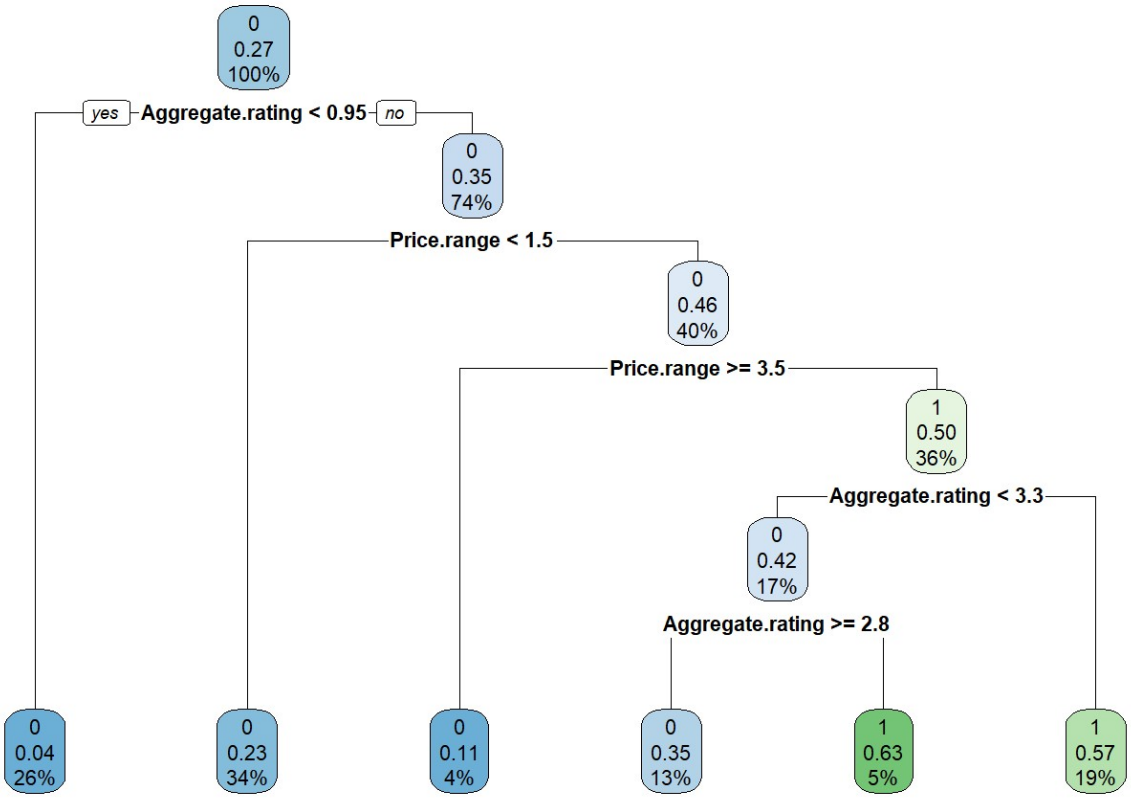
Next we look at the results.

```
print(fit1)
```

```
## n= 5473
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 5473 1489 0 (0.72793715 0.27206285)
##    2) Aggregate.rating< 0.95 1425   53 0 (0.96280702 0.03719298) *
##    3) Aggregate.rating>=0.95 4048 1436 0 (0.64525692 0.35474308)
##      6) Price.range< 1.5 1873  427 0 (0.77202349 0.22797651) *
##      7) Price.range>=1.5 2175 1009 0 (0.53609195 0.46390805)
##       14) Price.range>=3.5 210   23 0 (0.89047619 0.10952381) *
##       15) Price.range< 3.5 1965  979 1 (0.49821883 0.50178117)
##         30) Aggregate.rating< 3.35 947  401 0 (0.57655755 0.42344245)
##           60) Aggregate.rating>=2.75 690  240 0 (0.65217391 0.34782609) *
##           61) Aggregate.rating< 2.75 257   96 1 (0.37354086 0.62645914) *
##         31) Aggregate.rating>=3.35 1018  433 1 (0.42534381 0.57465619) *
```

Now we will plot this result.

```
rpart.plot(fit1)
```



Now we will build confusion matrix which summarize the errors in the classification tree as calculated and find accuracy of it.

```
pred1 <- predict(fit1, type="class")
(conf.mat <- table(zomatoDelhi$Has.Online.delivery, pred1))     # Actual first then pr
ediction
```

