Clustering_ananlysis_on_Abnb_Istanbul

Clustering done based on 2 approaches –

1. Based on 'Number of Reviews', 'Reviews per month' and 'Price'
2. Based on Latitude, Longitude and Price

Clustering Approach1: 'Number of Reviews', 'Reviews per month' and 'Price'

```
knitr::opts_chunk$set(echo = TRUE)

library(data.table)

## Warning: package 'data.table' was built under R version 3.6.2

library(fpp)

## Loading required package: forecast

## Warning: package 'forecast' was built under R version 3.6.2

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

## Loading required package: fma

## Warning: package 'fma' was built under R version 3.6.2

## Loading required package: expsmooth

## Loading required package: lmtest

## Loading required package: zoo

## Warning: package 'zoo' was built under R version 3.6.2

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: tseries
```

```r
library(fpp2)
```

```
## Loading required package: ggplot2

##
## Attaching package: 'fpp2'

## The following objects are masked from 'package:fpp':
##
##     ausair, ausbeer, austa, austourists, debitcards, departures,
##     elecequip, euretail, guinearice, oil, sunspotarea, usmelec
```

```r
library(cowplot)
```

```
## Warning: package 'cowplot' was built under R version 3.6.2

##
## ********************************************************

## Note: As of version 1.0.0, cowplot does not change the

##   default ggplot2 theme anymore. To recover the previous

##   behavior, execute:
##   theme_set(theme_cowplot())

## ********************************************************
```

```r
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.6.2

## -- Attaching packages -------------------------------------------
------- tidyverse 1.3.0 --

## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
## v purrr   0.3.3

## Warning: package 'tidyr' was built under R version 3.6.2

## Warning: package 'purrr' was built under R version 3.6.2

## Warning: package 'dplyr' was built under R version 3.6.2

## Warning: package 'forcats' was built under R version 3.6.2

## -- Conflicts ----------------------------------------------------
- tidyverse_conflicts() --
## x dplyr::between()   masks data.table::between()
```

```
## x dplyr::filter()    masks stats::filter()
## x dplyr::first()     masks data.table::first()
## x dplyr::lag()       masks stats::lag()
## x dplyr::last()      masks data.table::last()
## x purrr::transpose() masks data.table::transpose()
```

```r
library(psych)
```

```
## Warning: package 'psych' was built under R version 3.6.2
```

```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```r
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.6.2
```

```r
library(dplyr)
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 3.6.2
```

```
## corrplot 0.84 loaded
```

```r
library(GGally)
```

```
## Warning: package 'GGally' was built under R version 3.6.2
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
##
## Attaching package: 'GGally'
```

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

```
## The following object is masked from 'package:fma':
##
##     pigs
```

```r
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

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

## The following objects are masked from 'package:data.table':
##
##     dcast, melt

AirbnbIstanbul <- read.csv("C:/Pritesh/Rutgers/Courses/Projects/MVA/Da
taset/AirbnbIstanbul.csv", stringsAsFactors=FALSE)
Istanbul <- copy(AirbnbIstanbul)
class(Istanbul)

## [1] "data.frame"

setDT(Istanbul)

str(Istanbul)

## Classes 'data.table' and 'data.frame':   16251 obs. of  16 variable
s:
##  $ id                            : int  4826 20815 25436 27271 2827
7 28308 28318 29241 30697 33368 ...
##  $ name                          : chr  "The Place" "The Bosphorus
from The Comfy Hill" "House for vacation rental furnutare" "LOVELY APT
. IN PERFECT LOCATION" ...
##  $ host_id                       : int  6603 78838 105823 117026 12
1607 121695 121721 125742 132137 135136 ...
##  $ host_name                     : chr  "Kaan" "GÃ¼lder" "Yesim" "M
utlu" ...
##  $ neighbourhood_group           : logi  NA NA NA NA NA NA ...
##  $ neighbourhood                 : chr  "Uskudar" "Besiktas" "Besik
tas" "Beyoglu" ...
##  $ latitude                      : num  41.1 41.1 41.1 41 41 ...
##  $ longitude                     : num  29.1 29 29 29 29 ...
##  $ room_type                     : chr  "Entire home/apt" "Entire h
ome/apt" "Entire home/apt" "Entire home/apt" ...
##  $ price                         : int  554 100 211 237 591 237 633
264 596 295 ...
##  $ minimum_nights                : int  1 30 21 5 3 1 3 3 1 2 ...
##  $ number_of_reviews             : int  1 41 0 2 0 0 0 0 1 1 ...
##  $ last_review                   : chr  "2009-06-01" "2018-11-07" "
" "2018-05-04" ...
##  $ reviews_per_month             : num  0.01 0.38 NA 0.04 NA NA NA
NA 0.01 0.02 ...
##  $ calculated_host_listings_count: int  1 2 1 1 13 1 1 1 1 2 ...
##  $ availability_365              : int  365 49 83 228 356 365 365 3
```

```
65 365 232 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```r
Istanbul[,room_type:=factor(room_type)]
Istanbul[,neighbourhood:=factor(neighbourhood)]
Istanbul[,last_review:=as.Date(last_review,'%Y-%m-%d')] ## converting
last_review to date datatype

# datatypes looks better now. hence will see again for NA values
grep ('NA',Istanbul) # 2, 5, 13 and 14 column have NA values
```

```
## [1]  2  5 13 14
```

```r
Istanbul[is.na(neighbourhood_group),NROW(neighbourhood_group)] # entir
e obs. is blank, will drop this var
```

```
## [1] 16251
```

```r
Istanbul[is.na(last_review),NROW(last_review)] ## there are 8484 NA va
lues
```

```
## [1] 8484
```

```r
Istanbul[is.na(reviews_per_month),NROW(reviews_per_month)] ## there ar
e 8484 NA values
```

```
## [1] 8484
```

```r
Istanbul$neighbourhood_group <- NULL ## removing neighbourhood_group
column
Istanbul[is.na(reviews_per_month),reviews_per_month:=0] ## nearly 50%
of the dataset is filled with NA.
# hence we can't simply remove these many rows. Hence imputing with 0
values.
```

Removing Outliers

Removing 613 observations (out of 16000) which have Price >$1000

Keeping rows having Number of Reviews > 0.

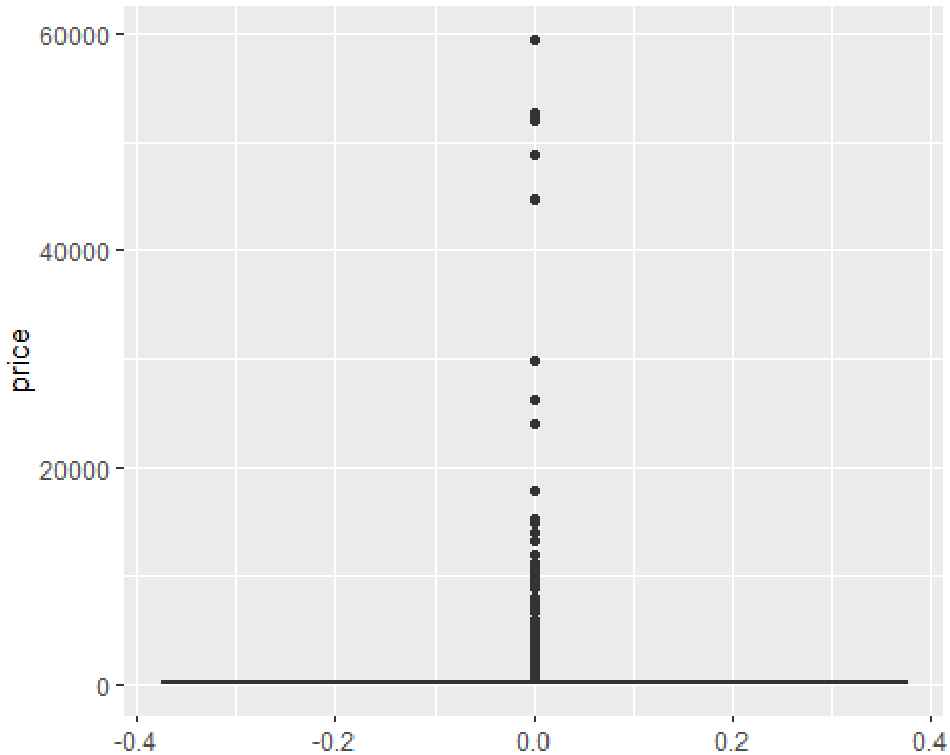
```r
range(Istanbul$price) ## range of price
```

```
## [1]     0 59561
```

```r
avgNeighbourhood=Istanbul[,avgneighprice:=mean(price),by=neighbourhood
]
summary(Istanbul$price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##     0.0   105.0   190.0   354.7   327.0 59561.0
```

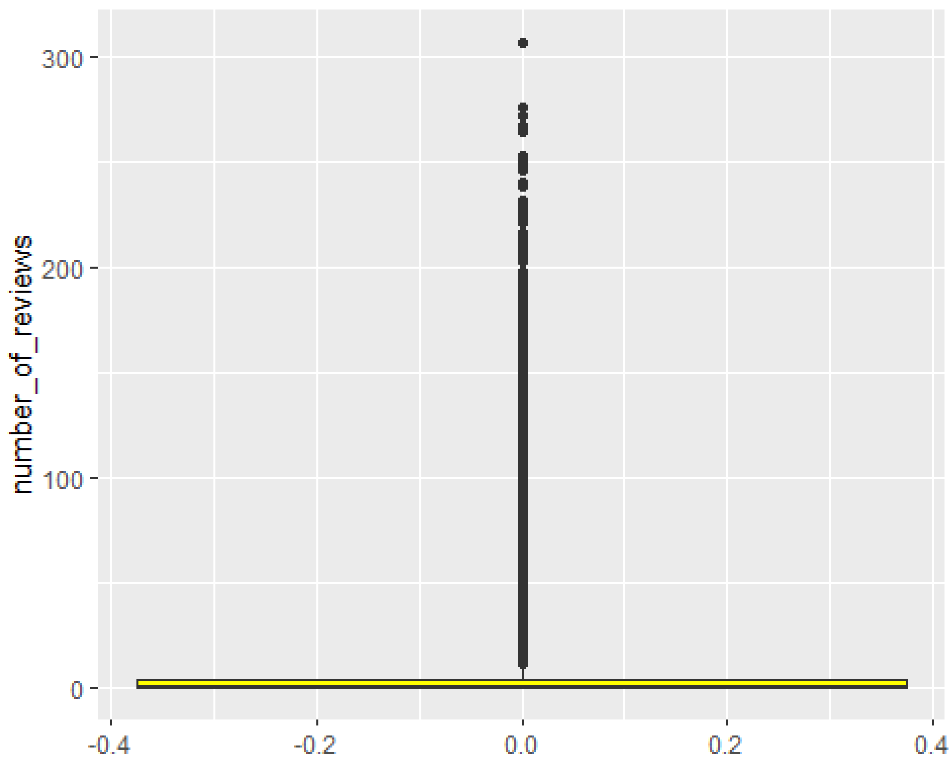
```
ggplot(Istanbul,aes(y=price)) + geom_boxplot(fill='yellow') # the boxp
lot shows that most of the units have price less than 10000
```



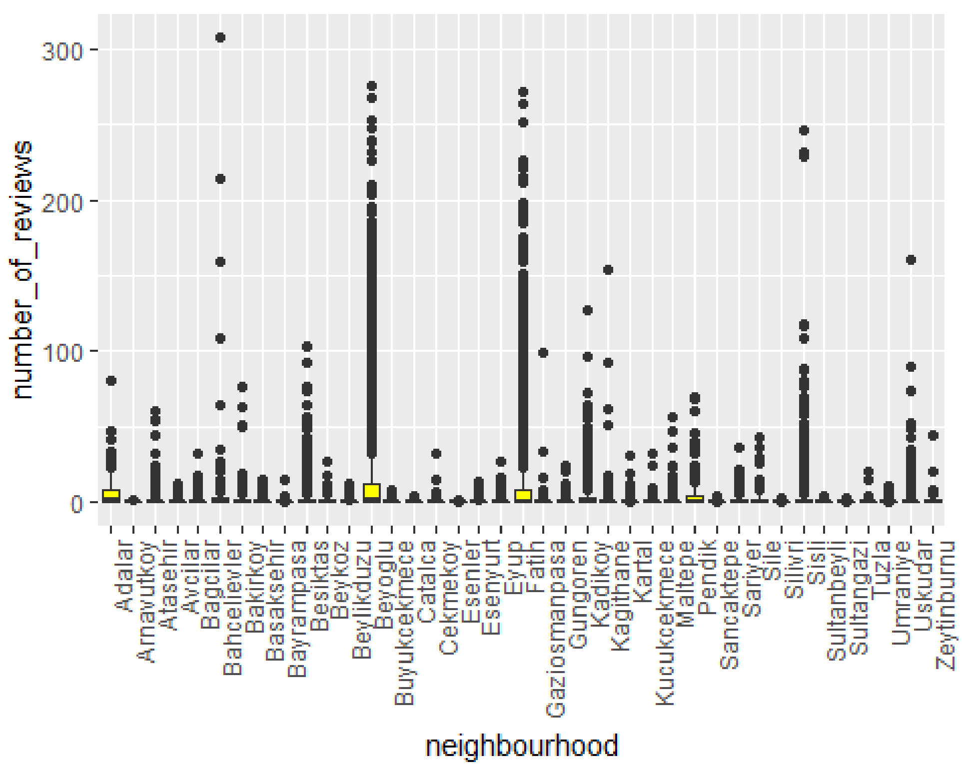

```
## no. of reviews and neighbourhood relation
summary(Istanbul$number_of_reviews)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   0.000   0.000   0.000   7.187   4.000 307.000
```

