# CHAPTER I : INTRODUCTION

## 1.1 Predictive Analytics

Predictive analytics is the use of advanced analytic techniques that leverage historical data to uncover real-time insights and to predict future events. The use of predictive analytics is a key milestone on your analytics journey - a point of confluence where classical statistical analysis meets the new world of Artificial Intelligence (AI).

- Growing volumes and types of data, and more interest in using data to produce valuable insights.
- Faster, cheaper computers.
- Easier-to-use software.
- Tougher economic conditions and a need for competitive differentiation.

Any industry can use predictive analytics to reduce risks, optimize operations and increase revenue. With predictive analytics, you can go beyond learning what happened and why to discovering insights about the future.

## 1.2 OVERVIEW

House price index represents the summarized price changes of residential housing. While for a single family house price predication, it needs more accurate method based on location, house type, size and some other factors which could affect house demand and supply. With limited dataset and data features, a practical and composite data pre-processing, creative feature engineering method is examined in this paper.

The dataset consists of samples collected from IIM, Bangalore. The paper proposes a random forest and boosting model to predict individual house price.

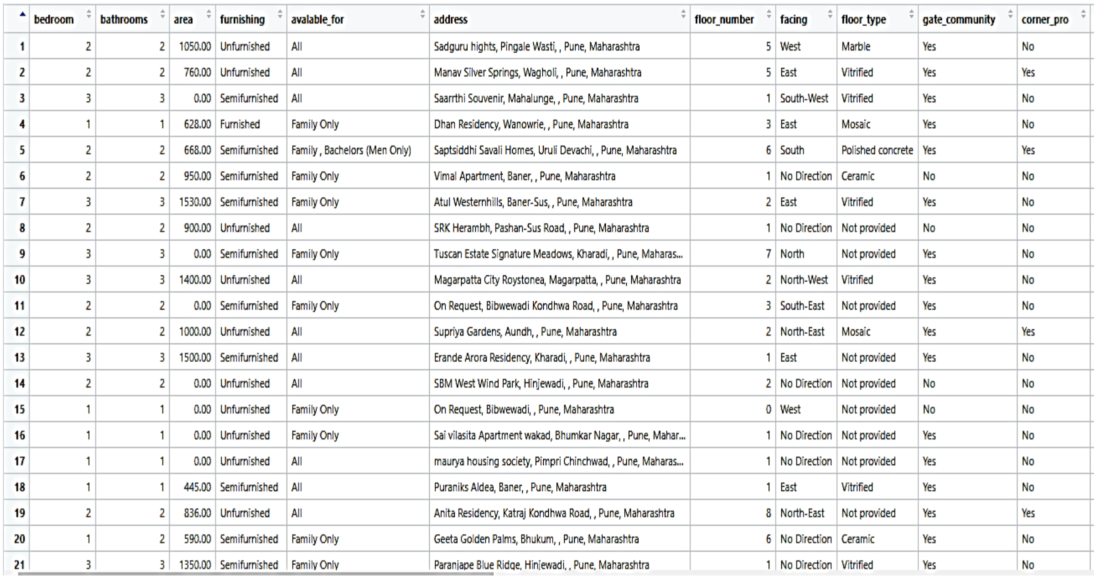
# Chapter II : Gathering Data

## 2.1 Data description:

Pune is IT capital of India. Every software engineer from India wanted to work in this city. So many apartments has rented.
I wanted to predict rent for both.

1. for owner who wanted to rent their home/ apartment
2. for customers who wanted to find home on rent

My aim is that predict home rent price on given data. This dataset has 10884 rows and 30 columns.



| | bedroom | bathrooms | area | furnishing | avalable_for | address | floor_number | facing | floor_type | gate_community | corner_pro |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 2 | 1050.00 | Unfurnished | All | Sadguru hights, Pingale Wasti, , Pune, Maharashtra | 5 | West | Marble | Yes | No |
| 2 | 2 | 2 | 760.00 | Unfurnished | All | Manav Silver Springs, Wagholi, , Pune, Maharashtra | 5 | East | Vitrified | Yes | Yes |
| 3 | 3 | 3 | 0.00 | Semifurnished | All | Saarrthi Souvenir, Mahalunge, , Pune, Maharashtra | 1 | South-West | Vitrified | Yes | No |
| 4 | 1 | 1 | 628.00 | Furnished | Family Only | Dhan Residency, Wanowrie, , Pune, Maharashtra | 3 | East | Mosaic | Yes | No |
| 5 | 2 | 2 | 668.00 | Semifurnished | Family , Bachelors (Men Only) | Saptsiddhi Savali Homes, Uruli Devachi, , Pune, Maharashtra | 6 | South | Polished concrete | Yes | Yes |
| 6 | 2 | 2 | 950.00 | Semifurnished | Family Only | Vimal Apartment, Baner, , Pune, Maharashtra | 1 | No Direction | Ceramic | No | No |
| 7 | 3 | 3 | 1530.00 | Semifurnished | Family Only | Atul Westernhills, Baner-Sus, , Pune, Maharashtra | 2 | East | Vitrified | Yes | No |
| 8 | 2 | 2 | 900.00 | Unfurnished | All | SRK Herambh, Pashan-Sus Road, , Pune, Maharashtra | 1 | No Direction | Not provided | No | No |
| 9 | 3 | 3 | 0.00 | Semifurnished | Family Only | Tuscan Estate Signature Meadows, Kharadi, , Pune, Maharas... | 7 | North | Not provided | Yes | No |
| 10 | 3 | 3 | 1400.00 | Unfurnished | All | Magarpatta City Roystonea, Magarpatta, , Pune, Maharashtra | 2 | North-West | Vitrified | Yes | No |
| 11 | 2 | 2 | 0.00 | Semifurnished | Family Only | On Request, Bibwewadi Kondhwa Road, , Pune, Maharashtra | 3 | South-East | Not provided | Yes | No |
| 12 | 2 | 2 | 1000.00 | Unfurnished | All | Supriya Gardens, Aundh, , Pune, Maharashtra | 2 | North-East | Mosaic | Yes | Yes |
| 13 | 3 | 3 | 1500.00 | Semifurnished | All | Erande Arora Residency, Kharadi, , Pune, Maharashtra | 1 | East | Not provided | Yes | No |
| 14 | 2 | 2 | 0.00 | Unfurnished | All | SBM West Wind Park, Hinjewadi, , Pune, Maharashtra | 2 | No Direction | Not provided | No | No |
| 15 | 1 | 1 | 0.00 | Unfurnished | Family Only | On Request, Bibwewadi, , Pune, Maharashtra | 0 | West | Not provided | No | No |
| 16 | 1 | 1 | 0.00 | Unfurnished | Family Only | Sai vilasita Apartment wakad, Bhumkar Nagar, , Pune, Mahar... | 1 | No Direction | Not provided | Yes | No |
| 17 | 1 | 1 | 0.00 | Unfurnished | All | maurya housing society, Pimpri Chinchwad, , Pune, Maharas... | 1 | No Direction | Not provided | Yes | No |
| 18 | 1 | 1 | 445.00 | Semifurnished | All | Puraniks Aldea, Baner, , Pune, Maharashtra | 1 | East | Vitrified | Yes | No |
| 19 | 2 | 2 | 836.00 | Unfurnished | All | Anita Residency, Katraj Kondhwa Road, , Pune, Maharashtra | 8 | North-East | Not provided | Yes | Yes |
| 20 | 1 | 2 | 590.00 | Semifurnished | Family Only | Geeta Golden Palms, Bhukum, , Pune, Maharashtra | 6 | No Direction | Ceramic | Yes | No |
| 21 | 3 | 3 | 1350.00 | Semifurnished | Family Only | Paranjape Blue Ridge, Hinjewadi, , Pune, Maharashtra | 1 | No Direction | Vitrified | Yes | No |

This data frame contains the following columns:

**Bedroom**

count of  bedrooms

**Bathrooms**

count of  bathrooms

**Area**

area of the property in sq.feet

**Furnishing**

furnishing status of the house ( Furnished, Semifurnished, Unfurnished)

**Available_ for**

to whom the property is available for:- family, men, female, bachelors

**Address**

destination of location

**Floor_ number**

floor number of apartment

**Facing**

direction of door (this is due to India has spiritual background & facing of our house is matter)

**Floor_ type**

types of floor material used ex. marble etc.

**Gate_ community**

date security available or not (Yes, No)

**Corner_ pro**

corner property column indicate is property belongs to corner location (Yes, No)

**Parking**

how much vehicles can park

**Wheelchairadption**

wheel chair adaption facility available or not (Yes, None)

**Petfacility**

pet facility available or not (Yes, None)

**AggDur**

agreement duration in month/year

**NoticeDur**

duration of notice period in month/year

**Lightbill**

bill separated for included or not (1 – Yes, 0 – No)

**Powerbackup**

power backup facility count

**Propertyage**

age of property in years

**No_room**

availability of no rooms(1 – Yes, 0 – No)

**Pooja_room**

separate room for pooja(1 – Yes, 0 – No)

**Study_room**

availability of study room(1 – Yes, 0 - No)

**Others**

number of other rooms (1 – Yes, 0 – No)

**Servant_room**

separate room for servants(1 – Yes, 0 – No)

**Store_room**

separate room to store items(1 – Yes, 0 – No)

**Maintenance_amt**

charges for maintenance

**Brok_amt**

brokerage charges amount

**Deposit_amt**

deposit amount for house

**Mnt_amt**

maintanence charges

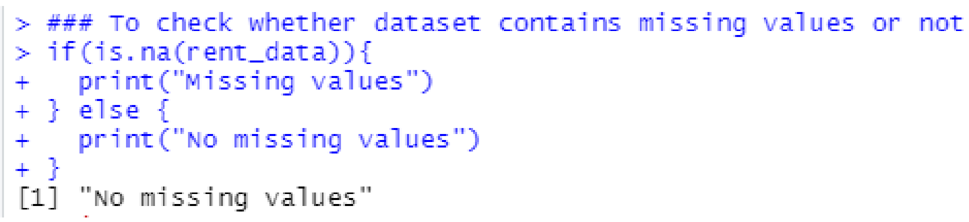
**Rent**

rent of the house in INR

The **“Rent”** is the response variable. Other above variables are predictor variables.

### 2.2 Data understanding

After loading the data, it’s a good practice to see if there are any missing values in the data.

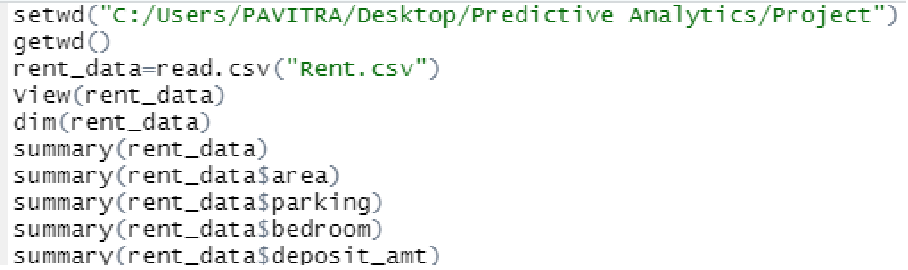


```
> ### To check whether dataset contains missing values or not
> if(is.na(rent_data)){
+   print("Missing values")
+ } else {
+   print("No missing values")
+ }
[1] "No missing values"
```

The above output shows that the dataset has no missing value.

This module explains data understanding. This dataset consist of different columns. Each and every columns we should find the summary() function. This function is used to calculate the average value and determine the maximum, minimum of the column in a dataframe.

The following code has been executed in R studio to read the entire dataset named rent.csv from the working directory .



```
setwd("C:/Users/PAVITRA/Desktop/Predictive Analytics/Project")
getwd()
rent_data=read.csv("Rent.csv")
View(rent_data)
dim(rent_data)
summary(rent_data)
summary(rent_data$area)
summary(rent_data$parking)
summary(rent_data$bedroom)
summary(rent_data$deposit_amt)
```

**BEDROOM**

It is a numeric variable. It gives number of bedroom in a house. It is a room situated within a residential or accommodation unit characterized by it usage for sleeping. Except in bungalows, ranch style homes, bedrooms are usually on one of the floors of a dwelling that is above ground level.

**Mean(Average)**

mean(rent_data$bedroom,na.rm=FALSE)

The average value of bedroom is 1.797868

**Max(Highest)**

max(rent_data$bedroom,na.rm=FALSE)

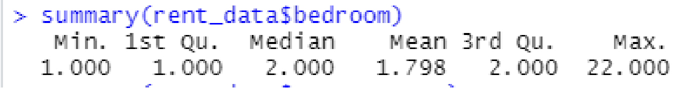
The Maximum value of crime rate is 22.

**Min(lowest)**

Min(rent_data$bedroom,na.rm=FALSE)

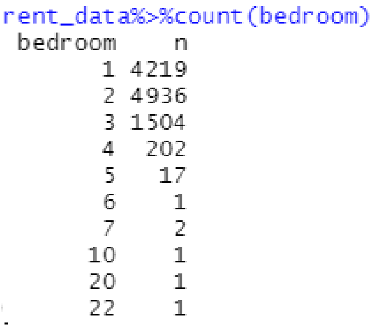
The minimum value of bathrooms is 1 .

**Summary(rent_data$bedroom)**



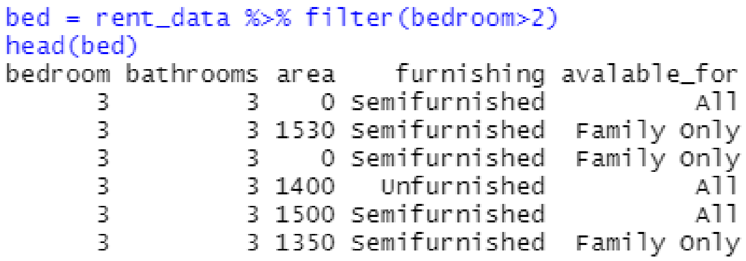
```
> summary(rent_data$bedroom)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  1.000   1.000   2.000   1.798   2.000  22.000 
```

By using plyr package, execute the count() command to know how many observations drop in this range.



```
rent_data%>%count(bedroom)
 bedroom    n
       1 4219
       2 4936
       3 1504
       4  202
       5   17
       6    1
       7    2
      10    1
      20    1
      22    1
```

The below code presents the result of filtering command applied on the variable bedroom. Greater than or equal to condition is used for this data to collect the record. Totally 10861 records are observed while the bedroom is greater than 2 .

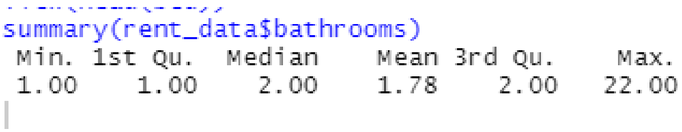


```
bed = rent_data %>% filter(bedroom>2)
head(bed)
bedroom bathrooms area    furnishing avalable_for
      3         3    0 Semifurnished          All
      3         3 1530 Semifurnished  Family Only
      3         3    0 Semifurnished  Family Only
      3         3 1400   Unfurnished          All
      3         3 1500 Semifurnished          All
      3         3 1350 Semifurnished  Family Only
```

**BATHROOM**

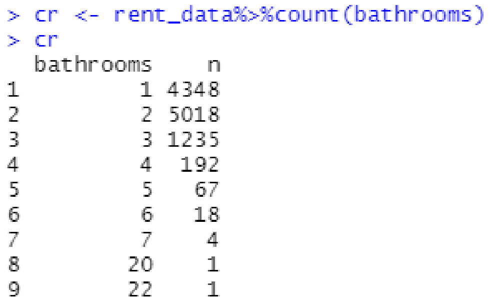
It is a numeric variable. It gives the number of bathrooms in a house. It is a room, typically in a home or other residential building, that contains either a bathtub or a shower (or both). In India, a toilet is typically included in the bathroom; in others, the toilet is a typically given a dedicated room separate from the one allocated for personal hygiene.

**Summary(rent_data$bathrooms)**



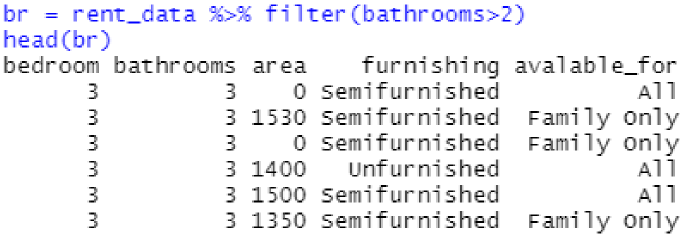
```
summary(rent_data$bathrooms)
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 1.00    1.00    2.00    1.78    2.00   22.00 
```

From the above output, it has been cleared that the Average bathrooms value is 1.78. The maximum value is 22.00.The minimum value is 1.00. The below R code explains the range of the column frequency for using count() function



```
> cr <- rent_data%>%count(bathrooms)
> cr
  bathrooms    n
1         1 4348
2         2 5018
3         3 1235
4         4  192
5         5   67
6         6   18
7         7    4
8        20    1
9        22    1
```

The below code presents the result of filtering command applied on the variable bathroom. Greater than  2 is used for this data to collect the record. Totally 1518 records are observed while the bathrooms  is greater than 2.