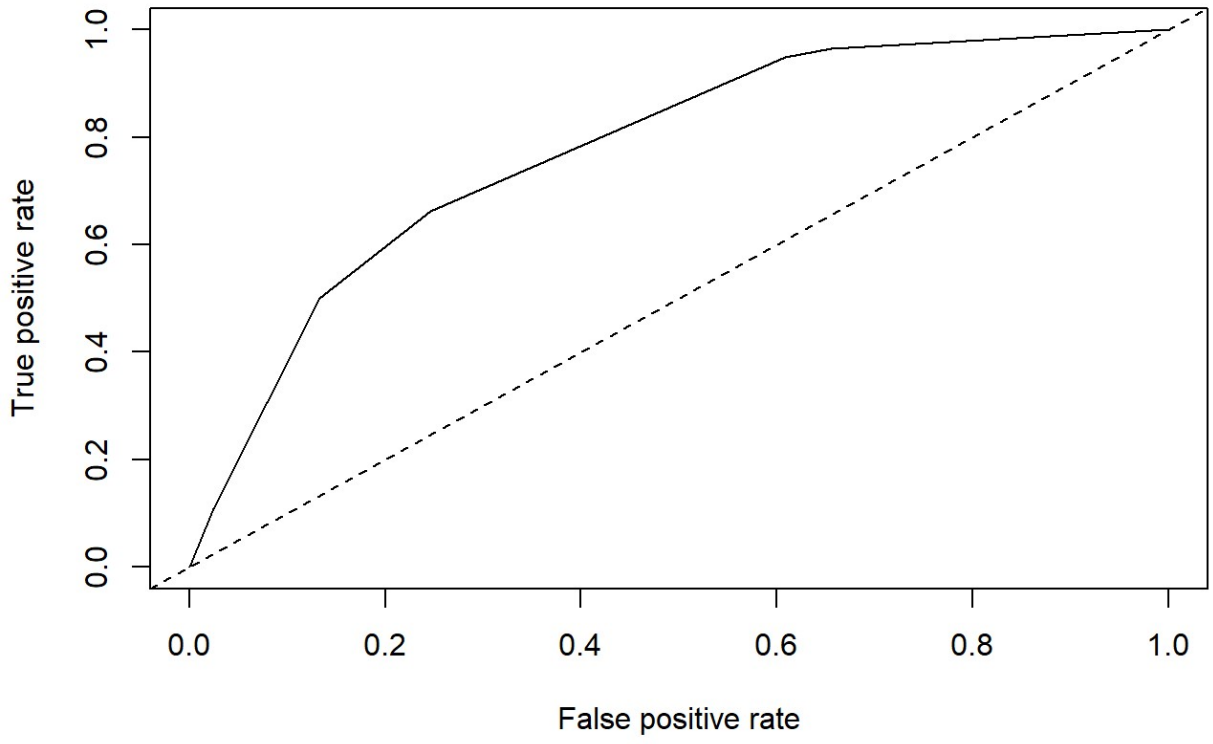
```
##    pred1
##        0    1
##   0 3455  529
##   1  743  746
```

Accuracy: =(3455+746)/(5473) =(4201)/(5473) =0.7675 Misclassification Rate:=(529+743)/(5473) = (1272)/(5473) =0.2324

From this we can say that 76% of data is classify correctly while 23% data is not classify correctly.

