Project

Charan Basireddy, Bala Aakash Malepati, Mary Sravani Thamtam

2023-04-25

Abstract

Airbnb has disrupted the hospitality industry by providing a platform for short-term rentals. Understanding the factors that determine Airbnb prices is essential for both hosts and guests. This report explores the determinants of Airbnb prices in Europe using data from the publicly available dataset 'Airbnb Price Determinants' from Kaggle. In this study, we have explored linear and polynomial regression and random forest models to determine the price of a room and found the main determinants based on the coefficient values.

Introduction

Airbnb has grown rapidly in Europe, providing a popular alternative to traditional hotels and accommodations. The platform enables hosts to rent out their homes or rooms to guests, often at lower prices than hotels. Airbnb offers guests the opportunity to experience local neighborhoods and culture more authentically. For hosts, Airbnb provides a source of income and an opportunity to meet people worldwide. However, Airbnb prices can vary widely, and understanding the factors determining these prices is important for hosts and guests.

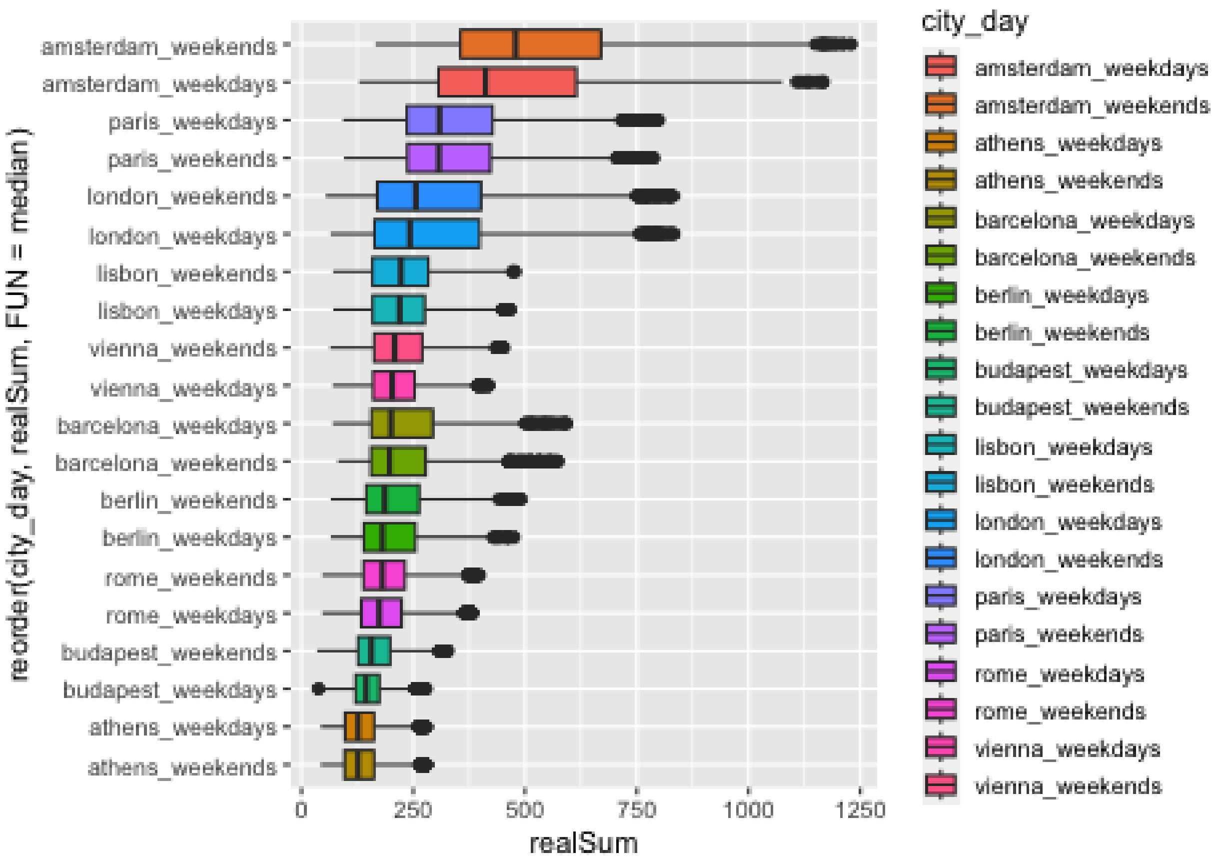
How can this information be used: Data can help travelers find accommodation that meets their needs without exceeding budget. Can help hosts set competitive pricing and optimize listings to get more bookings. Help investors evaluate the value of investing in real estate in different European cities based on pricing trends.

Methods

To explore the determinants of Airbnb prices in Europe, we used data on Airbnb listings in ten major European cities (Amsterdam, Athens, Barcelona, Berlin, Budapest, Lisbon, London, Paris, Rome, Vienna). The data consists of various attributes about the hotel. We used regression analysis and random forest to determine the factors that are strongly associated with Airbnb prices in Europe.