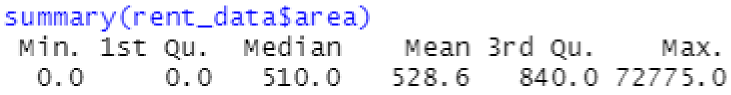


```
br  = rent_data %>% filter(bathrooms>2)
head(br)
bedroom bathrooms area    furnishing avalable_for
      3         3    0 Semifurnished          All
      3         3 1530 Semifurnished  Family Only
      3         3    0 Semifurnished  Family Only
      3         3 1400   Unfurnished          All
      3         3 1500 Semifurnished          All
      3         3 1350 Semifurnished  Family Only
```

### AREA

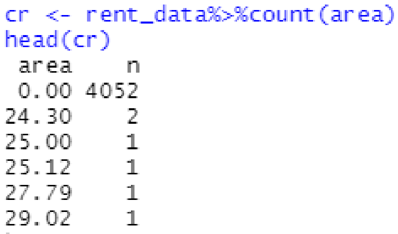
It is a continuous variable. It gives the area of a house in terms of square feet. A land is bifurcated into residential plots after making the necessary provision for roads, parks, schools, hospitals, markets and other amenities. In case of a residential property, the area is usually given in the form of Square Feet (Sq.ft).

**Summary(rent_data$area)**



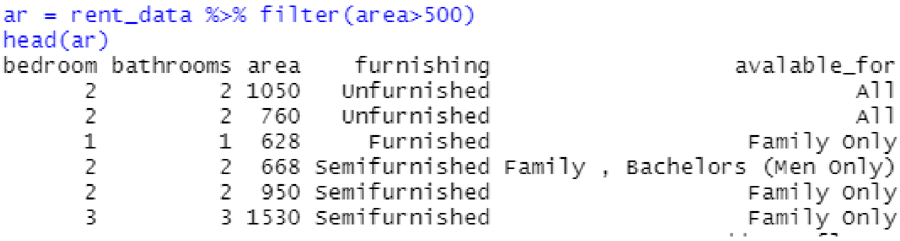
```
summary(rent_data$area)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   0.0     0.0   510.0   528.6   840.0 72775.0 
```

From the above output, it has been cleared that the Average area value is 528.6. The maximum value is 72775.0. The minimum value is 0.0. The below R code explains the range of the column frequency  for using  count() function



```
cr <- rent_data%>%count(area)
head(cr)
  area    n
  0.00 4052
 24.30    2
 25.00    1
 25.12    1
 27.79    1
 29.02    1
```

The below code presents the result of filtering command applied on the variable area. Greater than 500  is used for this data to collect the record. Totally 5462 records are observed while the area  is greater than 500.

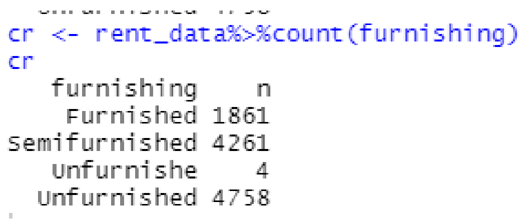


```
ar  = rent_data %>% filter(area>500)
head(ar)
bedroom bathrooms area    furnishing                   avalable_for
      2         2 1050   Unfurnished                            All
      2         2  760   Unfurnished                            All
      1         1  628     Furnished                    Family Only
      2         2  668 Semifurnished Family , Bachelors (Men Only)
      2         2  950 Semifurnished                    Family Only
      3         3 1530 Semifurnished                    Family Only
```

## FURNISHING

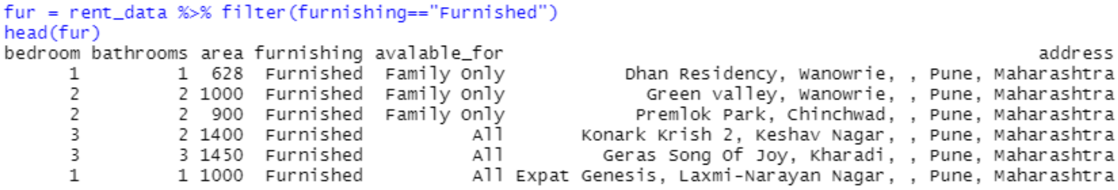
It is categorical variable. The whole dataset categorized into the values are Furnished, Unfurnished and Semi furnished. The furnishing of a room or house are the furniture, curtains, carpets and decorations such as pictures. The definitions are usually in plural, the instrumentalities (furniture and appliances and other movable accessories) that make a home livable.

The below R code explains the range of the column frequency for using count() function



```
cr <- rent_data%>%count(furnishing)
cr
   furnishing    n
    Furnished 1861
Semifurnished 4261
   Unfurnishe    4
  Unfurnished 4758
```

The below code presents the result of filtering command applied on the variable furnishing. The house which is furnished used for this data to collect the record. Totally 1861 records are observed while the furnishing status is furnished.

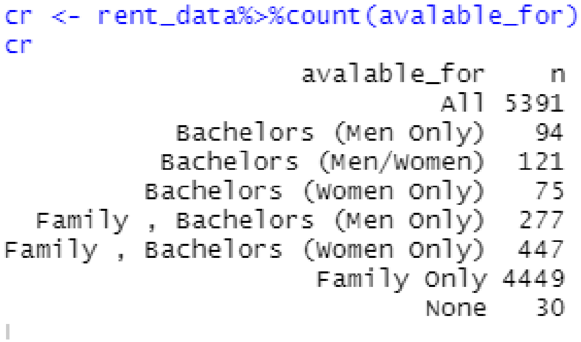


```
fur = rent_data %>% filter(furnishing=="Furnished")
head(fur)
bedroom bathrooms area furnishing avalable_for                                        address
      1         1  628  Furnished  Family Only          Dhan Residency, Wanowrie, , Pune, Maharashtra
      2         2 1000  Furnished  Family Only             Green valley, Wanowrie, , Pune, Maharashtra
      2         2  900  Furnished  Family Only            Premlok Park, Chinchwad, , Pune, Maharashtra
      3         2 1400  Furnished          All      Konark Krish 2, Keshav Nagar, , Pune, Maharashtra
      3         3 1450  Furnished          All        Geras Song Of Joy, Kharadi, , Pune, Maharashtra
      1         1 1000  Furnished          All Expat Genesis, Laxmi-Narayan Nagar, , Pune, Maharashtra
```

## AVALABLE FOR

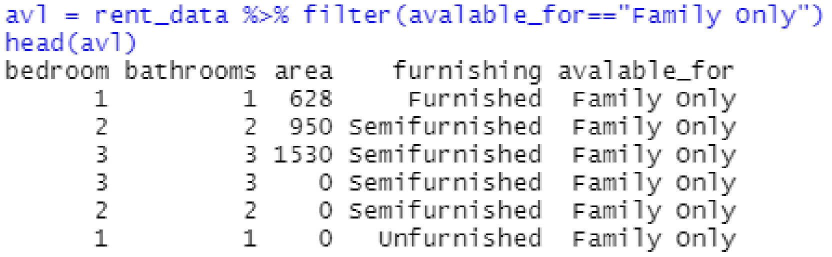
It is categorical variable. The whole dataset categorized into the values are Family Only, Family , Bachelors (Men Only), Family , Bachelors (Women Only), Bachelors (Men/Women), Bachelors (Women Only), Bachelors (Men Only), None. It depends on the owner's personal preference.

The below R code explains the range of the column frequency for using count() function



```
cr <- rent_data%>%count(avalable_for)
cr
                   avalable_for    n
                            All 5391
           Bachelors (Men Only)   94
          Bachelors (Men/Women)  121
         Bachelors (Women Only)   75
   Family , Bachelors (Men Only)  277
 Family , Bachelors (Women Only)  447
                    Family Only 4449
                           None   30
```

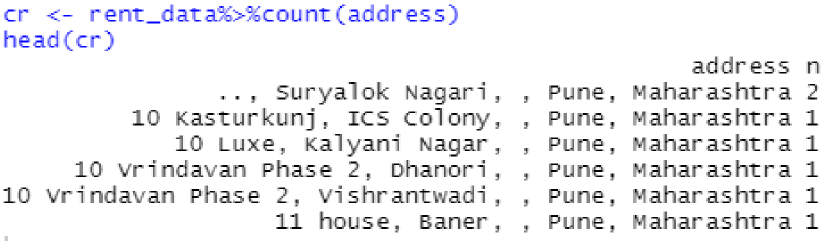
The below code presents the result of filtering command applied on the variable available for. The house which is family only used for this data to collect the record. Totally 4449 records are observed while the available for family only.



```
avl = rent_data %>% filter(avalable_for=="Family Only")
head(avl)
bedroom bathrooms area    furnishing avalable_for
      1         1  628     Furnished  Family Only
      2         2  950 Semifurnished  Family Only
      3         3 1530 Semifurnished  Family Only
      3         3    0 Semifurnished  Family Only
      2         2    0 Semifurnished  Family Only
      1         1    0   Unfurnished  Family Only
```

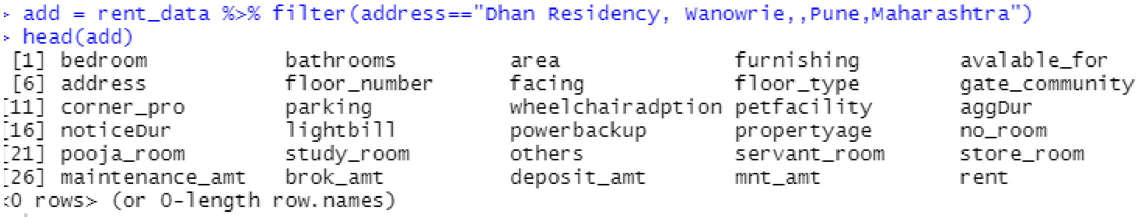
## ADDRESS

It is a categorical variable. It shows the destination address, city and of the house. The dataset has address in Pune city in Maharashtra State.



```
cr <- rent_data%>%count(address)
head(cr)
                                                 address n
                    .., Suryalok Nagari, , Pune, Maharashtra 2
            10 Kasturkunj, ICS Colony, , Pune, Maharashtra 1
               10 Luxe, Kalyani Nagar, , Pune, Maharashtra 1
       10 Vrindavan Phase 2, Dhanori, , Pune, Maharashtra 1
10 Vrindavan Phase 2, Vishrantwadi, , Pune, Maharashtra 1
                        11 house, Baner, , Pune, Maharashtra 1
```

The below code presents the result of filtering command applied on the variable address. The house which is in particular address used for this data to collect the record.



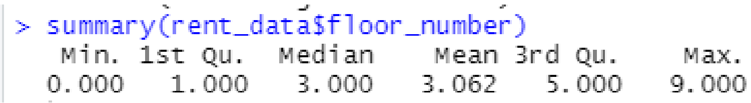
```
> add = rent_data %>% filter(address=="Dhan Residency, Wanowrie,,Pune,Maharashtra")
> head(add)
 [1] bedroom          bathrooms        area             furnishing       avalable_for
 [6] address          floor_number     facing           floor_type       gate_community
[11] corner_pro       parking          wheelchairadption petfacility     aggDur
[16] noticeDur        lightbill        powerbackup      propertyage      no_room
[21] pooja_room       study_room       others           servant_room     store_room
[26] maintenance_amt  brok_amt         deposit_amt      mnt_amt          rent
<0 rows> (or 0-length row.names)
```

The result tells that the particular address occurs at once.

## FLOOR NUMBER

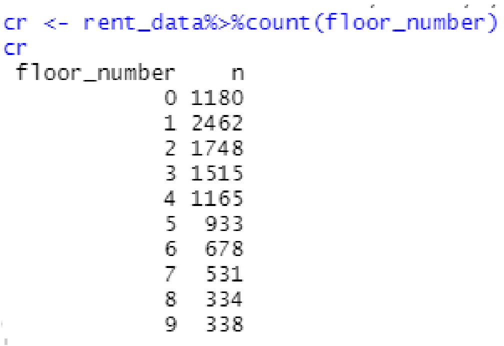
It is a numeric variable. It tells the floor number of the house in the apartment.

**Summary(rent_data$floor_number)**



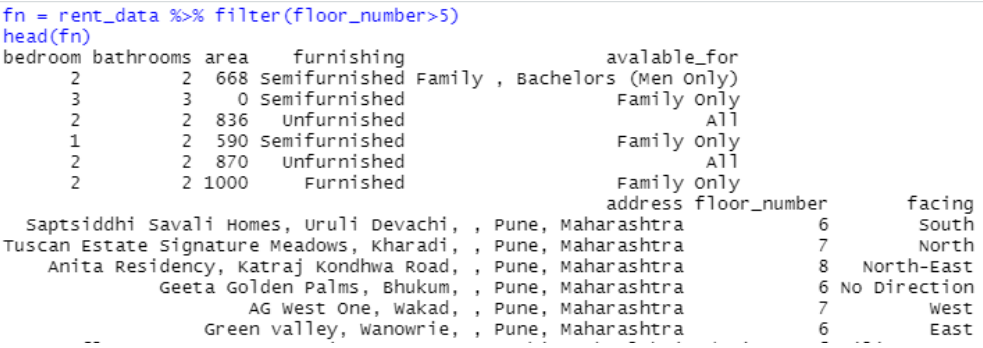
```
> summary(rent_data$floor_number)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   1.000   3.000   3.062   5.000   9.000
```

From the above output, it has been cleared that the Average floor number value is 3.062. The maximum value is 9.000. The minimum value is 0.000. The below R code explains the range of the column frequency for using count() function



```
cr <- rent_data%>%count(floor_number)
cr
 floor_number    n
            0 1180
            1 2462
            2 1748
            3 1515
            4 1165
            5  933
            6  678
            7  531
            8  334
            9  338
```

The below code presents the result of filtering command applied on the variable floor number. The house which is greater than 5 used for this data to collect the record. Totally 1881 records are observed while the floor number greater than 5.

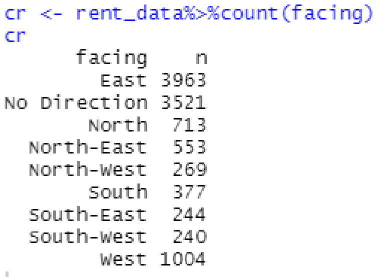


```
fn = rent_data %>% filter(floor_number>5)
head(fn)
bedroom bathrooms area    furnishing                   avalable_for
      2         2  668 Semifurnished Family , Bachelors (Men Only)
      3         3    0 Semifurnished                    Family Only
      2         2  836   Unfurnished                            All
      1         2  590 Semifurnished                    Family Only
      2         2  870   Unfurnished                            All
      2         2 1000     Furnished                    Family Only
                                                    address floor_number      facing
  Saptsiddhi Savali Homes, Uruli Devachi, , Pune, Maharashtra            6       South
Tuscan Estate Signature Meadows, Kharadi, , Pune, Maharashtra            7       North
    Anita Residency, Katraj Kondhwa Road, , Pune, Maharashtra            8  North-East
              Geeta Golden Palms, Bhukum, , Pune, Maharashtra            6 No Direction
                     AG West One, Wakad, , Pune, Maharashtra            7        West
                 Green valley, Wanowrie, , Pune, Maharashtra            6        East
```

## FACING

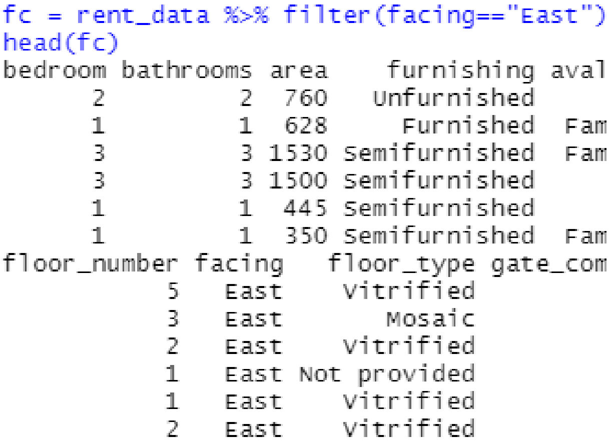
It is a categorical variable. The values are categorized into West, East, South-West, South, North, North-West, South-East, North-East and No Direction. It shows the direction of the house or is the direction you face, while coming out of the house. The direction of the main entrance as per Vastu, is the most important aspect, while taking a rental home.

The below R code explains the range of the column frequency  for using  count() function



```
cr <- rent_data%>%count(facing)
cr
      facing    n
        East 3963
No Direction 3521
       North  713
  North-East  553
  North-West  269
       South  377
  South-East  244
  South-West  240
        West 1004
```

The below code presents the result of filtering command applied on the variable facing. The house which having East direction used for this data to collect the record. Totally 3963 records are observed where the house facing is East.

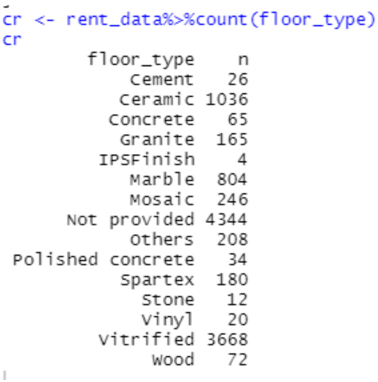


```
fc = rent_data %>% filter(facing=="East")
head(fc)
bedroom bathrooms area    furnishing aval
      2         2  760   Unfurnished
      1         1  628     Furnished  Fam
      3         3 1530 Semifurnished  Fam
      3         3 1500 Semifurnished
      1         1  445 Semifurnished
      1         1  350 Semifurnished  Fam
floor_number facing   floor_type gate_com
           5   East     Vitrified
           3   East        Mosaic
           2   East     Vitrified
           1   East Not provided
           1   East     Vitrified
           2   East     Vitrified
```

## FLOOR TYPE

It is a categorical variable. The values are categorized into Marble, Vitrified, Mosaic, Polished concrete, Ceramic, Wood, Stone, Spartex, Granite, Vinyl, Concrete, Cement, IPSFinish, Others and Not provided. It tells about the type of the floor of the house. It refers to the lower enclosing surface of spaces within buildings. Typically, it is a permanent covering laid over the floor.