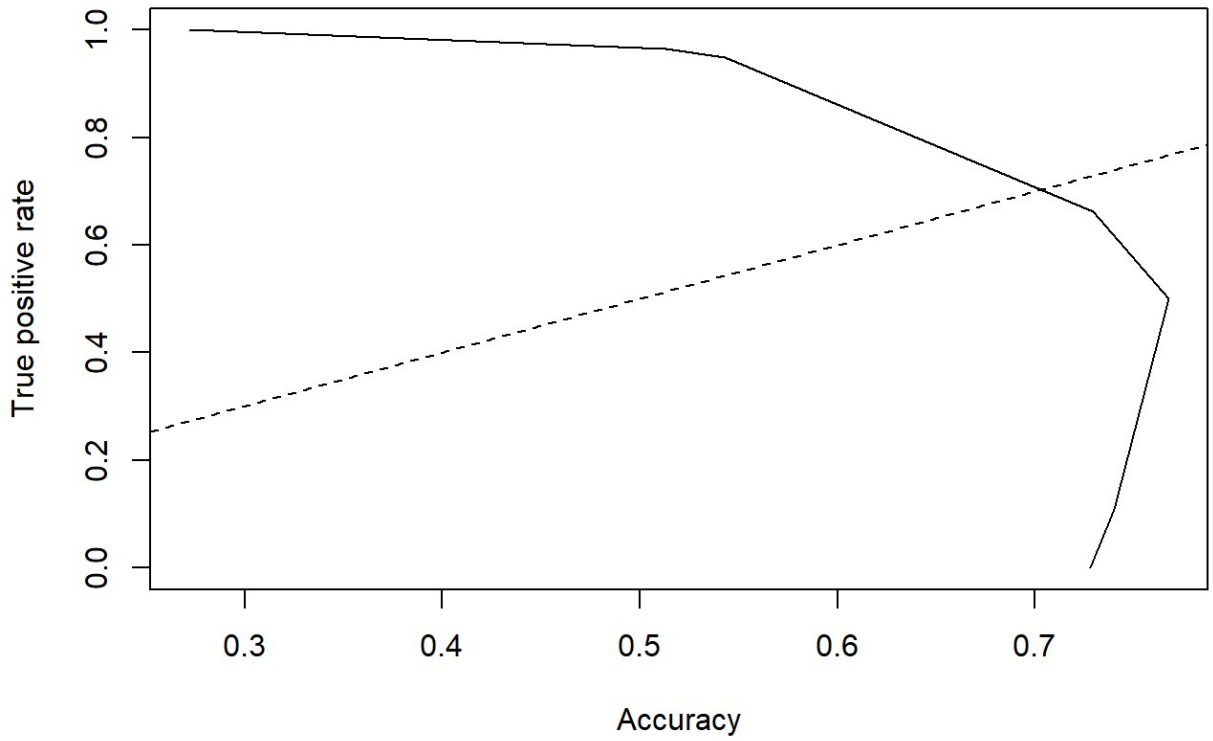
Now you can visualize one of many ROC curve that show the tradeoff between pairs of measurements from the above table and is a measure of the diagnostic ability of binary classification system.

```
library(randomForest)
#library(gplot)
library(rpart)
library(rpart.plot)
library(ROCR)
zomatoDelhi<- as.data.frame(zomatoDelhi)
pred <- prediction(predict(fit1, type = "prob")[, 2],zomatoDelhi[["Has.Online.deliver
y"]])

plot(performance(pred, "tpr", "fpr"))
abline(0,1,lty=2)
```