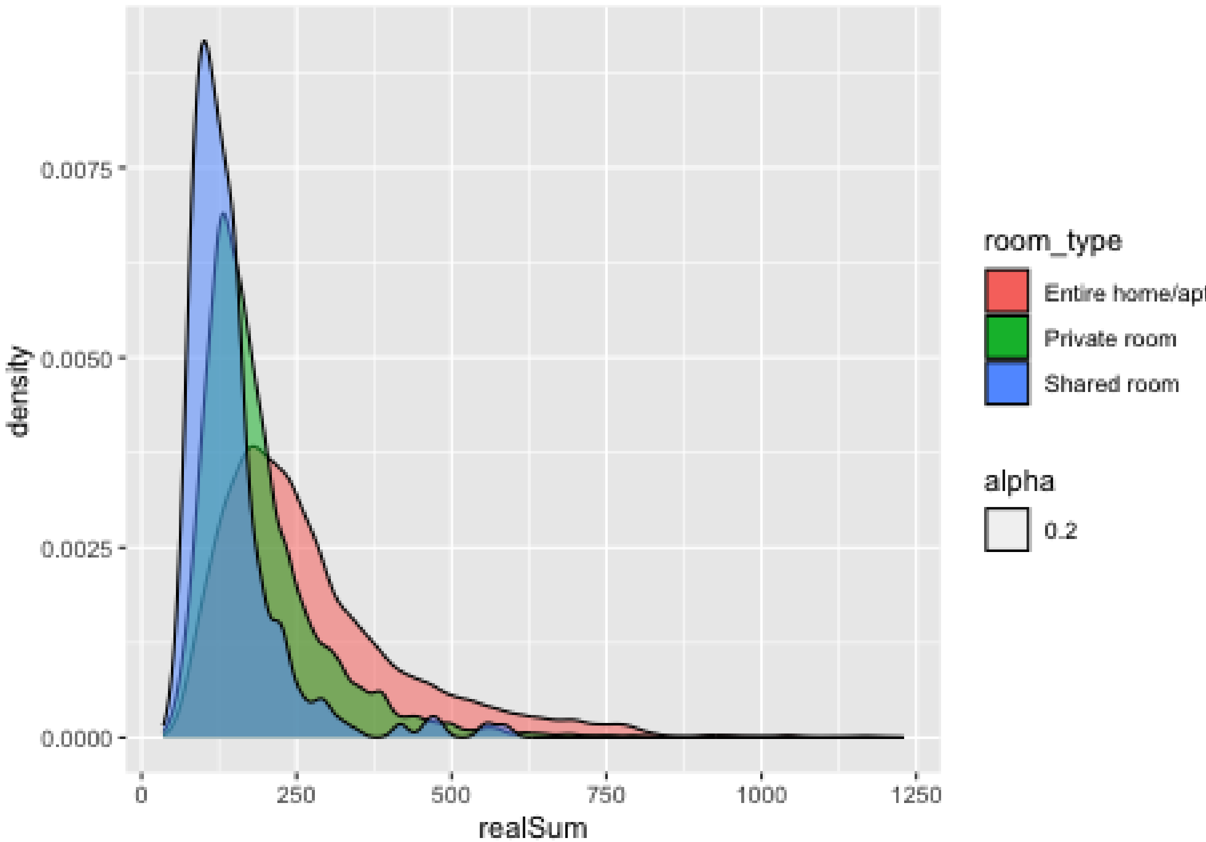
Exploratory Data Analysis

We did EDA for the following variables



$Figure 1$

The highest prices in Europe are found in Amsterdam.



$Figure 2$

The prices of entire home are high comparatively

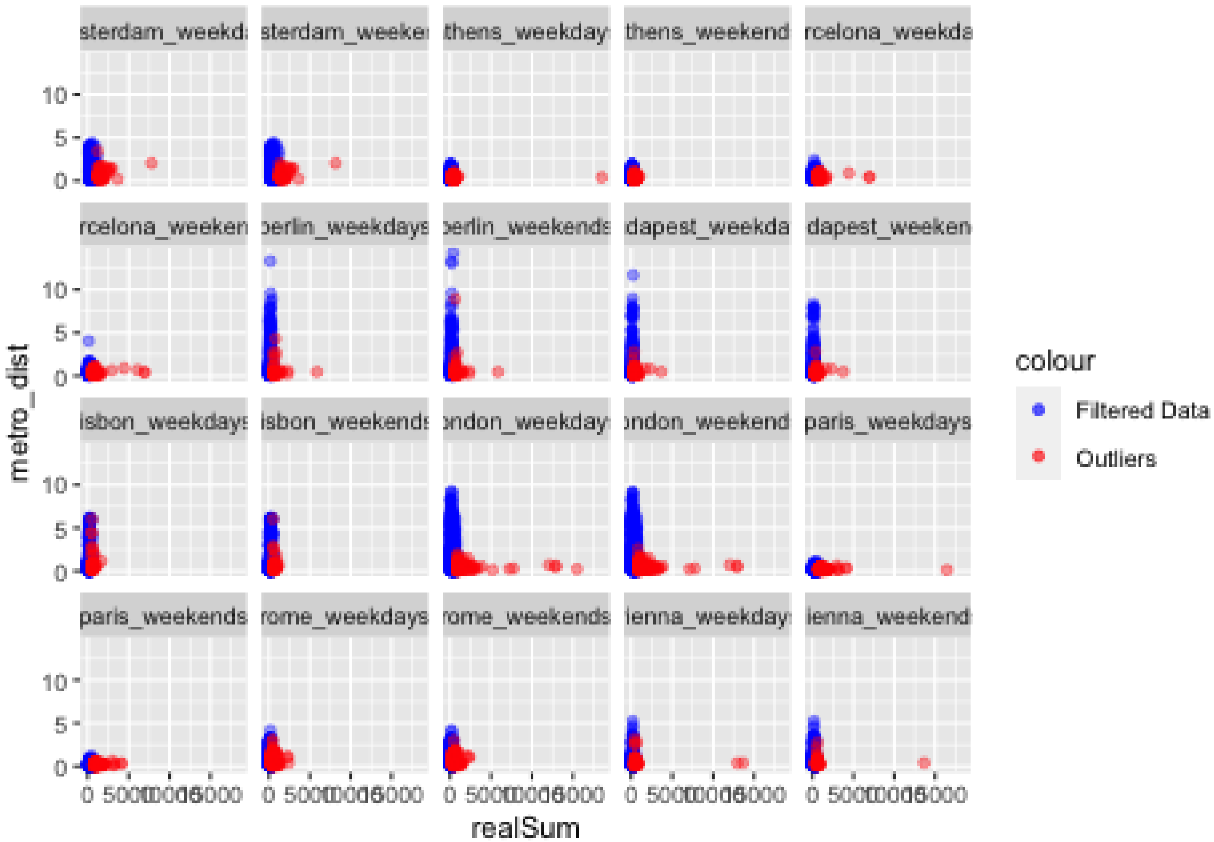
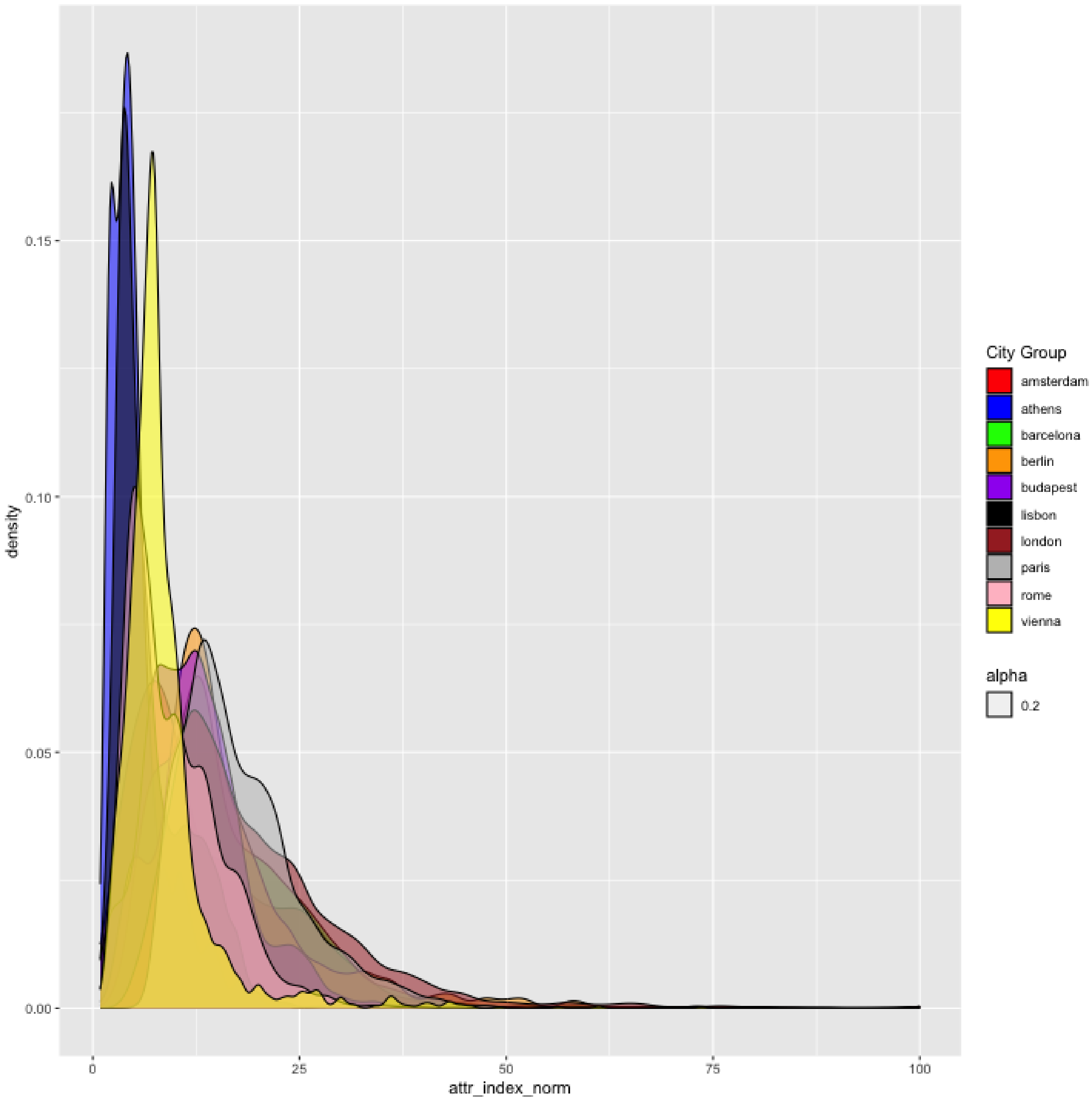