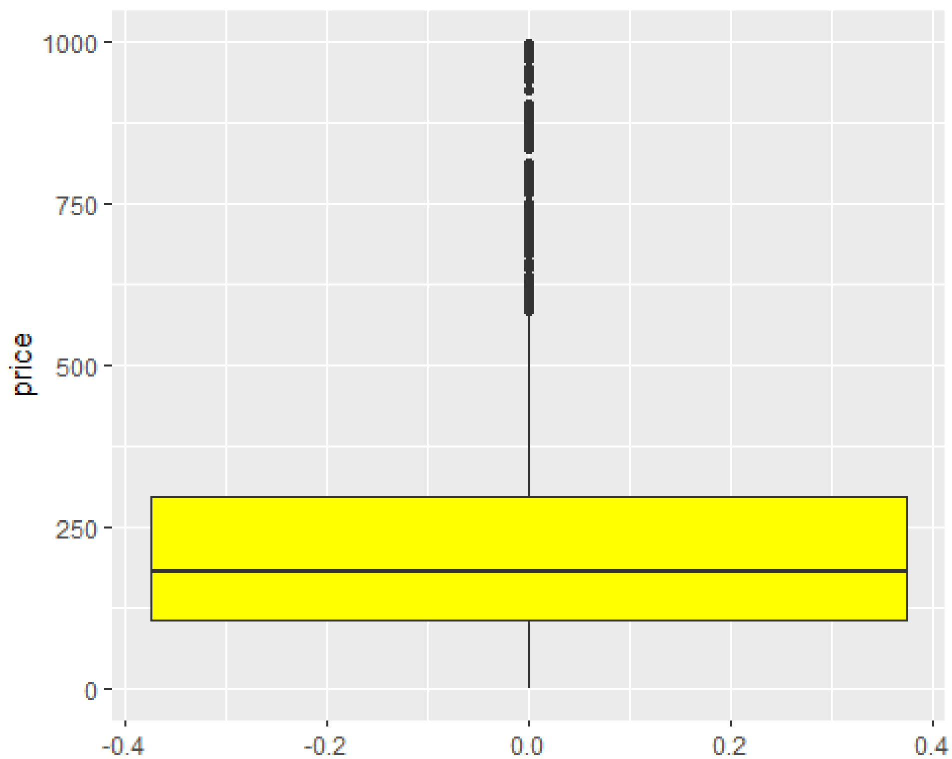
```
ggplot(Istanbul,aes(y=number_of_reviews)) + geom_boxplot(fill='yellow'
)
```




```r
ggplot(Istanbul,aes(x=neighbourhood,y=number_of_reviews)) + geom_boxplot(fill='yellow') + theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



```r
nrow(Istanbul[price > 1000]) ## price > 1000, there are only 613 units
out of ~16000 which have price > 1000

## [1] 613

# hence we'll remove those.
Istanbul.clust <- Istanbul[price < 1000 & number_of_reviews > 0] ## pr
ice > 1000
ggplot(Istanbul.clust,aes(y=price)) + geom_boxplot(fill='yellow') # gg
plot looks better now
```



So Now We have Average Price around $ 225 in our dataset which is input for Cluster analysis done below.

Clustering based on Number of Reviews, Reviews per month and Price.

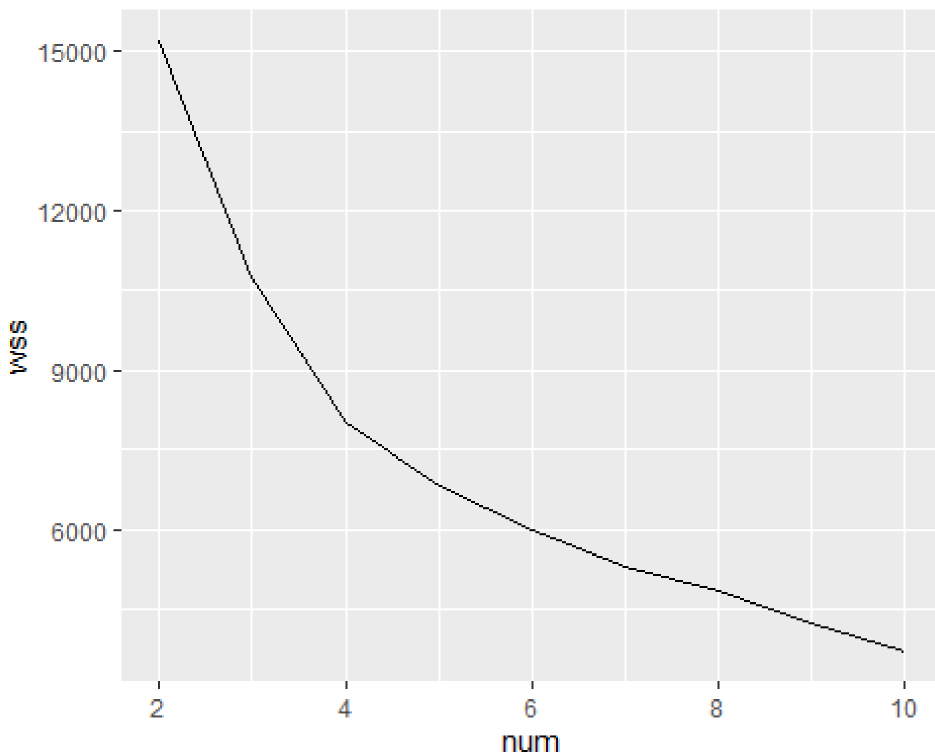
```r
########## K-means Clustering #########
library(cluster)

## Warning: package 'cluster' was built under R version 3.6.2
```

```
Istanbul_clus = data.frame(
  Istanbul.clust$price,
  Istanbul.clust$number_of_reviews,
  Istanbul.clust$reviews_per_month)

# Making property id as rownames, clusters will be formed with id as o
bservations.
rownames(Istanbul_clus) <- Istanbul.clust$id
##Scaling done to make the data on one scale.
Istanbul.Scale <- scale(Istanbul_clus[,1:3])
#Here we have selected first row to see how our scaled matrix is like
head(Istanbul.Scale,1)

##      Istanbul.clust.price Istanbul.clust.number_of_reviews
## 4826               1.9566                       -0.4855794
##      Istanbul.clust.reviews_per_month
## 4826                       -0.8383381

# We will find K-means by taking k=2, 3, 4, 5, 6...
# Centers (k's) are numbers thus, 10 random sets are chosen

#For 2 clusters, k-means = 2
set.seed(123)
kmeans2.Istanbul <- kmeans(Istanbul.Scale,2,nstart = 10)
# Computing the percentage of variation accounted for two clusters
perc_var_kmeans2 <- round(100*(1 - kmeans2.Istanbul$betweenss/kmeans2.
Istanbul$totss),1)
names(perc_var_kmeans2) <- "Perc. 2 clus"
perc_var_kmeans2

## Perc. 2 clus
##         66.8

# Computing the percentage of variation accounted for. Three clusters
kmeans3.Istanbul <- kmeans(Istanbul.Scale,3,nstart = 10)
perc.var.3 <- round(100*(1 - kmeans3.Istanbul$betweenss/kmeans3.Istanb
ul$totss),1)
names(perc.var.3) <- "Perc. 3 clus"
perc.var.3

## Perc. 3 clus
##         47.3

# Computing the percentage of variation accounted for. Four clusters
kmeans4.Istanbul  <- kmeans(Istanbul.Scale,4,nstart = 10)
perc.var.4 <- round(100*(1 - kmeans4.Istanbul$betweenss/kmeans4.Istanb
ul$totss),1)
```

```
names(perc.var.4) <- "Perc. 4 clus"
perc.var.4

## Perc. 4 clus
##         35.1

# Computing the percentage of variation accounted for. Five clusters
kmeans5.Istanbul  <- kmeans(Istanbul.Scale,5,nstart = 10)
perc.var.5 <- round(100*(1 - kmeans5.Istanbul$betweenss/kmeans5.Istanb
ul$totss),1)
names(perc.var.5) <- "Perc. 5 clus"
perc.var.5

## Perc. 5 clus
##         29.9

# Computing the percentage of variation accounted for. Six clusters
kmeans6.Istanbul  <- kmeans(Istanbul.Scale,6,nstart = 10)
perc.var.6 <- round(100*(1 - kmeans6.Istanbul$betweenss/kmeans6.Istanb
ul$totss),1)
names(perc.var.6) <- "Perc. 6 clus"
perc.var.6

## Perc. 6 clus
##         25.2
```

Elbow Plot to Identify the Best number of K Clusters

```
wss=c()########## empty vector to hold wss
for(i in 2:10)#### from 2 to 10 cluster
{
  km = kmeans(Istanbul.Scale[,1:3],i)
  wss[i-1]=km$tot.withinss
}
wss

## [1] 15197.254 10745.783  7987.996  6808.887  5980.367  5311.900  48
46.853
## [8]  4240.790  3709.000

elbowdt = data.table(num=2:10,wss)
ggplot(elbowdt,aes(x=num,y=wss)) + geom_line()
```




```
elbowdt
```

```
##    num       wss
## 1:   2 15197.254
## 2:   3 10745.783
## 3:   4  7987.996
## 4:   5  6808.887
## 5:   6  5980.367
## 6:   7  5311.900
## 7:   8  4846.853
## 8:   9  4240.790
## 9:  10  3709.000
```

For k = 6 the between sum of square/total sum of square ratio tends to change slowly and remain less changing as compared to others. Therefore, k = 6 should be a good choice for the number of clusters.

```
# Saving six k-means clusters in a list
clus1 <- matrix(names(kmeans6.Istanbul$cluster[kmeans6.Istanbul$cluster == 1]),
                ncol=1, nrow=length(kmeans6.Istanbul$cluster[kmeans6.I
```

```
stanbul$cluster == 1]))

colnames(clus1) <- "Cluster 1"

clus2 <- matrix(names(kmeans6.Istanbul$cluster[kmeans6.Istanbul$cluste
r == 2]),
                ncol=1, nrow=length(kmeans6.Istanbul$cluster[kmeans6.I
stanbul$cluster == 2]))
colnames(clus2) <- "Cluster 2"

clus3 <- matrix(names(kmeans6.Istanbul$cluster[kmeans6.Istanbul$cluste
r == 3]),
                ncol=1, nrow=length(kmeans6.Istanbul$cluster[kmeans6.I
stanbul$cluster == 3]))
colnames(clus3) <- "Cluster 3"

clus4 <- matrix(names(kmeans6.Istanbul$cluster[kmeans6.Istanbul$cluste
r == 4]),
                ncol=1, nrow=length(kmeans6.Istanbul$cluster[kmeans6.I
stanbul$cluster == 4]))
colnames(clus4) <- "Cluster 4"

clus5 <- matrix(names(kmeans6.Istanbul$cluster[kmeans6.Istanbul$cluste
r == 5]),
                ncol=1, nrow=length(kmeans6.Istanbul$cluster[kmeans6.I
stanbul$cluster == 5]))
colnames(clus5) <- "Cluster 5"

clus6 <- matrix(names(kmeans6.Istanbul$cluster[kmeans6.Istanbul$cluste
r == 6]),
                ncol=1, nrow=length(kmeans6.Istanbul$cluster[kmeans6.I
stanbul$cluster == 6]))
colnames(clus6) <- "Cluster 6"


#list(clus1,clus2,clus3,clus4,clus5,clus6)

Istanbul_clus_Out <- cbind(Istanbul_clus, clusterNumber = kmeans6.Ista
nbul$cluster)

class(Istanbul_clus_Out)

## [1] "data.frame"

setDT(Istanbul_clus_Out)