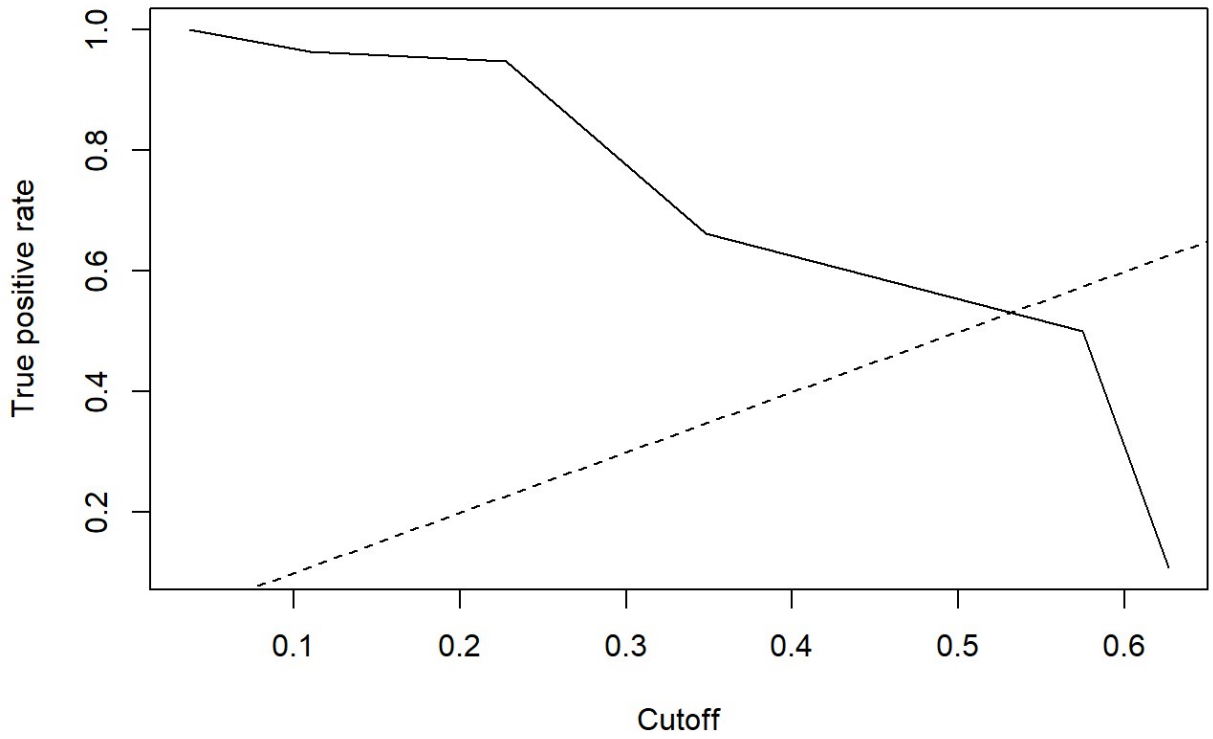


now we will see the plot of tpr vs accuracy

```
plot(performance(pred, "tpr", "acc"))
abline(0,1,lty=2)
```




```
plot(performance(pred, "tpr"))
abline(0,1,lty=2)
```



Now I will use K- means clustering in this data set.