Figure3

In general the rooms that are closer to metro have comparatively higher prices. But, in Rome city the distance to metro is almost same for both categories of price.



$Figure 4$

The attractive index of cities is varying across different cities it is higher for some cities like Paris.

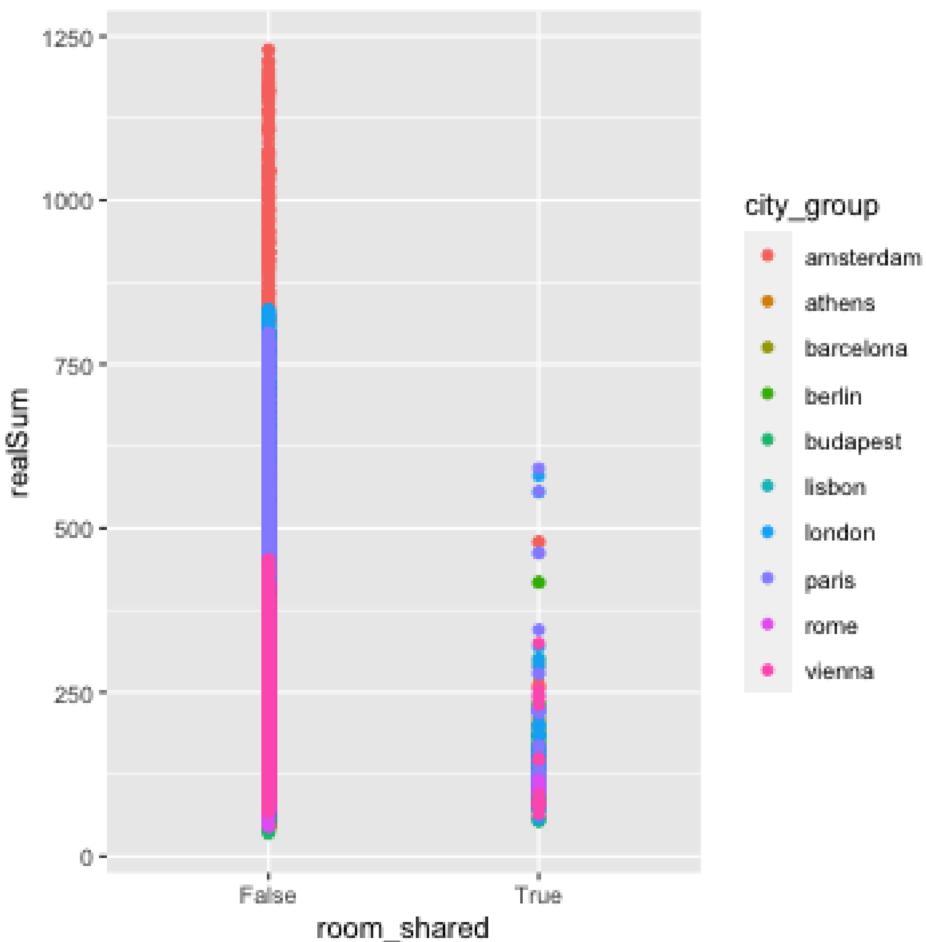


*Figure*5

There are differences in room prices between both share and non-shared rooms.

Pre Processing

The room_shared and room_private information is already embedded in room_type. The variables are multi-collinear, so we have removed room_shared and room_private.

Multivariate linear regression

Multivariate linear regression is a statistical method that models the relationship between multiple input variables and a continuous output variable. A linear equation is fitted to the input variables, with coefficients representing each input variable's contribution to the output variable. The model assumes that the relationship between the input and output variables is linear and that errors are normally distributed and independent.