Istanbul_cluster1 <- Istanbul_clus_Out[clusterNumber == 1]
```

```r
Istanbul_cluster2 <- Istanbul_clus_Out[clusterNumber == 2]
Istanbul_cluster3 <- Istanbul_clus_Out[clusterNumber == 3]
Istanbul_cluster4 <- Istanbul_clus_Out[clusterNumber == 4]
Istanbul_cluster5 <- Istanbul_clus_Out[clusterNumber == 5]
Istanbul_cluster6 <- Istanbul_clus_Out[clusterNumber == 6]

names(Istanbul_cluster1) <- c("price","number_of_reviews","reviews_per
_month","clusterNumber")
names(Istanbul_cluster2) <- c("price","number_of_reviews","reviews_per
_month","clusterNumber")
names(Istanbul_cluster3) <- c("price","number_of_reviews","reviews_per
_month","clusterNumber")
names(Istanbul_cluster4) <- c("price","number_of_reviews","reviews_per
_month","clusterNumber")
names(Istanbul_cluster5) <- c("price","number_of_reviews","reviews_per
_month","clusterNumber")
names(Istanbul_cluster6) <- c("price","number_of_reviews","reviews_per
_month","clusterNumber")

head(Istanbul_cluster1)

##    price number_of_reviews reviews_per_month clusterNumber
## 1:   142                13              3.64             1
## 2:   185                17              3.81             1
## 3:   190                54              5.47             1
## 4:   322                21              2.83             1
## 5:   448                52              3.70             1
## 6:   369                36              3.32             1

mean(Istanbul_cluster1$price)

## [1] 206.7967

mean(Istanbul_cluster1$number_of_reviews)

## [1] 25.51636

mean(Istanbul_cluster1$reviews_per_month)

## [1] 3.958248

head(Istanbul_cluster2)

##    price number_of_reviews reviews_per_month clusterNumber
## 1:   237                 2              0.04             2
## 2:   295                 1              0.02             2
## 3:   237                 8              0.15             2
## 4:   359                37              0.59             2
```

```
## 5:   353                 46              0.45             2
## 6:   248                  6              0.92             2
```

```r
mean(Istanbul_cluster2$price)
```

```
## [1] 322.6325
```

```r
mean(Istanbul_cluster2$number_of_reviews)
```

```
## [1] 6.37672
```

```r
mean(Istanbul_cluster2$reviews_per_month)
```

```
## [1] 0.4308716
```

```r
head(Istanbul_cluster3)
```

```
##    price number_of_reviews reviews_per_month clusterNumber
## 1:   554                  1              0.01             3
## 2:   596                  1              0.01             3
## 3:   501                 20              0.24             3
## 4:   738                  1              0.01             3
## 5:   533                 34              0.39             3
## 6:   791                  3              0.03             3
```

```r
mean(Istanbul_cluster3$price)
```

```
## [1] 643.1644
```

```r
mean(Istanbul_cluster3$number_of_reviews)
```

```
## [1] 10.37329
```

```r
mean(Istanbul_cluster3$reviews_per_month)
```

```
## [1] 0.6058733
```

```r
head(Istanbul_cluster4)
```

```
##    price number_of_reviews reviews_per_month clusterNumber
## 1:   232                 74              0.79             4
## 2:   322                 81              0.99             4
## 3:   158                 83              0.88             4
## 4:    90                 54              0.59             4
## 5:   264                 74              0.84             4
## 6:    53                 56              0.62             4
```

```r
mean(Istanbul_cluster4$price)
```

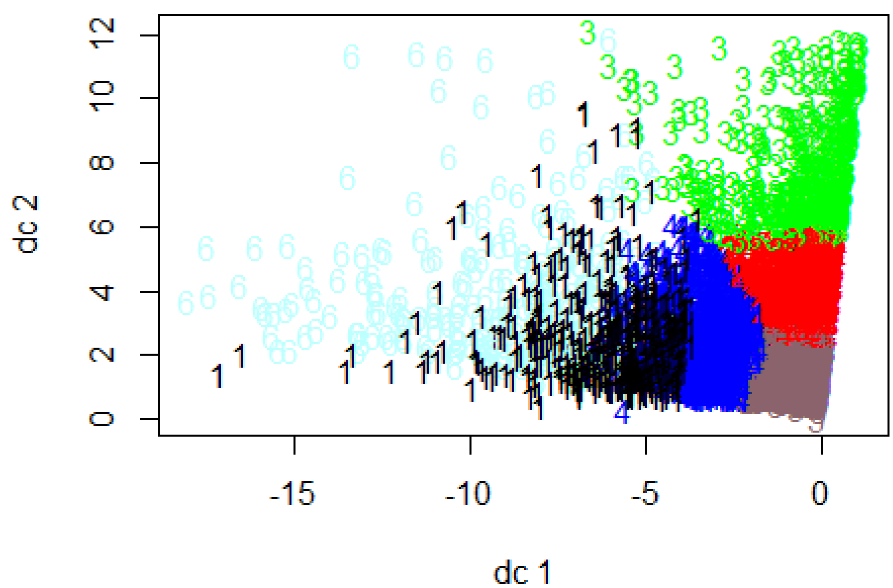
```
## [1] 186.7068
```

```r
mean(Istanbul_cluster4$number_of_reviews)

## [1] 27.27365

mean(Istanbul_cluster4$reviews_per_month)

## [1] 1.663276

head(Istanbul_cluster5)

##    price number_of_reviews reviews_per_month clusterNumber
## 1:   100                41              0.38             5
## 2:   158                10              0.09             5
## 3:   105                11              0.21             5
## 4:   179                16              0.19             5
## 5:   132                33              0.36             5
## 6:   105                 6              0.07             5

mean(Istanbul_cluster5$price)

## [1] 121.8919

mean(Istanbul_cluster5$number_of_reviews)

## [1] 4.596459

mean(Istanbul_cluster5$reviews_per_month)

## [1] 0.4090354

head(Istanbul_cluster6)

##    price number_of_reviews reviews_per_month clusterNumber
## 1:   295               128              1.38             6
## 2:   232               119              1.66             6
## 3:   316                99              1.10             6
## 4:   364               113              1.30             6
## 5:    58               106              1.26             6
## 6:   100               211              2.58             6

mean(Istanbul_cluster6$price)

## [1] 260.4667

mean(Istanbul_cluster6$number_of_reviews)

## [1] 137.0222

mean(Istanbul_cluster6$reviews_per_month)

## [1] 2.736667
```

From observing the mean price and no. of reviews for the all the six clusters,
cluster with mean price of 260 and average rating 137 is the best choice for customers

Now we will plot these clusters

```
library(fpc)

## Warning: package 'fpc' was built under R version 3.6.3

plotcluster(Istanbul_clus,kmeans6.Istanbul$cluster)
```



Clustering Based on Latitude, Longitude and Price

This is our second approach to cluster our dataset based on Latitude, Longitude and Price.

```
#install.packages("cluster", lib="/Library/Frameworks/R.framework/Vers
ions/3.5/Resources/library")
library(cluster)

## Warning: package 'cluster' was built under R version 3.6.2

library(data.table)#Data. table is an extension of data. frame package
in R. It is widely used for fast aggregation of large datasets,

## Warning: package 'data.table' was built under R version 3.6.2

library(Hmisc)#data analysis funs

## Warning: package 'Hmisc' was built under R version 3.6.2

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 3.6.2

## Loading required package: survival

## Warning: package 'survival' was built under R version 3.6.2

## Loading required package: Formula

## Loading required package: ggplot2

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':
##
##     format.pval, units

library(dplyr)

## Warning: package 'dplyr' was built under R version 3.6.2

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:Hmisc':
##
##     src, summarize
```

```
## The following objects are masked from 'package:data.table':
##
##     between, first, last

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

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

library(tidyverse)

## Warning: package 'tidyverse' was built under R version 3.6.2

## -- Attaching packages ---------------------------------------------
------ tidyverse 1.3.0 --

## v tibble  2.1.3     v purrr   0.3.3
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## Warning: package 'tidyr' was built under R version 3.6.2

## Warning: package 'purrr' was built under R version 3.6.2

## Warning: package 'stringr' was built under R version 3.6.2

## -- Conflicts -------------------------------------------------------
tidyverse_conflicts() --
## x dplyr::between()   masks data.table::between()
## x dplyr::filter()    masks stats::filter()
## x dplyr::first()     masks data.table::first()
## x dplyr::lag()       masks stats::lag()
## x dplyr::last()      masks data.table::last()
## x dplyr::src()       masks Hmisc::src()
## x dplyr::summarize() masks Hmisc::summarize()
## x purrr::transpose() masks data.table::transpose()

library(ggplot2)
library(plotly)

## Warning: package 'plotly' was built under R version 3.6.2

##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:Hmisc':
##
##     subplot

## The following object is masked from 'package:ggplot2':
##
##     last_plot

## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout

library(GGally)

## Warning: package 'GGally' was built under R version 3.6.2

## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2

##
## Attaching package: 'GGally'

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

library(ggthemes)

## Warning: package 'ggthemes' was built under R version 3.6.2

library(psych)

## Warning: package 'psych' was built under R version 3.6.2

##
## Attaching package: 'psych'

## The following object is masked from 'package:Hmisc':
##
##     describe

## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```r
library(relaimpo)

## Warning: package 'relaimpo' was built under R version 3.6.2

## Loading required package: MASS

##
## Attaching package: 'MASS'

## The following object is masked from 'package:plotly':
##
##     select

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

## Loading required package: boot

## Warning: package 'boot' was built under R version 3.6.2

##
## Attaching package: 'boot'

## The following object is masked from 'package:psych':
##
##     logit

## The following object is masked from 'package:survival':
##
##     aml

## The following object is masked from 'package:lattice':
##
##     melanoma

## Loading required package: survey

## Warning: package 'survey' was built under R version 3.6.2

## Loading required package: grid

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
##
## Attaching package: 'survey'

## The following object is masked from 'package:Hmisc':
##
##     deff

## The following object is masked from 'package:graphics':
##
##     dotchart

## Loading required package: mitools

## Warning: package 'mitools' was built under R version 3.6.2

## This is the global version of package relaimpo.

## If you are a non-US user, a version with the interesting additional
metric pmvd is available

## from Ulrike Groempings web site at prof.beuth-hochschule.de/groempi
ng.

library(e1071)

## Warning: package 'e1071' was built under R version 3.6.2

##
## Attaching package: 'e1071'

## The following object is masked from 'package:Hmisc':
##
##     impute

library(data.table)
library(fpp)

## Loading required package: forecast

## Warning: package 'forecast' was built under R version 3.6.2

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

## Loading required package: fma

## Warning: package 'fma' was built under R version 3.6.2

##
## Attaching package: 'fma'
```

```
## The following objects are masked from 'package:MASS':
##
##     cement, housing, petrol

## The following object is masked from 'package:GGally':
##
##     pigs

## Loading required package: expsmooth

## Loading required package: lmtest

## Loading required package: zoo

## Warning: package 'zoo' was built under R version 3.6.2

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: tseries

library(fpp2)

##
## Attaching package: 'fpp2'

## The following objects are masked from 'package:fpp':
##
##     ausair, ausbeer, austa, austourists, debitcards, departures,
##     elecequip, euretail, guinearice, oil, sunspotarea, usmelec

library(cowplot)

## Warning: package 'cowplot' was built under R version 3.6.2

##
## ********************************************************

## Note: As of version 1.0.0, cowplot does not change the

##   default ggplot2 theme anymore. To recover the previous

##   behavior, execute:
##   theme_set(theme_cowplot())

## ********************************************************
```

```
##
## Attaching package: 'cowplot'

## The following object is masked from 'package:ggthemes':
##
##     theme_map

library(corrplot)

## Warning: package 'corrplot' was built under R version 3.6.2

## corrplot 0.84 loaded

library(reshape2)

##
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':
##
##     smiths

## The following objects are masked from 'package:data.table':
##
##     dcast, melt
```

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see *http://rmarkdown.rstudio.com*.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

Including Plots

You can also embed plots, for example:

Loading Dataset:

```
AirbnbIstanbul <- read.csv("C:/Alok/OneDrive/Rutgers_MITA/Semester2/MV
A/R/AirbnbIstanbul.csv", stringsAsFactors=FALSE)
Istanbul <- copy(AirbnbIstanbul)
class(Istanbul)

## [1] "data.frame"
```

```r
setDT(Istanbul)

str(Istanbul)

## Classes 'data.table' and 'data.frame':   16251 obs. of  16 variable
s:
##  $ id                            : int  4826 20815 25436 27271 2827
7 28308 28318 29241 30697 33368 ...
##  $ name                          : chr  "The Place" "The Bosphorus
from The Comfy Hill" "House for vacation rental furnutare" "LOVELY APT
. IN PERFECT LOCATION" ...
##  $ host_id                       : int  6603 78838 105823 117026 12
1607 121695 121721 125742 132137 135136 ...
##  $ host_name                     : chr  "Kaan" "GÃ¼lder" "Yesim" "M
utlu" ...
##  $ neighbourhood_group           : logi  NA NA NA NA NA NA ...
##  $ neighbourhood                 : chr  "Uskudar" "Besiktas" "Besik
tas" "Beyoglu" ...
##  $ latitude                      : num  41.1 41.1 41.1 41 41 ...
##  $ longitude                     : num  29.1 29 29 29 29 ...
##  $ room_type                     : chr  "Entire home/apt" "Entire h
ome/apt" "Entire home/apt" "Entire home/apt" ...
##  $ price                         : int  554 100 211 237 591 237 633
264 596 295 ...
##  $ minimum_nights                : int  1 30 21 5 3 1 3 3 1 2 ...
##  $ number_of_reviews             : int  1 41 0 2 0 0 0 0 1 1 ...
##  $ last_review                   : chr  "6/1/2009" "11/7/2018" "" "
5/4/2018" ...
##  $ reviews_per_month             : num  0.01 0.38 NA 0.04 NA NA NA
NA 0.01 0.02 ...
##  $ calculated_host_listings_count: int  1 2 1 1 13 1 1 1 1 2 ...
##  $ availability_365              : int  365 49 83 228 356 365 365 3
65 365 232 ...
##  - attr(*, ".internal.selfref")=<externalptr>

Istanbul[,room_type:=factor(room_type)]
Istanbul[,neighbourhood:=factor(neighbourhood)]
Istanbul[,last_review:=as.Date(last_review,'%Y-%m-%d')] ## converting
last_review to date datatype

# datatypes looks better now. hence will see again for NA values
grep ('NA',Istanbul) # 2, 5, 13 and 14 column have NA values

## [1]  2  5 13 14

Istanbul[is.na(neighbourhood_group),NROW(neighbourhood_group)] # entir
e obs. is blank, will drop this var
```

```
## [1] 16251

Istanbul[is.na(last_review),NROW(last_review)] ## there are 8484 NA va
lues

## [1] 16251

Istanbul[is.na(reviews_per_month),NROW(reviews_per_month)] ## there ar
e 8484 NA values

## [1] 8484

Istanbul$neighbourhood_group <- NULL ## removing neighbourhood_group c
olumn
Istanbul[is.na(reviews_per_month),reviews_per_month:=0] ## nearly 50%
of the dataset is filled with NA.
# hence we can't simply remove these many rows. Hence imputing with 0
values.

