CW2 - Individual project Natalia Kusek - M00736050

This submission contains of three attempts to solve the given problem.
Third attempt uses two additional datasets.
I define problem definition and I do pre-processing for all three attempts in the next two sections.

Problem definition

Three datasets are available: bike_journeys, bike_stations and LondonCensus.
Additional datasets used in third attempt: pois (points of interest, i.e. landmarks, bars, restaurants, parks) and tfl_stations (underground stations, DLR stations).
Spatial granularity: each bike station.
Temporal granularity: one hour time slot.
Goal: predicting the total number of bikes rented in each bike station with the temporal granularity of one hour time slot.

Pre-processing:

Importing the data.

```
library(data.table)
bike_journeys = fread('data/bike_journeys.csv')
bike_stations = fread('data/bike_stations.csv')
census = fread('data/London_census.csv')
pois = fread('data/pois.csv')
tfl_stations = fread('data/london_stations_underground_dlr_with_entry_and_exit.csv'
)
```

Looking at the data.

```
head(bike_journeys)
```

```
##    Journey_Duration Journey_ID End_Date End_Month End_Year End_Hour End_Minute
## 1:             2040        953       19         9       17       18          0
## 2:             1800      12581       19         9       17       15         21
## 3:             1140       1159       15         9       17       17          1
## 4:              420       2375       14         9       17       12         16
## 5:             1200      14659       13         9       17       19         33
## 6:             1320       2351       14         9       17       14         53
##    End_Station_ID Start_Date Start_Month Start_Year Start_Hour Start_Minute
## 1:            478         19           9         17         17           26
## 2:            122         19           9         17         14           51
## 3:            639         15           9         17         16           42
## 4:            755         14           9         17         12            9
## 5:            605         13           9         17         19           13
## 6:            514         14           9         17         14           31
##    Start_Station_ID
## 1:              251
## 2:              550
## 3:              212
## 4:              163
## 5:               36
## 6:              589
```

```
head(bike_stations)
```

```
##    Station_ID Capacity Latitude Longitude                          Station_Name
## 1:          1       19 51.52916 -0.109970            River Street , Clerkenwell
## 2:          2       37 51.49961 -0.197574        Phillimore Gardens, Kensington
## 3:          3       32 51.52128 -0.084605 Christopher Street, Liverpool Street
## 4:          4       23 51.53006 -0.120973      St. Chad's Street, King's Cross
## 5:          5       27 51.49313 -0.156876         Sedding Street, Sloane Square
## 6:          6       18 51.51812 -0.144228       Broadcasting House, Marylebone
```

```
head(census)
```

```
##     WardCode       WardName              borough NESW AreaSqKm      lon
## 1: E05000026          Abbey Barking and Dagenham East      1.3 0.077935
## 2: E05000027         Alibon Barking and Dagenham East      1.4 0.148270
## 3: E05000028      Becontree Barking and Dagenham East      1.3 0.118957
## 4: E05000029 Chadwell Heath Barking and Dagenham East      3.4 0.139985
## 5: E05000030      Eastbrook Barking and Dagenham East      3.5 0.173581
## 6: E05000031       Eastbury Barking and Dagenham East      1.4 0.105683
##         lat IncomeScor LivingEnSc NoEmployee GrenSpace PopDen BornUK NotBornUK
## 1: 51.53971       0.27      42.76        7900      19.6 9884.6   5459       7327
## 2: 51.54559       0.28      27.96         800      22.4 7464.3   7824       2561
## 3: 51.55453       0.25      31.59        1100       3.0 8923.1   8075       3470
## 4: 51.58475       0.27      34.78        1700      56.4 2970.6   7539       2482
## 5: 51.55365       0.19      21.25        4000      51.1 3014.3   8514       1992
## 6: 51.53590       0.27      31.16        1000      18.1 8357.1   7880       3744
##    NoCTFtoH NoDwelling NoFlats NoHouses NoOwndDwel MedHPrice
## 1:      0.1       4733    3153     1600        1545    177000
## 2:      0.1       4045     574     3471        1849    160000
## 3:      0.1       4378     837     3541        2093    170000
## 4:      0.4       4050    1400     2662        2148    195000
## 5:      0.5       3976     742     3235        2646    191750
## 6:      0.0       4321     933     3388        1913    167250
```

```
head(pois)
```

```
##                               Factual ID              Name
## 1: e3403005-0e66-4c4f-8fed-79020d7b2d48    Somerset House
## 2: 084addc5-ca4d-41f3-a99f-ee7e78ca46c8 Buckingham Palace
## 3: c3fcbc43-e338-434a-b86b-0a4131ca8b8b   St James's Park
## 4: fa53d100-5a56-4f2d-b871-9925cdc724ce         Hyde Park
## 5: 3a6f1f6f-8e5f-44e5-8d77-f546c049b87c  Trafalgar Square
## 6: b5bdbfff-3066-4440-a558-64ef23ba876a    Battersea Park
##                    Address Locality (Town/City) Region (State/Province)
## 1: Somerset House, Strand               London          Greater London
## 2:   Buckingham Palace Rd               London          Greater London
## 3:        Horse Guards Rd               London          Greater London
## 4:          Rangers Lodge               London          Greater London
## 5:           Trafalgar Sq               London          Greater London
## 6:           Battersea Pk               London          Greater London
##    Post Code                                    Neighborhood Latitude Longitude
## 1:  WC2R 1LA    Strand,Covent Garden/ Holborn,Covent Garden 51.51100 -0.117106
## 2:  SW1A 1AA     Westminster,Victoria,Westminster/ St James 51.50120 -0.141833
## 3:  SW1A 2BJ                   St James's Park,St. James's 51.50287 -0.133767
## 4:    W2 2UH              Mayfair,St. James's,The West End 51.50870 -0.163192
## 5:  WC2N 5DN Trafalgar Square / Embankment,Leicester Square 51.50779 -0.128031
## 6:  SW11 4NJ              Battersea,Battersea Park,Chelsea 51.47927 -0.157158
##                       Category Labels
## 1: Landmarks,Buildings and Structures
## 2: Landmarks,Buildings and Structures
## 3:                   Landmarks,Parks
## 4:                   Landmarks,Parks
## 5:                        Landmarks
## 6:                   Landmarks,Parks
```

```
head(tfl_stations)
```

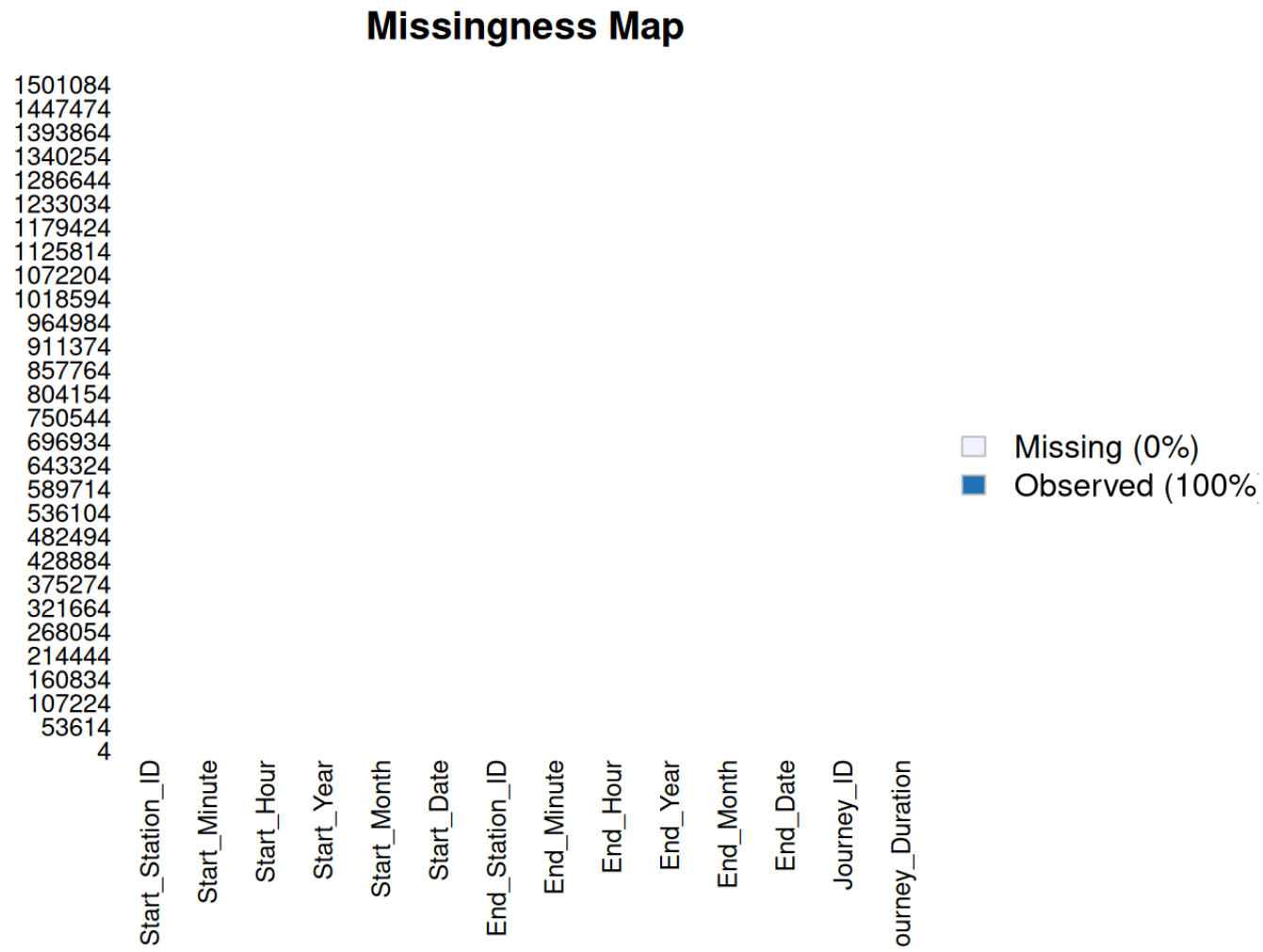
```
##    V1 latitude longitude         name total_lines rail      Station
## 1:  0  51.5028   -0.2801   Acton Town           2    0   Acton Town
## 2:  1  51.5143   -0.0755      Aldgate           2    0      Aldgate
## 3:  2  51.5154   -0.0726 Aldgate East           2    0 Aldgate East
## 4:  3  51.5407   -0.2997     Alperton           1    0     Alperton
## 5:  4  51.5322   -0.1058        Angel           1    0        Angel
## 6:  5  51.5653   -0.1353      Archway           1    0      Archway
##    EntryWeekday EntrySaturday EntrySunday ExitWeekday ExitSaturday ExitSunday
## 1:         9531          6716        4744        9382         6617       4785
## 2:        15080          4397        3261       16023         5909       4230
## 3:        22327         16166       13323       21071        13893      11347
## 4:         4495          3279        2345        5081         3392       2445
## 5:        30413         20944       13916       30099        22368      13832
## 6:        15336         10701        7424       13956         9023       6816
```

Checking for missing data in each dataset.

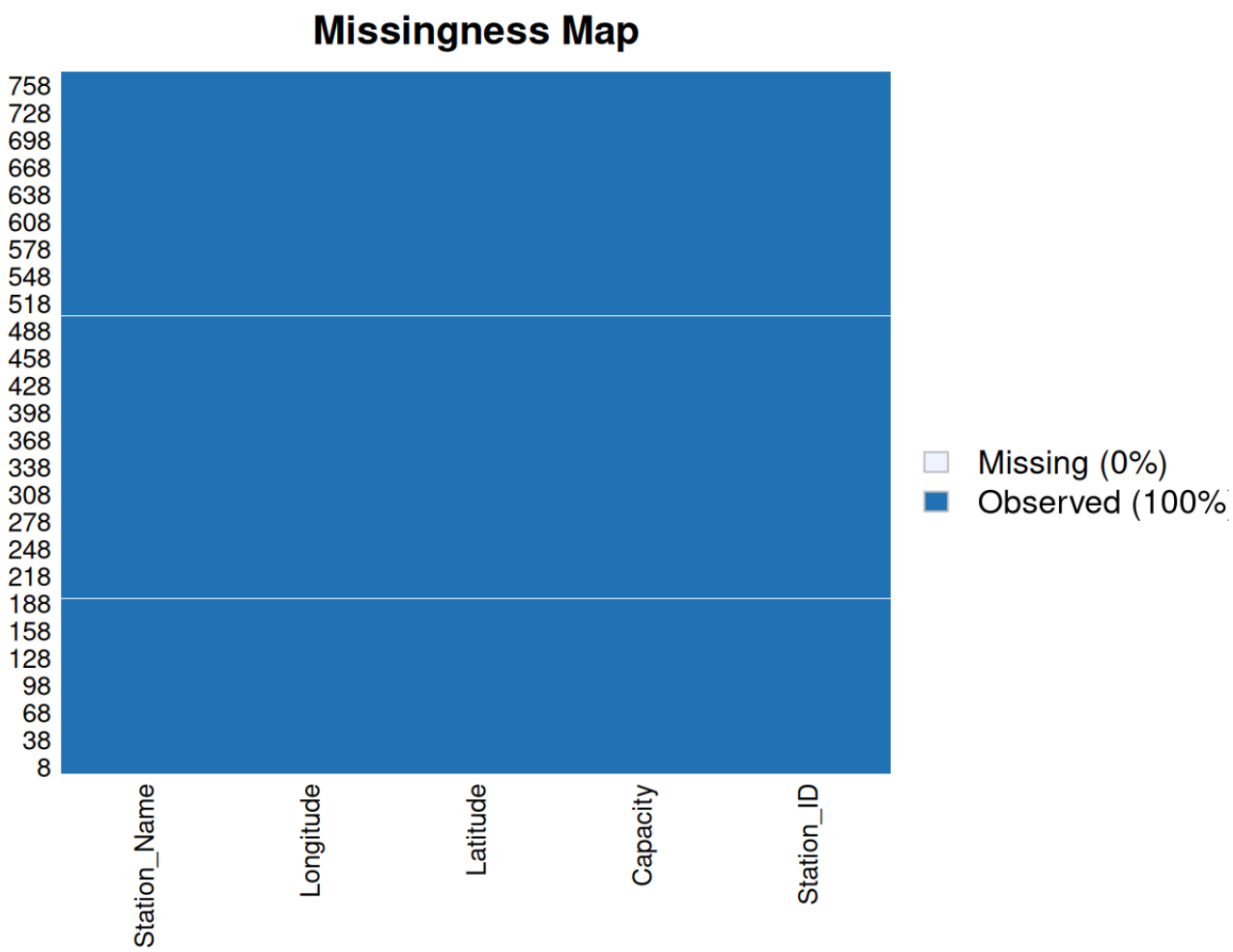
```
library(Rcpp)
library(Amelia)
```

```
## ##
## ## Amelia II: Multiple Imputation
## ## (Version 1.7.6, built: 2019-11-24)
## ## Copyright (C) 2005-2020 James Honaker, Gary King and Matthew Blackwell
## ## Refer to http://gking.harvard.edu/amelia/ for more information
## ##
```

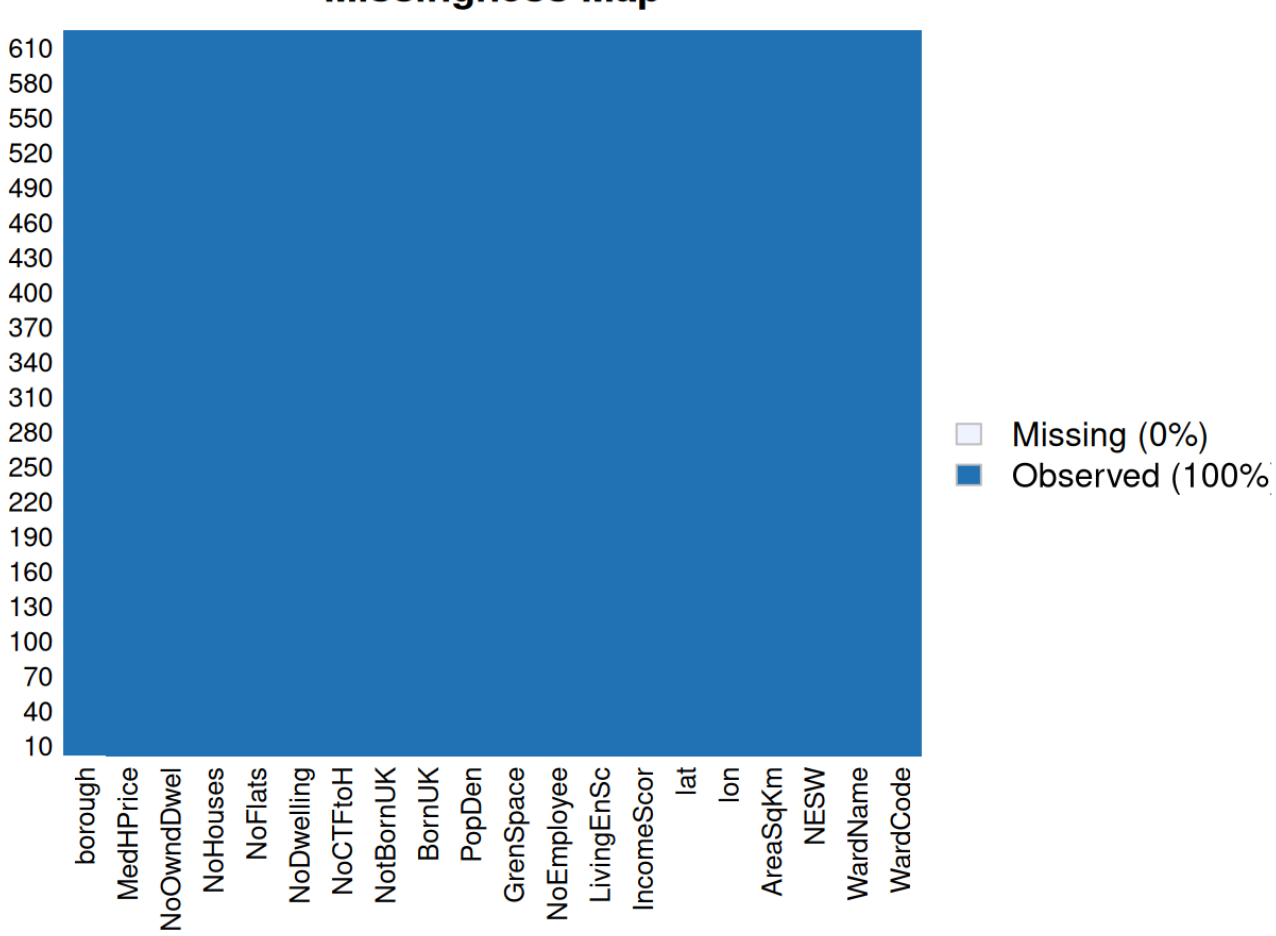
```
missmap(bike_journeys)
```



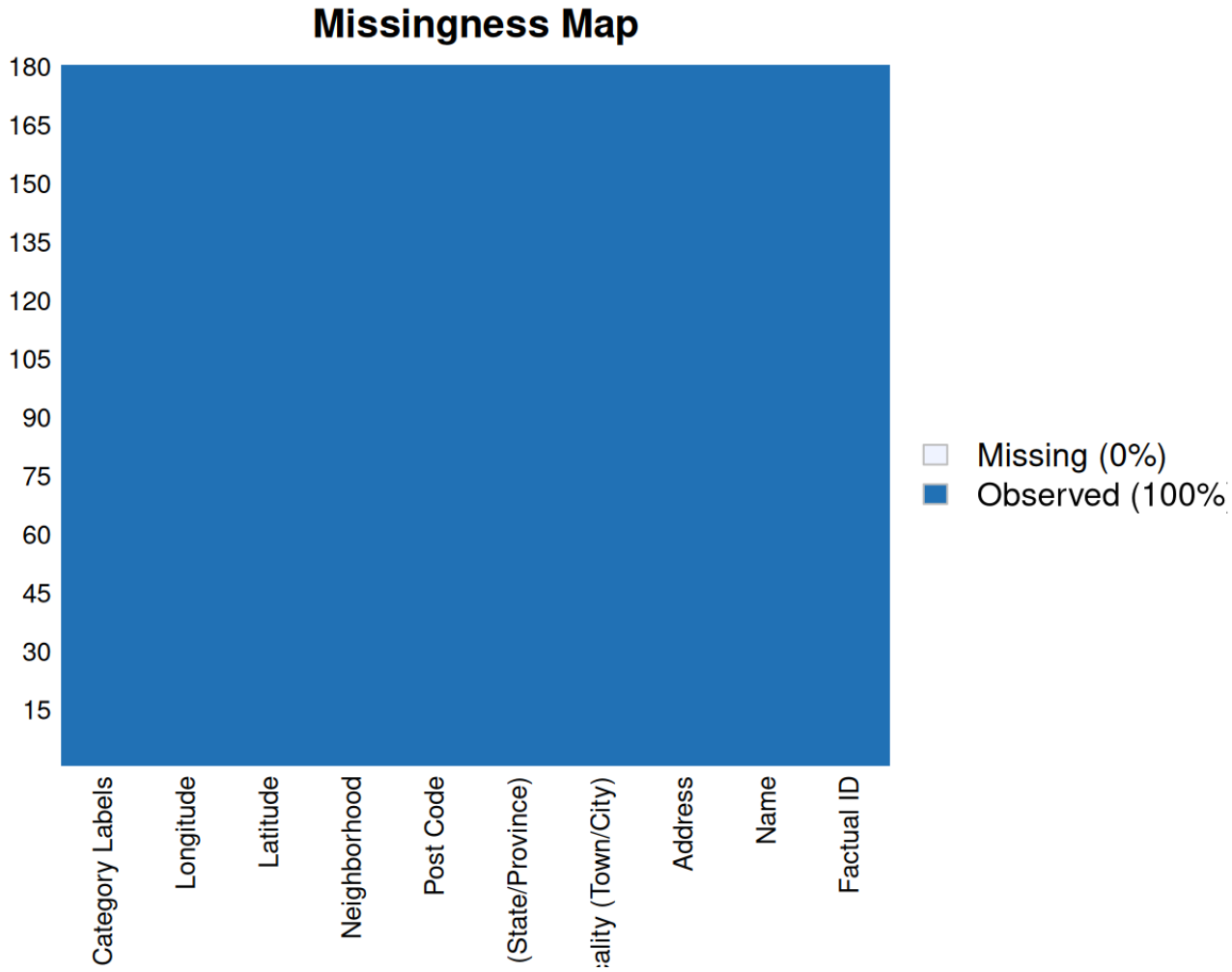

```
missmap(bike_stations)
```



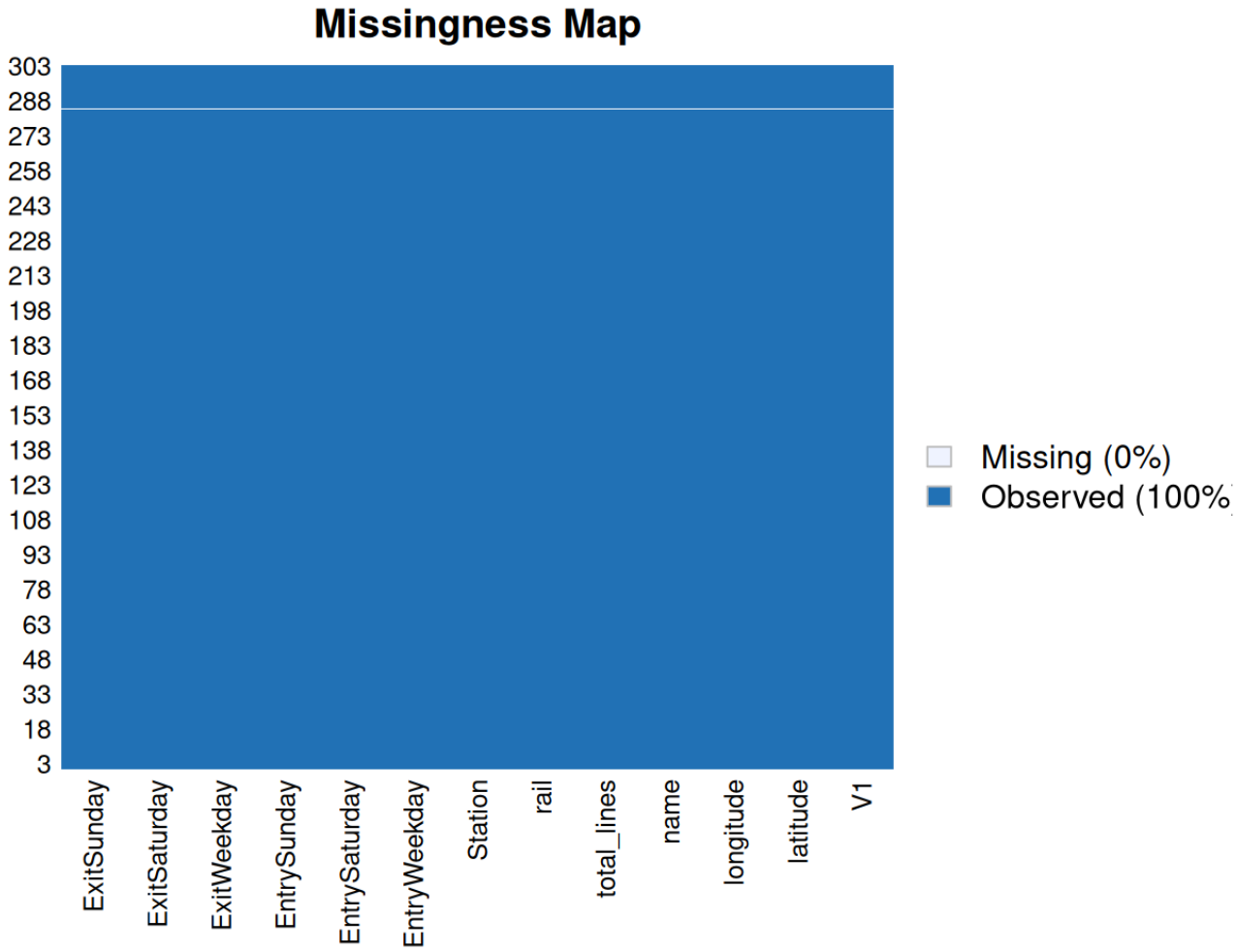

```
missmap(census)
```




```
missmap(pois)
```




```
missmap(tfl_stations)
```



Checking consistency between bike_journeys and bike_stations.
We have to join this datasets based on Start_Station_ID and StationID so we need to check whether they contain the same values.

```
length(unique(bike_journeys$Start_Station_ID))
```

```
## [1] 779
```

```
length(unique(bike_journeys$End_Station_ID))
```

```
## [1] 779
```

```
length(unique(bike_stations$Station_ID))
```

```
## [1] 773
```

```
length(unique(intersect(bike_stations$Station_ID, bike_journeys$Start_Station_ID)))
```

```
## [1] 771
```

Bike_journeys dataset contains 779 unique stations (the same number for end stations and start stations).
Bike_stations dataset contains 773 unique stations.
Both datasets have 771 matching unique stations which means that we will exclude data for 8 stations.
As the number of observations excluded is low and the process of cleaning the data is time consuming, we decided to leave this step for now.

Attempt 1

Hypothesis

H1. Bikes demand have an hourly trend.
H2. Bikes demand have a daily trend.
H3. Higher demand of bikes rented at stations which are close to central London.
H4. Higher demand of bikes rented where is high employment rate.
H5. Higher demand of bikes rented where is high population density.
H6. Higher demand of bikes rented where is high percentage of green space.
H7. Higher demand of bikes rented in deprived areas.
H8. Higher demand of bikes rented in poor areas.
H9. Higher demand of bikes rented where is high immigration rate.
H10. Higher demand of bikes rented where is high flats rate.
H11. Higher demand of bikes rented where is low number of owned properties rate.

Metrics

- Start_hour. Indicate the hour when the journey started. Linked to H1.
- DayOfWeek. Indicate the day when journey started. Linked to H2.
- DistanceFromCenter. Distance from a point in central London to each station. Linked to H3.
- EmployeeRate. Ratio of people who are employed. NoEmployee over PopDen times AreaSqKm. Linked to H4.
- PopDen. Population divided by the ward area. Linked to H5.
- GrenSpace. Percentage of green space associated with the ward. Linked to H6.
- LivingEnSc. Quality of the local environment. The less deprived the area, the lower the score. Linked to H7.
- IncomeScor. Proportion of the population experiencing deprivation relating to low income. Higher score means lower income and poorer areas. Linked to H8.
- MedHPrice. Median house price. The lower median means the poorer areas. Linked to H8.
- NoCTFtoHRate. Ratio of properties in council tax band F-H (the highest median house price). Lower score means the poorer areas. Linked H8.
- NotBornUKRate. Ratio of people who were not born in the UK. It is definied as NotBornUK over BornUK plus NotBornUK. Linked to H9.
- FlatsRate. Ratio of flats. It is definied as NoFlats over NoHouses. Linked to H10.
- NoOwndDwelRate. Ratio of owned properties in each ward. It is definied as NoOwndDwel over NoDwelling. Linked to H11.

Data processing

The data needs to be transformed from the format:

<Journey_Duration, Journey_ID, End_Date, End_Month, End_Year, End_Hour, End_Minute, End_Station_ID, Start_Date, Start_Month, Start_Year, Start_Hour, Start_Minute, Start_Station_ID>
<Station_ID, Capacity, Latitude, Longitude, Station_Name>
<WardCode, WardName, Borough, NESW, AreaSqKm, lon, lat, IncomeScor, LivingEnSc, NoEmployee, GrenSpace, PopDen, BornUK, NotBornUK, NoCTFtoH, NoDwelling, NoFlats, NoHouses, NoWndDwel, MedHPrice>

Into the format:

<Demand, Start_hour, DayOfWeek, DistanceFromCenter, EmployeeRate, PopDen, GrenSpace, LivingEnSc, IncomeScor, MedHPrice, NoCTFtoHRate, NotBornUKRate, FlatsRate, NoOwndDwelRate>

Implementing R code to tranform the datasets

The code below selects only columns needed for our task from bike journeys dataset.. It merges three columns (`Start_Year` , `Start_Month` , `Start_Date`) into a date format. Then it counts number of rented bikes for each station, each date and hour and saves it in `Demand` column. Then it computes `DayOfWeek` from `Date` column.

```
bike_journeys_attempt_1 = bike_journeys[, .(Start_Station_ID, Date = format(as.Date
(paste(Start_Year, Start_Month, Start_Date, sep='-'), '%y-%m-%d'), '%Y-%m-%d'), Sta
rt_Hour)]
bike_journeys_attempt_1 = bike_journeys_attempt_1[, .(Demand=sum(.N)), by=.(Start_S
tation_ID, Date, Start_Hour)]
bike_journeys_attempt_1$DayOfWeek = as.POSIXlt(bike_journeys_attempt_1$Date)$wday
str(bike_journeys_attempt_1)
```

```
## Classes 'data.table' and 'data.frame':   464968 obs. of  5 variables:
##  $ Start_Station_ID: int  251 550 212 163 36 589 478 153 396 732 ...
##  $ Date            : chr  "2017-09-19" "2017-09-19" "2017-09-15" "2017-09-14" ..
.
##  $ Start_Hour      : int  17 14 16 12 19 14 17 13 15 8 ...
##  $ Demand          : int  31 1 2 4 2 3 5 8 4 30 ...
##  $ DayOfWeek       : int  2 2 5 4 3 4 0 5 5 1 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

The code below takes only needed columns from bike stations dataset.
The code below computes distances between London coordinates and each station.