Here I make clusters.

```
zomatofinala<- zomatofinal %>% select(Longitude, Latitude, Aggregate.rating)
set.seed(20)
zoma <- kmeans(zomatofinala[, 1:2], 5, nstart = 20)
zoma
```

```
## K-means clustering with 5 clusters of sizes 8309, 615, 116, 423, 64
##
## Cluster means:
##    Longitude   Latitude
## 1  77.324477  27.764642
## 2  -2.051474  -4.818117
## 3   8.167571  49.171783
## 4 -92.928738  35.754172
## 5 163.456949 -36.509368
##
## Clustering vector:
##    [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2
##   [35] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##   [69] 2 2 2 2 2 2 2 2 2 2 2 2 2 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 4 4 4
##  [103] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
##  [137] 4 4 4 5 5 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
##  [171] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
##  [205] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
##  [239] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 4 4 4 4 4 4 4 4
##  [273] 4 4 4 4 4 4 4 4 4 4 4 5 4 5 5 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
##  [307] 5 5 5 5 5 4 4 5 5 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 4 5 4 5 4
##  [341] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 4 4 4 4 4 4 4 4 4 4 4
##  [375] 4 4 4 4 4 4 4 4 5 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
##  [409] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 1
##  [443] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4
##  [477] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 4 4 4 4 4 4 4
##  [511] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
##  [545] 4 4 4 4 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [579] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1
##  [613] 1 1 2 1 1 2 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [647] 1 1 1 1 2 1 2 1 1 2 2 2 1 1 1 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1 1 1 1
##  [681] 1 1 2 1 1 1 2 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 1 2 1 1 1 1 1 1
##  [715] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 2 1 1 2 1 2 1 1 1 1 2 1 2
##  [749] 1 2 1 1 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [783] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [817] 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 2 2 1 1 2 1 1 2 1 2 1 1 1 1 1
##  [851] 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [885] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1
##  [919] 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1
##  [953] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 2 1 1 1 2 1 1 1
##  [987] 1 2 1 1 2 1 1 1 1 2 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1021] 1 1 1 1 1 1 1 1 2 2 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1055] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1
## [1089] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1123] 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1157] 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1191] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1225] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
## [1259] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1293] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1327] 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1361] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1395] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1429] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1463] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1497] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1531] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1565] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1599] 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1
## [1633] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 2 1 1 2 1 1 1 2 1 1 1 1 1 1 1
## [1667] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1701] 1 1 1 2 1 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1
## [1735] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1
## [1769] 1 2 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 2 1 1 1 2 1 1 1
## [1803] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1837] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1
## [1871] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 2 1 1 1 1
## [1905] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1
## [1939] 2 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2
## [1973] 1 1 1 1 1 1 2 1 1 1 1 2 2 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1
## [2007] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2041] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2075] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1
## [2109] 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2143] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2177] 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1
## [2211] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2245] 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 2 1 1 1 1 1 1 1 2 1 1
## [2279] 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2313] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1
## [2347] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2381] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 2 1 2 1 1 1 1
## [2415] 1 1 1 1 2 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1
## [2449] 2 2 1 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2483] 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2517] 1 1 2 1 1 1 1 1 2 1 1 1 2 1 1 2 1 2 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1
## [2551] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1
## [2585] 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2619] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2653] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2687] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2721] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2755] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2789] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2823] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2857] 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2891] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 2 1 1 1 1 1 1 1
```

```
## [2925] 1 1 1 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2959] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2993] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3027] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3061] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3095] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3129] 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 2 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1
## [3163] 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3197] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3231] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3265] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3299] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3333] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3367] 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3401] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3435] 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1
## [3469] 1 1 1 2 1 2 1 2 2 2 1 2 1 2 1 1 1 1 1 1 1 1 2 3 2 1 1 1 1 1 1 1 1
## [3503] 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3537] 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3571] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 2 1 2 1 1
## [3605] 1 2 1 2 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2
## [3639] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3673] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3707] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3741] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 2 1 1 1 1 2 1 1 1 1 1
## [3775] 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 2 1 1 1 1
## [3809] 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 2 1 1 1
## [3843] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1
## [3877] 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3911] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3945] 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3979] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2
## [4013] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1
## [4047] 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1
## [4081] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4115] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4149] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4183] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4217] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4251] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1
## [4285] 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4319] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1
## [4353] 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4387] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4421] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1
## [4455] 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1
## [4489] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1
## [4523] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4557] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1
```

```