Polynomial regression

In polynomial regression, the input variables are raised to various powers. The degree of the polynomial determines the complexity of the model and the number of input variables used in the model. Polynomial regression is useful when the relationship between the input variables and the output variable is not linear and can provide a better fit to the data than linear regression.

Interaction Variables

Interaction variables, also known as interaction effects, refer to the impact that the combination of two or more input variables has on the output variable in a regression model. They capture the non-additive relationship between input variables and can help to explain better the relationship between the input variables and the output variable. The interaction variables are created by multiplying the values of two or more input variables and including them as additional terms in the model. The coefficient of an interaction variable represents the change in the response variable for a one-unit change in one input variable while holding the other input variables constant.

Random forest regression

Random forest regression is a machine learning technique that combines the power of decision trees with the concept of ensemble learning.

Decision trees work by recursively partitioning the input data into subsets based on the values of the input features to minimize the variance of the target variable within each subgroup. The result is a tree-like structure representing a set of rules for predicting the target variable. Each internal node in the tree corresponds to a test on one of the input features, and each leaf node corresponds to a predicted target variable value. During the prediction phase, the input data is traversed down the tree according to the rules represented by the internal nodes until a leaf node is reached, which provides the predicted value for the target variable. Decision trees handle non-linear relationships between features and the target variable. However, it is prone to overfitting, especially when the tree is deep, and can be sensitive to small changes in the input data.

In Random Forest, multiple decision trees are trained on different subsets of the input data, and the results are combined to make predictions. Each tree in the forest is trained on a random subset of the available features, which helps to reduce overfitting and increase the model's generalization performance. During the prediction phase, the output of each tree is aggregated to produce a final prediction.

Data

Each major city has its own dataset for weekend and weekdays Variables included in data set:

- Host ID (Id)
- Total price of listing (realSum)
- Room type: private, shared, entire home, apt (room_type)
- Whether or not room is shared (room_shared)
- Max number of people allowed in property (person_capacity)
- Whether or not host is superbost (host_is_superhost)
- Whether or not it is multiple rooms (multi)
- Whether for business or family use (biz)
- Distance from city center (dist)
- Distance from nearest metro (metro_dist)
- Latitude and longitude (lat lng)
- Guest satisfaction (guest_satisfaction_overall)
- Cleanliness (cleanliness_rating)
- Total quantity of bedrooms available among all properties for single host (bedrooms)
- Index of Attractions near the hotel (attr_index)

- Normalized Index of Attractions near the hotel (attr_index_norm)
- Index Restaurants near the hotel (rest_index)
- Normalized Index of Restaurants near the hotel (rest_index_norm)

The dataset consists of

- Continuous variables : realSum, dist, metro_dist, lat, lng, attr_index, attr_index_norm, rest_index, rest_index_norm
- Ordinal : person_capacity, guest_satisfaction_overall, cleanliness_rating, bedrooms
- Nominal : room_type, room_shared, host_is_superhost, multi, biz

Results

We have modeled multivariate regression for each city, and below are the coefficients of each model arranged in descending order. The coefficient with a larger value is the essential determinant of the hotel room price.

According to the table, all the models have the same descending order of coefficients. The order is as follows:

1 - room_type 2 - person_capacity 3 - host_is_superhost 4 - multi 5 - biz 6 - cleanliness_rating 7 - guest_satisfaction_overall 8 - bedrooms 9 - dist 10 - metro_dist 11 - attr_index_norm 12 - rest_index_norm, 13 - lng 14 - lat