```
library(geosphere)
bike_stations_attempt_1 = bike_stations[, .(Station_ID, Latitude, Longitude)]
London_coordinates = c(-0.127800, 51.507400)
bike_stations_attempt_1$DistanceFromCenter = distm(bike_stations_attempt_1[,c('Long
itude','Latitude')], London_coordinates, fun=distGeo)
str(bike_stations_attempt_1)
```

```
## Classes 'data.table' and 'data.frame':   773 obs. of  4 variables:
##  $ Station_ID        : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Latitude          : num  51.5 51.5 51.5 51.5 51.5 ...
##  $ Longitude         : num  -0.11 -0.1976 -0.0846 -0.121 -0.1569 ...
##  $ DistanceFromCenter: num  2719 4922 3373 2565 2568 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

The code below selects only needed columns and computes defined metrics for census dataset.

```
census_attempt_1 = census[, .(lon, lat, EmployeeRate=NoEmployee/(PopDen*AreaSqKm),
PopDen, GrenSpace, IncomeScor, LivingEnSc, MedHPrice, NoCTFtoHRate=NoCTFtoH/NoDwell
ing, NotBornUKRate=NotBornUK/(BornUK+NotBornUK), FlatsRate = NoFlats/NoHouses, NoOw
ndDwelRate=NoOwndDwel/NoDwelling)]
str(census_attempt_1)
```

```
## Classes 'data.table' and 'data.frame':   625 obs. of  12 variables:
##  $ lon           : num  0.0779 0.1483 0.119 0.14 0.1736 ...
##  $ lat           : num  51.5 51.5 51.6 51.6 51.6 ...
##  $ EmployeeRate  : num  0.6148 0.0766 0.0948 0.1683 0.3791 ...
##  $ PopDen        : num  9885 7464 8923 2971 3014 ...
##  $ GrenSpace     : num  19.6 22.4 3 56.4 51.1 18.1 20.3 17.1 38.4 30.3 ...
##  $ IncomeScor    : num  0.27 0.28 0.25 0.27 0.19 0.27 0.36 0.27 0.31 0.17 ...
##  $ LivingEnSc    : num  42.8 28 31.6 34.8 21.2 ...
##  $ MedHPrice     : int  177000 160000 170000 195000 191750 167250 145000 155000
155000 250000 ...
##  $ NoCTFtoHRate  : num  2.11e-05 2.47e-05 2.28e-05 9.88e-05 1.26e-04 ...
##  $ NotBornUKRate : num  0.573 0.247 0.301 0.248 0.19 ...
##  $ FlatsRate     : num  1.971 0.165 0.236 0.526 0.229 ...
##  $ NoOwndDwelRate: num  0.326 0.457 0.478 0.53 0.665 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

I want to assign station to a ward.
The code below computes distances between all stations and all wards. Then it takes index of a ward which has minimum distance to a station in each row. Next step is to map ward indexes to census data for each station. Then it binds bike stations data with census data. Then it merges bike journeys data with combined bike stations data and census data using station ID.

```
bike_station_to_ward_distance_matrix_attempt_1 = distm(bike_stations_attempt_1[,c('
Longitude','Latitude')], census_attempt_1[,c('lon','lat')], fun=distGeo)
data_attempt_1 = merge(bike_journeys_attempt_1, cbind(bike_stations_attempt_1, cens
us_attempt_1[apply(bike_station_to_ward_distance_matrix_attempt_1, 1, which.min)]),
by.x='Start_Station_ID', by.y='Station_ID')
str(data_attempt_1)
```

```
## Classes 'data.table' and 'data.frame':   461710 obs. of  20 variables:
##  $ Start_Station_ID  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Date              : chr  "2017-09-17" "2017-09-14" "2017-09-18" "2017-09-13"
...
##  $ Start_Hour        : int  12 7 6 6 6 9 8 8 19 19 ...
##  $ Demand            : int  2 4 1 1 4 7 8 10 1 3 ...
##  $ DayOfWeek         : int  0 4 1 3 2 5 5 2 0 3 ...
##  $ Latitude          : num  51.5 51.5 51.5 51.5 51.5 ...
##  $ Longitude         : num  -0.11 -0.11 -0.11 -0.11 -0.11 ...
##  $ DistanceFromCenter: num  2719 2719 2719 2719 2719 ...
##  $ lon               : num  -0.108 -0.108 -0.108 -0.108 -0.108 ...
##  $ lat               : num  51.5 51.5 51.5 51.5 51.5 ...
##  $ EmployeeRate      : num  3.82 3.82 3.82 3.82 3.82 ...
##  $ PopDen            : num  12778 12778 12778 12778 12778 ...
##  $ GrenSpace         : num  9.3 9.3 9.3 9.3 9.3 9.3 9.3 9.3 9.3 9.3 ...
##  $ IncomeScor        : num  0.21 0.21 0.21 0.21 0.21 0.21 0.21 0.21 0.21 0.21 ..
.
##  $ LivingEnSc        : num  51 51 51 51 51 ...
##  $ MedHPrice         : int  455000 455000 455000 455000 455000 455000 455000 455
000 455000 455000 ...
##  $ NoCTFtoHRate      : num  0.00424 0.00424 0.00424 0.00424 0.00424 ...
##  $ NotBornUKRate     : num  0.385 0.385 0.385 0.385 0.385 ...
##  $ FlatsRate         : num  9.48 9.48 9.48 9.48 9.48 ...
##  $ NoOwndDwelRate    : num  0.276 0.276 0.276 0.276 0.276 ...
##  - attr(*, ".internal.selfref")=<externalptr>
##  - attr(*, "sorted")= chr "Start_Station_ID"
```

I'm creating binary values for `Start_Hour` and `DayOfWeek`. **Unfortunately, `dummy` function returns path instead of generated column names when knitting. I couldn't find solution, so we worked around it.**

```
#library(dummies)
#start_hour_binary = dummy(data_attempt_1$Start_Hour, sep='_', drop = TRUE)
#day_of_week_binary = dummy(data_attempt_1$DayOfWeek, sep='_', drop = TRUE)
#data_attempt_1 = cbind(data_attempt_1, day_of_week_binary)
data_attempt_1$Start_Hour_0 = ifelse(data_attempt_1$Start_Hour == 0, 1, 0)
data_attempt_1$Start_Hour_1 = ifelse(data_attempt_1$Start_Hour == 1, 1, 0)
data_attempt_1$Start_Hour_2 = ifelse(data_attempt_1$Start_Hour == 2, 1, 0)
data_attempt_1$Start_Hour_3 = ifelse(data_attempt_1$Start_Hour == 3, 1, 0)
data_attempt_1$Start_Hour_4 = ifelse(data_attempt_1$Start_Hour == 4, 1, 0)
data_attempt_1$Start_Hour_5 = ifelse(data_attempt_1$Start_Hour == 5, 1, 0)
data_attempt_1$Start_Hour_6 = ifelse(data_attempt_1$Start_Hour == 6, 1, 0)
data_attempt_1$Start_Hour_7 = ifelse(data_attempt_1$Start_Hour == 7, 1, 0)
data_attempt_1$Start_Hour_8 = ifelse(data_attempt_1$Start_Hour == 8, 1, 0)
data_attempt_1$Start_Hour_9 = ifelse(data_attempt_1$Start_Hour == 9, 1, 0)
data_attempt_1$Start_Hour_10 = ifelse(data_attempt_1$Start_Hour == 10, 1, 0)
data_attempt_1$Start_Hour_11 = ifelse(data_attempt_1$Start_Hour == 11, 1, 0)
data_attempt_1$Start_Hour_12 = ifelse(data_attempt_1$Start_Hour == 12, 1, 0)
data_attempt_1$Start_Hour_13 = ifelse(data_attempt_1$Start_Hour == 13, 1, 0)
data_attempt_1$Start_Hour_14 = ifelse(data_attempt_1$Start_Hour == 14, 1, 0)
data_attempt_1$Start_Hour_15 = ifelse(data_attempt_1$Start_Hour == 15, 1, 0)
data_attempt_1$Start_Hour_16 = ifelse(data_attempt_1$Start_Hour == 16, 1, 0)
data_attempt_1$Start_Hour_17 = ifelse(data_attempt_1$Start_Hour == 17, 1, 0)
data_attempt_1$Start_Hour_18 = ifelse(data_attempt_1$Start_Hour == 18, 1, 0)
data_attempt_1$Start_Hour_19 = ifelse(data_attempt_1$Start_Hour == 19, 1, 0)
data_attempt_1$Start_Hour_20 = ifelse(data_attempt_1$Start_Hour == 20, 1, 0)
data_attempt_1$Start_Hour_21 = ifelse(data_attempt_1$Start_Hour == 21, 1, 0)
data_attempt_1$Start_Hour_22 = ifelse(data_attempt_1$Start_Hour == 22, 1, 0)
data_attempt_1$Start_Hour_23 = ifelse(data_attempt_1$Start_Hour == 23, 1, 0)

data_attempt_1$DayOfWeek_0 = ifelse(data_attempt_1$DayOfWeek == 0, 1, 0)
data_attempt_1$DayOfWeek_1 = ifelse(data_attempt_1$DayOfWeek == 1, 1, 0)
data_attempt_1$DayOfWeek_2 = ifelse(data_attempt_1$DayOfWeek == 2, 1, 0)
data_attempt_1$DayOfWeek_3 = ifelse(data_attempt_1$DayOfWeek == 3, 1, 0)
data_attempt_1$DayOfWeek_4 = ifelse(data_attempt_1$DayOfWeek == 4, 1, 0)
data_attempt_1$DayOfWeek_5 = ifelse(data_attempt_1$DayOfWeek == 5, 1, 0)
data_attempt_1$DayOfWeek_6 = ifelse(data_attempt_1$DayOfWeek == 6, 1, 0)
str(data_attempt_1)
```

```
## Classes 'data.table' and 'data.frame':   461710 obs. of  51 variables:
##  $ Start_Station_ID  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Date              : chr  "2017-09-17" "2017-09-14" "2017-09-18" "2017-09-13"
...
##  $ Start_Hour        : int  12 7 6 6 6 9 8 8 19 19 ...
##  $ Demand            : int  2 4 1 1 4 7 8 10 1 3 ...
##  $ DayOfWeek         : int  0 4 1 3 2 5 5 2 0 3 ...
##  $ Latitude          : num  51.5 51.5 51.5 51.5 51.5 ...
##  $ Longitude         : num  -0.11 -0.11 -0.11 -0.11 -0.11 ...
##  $ DistanceFromCenter: num  2719 2719 2719 2719 2719 ...
##  $ lon               : num  -0.108 -0.108 -0.108 -0.108 -0.108 ...
##  $ lat               : num  51.5 51.5 51.5 51.5 51.5 ...
##  $ EmployeeRate      : num  3.82 3.82 3.82 3.82 3.82 ...
##  $ PopDen            : num  12778 12778 12778 12778 12778 ...
##  $ GrenSpace         : num  9.3 9.3 9.3 9.3 9.3 9.3 9.3 9.3 9.3 9.3 ...
```

```
##  $ IncomeScor        : num  0.21 0.21 0.21 0.21 0.21 0.21 0.21 0.21 0.21 0.21 ..
.
##  $ LivingEnSc        : num  51 51 51 51 51 ...
##  $ MedHPrice         : int  455000 455000 455000 455000 455000 455000 455000 455
000 455000 455000 ...
##  $ NoCTFtoHRate      : num  0.00424 0.00424 0.00424 0.00424 0.00424 ...
##  $ NotBornUKRate     : num  0.385 0.385 0.385 0.385 0.385 ...
##  $ FlatsRate         : num  9.48 9.48 9.48 9.48 9.48 ...
##  $ NoOwndDwelRate    : num  0.276 0.276 0.276 0.276 0.276 ...
##  $ Start_Hour_0      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_1      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_2      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_3      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_4      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_5      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_6      : num  0 0 1 1 1 0 0 0 0 0 ...
##  $ Start_Hour_7      : num  0 1 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_8      : num  0 0 0 0 0 0 1 1 0 0 ...
##  $ Start_Hour_9      : num  0 0 0 0 0 1 0 0 0 0 ...
##  $ Start_Hour_10     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_11     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_12     : num  1 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_13     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_14     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_15     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_16     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_17     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_18     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_19     : num  0 0 0 0 0 0 0 0 1 1 ...
##  $ Start_Hour_20     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_21     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_22     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_23     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ DayOfWeek_0       : num  1 0 0 0 0 0 0 0 1 0 ...
##  $ DayOfWeek_1       : num  0 0 1 0 0 0 0 0 0 0 ...
##  $ DayOfWeek_2       : num  0 0 0 0 1 0 0 1 0 0 ...
##  $ DayOfWeek_3       : num  0 0 0 1 0 0 0 0 0 1 ...
##  $ DayOfWeek_4       : num  0 1 0 0 0 0 0 0 0 0 ...
##  $ DayOfWeek_5       : num  0 0 0 0 0 1 1 0 0 0 ...
##  $ DayOfWeek_6       : num  0 0 0 0 0 0 0 0 0 0 ...
##  - attr(*, ".internal.selfref")=<externalptr>
##  - attr(*, "sorted")= chr "Start_Station_ID"
```

I clear the data.

```
data_attempt_1$lon = NULL
data_attempt_1$lat = NULL
data_attempt_1$Date = NULL
data_attempt_1$Longitude = NULL
data_attempt_1$Latitude = NULL
data_attempt_1$Start_Station_ID = NULL
data_attempt_1$Start_Hour = NULL
data_attempt_1$DayOfWeek = NULL
str(data_attempt_1)
```

```
## Classes 'data.table' and 'data.frame':   461710 obs. of  43 variables:
##  $ Demand            : int  2 4 1 1 4 7 8 10 1 3 ...
##  $ DistanceFromCenter: num  2719 2719 2719 2719 2719 ...
##  $ EmployeeRate      : num  3.82 3.82 3.82 3.82 3.82 ...
##  $ PopDen            : num  12778 12778 12778 12778 12778 ...
##  $ GrenSpace         : num  9.3 9.3 9.3 9.3 9.3 9.3 9.3 9.3 9.3 9.3 ...
##  $ IncomeScor        : num  0.21 0.21 0.21 0.21 0.21 0.21 0.21 0.21 0.21 0.21 ..
.
##  $ LivingEnSc        : num  51 51 51 51 51 ...
##  $ MedHPrice         : int  455000 455000 455000 455000 455000 455000 455000 455
000 455000 455000 ...
##  $ NoCTFtoHRate      : num  0.00424 0.00424 0.00424 0.00424 0.00424 ...
##  $ NotBornUKRate     : num  0.385 0.385 0.385 0.385 0.385 ...
##  $ FlatsRate         : num  9.48 9.48 9.48 9.48 9.48 ...
##  $ NoOwndDwelRate    : num  0.276 0.276 0.276 0.276 0.276 ...
##  $ Start_Hour_0      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_1      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_2      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_3      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_4      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_5      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_6      : num  0 0 1 1 1 0 0 0 0 0 ...
##  $ Start_Hour_7      : num  0 1 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_8      : num  0 0 0 0 0 0 1 1 0 0 ...
##  $ Start_Hour_9      : num  0 0 0 0 0 1 0 0 0 0 ...
##  $ Start_Hour_10     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_11     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_12     : num  1 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_13     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_14     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_15     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_16     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_17     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_18     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_19     : num  0 0 0 0 0 0 0 0 1 1 ...
##  $ Start_Hour_20     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_21     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_22     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Start_Hour_23     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ DayOfWeek_0       : num  1 0 0 0 0 0 0 0 1 0 ...
##  $ DayOfWeek_1       : num  0 0 1 0 0 0 0 0 0 0 ...
##  $ DayOfWeek_2       : num  0 0 0 0 1 0 0 1 0 0 ...
##  $ DayOfWeek_3       : num  0 0 0 1 0 0 0 0 0 1 ...
##  $ DayOfWeek_4       : num  0 1 0 0 0 0 0 0 0 0 ...
##  $ DayOfWeek_5       : num  0 0 0 0 0 1 1 0 0 0 ...
##  $ DayOfWeek_6       : num  0 0 0 0 0 0 0 0 0 0 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Summary of the final dataset.

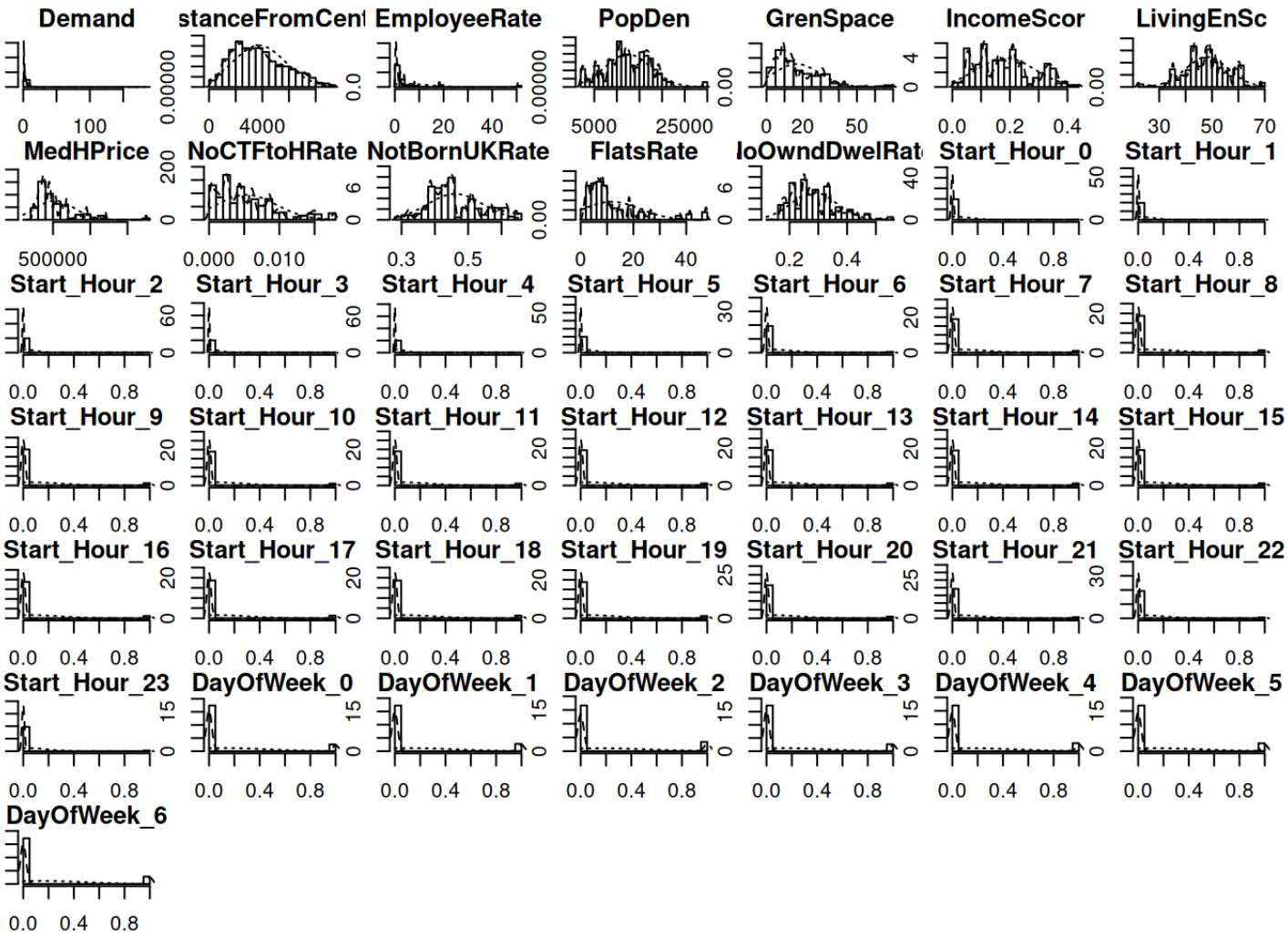
```
summary(data_attempt_1)
```

```
##      Demand         DistanceFromCenter  EmployeeRate         PopDen
##  Min.   :  1.000   Min.   : 146.2     Min.   : 0.1321   Min.   : 2312
##  1st Qu.:  1.000   1st Qu.:2184.4     1st Qu.: 0.4967   1st Qu.: 9107
##  Median :  2.000   Median :3497.8     Median : 1.1067   Median :12357
##  Mean   :  3.314   Mean   :3756.1     Mean   : 4.2202   Mean   :12437
##  3rd Qu.:  4.000   3rd Qu.:5092.2     3rd Qu.: 3.7458   3rd Qu.:16333
##  Max.   :182.000   Max.   :9354.3     Max.   :50.5540   Max.   :29375
##    GrenSpace        IncomeScor       LivingEnSc      MedHPrice
##  Min.   : 0.00   Min.   :0.0100   Min.   :22.05   Min.   : 188000
##  1st Qu.: 7.60   1st Qu.:0.1100   1st Qu.:42.97   1st Qu.: 356250
##  Median :13.50   Median :0.1800   Median :48.05   Median : 445000
##  Mean   :16.69   Mean   :0.1859   Mean   :48.10   Mean   : 525908
##  3rd Qu.:24.20   3rd Qu.:0.2400   3rd Qu.:53.14   3rd Qu.: 620000
##  Max.   :69.10   Max.   :0.4400   Max.   :68.06   Max.   :1750000
##   NoCTFtoHRate        NotBornUKRate      FlatsRate      NoOwndDwelRate
##  Min.   :2.661e-05   Min.   :0.2888   Min.   : 1.185   Min.   :0.1380
##  1st Qu.:2.094e-03   1st Qu.:0.4001   1st Qu.: 5.045   1st Qu.:0.2195
##  Median :4.280e-03   Median :0.4441   Median : 8.135   Median :0.2715
##  Mean   :5.311e-03   Mean   :0.4608   Mean   :11.775   Mean   :0.2847
##  3rd Qu.:7.803e-03   3rd Qu.:0.5118   3rd Qu.:14.990   3rd Qu.:0.3352
##  Max.   :1.794e-02   Max.   :0.6457   Max.   :47.509   Max.   :0.5476
##   Start_Hour_0      Start_Hour_1      Start_Hour_2       Start_Hour_3
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.000000   Min.   :0.000000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.000000   1st Qu.:0.000000
##  Median :0.00000   Median :0.00000   Median :0.000000   Median :0.000000
##  Mean   :0.01833   Mean   :0.01251   Mean   :0.008672   Mean   :0.006106
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.000000   3rd Qu.:0.000000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.000000   Max.   :1.000000
##   Start_Hour_4       Start_Hour_5      Start_Hour_6      Start_Hour_7
##  Min.   :0.000000   Min.   :0.00000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.000000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.0000
##  Median :0.000000   Median :0.00000   Median :0.00000   Median :0.0000
##  Mean   :0.005739   Mean   :0.01063   Mean   :0.03146   Mean   :0.0511
##  3rd Qu.:0.000000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.0000
##  Max.   :1.000000   Max.   :1.00000   Max.   :1.00000   Max.   :1.0000
##   Start_Hour_8      Start_Hour_9     Start_Hour_10     Start_Hour_11
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.0000
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :0.0000
##  Mean   :0.06207   Mean   :0.05986   Mean   :0.05257   Mean   :0.0534
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.0000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.0000
##  Start_Hour_12     Start_Hour_13     Start_Hour_14     Start_Hour_15
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000
##  Mean   :0.05663   Mean   :0.05704   Mean   :0.05568   Mean   :0.05701
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
##  Start_Hour_16     Start_Hour_17     Start_Hour_18    Start_Hour_19
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.00000
##  Median :0.00000   Median :0.00000   Median :0.0000   Median :0.00000
```

```
##  Mean   :0.06073   Mean   :0.06671   Mean   :0.0667   Mean   :0.06023
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.0000   Max.   :1.00000
##  Start_Hour_20     Start_Hour_21     Start_Hour_22    Start_Hour_23
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.00000
##  Median :0.00000   Median :0.00000   Median :0.0000   Median :0.00000
##  Mean   :0.04944   Mean   :0.03844   Mean   :0.0331   Mean   :0.02584
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.0000   Max.   :1.00000
##   DayOfWeek_0       DayOfWeek_1       DayOfWeek_2       DayOfWeek_3
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.0000   Median :0.0000   Median :0.0000   Median :0.0000
##  Mean   :0.1283   Mean   :0.1372   Mean   :0.1678   Mean   :0.1322
##  3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##   DayOfWeek_4      DayOfWeek_5       DayOfWeek_6
##  Min.   :0.000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.000   Median :0.0000   Median :0.0000
##  Mean   :0.149   Mean   :0.1469   Mean   :0.1385
##  3rd Qu.:0.000   3rd Qu.:0.0000   3rd Qu.:0.0000
##  Max.   :1.000   Max.   :1.0000   Max.   :1.0000
```

Plotting distribution of each column in the final dataset.

```
library(psych)
multi.hist(data_attempt_1)
```



Few variables are not normally distributed, such as `Demand` and `EmployeeRate` and it can be transformed to log value.

```
data_attempt_1$Demand = log10(data_attempt_1$Demand + min(data_attempt_1[Demand!=0]
$Demand))
data_attempt_1$EmployeeRate = log10(data_attempt_1$EmployeeRate + min(data_attempt_
1[EmployeeRate!=0]$EmployeeRate))
```

Summary of the dataset.

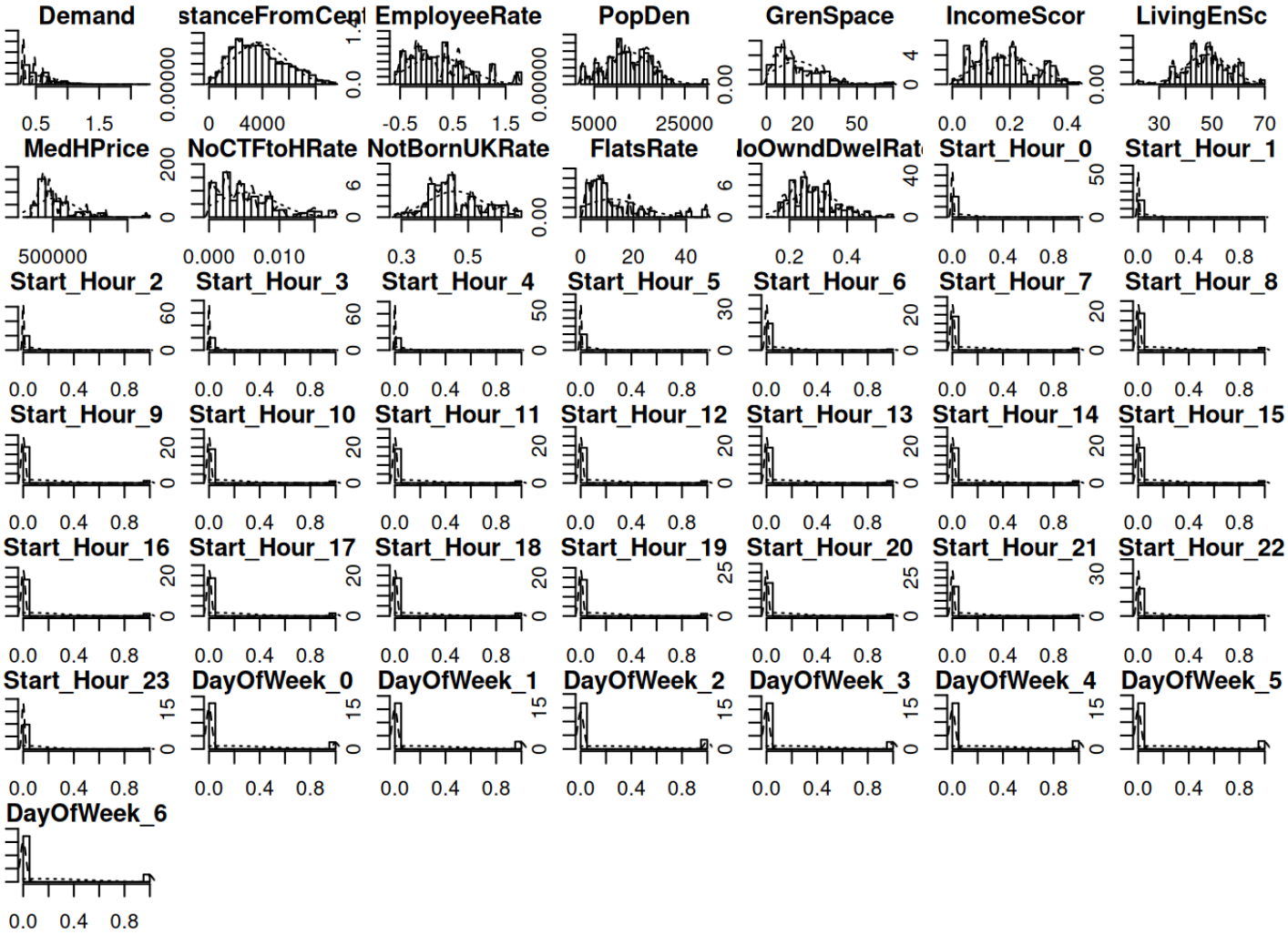
```
summary(data_attempt_1)
```

```
##      Demand        DistanceFromCenter  EmployeeRate          PopDen
##  Min.   :0.3010   Min.   : 146.2     Min.   :-0.57796   Min.   : 2312
##  1st Qu.:0.3010   1st Qu.:2184.4     1st Qu.:-0.20144   1st Qu.: 9107
##  Median :0.4771   Median :3497.8     Median : 0.09302   Median :12357
##  Mean   :0.5447   Mean   :3756.1     Mean   : 0.20536   Mean   :12437
##  3rd Qu.:0.6990   3rd Qu.:5092.2     3rd Qu.: 0.58860   3rd Qu.:16333
##  Max.   :2.2625   Max.   :9354.3     Max.   : 1.70489   Max.   :29375
##    GrenSpace       IncomeScor       LivingEnSc      MedHPrice
##  Min.   : 0.00   Min.   :0.0100   Min.   :22.05   Min.   : 188000
##  1st Qu.: 7.60   1st Qu.:0.1100   1st Qu.:42.97   1st Qu.: 356250
##  Median :13.50   Median :0.1800   Median :48.05   Median : 445000
##  Mean   :16.69   Mean   :0.1859   Mean   :48.10   Mean   : 525908
##  3rd Qu.:24.20   3rd Qu.:0.2400   3rd Qu.:53.14   3rd Qu.: 620000
```

```
##  Max.   :69.10   Max.   :0.4400   Max.   :68.06   Max.   :1750000
##   NoCTFtoHRate       NotBornUKRate      FlatsRate      NoOwndDwelRate
##  Min.   :2.661e-05   Min.   :0.2888   Min.   : 1.185   Min.   :0.1380
##  1st Qu.:2.094e-03   1st Qu.:0.4001   1st Qu.: 5.045   1st Qu.:0.2195
##  Median :4.280e-03   Median :0.4441   Median : 8.135   Median :0.2715
##  Mean   :5.311e-03   Mean   :0.4608   Mean   :11.775   Mean   :0.2847
##  3rd Qu.:7.803e-03   3rd Qu.:0.5118   3rd Qu.:14.990   3rd Qu.:0.3352
##  Max.   :1.794e-02   Max.   :0.6457   Max.   :47.509   Max.   :0.5476
##   Start_Hour_0      Start_Hour_1      Start_Hour_2       Start_Hour_3
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.000000   Min.   :0.000000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.000000   1st Qu.:0.000000
##  Median :0.00000   Median :0.00000   Median :0.000000   Median :0.000000
##  Mean   :0.01833   Mean   :0.01251   Mean   :0.008672   Mean   :0.006106
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.000000   3rd Qu.:0.000000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.000000   Max.   :1.000000
##   Start_Hour_4       Start_Hour_5      Start_Hour_6      Start_Hour_7
##  Min.   :0.000000   Min.   :0.00000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.000000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.0000
##  Median :0.000000   Median :0.00000   Median :0.00000   Median :0.0000
##  Mean   :0.005739   Mean   :0.01063   Mean   :0.03146   Mean   :0.0511
##  3rd Qu.:0.000000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.0000
##  Max.   :1.000000   Max.   :1.00000   Max.   :1.00000   Max.   :1.0000
##   Start_Hour_8      Start_Hour_9     Start_Hour_10     Start_Hour_11
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.0000
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :0.0000
##  Mean   :0.06207   Mean   :0.05986   Mean   :0.05257   Mean   :0.0534
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.0000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.0000
##  Start_Hour_12     Start_Hour_13     Start_Hour_14     Start_Hour_15
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000
##  Mean   :0.05663   Mean   :0.05704   Mean   :0.05568   Mean   :0.05701
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
##  Start_Hour_16     Start_Hour_17     Start_Hour_18    Start_Hour_19
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.00000
##  Median :0.00000   Median :0.00000   Median :0.0000   Median :0.00000
##  Mean   :0.06073   Mean   :0.06671   Mean   :0.0667   Mean   :0.06023
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.0000   Max.   :1.00000
##  Start_Hour_20     Start_Hour_21     Start_Hour_22    Start_Hour_23
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.00000
##  Median :0.00000   Median :0.00000   Median :0.0000   Median :0.00000
##  Mean   :0.04944   Mean   :0.03844   Mean   :0.0331   Mean   :0.02584
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.0000   Max.   :1.00000
##   DayOfWeek_0       DayOfWeek_1       DayOfWeek_2       DayOfWeek_3
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.0000   Median :0.0000   Median :0.0000   Median :0.0000
```

```
##  Mean   :0.1283   Mean   :0.1372   Mean   :0.1678   Mean   :0.1322
##  3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##   DayOfWeek_4      DayOfWeek_5      DayOfWeek_6
##  Min.   :0.000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.000   Median :0.0000   Median :0.0000
##  Mean   :0.149   Mean   :0.1469   Mean   :0.1385
##  3rd Qu.:0.000   3rd Qu.:0.0000   3rd Qu.:0.0000
##  Max.   :1.000   Max.   :1.0000   Max.   :1.0000
```

```
multi.hist(data_attempt_1)
```