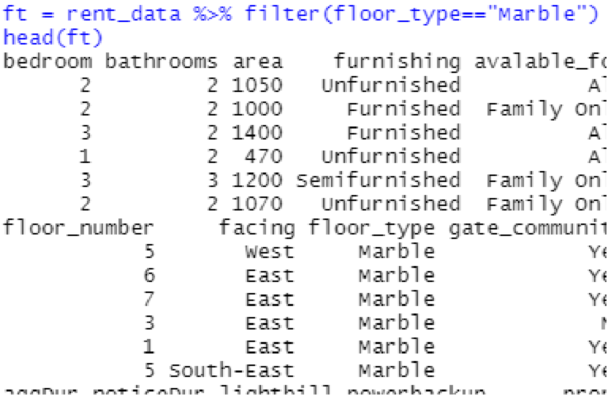
The below R code explains the range of the column frequency  for using  count() function



```
cr <- rent_data%>%count(floor_type)
cr
        floor_type    n
            Cement   26
           Ceramic 1036
          Concrete   65
           Granite  165
         IPSFinish    4
            Marble  804
            Mosaic  246
      Not provided 4344
            Others  208
 Polished concrete   34
           Spartex  180
             Stone   12
             Vinyl   20
         Vitrified 3668
              Wood   72
```

The below code presents the result of filtering command applied on the variable floor type.

The house which having floor type as Marble used for this data to collect the record. Totally 804 records are observed where the floor type is Marble.

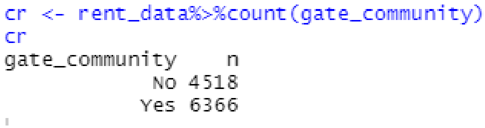


```
ft = rent_data %>% filter(floor_type=="Marble")
head(ft)
bedroom bathrooms area    furnishing avalable_f
      2         2 1050   Unfurnished          A
      2         2 1000     Furnished  Family On
      3         2 1400     Furnished          A
      1         2  470   Unfurnished          A
      3         3 1200 Semifurnished  Family On
      2         2 1070   Unfurnished  Family On
floor_number      facing floor_type gate_communit
           5        West     Marble            Y
           6        East     Marble            Y
           7        East     Marble            Y
           3        East     Marble             N
           1        East     Marble            Y
           5  South-East     Marble            Y
```

## GATE COMMUNITY

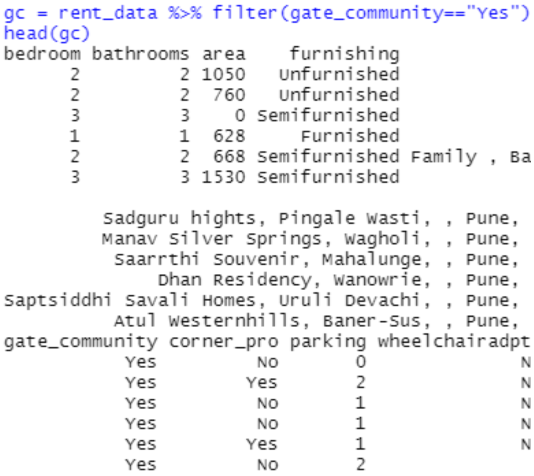
It is a categorical variable. The values are categorized into Yes and No. It is a residential community or housing property having a name and exact geographic demarcation as set apart by the boundaries and gates that exact control access to the area. One can spot gated communities in any location including cities, towns and even outskirts.

The below R code explains the range of the column frequency  for using  count() function



```
cr <- rent_data%>%count(gate_community)
cr
gate_community    n
            No 4518
           Yes 6366
```

The below code presents the result of filtering command applied on the variable gate community. The house which having gate community used for this data to collect the record. Totally 6366 records are observed which houses are having gate community value as yes.

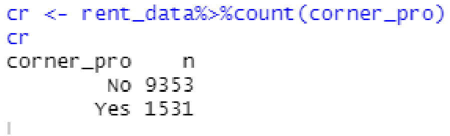


```
gc = rent_data %>% filter(gate_community=="Yes")
head(gc)
bedroom bathrooms area    furnishing
      2         2 1050   Unfurnished
      2         2  760   Unfurnished
      3         3    0 Semifurnished
      1         1  628     Furnished
      2         2  668 Semifurnished Family , Ba
      3         3 1530 Semifurnished

         Sadguru hights, Pingale Wasti, , Pune,
         Manav Silver Springs, Wagholi, , Pune,
          Saarrthi Souvenir, Mahalunge, , Pune,
              Dhan Residency, Wanowrie, , Pune,
Saptsiddhi Savali Homes, Uruli Devachi, , Pune,
          Atul Westernhills, Baner-Sus, , Pune,
gate_community corner_pro parking wheelchairadpt
           Yes         No       0              N
           Yes        Yes       2              N
           Yes         No       1              N
           Yes         No       1              N
           Yes        Yes       1              N
           Yes         No       2
```

**CORNER PRO**

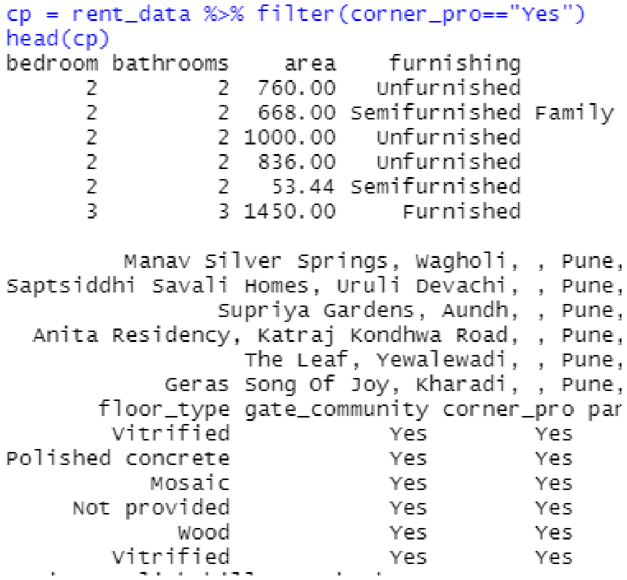
It is a categorical variable. The values are categorized into Yes and No. House on corner plots always offer a clear view of the street. While most house owners  have windows and balconies overlooking their own front yard or the neighbour’s lawn.

The below R code explains the range of the column frequency  for using  count() function



```
cr <- rent_data%>%count(corner_pro)
cr
corner_pro    n
        No 9353
       Yes 1531
```

The below code presents the result of filtering command applied on the variable corner property. The house which is corner property used for this data to collect the record. Totally 1531 records are observed which houses are having corner property value as yes.

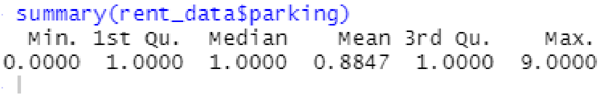


```
cp = rent_data %>% filter(corner_pro=="Yes")
head(cp)
bedroom bathrooms    area    furnishing
      2         2  760.00   Unfurnished
      2         2  668.00 Semifurnished Family
      2         2 1000.00   Unfurnished
      2         2  836.00   Unfurnished
      2         2   53.44 Semifurnished
      3         3 1450.00     Furnished

        Manav Silver Springs, Wagholi, , Pune,
Saptsiddhi Savali Homes, Uruli Devachi, , Pune,
                Supriya Gardens, Aundh, , Pune,
  Anita Residency, Katraj Kondhwa Road, , Pune,
                  The Leaf, Yewalewadi, , Pune,
             Geras Song Of Joy, Kharadi, , Pune,
      floor_type gate_community corner_pro par
       Vitrified            Yes        Yes
Polished concrete            Yes        Yes
          Mosaic            Yes        Yes
    Not provided            Yes        Yes
            Wood            Yes        Yes
       Vitrified            Yes        Yes
```

## PARKING

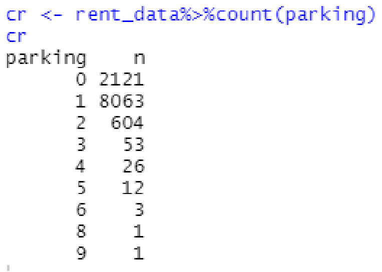
It is a numerical variable. It is the act of stopping a vehicle and leaving it unoccupied. Parking on one or both sides of a road is often permitted, though sometimes with restrictions. Some buildings have parking facilities for use of the buildings users. In apartments, parking facilities allocated for each house.

**Summary(rent_data$parking)**



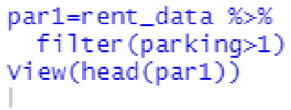
```
summary(rent_data$parking)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
0.0000  1.0000  1.0000  0.8847  1.0000  9.0000 
```

From the above output, it has been cleared that the Average area value is 0.8847. The maximum value is 9.000. The minimum value is 0.000. The below R code explains the range of the column frequency  for using  count() function

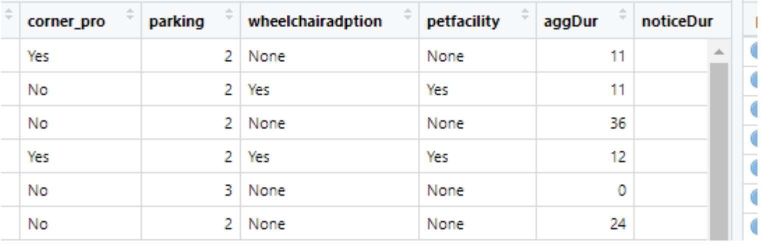


```
cr <- rent_data%>%count(parking)
cr
parking    n
      0 2121
      1 8063
      2  604
      3   53
      4   26
      5   12
      6    3
      8    1
      9    1
```

The below code presents the result of filtering command applied on the variable parking. The house which is greater than 1 used for this data to collect the record. Totally 700 records are observed which houses are greater than 1.



```
par1=rent_data %>%
  filter(parking>1)
View(head(par1))
```

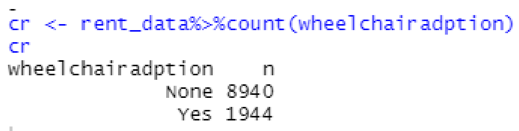


| corner_pro | parking | wheelchairadption | petfacility | aggDur | noticeDur |
|---|---|---|---|---|---|
| Yes | 2 | None | None | 11 | |
| No | 2 | Yes | Yes | 11 | |
| No | 2 | None | None | 36 | |
| Yes | 2 | Yes | Yes | 12 | |
| No | 3 | None | None | 0 | |
| No | 2 | None | None | 24 | |

## WHEELCHAIRADPTION

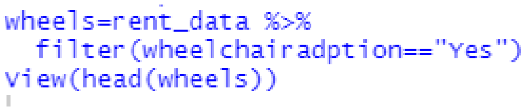
It is categorical variable. The values are categorized into Yes and None. It shows whether it has facility of wheel chair adaption or not. It is a facility a stairlift or a banister on the stairs, adding a bath lift, walk-in shower or a rail hold to pull person themselves out of bath widening doorways, lowering kitchen worktops.

The below R code explains the range of the column frequency  for using  count() function

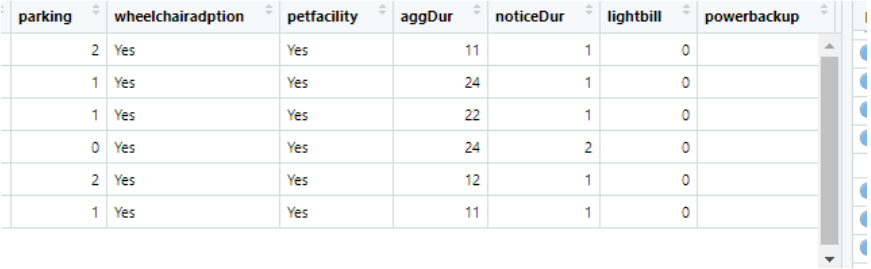


```
cr <- rent_data%>%count(wheelchairadption)
cr
wheelchairadption    n
              None 8940
               Yes 1944
```

The below code presents the result of filtering command applied on the variable wheel chair adaption. The house which having wheel chair adaption used for this data to collect the record. Totally 1944 records are observed which houses are having wheel chair adaption value as yes.



```
wheels=rent_data %>%
  filter(wheelchairadption=="Yes")
View(head(wheels))
```

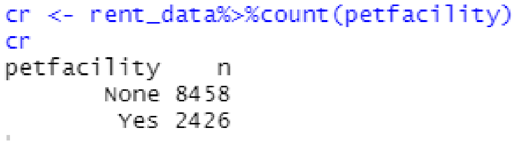


| parking | wheelchairadption | petfacility | aggDur | noticeDur | lightbill | powerbackup |
|---|---|---|---|---|---|---|
| 2 | Yes | Yes | 11 | 1 | 0 | |
| 1 | Yes | Yes | 24 | 1 | 0 | |
| 1 | Yes | Yes | 22 | 1 | 0 | |
| 0 | Yes | Yes | 24 | 2 | 0 | |
| 2 | Yes | Yes | 12 | 1 | 0 | |
| 1 | Yes | Yes | 11 | 1 | 0 | |

**PETFACILITY**

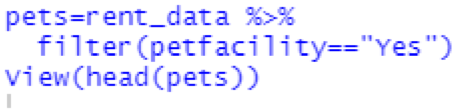
It is a categorical variable. The values are categorized into None and Yes. Some property owners who restrict pets at their properties are losing out on significant rental income in the form of shorter-term leases.

The below R code explains the range of the column frequency  for using  count() function

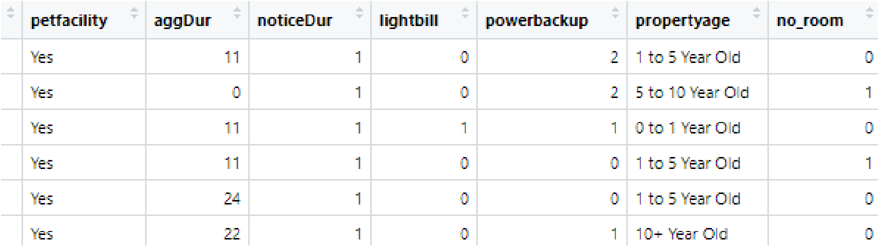


```
cr <- rent_data%>%count(petfacility)
cr
petfacility    n
       None 8458
        Yes 2426
```

The below code presents the result of filtering command applied on the variable pet facility. The house which having pet facility used for this data to collect the record. Totally 2426 records are observed which houses are having pet facility value as yes.



```
pets=rent_data %>%
  filter(petfacility=="Yes")
View(head(pets))
```

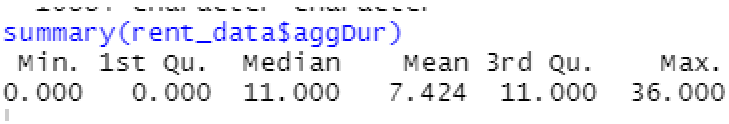


| petfacility | aggDur | noticeDur | lightbill | powerbackup | propertyage | no_room |
|---|---|---|---|---|---|---|
| Yes | 11 | 1 | 0 | 2 | 1 to 5 Year Old | 0 |
| Yes | 0 | 1 | 0 | 2 | 5 to 10 Year Old | 1 |
| Yes | 11 | 1 | 1 | 1 | 0 to 1 Year Old | 0 |
| Yes | 11 | 1 | 0 | 0 | 1 to 5 Year Old | 1 |
| Yes | 24 | 1 | 0 | 0 | 1 to 5 Year Old | 0 |
| Yes | 22 | 1 | 0 | 1 | 10+ Year Old | 0 |

## AGGDUR

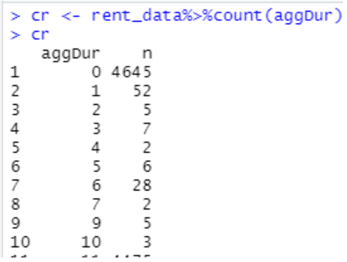
It is a numerical variable. It tells the duration of agreement in month. If the person have ever put a property on rent or have lived on a rented house, they must have signed a rent agreement. Most of the rent agreements are signed for 11 months so that they can avoid stamp duty and other charges.