Modelling

MVLR Seperated by City and Day

Model	var_names	var_coefs	Model	var_names	var_coefs	Model	var_names	var_coefs	Model	var_names	var_coefs	Model	var_names	var_coefs
M_0	(Intercept)	453.805963	M_1	(Intercept)	677.7774655	M_2	(Intercept)	81087.31478	M_3	(Intercept)	3313.326331	M_4	(Intercept)	14214.11864
M_0	room_typePrivate room	384.5953101	M_1	room_typePrivate room	95.38419868	M_2	room_typePrivate room	148.9417279	M_3	room_typePrivate room	83.28285599	M_4	room_typePrivate room	25.47757592
M_0	room_typeShared room	136.0046307	M_1	room_typeShared room	42.51664607	M_2	room_typeShared room	78.35566832	M_3	room_typeShared room	32.55688542	M_4	room_typeShared room	12.36521125
M_0	person_capacity	119.9608563	M_1	person_capacity	31.890407	M_2	person_capacity	38.86525794	M_3	person_capacity	32.36241001	M_4	person_capacity	6.824991537
M_0	host_is_superhostTrue	42.08786555	M_1	host_is_superhostTrue	16.91133475	M_2	host_is_superhostTrue	18.18910642	M_3	host_is_superhostTrue	20.77860946	M_4	host_is_superhostTrue	5.296527081
M_0	multi	20.6527087	M_1	multi	4.421357529	M_2	multi	12.07097956	M_3	multi	12.77790391	M_4	multi	1.652498502
M_0	biz	16.61275571	M_1	biz	2.725004443	M_2	biz	6.985320078	M_3	biz	6.194671034	M_4	biz	0.462524562
M_0	cleanliness_rating	3.757397212	M_1	cleanliness_rating	2.640302855	M_2	cleanliness_rating	2.492178554	M_3	cleanliness_rating	4.446168901	M_4	cleanliness_rating	-0.492631965
M_0	guest_satisfaction_overall	2.738196195	M_1	guest_satisfaction_overall	2.335020293	M_2	guest_satisfaction_overall	1.774791262	M_3	guest_satisfaction_overall	1.658915477	M_4	guest_satisfaction_overall	-2.915525964
M_0	bedrooms	2.025200616	M_1	bedrooms	1.221665363	M_2	bedrooms	0.215433391	M_3	bedrooms	0.831750876	M_4	bedrooms	-13.33265099
M_0	dist	-2.83720865	M_1	dist	1.053387666	M_2	dist	0.055349136	M_3	dist	0.423821302	M_4	dist	-13.38768559
M_0	metro_dist	-7.074399447	M_1	metro_dist	-23.4210691	M_2	metro_dist	-9.749578513	M_3	metro_dist	0.346180902	M_4	metro_dist	-17.02253017
M_0	attr_index_norm	-21.61855664	M_1	attr_index_norm	-24.66793817	M_2	attr_index_norm	-302.2199647	M_3	attr_index_norm	-39.28949375	M_4	attr_index_norm	-47.92355904
M_0	rest_index_norm	-60.95450514	M_1	rest_index_norm	-29.95296764	M_2	rest_index_norm	-311.0956042	M_3	rest_index_norm	-102.8898274	M_4	rest_index_norm	-48.61561818
M_0	lng	-175.0198029	M_1	lng	-65.86473148	M_2	lng	-384.8001582	M_3	lng	-120.4611811	M_4	lng	-114.5660309
M_0	lat	-340.2468974	M_1	lat	-28023.54948	M_2	lat	-1939.911175	M_3	lat	-221.3241991	M_4	lat	-251.7212566
M_5	(Intercept)	24.49430744	M_6	(Intercept)	6546.714298	M_7	(Intercept)	41416.99566	M_8	(Intercept)	428.4623235	M_9	(Intercept)	408.940146
M_5	room_typePrivate room	20.76733214	M_6	room_typePrivate room	163.4989324	M_7	room_typePrivate room	130.3445018	M_8	room_typePrivate room	44.47609905	M_9	room_typePrivate room	88.51887853
M_5	room_typeShared room	8.685270433	M_6	room_typeShared room	36.38305957	M_7	room_typeShared room	81.79791623	M_8	room_typeShared room	12.32210245	M_9	room_typeShared room	36.04103653
M_5	person_capacity	5.728057336	M_6	person_capacity	25.54307154	M_7	person_capacity	56.22439359	M_8	person_capacity	3.882518636	M_9	person_capacity	20.32350711
M_5	host_is_superhostTrue	4.819142536	M_6	host_is_superhostTrue	12.82925563	M_7	host_is_superhostTrue	43.15533016	M_8	host_is_superhostTrue	3.699383339	M_9	host_is_superhostTrue	17.90834436
M_5	multi	1.238674167	M_6	multi	9.982245594	M_7	multi	25.20968259	M_8	multi	3.695177412	M_9	multi	4.390508977
M_5	biz	0.374695799	M_6	biz	0.933858045	M_7	biz	12.5560253	M_8	biz	1.475050781	M_9	biz	1.60480035
M_5	cleanliness_rating	0.177646654	M_6	cleanliness_rating	0.207399315	M_7	cleanliness_rating	8.107228556	M_8	cleanliness_rating	0.716854353	M_9	cleanliness_rating	1.352556262
M_5	guest_satisfaction_overall	0.130448297	M_6	guest_satisfaction_overall	0.113545147	M_7	guest_satisfaction_overall	7.409335312	M_8	guest_satisfaction_overall	0.513052541	M_9	guest_satisfaction_overall	1.242051249
M_5	bedrooms	-8.337627302	M_6	bedrooms	-19.2800813	M_7	bedrooms	1.641809145	M_8	bedrooms	-0.225693578	M_9	bedrooms	-12.84826556
M_5	dist	-12.56848529	M_6	dist	-21.77143781	M_7	dist	1.3085809	M_8	dist	-0.783530606	M_9	dist	-13.17443006
M_5	metro_dist	-59.19710383	M_6	metro_dist	-22.96385155	M_7	metro_dist	-4.603578145	M_8	metro_dist	-1.508888739	M_9	metro_dist	-13.85744129
M_5	attr_index_norm	-82.58885424	M_6	attr_index_norm	-130.3824975	M_7	attr_index_norm	-97.88170929	M_8	attr_index_norm	-6.252414347	M_9	attr_index_norm	-16.13640861
M_5	rest_index_norm	-190.4190079	M_6	rest_index_norm	-181.4255816	M_7	rest_index_norm	-284.4850304	M_8	rest_index_norm	-39.8036185	M_9	rest_index_norm	-28.48935415
M_5	lng	-474.7694686	M_6	lng	-193.4634265	M_7	lng	-715.552286	M_8	lng	-94.72375311	M_9	lng	-120.2455734
M_5	lat	-2021.541971	M_6	lat	-212.9519776	M_7	lat	-821.2043548	M_8	lat	-17868.39131	M_9	lat	-20955.97861

M_0 - Amsterdam, M_1 - Athens, M_2 - Barcelona, M_3 - Berlin, M_4 - Budapest, M_5 - Lisbon, M_6 - London, M_7 - Paris, M_8 - Rome, M_9 - Vienna

MVLR Combined of all Cities

term	estimate	std.error	statistic	p.value
city_dayathens_weekdays	6315.09061077375	1388.53511239174	4.54802370816251	5.43276632399387E-06
city_dayathens_weekends	6303.53106566887	1388.65271393214	4.53931425937278	5.66191686814906E-06
city_daybudapest_weekends	3929.67339172125	706.843976888775	5.55946364432218	2.72512019627685E-08
city_daybudapest_weekdays	3902.35110769949	706.88057311247	5.52052391327863	3.40308057888361E-08
city_dayvienna_weekdays	3231.8185121282	582.709191916445	5.54619449454577	2.93992522226129E-08
city_dayvienna_weekends	3230.26798535313	582.79716507451	5.54269680591213	2.9992285987412E-08
city_dayrome_weekends	2952.53521434873	881.429677236262	3.34971159991624	0.000809784171597337
city_dayrome_weekdays	2947.68523500212	881.398370960292	3.34432798167143	0.00082565958973142
city_dayberlin_weekends	1958.88440687424	342.040071097647	5.72706116154153	1.02993861536248E-08
city_dayberlin_weekdays	1949.04242701527	342.124495503847	5.69688067539542	1.22965025210763E-08
city_daybarcelona_weekends	429.652941610428	837.810922281234	0.512828050081453	0.608074739358428
city_daybarcelona_weekdays	411.771664434536	837.79087513094	0.491496955454641	0.623077988219532
lat	123.211714028539	76.5227515358753	1.61013177853093	0.107377819824515
bedrooms	86.0154191816913	3.18878180193731	26.9743822325609	1.10487900248437E-158
city_dayamsterdam_weekends	67.9410312574828	16.0017245274658	4.24585682254852	2.18302692822801E-05
biz	33.2805902980968	4.18846537329009	7.94577185962373	1.98543351601859E-15
person_capacity	23.962639711116	1.76448699475381	13.580513646381	6.6229089695509E-42
multi	9.60044781984104	4.13239391830793	2.32321700438764	0.0201730153901032
attr_index_norm	6.37045498927847	0.294558501285674	21.6271299639054	4.50013019203766E-103
cleanliness_rating	5.03832622544896	2.41532715274561	2.08598086587221	0.0369873477630624
host_is_superhostTrue	1.07487304905057	3.93436342704325	0.273201260885642	0.784700071060939
guest_satisfaction_overall	0.776010513381954	0.261452270487271	2.96807716351323	0.00299865383859512
rest_index_norm	-0.183747213632232	0.177364361505566	-1.03598723031214	0.300215028045804
dist	-1.53304759323782	1.26282244844453	-1.21398506585478	0.224761355845419
metro_dist	-3.99666546744126	2.50252382009665	-1.59705391626878	0.110262427719338
room_typePrivate room	-114.36546946963	4.28228402903422	-26.7066520329393	1.28065674381786E-155
room_typeShared room	-204.184154195175	18.9348057155667	-10.7835357416586	4.52732676348359E-27
lng	-262.890929428664	40.1930553101859	-6.54070528850891	6.20447859714726E-11
city_dayparis_weekdays	-403.028852808039	278.481924100759	-1.44723523478032	0.147839720417429
city_dayparis_weekends	-422.138907209481	278.643708051156	-1.51497735284222	0.129786880370265
city_daylondon_weekdays	-1409.29965386078	206.104550214087	-6.83779010408502	8.17001208243907E-12
city_daylondon_weekends	-1410.93275121758	206.122303369817	-6.84512412364295	7.76270471372057E-12
city_daylisbon_weekends	-2304.00669656451	1143.81264744771	-2.0143217525227	0.0439831538834079
city_daylisbon_weekdays	-2312.9309439741	1143.89771522079	-2.02197356738986	0.0431864319357264
(Intercept)	-4954.90756116218	3996.18236913533	-1.2399102702198	0.215016631100909