Standarising the data.

```
data_std_attempt_1 = as.data.table(scale(data_attempt_1))
summary(data_std_attempt_1)
```

```
##      Demand         DistanceFromCenter  EmployeeRate         PopDen
##  Min.   :-0.9703   Min.   :-1.8292    Min.   :-1.4414   Min.   :-1.99415
##  1st Qu.:-0.9703   1st Qu.:-0.7964    1st Qu.:-0.7486   1st Qu.:-0.65585
##  Median :-0.2691   Median :-0.1309    Median :-0.2067   Median :-0.01571
##  Mean   : 0.0000   Mean   : 0.0000    Mean   : 0.0000   Mean   : 0.00000
##  3rd Qu.: 0.6143   3rd Qu.: 0.6770    3rd Qu.: 0.7052   3rd Qu.: 0.76746
##  Max.   : 6.8399   Max.   : 2.8367    Max.   : 2.7593   Max.   : 3.33621
```

```
##    GrenSpace          IncomeScor         LivingEnSc         MedHPrice
##  Min.   :-1.3260   Min.   :-1.74613   Min.   :-3.21034   Min.   :-1.2880
##  1st Qu.:-0.7222   1st Qu.:-0.75331   1st Qu.:-0.63211   1st Qu.:-0.6467
##  Median :-0.2533   Median :-0.05833   Median :-0.00604   Median :-0.3084
##  Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.0000
##  3rd Qu.: 0.5969   3rd Qu.: 0.53736   3rd Qu.: 0.62126   3rd Qu.: 0.3587
##  Max.   : 4.1647   Max.   : 2.52300   Max.   : 2.46004   Max.   : 4.6660
##   NoCTFtoHRate      NotBornUKRate       FlatsRate        NoOwndDwelRate
##  Min.   :-1.2486   Min.   :-2.0909   Min.   :-0.9846   Min.   :-1.8782
##  1st Qu.:-0.7600   1st Qu.:-0.7379   1st Qu.:-0.6257   1st Qu.:-0.8353
##  Median :-0.2435   Median :-0.2037   Median :-0.3385   Median :-0.1701
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.5887   3rd Qu.: 0.6198   3rd Qu.: 0.2988   3rd Qu.: 0.6458
##  Max.   : 2.9834   Max.   : 2.2462   Max.   : 3.3222   Max.   : 3.3656
##   Start_Hour_0       Start_Hour_1       Start_Hour_2        Start_Hour_3
##  Min.   :-0.1366   Min.   :-0.1126   Min.   :-0.09353   Min.   :-0.07838
##  1st Qu.:-0.1366   1st Qu.:-0.1126   1st Qu.:-0.09353   1st Qu.:-0.07838
##  Median :-0.1366   Median :-0.1126   Median :-0.09353   Median :-0.07838
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.00000
##  3rd Qu.:-0.1366   3rd Qu.:-0.1126   3rd Qu.:-0.09353   3rd Qu.:-0.07838
##  Max.   : 7.3182   Max.   : 8.8838   Max.   :10.69168   Max.   :12.75871
##   Start_Hour_4       Start_Hour_5       Start_Hour_6       Start_Hour_7
##  Min.   :-0.07598   Min.   :-0.1037   Min.   :-0.1802   Min.   :-0.2321
##  1st Qu.:-0.07598   1st Qu.:-0.1037   1st Qu.:-0.1802   1st Qu.:-0.2321
##  Median :-0.07598   Median :-0.1037   Median :-0.1802   Median :-0.2321
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.:-0.07598   3rd Qu.:-0.1037   3rd Qu.:-0.1802   3rd Qu.:-0.2321
##  Max.   :13.16168   Max.   : 9.6474   Max.   : 5.5488   Max.   : 4.3094
##   Start_Hour_8       Start_Hour_9      Start_Hour_10      Start_Hour_11
##  Min.   :-0.2573   Min.   :-0.2523   Min.   :-0.2356   Min.   :-0.2375
##  1st Qu.:-0.2573   1st Qu.:-0.2523   1st Qu.:-0.2356   1st Qu.:-0.2375
##  Median :-0.2573   Median :-0.2523   Median :-0.2356   Median :-0.2375
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.:-0.2573   3rd Qu.:-0.2523   3rd Qu.:-0.2356   3rd Qu.:-0.2375
##  Max.   : 3.8872   Max.   : 3.9629   Max.   : 4.2452   Max.   : 4.2104
##  Start_Hour_12     Start_Hour_13     Start_Hour_14     Start_Hour_15
##  Min.   :-0.245   Min.   :-0.2459   Min.   :-0.2428   Min.   :-0.2459
##  1st Qu.:-0.245   1st Qu.:-0.2459   1st Qu.:-0.2428   1st Qu.:-0.2459
##  Median :-0.245   Median :-0.2459   Median :-0.2428   Median :-0.2459
##  Mean   : 0.000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.:-0.245   3rd Qu.:-0.2459   3rd Qu.:-0.2428   3rd Qu.:-0.2459
##  Max.   : 4.082   Max.   : 4.0661   Max.   : 4.1182   Max.   : 4.0670
##  Start_Hour_16      Start_Hour_17      Start_Hour_18      Start_Hour_19
##  Min.   :-0.2543   Min.   :-0.2673   Min.   :-0.2673   Min.   :-0.2532
##  1st Qu.:-0.2543   1st Qu.:-0.2673   1st Qu.:-0.2673   1st Qu.:-0.2532
##  Median :-0.2543   Median :-0.2673   Median :-0.2673   Median :-0.2532
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.:-0.2543   3rd Qu.:-0.2673   3rd Qu.:-0.2673   3rd Qu.:-0.2532
##  Max.   : 3.9327   Max.   : 3.7405   Max.   : 3.7407   Max.   : 3.9502
##  Start_Hour_20     Start_Hour_21     Start_Hour_22     Start_Hour_23
##  Min.   :-0.2281   Min.   :-0.200   Min.   :-0.185   Min.   :-0.1629
##  1st Qu.:-0.2281   1st Qu.:-0.200   1st Qu.:-0.185   1st Qu.:-0.1629
##  Median :-0.2281   Median :-0.200   Median :-0.185   Median :-0.1629
##  Mean   : 0.0000   Mean   : 0.000   Mean   : 0.000   Mean   : 0.0000
```

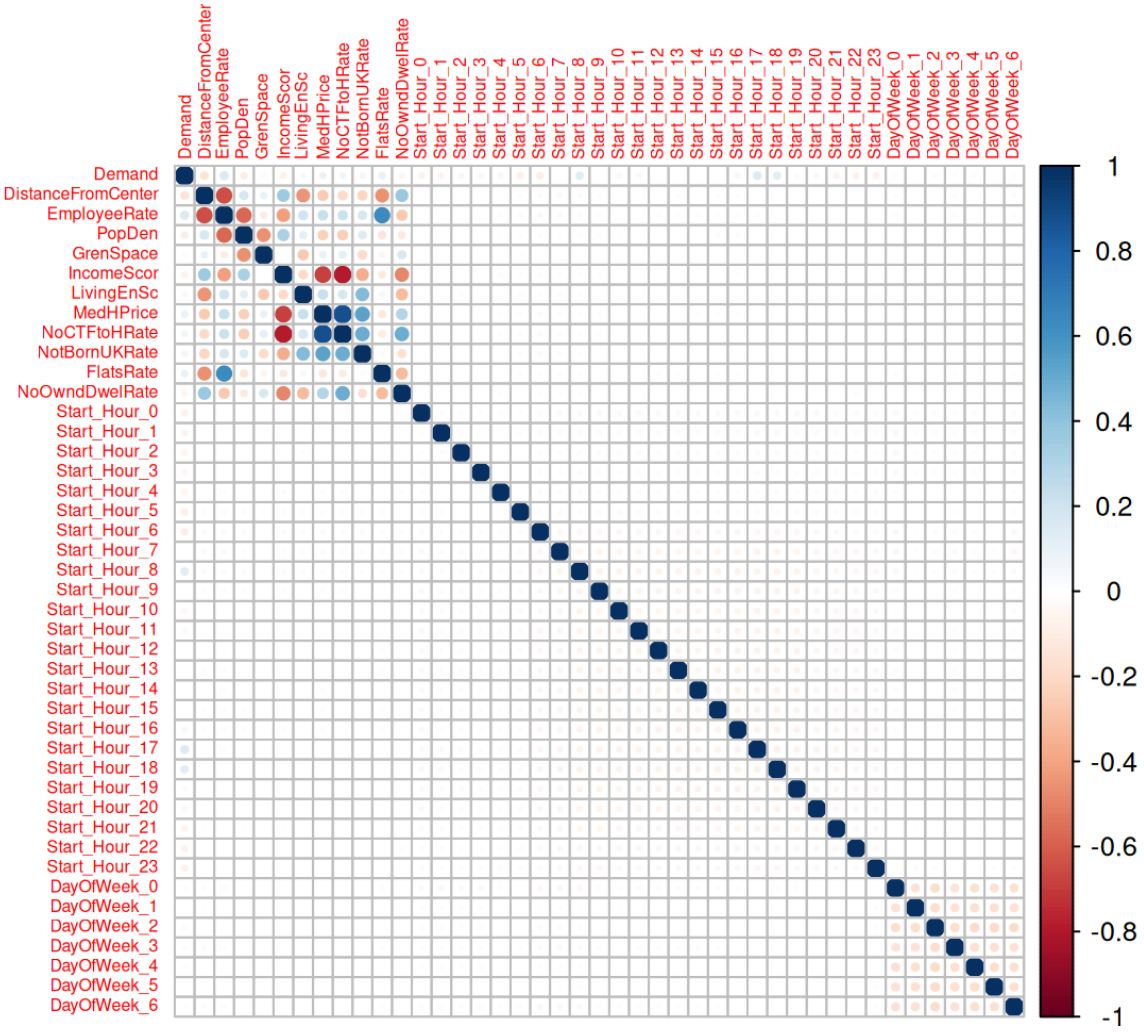
```
##  3rd Qu.:-0.2281   3rd Qu.:-0.200   3rd Qu.:-0.185   3rd Qu.:-0.1629
##  Max.   : 4.3846   Max.   : 5.001   Max.   : 5.405   Max.   : 6.1396
##   DayOfWeek_0       DayOfWeek_1       DayOfWeek_2       DayOfWeek_3
##  Min.   :-0.3837   Min.   :-0.3988   Min.   :-0.449   Min.   :-0.3904
##  1st Qu.:-0.3837   1st Qu.:-0.3988   1st Qu.:-0.449   1st Qu.:-0.3904
##  Median :-0.3837   Median :-0.3988   Median :-0.449   Median :-0.3904
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.000   Mean   : 0.0000
##  3rd Qu.:-0.3837   3rd Qu.:-0.3988   3rd Qu.:-0.449   3rd Qu.:-0.3904
##  Max.   : 2.6065   Max.   : 2.5072   Max.   : 2.227   Max.   : 2.5617
##   DayOfWeek_4       DayOfWeek_5       DayOfWeek_6
##  Min.   :-0.4184   Min.   :-0.415   Min.   :-0.401
##  1st Qu.:-0.4184   1st Qu.:-0.415   1st Qu.:-0.401
##  Median :-0.4184   Median :-0.415   Median :-0.401
##  Mean   : 0.0000   Mean   : 0.000   Mean   : 0.000
##  3rd Qu.:-0.4184   3rd Qu.:-0.415   3rd Qu.:-0.401
##  Max.   : 2.3901   Max.   : 2.409   Max.   : 2.494
```

Checking multicollinearity.

```
library(corrplot)
```

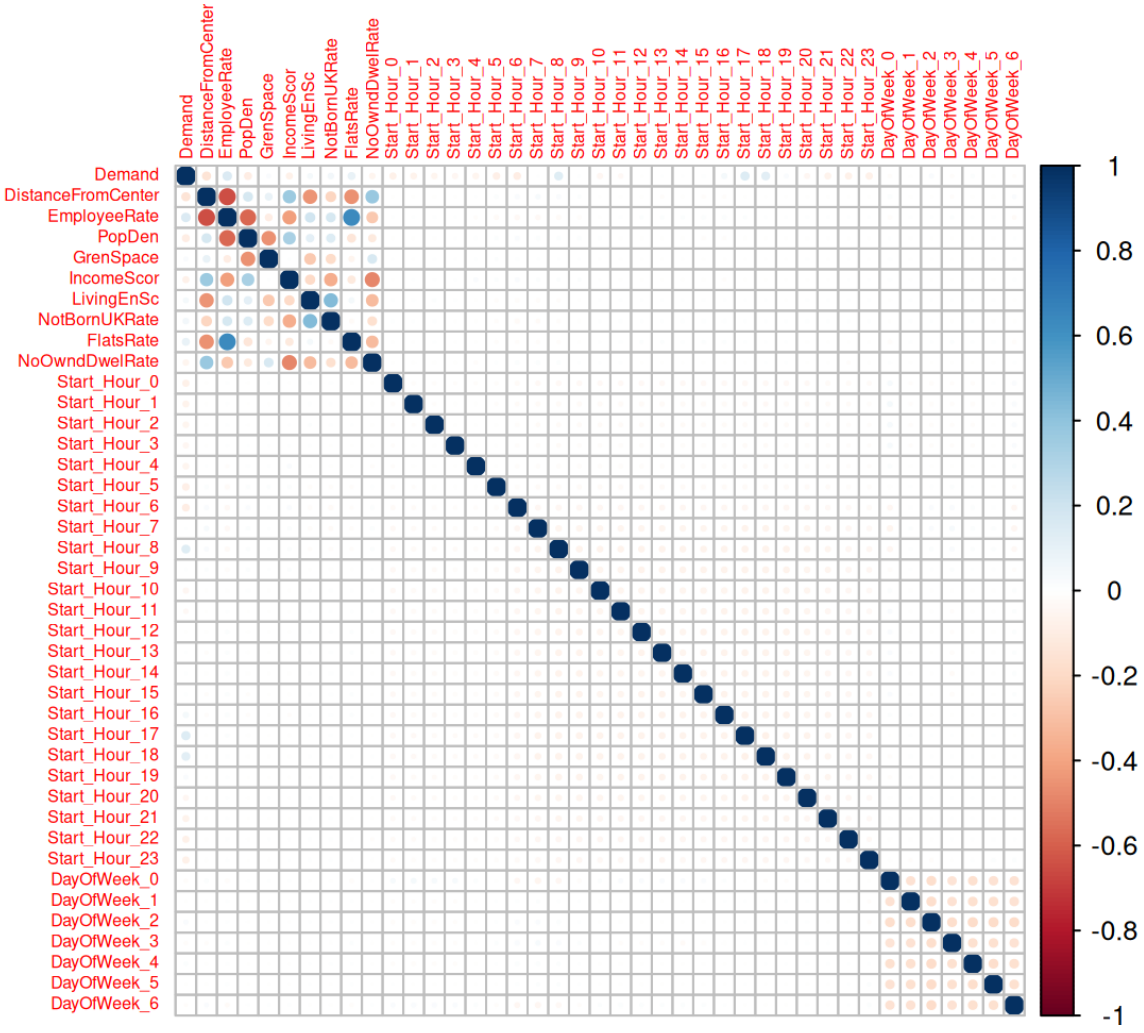
```
## corrplot 0.84 loaded
```

```
corrplot(cor(data_std_attempt_1), tl.cex=0.5)
```



There is high correlation between `NoCTFtoHRate` and `MedHPrice` so they will be romoved from the model.
Again checking multicollinearity.

```
data_std_attempt_1$NoCTFtoHRate = NULL
data_std_attempt_1$MedHPrice = NULL
corrplot(cor(data_std_attempt_1), tl.cex=0.5)
```



Algorithms

The goal requires to predict number of rented bikes so we have to implement regression model.

```
set.seed(0)
trainIdx_attempt_1 = sample(1:nrow(data_std_attempt_1), 0.75*nrow(data_std_attempt_
1))
train_attempt_1 = data_std_attempt_1[trainIdx_attempt_1]
test_attempt_1 = data_std_attempt_1[-trainIdx_attempt_1]
```

```
library(Metrics)
lr_attempt_1 = lm(Demand ~ ., data=train_attempt_1)
train_preds_attempt_1 = predict(lr_attempt_1, train_attempt_1)
```

```
## Warning in predict.lm(lr_attempt_1, train_attempt_1): prediction from a rank-
## deficient fit may be misleading
```

```
test_preds_attempt_1 = predict(lr_attempt_1, test_attempt_1)
```

```
## Warning in predict.lm(lr_attempt_1, test_attempt_1): prediction from a rank-
## deficient fit may be misleading
```

```
print(paste("R2 on train:", cor(train_preds_attempt_1, train_attempt_1$Demand)^2))
```

```
## [1] "R2 on train: 0.140141619250823"
```

```
print(paste("R2 on test:", cor(test_preds_attempt_1, test_attempt_1$Demand)^2))
```

```
## [1] "R2 on test: 0.137187607450111"
```

```
print(paste("RMSE on train:", rmse(train_preds_attempt_1, train_attempt_1$Demand)))
```

```
## [1] "RMSE on train: 0.92783218311877"
```

```
print(paste("RMSE on test:", rmse(test_preds_attempt_1, test_attempt_1$Demand)))
```

```
## [1] "RMSE on test: 0.927237613052377"
```

There's no difference between train and test scores which indicates that our model is stable. We do not overfit which means we don't need any regularisation.

Data understanding

```
lr_attempt_1 = lm(Demand ~ ., data=data_std_attempt_1)
summary(lr_attempt_1)
```

```
##
## Call:
## lm(formula = Demand ~ ., data = data_std_attempt_1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9413 -0.7108 -0.1480  0.5648  6.5400
##
## Coefficients: (2 not defined because of singularities)
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -4.936e-14  1.365e-03   0.000   1.0000
## DistanceFromCenter -9.980e-02  2.392e-03 -41.721  < 2e-16 ***
```

```
## EmployeeRate        9.226e-02  3.796e-03  24.306  < 2e-16 ***
## PopDen             -3.070e-02  2.926e-03 -10.494  < 2e-16 ***
## GrenSpace           3.218e-02  1.969e-03  16.341  < 2e-16 ***
## IncomeScor          5.132e-02  2.915e-03  17.607  < 2e-16 ***
## LivingEnSc         -1.332e-02  1.756e-03  -7.588 3.26e-14 ***
## NotBornUKRate       5.731e-02  1.964e-03  29.187  < 2e-16 ***
## FlatsRate           4.818e-03  2.193e-03   2.196   0.0281 *
## NoOwndDwelRate      2.585e-02  2.719e-03   9.504  < 2e-16 ***
## Start_Hour_0       -9.887e-03  1.770e-03  -5.587 2.31e-08 ***
## Start_Hour_1       -1.197e-02  1.655e-03  -7.237 4.60e-13 ***
## Start_Hour_2       -1.406e-02  1.573e-03  -8.940  < 2e-16 ***
## Start_Hour_3       -1.312e-02  1.515e-03  -8.659  < 2e-16 ***
## Start_Hour_4       -2.231e-02  1.506e-03 -14.814  < 2e-16 ***
## Start_Hour_5       -3.163e-02  1.615e-03 -19.586  < 2e-16 ***
## Start_Hour_6        2.696e-04  2.005e-03   0.134   0.8930
## Start_Hour_7        1.143e-01  2.300e-03  49.711  < 2e-16 ***
## Start_Hour_8        2.430e-01  2.442e-03  99.487  < 2e-16 ***
## Start_Hour_9        1.236e-01  2.412e-03  51.235  < 2e-16 ***
## Start_Hour_10       6.041e-02  2.316e-03  26.087  < 2e-16 ***
## Start_Hour_11       7.691e-02  2.327e-03  33.050  < 2e-16 ***
## Start_Hour_12       1.056e-01  2.370e-03  44.544  < 2e-16 ***
## Start_Hour_13       1.119e-01  2.375e-03  47.123  < 2e-16 ***
## Start_Hour_14       1.100e-01  2.358e-03  46.639  < 2e-16 ***
## Start_Hour_15       1.168e-01  2.375e-03  49.187  < 2e-16 ***
## Start_Hour_16       1.563e-01  2.423e-03  64.519  < 2e-16 ***
## Start_Hour_17       2.523e-01  2.497e-03 101.045  < 2e-16 ***
## Start_Hour_18       2.387e-01  2.497e-03  95.602  < 2e-16 ***
## Start_Hour_19       1.369e-01  2.417e-03  56.658  < 2e-16 ***
## Start_Hour_20       6.204e-02  2.273e-03  27.296  < 2e-16 ***
## Start_Hour_21       2.599e-02  2.112e-03  12.306  < 2e-16 ***
## Start_Hour_22       1.389e-02  2.028e-03   6.850 7.41e-12 ***
## Start_Hour_23              NA         NA      NA       NA
## DayOfWeek_0        -3.826e-03  1.770e-03  -2.161   0.0307 *
## DayOfWeek_1        -1.303e-02  1.797e-03  -7.251 4.15e-13 ***
## DayOfWeek_2         1.069e-02  1.864e-03   5.734 9.80e-09 ***
## DayOfWeek_3        -2.200e-02  1.791e-03 -12.285  < 2e-16 ***
## DayOfWeek_4         4.407e-03  1.824e-03   2.416   0.0157 *
## DayOfWeek_5        -1.158e-02  1.817e-03  -6.372 1.87e-10 ***
## DayOfWeek_6                NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9277 on 461671 degrees of freedom
## Multiple R-squared:  0.1394, Adjusted R-squared:  0.1394
## F-statistic:  1968 on 38 and 461671 DF,  p-value: < 2.2e-16
```

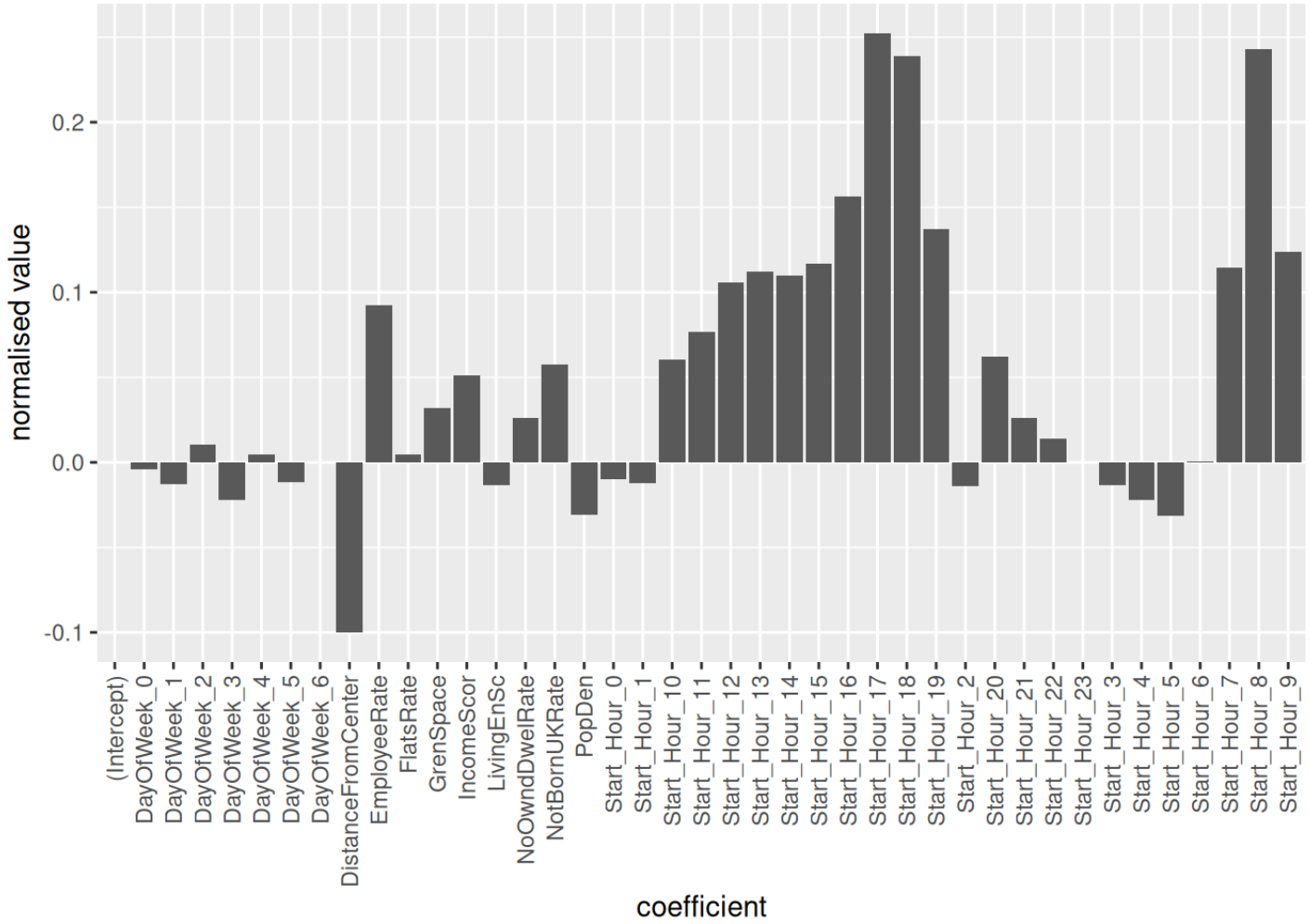
```
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'
```

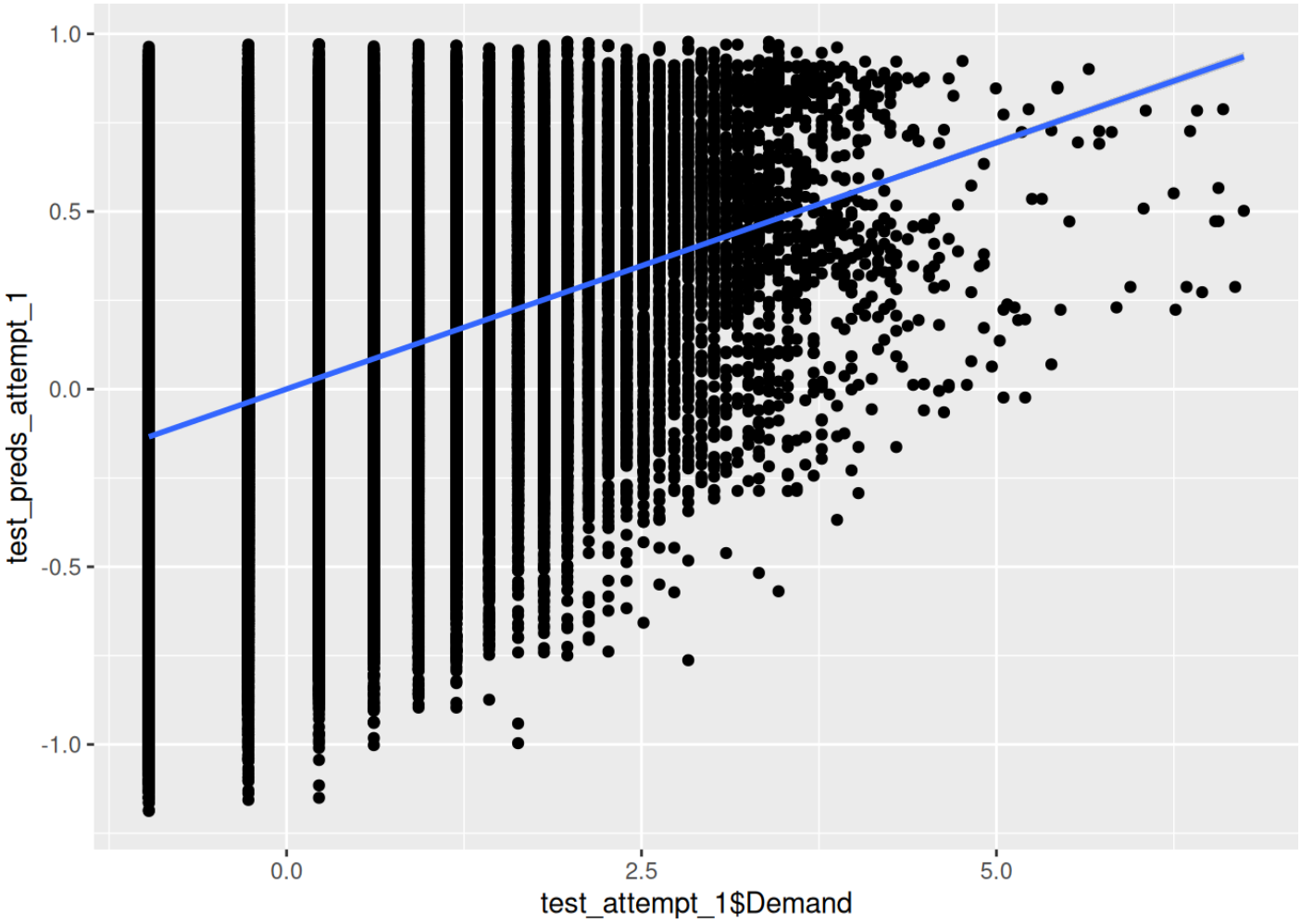
```
## The following objects are masked from 'package:psych':
##
##     %+%, alpha
```

```
ggplot(, aes(x = names(lr_attempt_1$coefficients), y=lr_attempt_1$coefficients)) +
  geom_bar(stat="identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5)) +
  xlab("coefficient") +
  ylab("normalised value")
```