**Summary(rent_data$aggDur)**



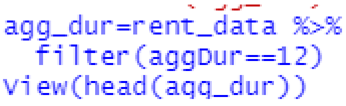
```
summary(rent_data$aggDur)
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
0.000   0.000  11.000   7.424  11.000  36.000 
```

From the above output, it has been cleared that the Average agreement duration value is 11.000. The maximum value is 36.000. The minimum value is 0.000. The below R code explains the range of the column frequency  for using  count() function

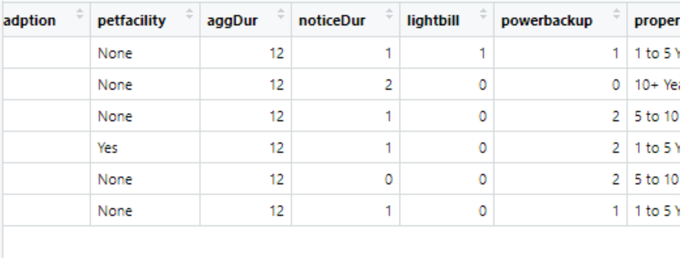


```
> cr <- rent_data%>%count(aggDur)
> cr
   aggDur    n
1       0 4645
2       1   52
3       2    5
4       3    7
5       4    2
6       5    6
7       6   28
8       7    2
9       9    5
10     10    3
```

The below code presents the result of filtering command applied on the variable agreement duration. The house which is equal to 12 used for this data to collect the record. Totally 852 records are observed which houses are equal to  12.



```
agg_dur=rent_data %>%
  filter(aggDur==12)
view(head(agg_dur))
```

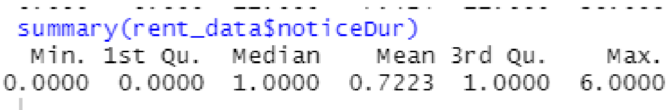


| adption | petfacility | aggDur | noticeDur | lightbill | powerbackup | proper |
|---|---|---|---|---|---|---|
| | None | 12 | 1 | 1 | 1 | 1 to 5 Y |
| | None | 12 | 2 | 0 | 0 | 10+ Ye |
| | None | 12 | 1 | 0 | 2 | 5 to 10 |
| | Yes | 12 | 1 | 0 | 2 | 1 to 5 Y |
| | None | 12 | 0 | 0 | 2 | 5 to 10 |
| | None | 12 | 1 | 0 | 1 | 1 to 5 Y |

## NOTICEDUR

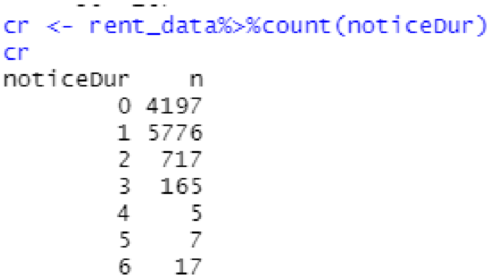
It is a numerical variable. It tells the duration of notice period in months. The notice period duration specifies in lock-in period. If the notice period is two months, person will have to give a two month notice to landlord in case of plan to vacate the house. The notice period is typically not valid during the lock-in period for either party.

**Summary(rent_data$aggDur)**



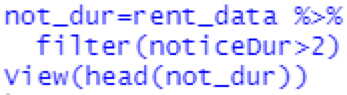
```
summary(rent_data$noticeDur)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
0.0000  0.0000  1.0000  0.7223  1.0000  6.0000 
```

From the above output, it has been cleared that the Average notice period duration value is 0.7223. The maximum value is 6.000. The minimum value is 0.000. The below R code explains the range of the column frequency  for using  count() function

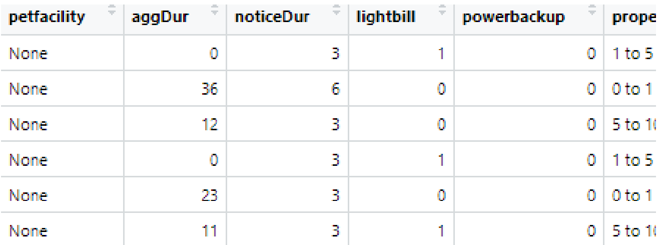


```
cr <- rent_data%>%count(noticeDur)
cr
noticeDur    n
        0 4197
        1 5776
        2  717
        3  165
        4    5
        5    7
        6   17
```

The below code presents the result of filtering command applied on the variable notice period duration. The house which is greater than 2 used for this data to collect the record. Totally 194 records are observed which houses are greater than 2.



```
not_dur=rent_data %>%
  filter(noticeDur>2)
view(head(not_dur))
```

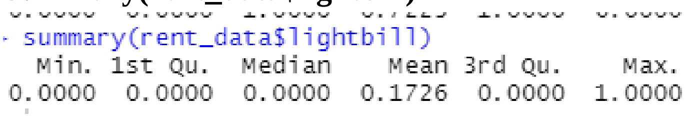


| petfacility | aggDur | noticeDur | lightbill | powerbackup | prope |
|---|---|---|---|---|---|
| None | 0 | 3 | 1 | 0 | 1 to 5 |
| None | 36 | 6 | 0 | 0 | 0 to 1 |
| None | 12 | 3 | 0 | 0 | 5 to 1 |
| None | 0 | 3 | 1 | 0 | 1 to 5 |
| None | 23 | 3 | 0 | 0 | 0 to 1 |
| None | 11 | 3 | 1 | 0 | 5 to 1 |

## LIGHTBILL

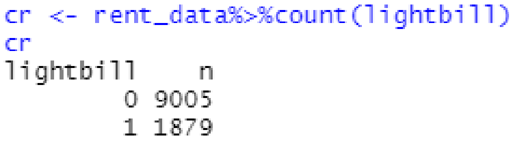
It is a numerical variable. The values fall into either 0 or 1. It tells about whether the electricity bill was included or not.  It is a bill for money owed for electricity used. The bill that a local utility issues to a consumer for the electricity that their home consumes. Anytime people turn on a light, plug in a device, or let our refrigerator run, people are utilizing electricity that has been generated for their use by a company.

**Summary(rent_data$lightbill)**



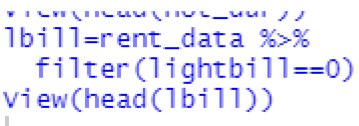
```
summary(rent_data$lightbill)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
0.0000  0.0000  0.0000  0.1726  0.0000  1.0000 
```

From the above output, it has been cleared that the Average light value is 0.1726. The maximum value is 1.000. The minimum value is 0.000. The below R code explains the range of the column frequency  for using  count() function

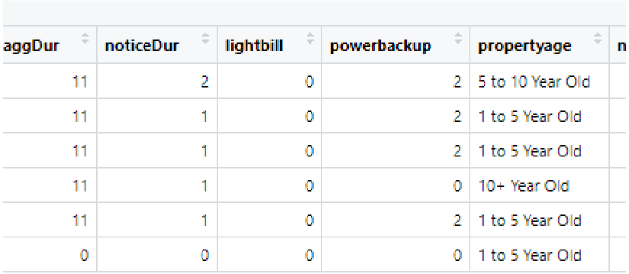


```
cr <- rent_data%>%count(lightbill)
cr
lightbill     n
        0  9005
        1  1879
```

The below code presents the result of filtering command applied on the variable lightbill. The house which having no light bill used for this data to collect the record. Totally 194 records are observed which houses are light bill value as 0.



```
lbill=rent_data %>%
  filter(lightbill==0)
view(head(lbill))
```



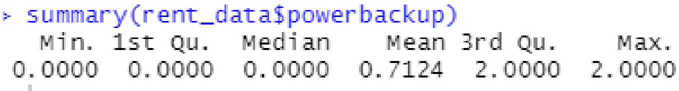
| aggDur | noticeDur | lightbill | powerbackup | propertyage | n |
|---|---|---|---|---|---|
| 11 | 2 | 0 | 2 | 5 to 10 Year Old | |
| 11 | 1 | 0 | 2 | 1 to 5 Year Old | |
| 11 | 1 | 0 | 2 | 1 to 5 Year Old | |
| 11 | 1 | 0 | 0 | 10+ Year Old | |
| 11 | 1 | 0 | 2 | 1 to 5 Year Old | |
| 0 | 0 | 0 | 0 | 1 to 5 Year Old | |

## POWERBACKUP

It is a numerical variable. The values fall into either 0 or 1. It is used to provide energy when the primary source fails. This is very important since an uninterruptible power supply is crucial for any operation. The current backup systems include batteries, and generators which operate on diesel, propel or gasoline. One of the most  power backup options to install in an apartment society is the fixed generator through which AC loads, electrical appliances and essential lights can work.

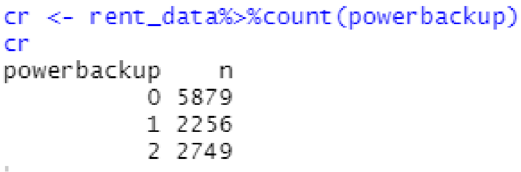
For example, If the apartment has 3 bedrooms, a 5000 watt generator would be a perfect power backup option.

**Summary(rent_data$powerbackup)**



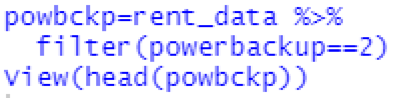
```
> summary(rent_data$powerbackup)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.0000  0.0000  0.0000  0.7124  2.0000  2.0000 
```

From the above output, it has been cleared that the Average power backup value is 0.7124. The maximum value is 2.000. The minimum value is 0.000. The below R code explains the range of the column frequency  for using  count() function

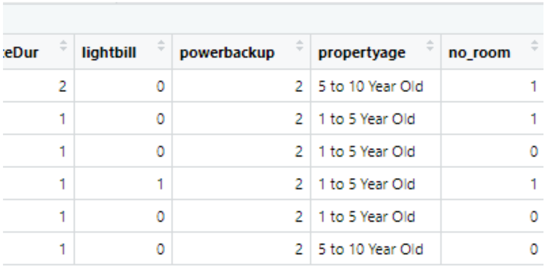


```
cr <- rent_data%>%count(powerbackup)
cr
powerbackup     n
          0  5879
          1  2256
          2  2749
```

The below code presents the result of filtering command applied on the variable power backup. The house which having equal to 2 used for this data to collect the record. Totally 2749 records are observed which houses are equal to 2.



```
powbckp=rent_data %>%
  filter(powerbackup==2)
view(head(powbckp))
```



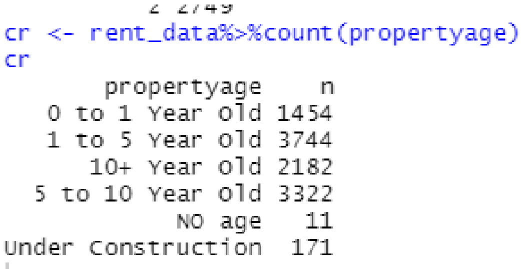
| eDur | lightbill | powerbackup | propertyage | no_room |
|---|---|---|---|---|
| 2 | 0 | 2 | 5 to 10 Year Old | 1 |
| 1 | 0 | 2 | 1 to 5 Year Old | 1 |
| 1 | 0 | 2 | 1 to 5 Year Old | 0 |
| 1 | 1 | 2 | 1 to 5 Year Old | 1 |
| 1 | 0 | 2 | 1 to 5 Year Old | 0 |
| 1 | 0 | 2 | 5 to 10 Year Old | 0 |

## PROPERTY AGE

It is a categorical variable. It tells the age of a property in distribution of years. It means a lot in the rent. If a property was sold by the developer who built it though, could find out its approximate age using the date of the first transfer or lease by the developer, as this date is often referred to in the register. If a property was not sold by its developer who built it, won't have any information about its age.

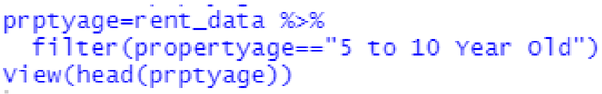
For example, The property was constructed about 14 years ago. However, the land area for the property 27000 square feet, which is very huge. The rate of the land would be close to Rs.40,000 per square feet and the property cost would be anywhere between Rs 1.3 crore and Rs. 1.4 crore.

The below R code explains the range of the column frequency  for using  count() function

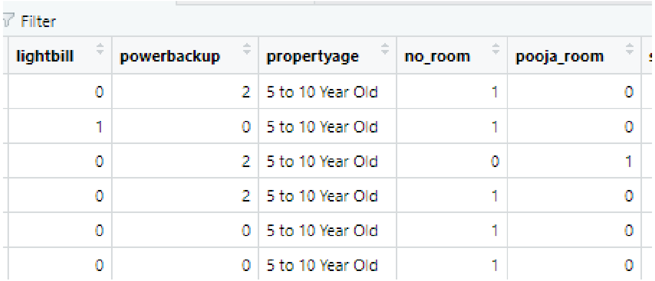


```
             2 2749
cr <- rent_data%>%count(propertyage)
cr
       propertyage    n
    0 to 1 Year old 1454
    1 to 5 Year old 3744
       10+ Year old 2182
   5 to 10 Year old 3322
             NO age   11
Under Construction  171
```

The below code presents the result of filtering command applied on the variable property age. The house which having property age is 5 to 10 year old used for this data to collect the record. Totally 3322 records are observed which houses are having pet facility value as yes.



```
prptyage=rent_data %>%
  filter(propertyage=="5 to 10 Year old")
view(head(prptyage))
```



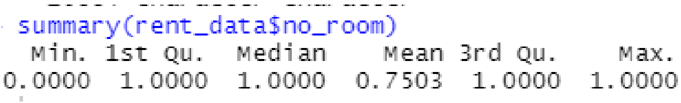
| lightbill | powerbackup | propertyage | no_room | pooja_room |
|---|---|---|---|---|
| 0 | 2 | 5 to 10 Year Old | 1 | 0 |
| 1 | 0 | 5 to 10 Year Old | 1 | 0 |
| 0 | 2 | 5 to 10 Year Old | 0 | 1 |
| 0 | 2 | 5 to 10 Year Old | 1 | 0 |
| 0 | 0 | 5 to 10 Year Old | 1 | 0 |
| 0 | 0 | 5 to 10 Year Old | 1 | 0 |

## NO_ROOM

It is a numerical variable. The values fall into either 0 or 1. It tells whether any extra space available for extra room or not, unsuitable or unacceptance.

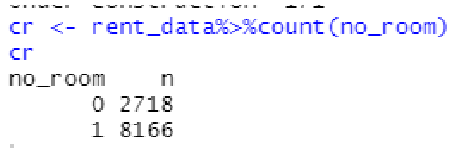
For example, if a person is a musician, he/ she want to extra room for practice, composing music.

**Summary(rent_data$no_room)**



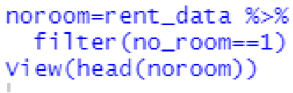
```
summary(rent_data$no_room)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.0000  1.0000  1.0000  0.7503  1.0000  1.0000
```

From the above output, it has been cleared that the Average no room value is 0.7503. The maximum value is 1.000. The minimum value is 0.000. The below R code explains the range of the column frequency  for using  count() function

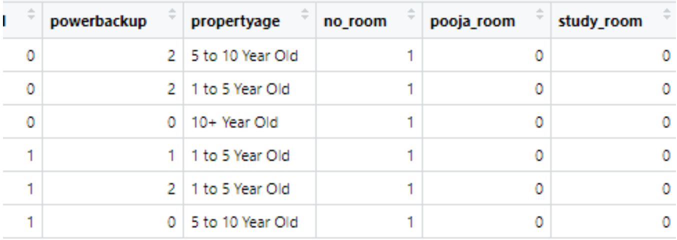


```
cr <- rent_data%>%count(no_room)
cr
no_room    n
      0 2718
      1 8166
```

The below code presents the result of filtering command applied on the variable no room. The house which having no room used for this data to collect the record. Totally 8166 records are observed which houses are having no room value as 1.



```
noroom=rent_data %>%
  filter(no_room==1)
view(head(noroom))
```

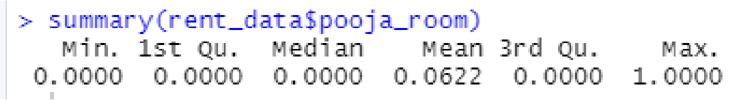


| l | powerbackup | propertyage | no_room | pooja_room | study_room |
|---|---|---|---|---|---|
| 0 | 2 | 5 to 10 Year Old | 1 | 0 | 0 |
| 0 | 2 | 1 to 5 Year Old | 1 | 0 | 0 |
| 0 | 0 | 10+ Year Old | 1 | 0 | 0 |
| 1 | 1 | 1 to 5 Year Old | 1 | 0 | 0 |
| 1 | 2 | 1 to 5 Year Old | 1 | 0 | 0 |
| 1 | 0 | 5 to 10 Year Old | 1 | 0 | 0 |

## POOJA_ROOM

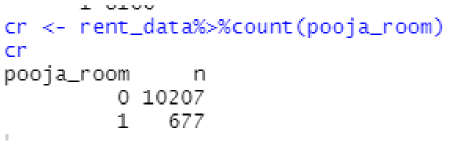
It is a numerical variable. The values fall into either 0 or 1. It is one of the most important aspects of any Indian household. The pooja room, also known as the prayer room is a scared space in most Indian homes. It is a customized space dedicated to conducting spiritual activities in daily prayers, poojas etc. to worship God.