Apart from Cityday, below are the ranking of coefficients of the model

1. Lat, 2. bedrooms, 3. biz, 4. person_capacity, 5. Multi, 6. Attr_index_norm, 7. Cleanliness_rating,

8. host_is_superhost, 9. guest_satisfaction_overall, 10. rest_index_norm 11. dist, 12. metro_dist, 13. room_type, 14. lng

The city day coefficients are almost ranked at the top.

Other Models

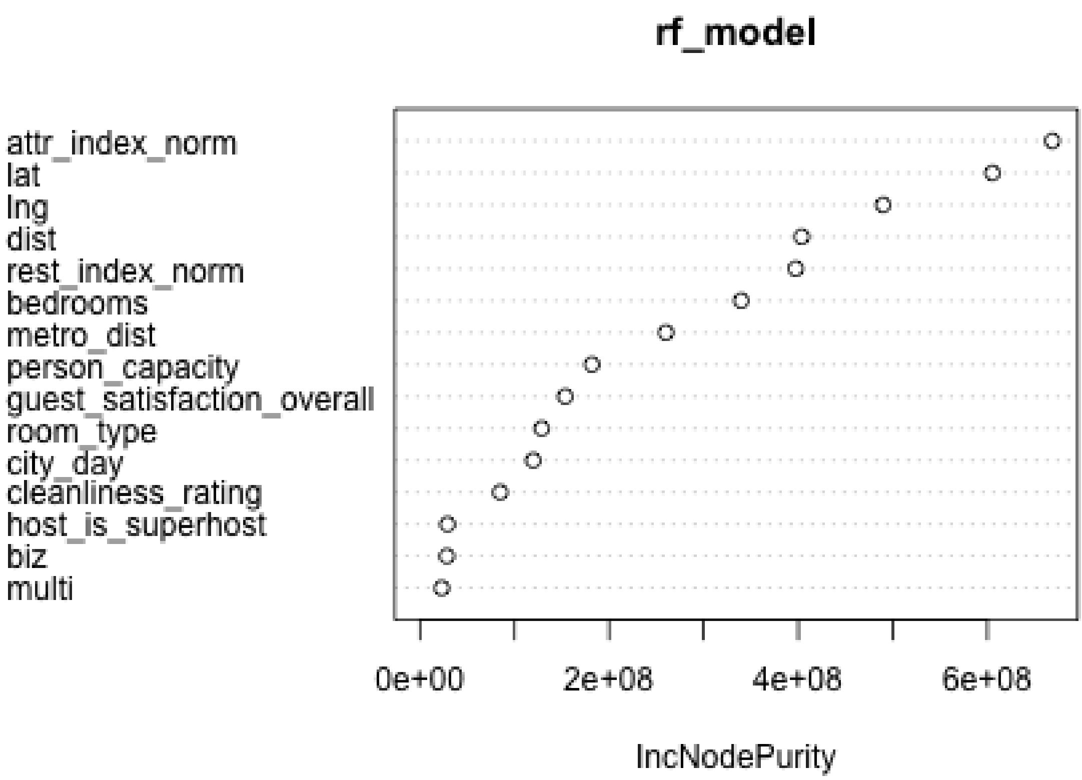
Model	Train R^2	Train Adj R^2	Train $RMSE$	Test R^2	Test Adj R^2	Test $RMSE$
MLR	0.2150414	0.2146725	304.9175	0.33744	0.3371287	233.2618
MLR with Step	0.2149964	0.2146275	304.9262	0.3373441	0.3370327	233.2787
MLR with IVs	0.2159815	0.215613	304.7348	0.3379444	0.3376333	233.173
MLR with IVs, Step	0.2159435	0.215575	304.7422	0.3380392	0.3377281	233.1563
Poly with Order 2	0.2154699	0.2151012	304.8342	0.3373538	0.3370424	233.277
Poly with Order 2, Step	0.2154289	0.2150602	304.8422	0.3373204	0.337009	233.2829
Poly with Order 2, IVs	0.22214	0.2217744	303.5356	0.334993	0.3346805	233.6922
Poly with Order 2, IVs, Step	0.2221344	0.2217689	303.5367	0.3350694	0.3347569	233.6788
Poly with Order 3	0.2160624	0.215694	304.7191	0.3375855	0.3372742	233.2362
Poly with Order 3, Step	0.2160337	0.2156653	304.7247	0.3376519	0.3373407	233.2245
Poly with Order 3, IVs	0.2330663	0.2327059	301.3962	0.1901115	0.189731	257.8955
Poly with Order 3, IVs, Step	0.2285384	0.2281759	302.2846	0.3281442	0.3281442	234.8925
Lasso With Order 1	0.215693	0.2147166	325.7776	0.3372306	0.3353023	233.2987
Lasso With Order 2	0.2201963	0.217819	326.6683	0.1830879	0.1772535	259.0113
Lasso With Order 3	0.2228371	0.2166688	326.2303	0.02854527	0.01036279	282.4505
Random Forest	0.8747755	0.8724317	121.7878	0.7588743	0.7543612	140.719

* IVs here is Interaction variables

Random Forest

We have trained Random Forest on all cities' combined data, and below are the important variables.