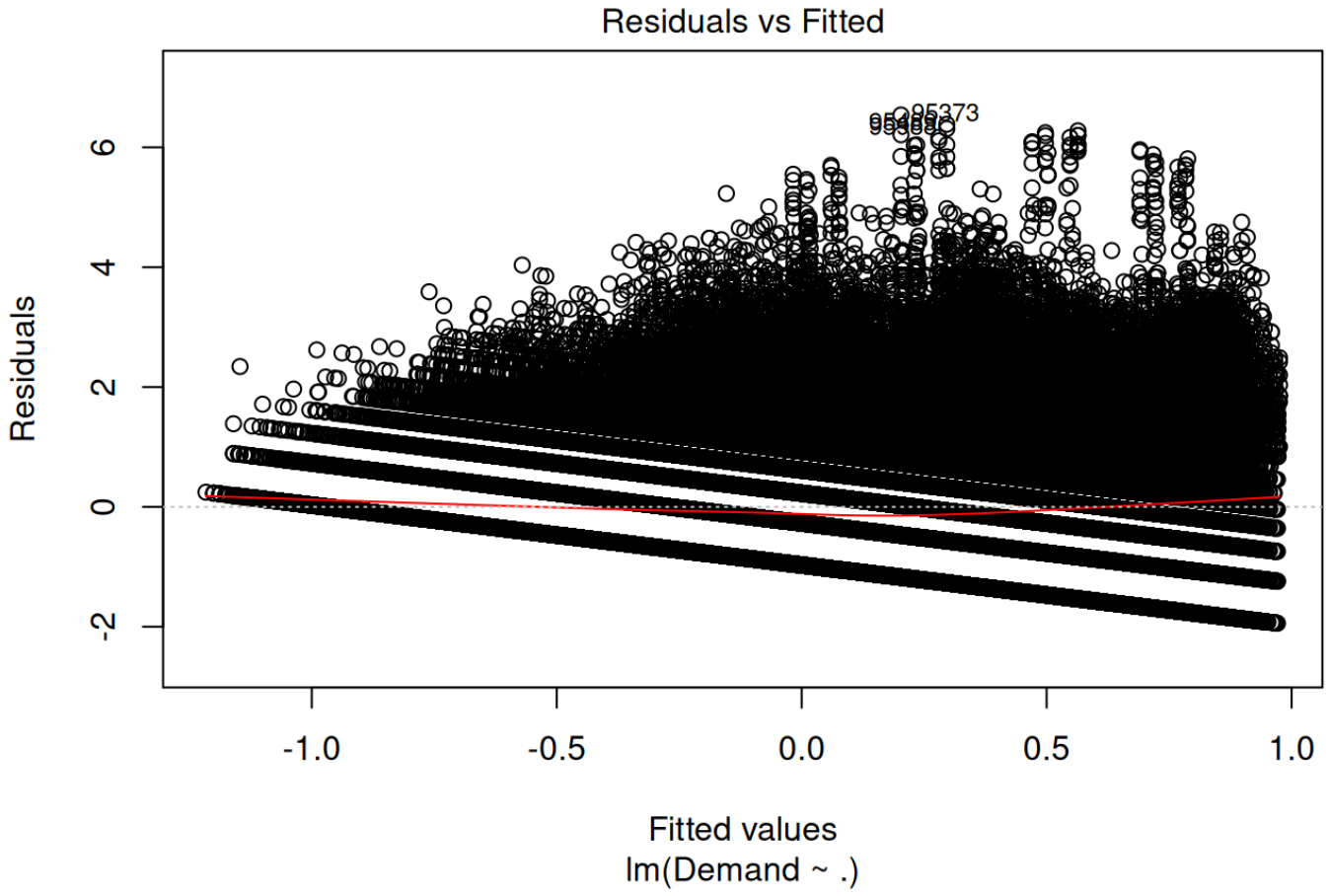
```
## Warning: Removed 2 rows containing missing values (position_stack).
```

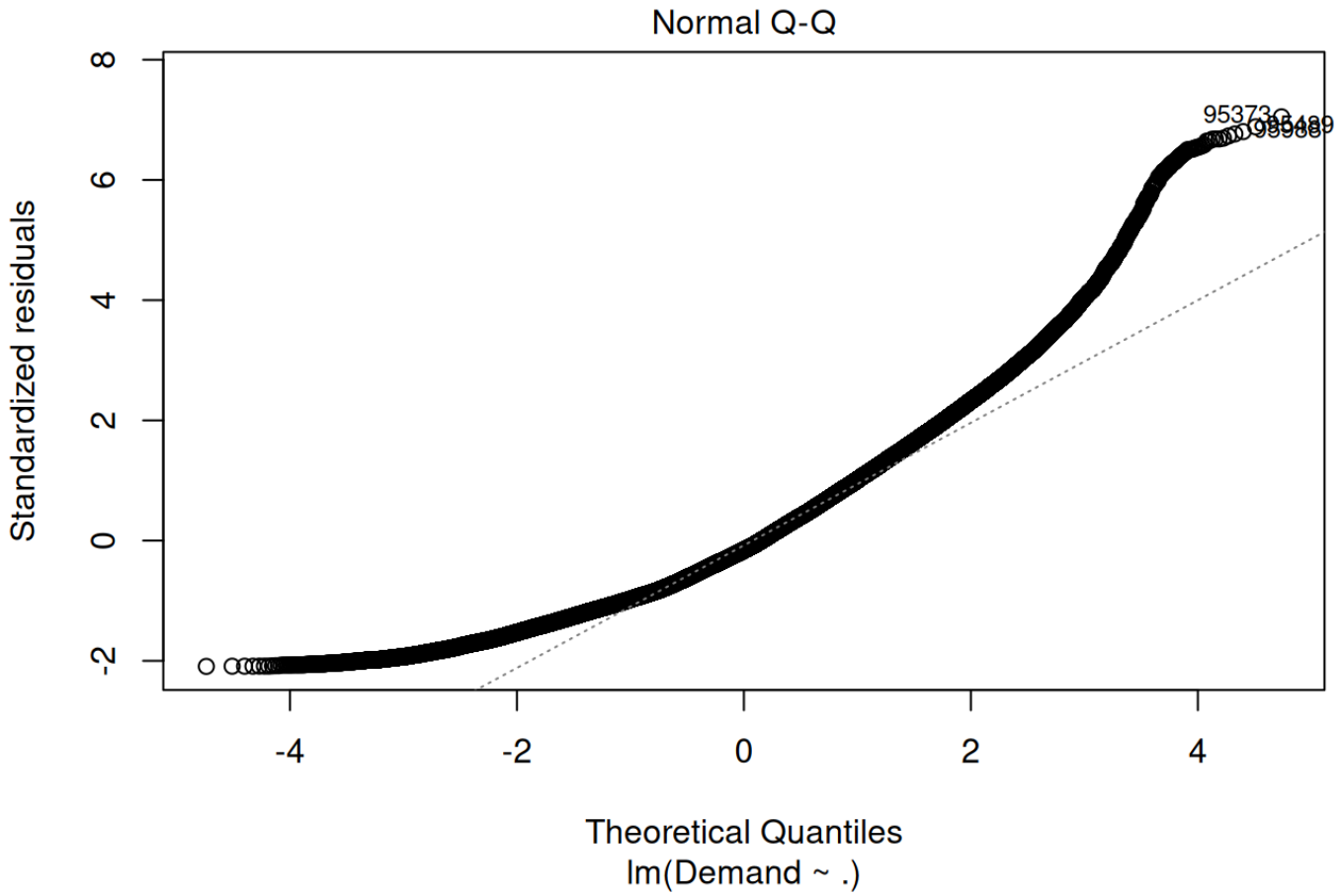


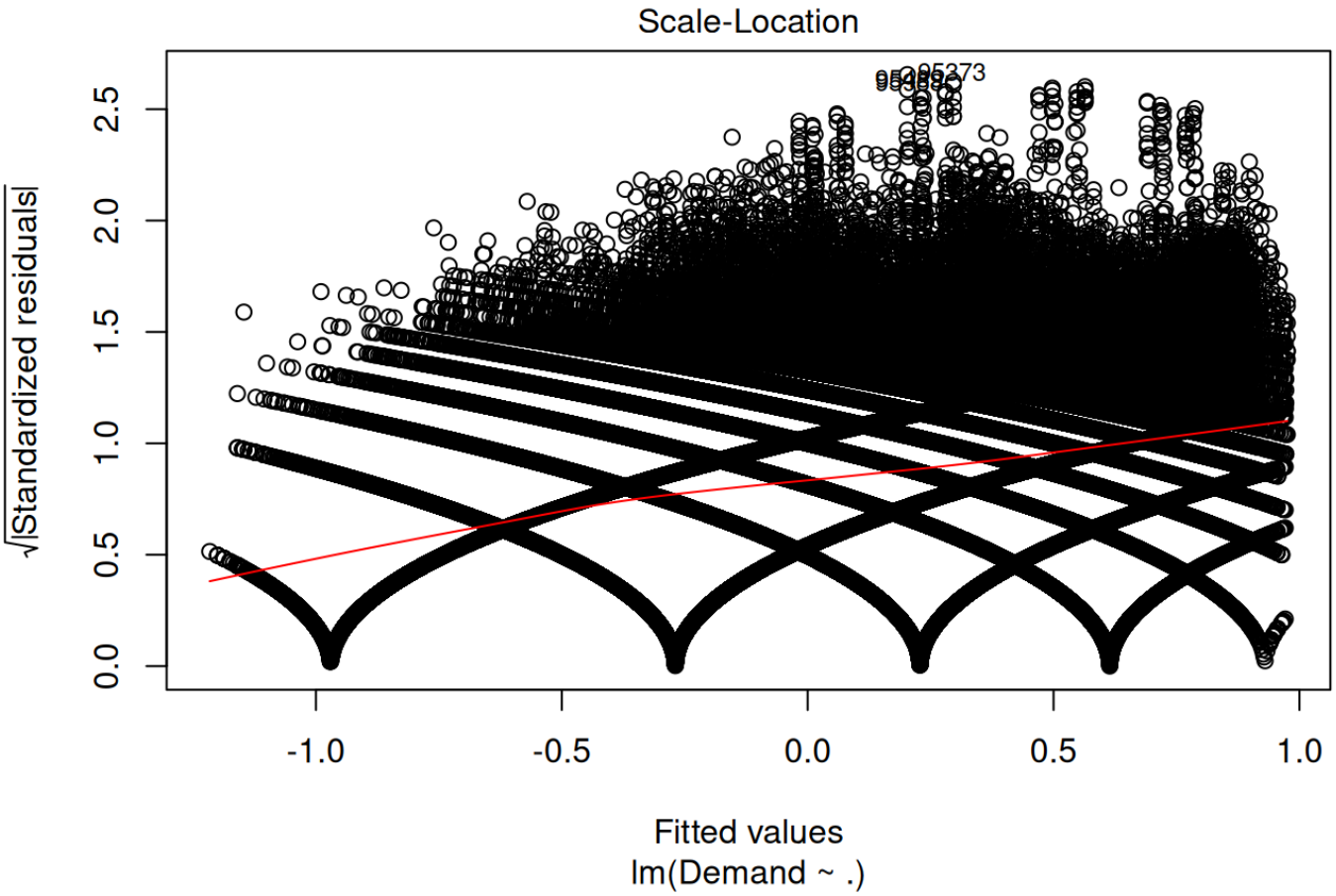

```
ggplot(test_attempt_1, aes(x = test_attempt_1$Demand, y=test_preds_attempt_1)) +
  geom_point() +
  geom_smooth(method = "lm")
```

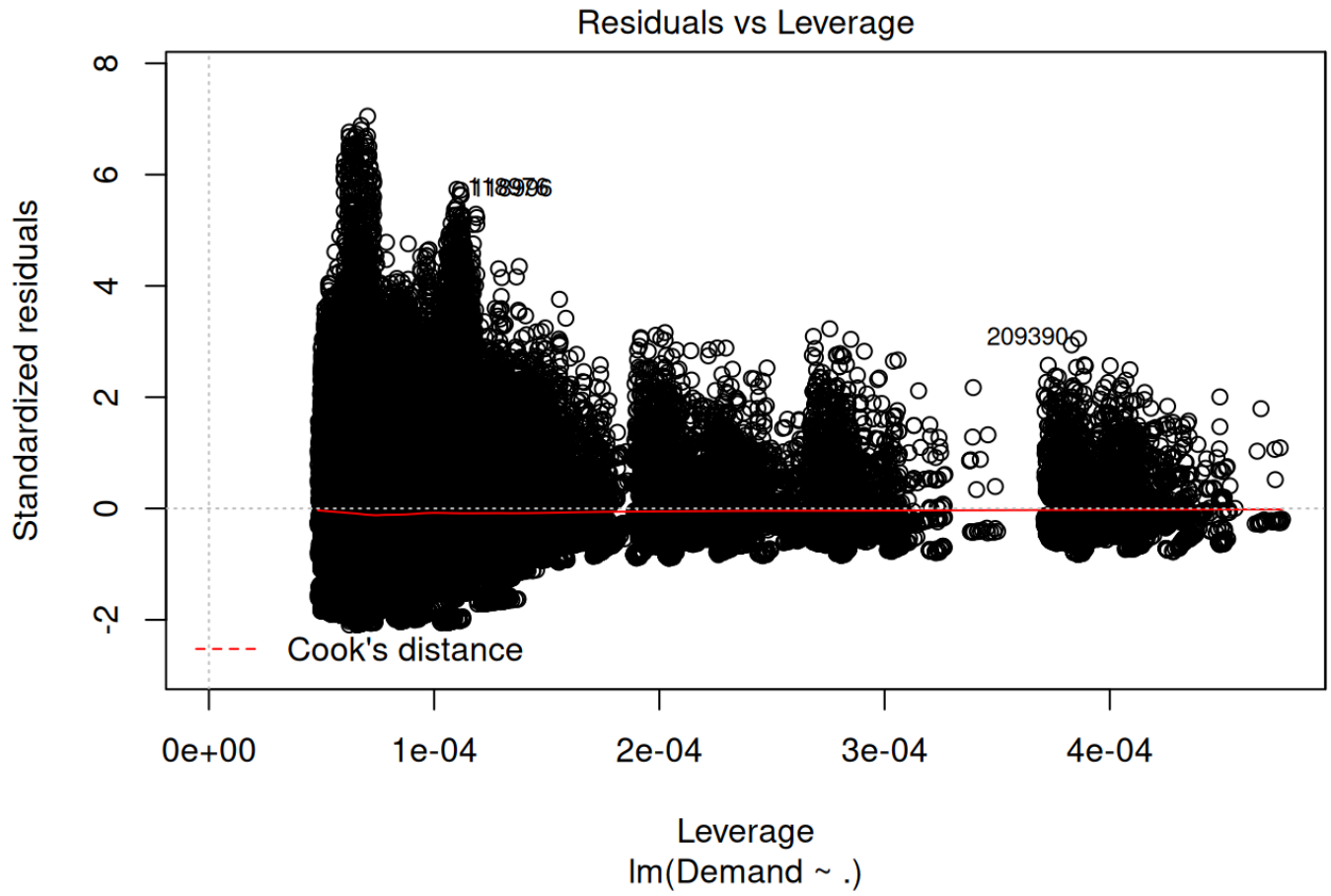



```
plot(lr_attempt_1)
```

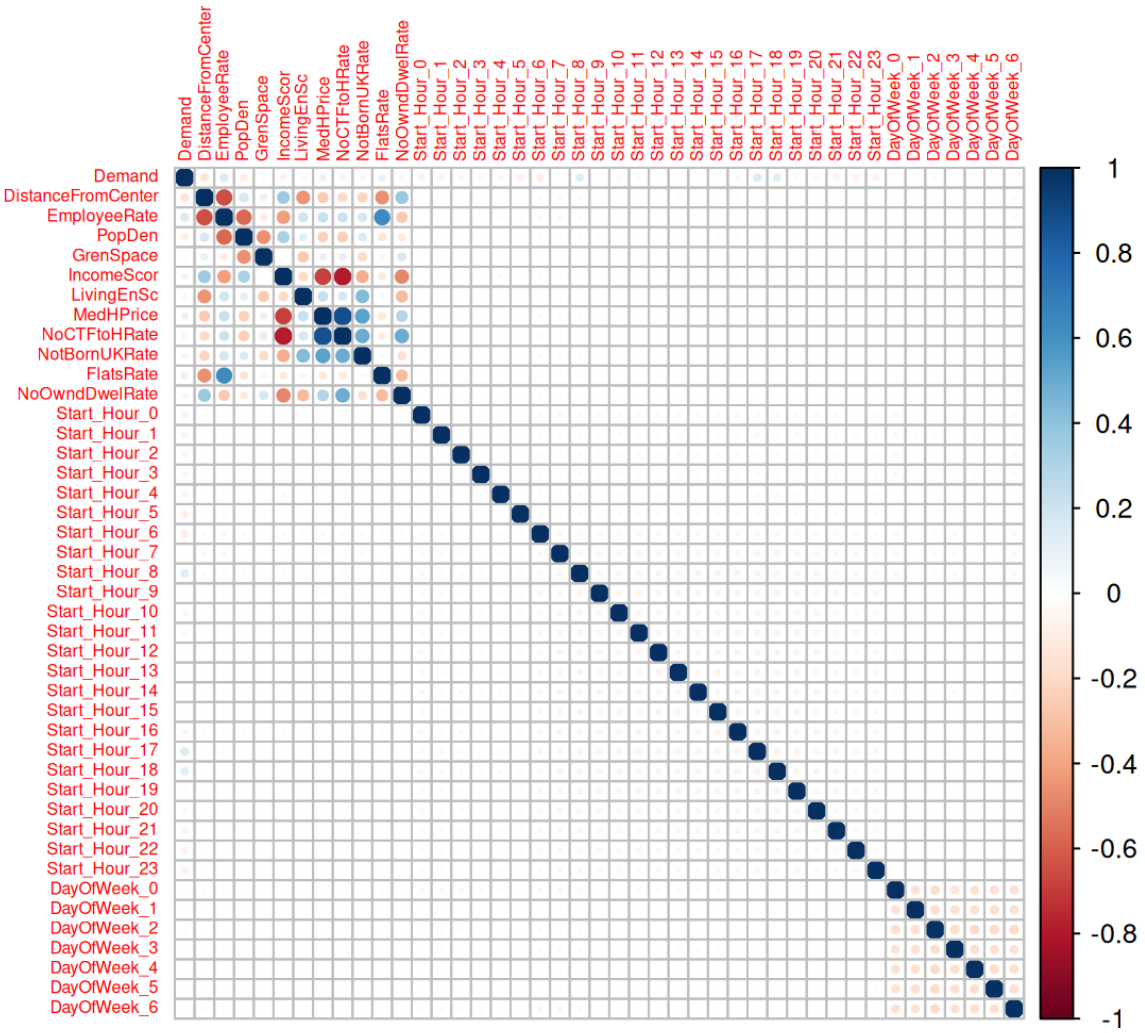

Residuals vs Fitted
Residuals
Fitted values
lm(Demand ~ .)


Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(Demand ~ .)


Scale-Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
-1.0
-0.5
0.0
0.5
1.0
Fitted values
lm(Demand ~ .)


Residuals vs Leverage
Standardized residuals
8
6
4
2
0
-2
209390
Cook's distance
0e+00
1e-04
2e-04
3e-04
4e-04
Leverage
lm(Demand ~ .)

```
corrplot(cor(data_attempt_1), tl.cex=0.5)
```



All visualisations above indicates: H1. Bikes demand have an hourly trend. *True*
H2. Bikes demand have a daily trend. *False*
H3. Higher demand of bikes rented at stations which are close to central London. *True*
H4. Higher demand of bikes rented where is high employment rate. *True*
H5. Higher demand of bikes rented where is high population density. *False*
H6. Higher demand of bikes rented where is high percentage of green space. *False*
H7. Higher demand of bikes rented in deprived areas. *False*
H8. Higher demand of bikes rented in poor areas. *False*
H9. Higher demand of bikes rented where is high immigration rate. *False*
H10. Higher demand of bikes rented where is high flats rate. *False*
H11. Higher demand of bikes rented where is low number of owned properties rate. *False*

Limitation

Almost 87% of the variation of the dependent variable is not described by independent variables. This means it is a very poor model.
I also couldn't use two metrics because in the model because of the multicollinearity.

Attempt 2

In the second attempt I remove H2 and H5-H11, change H1 and add H12.

Hypothesis

H1. Higher demand of bikes rented during peak hours 8 and 17 to 18.
~~H2. Bikes demand have a daily trend.~~
H3. Higher demand of bikes rented at stations which are close to central London.
H4. Higher demand of bikes rented where is high employment rate.
~~H5. Higher demand of bikes rented where is high population density.~~
~~H6. Higher demand of bikes rented where is high percentage of green space.~~
~~H7. Higher demand of bikes rented in deprived areas.~~
~~H8. Higher demand of bikes rented in poor areas.~~
~~H9. Higher demand of bikes rented where is high immigration rate.~~
~~H10. Higher demand of bikes rented where is high flats rate.~~
~~H11. Higher demand of bikes rented where is low number of owned properties rate.~~
H12. Bikes demand rented at each station depends on previous demand at the same station.

Metrics

- PeakHour Start_Hour binned into time slots containing peak hours 8 and 17-18. Linked to H1.
- DistanceFromCenter. Distance from a point in central London to each station. Linked to H3.
- EmployeeRate. Ratio of people who are employed. NoEmployee over PopDen times AreaSqKm. Linked to H4.
- PreviousWeekDemand. Previous week demand for each station and each hour. Linked to H12.

Data processing

The data needs to be transformed from the format:

<Journey_Duration, Journey_ID, End_Date, End_Month, End_Year, End_Hour, End_Minute, End_Station_ID, Start_Date, Start_Month, Start_Year, Start_Hour, Start_Minute, Start_Station_ID>
<Station_ID, Capacity, Latitude, Longitude, Station_Name>
<WardCode, WardName, Borough, NESW, AreaSqKm, lon, lat, IncomeScor, LivingEnSc, NoEmployee, GrenSpace, PopDen, BornUK, NotBornUK, NoCTFtoH, NoDwelling, NoFlats, NoHouses, NoWndDwel, MedHPrice>

Into the format:

<Demand, PeakHour, DistanceFromCenter, EmployeeRate, PreviousWeekDemand>

Implementing R code to tranform the datasets.

The code below selects only columns needed for our task from bike journeys dataset. It merges three columns (`Start_Year` , `Start_Month` , `Start_Date`) into a date format. Then it counts number of rented bikes for each station, each date and hour and saves it in `Demand` column. Then it computes `DayOfWeek` , `PeakHourWeekday` , `Weekend` from `Date` column.
I compute `PreviousWeekDemand` for each station, day of week and hour by grouping, ordering and shifting `Demand` column. As it creates rows without data I remove those from the dataset.

```
bike_journeys_attempt_2 = bike_journeys[, .(Start_Station_ID, Date = format(as.Date
(paste(Start_Year, Start_Month, Start_Date, sep='-'), '%y-%m-%d'), '%Y-%m-%d'), Sta
rt_Hour)]
bike_journeys_attempt_2 = bike_journeys_attempt_2[, .(Demand=sum(.N)), by=.(Start_S
tation_ID, Date, Start_Hour)]
bike_journeys_attempt_2$DayOfWeek = as.POSIXlt(bike_journeys_attempt_2$Date)$wday

bike_journeys_attempt_2$PeakHour = ifelse((bike_journeys_attempt_2$Start_Hour >= 8
& bike_journeys_attempt_2$Start_Hour <= 8) | (bike_journeys_attempt_2$Start_Hour >=
17 & bike_journeys_attempt_2$Start_Hour <= 18), 1, 0)

bike_journeys_attempt_2 = bike_journeys_attempt_2[order(-Date), .(Demand, Date, Pea
kHour, PreviousWeekDemand=shift(Demand, type="lead")), by=c('Start_Station_ID', 'Da
yOfWeek', 'Start_Hour')]
bike_journeys_attempt_2 = bike_journeys_attempt_2[complete.cases(bike_journeys_atte
mpt_2)]

str(bike_journeys_attempt_2)
```

```
## Classes 'data.table' and 'data.frame':   356389 obs. of  7 variables:
##  $ Start_Station_ID  : int  251 251 251 251 251 251 251 550 550 105 ...
##  $ DayOfWeek         : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ Start_Hour        : int  17 17 17 17 17 17 17 14 14 14 ...
##  $ Demand            : int  31 31 38 44 30 19 31 1 3 3 ...
##  $ Date              : chr  "2017-09-19" "2017-09-12" "2017-09-05" "2017-08-29"
...
##  $ PeakHour          : num  1 1 1 1 1 1 1 0 0 0 ...
##  $ PreviousWeekDemand: int  31 38 44 30 19 31 33 3 2 1 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

The code below takes only needed columns from bike stations dataset.
The code below computes distances between London coordinates and each station.

```
library(geosphere)
bike_stations_attempt_2 = bike_stations[, .(Station_ID, Latitude, Longitude)]
London_coordinates = c(-0.127800, 51.507400)
bike_stations_attempt_2$DistanceFromCenter = distm(bike_stations_attempt_2[,c('Long
itude','Latitude')], London_coordinates, fun=distGeo)
str(bike_stations_attempt_2)
```

```
## Classes 'data.table' and 'data.frame':   773 obs. of  4 variables:
##  $ Station_ID        : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Latitude          : num  51.5 51.5 51.5 51.5 51.5 ...
##  $ Longitude         : num  -0.11 -0.1976 -0.0846 -0.121 -0.1569 ...
##  $ DistanceFromCenter: num  2719 4922 3373 2565 2568 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

The code below selects only needed columns and computes defined metrics for census dataset.

```
census_attempt_2 = census[, .(lon, lat, EmployeeRate=NoEmployee/(PopDen*AreaSqKm))]
str(census_attempt_2)
```

```
## Classes 'data.table' and 'data.frame':   625 obs. of  3 variables:
##  $ lon         : num  0.0779 0.1483 0.119 0.14 0.1736 ...
##  $ lat         : num  51.5 51.5 51.6 51.6 51.6 ...
##  $ EmployeeRate: num  0.6148 0.0766 0.0948 0.1683 0.3791 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

I want to assign station to a ward.
The code below computes distances between all stations and all wards. Then it takes index of a ward which has minimum distance to a station in each row. Next step is to map ward indexes to census data for each station. Then it binds bike stations data with census data. Then it merges bike journeys data with combined bike stations data and census data using station ID.

```
bike_station_to_ward_distance_matrix_attempt_2 = distm(bike_stations_attempt_2[,c('
Longitude','Latitude')], census_attempt_2[,c('lon','lat')], fun=distGeo)
data_attempt_2 = merge(bike_journeys_attempt_2, cbind(bike_stations_attempt_2, cens
us_attempt_2[apply(bike_station_to_ward_distance_matrix_attempt_2, 1, which.min)]),
by.x='Start_Station_ID', by.y='Station_ID')
str(data_attempt_2)
```

```
## Classes 'data.table' and 'data.frame':   354077 obs. of  13 variables:
##  $ Start_Station_ID  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ DayOfWeek         : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ Start_Hour        : int  6 6 6 6 6 6 6 8 8 8 ...
##  $ Demand            : int  4 3 3 3 2 3 4 10 8 9 ...
##  $ Date              : chr  "2017-09-19" "2017-09-12" "2017-09-05" "2017-08-29"
...
##  $ PeakHour          : num  0 0 0 0 0 0 0 1 1 1 ...
##  $ PreviousWeekDemand: int  3 3 3 2 3 4 2 8 9 1 ...
##  $ Latitude          : num  51.5 51.5 51.5 51.5 51.5 ...
##  $ Longitude         : num  -0.11 -0.11 -0.11 -0.11 -0.11 ...
##  $ DistanceFromCenter: num  2719 2719 2719 2719 2719 ...
##  $ lon               : num  -0.108 -0.108 -0.108 -0.108 -0.108 ...
##  $ lat               : num  51.5 51.5 51.5 51.5 51.5 ...
##  $ EmployeeRate      : num  3.82 3.82 3.82 3.82 3.82 ...
##  - attr(*, ".internal.selfref")=<externalptr>
##  - attr(*, "sorted")= chr "Start_Station_ID"
```

Feature engineering

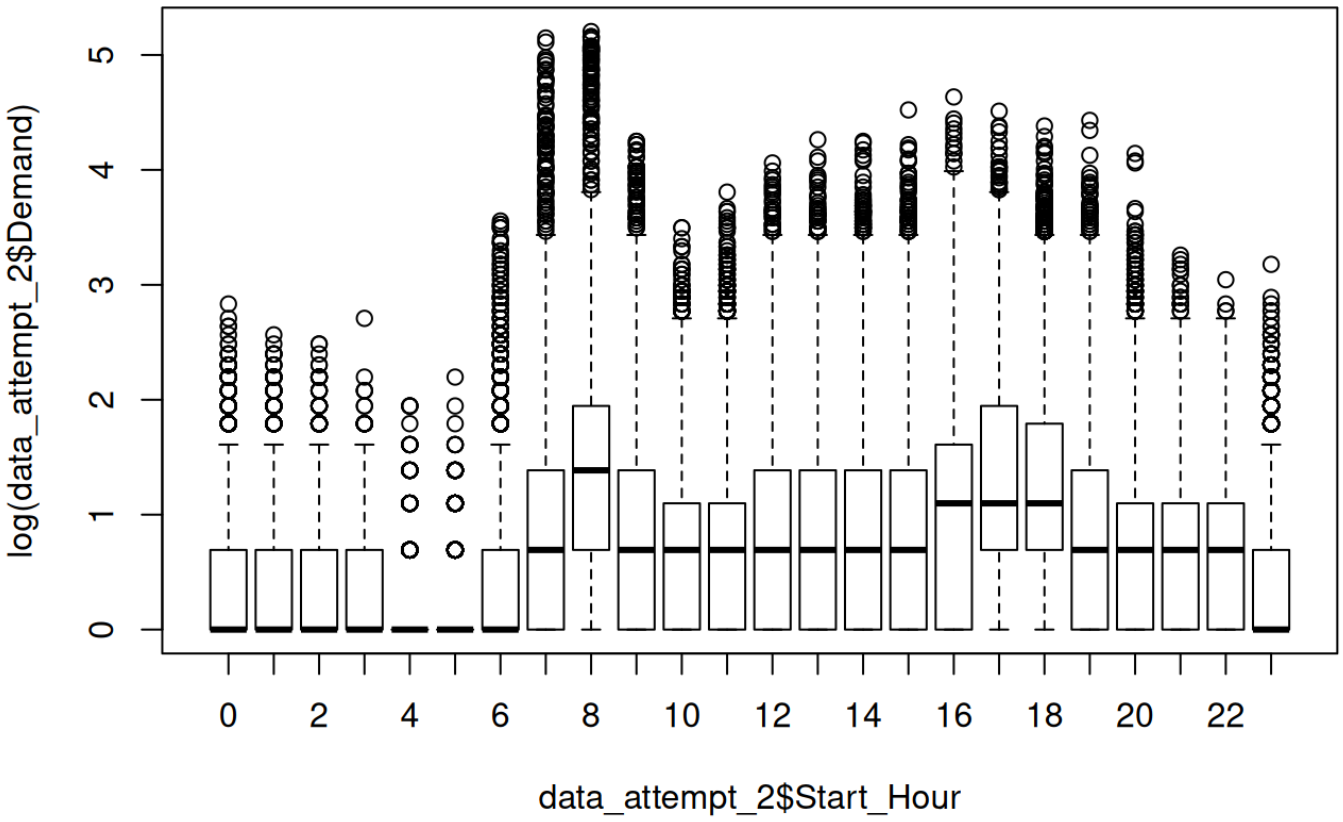
Based of first attempt I create `PeakHourWeekday` and `PeakHourWeekend` features.

I'm creating a feature `Weekday` describing whether the journey was taken during a week or on a weekend.