**Summary(rent_data$pooja_room)**



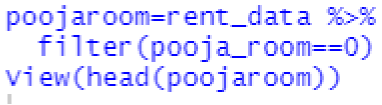
```
> summary(rent_data$pooja_room)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.0000  0.0000  0.0000  0.0622  0.0000  1.0000 
```

From the above output, it has been cleared that the Average no room value is 0.0622. The maximum value is 1.000. The minimum value is 0.000. The below R code explains the range of the column frequency  for using  count() function

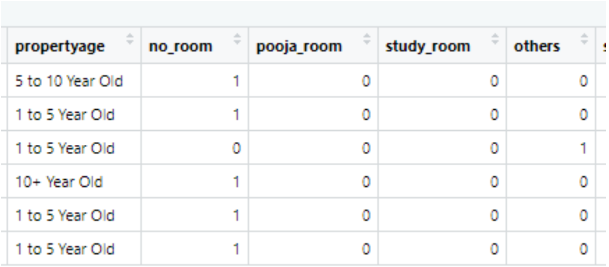


```
cr <- rent_data%>%count(pooja_room)
cr
pooja_room     n
         0 10207
         1   677
```

The below code presents the result of filtering command applied on the variable pooja room. The house which having 0 no room used for this data to collect the record. Totally 10207 records are observed which houses are having pooja room value as 0.



```
poojaroom=rent_data %>%
  filter(pooja_room==0)
view(head(poojaroom))
```

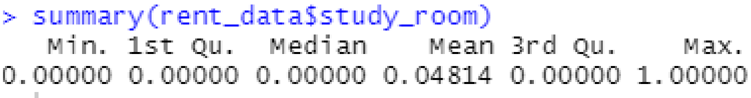


| propertyage | no_room | pooja_room | study_room | others |
|---|---|---|---|---|
| 5 to 10 Year Old | 1 | 0 | 0 | 0 |
| 1 to 5 Year Old | 1 | 0 | 0 | 0 |
| 1 to 5 Year Old | 0 | 0 | 0 | 1 |
| 10+ Year Old | 1 | 0 | 0 | 0 |
| 1 to 5 Year Old | 1 | 0 | 0 | 0 |
| 1 to 5 Year Old | 1 | 0 | 0 | 0 |

## STUDY_ROOM

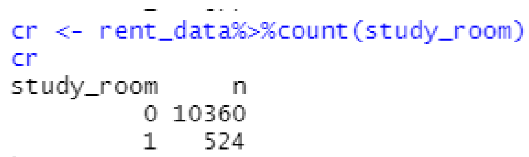
It is a numerical variable. The values fall into either 0 or 1. It is used for paperwork, computer work, or reading. It is reserved for use as the private office and reading room of a family father as the formal head of the household, but today studies are generally either used to operate a home business or else open to the whole family.

**Summary(rent_data$study_room)**



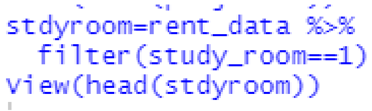
```
> summary(rent_data$study_room)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
0.00000 0.00000 0.00000 0.04814 0.00000 1.00000 
```

From the above output, it has been cleared that the Average study room value is 0.04814. The maximum value is 1.000. The minimum value is 0.000. The below R code explains the range of the column frequency  for using  count() function

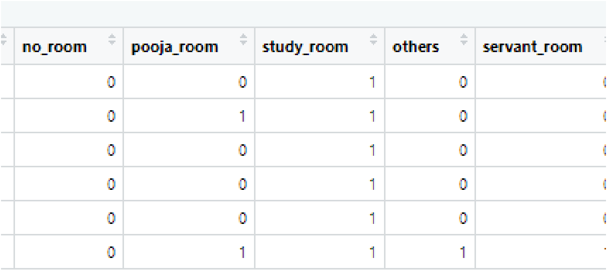


```
cr <- rent_data%>%count(study_room)
cr
study_room     n
         0 10360
         1   524
```

The below code presents the result of filtering command applied on the variable study room. The house which having study  room used for this data to collect the record. Totally 524 records are observed which houses are having study room value as 1.



```
stdyroom=rent_data %>%
  filter(study_room==1)
view(head(stdyroom))
```

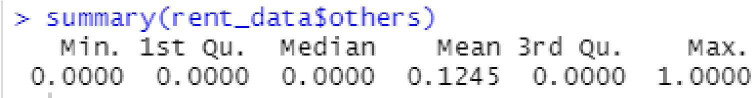


| no_room | pooja_room | study_room | others | servant_room |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | |
| 0 | 1 | 1 | 0 | |
| 0 | 0 | 1 | 0 | |
| 0 | 0 | 1 | 0 | |
| 0 | 0 | 1 | 0 | |
| 0 | 1 | 1 | 1 | |

**OTHERS**

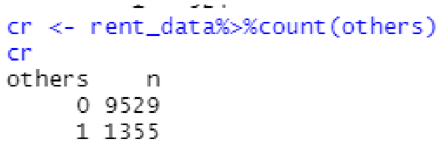
It is a numerical variable. The values fall into either 0 or 1. It tells whether any other room facility available or not except the mention room in the dataset. Other rooms may like club house, drawing room, dining room, laundry room etc.

**Summary(rent_data$others)**



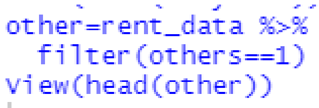
```
> summary(rent_data$others)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.0000  0.0000  0.0000  0.1245  0.0000  1.0000 
```

From the above output, it has been cleared that the Average others value is 0.1245. The maximum value is 1.000. The minimum value is 0.000. The below R code explains the range of the column frequency  for using  count() function

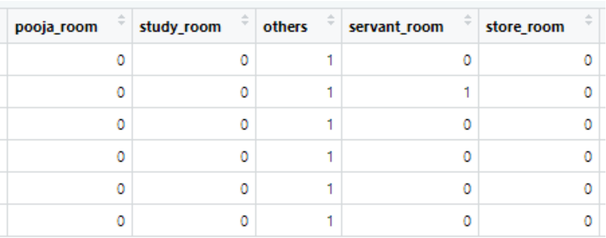


```
cr <- rent_data%>%count(others)
cr
others    n
     0 9529
     1 1355
```

The below code presents the result of filtering command applied on the variable others. The house which having other room used for this data to collect the record. Totally 524 records are observed which houses are having others value as 1.



```
other=rent_data %>%
  filter(others==1)
view(head(other))
```

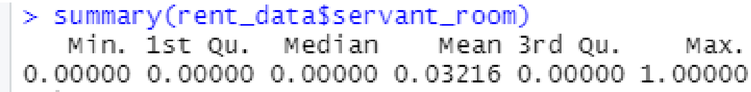


| pooja_room | study_room | others | servant_room | store_room |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |

## SERVANT_ROOM

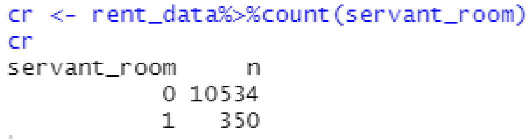
It is a numerical variable. The values fall into either 0 or 1. It is part of a building, traditionally in a private house, which contain the domestic offices and staff accommodation. From the late 17th century until the early 20th Century, they were a common feature in many large houses.

**Summary(rent_data$servant_room)**



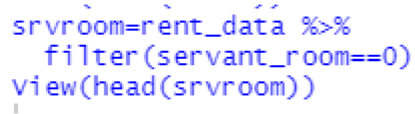
```
> summary(rent_data$servant_room)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
0.00000 0.00000 0.00000 0.03216 0.00000 1.00000 
```

From the above output, it has been cleared that the Average others value is 0.3216. The maximum value is 1.000. The minimum value is 0.000. The below R code explains the range of the column frequency for using count() function

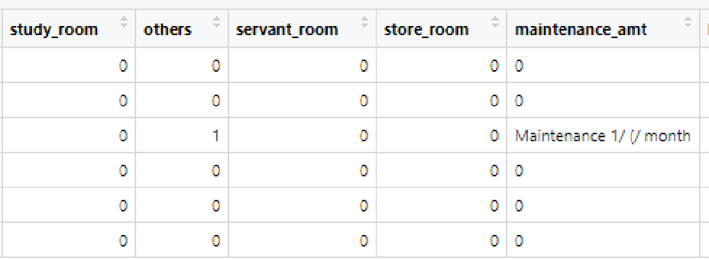


```
cr <- rent_data%>%count(servant_room)
cr
servant_room     n
           0 10534
           1   350
```

The below code presents the result of filtering command applied on the variable servant room. The house which having no servant room used for this data to collect the record. Totally 10534 records are observed which houses are having servant room value as 0.



```
srvroom=rent_data %>%
  filter(servant_room==0)
view(head(srvroom))
```

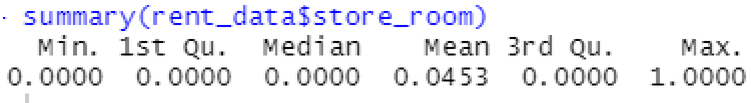


| study_room | others | servant_room | store_room | maintenance_amt |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | Maintenance 1/ (/ month |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |

## STORE_ROOM

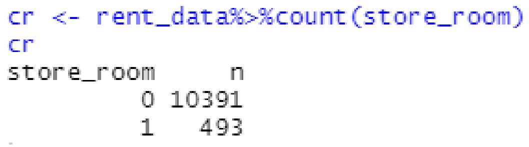
It is a numerical variable. The values fall into either 0 or 1. It is the place for storing grain, foodstuff and/or junk in the house for their ready availability and use in emergency. It is usually near the Kitchen whereas the junk store room may under a staircase or an unused room or a closet.

**Summary(rent_data$store_room)**



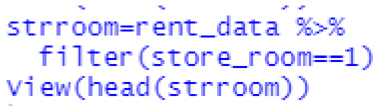
```
summary(rent_data$store_room)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
0.0000  0.0000  0.0000  0.0453  0.0000  1.0000 
```

From the above output, it has been cleared that the Average store room value is 0.0453. The maximum value is 1.000. The minimum value is 0.000. The below R code explains the range of the column frequency  for using  count() function

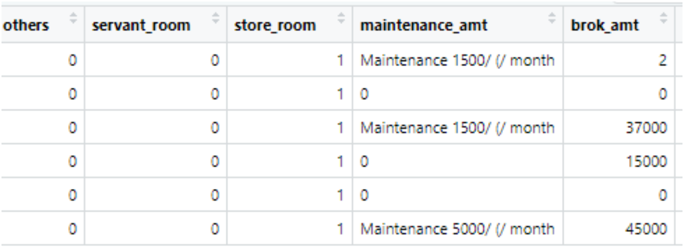


```
cr <- rent_data%>%count(store_room)
cr
store_room     n
         0 10391
         1   493
```

The below code presents the result of filtering command applied on the variable store room. The house which having store room used for this data to collect the record. Totally 10391 records are observed which houses are having store room value as 1.



```
strroom=rent_data %>%
  filter(store_room==1)
view(head(strroom))
```

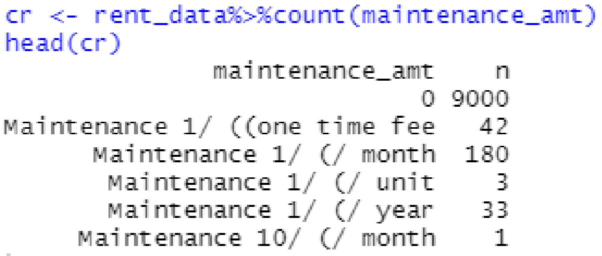


| others | servant_room | store_room | maintenance_amt | brok_amt |
|---|---|---|---|---|
| 0 | 0 | 1 | Maintenance 1500/ (/ month | 2 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | Maintenance 1500/ (/ month | 37000 |
| 0 | 0 | 1 | 0 | 15000 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | Maintenance 5000/ (/ month | 45000 |

**MAINTENANCE_AMT**

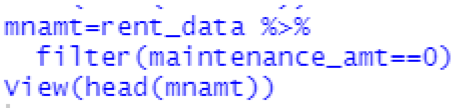
It is a categorical variable. The values fall on either 0 or other calculation. Usually housing societies levy maintenance charges as per the area of the flat or on other variables if the apartments area of same size. There are instances when for one or two years at the time of possession.

The below R code explains the range of the column frequency  for using  count() function

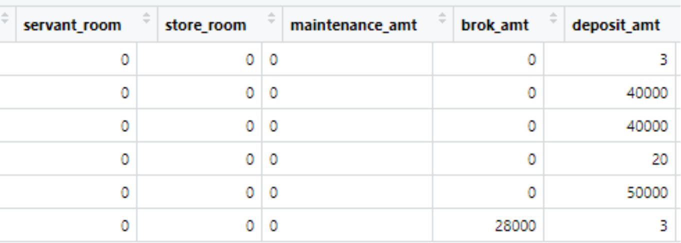


```
cr <- rent_data%>%count(maintenance_amt)
head(cr)
              maintenance_amt    n
                            0 9000
Maintenance 1/ ((one time fee   42
      Maintenance 1/ (/ month  180
       Maintenance 1/ (/ unit    3
       Maintenance 1/ (/ year   33
     Maintenance 10/ (/ month    1
```

The below code presents the result of filtering command applied on the variable maintenance room. The house which having equal to 0  used for this data to collect the record. Totally 9000 records are observed which houses are having equal to 0.



```
mnamt=rent_data %>%
  filter(maintenance_amt==0)
view(head(mnamt))
```

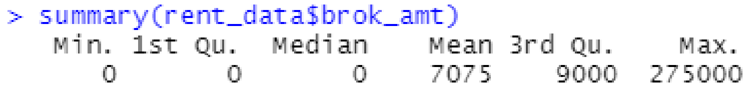


| servant_room | store_room | maintenance_amt | brok_amt | deposit_amt |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 3 |
| 0 | 0 | 0 | 0 | 40000 |
| 0 | 0 | 0 | 0 | 40000 |
| 0 | 0 | 0 | 0 | 20 |
| 0 | 0 | 0 | 0 | 50000 |
| 0 | 0 | 0 | 28000 | 3 |

**BROK_AMT**

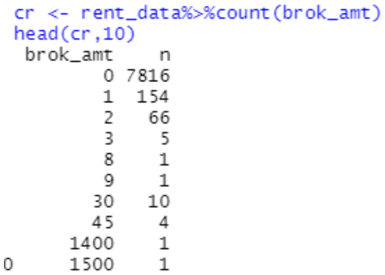
It is a numerical variable. It tells the amount for real estate broker. It is type of real estate broker who acts as the middleman between prospective tenants and property owners or management companies of rental properties. Think of an apartment broker is similar concept to a real estate agent when buying a home.

**Summary(rent_data$brok_amt)**



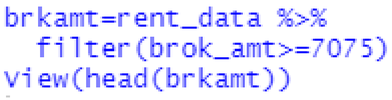
```
> summary(rent_data$brok_amt)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
      0       0       0    7075    9000  275000 
```

From the above output, it has been cleared that the Average brokerage amount value is 7075. The maximum value is 275000. The minimum value is 0. The below R code explains the range of the column frequency  for using  count() function

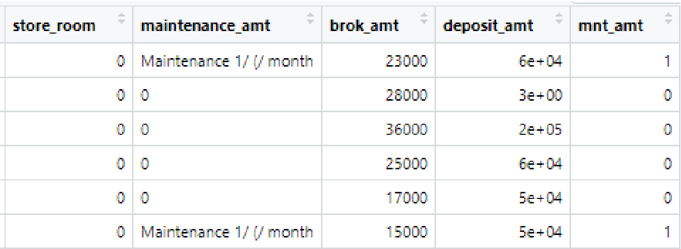


```
cr <- rent_data%>%count(brok_amt)
head(cr,10)
 brok_amt    n
        0 7816
        1  154
        2   66
        3    5
        8    1
        9    1
       30   10
       45    4
     1400    1
0    1500    1
```

The below code presents the result of filtering command applied on the variable brokerage amount. The house which having greater than or equal to average value 7075 used for this data to collect the record. Totally 2766 records are observed which houses are having brokerage value greater than or equal to 7075.



```
brkamt=rent_data %>%
  filter(brok_amt>=7075)
view(head(brkamt))
```

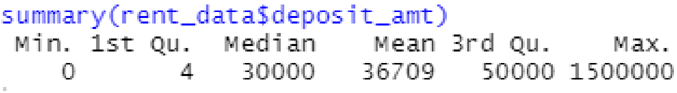


| store_room | maintenance_amt | brok_amt | deposit_amt | mnt_amt |
|---|---|---|---|---|
| 0 | Maintenance 1/ (/ month | 23000 | 6e+04 | 1 |
| 0 | 0 | 28000 | 3e+00 | 0 |
| 0 | 0 | 36000 | 2e+05 | 0 |
| 0 | 0 | 25000 | 6e+04 | 0 |
| 0 | 0 | 17000 | 5e+04 | 0 |
| 0 | Maintenance 1/ (/ month | 15000 | 5e+04 | 1 |

## DEPOSIT_AMT

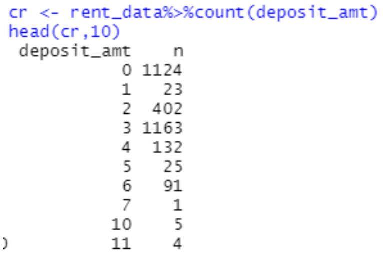
It is numerical variable. The amount collected as a security deposit usually varies from city to city. While renting out any premises, a landlord expects a deposit amount which is usually certain months of rent.