	IncNodePurity
room_type	122087850
host_is_superhost	26594131
multi	21575815
biz	24190212
city_day	114299029
person_capacity	183757090
cleanliness_rating	79985960
guest_satisfaction_overall	162106050
bedrooms	342017315
dist	423128244
metro_dist	248206656
attr_index_norm	662153431
rest_index_norm	435838043
lng	482308349
lat	588248566



The random forest model has chosen the attraction index as the most important variable, which makes more sense from a general point of view because the places with more attractions would attribute to pricing differences. The other variables ranking also seems consistent with common sense, such as latitude and longitude, which indicates location and distance from the city center, etc.

Overall the random forest has performed better than the other models with a Adjusted. R^2 value of 0.7543 and RMSE of 140.719 compared to the best regression model, which has Adjusted R^2 value of 0.3377 and RMSE of 233.1563 on the test set.

Discussion

When we applied separate models for each city data, on linear regression, latitude was least ranked, but when we combined all the data latitude is on top rank next to city day, it indicates the information about location is being captured by the latitude when all data is combined, in separate models all the data about city represents it location so latitude is least ranked.

The attraction index is ranked higher in all cities combined model than separated models. (From *Figure* 4)

The host_is_super host is ranked high in individual models than combined model, but its significance value is very low.

Room Type is ranked at the top in separate models, but in the combined model is at quite a low rank. It is due to the information of Room Type capturing the information about the location; it can be inferred from the graph *Figure* 5

Summary

This report investigates the factors that determine Airbnb room prices in Europe using data from the publicly available dataset 'Airbnb Price Determinants.' To identify the main determinants of Airbnb room prices in Europe, the report employed a regression analysis and random forest model. The dataset was collected from ten major European cities, including Amsterdam, Athens, Barcelona, Berlin, Budapest, Lisbon, London, Paris, Rome, and Vienna. Exploratory data analysis was conducted on various attributes of the data. The report discovered that Amsterdam has the highest Airbnb room prices in Europe. In general, entire homes have higher prices compared to shared rooms. The proximity of a room to a metro station was found to be correlated with higher prices. However, in Rome, the distance to the metro station has a negligible impact on room prices. The study also discovered that there are differences in room prices between shared and non-shared rooms. Furthermore, the study used multivariate linear regression, polynomial regression, interaction variables, and random forest regression to identify the factors that are strongly associated with Airbnb prices in Europe. The report demonstrates that the main determinants of Airbnb room prices in Europe include the number of people allowed in a room, the room type, and the host's superhost status. The report provides valuable insights for Airbnb guests, hosts, and investors in evaluating the value of investing in real estate in different European cities based on pricing trends.

Appendix for Code and Detailed Analysis

Pre Processing and Cleaning the Data

Data loading

```r
# Set the relative directory path
my_dir <- "./archive"

# List all the files in the directory
files <- list.files(path = my_dir, full.names = TRUE)
```

Combining the Data from all Files

```r
# Get a list of all the csv files in the directory
file_list <- list.files(path = my_dir, pattern = "*.csv", full.names = TRUE)

# Initialize an empty list to store the data frames
df_list <- list()

# Loop through each file and read it into a data frame
for (i in seq_along(file_list)) {
    df <- read.csv(file_list[i])

    # Add a new column with the city_day
    df$city_day <- basename(file_list[i])

    # Append the data frame to the list
    df_list[[i]] <- df
}

# Combine all the data frames into a single dataset
my_data <- bind_rows(df_list)

# Removing the .csv ext
my_data$city_day <- gsub("\\.csv", "", my_data$city_day)

# Print the first few rows of the data
head(my_data)
```

```r
print(unique(my_data[my_data$room_shared == my_data$room_private,
    ]$room_type))  # if the room is shared
```

```
## [1] "Entire home/apt"
```

```r
print(unique(my_data[my_data$room_private == "False", ]$room_type))
```

```
## [1] "Entire home/apt" "Shared room"
```

```r
print(unique(my_data[my_data$room_shared == "True", ]$room_type))
```

```
## [1] "Shared room"
```

```r
print(unique(my_data[my_data$room_shared == "False", ]$room_type))
```

```
## [1] "Private room"    "Entire home/apt"
```

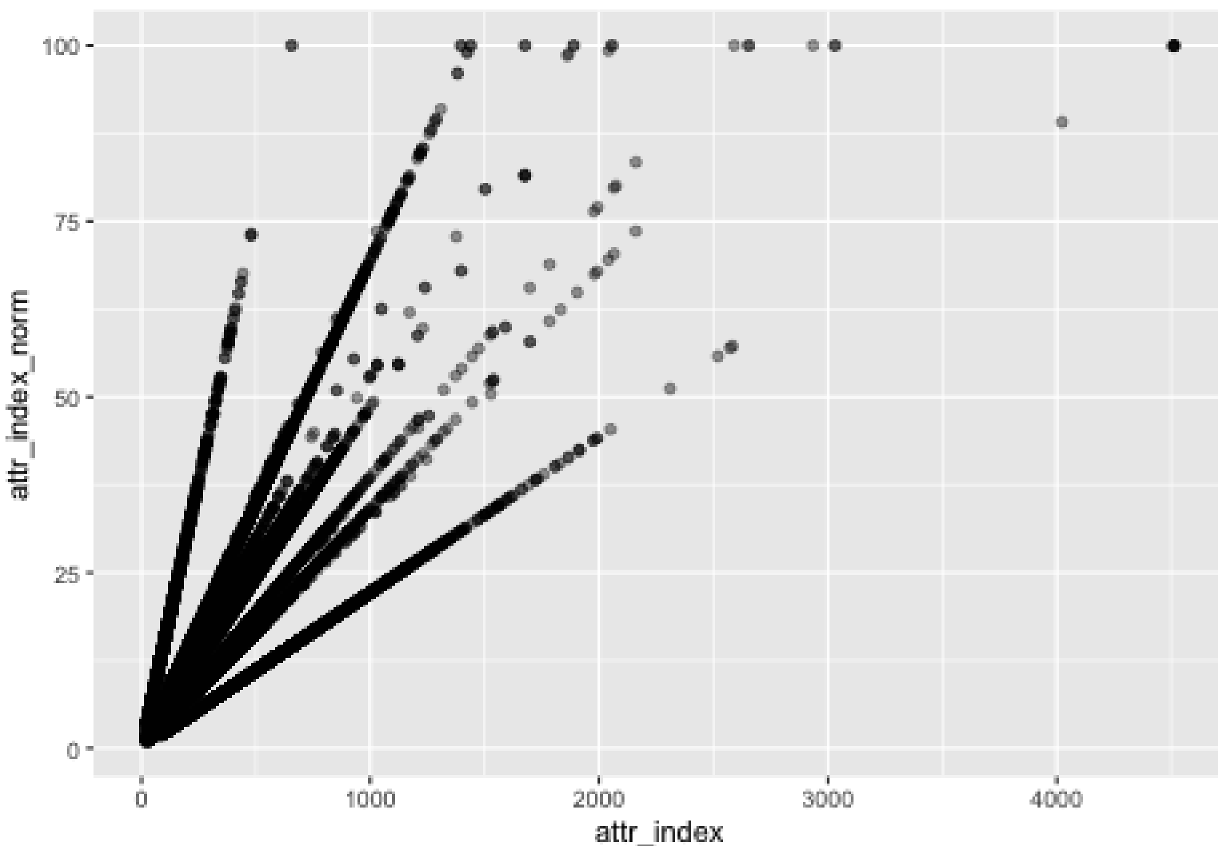
The room_shared and room_private information is already embedded in room_type. The variables are multi-collinear, so we can remove room_shared and room_private.