#Removing last_review
Istanbul_ip<-Istanbul[,-c(12)]
names(Istanbul_ip)

##  [1] "id"                             "name"
##  [3] "host_id"                        "host_name"
##  [5] "neighbourhood"                  "latitude"
##  [7] "longitude"                      "room_type"
##  [9] "price"                          "minimum_nights"
## [11] "number_of_reviews"              "reviews_per_month"
## [13] "calculated_host_listings_count" "availability_365"

sum(is.na(Istanbul_ip)) #8484

## [1] 0

#To get the column names that have null values
!!colSums(is.na(Istanbul_ip))

##                             id                           name
##                          FALSE                          FALSE
##                        host_id                      host_name
##                          FALSE                          FALSE
##                  neighbourhood                       latitude
##                          FALSE                          FALSE
##                      longitude                      room_type
##                          FALSE                          FALSE
##                          price                 minimum_nights
##                          FALSE                          FALSE
##              number_of_reviews              reviews_per_month
```

```
##                              FALSE                              FALSE
## calculated_host_listings_count                   availability_365
##                              FALSE                              FALSE

#reviews_per_month has NULL values
#how manu null values in reviews_per_month
#sum(is.na(reviews_per_month)) #8484
summary(Istanbul_ip)

##        id                 name              host_id           host_na
me
##  Min.   :    4826   Length:16251       Min.   :     6603   Length:1
6251
##  1st Qu.: 8500978   Class :character   1st Qu.: 17882300   Class :c
haracter
##  Median :21619750   Mode  :character   Median : 52107399   Mode  :c
haracter
##  Mean   :18856396                      Mean   : 88887056
##  3rd Qu.:28702192                      3rd Qu.:168134520
##  Max.   :32457561                      Max.   :243734065
##
##   neighbourhood     latitude       longitude               room_typ
e
##  Beyoglu :4245   Min.   :40.81   Min.   :28.03   Entire home/apt:71
91
##  Sisli   :2348   1st Qu.:41.00   1st Qu.:28.97   Private room   :85
65
##  Fatih   :2146   Median :41.03   Median :28.98   Shared room    : 4
95
##  Kadikoy :1717   Mean   :41.03   Mean   :28.99
##  Besiktas:1367   3rd Qu.:41.05   3rd Qu.:29.02
##  Uskudar : 594   Max.   :41.41   Max.   :29.91
##  (Other) :3834
##      price          minimum_nights     number_of_reviews reviews_per
_month
##  Min.   :    0.0   Min.   :   1.000   Min.   :  0.000   Min.   : 0.
0000
##  1st Qu.:  105.0   1st Qu.:   1.000   1st Qu.:  0.000   1st Qu.: 0.
0000
##  Median :  190.0   Median :   1.000   Median :  0.000   Median : 0.
0000
##  Mean   :  354.7   Mean   :   4.693   Mean   :  7.187   Mean   : 0.
4372
##  3rd Qu.:  327.0   3rd Qu.:   2.000   3rd Qu.:  4.000   3rd Qu.: 0.
4700
##  Max.   :59561.0   Max.   :1125.000   Max.   :307.000   Max.   :12.
```

```
0000
##
##  calculated_host_listings_count availability_365
##  Min.   : 1.000                 Min.   :  0.0
##  1st Qu.: 1.000                 1st Qu.:101.0
##  Median : 1.000                 Median :340.0
##  Mean   : 4.104                 Mean   :249.5
##  3rd Qu.: 4.000                 3rd Qu.:365.0
##  Max.   :77.000                 Max.   :365.0
##

#Imputing zeros where reviews_per_month is null
#reviews_per_month[is.na(reviews_per_month)] <- 0
#Aganin checking for null values after imputation
#sum(is.na(reviews_per_month)) #op=0

names(Istanbul_ip)

##  [1] "id"                             "name"
##  [3] "host_id"                        "host_name"
##  [5] "neighbourhood"                  "latitude"
##  [7] "longitude"                      "room_type"
##  [9] "price"                          "minimum_nights"
## [11] "number_of_reviews"              "reviews_per_month"
## [13] "calculated_host_listings_count" "availability_365"

range(Istanbul$price) ## range of price

## [1]     0 59561

avgNeighbourhood=Istanbul[,avgneighprice:=mean(price),by=neighbourhood
]
Istanbul.1 <- avgNeighbourhood[price > avgneighprice]
head(avgNeighbourhood)

##       id                                name host_id host_name neig
hbourhood
## 1:  4826                           The Place    6603      Kaan
Uskudar
## 2: 20815   The Bosphorus from The Comfy Hill   78838    GÃ¼lder
Besiktas
## 3: 25436 House for vacation rental furnutare  105823     Yesim
Besiktas
## 4: 27271     LOVELY APT. IN PERFECT LOCATION  117026     Mutlu
Beyoglu
## 5: 28277      Duplex Apartment with Terrace   121607      Alen
Sisli
```

```
## 6: 28308      Great apartment in Cihangir...  121695   Mustafa
Beyoglu
##    latitude longitude       room_type price minimum_nights number_o
f_reviews
## 1: 41.05650  29.05367 Entire home/apt   554              1
1
## 2: 41.06984  29.04545 Entire home/apt   100             30
41
## 3: 41.07731  29.03891 Entire home/apt   211             21
0
## 4: 41.03220  28.98216 Entire home/apt   237              5
2
## 5: 41.04471  28.98567 Entire home/apt   591              3
0
## 6: 41.03105  28.98297 Entire home/apt   237              1
0
##    last_review reviews_per_month calculated_host_listings_count
## 1:        <NA>              0.01                              1
## 2:        <NA>              0.38                              2
## 3:        <NA>              0.00                              1
## 4:        <NA>              0.04                              1
## 5:        <NA>              0.00                             13
## 6:        <NA>              0.00                              1
##    availability_365 avgneighprice
## 1:              365      242.5101
## 2:               49      299.4865
## 3:               83      299.4865
## 4:              228      373.1771
## 5:              356      342.1759
## 6:              365      373.1771
```

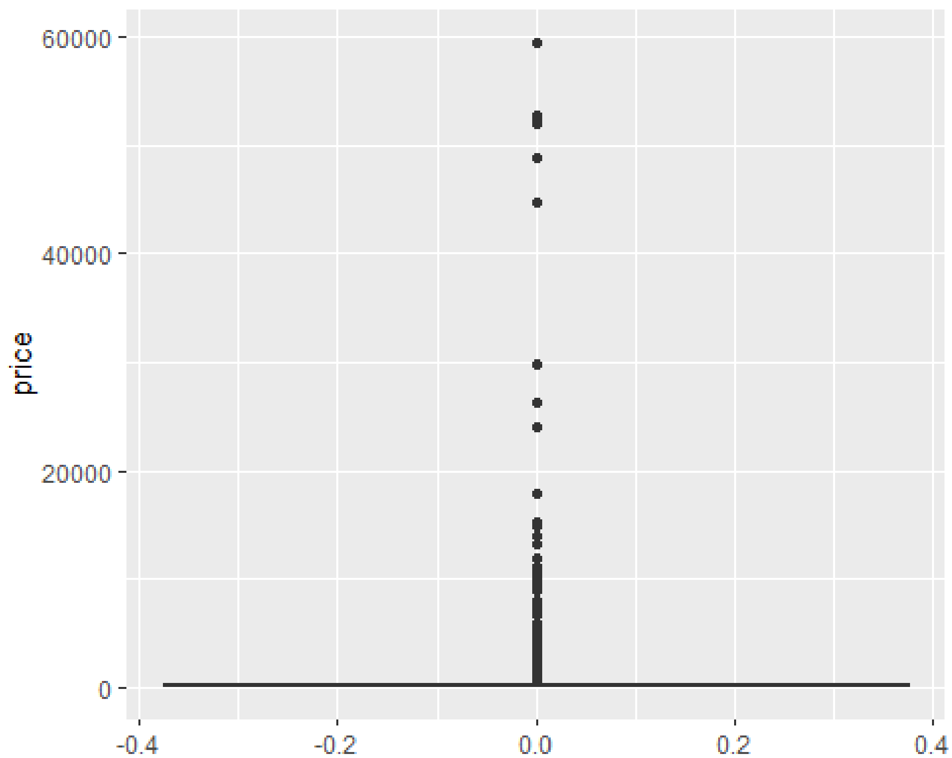
```
summary(Istanbul.1$price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   158.0   385.0   527.0   954.9   749.0 59561.0
```

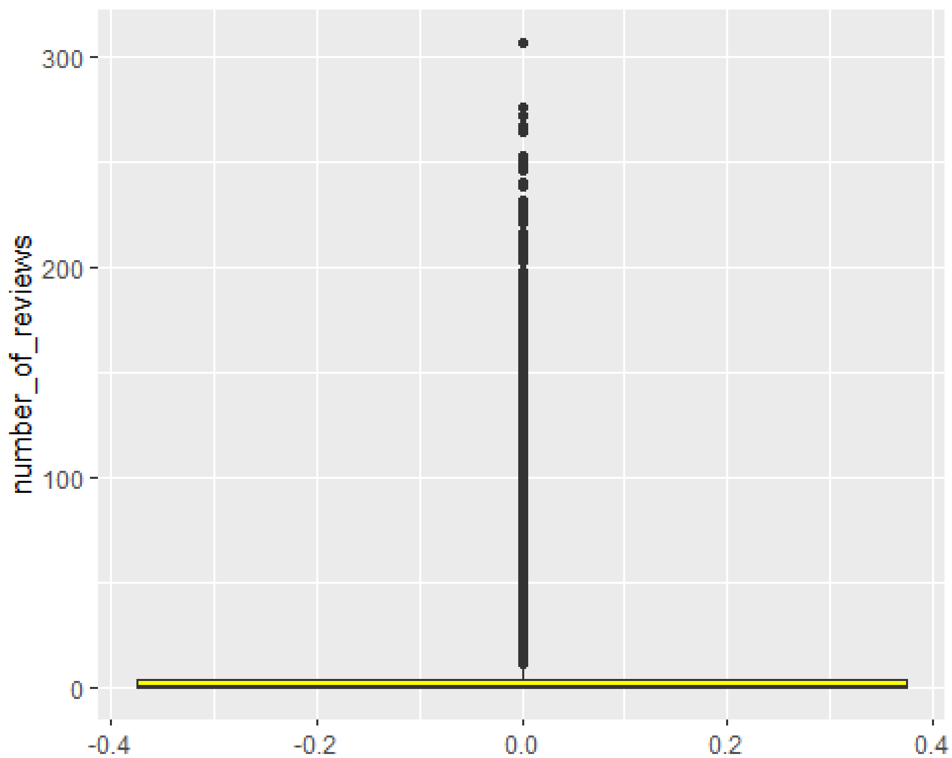
```
summary(Istanbul$price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0   105.0   190.0   354.7   327.0 59561.0
```

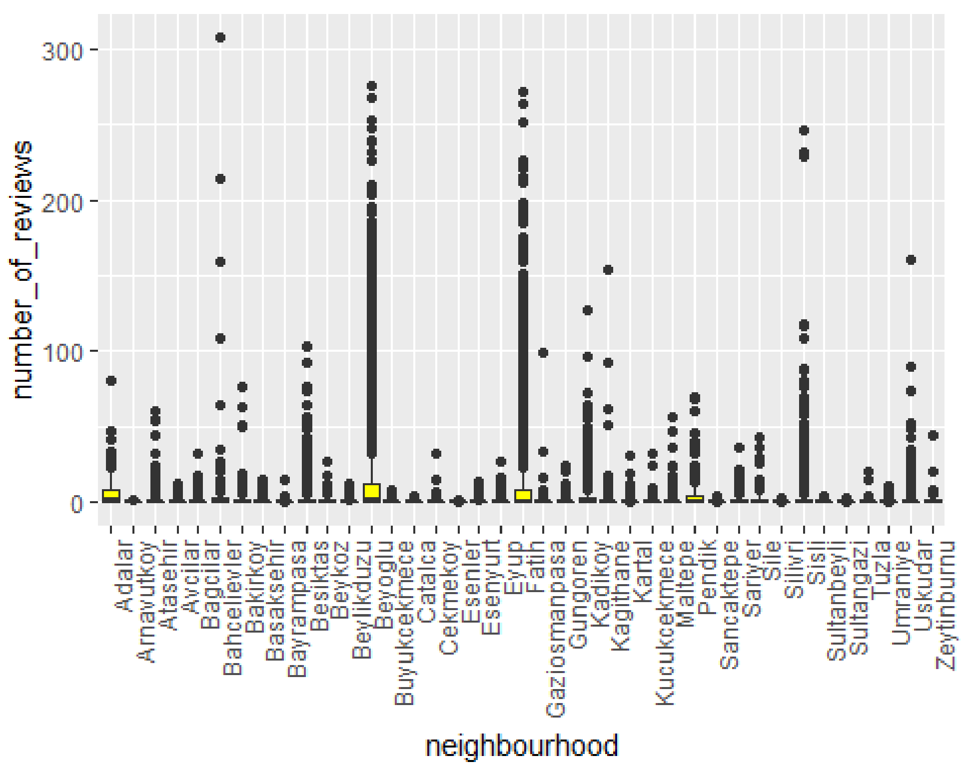
```
ggplot(Istanbul,aes(y=price)) + geom_boxplot(fill='yellow')
```



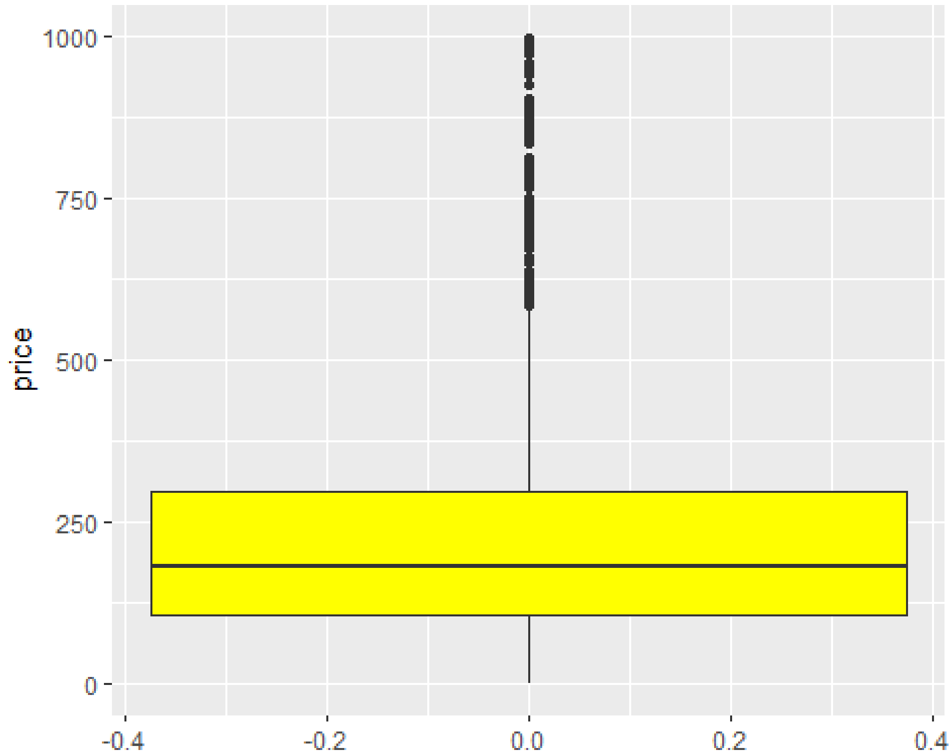