**Summary(rent_data$deposit_amt)**



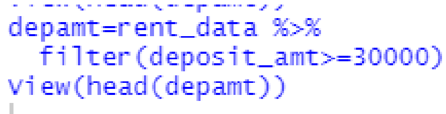
```
summary(rent_data$deposit_amt)
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
    0       4   30000   36709   50000 1500000 
```

From the above output, it has been cleared that the Average deposit amount value is 30000. The maximum value is 1500000. The minimum value is 0. The below R code explains the range of the column frequency  for using  count() function

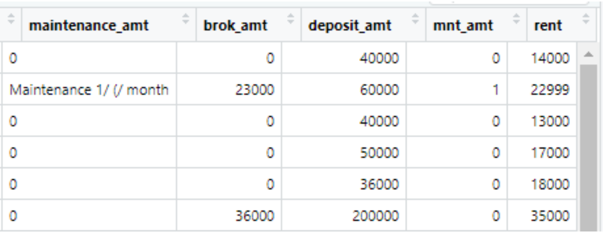


```
cr  <- rent_data%>%count(deposit_amt)
head(cr,10)
 deposit_amt    n
           0 1124
           1   23
           2  402
           3 1163
           4  132
           5   25
           6   91
           7    1
          10    5
          11    4
```

The below code presents the result of filtering command applied on the variable deposit amount. The house which having greater than or equal to average value 30000 used for this data to collect the record. Totally 5652 records are observed which houses are having brokerage value greater than or equal to 30000.



```
depamt=rent_data %>%
  filter(deposit_amt>=30000)
view(head(depamt))
```

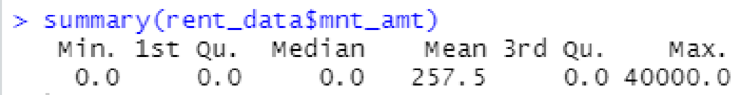


| maintenance_amt | brok_amt | deposit_amt | mnt_amt | rent |
|---|---|---|---|---|
| 0 | 0 | 40000 | 0 | 14000 |
| Maintenance 1/ (/ month | 23000 | 60000 | 1 | 22999 |
| 0 | 0 | 40000 | 0 | 13000 |
| 0 | 0 | 50000 | 0 | 17000 |
| 0 | 0 | 36000 | 0 | 18000 |
| 0 | 36000 | 200000 | 0 | 35000 |

## MNT_AMT

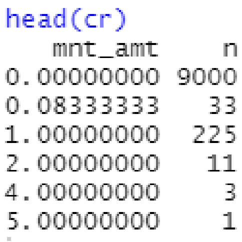
It is a numerical variable. The values falls on either 0 or 1. It tells whether the house has maintenance amount or not according to the value maintenance_amount in 26th column of this dataset.

**Summary(rent_data$mnt_amt)**



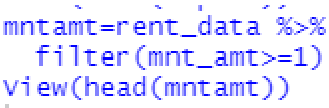
```
> summary(rent_data$mnt_amt)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
    0.0     0.0     0.0   257.5     0.0 40000.0 
```

From the above output, it has been cleared that the Average maintenance amount value is 257.5. The maximum value is 40000.0. The minimum value is 0.0. The below R code explains the range of the column frequency for using count() function

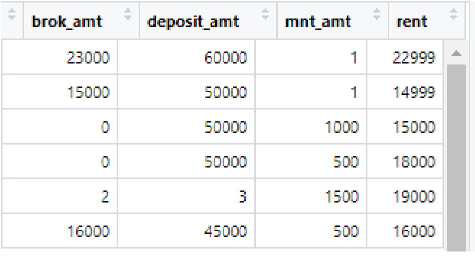


```
head(cr)
   mnt_amt    n
0.00000000 9000
0.08333333   33
1.00000000  225
2.00000000   11
4.00000000    3
5.00000000    1
```

The below code presents the result of filtering command applied on the variable maintenance amount. The house which having greater than or equal to 1 used for this data to collect the record. Totally 1851 records are observed which houses are having maintenance value greater than or equal to 1.



```
mntamt=rent_data %>%
  filter(mnt_amt>=1)
view(head(mntamt))
```

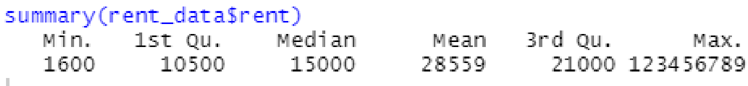


| brok_amt | deposit_amt | mnt_amt | rent |
|---|---|---|---|
| 23000 | 60000 | 1 | 22999 |
| 15000 | 50000 | 1 | 14999 |
| 0 | 50000 | 1000 | 15000 |
| 0 | 50000 | 500 | 18000 |
| 2 | 3 | 1500 | 19000 |
| 16000 | 45000 | 500 | 16000 |

**RENT**

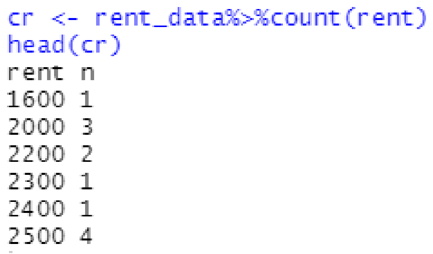
It is a numeric variable. It tells the rent of the house for the features provided in the remaining variables. It is the response variable. It is the value to be predict with the remaining predictors.

**Summary(rent_data$rent)**



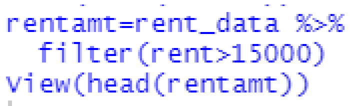
```
summary(rent_data$rent)
   Min.   1st Qu.    Median      Mean   3rd Qu.      Max. 
   1600     10500     15000     28559     21000 123456789 
```

From the above output, it has been cleared that the Average rent value is 28559. The maximum value is 123456789. The minimum value is 1600. The below R code explains the range of the column frequency for using count() function

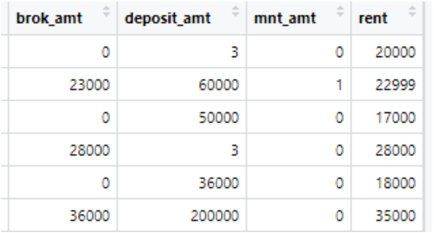


```
cr <- rent_data%>%count(rent)
head(cr)
rent n
1600 1
2000 3
2200 2
2300 1
2400 1
2500 4
```

The below code presents the result of filtering command applied on the variable rent amount. The house which having greater than middle value 15000 used for this data to collect the record. Totally 1851 records are observed which houses are having rent value greater than 15000.



```
rentamt=rent_data %>%
  filter(rent>15000)
view(head(rentamt))
```

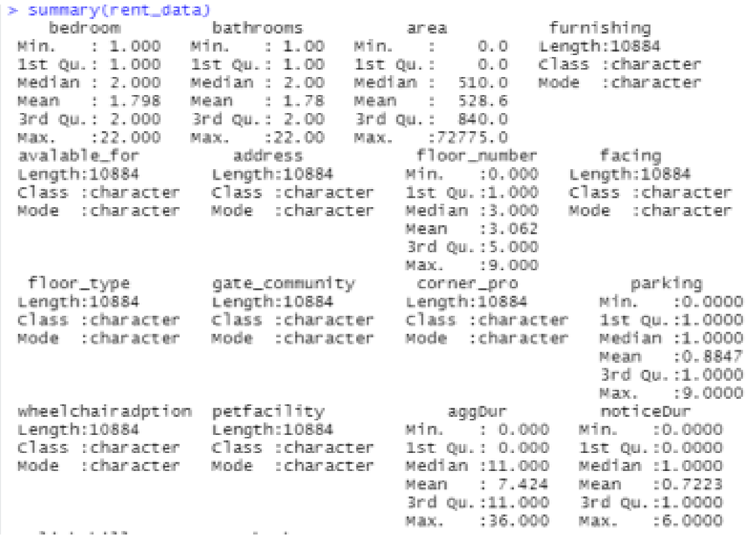


| brok_amt | deposit_amt | mnt_amt | rent |
|---|---|---|---|
| 0 | 3 | 0 | 20000 |
| 23000 | 60000 | 1 | 22999 |
| 0 | 50000 | 0 | 17000 |
| 28000 | 3 | 0 | 28000 |
| 0 | 36000 | 0 | 18000 |
| 36000 | 200000 | 0 | 35000 |

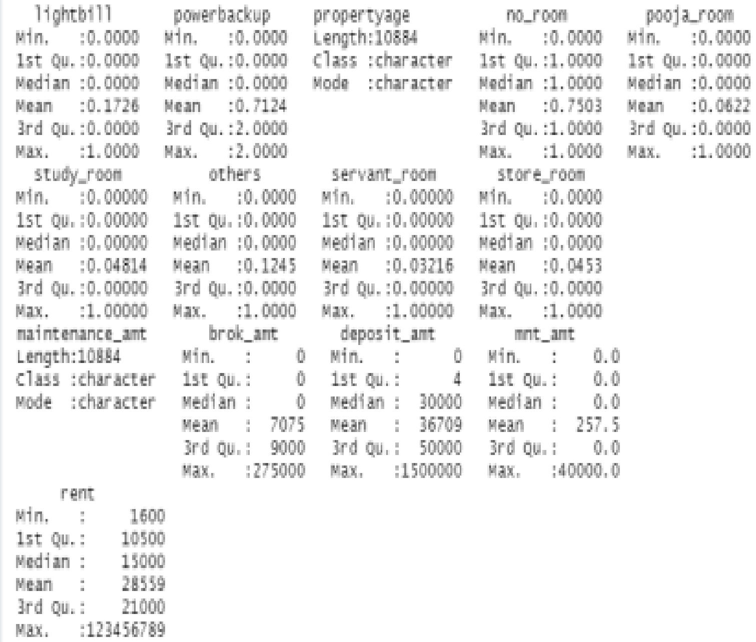
This section examines the nature of all variables available in the given dataset and the values, its count and range in a deep way using R studio.

## CHAPTER III : Exploratory Data Analysis

**Exploratory data** analysis (**EDA**) is used to analyze and investigate **data** sets and summarize their main characteristics, often employing **data** visualization methods. It can also help to determine if the statistical techniques that are considering for **data** analysis are appropriate. Summary() function helps to see the summary of all the variables and a raw information about the values in a single view.



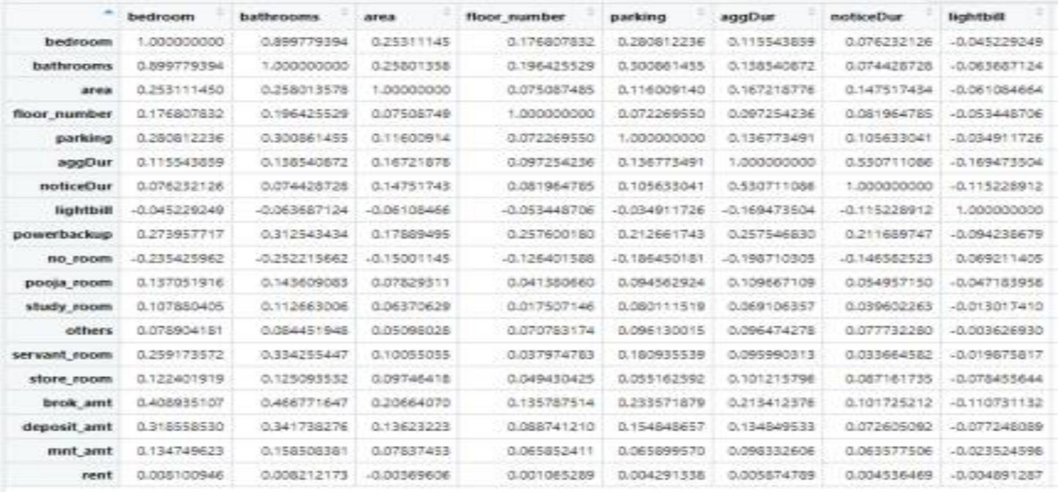
```
> summary(rent_data)
    bedroom          bathrooms           area            furnishing       
 Min.   : 1.000   Min.   : 1.00   Min.   :    0.0   Length:10884      
 1st Qu.: 1.000   1st Qu.: 1.00   1st Qu.:    0.0   Class :character  
 Median : 2.000   Median : 2.00   Median :  510.0   Mode  :character  
 Mean   : 1.798   Mean   : 1.78   Mean   :  528.6                     
 3rd Qu.: 2.000   3rd Qu.: 2.00   3rd Qu.:  840.0                     
 Max.   :22.000   Max.   :22.00   Max.   :72775.0                     
 avalable_for         address           floor_number      facing         
 Length:10884       Length:10884       Min.   :0.000   Length:10884      
 Class :character   Class :character   1st Qu.:1.000   Class :character  
 Mode  :character   Mode  :character   Median :3.000   Mode  :character  
                                       Mean   :3.062                     
                                       3rd Qu.:5.000                     
                                       Max.   :9.000                     
  floor_type        gate_community      corner_pro           parking      
 Length:10884       Length:10884       Length:10884       Min.   :0.0000  
 Class :character   Class :character   Class :character   1st Qu.:1.0000  
 Mode  :character   Mode  :character   Mode  :character   Median :1.0000  
                                                          Mean   :0.8847  
                                                          3rd Qu.:1.0000  
                                                          Max.   :9.0000  
 wheelchairadption  petfacility            aggDur         noticeDur     
 Length:10884       Length:10884       Min.   : 0.000   Min.   :0.0000  
 Class :character   Class :character   1st Qu.: 0.000   1st Qu.:0.0000  
 Mode  :character   Mode  :character   Median :11.000   Median :1.0000  
                                       Mean   : 7.424   Mean   :0.7223  
                                       3rd Qu.:11.000   3rd Qu.:1.0000  
                                       Max.   :36.000   Max.   :6.0000  
```



```
   lightbill       powerbackup      propertyage            no_room          pooja_room    
 Min.   :0.0000   Min.   :0.0000   Length:10884       Min.   :0.0000   Min.   :0.0000  
 1st Qu.:0.0000   1st Qu.:0.0000   Class :character   1st Qu.:1.0000   1st Qu.:0.0000  
 Median :0.0000   Median :0.0000   Mode  :character   Median :1.0000   Median :0.0000  
 Mean   :0.1726   Mean   :0.7124                      Mean   :0.7503   Mean   :0.0622  
 3rd Qu.:0.0000   3rd Qu.:2.0000                      3rd Qu.:1.0000   3rd Qu.:0.0000  
 Max.   :1.0000   Max.   :2.0000                      Max.   :1.0000   Max.   :1.0000  
   study_room          others         servant_room       store_room    
 Min.   :0.00000   Min.   :0.0000   Min.   :0.00000   Min.   :0.0000  
 1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.0000  
 Median :0.00000   Median :0.0000   Median :0.00000   Median :0.0000  
 Mean   :0.04814   Mean   :0.1245   Mean   :0.03216   Mean   :0.0453  
 3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:0.0000  
 Max.   :1.00000   Max.   :1.0000   Max.   :1.00000   Max.   :1.0000  
 maintenance_amt       brok_amt       deposit_amt          mnt_amt       
 Length:10884       Min.   :     0   Min.   :      0   Min.   :    0.0  
 Class :character   1st Qu.:     0   1st Qu.:      4   1st Qu.:    0.0  
 Mode  :character   Median :     0   Median :  30000   Median :    0.0  
                    Mean   :  7075   Mean   :  36709   Mean   :  257.5  
                    3rd Qu.:  9000   3rd Qu.:  50000   3rd Qu.:    0.0  
                    Max.   :275000   Max.   :1500000   Max.   :40000.0  
      rent          
 Min.   :     1600  
 1st Qu.:    10500  
 Median :    15000  
 Mean   :    28559  
 3rd Qu.:    21000  
 Max.   :123456789  
```

To see the relationship between the numeric variables by using cor() function.

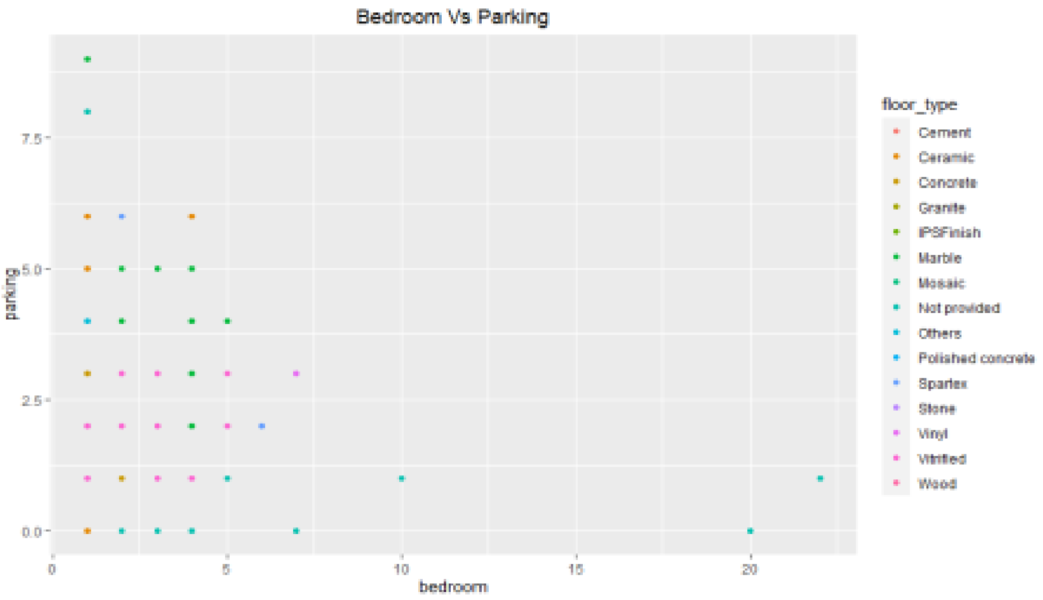
cor(rent_data[,-c(4,5,6,8,9,10,11,13,14,19,26)])