```
library(timeDate)
data_attempt_2$Weekday = as.numeric(isWeekday(data_attempt_2$Date, wday=1:5))
```

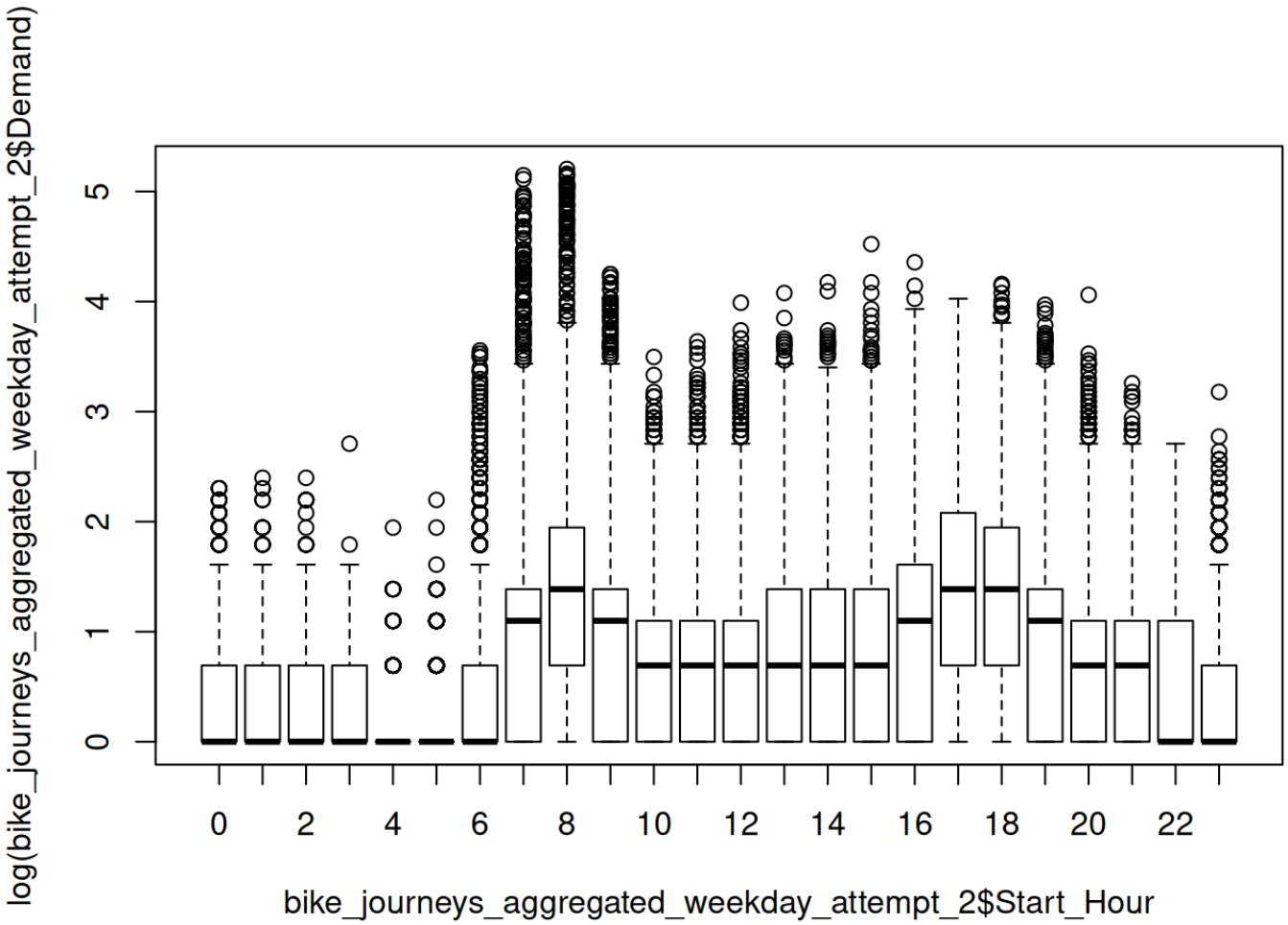
I'm plotting demand to see how it looks on each hour.

```
boxplot(log(data_attempt_2$Demand)~data_attempt_2$Start_Hour)
```



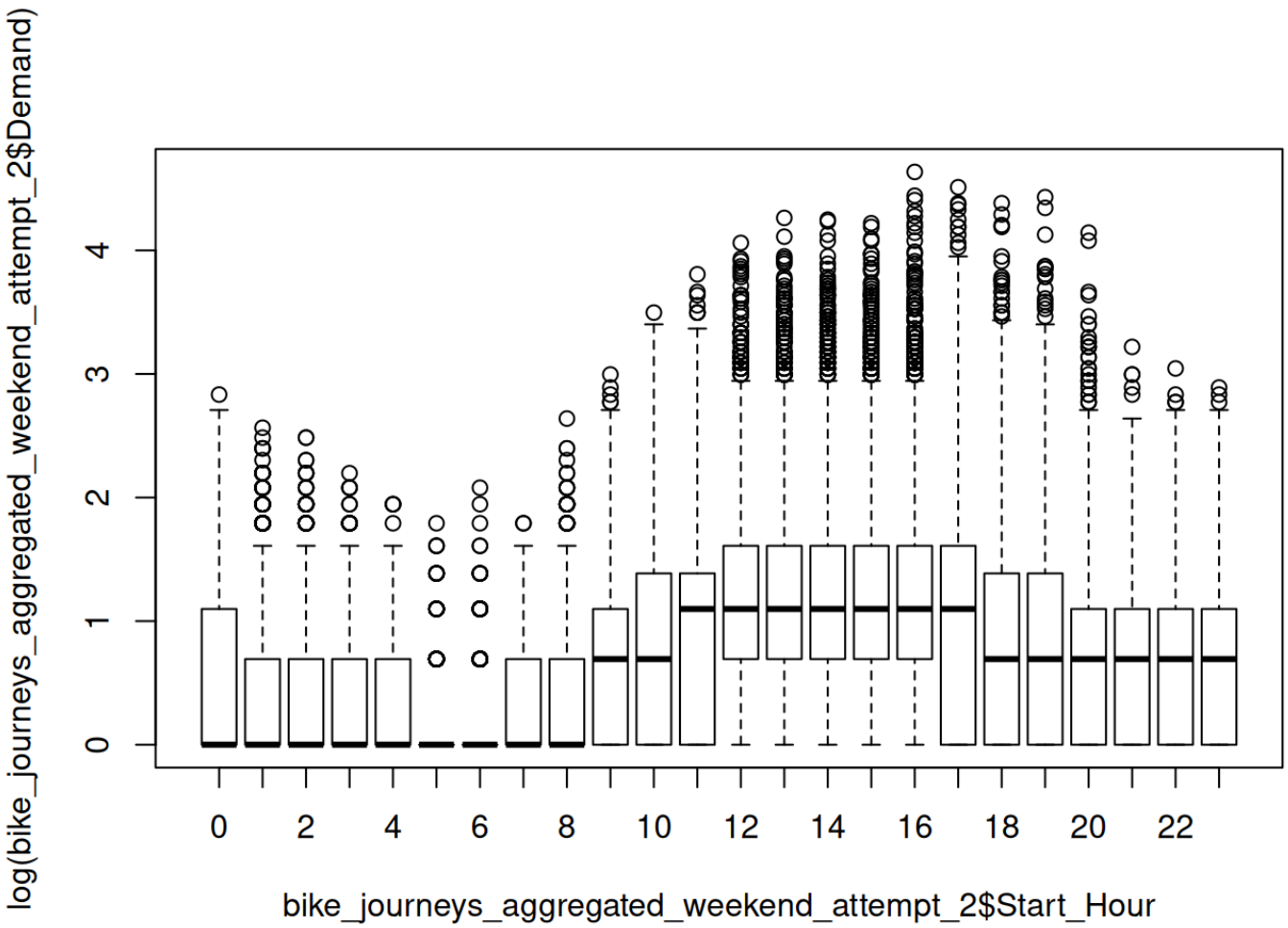
Now I'm checking how the demand looks during week days.

```
bike_journeys_aggregated_weekday_attempt_2 = data_attempt_2[data_attempt_2$Weekday
== 1]
boxplot(log(bike_journeys_aggregated_weekday_attempt_2$Demand)~bike_journeys_aggreg
ated_weekday_attempt_2$Start_Hour)
```



And how during the weekend.

```
bike_journeys_aggregated_weekend_attempt_2 = data_attempt_2[data_attempt_2$Weekday
== 0]
boxplot(log(bike_journeys_aggregated_weekend_attempt_2$Demand)~bike_journeys_aggreg
ated_weekend_attempt_2$Start_Hour)
```



Based on the plots above, I compute `PeakHourWeekday` and `PeakHourWeekend` to select journeys made during week day peak hours and weekend peak hours.

```
data_attempt_2$PeakHourWeekday = ifelse(data_attempt_2$Weekday == 1 & data_attempt_
2$Start_Hour == 8 | (data_attempt_2$Start_Hour >= 17 & data_attempt_2$Start_Hour <=
18), 1, 0)
data_attempt_2$PeakHourWeekend = ifelse(data_attempt_2$Weekday == 1 & data_attempt_
2$Start_Hour >= 11 & data_attempt_2$Start_Hour <= 17, 1, 0)
```

```
data_attempt_2$lon = NULL
data_attempt_2$lat = NULL
data_attempt_2$Date = NULL
data_attempt_2$Longitude = NULL
data_attempt_2$Latitude = NULL
data_attempt_2$Start_Station_ID = NULL
data_attempt_2$DayOfWeek = NULL
data_attempt_2$Start_Hour = NULL
str(data_attempt_2)
```

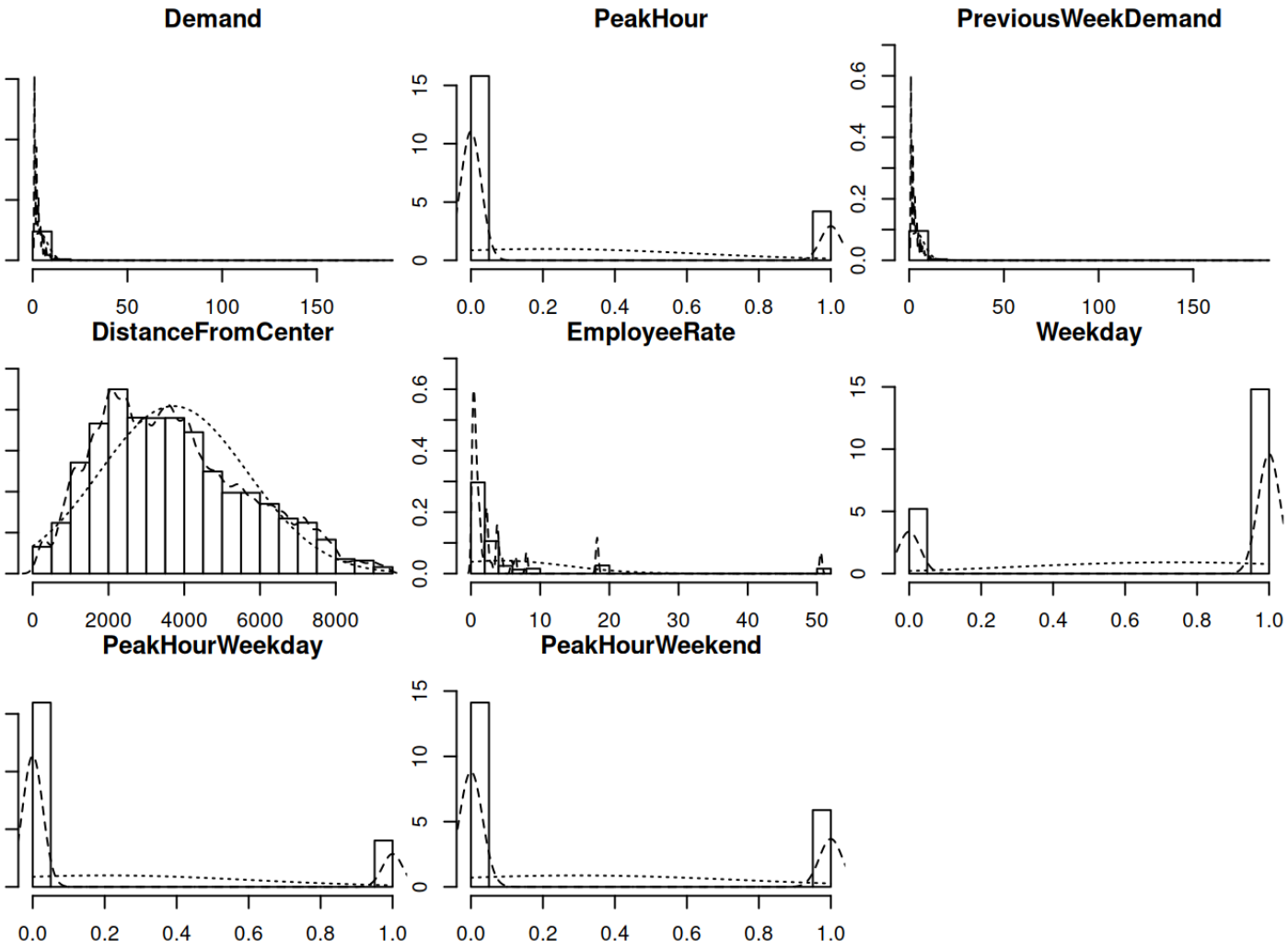
```
## Classes 'data.table' and 'data.frame':   354077 obs. of  8 variables:
##  $ Demand            : int  4 3 3 3 2 3 4 10 8 9 ...
##  $ PeakHour          : num  0 0 0 0 0 0 0 1 1 1 ...
##  $ PreviousWeekDemand: int  3 3 3 2 3 4 2 8 9 1 ...
##  $ DistanceFromCenter: num  2719 2719 2719 2719 2719 ...
##  $ EmployeeRate      : num  3.82 3.82 3.82 3.82 3.82 ...
##  $ Weekday           : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ PeakHourWeekday   : num  0 0 0 0 0 0 0 1 1 1 ...
##  $ PeakHourWeekend   : num  0 0 0 0 0 0 0 0 0 0 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Summary of the data.

```
summary(data_attempt_2)
```

```
##      Demand          PeakHour    PreviousWeekDemand DistanceFromCenter
##  Min.   :  1.00   Min.   :0.00   Min.   :  1.000    Min.   : 146.2
##  1st Qu.:  1.00   1st Qu.:0.00   1st Qu.:  1.000    1st Qu.:2170.8
##  Median :  2.00   Median :0.00   Median :  2.000    Median :3471.2
##  Mean   :  3.47   Mean   :0.21   Mean   :  3.525    Mean   :3715.1
##  3rd Qu.:  4.00   3rd Qu.:0.00   3rd Qu.:  4.000    3rd Qu.:4959.9
##  Max.   :182.00   Max.   :1.00   Max.   :182.000    Max.   :9354.3
##   EmployeeRate        Weekday       PeakHourWeekday  PeakHourWeekend
##  Min.   : 0.1321   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.: 0.5018   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median : 1.1104   Median :1.0000   Median :0.0000   Median :0.0000
##  Mean   : 4.2903   Mean   :0.7401   Mean   :0.2012   Mean   :0.2942
##  3rd Qu.: 3.7458   3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:1.0000
##  Max.   :50.5540   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

```
library(psych)
multi.hist(data_attempt_2)
```



Few variables are not normally distributed, such as `Demand`, `PreviousWeekDemand` and `EmployeeRate` and it can be transformed to log value.

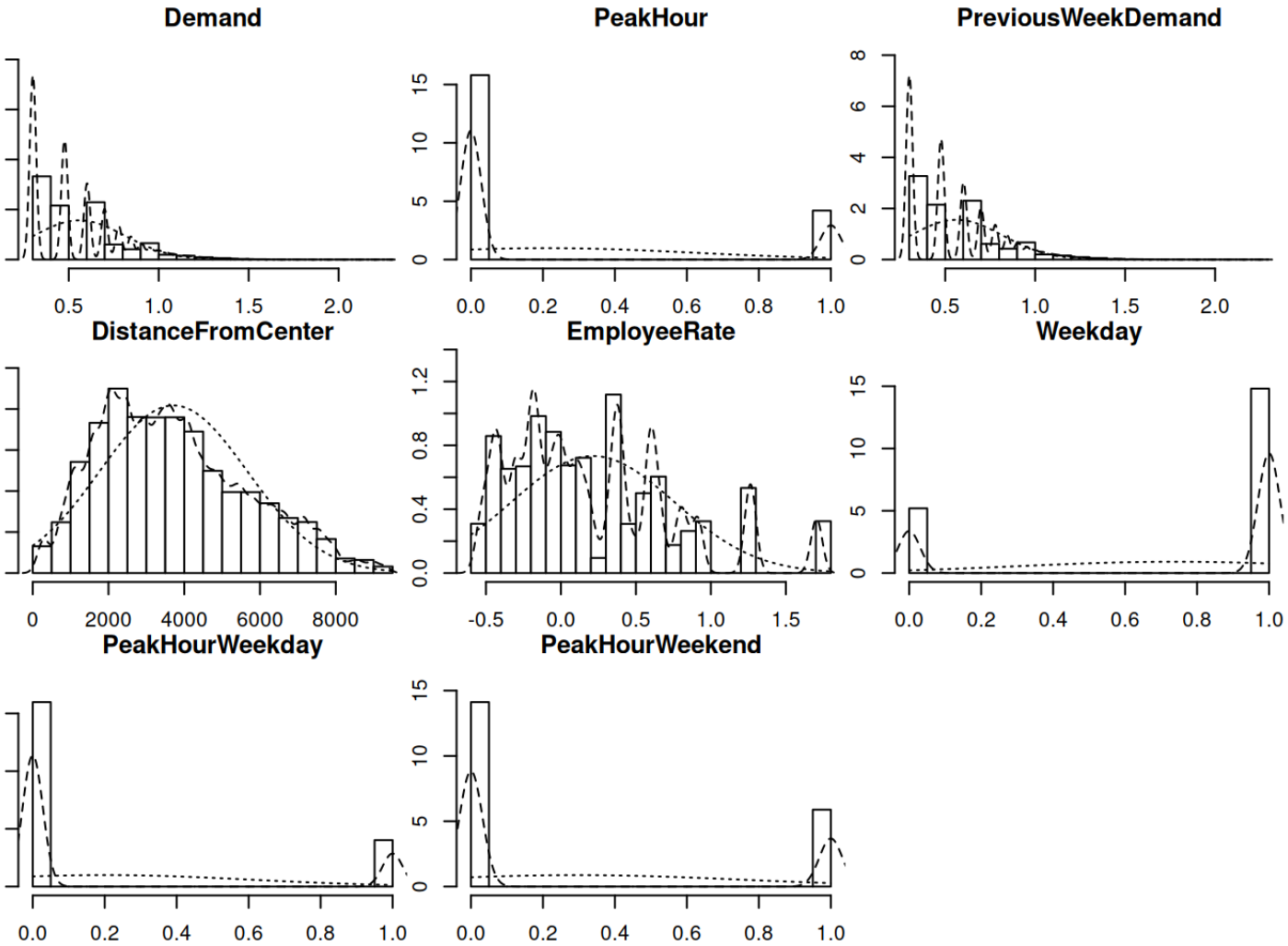
```
data_attempt_2$Demand = log10(data_attempt_2$Demand + min(data_attempt_2[Demand!=0]
$Demand))
data_attempt_2$EmployeeRate = log10(data_attempt_2$EmployeeRate + min(data_attempt_
2[EmployeeRate!=0]$EmployeeRate))
data_attempt_2$PreviousWeekDemand = log10(data_attempt_2$PreviousWeekDemand + min(d
ata_attempt_2[PreviousWeekDemand!=0]$PreviousWeekDemand))
```

Summary of the dataset.

```
summary(data_attempt_2)
```

```
##      Demand          PeakHour    PreviousWeekDemand DistanceFromCenter
##  Min.   :0.3010   Min.   :0.00   Min.   :0.3010     Min.   : 146.2
##  1st Qu.:0.3010   1st Qu.:0.00   1st Qu.:0.3010     1st Qu.:2170.8
##  Median :0.4771   Median :0.00   Median :0.4771     Median :3471.2
##  Mean   :0.5588   Mean   :0.21   Mean   :0.5631     Mean   :3715.1
##  3rd Qu.:0.6990   3rd Qu.:0.00   3rd Qu.:0.6990     3rd Qu.:4959.9
##  Max.   :2.2625   Max.   :1.00   Max.   :2.2625     Max.   :9354.3
##   EmployeeRate        Weekday       PeakHourWeekday  PeakHourWeekend
##  Min.   :-0.5780   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:-0.1980   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median : 0.0943   Median :1.0000   Median :0.0000   Median :0.0000
##  Mean   : 0.2132   Mean   :0.7401   Mean   :0.2012   Mean   :0.2942
##  3rd Qu.: 0.5886   3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:1.0000
##  Max.   : 1.7049   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

```
multi.hist(data_attempt_2)
```



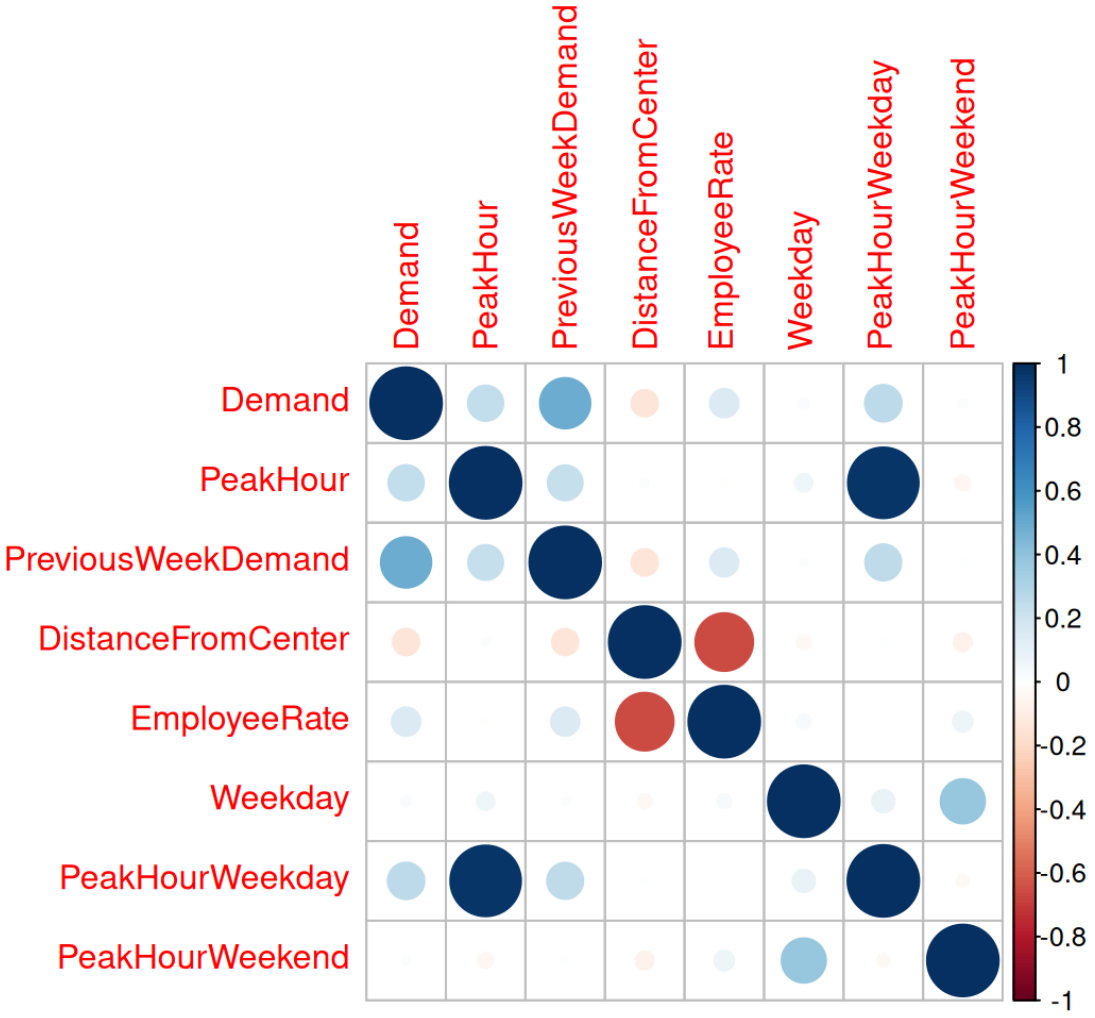
Standarising the data.

```
data_std_attempt_2 = as.data.table(scale(data_attempt_2))
summary(data_std_attempt_2)
```

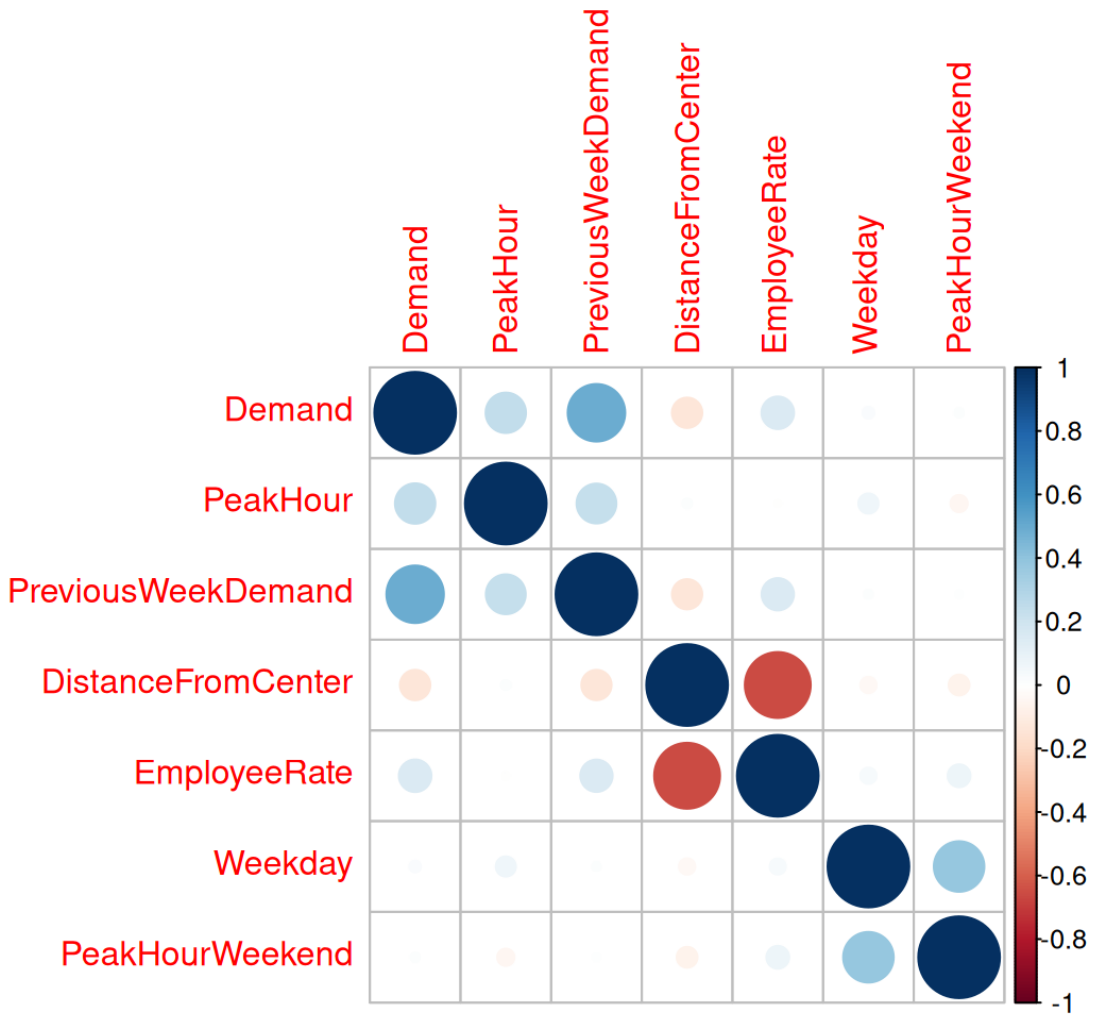
```
##      Demand            PeakHour       PreviousWeekDemand DistanceFromCenter
##  Min.   :-1.0142   Min.   :-0.5155   Min.   :-1.0238    Min.   :-1.8282
##  1st Qu.:-1.0142   1st Qu.:-0.5155   1st Qu.:-1.0238    1st Qu.:-0.7911
##  Median :-0.3215   Median :-0.5155   Median :-0.3360    Median :-0.1250
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000    Mean   : 0.0000
##  3rd Qu.: 0.5512   3rd Qu.:-0.5155   3rd Qu.: 0.5305    3rd Qu.: 0.6376
##  Max.   : 6.7017   Max.   : 1.9398   Max.   : 6.6376    Max.   : 2.8887
##   EmployeeRate         Weekday        PeakHourWeekday   PeakHourWeekend
##  Min.   :-1.4541   Min.   :-1.6875   Min.   :-0.5019   Min.   :-0.6456
##  1st Qu.:-0.7557   1st Qu.:-1.6875   1st Qu.:-0.5019   1st Qu.:-0.6456
##  Median :-0.2186   Median : 0.5926   Median :-0.5019   Median :-0.6456
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.6898   3rd Qu.: 0.5926   3rd Qu.:-0.5019   3rd Qu.: 1.5489
##  Max.   : 2.7413   Max.   : 0.5926   Max.   : 1.9923   Max.   : 1.5489
```