```
#View(Istanbul.1)
## no. of reviews and neighbourhood relation
summary(Istanbul$number_of_reviews)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   0.000   7.187   4.000 307.000

nrow(Istanbul[price > 1000]) ## price > 1000

## [1] 613

ggplot(Istanbul,aes(y=number_of_reviews)) + geom_boxplot(fill='yellow'
)
```




```
ggplot(Istanbul,aes(x=neighbourhood,y=number_of_reviews)) + geom_boxpl
ot(fill='yellow') + theme(axis.text.x = element_text(angle = 90, hjust
= 1))
```



```
Istanbul.clust <- Istanbul[price < 1000 & number_of_reviews > 0] ## pr
ice > 1000
ggplot(Istanbul.clust,aes(y=price)) + geom_boxplot(fill='yellow')
```




```
grep('NA',Istanbul.clust)

## [1]  2 12

names(Istanbul.clust)

##  [1] "id"                             "name"
##  [3] "host_id"                        "host_name"
##  [5] "neighbourhood"                  "latitude"
##  [7] "longitude"                      "room_type"
##  [9] "price"                          "minimum_nights"
## [11] "number_of_reviews"              "last_review"
## [13] "reviews_per_month"              "calculated_host_listings_cou
nt"
## [15] "availability_365"               "avgneighprice"

#Now Istanbul.clust is the input dataset for clustering
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

Clustering Approach 2:

K Means Clustering for Clustering only with latitude longitude and price

```
#K Means Clustering for Clustering only with lattitude longitude and p
rice


library(cluster)
Istanbul_clus2 = data.frame(
  Istanbul.clust$price,
  Istanbul.clust$latitude,
  Istanbul.clust$longitude)

head(Istanbul_clus2)

##   Istanbul.clust.price Istanbul.clust.latitude Istanbul.clust.longi
tude
## 1                  554                41.05650                 29.0
5367
## 2                  100                41.06984                 29.0
4545
## 3                  237                41.03220                 28.9
8216
## 4                  596                41.03350                 28.9
7626
## 5                  295                41.05382                 28.9
9739
## 6                  158                41.07687                 29.0
2714

#Adding ID (property id from original datatset as index)
rownames(Istanbul_clus2) <- Istanbul.clust$id
##Scaling done to make the data on one scale.
Istanbul.Scale1 <- scale(Istanbul_clus2[,1:3])

#Here we have selected first row to see how our scaled matrix is like
head(Istanbul.Scale1,1)

##      Istanbul.clust.price Istanbul.clust.latitude Istanbul.clust.lo
ngitude
## 4826               1.9566               0.8009274                    0
.679537
```

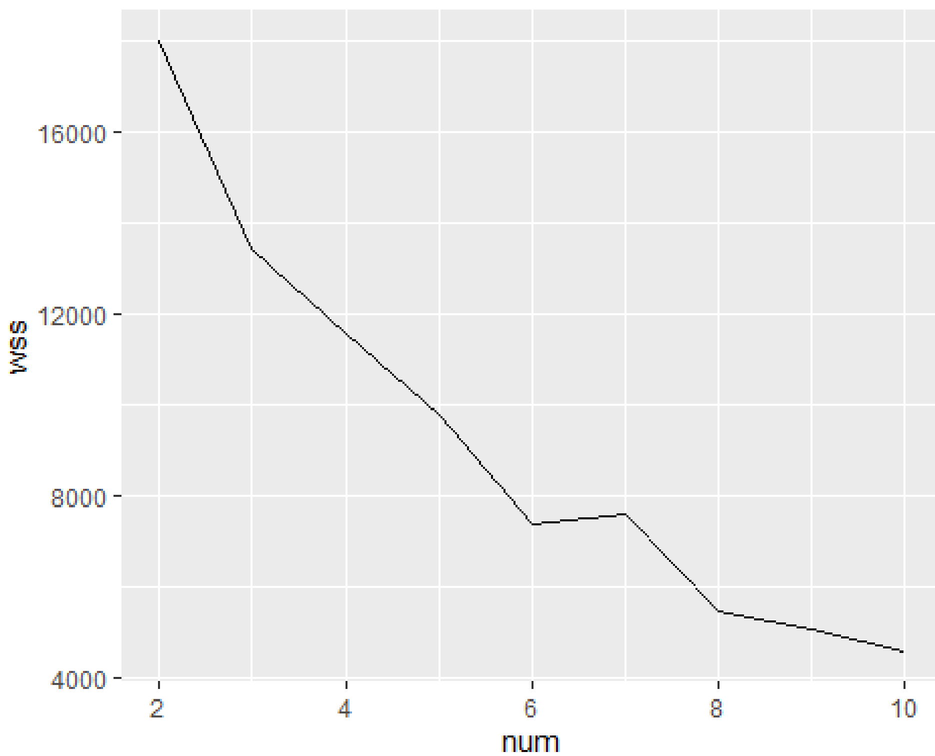
```r
# We will find K-means by taking k=2, 3, 4, 5, 6...
# Centers (k's) are numbers thus, 10 random sets are chosen

#Elbow Plot to Identify the Best number of K Clusters
wss=c()########## empty vector to hold wss
for(i in 2:10)#### from 2 to 10 cluster
{
  km = kmeans(Istanbul.Scale1[,1:3],i)
  wss[i-1]=km$tot.withinss
}
wss
```

```
## [1] 18015.808 13431.861 11571.875  9752.943  7351.072  7593.524  54
57.330
## [8]  5046.983  4569.689
```

```r
## [1] 15197.254 10745.783  7987.996  6808.887  5980.367  5311.900  48
46.853
## [8]  4240.790  3709.000
#Creating a 'elbowdt' data table with column names num and wss with th
e contents of wss
elbowdt = data.table(num=2:10,wss)
elbowdt
```

```
##    num       wss
## 1:   2 18015.808
## 2:   3 13431.861
## 3:   4 11571.875
## 4:   5  9752.943
## 5:   6  7351.072
## 6:   7  7593.524
## 7:   8  5457.330
## 8:   9  5046.983
## 9:  10  4569.689
```

```r
#Plotting
ggplot(elbowdt,aes(x=num,y=wss)) + geom_line()
```

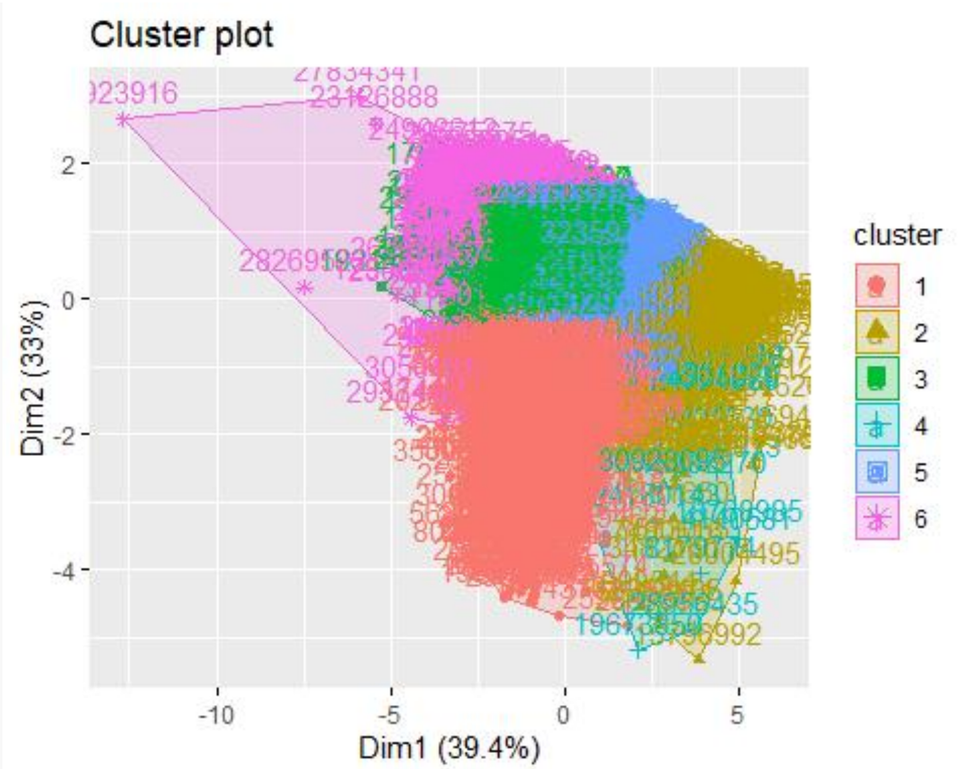


For k = 6 the between sum of square/total sum of square ratio tends to change slowly and remain less changing as compared to others. Therefore, k = 6 should be a good choice for the number of clusters.