## [4591] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4625] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4659] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4693] 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 2
## [4727] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 2
## [4761] 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1 2 1 2 1
## [4795] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1
## [4829] 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4863] 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4897] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1
## [4931] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1
## [4965] 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [4999] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5033] 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5067] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5101] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5135] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5169] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5203] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5237] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 2 1 1 1
## [5271] 2 1 2 2 1 1 1 1 2 2 2 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5305] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5339] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5373] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5407] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5441] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5475] 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1
## [5509] 1 1 2 1 1 1 1 1 1 1 1 2 2 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1
## [5543] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1
## [5577] 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5611] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5645] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5679] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5713] 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5747] 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1
## [5781] 1 1 1 1 1 1 2 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5815] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [5849] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1
## [5883] 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 2 1 2
## [5917] 2 2 2 2 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1
## [5951] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1
## [5985] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [6019] 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [6053] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [6087] 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [6121] 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [6155] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [6189] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [6223] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
## [6257] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1
## [6291] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [6325] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1
## [6359] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [6393] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [6427] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2
## [6461] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [6495] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1
## [6529] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [6563] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1
## [6597] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [6631] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1
## [6665] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [6699] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [6733] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2
## [6767] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1
## [6801] 1 1 1 1 1 1 1 1 1 2 1 2 2 2 1 1 2 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1
## [6835] 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 2 2 1 1 1 1 1
## [6869] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [6903] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [6937] 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1
## [6971] 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1
## [7005] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [7039] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [7073] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [7107] 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1 2
## [7141] 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [7175] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [7209] 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1
## [7243] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 2 1 1 1 2 2 1 1 1 1 1 1 1 1
## [7277] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [7311] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 1 2 2 1 1 1 1 1 1 1 1
## [7345] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1
## [7379] 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [7413] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [7447] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [7481] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [7515] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [7549] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1
## [7583] 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [7617] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1
## [7651] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [7685] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1
## [7719] 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 2 2
## [7753] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 2 2
## [7787] 1 2 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 2 1 1 1 1 1
## [7821] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [7855] 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [7889] 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
## [7923] 1 1 1 1 2 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [7957] 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [7991] 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [8025] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1
## [8059] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [8093] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [8127] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1
## [8161] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1
## [8195] 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1
## [8229] 1 1 1 1 1 1 1 1 2 1 1 2 1 1 1 2 2 1 1 2 1 2 2 1 1 1 1 1 1 1 1 1 1
## [8263] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1
## [8297] 2 1 1 1 1 1 1 1 1 2 1 1 2 2 2 1 1 1 2 1 1 1 1 1 1 1 1 2 2 1 1 1 2 1
## [8331] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1
## [8365] 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [8399] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [8433] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1
## [8467] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2
## [8501] 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1
## [8535] 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [8569] 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 3 1 1 2
## [8603] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [8637] 1 2 2 2 1 1 2 2 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 2 2
## [8671] 1 2 1 1 2 1 1 2 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [8705] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [8739] 2 1 1 1 1 1 1 1 1 1 2 2 2 2 1 1 1 1 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1
## [8773] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [8807] 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [8841] 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1
## [8875] 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 1 2
## [8909] 1 1 2 2 2 2 1 1 1 1 1 1 1 1 2 2 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [8943] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 2
## [8977] 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1
## [9011] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [9045] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2
## [9079] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [9113] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [9147] 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [9181] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 2 2 2 1 1 2 1 1 2 2 2 2 1 1
## [9215] 1 1 2 2 2 2 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 2 2 2 1 1 1 2 1 1 1
## [9249] 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 5 5 5 5 5 5 5 5 5
## [9283] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 3 3 3
## [9317] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [9351] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [9385] 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2
## [9419] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [9453] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1
## [9487] 1 2 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [9521] 3 3 3 3 3 3 3
##
```

```
## Within cluster sum of squares by cluster:
## [1] 235294.85 225038.35  32043.11 140886.84  18401.18
##  (between_SS / total_SS =  96.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

Now we have a prediction.

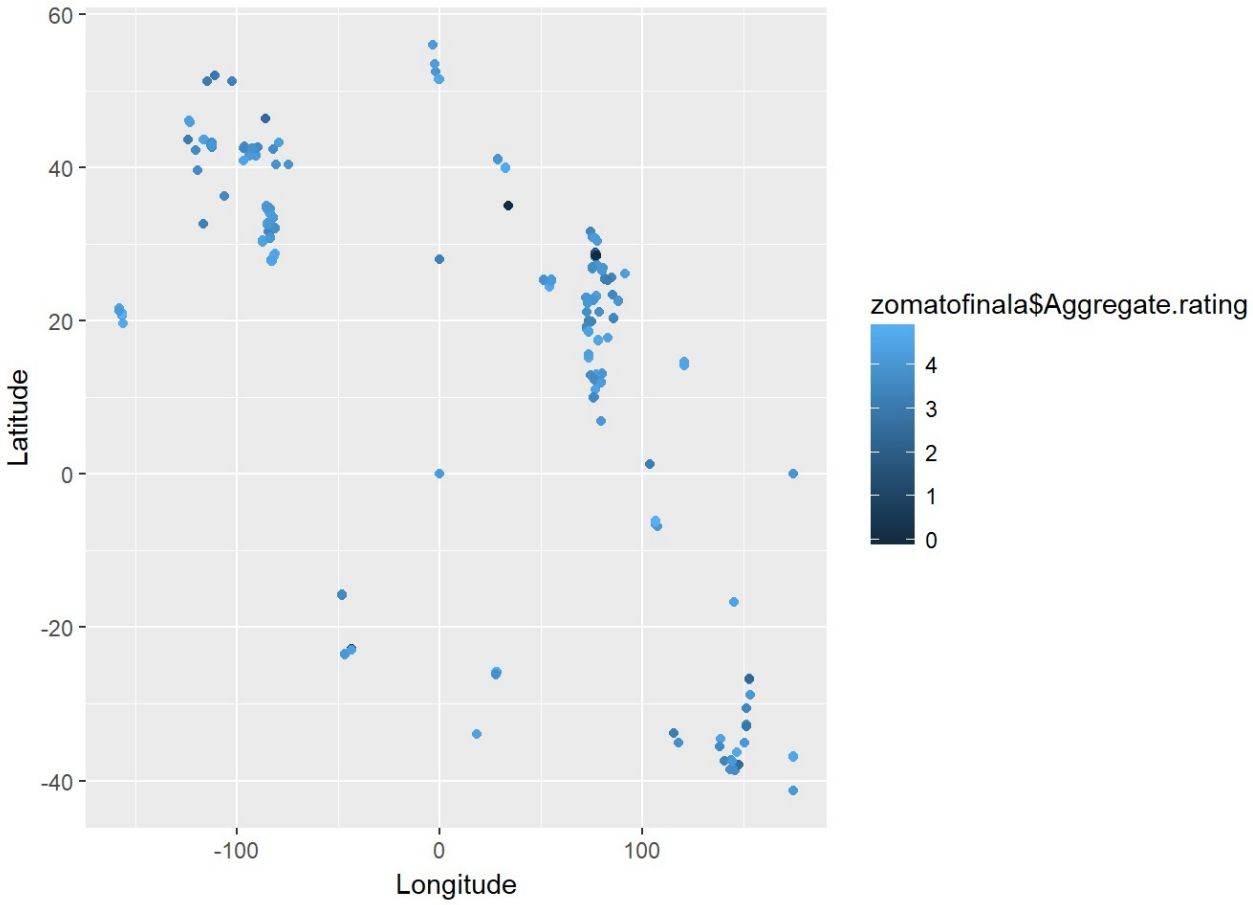
```
table(zoma$cluster,round(zomatofinala$Aggregate.rating))
```