Checking multicollinearity.

```
library(corrplot)
corrplot(cor(data_std_attempt_2))
```




```
data_std_attempt_2$PeakHourWeekday = NULL
corrplot(cor(data_std_attempt_2))
```



Algorithms

Goal requires to predict number of rented bikes so we have to implement regression model.

```
set.seed(0)
trainIdx_attempt_2 = sample(1:nrow(data_std_attempt_2), 0.75*nrow(data_std_attempt_
2))
train_attempt_2 = data_std_attempt_2[trainIdx_attempt_2]
test_attempt_2 = data_std_attempt_2[-trainIdx_attempt_2]
```

```
lr_attempt_2 = lm(Demand ~ ., data=train_attempt_2)
train_preds_attempt_2 = predict(lr_attempt_2, train_attempt_2)
test_preds_attempt_2 = predict(lr_attempt_2, test_attempt_2)
print(paste("R2 on train:", cor(train_preds_attempt_2, train_attempt_2$Demand)^2))
```

```
## [1] "R2 on train: 0.270830364966547"
```

```
print(paste("R2 on test:", cor(test_preds_attempt_2, test_attempt_2$Demand)^2))
```

```
## [1] "R2 on test: 0.269417868978199"
```

```
print(paste("RMSE on train:", rmse(train_preds_attempt_2, train_attempt_2$Demand)))
```

```
## [1] "RMSE on train: 0.854155885015494"
```

```
print(paste("RMSE on test:", rmse(test_preds_attempt_2, test_attempt_2$Demand)))
```

```
## [1] "RMSE on test: 0.854012468056662"
```

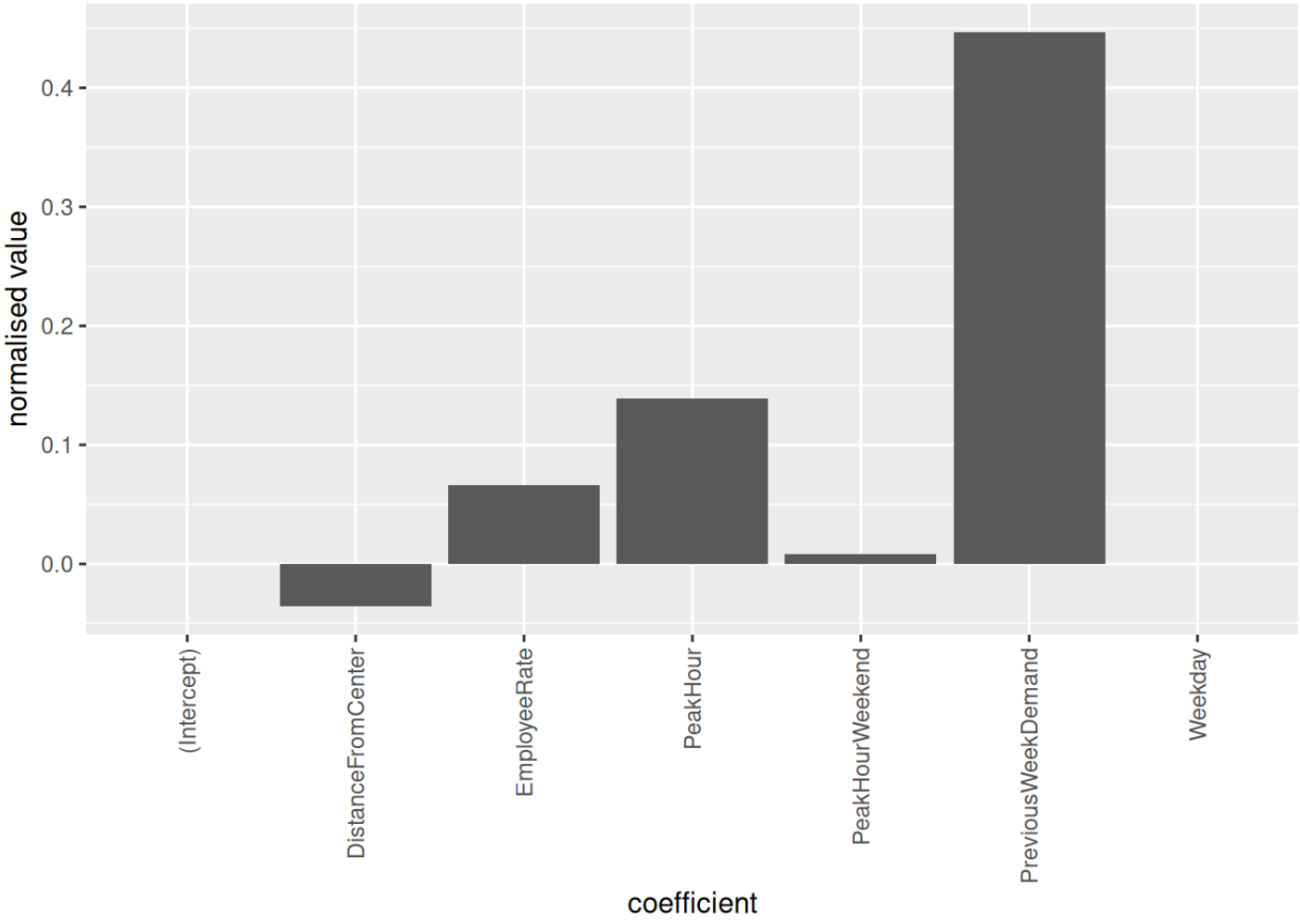
There's no difference between train and test scores which indicates that our model is stable. We do not overfit which means we don't need any regularisation.

Data understanding

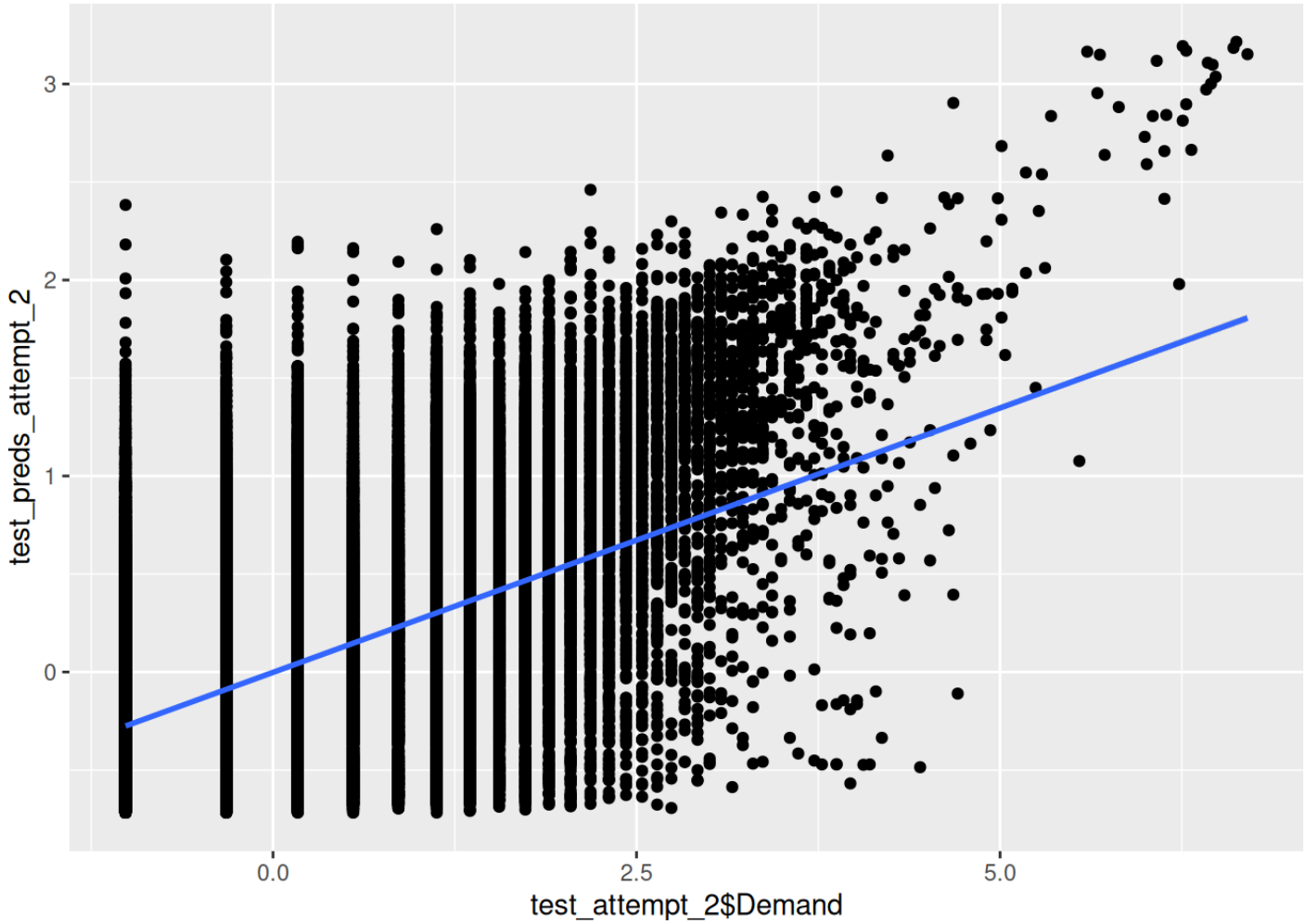
```
lr_attempt_2 = lm(Demand ~ ., data=data_std_attempt_2)
summary(lr_attempt_2)
```

```
##
## Call:
## lm(formula = Demand ~ ., data = data_std_attempt_2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3983 -0.6077 -0.0917  0.5661  6.4067
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -2.901e-13  1.435e-03   0.000    1.000
## PeakHour            1.388e-01  1.486e-03  93.405  < 2e-16 ***
## PreviousWeekDemand  4.470e-01  1.500e-03 298.016  < 2e-16 ***
## DistanceFromCenter -3.542e-02  1.893e-03 -18.706  < 2e-16 ***
## EmployeeRate        6.578e-02  1.900e-03  34.620  < 2e-16 ***
## Weekday            -2.714e-04  1.559e-03  -0.174    0.862
## PeakHourWeekend     7.910e-03  1.561e-03   5.065 4.08e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8541 on 354070 degrees of freedom
## Multiple R-squared:  0.2705, Adjusted R-squared:  0.2705
## F-statistic: 2.188e+04 on 6 and 354070 DF,  p-value: < 2.2e-16
```

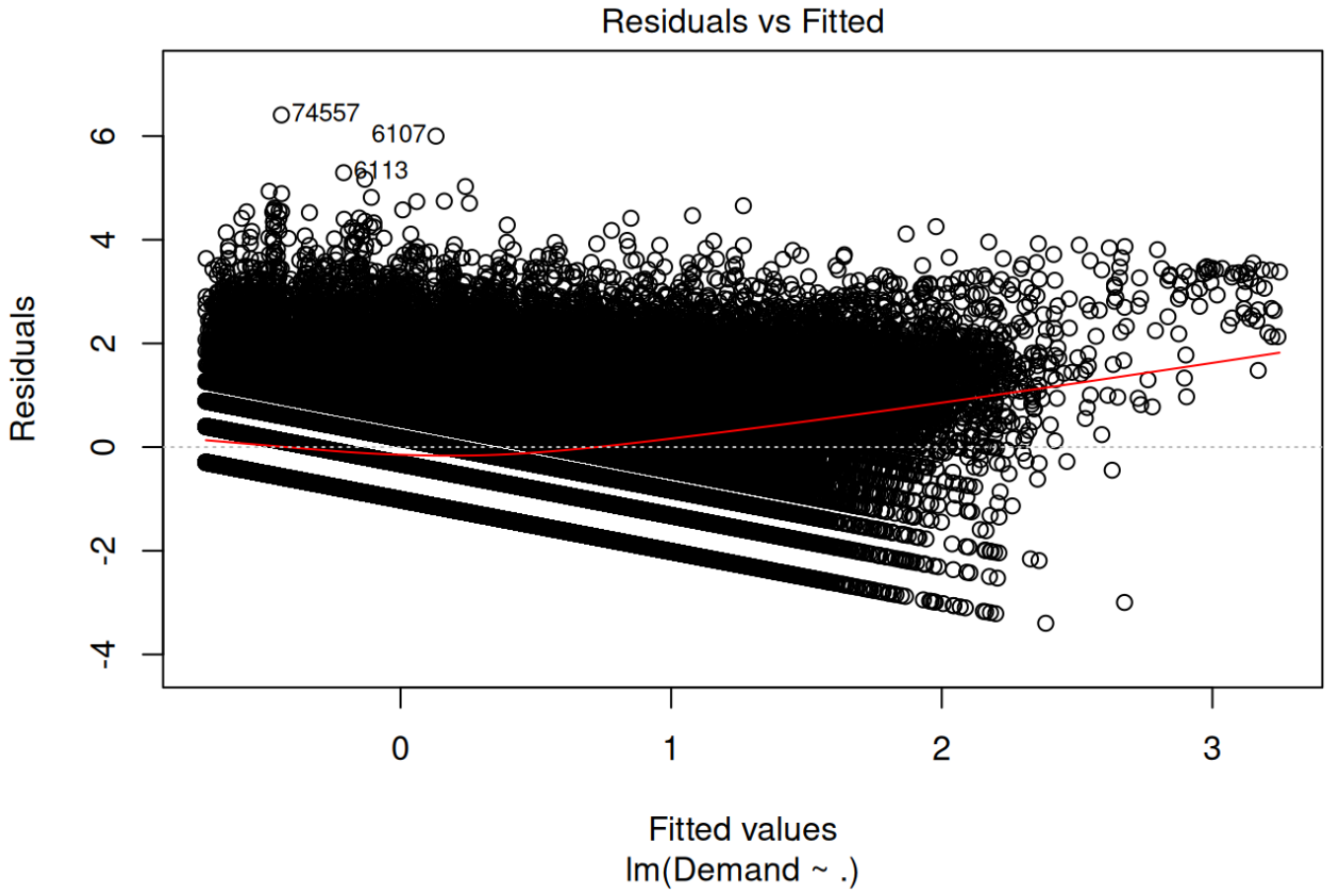
```
library(ggplot2)
ggplot(, aes(x = names(lr_attempt_2$coefficients), y=lr_attempt_2$coefficients)) +
  geom_bar(stat="identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5)) +
  xlab("coefficient") +
  ylab("normalised value")
```

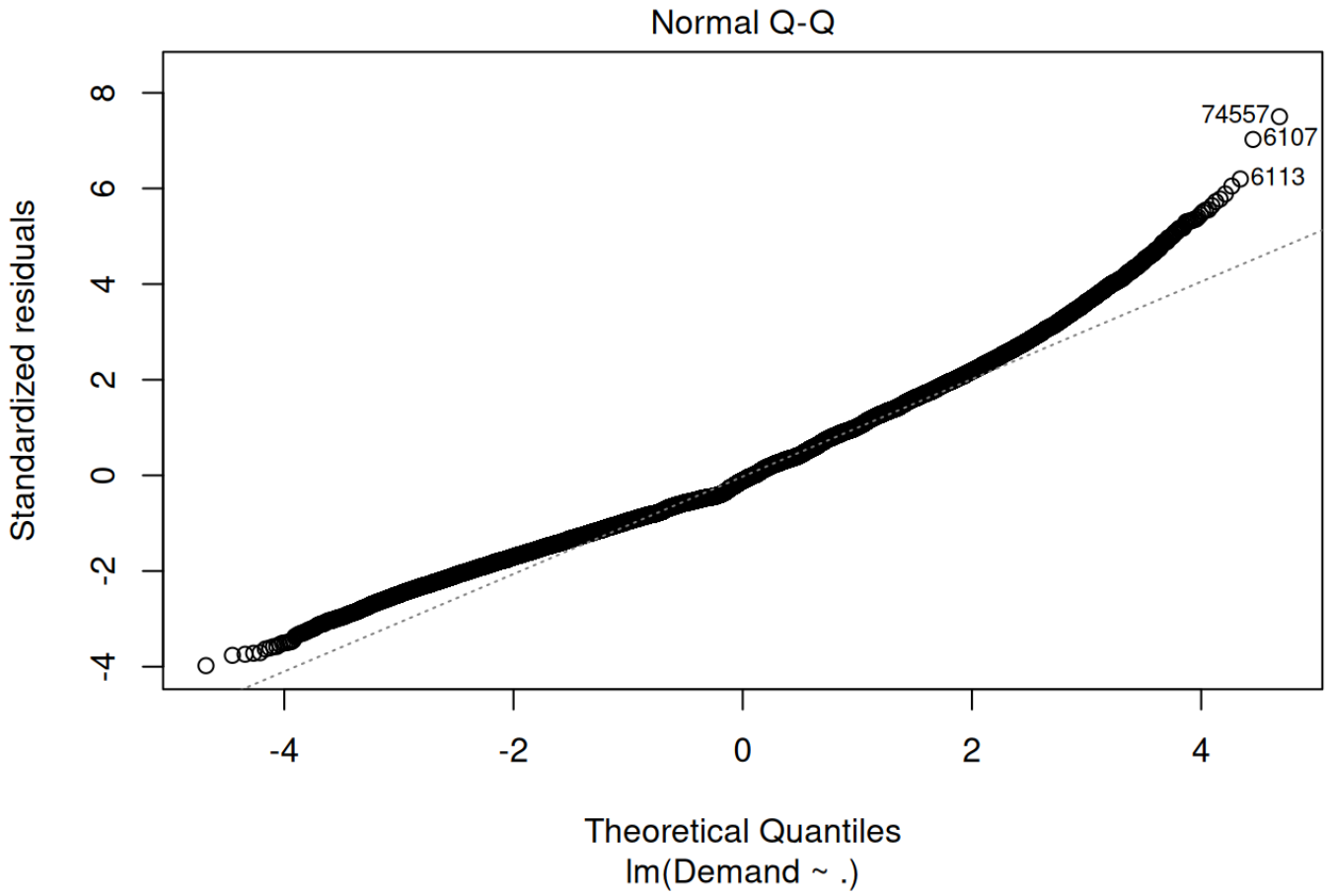


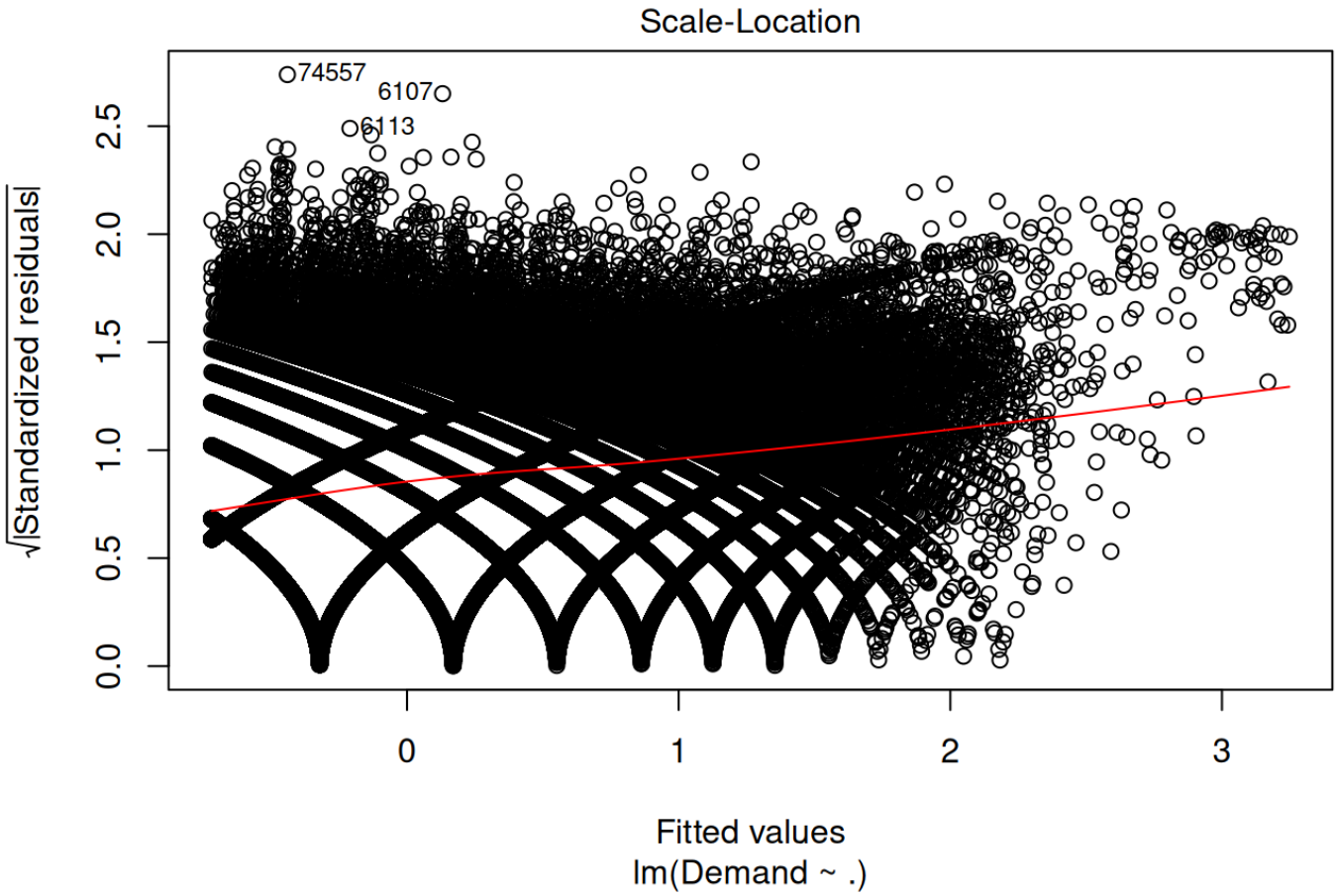

```
ggplot(test_attempt_2, aes(x = test_attempt_2$Demand, y=test_preds_attempt_2)) +
  geom_point() +
  geom_smooth(method = "lm")
```

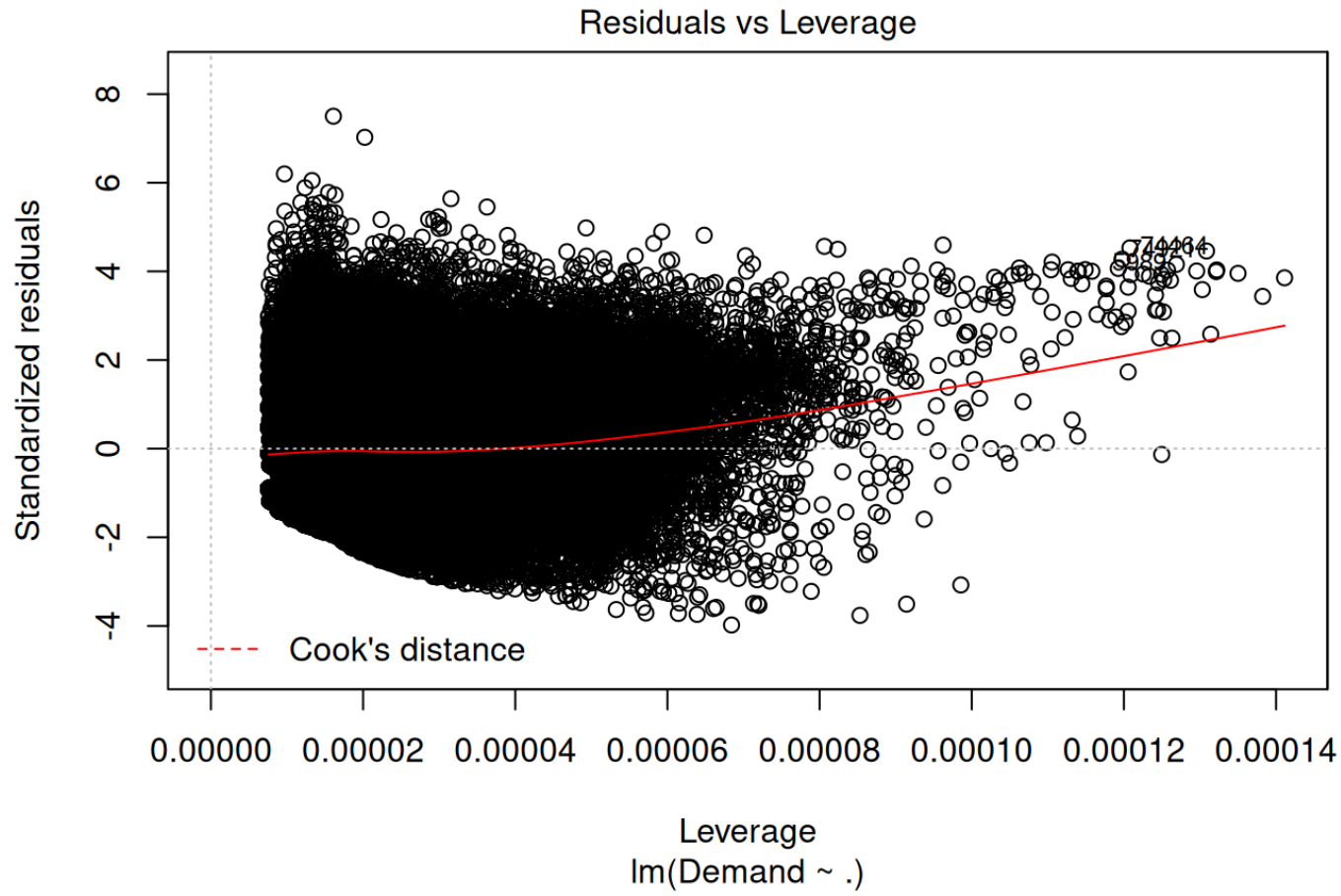



```
plot(lr_attempt_2)
```

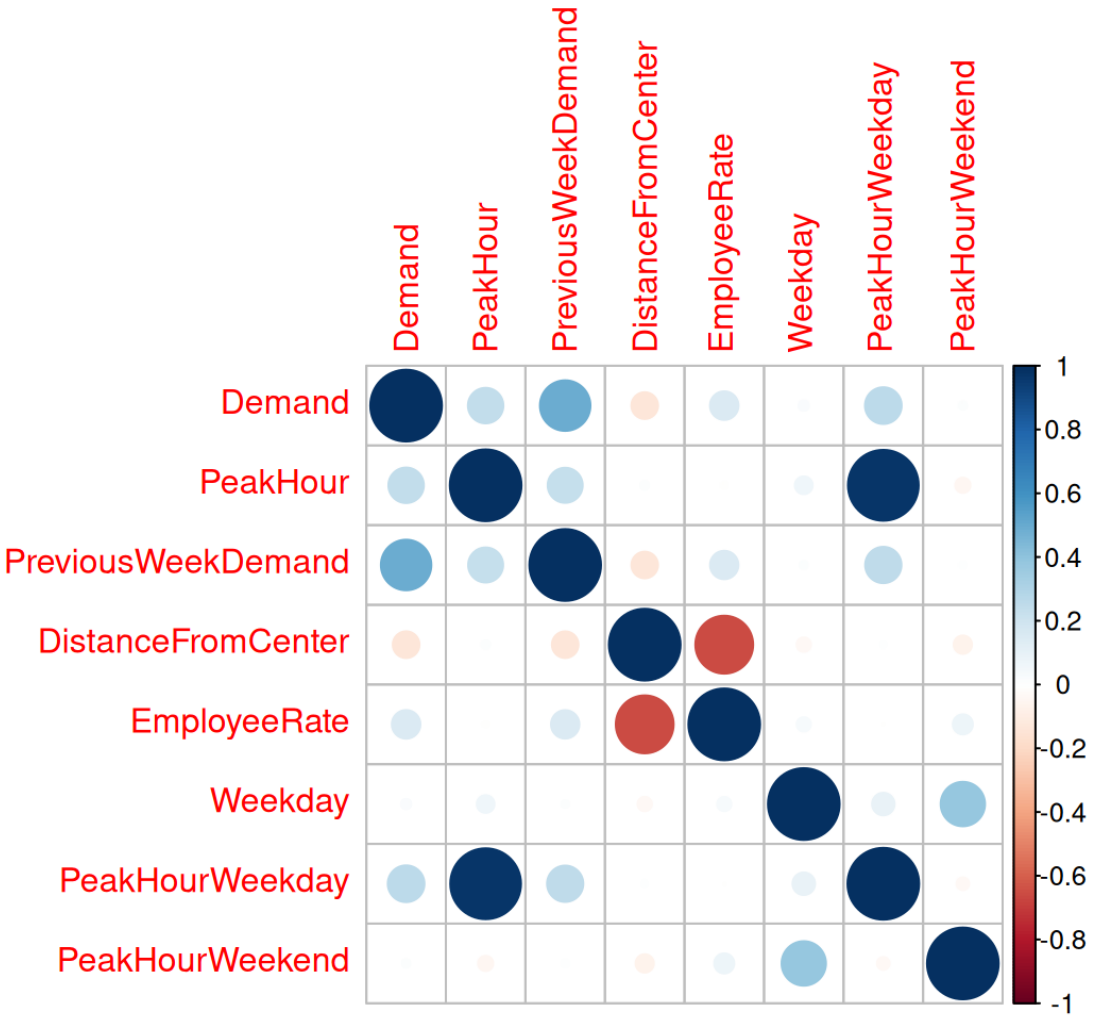

Residuals vs Fitted
74557
6107
6113
Residuals
6
4
2
0
-2
-4
0
1
2
3
Fitted values
lm(Demand ~ .)


Normal Q-Q
74557
6107
6113
Standardized residuals
8
6
4
2
0
-2
-4
-4
-2
0
2
4
Theoretical Quantiles
lm(Demand ~ .)


Scale-Location
74557
6107
6113
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
0
1
2
3
Fitted values
lm(Demand ~ .)


Residuals vs Leverage
Standardized residuals
8
6
4
2
0
-2
-4
Cook's distance
0.00000
0.00002
0.00004
0.00006
0.00008
0.00010
0.00012
0.00014
Leverage
lm(Demand ~ .)

```
corrplot(cor(data_attempt_2))
```



H1. Higher demand of bikes rented during peak hours 8 and 17 to 18 *True*
H3. Higher demand of bikes rented at stations which are close to central London. *False*
H4. Higher demand of bikes rented where is high employment rate. *False*
H12. Bikes demand rented at each station depends on previous demand at the same station. *True*

Limitation

Almost 27% of the variation of the dependent variable is described by independent variables. This means it is not good model.