For 6 clusters, k-means = 6

```
kmeans6.Istanbul <- kmeans(Istanbul.Scale1,6,nstart = 10)

#Printing
#kmeans6.Istanbul

#plotting output of kmeans for 6 clusters
library(factoextra)

## Warning: package 'factoextra' was built under R version 3.6.3

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

fviz_cluster(kmeans6.Istanbul,data=Istanbul.Scale1)
```

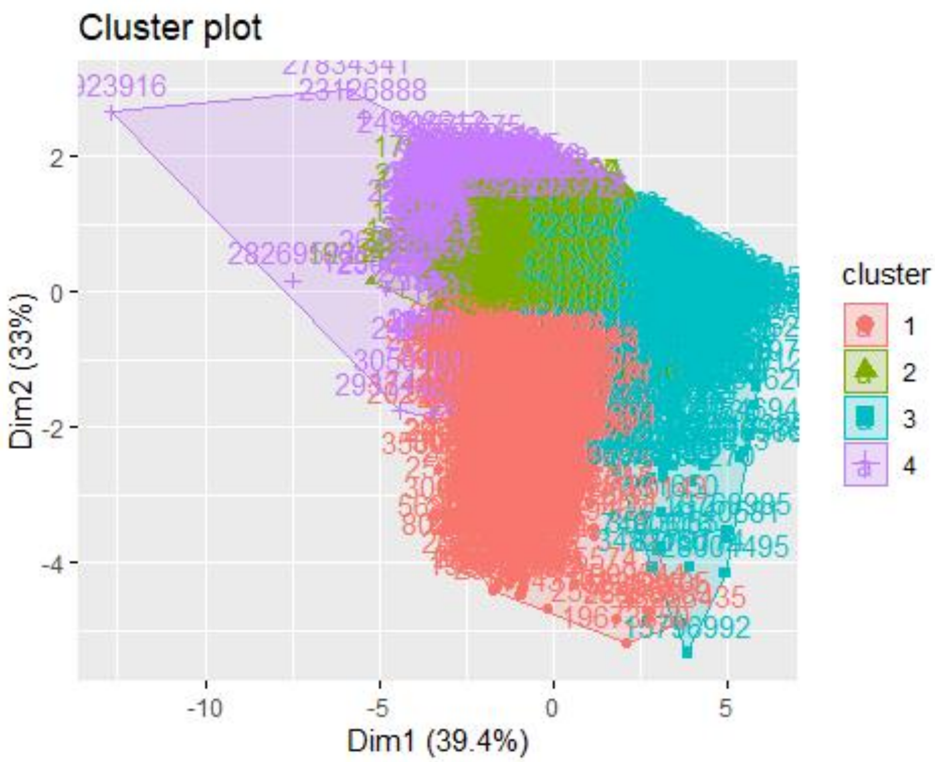
Cluster Plot with k = 6



From above plot, one can not identify the cluster boundaries
Especially for cluster 2
Also, clusters 1 and 6 look bit overlapped.
Hence, I infer that k=6 does not correctly apply clustering on my inpput dataset
As per general idea about my dataset, the Airbnb property locations looks to be divided into 4 major groups
So applying k-means clustering with '4' clusters

```
kmeans4.Istanbul <- kmeans(Istanbul.Scale1,4,nstart = 10)

#Printing
#kmeans4.Istanbul

#plotting output of kmeans
library(factoextra)
fviz_cluster(kmeans4.Istanbul,data=Istanbul.Scale1)
```

Cluster Plot with k = 4



As per above plot, you can see 4 clusters with much clear distinction amongst them

```
# Computing the percentage of variation accounted for two clusters
perc_var_kmeans4 <- round(100*(1 - kmeans4.Istanbul$betweenss/kmeans4.Istanbul$totss),1)
names(perc_var_kmeans4) <- "Perc. 4 clus"
perc_var_kmeans4

## Perc. 4 clus
##         48.3

# Saving four k-means clusters in a list
head(kmeans4.Istanbul$cluster)

##  4826 20815 27271 30697 33368 33580
##     1     2     2     1     2     2

clus1 <- matrix(names(kmeans4.Istanbul$cluster[kmeans4.Istanbul$cluster == 1]),
                ncol=1, nrow=length(kmeans4.Istanbul$cluster[kmeans4.Istanbul$cluster == 1]))
colnames(clus1) <- "Cluster 1"
head(clus1)
```

```
##      Cluster 1
## [1,] "4826"
## [2,] "30697"
## [3,] "35580"
## [4,] "41753"
## [5,] "47264"
## [6,] "52828"
```

```
clus2 <- matrix(names(kmeans4.Istanbul$cluster[kmeans4.Istanbul$cluste
r == 2]),
                ncol=1, nrow=length(kmeans4.Istanbul$cluster[kmeans4.I
stanbul$cluster == 2]))
colnames(clus2) <- "Cluster 2"
head(clus2)
```

```
##      Cluster 2
## [1,] "20815"
## [2,] "27271"
## [3,] "33368"
## [4,] "33580"
## [5,] "33730"
## [6,] "34177"
```

```
clus3 <- matrix(names(kmeans4.Istanbul$cluster[kmeans4.Istanbul$cluste
r == 3]),
                ncol=1, nrow=length(kmeans4.Istanbul$cluster[kmeans4.I
stanbul$cluster == 3]))
colnames(clus3) <- "Cluster 3"
head(clus3)
```

```
##      Cluster 3
## [1,] "81016"
## [2,] "130225"
## [3,] "139804"
## [4,] "155757"
## [5,] "223519"
## [6,] "230296"
```

```
clus4 <- matrix(names(kmeans4.Istanbul$cluster[kmeans4.Istanbul$cluste
r == 4]),
                ncol=1, nrow=length(kmeans4.Istanbul$cluster[kmeans4.I
stanbul$cluster == 4]))
colnames(clus4) <- "Cluster 4"
head(clus4)
```

```
##      Cluster 4
## [1,] "284645"
```

```
## [2,] "478289"
## [3,] "511597"
## [4,] "553409"
## [5,] "684354"
## [6,] "767323"
```

```r
list(clus1,clus2,clus3,clus4)
```

```
## [[1]]
##         Cluster 1
##    [1,] "4826"
##    [2,] "30697"
##    [3,] "35580"
##    [4,] "41753"
##    [5,] "47264"
##    [6,] "52828"
##    [7,] "53612"
##    [8,] "108038"
##    [9,] "114786"
##   [10,] "130231"
##   [11,] "139351"
##   [12,] "162180"
##   [13,] "164216"
##   [14,] "171593"
##   [15,] "181146"
##   [16,] "213816"
##   [17,] "220149"
##   [18,] "226255"
##   [19,] "229498"
##   [20,] "248304"
##   [21,] "253055"
##   [22,] "260378"
##   [23,] "277589"
##   [24,] "280776"
##   [25,] "282148"
##   [26,] "282266"
##   [27,] "282289"
##   [28,] "282295"
##   [29,] "282881"
##   [30,] "290608"
##   [31,] "293754"
##   [32,] "308216"
##   [33,] "314848"
##   [34,] "324576"
##   [35,] "327123"
##   [36,] "371051"
```

```
##  [37,] "378120"
##  [38,] "391645"
##  [39,] "408294"
##  [40,] "412766"
##  [41,] "423764"
##  [42,] "429200"
##  [43,] "516610"
##  [44,] "519364"
##  [45,] "520189"
##  [46,] "520238"
##  [47,] "529151"
##  [48,] "537090"
##  [49,] "541989"
##  [50,] "556293"
##  [51,] "559714"
##  [52,] "568452"
##  [53,] "581984"
##  [54,] "595615"
##  [55,] "607323"
##  [56,] "607344"
##  [57,] "612327"
##  [58,] "629373"
##  [59,] "632120"
##  [60,] "638542"
##  [61,] "645976"
##  [62,] "651039"
##  [63,] "651276"
##  [64,] "652580"
##  [65,] "652610"
##  [66,] "652635"
##  [67,] "654046"
##  [68,] "670658"
##  [69,] "680374"
##  [70,] "705394"
##  [71,] "713217"
##  [72,] "723680"
##  [73,] "728601"
##  [74,] "736719"
##  [75,] "739616"
##  [76,] "743975"
##  [77,] "745923"
##  [78,] "759095"
##  [79,] "767463"
##  [80,] "782389"
##  [81,] "785494"
##  [82,] "788715"
```

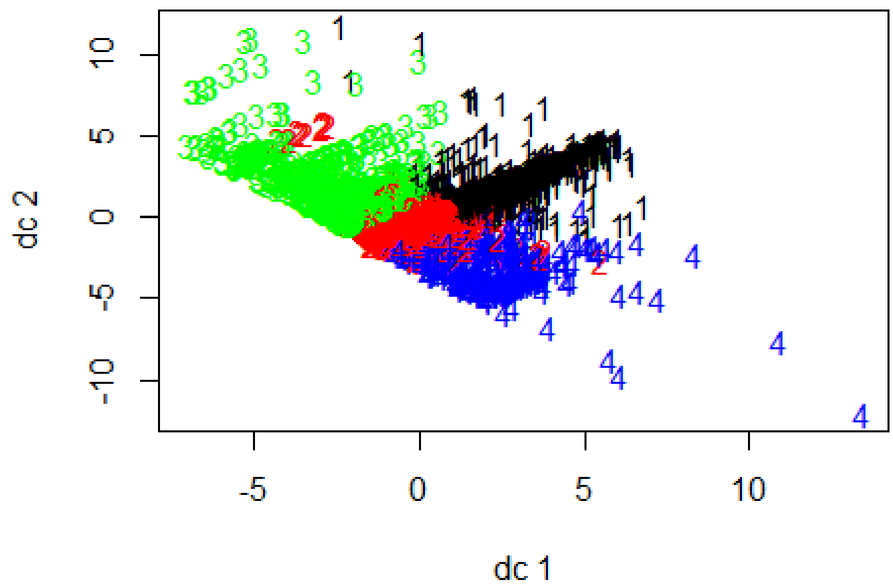
```
##   [83,] "790189"
##   [84,] "791056"
##   [85,] "800706"
##   [86,] "804340"
##   [87,] "813948"
##   [88,] "814023"
##   [89,] "814155"
##   [90,] "846376"
##   [91,] "846394"
##   [92,] "849225"
##   [93,] "854850"
##
## [[3]]
##         Cluster 3
##    [1,] "81016"
##    [2,] "130225"
##    [3,] "139804"
##    [4,] "155757"
##    [5,] "223519"
##    [6,] "230296"
##    [7,] "237993"
##    [8,] "237994"
##    [9,] "241516"
##   [10,] "277581"
##   [11,] "277592"
##   [12,] "318933"
##   [13,] "378383"
## [1199,] "32297760"
## [1200,] "32359796"
##
## [[4]]
##        Cluster 4
##   [1,] "284645"
##   [2,] "478289"
##   [3,] "511597"
##   [4,] "553409"
##   [5,] "684354"
##   [6,] "767323"
##   [7,] "1086151"
##   [8,] "1092753"
##   [9,] "1120982"
##  [10,] "2048291"
##  [11,] "2159879"
##  [12,] "2196825"
##  [13,] "2328304"
##  [14,] "2614081"
```

```
##  [15,] "2648610"
##  [16,] "2821675"
##  [17,] "3355627"
##  [18,] "3500483"
##  [19,] "3622631"
##  [20,] "3623153"
##  [21,] "4042565"
##  [22,] "4132752"
##  [23,] "4144789"
##  [24,] "4144879"
##  [25,] "4151263"
##  [26,] "4265793"
##  [27,] "4281028"
##  [28,] "4326275"
##  [29,] "4688064"
##  [30,] "4729540"
##  [31,] "4729778"
##  [32,] "4853218"
##  [33,] "5066978"
##  [34,] "5718337"
##  [35,] "5760115"
##  [36,] "6178029"
##  [37,] "6225962"
##  [38,] "6447205"
##  [39,] "6539928"
##  [40,] "6620290"
##  [41,] "6691931"
##  [42,] "6736330"
##  [43,] "6817784"
##  [44,] "7015248"
##  [45,] "7355084"
## [396,] "31281882"
## [397,] "31336755"
## [398,] "31395835"
## [399,] "31459080"
## [400,] "31588413"
## [401,] "31679018"
## [402,] "31713334"
## [403,] "31839884"
## [404,] "31845503"
## [405,] "32004014"
## [406,] "32085065"
## [407,] "32112345"
## [408,] "32360375"
```

```
#This is the clusters having groups of property ids
#Trying to print the Price and longitude lattitude corresponding to th
ese ids

out <- cbind(Istanbul.Scale1, clusterNum = kmeans4.Istanbul$cluster)
#This is the input dataset with respective Clusters assigned to them

head(out,5)

##       Istanbul.clust.price Istanbul.clust.latitude Istanbul.clust.l
ongitude
## 4826           1.95659979               0.8009274               0.
67953702
## 20815         -0.77571136               1.1384843               0.
59255780
## 27271          0.04879663               0.1860374              -0.
07713989
## 30697          2.20936866               0.2189327              -0.
13957023
## 33368          0.39785841               0.7331124               0.
08401505
##       clusterNum
## 4826           1
## 20815          2
## 27271          2
## 30697          1
## 33368          2

#View(kmeans4.Istanbul)
```