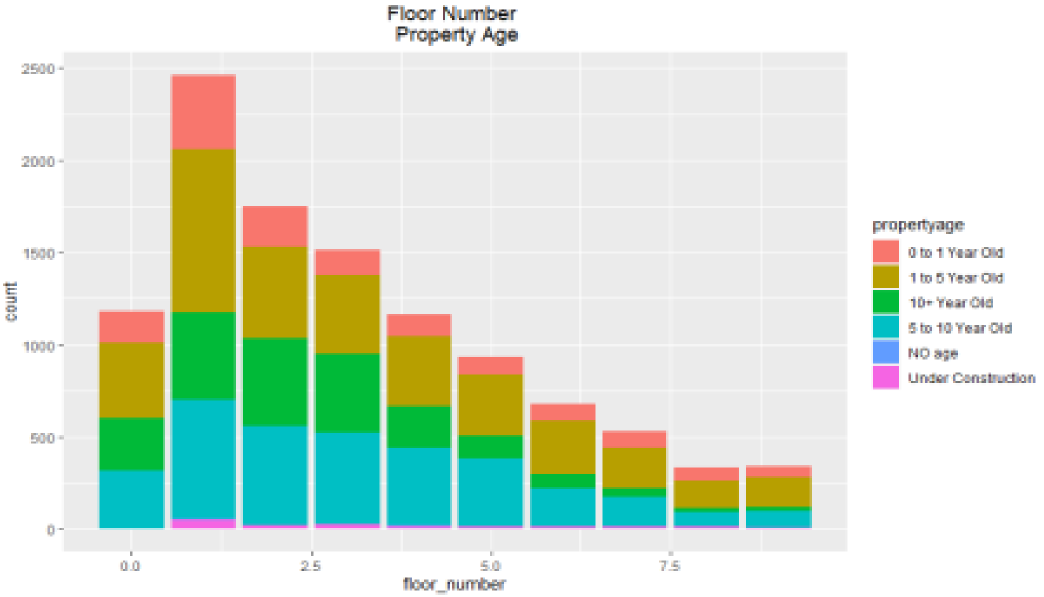


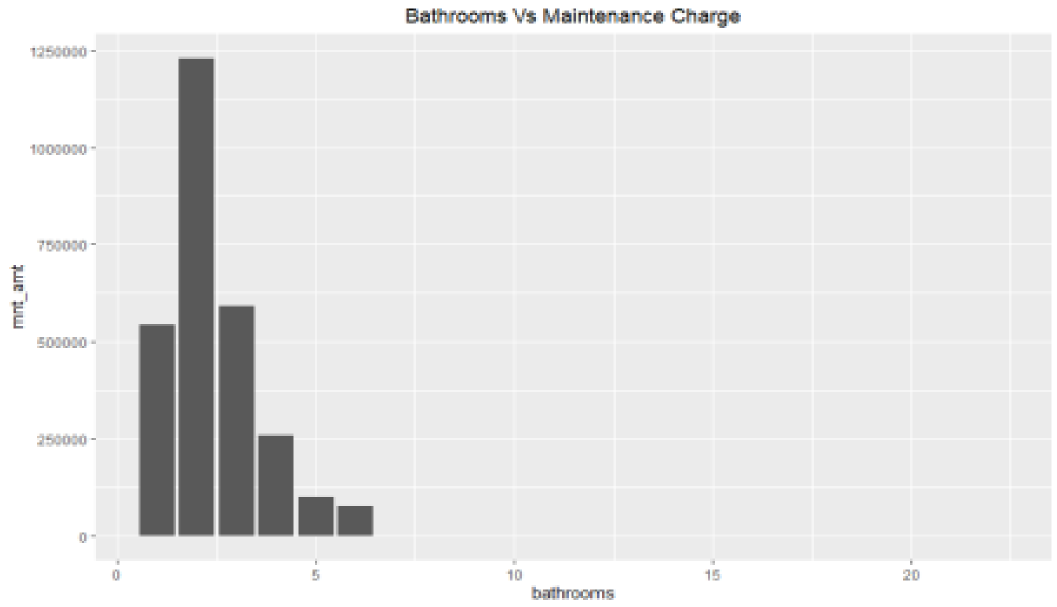
| | bedroom | bathrooms | area | floor_number | parking | aggDur | noticeDur | lightbill |
|---|---|---|---|---|---|---|---|---|
| bedroom | 1.000000000 | 0.899779394 | 0.25311145 | 0.176807832 | 0.280812236 | 0.115543859 | 0.076232126 | -0.045229249 |
| bathrooms | 0.899779394 | 1.000000000 | 0.25801358 | 0.196425529 | 0.300861455 | 0.138540872 | 0.074428728 | -0.063687124 |
| area | 0.253111450 | 0.258013578 | 1.00000000 | 0.075087485 | 0.116009140 | 0.167218776 | 0.147517434 | -0.061084664 |
| floor_number | 0.176807832 | 0.196425529 | 0.07508749 | 1.000000000 | 0.072269550 | 0.097254236 | 0.081964785 | -0.053448706 |
| parking | 0.280812236 | 0.300861455 | 0.11600914 | 0.072269550 | 1.000000000 | 0.136773491 | 0.105633041 | -0.034911726 |
| aggDur | 0.115543859 | 0.138540872 | 0.16721878 | 0.097254236 | 0.136773491 | 1.000000000 | 0.530711086 | -0.169473504 |
| noticeDur | 0.076232126 | 0.074428728 | 0.14751743 | 0.081964785 | 0.105633041 | 0.530711086 | 1.000000000 | -0.115228912 |
| lightbill | -0.045229249 | -0.063687124 | -0.06108466 | -0.053448706 | -0.034911726 | -0.169473504 | -0.115228912 | 1.000000000 |
| powerbackup | 0.273957717 | 0.312543434 | 0.17889495 | 0.257600180 | 0.212661743 | 0.257546830 | 0.211689747 | -0.094238679 |
| no_room | -0.235425962 | -0.252215662 | -0.15001145 | -0.126401588 | -0.186450181 | -0.198710305 | -0.146562523 | 0.069211405 |
| pooja_room | 0.137051916 | 0.143609083 | 0.07829311 | 0.041380660 | 0.094562924 | 0.109667109 | 0.054957150 | -0.047183958 |
| study_room | 0.107880405 | 0.112663006 | 0.06370629 | 0.017507146 | 0.080111519 | 0.069106357 | 0.039602263 | -0.013017410 |
| others | 0.078904181 | 0.084451948 | 0.05098028 | 0.070783174 | 0.096130015 | 0.096474278 | 0.077732280 | -0.003626930 |
| servant_room | 0.259173572 | 0.334255447 | 0.10055055 | 0.037974783 | 0.180935539 | 0.095990313 | 0.033664582 | -0.019675817 |
| store_room | 0.122401919 | 0.125093532 | 0.09746418 | 0.049430425 | 0.055162592 | 0.101215798 | 0.087161735 | -0.078455644 |
| brok_amt | 0.408935107 | 0.466771647 | 0.20664070 | 0.135787514 | 0.233571879 | 0.213412376 | 0.101725212 | -0.110731132 |
| deposit_amt | 0.318558530 | 0.341738276 | 0.13623223 | 0.088741210 | 0.154848657 | 0.134849533 | 0.072605092 | -0.077248089 |
| mnt_amt | 0.134749623 | 0.158508381 | 0.07837453 | 0.065852411 | 0.065899570 | 0.098332606 | 0.063577506 | -0.023524598 |
| rent | 0.008100946 | 0.008212173 | -0.00369606 | 0.001065289 | 0.004291338 | 0.005874789 | 0.004536469 | -0.004891287 |

This correlation matrix gives an overview of the correlations for all combinations of two variables. This matrix results variables except pooja room, study room, others, store room, mnt amt, lightbill has positive correlated with the response variable rent.

The following figures depicts the relationship among the variables in the dataset.

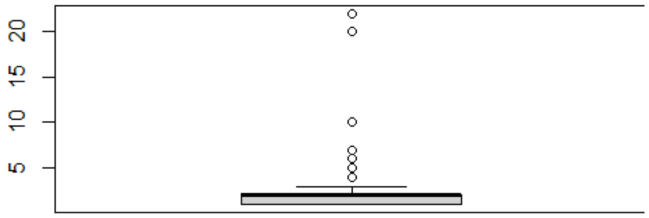






From the Bedroom vs Parking figure, shows that there was an outlier in bedroom variable. Boxplot helps to show the presence of outlier in the particular variable.

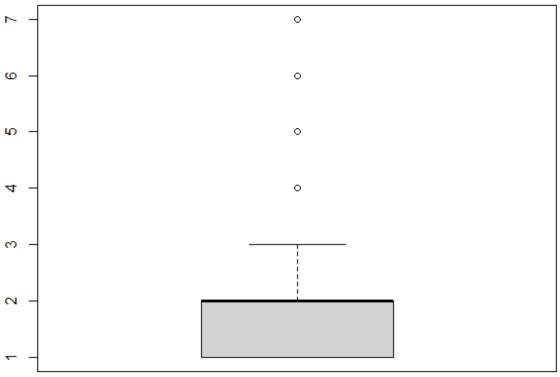
```
boxplot(rentdata$bedroom)
```



Here, the values above the third quartile, a black line out of the box are considered as Outliers.

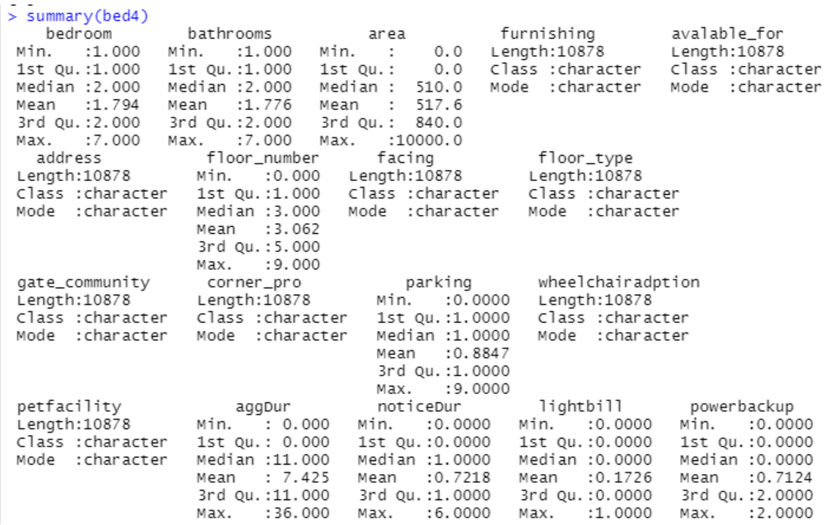
Now, filter command is used to filter the outlier values and assign it into a different variable.

```
bed4 <- rentdata%>%filter(bedroom<=8)
boxplot(bed4$bedroom)
```

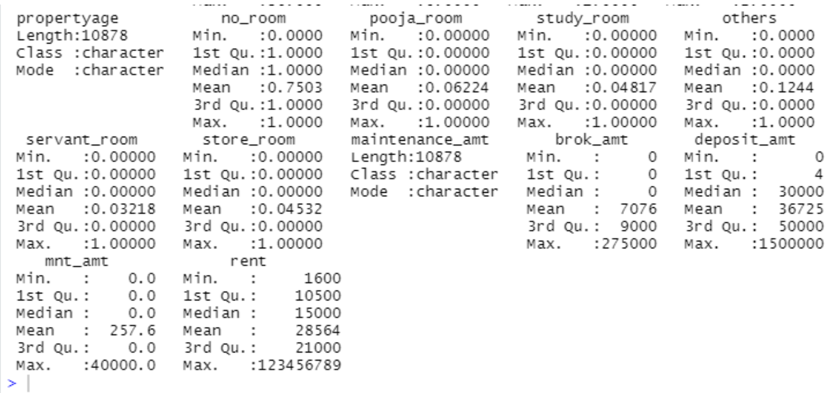


In this case, there may a chance of accuracy is low, if removing the outliers 4 to 7, many of the observations were removed because amount of  observations lies is large. . So here the values of outliers keep as it is and assign to the variable bed4.

The following summary() function is to see the summary of the new dataset bed4.

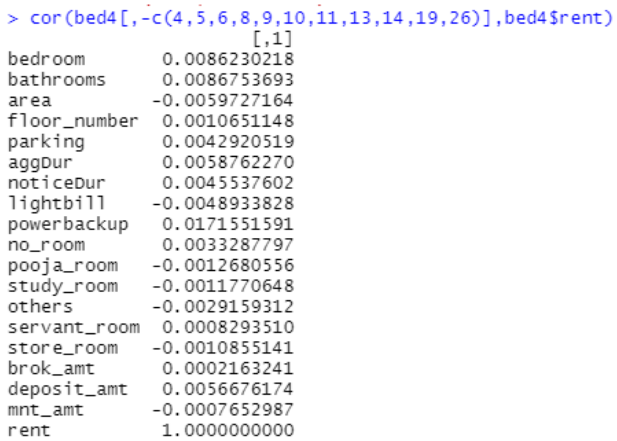


```
> summary(bed4)
    bedroom        bathrooms         area         furnishing         avalable_for
 Min.   :1.000   Min.   :1.000   Min.   :    0.0   Length:10878       Length:10878
 1st Qu.:1.000   1st Qu.:1.000   1st Qu.:    0.0   Class :character   Class :character
 Median :2.000   Median :2.000   Median :  510.0   Mode  :character   Mode  :character
 Mean   :1.794   Mean   :1.776   Mean   :  517.6
 3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:  840.0
 Max.   :7.000   Max.   :7.000   Max.   :10000.0
   address           floor_number      facing             floor_type
 Length:10878       Min.   :0.000   Length:10878       Length:10878
 Class :character   1st Qu.:1.000   Class :character   Class :character
 Mode  :character   Median :3.000   Mode  :character   Mode  :character
                    Mean   :3.062
                    3rd Qu.:5.000
                    Max.   :9.000
 gate_community      corner_pro           parking        wheelchairadption
 Length:10878       Length:10878       Min.   :0.0000   Length:10878
 Class :character   Class :character   1st Qu.:1.0000   Class :character
 Mode  :character   Mode  :character   Median :1.0000   Mode  :character
                                       Mean   :0.8847
                                       3rd Qu.:1.0000
                                       Max.   :9.0000
 petfacility            aggDur         noticeDur        lightbill        powerbackup
 Length:10878       Min.   : 0.000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 Class :character   1st Qu.: 0.000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
 Mode  :character   Median :11.000   Median :1.0000   Median :0.0000   Median :0.0000
                    Mean   : 7.425   Mean   :0.7218   Mean   :0.1726   Mean   :0.7124
                    3rd Qu.:11.000   3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:2.0000
                    Max.   :36.000   Max.   :6.0000   Max.   :1.0000   Max.   :2.0000
```



```
 propertyage           no_room         pooja_room        study_room           others
 Length:10878       Min.   :0.0000   Min.   :0.00000   Min.   :0.00000   Min.   :0.0000
 Class :character   1st Qu.:1.0000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.0000
 Mode  :character   Median :1.0000   Median :0.00000   Median :0.00000   Median :0.0000
                    Mean   :0.7503   Mean   :0.06224   Mean   :0.04817   Mean   :0.1244
                    3rd Qu.:1.0000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.0000
                    Max.   :1.0000   Max.   :1.00000   Max.   :1.00000   Max.   :1.0000
  servant_room       store_room      maintenance_amt       brok_amt        deposit_amt
 Min.   :0.00000   Min.   :0.00000   Length:10878       Min.   :     0   Min.   :      0
 1st Qu.:0.00000   1st Qu.:0.00000   Class :character   1st Qu.:     0   1st Qu.:      4
 Median :0.00000   Median :0.00000   Mode  :character   Median :     0   Median :  30000
 Mean   :0.03218   Mean   :0.04532                      Mean   :  7076   Mean   :  36725
 3rd Qu.:0.00000   3rd Qu.:0.00000                      3rd Qu.:  9000   3rd Qu.:  50000
 Max.   :1.00000   Max.   :1.00000                      Max.   :275000   Max.   :1500000
    mnt_amt             rent
 Min.   :    0.0   Min.   :     1600
 1st Qu.:    0.0   1st Qu.:    10500
 Median :    0.0   Median :    15000
 Mean   :  257.6   Mean   :    28564
 3rd Qu.:    0.0   3rd Qu.:    21000
 Max.   :40000.0   Max.   :123456789
> 
```

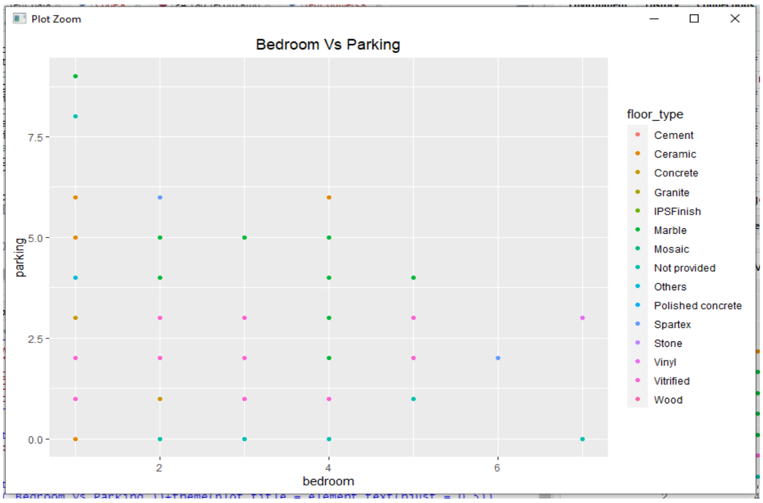
The following cor() is to see the relationship between the combination of all numeric variables for the response variable in the dataset bed4.



```
> cor(bed4[,-c(4,5,6,8,9,10,11,13,14,19,26)],bed4$rent)
                       [,1]
bedroom         0.0086230218
bathrooms       0.0086753693
area           -0.0059727164
floor_number    0.0010651148
parking         0.0042920519
aggDur          0.0058762270
noticeDur       0.0045537602
lightbill      -0.0048933828
powerbackup     0.0171551591
no_room         0.0033287797
pooja_room     -0.0012680556
study_room     -0.0011770648
others         -0.0029159312
servant_room    0.0008293510
store_room     -0.0010855141
brok_amt        0.0002163241
deposit_amt     0.0056676174
mnt_amt        -0.0007652987
rent            1.0000000000
```

The above result shows that except area, lightbill, pooja_room, study_room, others, store_room, mnt_amt all the remaining variables has positive correlated with the response variable rent.