Attempt 3

In the second attempt I remove H2 and H3 and add H13, H14 and H15.

Hypothesis

H1. Higher demand of bikes rented during peak hours 8 and 17 to 18.
~~H3. Higher demand of bikes rented at stations which are close to central London.~~
~~H4. Higher demand of bikes rented where is high employment rate.~~
H12. Bikes demand rented at each station depends on previous demand at the same station.

H13. Bikes demand rented at each station depends on previous demand at the closest stations.
H14. Higher demand of bikes rented at stations which are close to the tfl station.
H15. Higher demand of bikes rented at stations which are close to the park.

Metrics

- PeakHour Start_Hour binned into time slots containing peak hours 8 and 17-18. Linked to H1.
- FirstNearestStationPreviousDemand. Previous week demand at the first nearest station. Linked to H13.
- SecondNearestStationPreviousDemand. Previous week demand at the second nearest station. Linked to H13.
- closest_tfl_stations_distance. Distance from each station to the closest tfl station. Linked to H14.
- closest_park_distance. Distance from each station to the closest park. Linked to H15.

Data processing

The data needs to be transformed from the format:

<Journey_Duration, Journey_ID, End_Date, End_Month, End_Year, End_Hour, End_Minute, End_Station_ID, Start_Date, Start_Month, Start_Year, Start_Hour, Start_Minute, Start_Station_ID>
<Station_ID, Capacity, Latitude, Longitude, Station_Name>
<WardCode, WardName, Borough, NESW, AreaSqKm, lon, lat, IncomeScor, LivingEnSc, NoEmployee, GrenSpace, PopDen, BornUK, NotBornUK, NoCTFtoH, NoDwelling, NoFlats, NoHouses, NoWndDwel, MedHPrice>

Into the format:

<Demand, PeakHour, FirstNearestStationPreviousDemand, SecondNearestStationPreviousDemand, closest_tfl_stations_distance, closest_park_distance>

Implementing R code to tranform the datasets.

The code below selects only columns needed for our task from bike journeys dataset. It merges three columns (`Start_Year` , `Start_Month` , `Start_Date`) into a date format. Then it counts number of rented bikes for each station, each date and hour and saves it in `Demand` column. Then it computes `DayOfWeek` , `PeakHour` , `Weekend` from `Date` column.
I compute `PreviousWeekDemand` for each station, day of week and hour by grouping, ordering and shifting `Demand` column. As it creates rows without data I remove those from the dataset.

```
bike_journeys_attempt_3 = bike_journeys[, .(Start_Station_ID, Date = format(as.Date
(paste(Start_Year, Start_Month, Start_Date, sep='-'), '%y-%m-%d'), '%Y-%m-%d'), Sta
rt_Hour)]
bike_journeys_attempt_3 = bike_journeys_attempt_3[, .(Demand=sum(.N)), by=.(Start_S
tation_ID, Date, Start_Hour)]
bike_journeys_attempt_3$DayOfWeek = as.numeric(as.POSIXlt(bike_journeys_attempt_3$D
ate)$wday)

bike_journeys_attempt_3$PeakHour = ifelse((bike_journeys_attempt_3$Start_Hour >= 8
& bike_journeys_attempt_3$Start_Hour <= 8) | (bike_journeys_attempt_3$Start_Hour >=
17 & bike_journeys_attempt_3$Start_Hour <= 18), 1, 0)

bike_journeys_attempt_3 = bike_journeys_attempt_3[order(-Date), .(Demand, Date, Pea
kHour, PreviousWeekDemand=shift(Demand, type="lead")), by=c('Start_Station_ID', 'Da
yOfWeek', 'Start_Hour')]
bike_journeys_attempt_3 = bike_journeys_attempt_3[complete.cases(bike_journeys_atte
mpt_3)]

str(bike_journeys_attempt_3)
```

```
## Classes 'data.table' and 'data.frame':   356389 obs. of  7 variables:
##  $ Start_Station_ID  : int  251 251 251 251 251 251 251 550 550 105 ...
##  $ DayOfWeek         : num  2 2 2 2 2 2 2 2 2 ...
##  $ Start_Hour        : int  17 17 17 17 17 17 17 14 14 14 ...
##  $ Demand            : int  31 31 38 44 30 19 31 1 3 3 ...
##  $ Date              : chr  "2017-09-19" "2017-09-12" "2017-09-05" "2017-08-29"
...
##  $ PeakHour          : num  1 1 1 1 1 1 1 0 0 0 ...
##  $ PreviousWeekDemand: int  31 38 44 30 19 31 33 3 2 1 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

I find the nearest two bike stations for each bike station.
To do that I use `which.minn` which returns index of `n` smallest distances from which the first is 0 distance to self, so I ignore it and take second and third.

```
library(doBy)
library(geosphere)

nearest_bike_stations_distance_matrix = distm(bike_stations[,c('Longitude','Latitud
e')], bike_stations[,c('Longitude','Latitude')], fun=distGeo)
nearest_bike_stations_indexes = apply(nearest_bike_stations_distance_matrix, 1, FUN
= function(x) {which.minn(x, 3)})
nearest_bike_stations_first = bike_stations[nearest_bike_stations_indexes[2,]]
nearest_bike_stations_second = bike_stations[nearest_bike_stations_indexes[3,]]
nearest_bike_stations_first = nearest_bike_stations_first[, .(First_Nearest_Station
_ID=Station_ID, First_Nearest_Station_Name=Station_Name)]
nearest_bike_stations_second = nearest_bike_stations_second[, .(Second_Nearest_Stat
ion_ID=Station_ID, Second_Nearest_Station_Name=Station_Name)]
bike_station_and_nearest_bike_stations = cbind(bike_stations, nearest_bike_stations
_first, nearest_bike_stations_second)
bike_station_and_nearest_bike_stations = bike_station_and_nearest_bike_stations[, c
('Station_ID', 'Station_Name', 'First_Nearest_Station_ID', 'First_Nearest_Station_N
ame', 'Second_Nearest_Station_ID', 'Second_Nearest_Station_Name')]
```

I merge bike station dataset with bike journeys dataset.

```
bike_journeys_with_nearest_stations = merge(bike_journeys_attempt_3, bike_station_a
nd_nearest_bike_stations, by.x='Start_Station_ID', by.y='Station_ID')
str(bike_journeys_with_nearest_stations)
```

```
## Classes 'data.table' and 'data.frame':   354077 obs. of  12 variables:
##  $ Start_Station_ID           : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ DayOfWeek                  : num  2 2 2 2 2 2 2 2 2 2 ...
##  $ Start_Hour                 : int  6 6 6 6 6 6 6 8 8 8 ...
##  $ Demand                     : int  4 3 3 3 2 3 4 10 8 9 ...
##  $ Date                       : chr  "2017-09-19" "2017-09-12" "2017-09-05" "201
7-08-29" ...
##  $ PeakHour                   : num  0 0 0 0 0 0 0 1 1 1 ...
##  $ PreviousWeekDemand         : int  3 3 3 2 3 4 2 8 9 1 ...
##  $ Station_Name               : chr  "River Street , Clerkenwell" "River Street
, Clerkenwell" "River Street , Clerkenwell" "River Street , Clerkenwell" ...
##  $ First_Nearest_Station_ID   : int  170 170 170 170 170 170 170 170 170 170 ...
##  $ First_Nearest_Station_Name : chr  "Hardwick Street, Clerkenwell" "Hardwick St
reet, Clerkenwell" "Hardwick Street, Clerkenwell" "Hardwick Street, Clerkenwell" ..
.
##  $ Second_Nearest_Station_ID  : int  189 189 189 189 189 189 189 189 189 189 ...
##  $ Second_Nearest_Station_Name: chr  "Claremont Square, Angel" "Claremont Square
, Angel" "Claremont Square, Angel" "Claremont Square, Angel" ...
##  - attr(*, ".internal.selfref")=<externalptr>
##  - attr(*, "sorted")= chr "Start_Station_ID"
```

I calculate previous week demand for the nearest two bike stations by joining bike journeys dataset with ifself with renamed `PreviousWeekDemand` column.

```
previous_week_demand_for_first_nearest_bike_station = bike_journeys_attempt_3[, .(S
tart_Station_ID, Date, FirstNearestStationPreviousDemand=PreviousWeekDemand, Start_
Hour)]
previous_week_demand_for_second_nearest_bike_station = bike_journeys_attempt_3[, .(
Start_Station_ID, Date, SecondNearestStationPreviousDemand=PreviousWeekDemand, Star
t_Hour)]

bike_journeys_with_nearest_stations = merge(bike_journeys_with_nearest_stations, pr
evious_week_demand_for_first_nearest_bike_station, by.x=c('First_Nearest_Station_ID
', 'Date', 'Start_Hour'), by.y=c('Start_Station_ID', 'Date', 'Start_Hour'))
bike_journeys_with_nearest_stations = merge(bike_journeys_with_nearest_stations, pr
evious_week_demand_for_second_nearest_bike_station, by.x=c('Second_Nearest_Station_
ID', 'Date', 'Start_Hour'), by.y=c('Start_Station_ID', 'Date', 'Start_Hour'))
str(bike_journeys_with_nearest_stations)
```

```
## Classes 'data.table' and 'data.frame':   168312 obs. of  14 variables:
##  $ Second_Nearest_Station_ID         : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Date                              : chr  "2017-08-08" "2017-08-08" "2017-08-0
8" "2017-08-08" ...
##  $ Start_Hour                        : int  7 7 8 8 9 9 11 12 18 18 ...
##  $ First_Nearest_Station_ID          : int  264 264 264 264 264 264 264 264 264
264 ...
##  $ Start_Station_ID                  : int  170 204 170 204 170 204 170 170 170
204 ...
##  $ DayOfWeek                         : num  2 2 2 2 2 2 2 2 2 2 ...
##  $ Demand                            : int  4 1 14 1 5 3 6 2 3 8 ...
##  $ PeakHour                          : num  0 0 1 1 0 0 0 0 1 1 ...
##  $ PreviousWeekDemand                : int  2 3 8 1 10 4 1 1 9 5 ...
##  $ Station_Name                      : chr  "Hardwick Street, Clerkenwell" "Marg
ery Street, Clerkenwell" "Hardwick Street, Clerkenwell" "Margery Street, Clerkenwel
l" ...
##  $ First_Nearest_Station_Name        : chr  "Tysoe Street, Clerkenwell" "Tysoe S
treet, Clerkenwell" "Tysoe Street, Clerkenwell" "Tysoe Street, Clerkenwell" ...
##  $ Second_Nearest_Station_Name       : chr  "River Street , Clerkenwell" "River
Street , Clerkenwell" "River Street , Clerkenwell" "River Street , Clerkenwell" ...
##  $ FirstNearestStationPreviousDemand : int  3 3 10 10 5 5 6 4 15 15 ...
##  $ SecondNearestStationPreviousDemand: int  3 3 8 8 4 4 1 1 5 5 ...
##  - attr(*, ".internal.selfref")=<externalptr>
##  - attr(*, "sorted")= chr  "Second_Nearest_Station_ID" "Date" "Start_Hour"
```

```
bike_journeys_with_nearest_stations$First_Nearest_Station_ID = NULL
bike_journeys_with_nearest_stations$Second_Nearest_Station_ID = NULL
bike_journeys_with_nearest_stations$First_Nearest_Station_Name = NULL
bike_journeys_with_nearest_stations$Second_Nearest_Station_Name = NULL
bike_journeys_with_nearest_stations$Station_Name = NULL
```

I select parks from the “points of interest” dataset.

```
library(dplyr)
```

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

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

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

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

```
library(stringr)
names(pois) = str_replace_all(names(pois), c(" " = "_" , "," = "" ))
parks = filter(pois, str_detect(Category_Labels, "Parks"))
```

I calculate distances from parks and TfL stations. I choose the minimal distances for each bike station and bind it to bike stations dataset.

```
bike_station_to_park_distance_matrix = distm(bike_stations[,c('Longitude','Latitude
')], parks[,c('Longitude','Latitude')], fun=distGeo)
bike_station_to_tfl_stations_distance_matrix = distm(bike_stations[,c('Longitude','
Latitude')], tfl_stations[,c('longitude','latitude')], fun=distGeo)

closest_park_distance = apply(bike_station_to_park_distance_matrix, 1, min)
closest_tfl_stations_distance = apply(bike_station_to_tfl_stations_distance_matrix,
1, min)

bike_stations_attempt_3 = bike_stations[, .(Station_ID, Latitude, Longitude)]
bike_stations_attempt_3 = cbind(bike_stations_attempt_3, closest_park_distance)
bike_stations_attempt_3 = cbind(bike_stations_attempt_3, closest_tfl_stations_dista
nce)

str(bike_stations_attempt_3)
```

```
## Classes 'data.table' and 'data.frame':   773 obs. of  5 variables:
##  $ Station_ID                  : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Latitude                    : num  51.5 51.5 51.5 51.5 51.5 ...
##  $ Longitude                   : num  -0.11 -0.1976 -0.0846 -0.121 -0.1569 ...
##  $ closest_park_distance       : num  1300 459 396 781 715 ...
##  $ closest_tfl_stations_distance: num  444.9 380.6 407.4 212.8 85.3 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
bike_stations_attempt_3$Latitude = NULL
bike_stations_attempt_3$Longitude = NULL
```

```
data_attempt_3 = merge(bike_journeys_with_nearest_stations, bike_stations_attempt_3
, by.x='Start_Station_ID', by.y='Station_ID')
str(data_attempt_3)
```

```
## Classes 'data.table' and 'data.frame':   168312 obs. of  11 variables:
##  $ Start_Station_ID                 : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Date                             : chr  "2017-08-08" "2017-08-08" "2017-08-0
8" "2017-08-08" ...
##  $ Start_Hour                       : int  8 9 11 12 18 8 9 7 8 17 ...
##  $ DayOfWeek                        : num  2 2 2 2 2 3 3 4 4 4 ...
##  $ Demand                           : int  3 2 3 1 4 5 4 2 8 3 ...
##  $ PeakHour                         : num  1 0 0 0 1 1 0 0 1 1 ...
##  $ PreviousWeekDemand               : int  8 4 1 1 5 15 6 1 10 2 ...
##  $ FirstNearestStationPreviousDemand : int  8 10 1 1 9 8 5 5 4 6 ...
##  $ SecondNearestStationPreviousDemand: int  2 4 2 3 2 7 2 2 5 3 ...
##  $ closest_park_distance            : num  1300 1300 1300 1300 1300 ...
##  $ closest_tfl_stations_distance    : num  445 445 445 445 445 ...
##  - attr(*, ".internal.selfref")=<externalptr>
##  - attr(*, "sorted")= chr "Start_Station_ID"
```

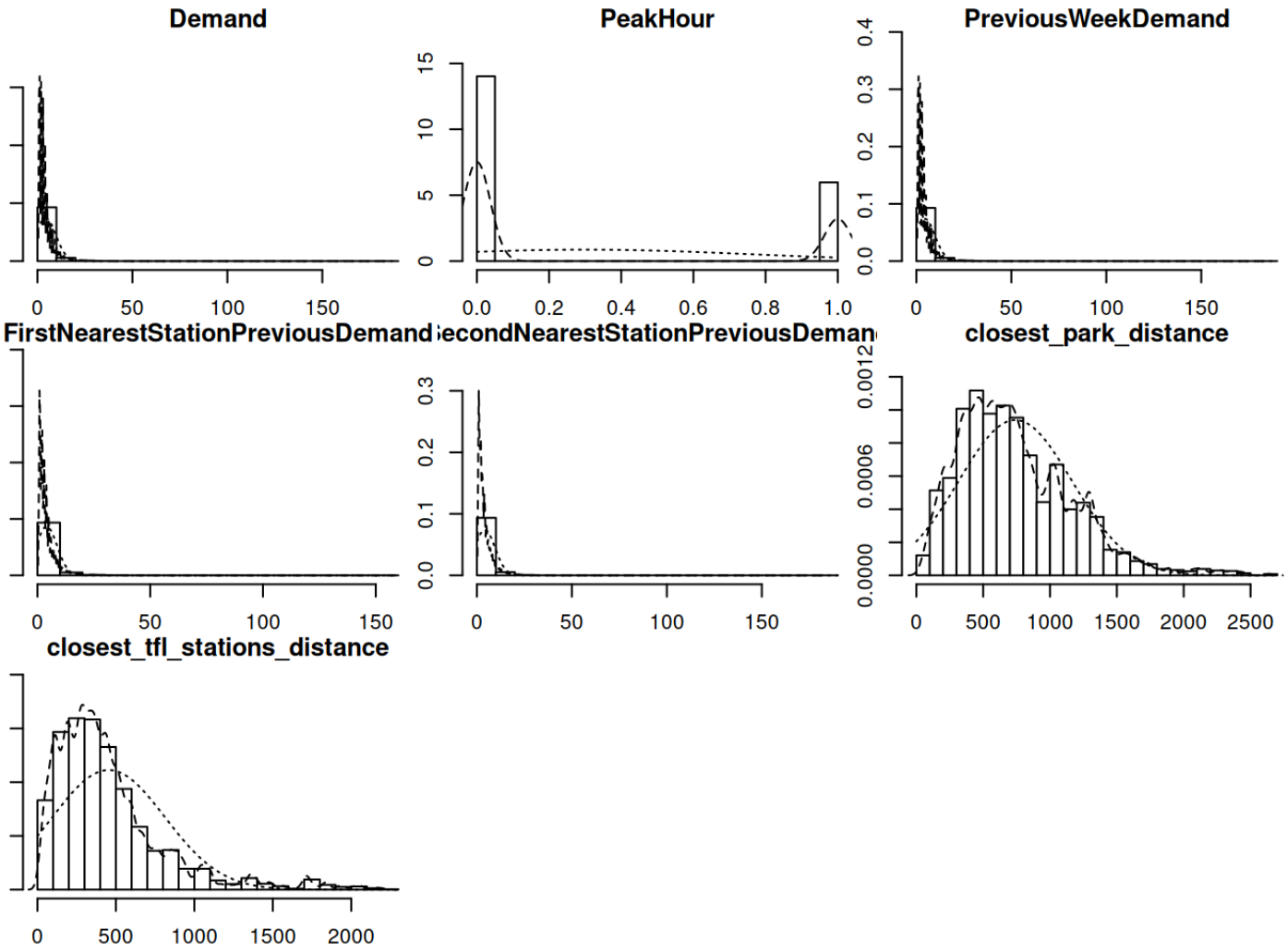
```
data_attempt_3$Date = NULL
data_attempt_3$Start_Station_ID = NULL
data_attempt_3$DayOfWeek = NULL
data_attempt_3$Start_Hour = NULL
str(data_attempt_3)
```

```
## Classes 'data.table' and 'data.frame':   168312 obs. of  7 variables:
##  $ Demand                           : int  3 2 3 1 4 5 4 2 8 3 ...
##  $ PeakHour                         : num  1 0 0 0 1 1 0 0 1 1 ...
##  $ PreviousWeekDemand               : int  8 4 1 1 5 15 6 1 10 2 ...
##  $ FirstNearestStationPreviousDemand : int  8 10 1 1 9 8 5 5 4 6 ...
##  $ SecondNearestStationPreviousDemand: int  2 4 2 3 2 7 2 2 5 3 ...
##  $ closest_park_distance            : num  1300 1300 1300 1300 1300 ...
##  $ closest_tfl_stations_distance    : num  445 445 445 445 445 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
summary(data_attempt_3)
```

```
##       Demand           PeakHour      PreviousWeekDemand
##  Min.   :  1.000   Min.   :0.0000   Min.   :  1.000
##  1st Qu.:  2.000   1st Qu.:0.0000   1st Qu.:  2.000
##  Median :  3.000   Median :0.0000   Median :  3.000
##  Mean   :  4.329   Mean   :0.2989   Mean   :  4.329
##  3rd Qu.:  5.000   3rd Qu.:1.0000   3rd Qu.:  5.000
##  Max.   :182.000   Max.   :1.0000   Max.   :182.000
##  FirstNearestStationPreviousDemand SecondNearestStationPreviousDemand
##  Min.   :  1.000                   Min.   :  1.000
##  1st Qu.:  1.000                   1st Qu.:  1.000
##  Median :  3.000                   Median :  3.000
##  Mean   :  4.104                   Mean   :  4.168
##  3rd Qu.:  5.000                   3rd Qu.:  5.000
##  Max.   :158.000                   Max.   :182.000
##  closest_park_distance closest_tfl_stations_distance
##  Min.   :  44.22       Min.   :   9.13
##  1st Qu.: 430.72       1st Qu.: 214.50
##  Median : 663.83       Median : 368.03
##  Mean   : 740.42       Mean   : 453.82
##  3rd Qu.:1004.97       3rd Qu.: 573.97
##  Max.   :2638.18       Max.   :2226.79
```

```
library(psych)
multi.hist(data_attempt_3)
```



Few variables are not normally distributed and they can be transformed to log value.

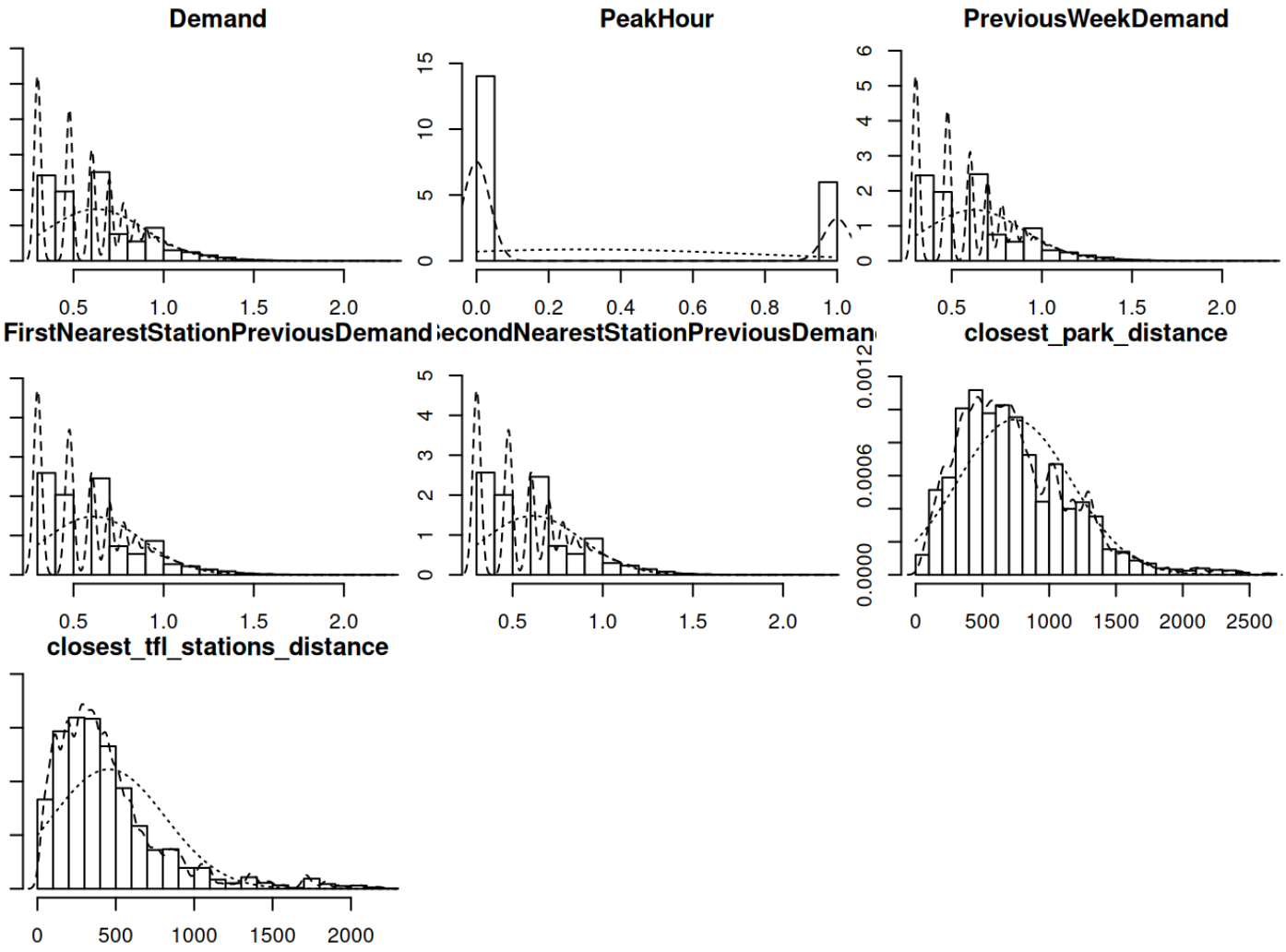
```
data_attempt_3$Demand = log10(data_attempt_3$Demand + min(data_attempt_3[Demand!=0]
$Demand))
data_attempt_3$PreviousWeekDemand = log10(data_attempt_3$PreviousWeekDemand + min(d
ata_attempt_3[PreviousWeekDemand!=0]$PreviousWeekDemand))
data_attempt_3$FirstNearestStationPreviousDemand = log10(data_attempt_3$FirstNeares
tStationPreviousDemand + min(data_attempt_3[FirstNearestStationPreviousDemand!=0]$F
irstNearestStationPreviousDemand))
data_attempt_3$SecondNearestStationPreviousDemand = log10(data_attempt_3$SecondNear
estStationPreviousDemand + min(data_attempt_3[SecondNearestStationPreviousDemand!=0
]$SecondNearestStationPreviousDemand))
```

Summary of the dataset.

```
summary(data_attempt_3)
```

```
##      Demand          PeakHour      PreviousWeekDemand
##  Min.   :0.3010   Min.   :0.0000   Min.   :0.3010
##  1st Qu.:0.4771   1st Qu.:0.0000   1st Qu.:0.4771
##  Median :0.6021   Median :0.0000   Median :0.6021
##  Mean   :0.6252   Mean   :0.2989   Mean   :0.6244
##  3rd Qu.:0.7782   3rd Qu.:1.0000   3rd Qu.:0.7782
##  Max.   :2.2625   Max.   :1.0000   Max.   :2.2625
##  FirstNearestStationPreviousDemand SecondNearestStationPreviousDemand
##  Min.   :0.3010                    Min.   :0.3010
##  1st Qu.:0.3010                    1st Qu.:0.3010
##  Median :0.6021                    Median :0.6021
##  Mean   :0.6103                    Mean   :0.6135
##  3rd Qu.:0.7782                    3rd Qu.:0.7782
##  Max.   :2.2014                    Max.   :2.2625
##  closest_park_distance closest_tfl_stations_distance
##  Min.   :  44.22       Min.   :   9.13
##  1st Qu.: 430.72       1st Qu.: 214.50
##  Median : 663.83       Median : 368.03
##  Mean   : 740.42       Mean   : 453.82
##  3rd Qu.:1004.97       3rd Qu.: 573.97
##  Max.   :2638.18       Max.   :2226.79
```

```
multi.hist(data_attempt_3)
```

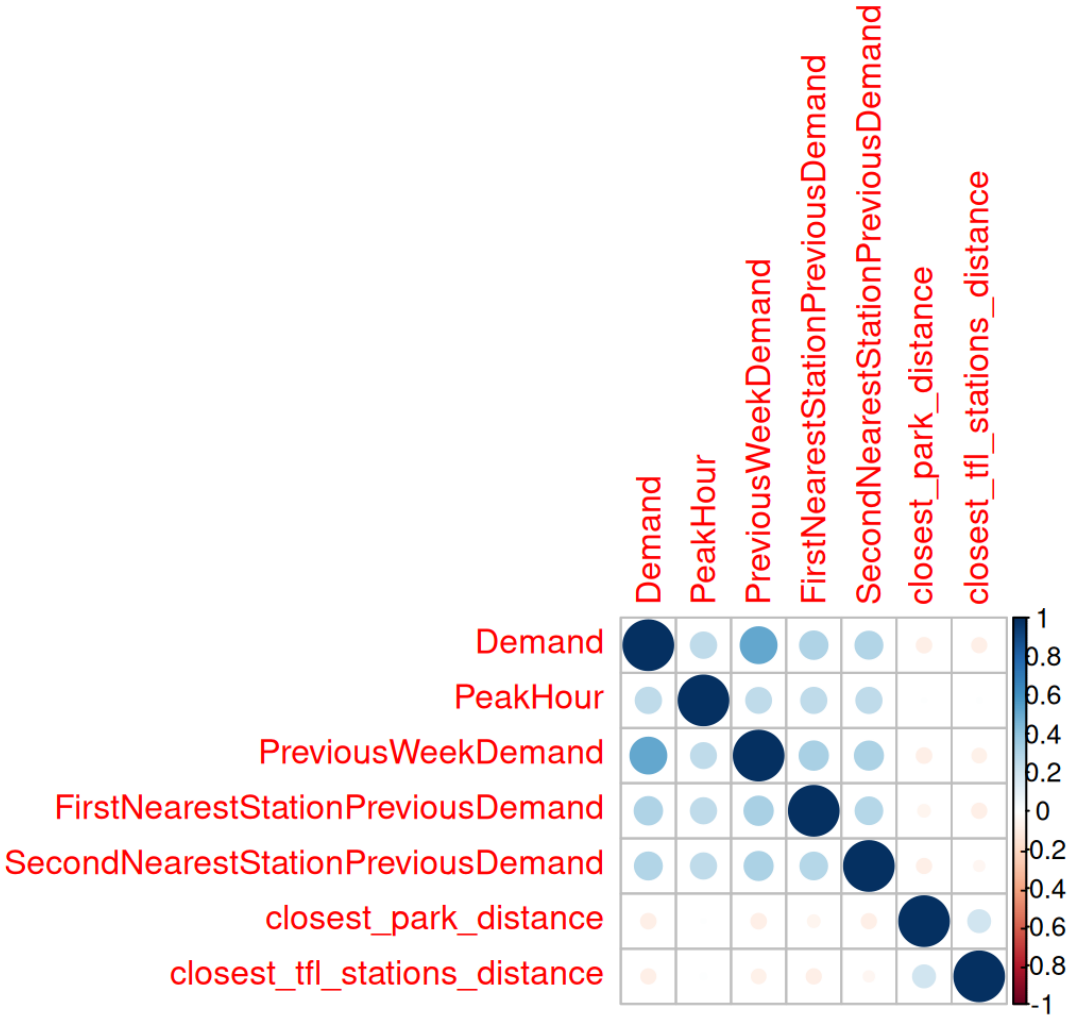



```
data_std_attempt_3 = as.data.table(scale(data_attempt_3))
summary(data_std_attempt_3)
```

```
##      Demand             PeakHour       PreviousWeekDemand
##  Min.   :-1.18755   Min.   :-0.6529   Min.   :-1.18024
##  1st Qu.:-0.54248   1st Qu.:-0.6529   1st Qu.:-0.53761
##  Median :-0.08479   Median :-0.6529   Median :-0.08166
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.00000
##  3rd Qu.: 0.56028   3rd Qu.: 1.5316   3rd Qu.: 0.56097
##  Max.   : 5.99768   Max.   : 1.5316   Max.   : 5.97778
##  FirstNearestStationPreviousDemand SecondNearestStationPreviousDemand
##  Min.   :-1.15013                  Min.   :-1.15939
##  1st Qu.:-1.15013                  1st Qu.:-1.15939
##  Median :-0.03059                  Median :-0.04253
##  Mean   : 0.00000                  Mean   : 0.00000
##  3rd Qu.: 0.62430                  3rd Qu.: 0.61078
##  Max.   : 5.91737                  Max.   : 6.11771
##  closest_park_distance closest_tfl_stations_distance
##  Min.   :-1.6393       Min.   :-1.2408
##  1st Qu.:-0.7292       1st Qu.:-0.6678
##  Median :-0.1804       Median :-0.2394
##  Mean   : 0.0000       Mean   : 0.0000
##  3rd Qu.: 0.6229       3rd Qu.: 0.3352
##  Max.   : 4.4685       Max.   : 4.9471
```

Checking multicollinearity.