Plotting these clusters

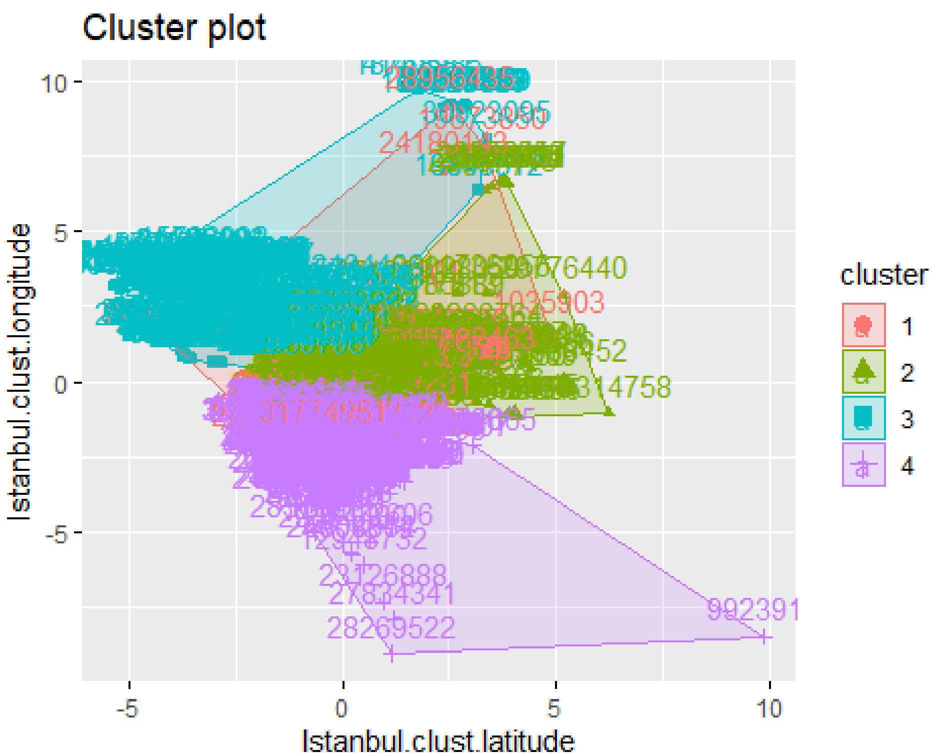
```
#fviz_cluster(kmeans4.Istanbul,data=Istanbul.Scale1)

#other way of plotting the clusters
library(fpc)

## Warning: package 'fpc' was built under R version 3.6.3

plotcluster(Istanbul.Scale1,kmeans4.Istanbul$cluster)
```




```r
#str(out)
#View(out)
```

Trying plotting only with Latitudes and Longitudes to see if the clustering is done based on locations

```r
#View(Istanbul_clus2)
names(Istanbul_clus2)

## [1] "Istanbul.clust.price"     "Istanbul.clust.latitude"
## [3] "Istanbul.clust.longitude"

onlylattitudeLongitude<-Istanbul_clus2[,-c(1)]
#onlyprice<-data.frame(Istanbul_clus2$price)
names(onlylattitudeLongitude)

## [1] "Istanbul.clust.latitude"  "Istanbul.clust.longitude"

#View(onlyprice)
#Plotting for only Lattitude and Longitude
fviz_cluster(kmeans4.Istanbul,data=onlylattitudeLongitude)
```




```
#They do not seem to be divided as per the latitudes and longitudes
#plotcluster(onlylattitudeLongitude,kmeans4.Istanbul$cluster)
```

Making Subsets for 4 clusters using Row filtering from the Original datase t (Not the scaled one)

So below are the 4 cluster sets of Original entire dataset

```
AirIstanbul_clust1<-subset(Istanbul_ip,Istanbul_ip$id %in% clus1)
AirIstanbul_clust2<-subset(Istanbul_ip,Istanbul_ip$id %in% clus2)
AirIstanbul_clust3<-subset(Istanbul_ip,Istanbul_ip$id %in% clus3)
AirIstanbul_clust4<-subset(Istanbul_ip,Istanbul_ip$id %in% clus4)

head(AirIstanbul_clust1,3)
```

```
##       id                           name host_id host_name neighbourhood
latitude
## 1:  4826                      The Place    6603      Kaan       Uskudar
41.05650
## 2: 30697 nice home in popular area  132137       Nan       Beyoglu
41.03350
## 3: 35580   Sea View terrace  House  153032    Michel       Beyoglu
41.03658
```

```
##    longitude       room_type price minimum_nights number_of_reviews
## 1:  29.05367 Entire home/apt   554              1                 1
## 2:  28.97626    Private room   596              1                 1
## 3:  28.97213 Entire home/apt   359             60                37
##    reviews_per_month calculated_host_listings_count availability_36
5
## 1:              0.01                              1              36
5
## 2:              0.01                              1              36
5
## 3:              0.59                              2              33
9
```

```r
head(AirIstanbul_clust2,3)
```

```
##       id                             name host_id host_name neighb
ourhood
## 1: 20815 The Bosphorus from The Comfy Hill   78838   GÃ¼lder      B
esiktas
## 2: 27271   LOVELY APT. IN PERFECT LOCATION  117026     Mutlu
Beyoglu
## 3: 33368 Deluxe double bedroom @ Nisantasi  135136     Ozlem
Sisli
##    latitude longitude       room_type price minimum_nights number_o
f_reviews
## 1: 41.06984  29.04545 Entire home/apt   100             30
41
## 2: 41.03220  28.98216 Entire home/apt   237              5
2
## 3: 41.05382  28.99739    Private room   295              2
1
##    reviews_per_month calculated_host_listings_count availability_36
5
## 1:              0.38                              2               4
9
## 2:              0.04                              1              22
8
## 3:              0.02                              2              23
2
```

```r
head(AirIstanbul_clust3,3)
```

```
##        id                            name host_id host_name neighb
ourhood
## 1:  81016 wake up with a gorgeous sea view  438714      Esin
Adalar
## 2: 130225           Room in a modern home.  641487       Efe
```

```
Kadikoy
## 3: 139804  Entire house in central Kadikoy  681763     Deniz
Kadikoy
##    latitude longitude       room_type price minimum_nights number_o
f_reviews
## 1: 40.86947  29.11737 Entire home/apt   322              2
81
## 2: 40.97618  29.04442    Private room   264              1
1
## 3: 40.98373  29.02865 Entire home/apt   237              4
7
##    reviews_per_month calculated_host_listings_count availability_36
5
## 1:              0.99                              2              33
7
## 2:              0.01                              1              36
5
## 3:              0.08                              1              36
2
```

```r
head(AirIstanbul_clust4,3)
```

```
##        id                                        name host_id host
_name
## 1: 284645          Cute room opening to garden in Moda  748852
Asiye
## 2: 478289 ULTRA LUXE RESIDENCE WITH FREE SWIM.POOL ETC 2368759
Angie
## 3: 511597          Flats in Taksim 2 min. to square #2 2519004 Veda
t Ã–z
##    neighbourhood latitude longitude       room_type price minimum_n
ights
## 1:  Bahcelievler 41.00837  28.85343    Private room   158
5
## 2:    Basaksehir 41.07180  28.67921 Entire home/apt   179
6
## 3:      Bagcilar 41.03183  28.84544 Entire home/apt   179
2
##    number_of_reviews reviews_per_month calculated_host_listings_cou
nt
## 1:                 5              0.06
2
## 2:                12              0.28
1
## 3:                 2              0.03
1
```

```
##    availability_365
## 1:              365
## 2:              261
## 3:              365
```

As per above head outputs, the clusters are formed based on locations

Checking the means of these 4 clusters

```
kmeans4.Istanbul$centers

##   Istanbul.clust.price Istanbul.clust.latitude Istanbul.clust.longi
tude
## 1            1.6887086               0.1936568              -0.0396
6466
## 2           -0.3639629               0.3334421              -0.0423
1894
## 3           -0.3875858              -1.4108086               1.0849
3101
## 4           -0.2532250              -0.2743288              -2.5801
2824

#Printing Neighbourhoods particular to the clusters to check if they a
re saggregated based on neighbourhoods
unique(Istanbul.1$neighbourhood) #We have total 39 unique neighbourhoo
ds

##  [1] Uskudar       Sisli         Beyoglu       Besiktas      Ataseh
ir
##  [6] Kadikoy       Kagithane     Adalar        Sariyer       Maltep
e
## [11] Bakirkoy      Esenyurt      Beykoz        Basaksehir    Gazios
manpasa
## [16] Bahcelievler  Fatih         Silivri       Beylikduzu    Umrani
ye
## [21] Sile          Cekmekoy      Bagcilar      Sancaktepe    Pendik
## [26] Kartal        Buyukcekmece  Gungoren      Eyup          Catalc
a
## [31] Avcilar       Zeytinburnu   Tuzla         Sultanbeyli   Esenle
r
## [36] Bayrampasa    Sultangazi    Kucukcekmece  Arnavutkoy
## 39 Levels: Adalar Arnavutkoy Atasehir Avcilar Bagcilar ... Zeytinbu
rnu

unique(AirIstanbul_clust1$neighbourhood)

##  [1] Uskudar       Beyoglu       Besiktas      Fatih         Kadiko
y
##  [6] Gaziosmanpasa Bahcelievler  Sisli         Sariyer       Beykoz
```

```
## [11] Kagithane     Gungoren      Bagcilar      Basaksehir    Bakirk
oy
## [16] Eyup          Kartal        Adalar        Atasehir      Cekmek
oy
## [21] Sile          Umraniye      Kucukcekmece  Maltepe       Zeytin
burnu
## 39 Levels: Adalar Arnavutkoy Atasehir Avcilar Bagcilar ... Zeytinbu
rnu
```

```
unique(AirIstanbul_clust2$neighbourhood)
```

```
##  [1] Besiktas      Beyoglu       Sisli         Beykoz        Uskuda
r
##  [6] Fatih         Sariyer       Kagithane     Kadikoy       Gazios
manpasa
## [11] Zeytinburnu   Eyup          Gungoren      Sile          Bayram
pasa
## [16] Sultangazi    Esenler       Bagcilar      Bakirkoy      Cekmek
oy
## [21] Umraniye
## 39 Levels: Adalar Arnavutkoy Atasehir Avcilar Bagcilar ... Zeytinbu
rnu
```

```
unique(AirIstanbul_clust3$neighbourhood)
```

```
##  [1] Adalar      Kadikoy     Pendik      Atasehir    Maltepe     Sa
ncaktepe
##  [7] Cekmekoy    Uskudar     Kartal      Tuzla       Umraniye    Si
le
## [13] Sultanbeyli
## 39 Levels: Adalar Arnavutkoy Atasehir Avcilar Bagcilar ... Zeytinbu
rnu
```

```
unique(AirIstanbul_clust4$neighbourhood)
```

```
##  [1] Bahcelievler Basaksehir   Bagcilar     Buyukcekmece Beylikduzu
##  [6] Bakirkoy     Kucukcekmece Esenyurt     Avcilar      Catalca
## [11] Gungoren     Silivri      Esenler
## 39 Levels: Adalar Arnavutkoy Atasehir Avcilar Bagcilar ... Zeytinbu
rnu
```

```
#Lets check average Price in these clusters
mean(AirIstanbul_clust1$price)
```

```
## [1] 509.4873
```

```
mean(AirIstanbul_clust2$price)
```

```
## [1] 168.416
```

```r
mean(AirIstanbul_clust3$price)
```