```
##
##        0    2    3    4    5
##   1 1854  291 3478 2580  106
##   2  289    1  112  189   24
##   3    2    0    7   85   22
##   4    3    2   21  354   43
##   5    0    2    4   49    9
```

Overall Accuracy: 5.49% Refrence:http://www.marcovanetti.com/pages/cfmatrix/?noc=5 (http://www.marcovanetti.com/pages/cfmatrix/?noc=5)

Here I will plot aggregate rating in Latitude vs Longitude.

```
zoma$cluster <- as.factor(zoma$cluster)
ggplot(zomatofinala, aes(Longitude,Latitude, color = zomatofinala$Aggregate.rating))
+ geom_point()
```



From this plot we can say that there are less points for 0 aggregate rating while there are more points for 3 or above aggregate raing.

Here I assign the geocode to countries which are in zomato detaset.

```
library("ggmap")
library(maptools)
library(maps)


visited <- c("India", "Australia", "Brazil", "Canada", "Indonesia", "New Zealand" , "P
hillipines" , "Qatar" , "Singapore" , "South Africa" , "Sri Lanka" , "Turkey" , "UAE"
, "United Kingdom" , "United States")
ll.visited <- geocode(visited)
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Ind
ia&sensor=false
```

```
## Warning: geocode failed with status OVER_QUERY_LIMIT, location = "India"
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Aus
tralia&sensor=false
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Bra
zil&sensor=false
```

```
## Warning: geocode failed with status OVER_QUERY_LIMIT, location = "Brazil"
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Can
ada&sensor=false
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Ind
onesia&sensor=false
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Ne
w%20Zealand&sensor=false
```

```
## Warning: geocode failed with status OVER_QUERY_LIMIT, location = "New
## Zealand"
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Phi
llipines&sensor=false
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Qat
ar&sensor=false
```

```
## Warning: geocode failed with status OVER_QUERY_LIMIT, location = "Qatar"
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Sin
gapore&sensor=false
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Sou
th%20Africa&sensor=false
```

```
## Warning: geocode failed with status OVER_QUERY_LIMIT, location = "South
## Africa"
```

```
## .
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Sr
i%20Lanka&sensor=false
```

```
## .Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Tu
rkey&sensor=false
## .Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=UA
E&sensor=false
## .Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Un
ited%20Kingdom&sensor=false
## .Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Un
ited%20States&sensor=false
```

```
## Warning: geocode failed with status OVER_QUERY_LIMIT, location = "United
## States"
```

```
visit.x <- ll.visited$lon
visit.y <- ll.visited$lat
#> dput(visit.x)
#c(-122.389979, 80.249583, -0.1198244, 144.96328, 28.06084)
#> dput(visit.y)
#c(37.615223, 13.060422, 51.5112139, -37.814107, -26.1319199)
```

Here I plot above longitude and latitude in world map.