```
library(corrplot)
corrplot(cor(data_std_attempt_3))
```



Algorithms

Goal requires to predict number of rented bikes so we have to implement regression model.

```
set.seed(0)
trainIdx_attempt_3 = sample(1:nrow(data_std_attempt_3), 0.75*nrow(data_std_attempt_
3))
train_attempt_3 = data_std_attempt_3[trainIdx_attempt_3]
test_attempt_3 = data_std_attempt_3[-trainIdx_attempt_3]
```

```
lr_attempt_3 = lm(Demand ~ ., data=train_attempt_3)
train_preds_attempt_3 = predict(lr_attempt_3, train_attempt_3)
test_preds_attempt_3 = predict(lr_attempt_3, test_attempt_3)
print(paste("R2 on train:", cor(train_preds_attempt_3, train_attempt_3$Demand)^2))
```

```
## [1] "R2 on train: 0.313705297212082"
```

```
print(paste("R2 on test:", cor(test_preds_attempt_3, test_attempt_3$Demand)^2))
```

```
## [1] "R2 on test: 0.306043796726195"
```

```
print(paste("RMSE on train:", rmse(train_preds_attempt_2, train_attempt_2$Demand)))
```

```
## [1] "RMSE on train: 0.854155885015494"
```

```
print(paste("RMSE on test:", rmse(test_preds_attempt_2, test_attempt_2$Demand)))
```

```
## [1] "RMSE on test: 0.854012468056662"
```

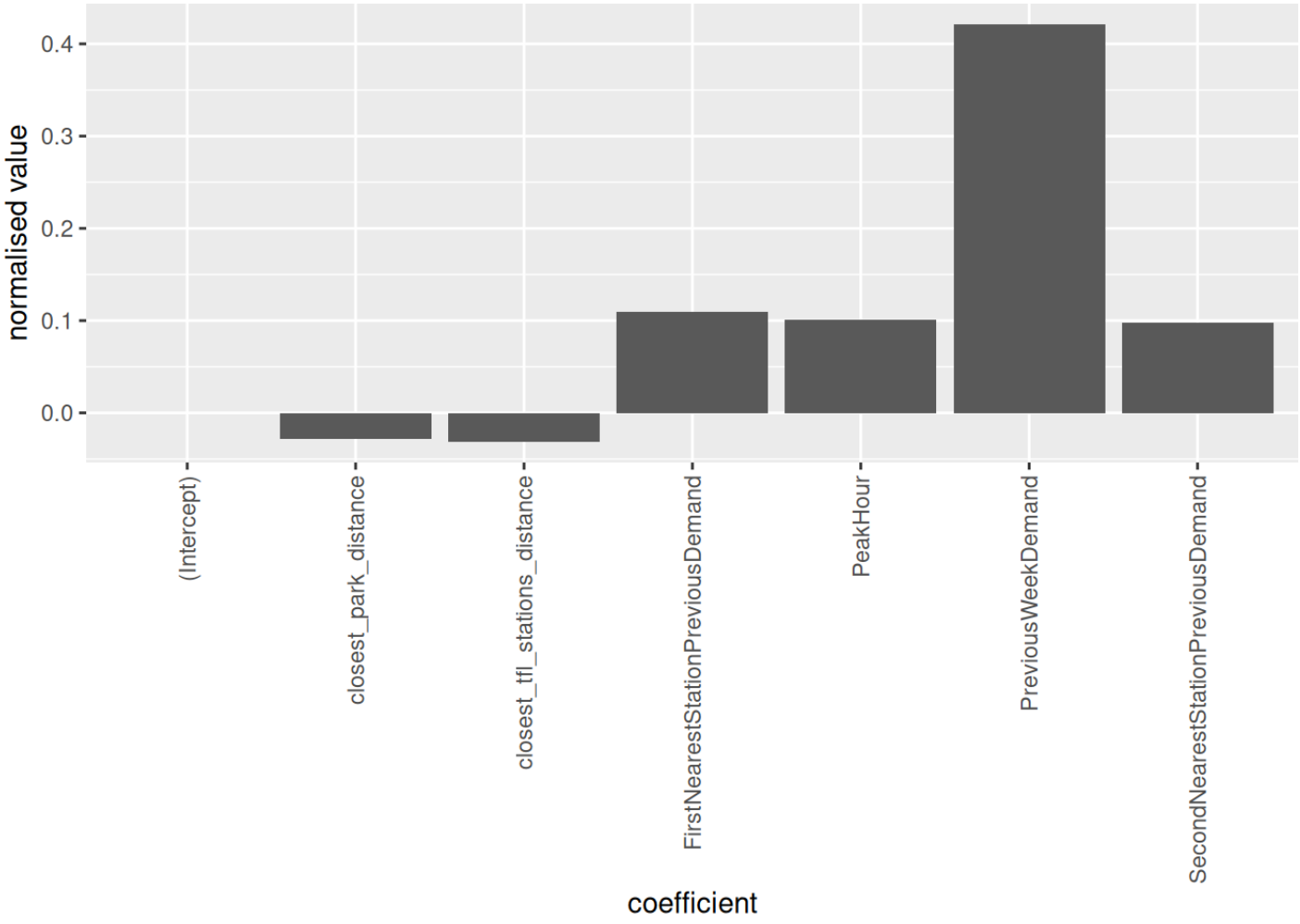
There's no difference between train and test scores which indicates that our model is stable. We do not overfit which means we don't need any regularisation.

Data understanding

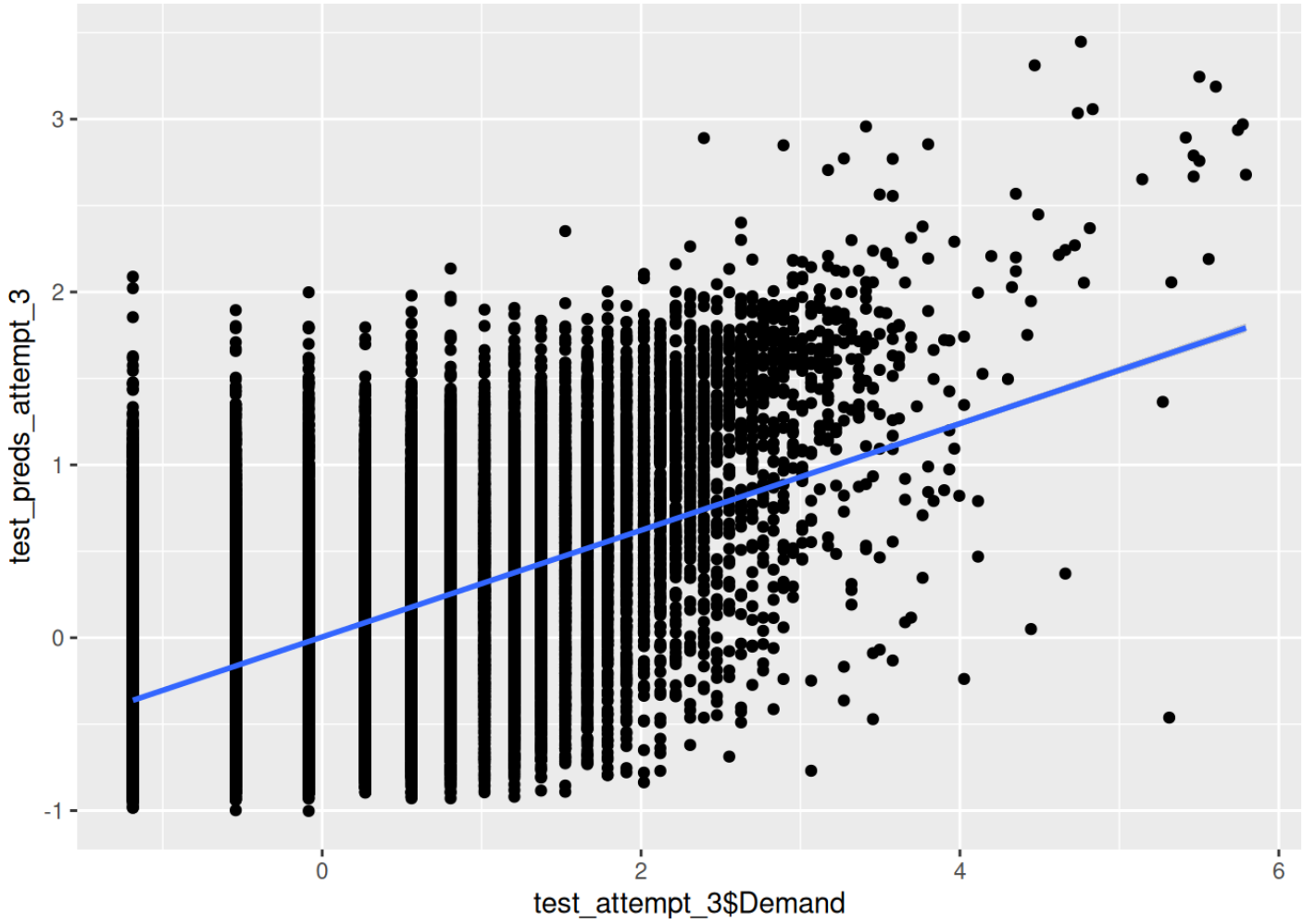
```
lr_attempt_3 = lm(Demand ~ ., data=data_std_attempt_3)
summary(lr_attempt_3)
```

```
##
## Call:
## lm(formula = Demand ~ ., data = data_std_attempt_3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2653 -0.6057 -0.0278  0.5687  5.7747
##
## Coefficients:
##                                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     -4.565e-14  2.022e-03    0.00        1
## PeakHour                         1.014e-01  2.159e-03   46.97   <2e-16 ***
## PreviousWeekDemand               4.214e-01  2.237e-03  188.40   <2e-16 ***
## FirstNearestStationPreviousDemand  1.097e-01  2.217e-03   49.48   <2e-16 ***
## SecondNearestStationPreviousDemand  9.761e-02  2.213e-03   44.12   <2e-16 ***
## closest_park_distance           -2.861e-02  2.071e-03  -13.82   <2e-16 ***
## closest_tfl_stations_distance   -3.113e-02  2.069e-03  -15.04   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8296 on 168305 degrees of freedom
## Multiple R-squared:  0.3118, Adjusted R-squared:  0.3118
## F-statistic: 1.271e+04 on 6 and 168305 DF,  p-value: < 2.2e-16
```

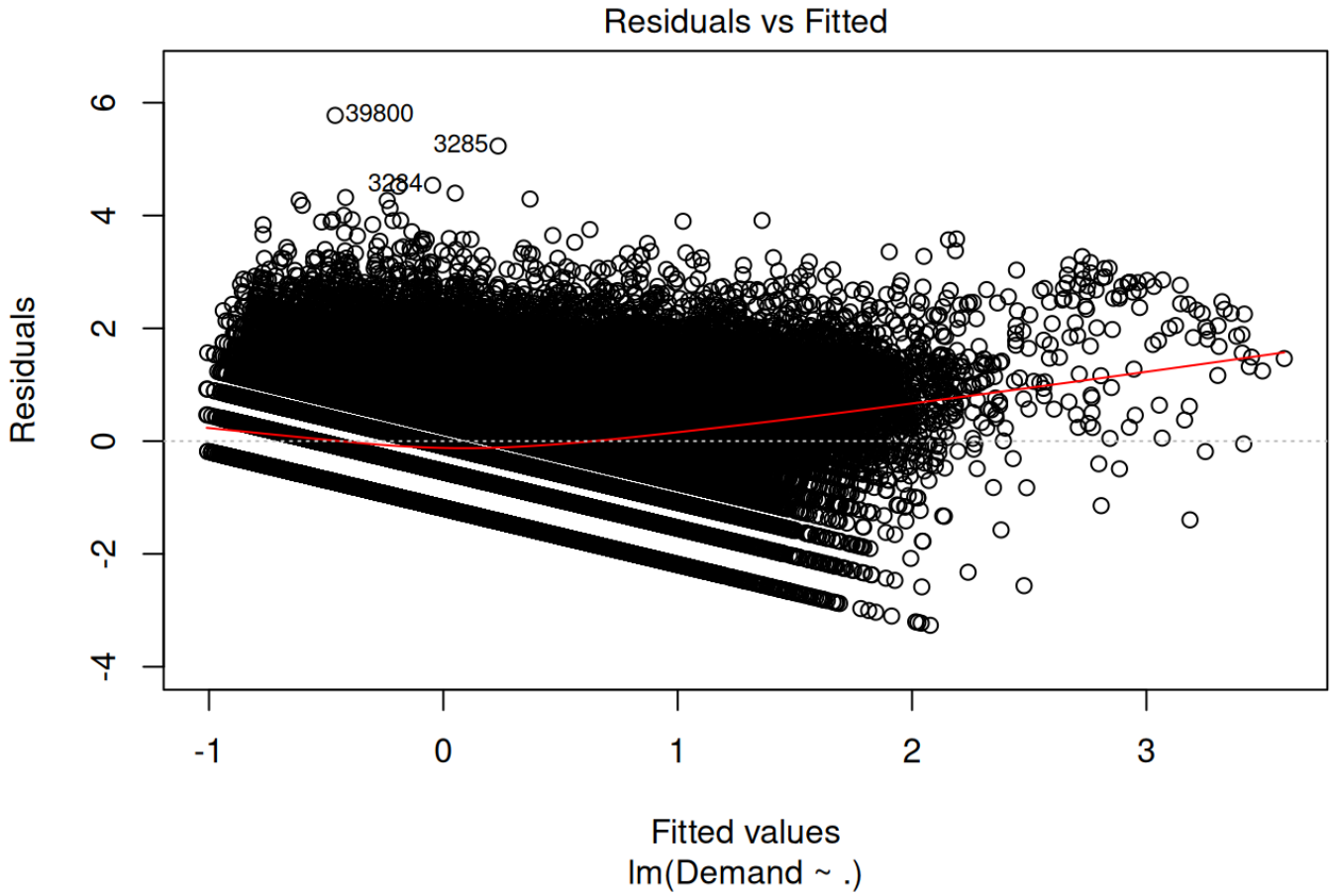
```
library(ggplot2)
ggplot(, aes(x = names(lr_attempt_3$coefficients), y=lr_attempt_3$coefficients)) +
  geom_bar(stat="identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5)) +
  xlab("coefficient") +
  ylab("normalised value")
```

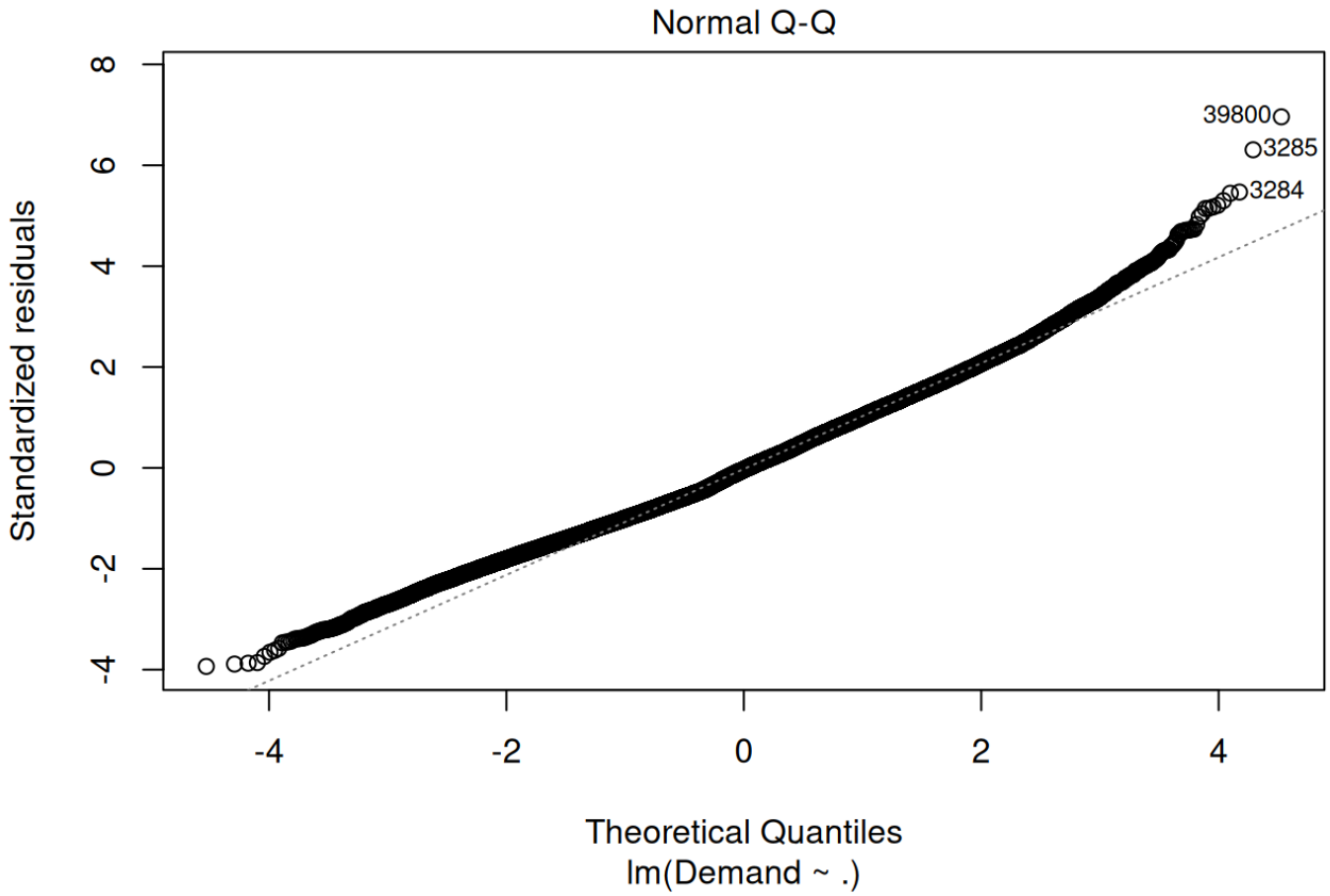


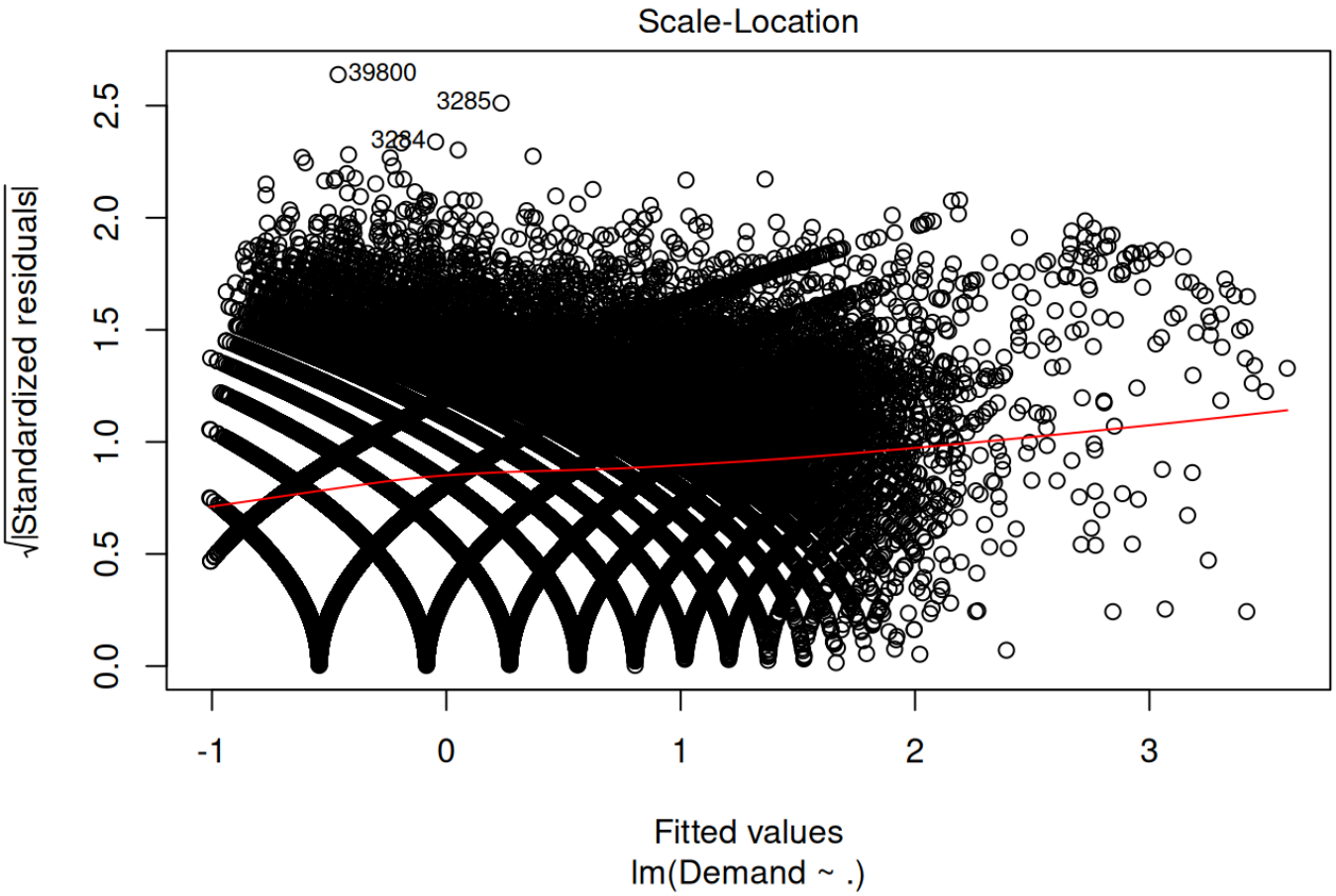

```
ggplot(test_attempt_3, aes(x = test_attempt_3$Demand, y=test_preds_attempt_3)) +
  geom_point() +
  geom_smooth(method = "lm")
```

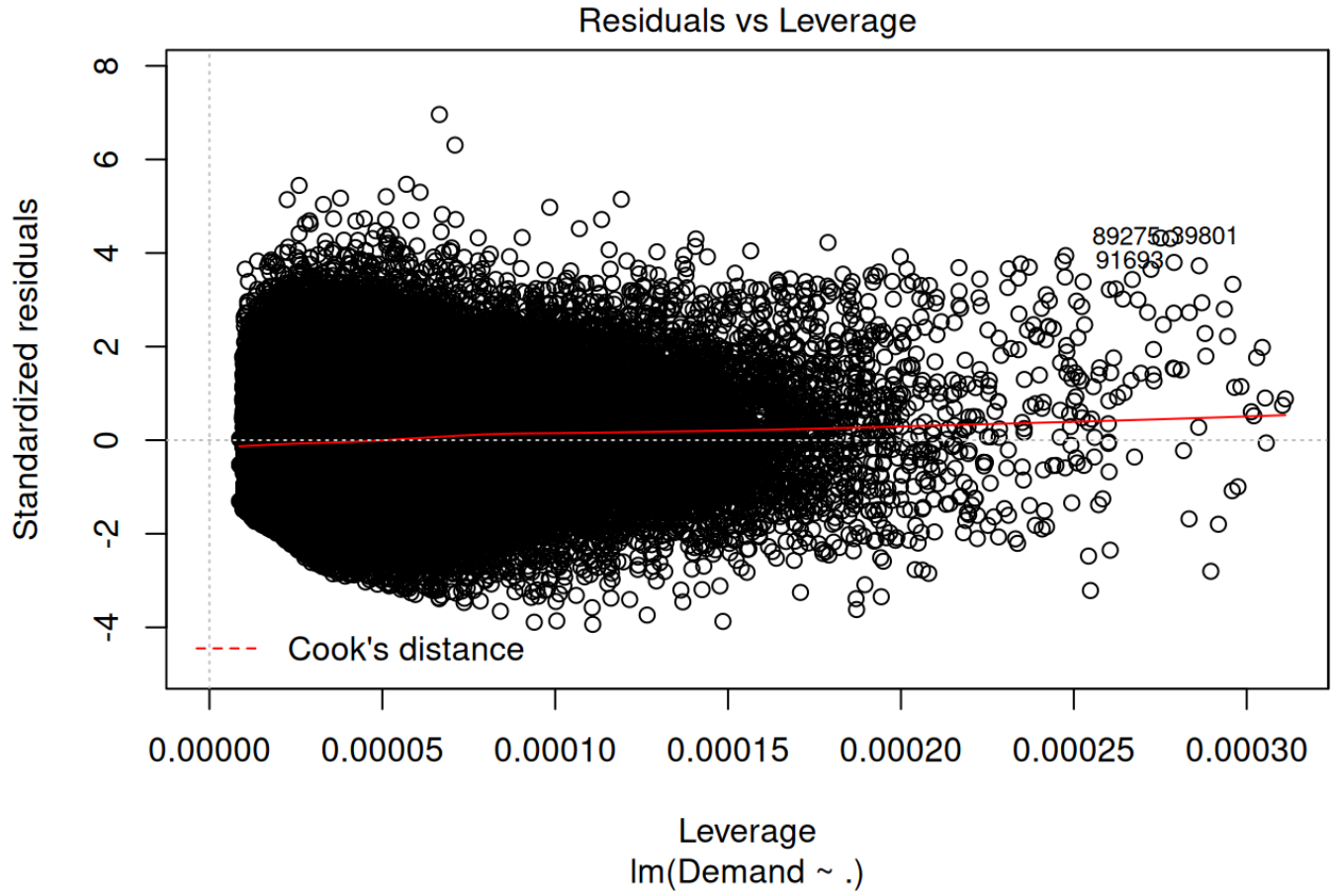



```
plot(lr_attempt_3)
```

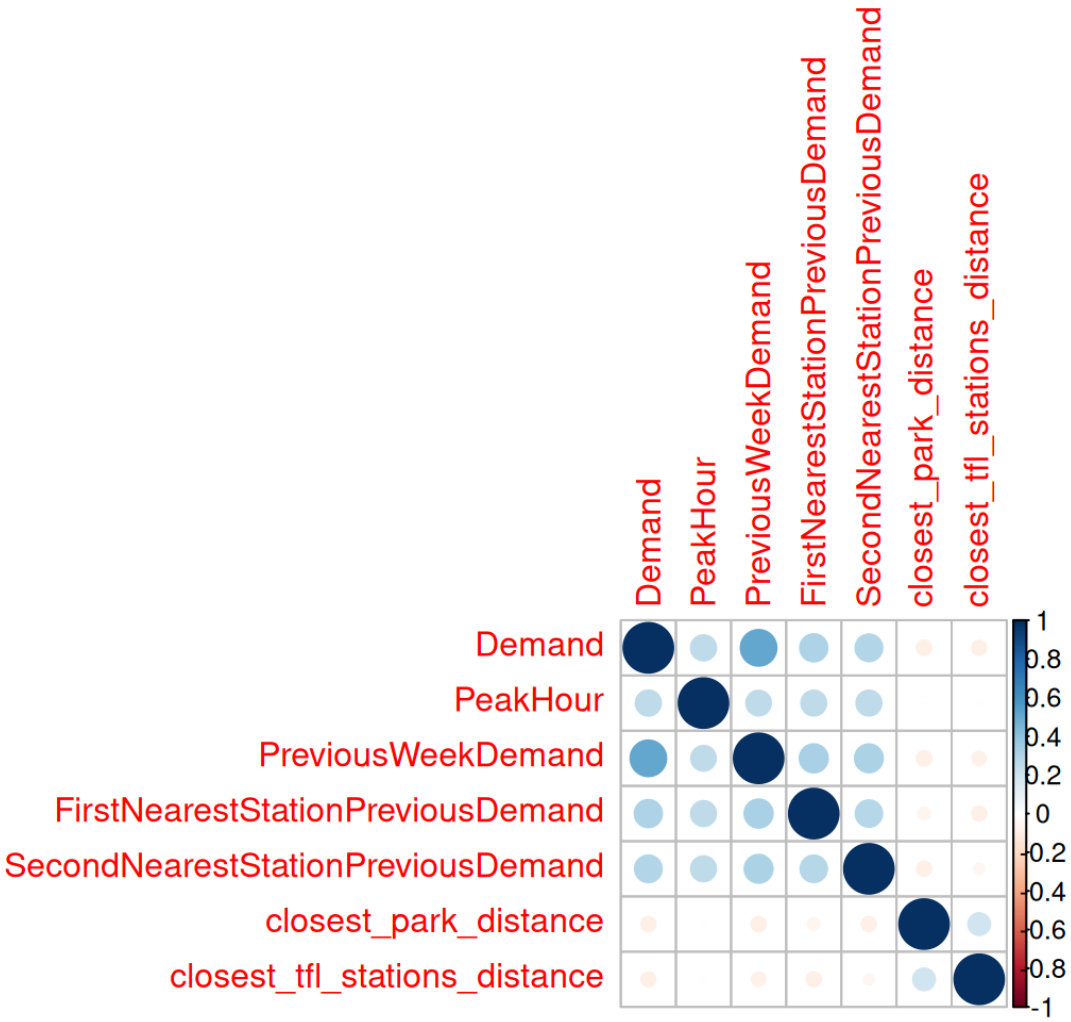

Residuals vs Fitted
39800
3285
3284
Residuals
6
4
2
0
-2
-4
-1
0
1
2
3
Fitted values
lm(Demand ~ .)


Normal Q-Q
39800
3285
3284
Standardized residuals
8
6
4
2
0
-2
-4
-4
-2
0
2
4
Theoretical Quantiles
lm(Demand ~ .)


Scale-Location
39800
3285
3284
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
-1
0
1
2
3
Fitted values
lm(Demand ~ .)


Residuals vs Leverage
89275
39801
91693
Standardized residuals
8
6
4
2
0
-2
-4
Cook's distance
0.00000
0.00005
0.00010
0.00015
0.00020
0.00025
0.00030
Leverage
lm(Demand ~ .)

```
corrplot(cor(data_attempt_3))
```



H1. Higher demand of bikes rented during peak hours 8 and 17 to 18. *True*
H12. Bikes demand rented at each station depends on previous demand at the same station. *True*
H13. Bikes demand rented at each station depends on previous demand at the closest stations. *True*
H14. Higher demand of bikes rented at stations which are close to the tfl station. *False*
H15. Higher demand of bikes rented at stations which are close to the park. *False*

Limitation

Almost 31% of the variation of the dependent variable is described by independent variables. This means it is still not good model.