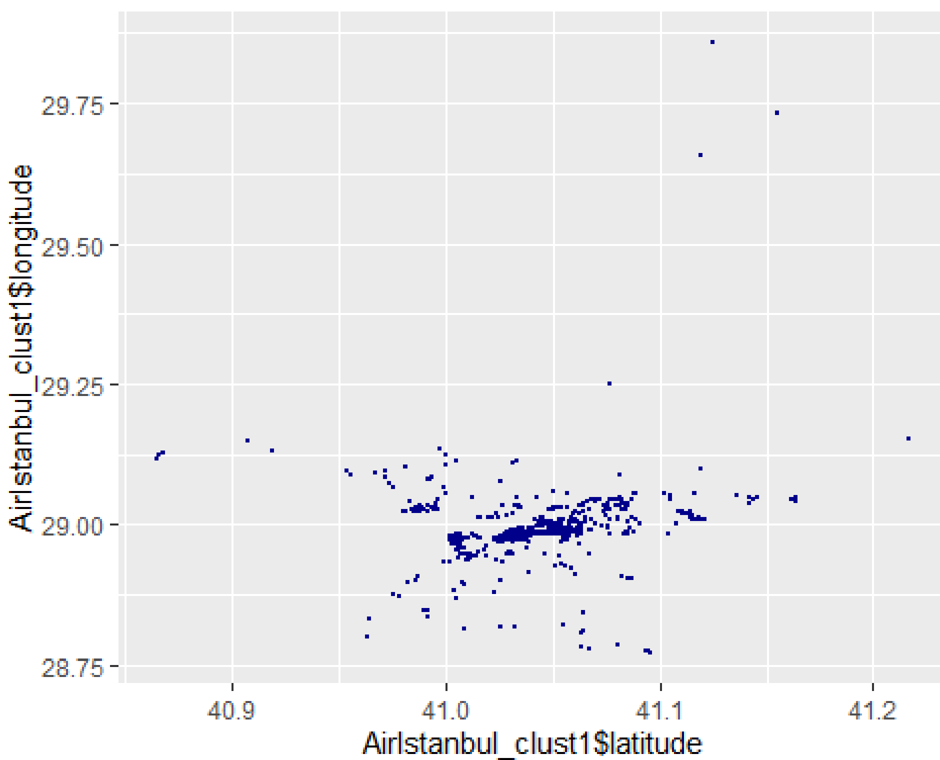
```
## [1] 164.4908
```

```r
mean(AirIstanbul_clust4$price)
```

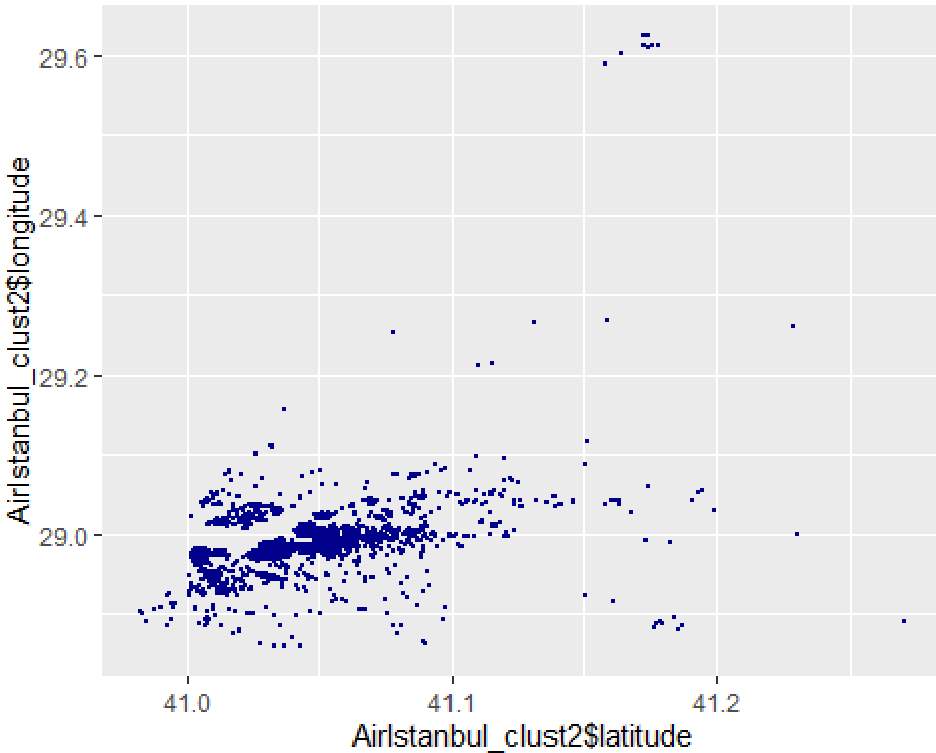
```
## [1] 186.8162
```

The Properties in clusters 1,3 and 4 are pretty much affordable as mean Price around $180
Cluster 2 properties are very expensive ones
Plotting cluster1

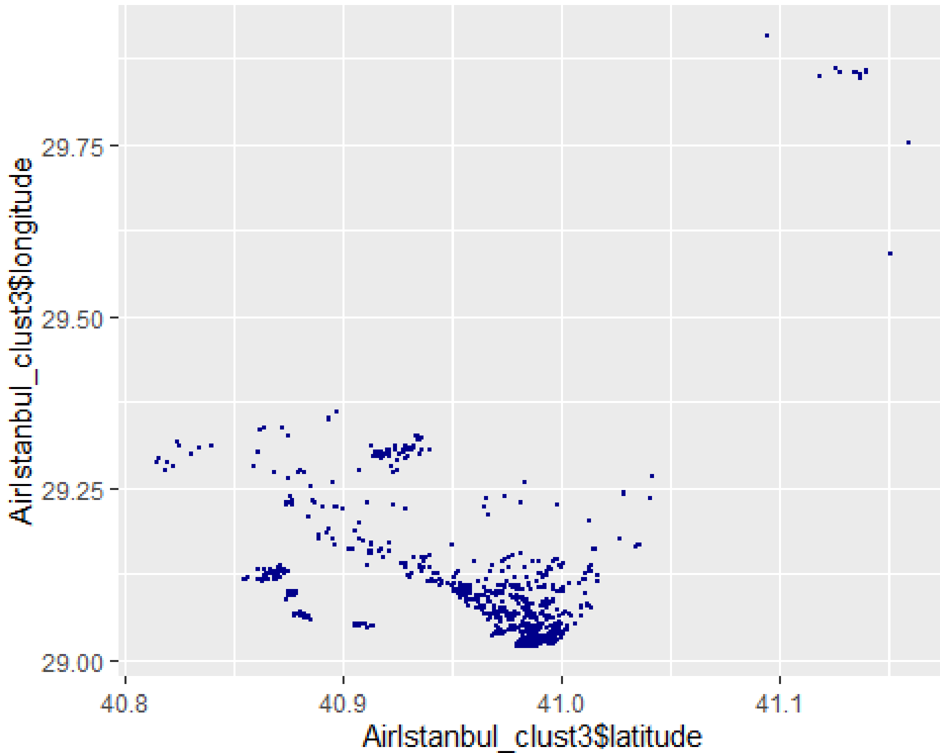
```r
ggplot(AirIstanbul_clust1,
aes(x=AirIstanbul_clust1$latitude,y=AirIstanbul_clust1$longitude))+
geom_point(size=0.1,color='dark blue')
```



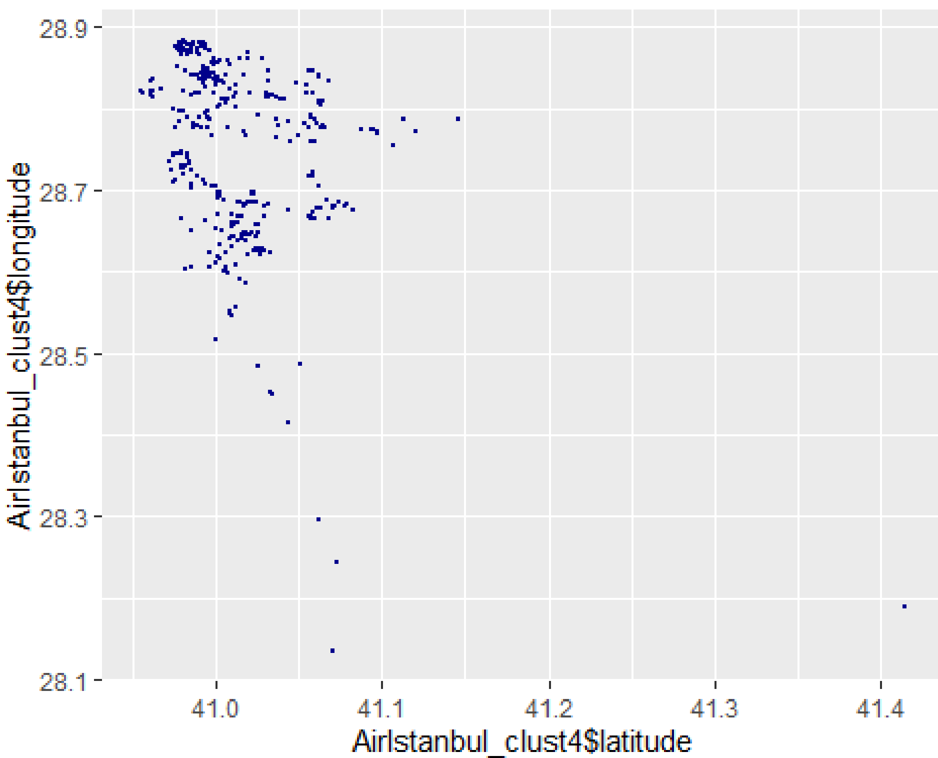

```r
#Plotting cluster2

ggplot(AirIstanbul_clust2,
       aes(x=AirIstanbul_clust2$latitude,y=AirIstanbul_clust2$longitud
e))+
  geom_point(size=0.1,color='dark blue')
```




```
#Plotting cluster3

ggplot(AirIstanbul_clust3,
       aes(x=AirIstanbul_clust3$latitude,y=AirIstanbul_clust3$longitud
e))+
  geom_point(size=0.1,color='dark blue')
```



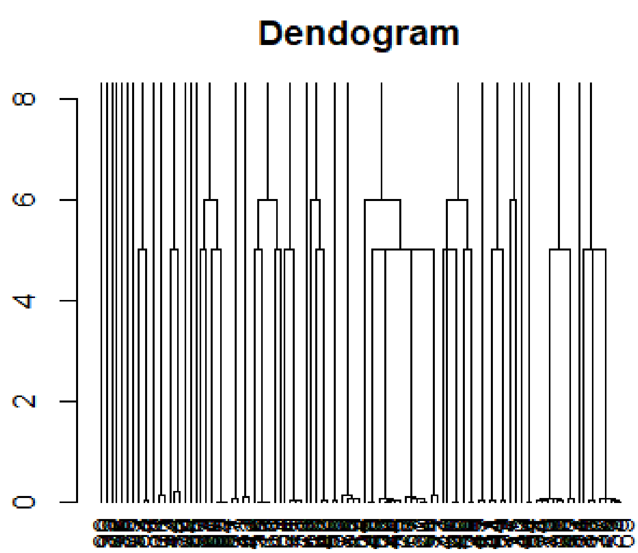

```
#Plotting cluster4

ggplot(AirIstanbul_clust4,
       aes(x=AirIstanbul_clust4$latitude,y=AirIstanbul_clust4$longitud
e))+
  geom_point(size=0.1,color='dark blue')
```



The above 4 graphs show
How the Properties are clustered as per Price and longitudes and latitudes
.

```
######## hierarchical clustering ##########
# Since our dataset is too large, the dendogram will not be upto the m
ark. Thus we have taken a small subset of data and plotted the dendogr
am of it.
library(data.table)
Istanbul_clus <- Istanbul.clust[,c("latitude","longitude","price","min
imum_nights","number_of_reviews","reviews_per_month","calculated_host_
listings_count","availability_365")]
dist_Istanbul <- dist(Istanbul_clus, method="euclidean")
Istanbul.hclust <- hclust(dist_Istanbul, method = "single")
#plot(as.dendrogram(Istanbul.hclust),ylab="Distance between..",ylim=c(
0,2.5),main="Dendrogram of..")
dim(dist_Istanbul)

## NULL

head(dist_Istanbul)

## [1] 555.35046 345.36214  42.00008 291.15632 396.10858 485.77365
```

```
#airbnb <- read.csv("C:/Users/prach/Desktop/MVA/Copy_of_AirbnbIstanbul
.csv",stringsAsFactors = FALSE)
Istanbul_clus2 = data.frame(
  Istanbul.clust$price,
  Istanbul.clust$latitude,
  Istanbul.clust$longitude)
View(Istanbul_clus2)
dim(Istanbul_clus2)

## [1] 7581    3

# Standardizing the data with scale()
matstd_airbnb <- scale(Istanbul_clus2[,1:3])

#Only 100 rows have been used to plot the dendogram
matstd_airbnb <- Istanbul_clus2[1:100,]

# Creating a (Euclidean) distance matrix of the standardized data
dist.Istanbul_clus2 <- dist(matstd_airbnb, method="euclidean")

# Invoking hclust command (cluster analysis by single linkage method)
clusairbnb.nn <- hclust(dist.Istanbul_clus2, method = "single")

#Plotting
# Create extra margin room in the dendrogram, on the bottom (Countries
labels)
par(mar=c(4, 5, 3, 4) + 0.1)
plot(as.dendrogram(clusairbnb.nn),main="Dendogram",ylim = c(0,8))
```



```
#Horizontal Dendrogram
dev.new()
par(mar=c(4, 5, 6, 4) +0.1)
plot(as.dendrogram(clusairbnb.nn), xlim=c(8,0),horiz = TRUE,main="Dend
ogram")
```