The following figures with the same axis for bed4 dataset, which after removing some outliers.



# Chapter IV : Model Building

## 4.1 Algorithm

Random Forest is a powerful tool used machine learning algorithm extensively across a multitude of fields. It is a way of averaging multiple deep decision trees. It is an ensemble learning method for regression that operate by constructing a lot of decision trees. It comes at the expense of a some loss of interpretability, but generally greatly boosts the performance of the final model. Every observation is fed into every decision tree. The most common outcome for each observation is used as the final output. An error estimate is made for the cases which were not used while building the tree. That is called an OOB (Out-Of-Bag) error estimate which is mentioned as a percentage. The R package **“randomForest”** is used to create random forests.

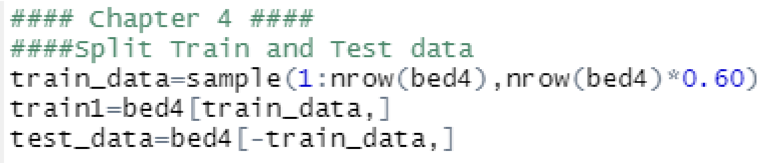
The term ‘Boosting’ refers to a family of algorithms which converts weak learner to strong learners. It is a general ensemble method. This is done by building a model from the training data, then creating a second model that attempts to correct the errors from the first model. Models are added until the training set is predicted perfectly or a maximum number of models are added. The learners are trained sequentially, in order to be able to perform the task of data weighting or filtering. The R package **“gbm”** is used to create boosting.

This section describes the model building using Random Forest and Boosting algorithms with the given dataset. Create models using randomForest() and gbm() functions in R and comparing the two models. This function creates model between the predictor and the response variable.

### 4.2 Training and test dataset

This project with the rent dataset. The goal of the dataset is to predict the house price rate of individuals based on different independent variables. Split the dataset into training set and testing set before the model building. The 60% data will be split into training set and 40% data will be split into testing set.

Analysis Services randomly samples the data to help ensure that the testing and training sets are similar. By using similar data for training and testing, it can minimize the effects of data discrepancies and better understand the characteristics of the model. After a model has been processed by using the training set, test the model by making predictions against the test set. Because the data in the testing set already contains known values for the attribute that to predict, it is easy to determine whether the model’s guesses are correct.

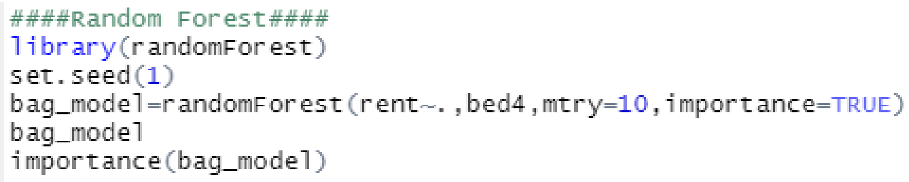


```
#### Chapter 4 ####
####Split Train and Test data
train_data=sample(1:nrow(bed4),nrow(bed4)*0.60)
train1=bed4[train_data,]
test_data=bed4[-train_data,]
```

### 4.3 Model

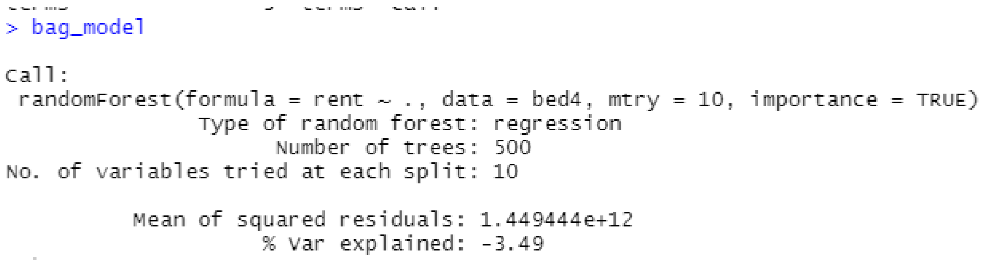
Once the dataset splitted into training and test dataset, build a model with training dataset. The following R code has been implemented the random forest and the boosting model, and predict the target variable Rent.

In this analysis, build two different models for predicting the target variable Rent. For the both model consider all features as predictors and pass the training dataset. By analysing the summary of model it is observed that how the predictors and target variables are related. Let’s try with the random forest model based on all variables vs rent.



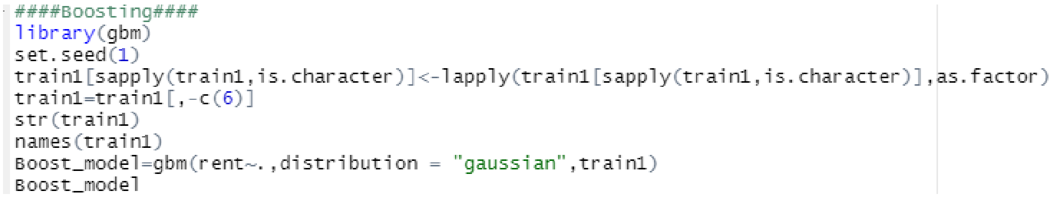
```
####Random Forest####
library(randomForest)
set.seed(1)
bag_model=randomForest(rent~.,bed4,mtry=10,importance=TRUE)
bag_model
importance(bag_model)
```

The above R code built a model with the default characteristics as shown in the following figure.



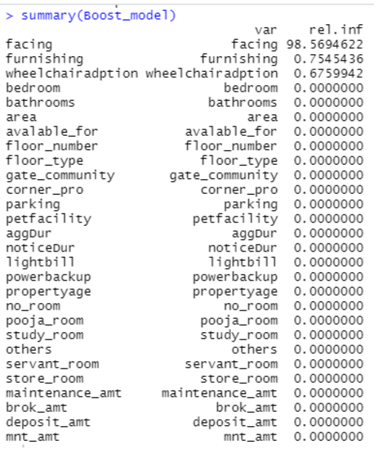
```
> bag_model

Call:
 randomForest(formula = rent ~ ., data = bed4, mtry = 10, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 10

          Mean of squared residuals: 1.449444e+12
                    % Var explained: -3.49
```

From the above summary, the main variable is mtry = 10, the value which is used to number of variables tried at each split, calculated by p/3, p is the number of predictors. The negative value of explained variance which is also called R^2 tells that the model has worse fit. Build different models based on the variables as pre-processed below and will try to find out the best model fit for the data and compare it with the already built model.



```
####Boosting####
library(gbm)
set.seed(1)
train1[sapply(train1,is.character)]<-lapply(train1[sapply(train1,is.character)],as.factor)
train1=train1[,-c(6)]
str(train1)
names(train1)
Boost_model=gbm(rent~.,distribution = "gaussian",train1)
Boost_model
```

Boosting is used to create a model only for the non-numeric variables. Here the changing the data type of character into factor was done. But the variable “address”  has length, so data type conversion is not possible. So, the above R code was built by removing the variable “address”.



```
> summary(Boost_model)
                                 var    rel.inf
facing                        facing 98.5694622
furnishing                furnishing  0.7545436
wheelchairadption wheelchairadption  0.6759942
bedroom                      bedroom  0.0000000
bathrooms                  bathrooms  0.0000000
area                            area  0.0000000
avalable_for            avalable_for  0.0000000
floor_number            floor_number  0.0000000
floor_type                floor_type  0.0000000
gate_community        gate_community  0.0000000
corner_pro                corner_pro  0.0000000
parking                      parking  0.0000000
petfacility              petfacility  0.0000000
aggDur                        aggDur  0.0000000
noticeDur                  noticeDur  0.0000000
lightbill                  lightbill  0.0000000
powerbackup              powerbackup  0.0000000
propertyage              propertyage  0.0000000
no_room                      no_room  0.0000000
pooja_room                pooja_room  0.0000000
study_room                study_room  0.0000000
others                        others  0.0000000
servant_room            servant_room  0.0000000
store_room                store_room  0.0000000
maintenance_amt      maintenance_amt  0.0000000
brok_amt                    brok_amt  0.0000000
deposit_amt              deposit_amt  0.0000000
mnt_amt                      mnt_amt  0.0000000
```

From the above summary, the three variables “facing”, “furnishing”, “wheelchairadption” influencing non-zero. The three variables are factor type. So, approximately the rent of the house might be based on this three variables among all the variables.

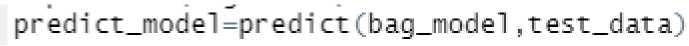
This chapter built 2 different models and assess its summary with various aspects.

# CHAPTER V : Evaluation of model

## 5.1 Model Evaluation

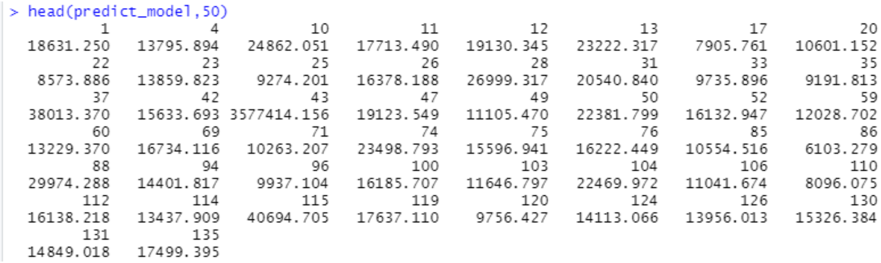
Evaluating algorithm is an essential part of any project. The model may give satisfying results when evaluated using a metric accuracy score but may give poor results when evaluated against the model which is not suited for the data. The performance measure is the way to evaluate a solution to the problem. It is the measurement that will make of the predictions made by a trained model on the test model. Performance measures are typically specialized to the class of problem that are working with, for example classification, regression and clustering. Many standard performance measures will give a score that is meaningful to the problem domain.

Since this project is related to  regression model,  the commonly used performance measure is Mean Squared Error (MSE). It measures the average of the squares – that is, the average squared difference between the estimated  values and what is estimated. It is a risk function, corresponding to the expected value of the square error loss. The fact that the MSE is almost always strictly positive (and not zero) is because of randomness or because the estimator does not account for information that could produce a more accurate estimate. The next measure is Residual Sum of Squares (RSS) is a statistical technique used to measure the amount of variance in a dataset that is not explained by a regression model itself. Instead, it estimates the variance in the residuals or error term. Before evaluate the test error, apply the Built model to the test data.



```
predict_model=predict(bag_model,test_data)
```

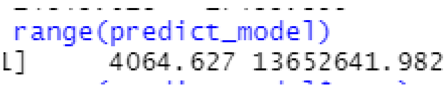
The above code predicts the house price for test data by using the random forest model that has been built in the previous chapter. Now,  let’s look at the first few values of prediction.



```
> head(predict_model,50)
        1         4        10        11        12        13        17        20 
18631.250 13795.894 24862.051 17713.490 19130.345 23222.317  7905.761 10601.152 
       22        23        25        26        28        31        33        35 
 8573.886 13859.823  9274.201 16378.188 26999.317 20540.840  9735.896  9191.813 
       37        42        43        47        49        50        52        59 
38013.370 15633.693 3577414.156 19123.549 11105.470 22381.799 16132.947 12028.702 
       60        69        71        74        75        76        85        86 
13229.370 16734.116 10263.207 23498.793 15596.941 16222.449 10554.516  6103.279 
       88        94        96       100       103       104       106       110 
29974.288 14401.817  9937.104 16185.707 11646.797 22469.972 11041.674  8096.075 
      112       114       115       119       120       124       126       130 
16138.218 13437.909 40694.705 17637.110  9756.427 14113.066 13956.013 15326.384 
      131       135 
14849.018 17499.395 
```

### Range

Range function in R returns a vector containing the minimum and maximum of predictions produced for test data.

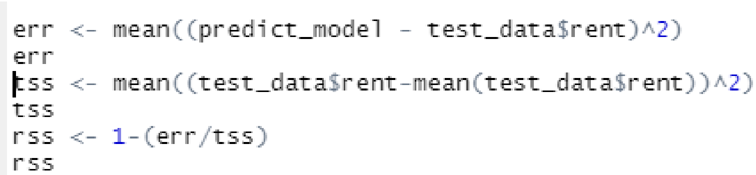


```
range(predict_model)
L]      4064.627 13652641.982
```

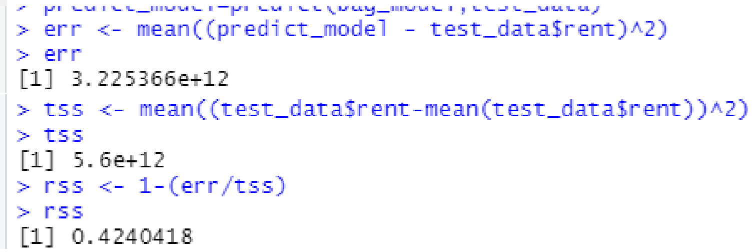
The model built in the analysis predicts the value of rent of the house varies from 4064.62 to 13652641.98.

**RSS**

It measures the amount of error remaining between the regression function and the dataset after the model has been run. The smaller the RSS value, the better the model fits for data.

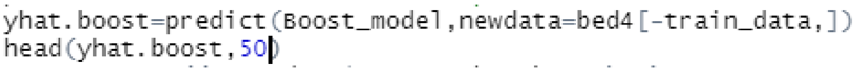


```
err <- mean((predict_model - test_data$rent)^2)
err
tss <- mean((test_data$rent-mean(test_data$rent))^2)
tss
rss <- 1-(err/tss)
rss
```



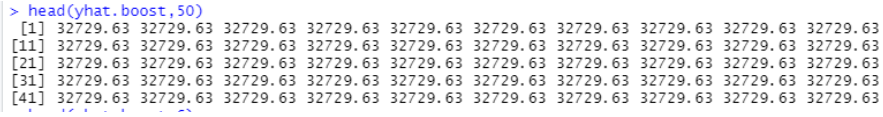
```
> err <- mean((predict_model - test_data$rent)^2)
> err
[1] 3.225366e+12
> tss <- mean((test_data$rent-mean(test_data$rent))^2)
> tss
[1] 5.6e+12
> rss <- 1-(err/tss)
> rss
[1] 0.4240418
```

From the above analysis, 42% of model fits for the data.



```
yhat.boost=predict(Boost_model,newdata=bed4[-train_data,])
head(yhat.boost,50)
```

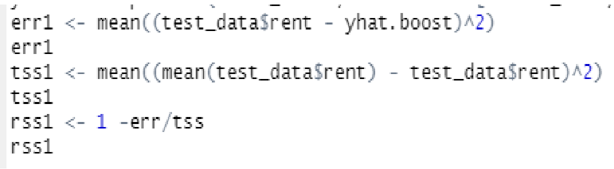
The above code predicts the house price for test data by using the boosting model that has been built in the previous chapter. Now,  let’s look at the first few values of prediction.



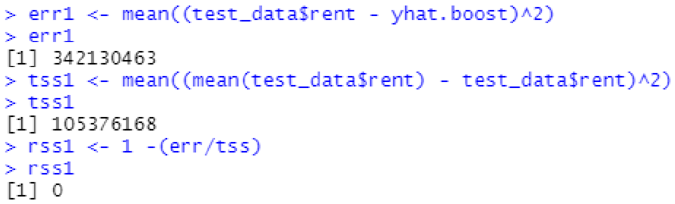
```
> head(yhat.boost,50)
 [1] 32729.63 32729.63 32729.63 32729.63 32729.63 32729.63 32729.63 32729.63 32729.63 32729.63
[11] 32729.63 32729.63 32729.63 32729.63 32729.63 32729.63 32729.63 32729.63 32729.63 32729.63
[21] 32729.63 32729.63 32729.63 32729.63 32729.63 32729.63 32729.63 32729.63 32729.63 32729.63
[31] 32729.63 32729.63 32729.63 32729.63 32729.63 32729.63 32729.63 32729.63 32729.63 32729.63
[41] 32729.63 32729.63 32729.63 32729.63 32729.63 32729.63 32729.63 32729.63 32729.63 32729.63
```

**RSS**

It measures the amount of error remaining between the regression function and the dataset after the model has been run. The smaller the RSS value, the better the model fits for data.



```
err1 <- mean((test_data$rent - yhat.boost)^2)
err1
tss1 <- mean((mean(test_data$rent) - test_data$rent)^2)
tss1
rss1 <- 1 -err/tss
rss1
```



```
> err1 <- mean((test_data$rent - yhat.boost)^2)
> err1
[1] 342130463
> tss1 <- mean((mean(test_data$rent) - test_data$rent)^2)
> tss1
[1] 105376168
> rss1 <- 1 -(err/tss)
> rss1
[1] 0
```

From the above analysis, the value of RSS is 0. That is 100% the above model is perfect fit for the data.

# CHAPTER VI : CONCLUSION

The built model predicts the rent of the house of the given dataset named rent_data, with following conclusions.

- The RSS value of random forest model is 0.42.
- The RSS value of boosting model is 0.
- Comparing the two values, RSS value of boosting model is small and zero.
- As truth, RSS value of zero means your model is a perfect fit for the data.
- Therefore, the final conclusion is Boosting Model is perfect fit for the data.