```
#mapWorld <- borders("world", colour="gray50", fill="gray50") # create a layer of bord
ers
#mp <- ggplot() +   mapWorld

#Now Layer the cities on top
#mp <- mp+ geom_point(aes(x=visit.x, y=visit.y) )
#mp
```

Here I have ploted aggregate rating on longitude and latitude.

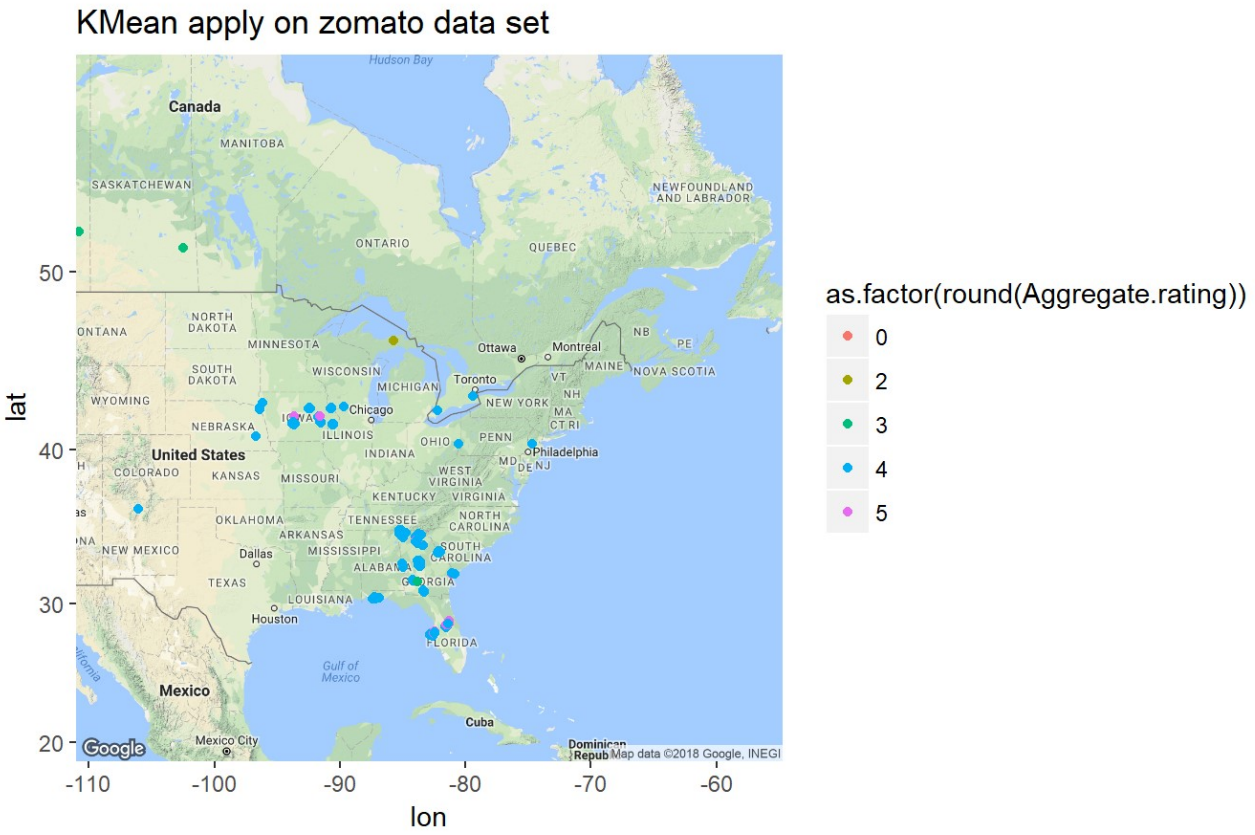
```
#library(ggmap)
library(ggplot2)
MAPzom <- get_map("mp", zoom = 4)
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=mp&zoom=4&size=
640x640&scale=2&maptype=terrain&language=en-EN&sensor=false
```

```
## .Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=mp
&sensor=false
```

```
ggmap(MAPzom) + geom_point(aes(x = Longitude[], y = Latitude[], colour = as.factor(rou
nd(Aggregate.rating))),data = zomatofinala) +
  ggtitle("KMean apply on zomato data set")
```

```
## Warning: Removed 9168 rows containing missing values (geom_point).
```



Modeling result from this model: From this model we can say that there are many cluster for aggregate rating 4 in this map while few points for other aggregate rating.There are few points in canada which have aggregate rating 3.So from this we can say that in cananda average restaurants are available.

Future Plan: I will identify which is the top 3 restaurants in New Delhi. I will convert currency of different coutries into one unit and then try to predict price range of restaurants.