Dropping columns of room_shared and room_private

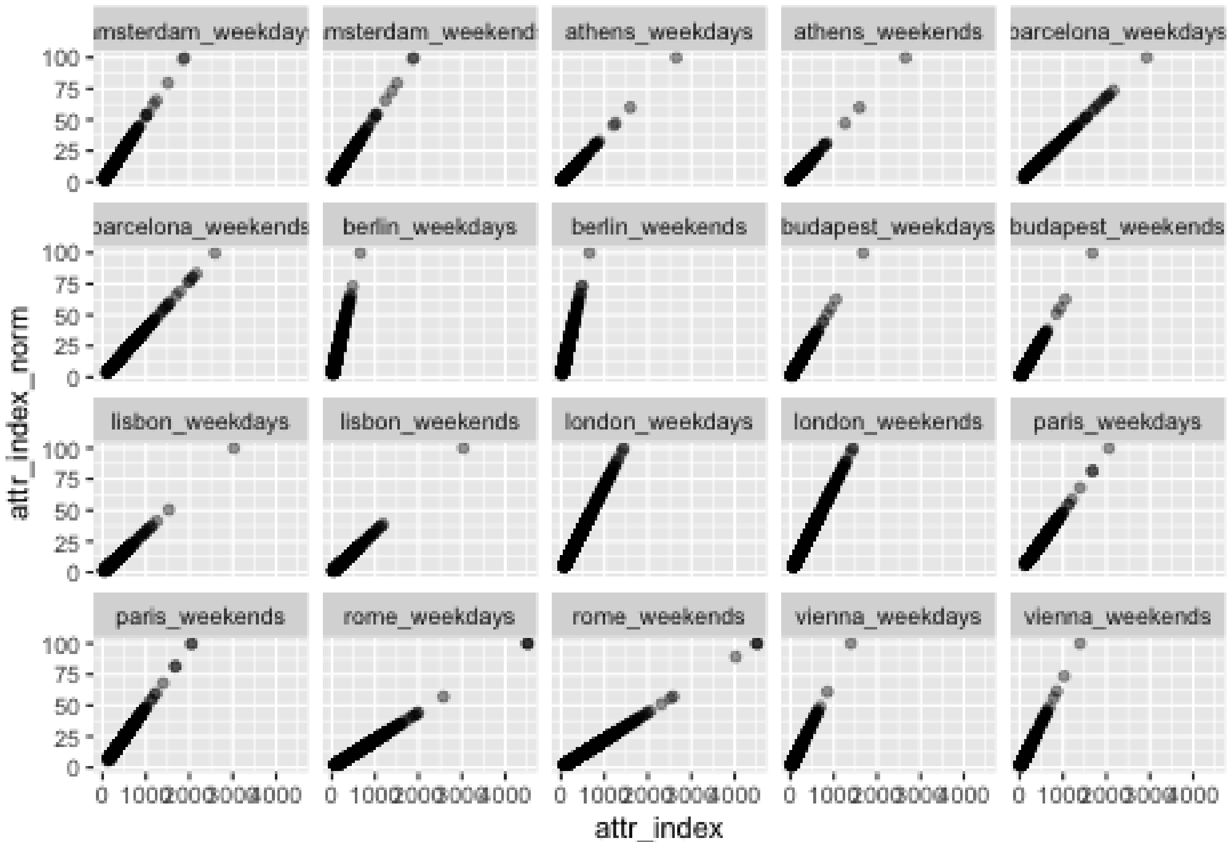
```r
my_data = select(my_data, -c(room_shared, room_private))
head(my_data)
```

```
##   X  realSum    room_type person_capacity host_is_superhost multi biz
## 1 0 194.0337 Private room               2             False     1   0
## 2 1 344.2458 Private room               4             False     0   0
## 3 2 264.1014 Private room               2             False     0   1
## 4 3 433.5294 Private room               4             False     0   1
## 5 4 485.5529 Private room               2              True     0   0
## 6 5 552.8086 Private room               3             False     0   0
##   cleanliness_rating guest_satisfaction_overall bedrooms      dist metro_dist
## 1                 10                         93        1 5.0229638  2.5393800
## 2                  8                         85        1 0.4883893  0.2394039
## 3                  9                         87        1 5.7483119  3.6516213
## 4                  9                         90        2 0.3848620  0.4398761
## 5                 10                         98        1 0.5447382  0.3186926
## 6                  8                        100        2 2.1314201  1.9046682
##   attr_index attr_index_norm rest_index rest_index_norm     lng      lat
## 1   78.69038        4.166708   98.25390        6.846473 4.90569 52.41772
## 2  631.17638       33.421209  837.28076       58.342928 4.90005 52.37432
## 3   75.27588        3.985908   95.38695        6.646700 4.97512 52.36103
## 4  493.27253       26.119108  875.03310       60.973565 4.89417 52.37663
## 5  552.83032       29.272733  815.30574       56.811677 4.90051 52.37508
## 6  174.78896        9.255191  225.20166       15.692376 4.87699 52.38966
##             city_day
## 1 amsterdam_weekdays
## 2 amsterdam_weekdays
## 3 amsterdam_weekdays
## 4 amsterdam_weekdays
## 5 amsterdam_weekdays
## 6 amsterdam_weekdays
```

```r
ggplot() + geom_point(data = my_data, aes(x = attr_index, y = attr_index_norm),
    alpha = 0.4)
```

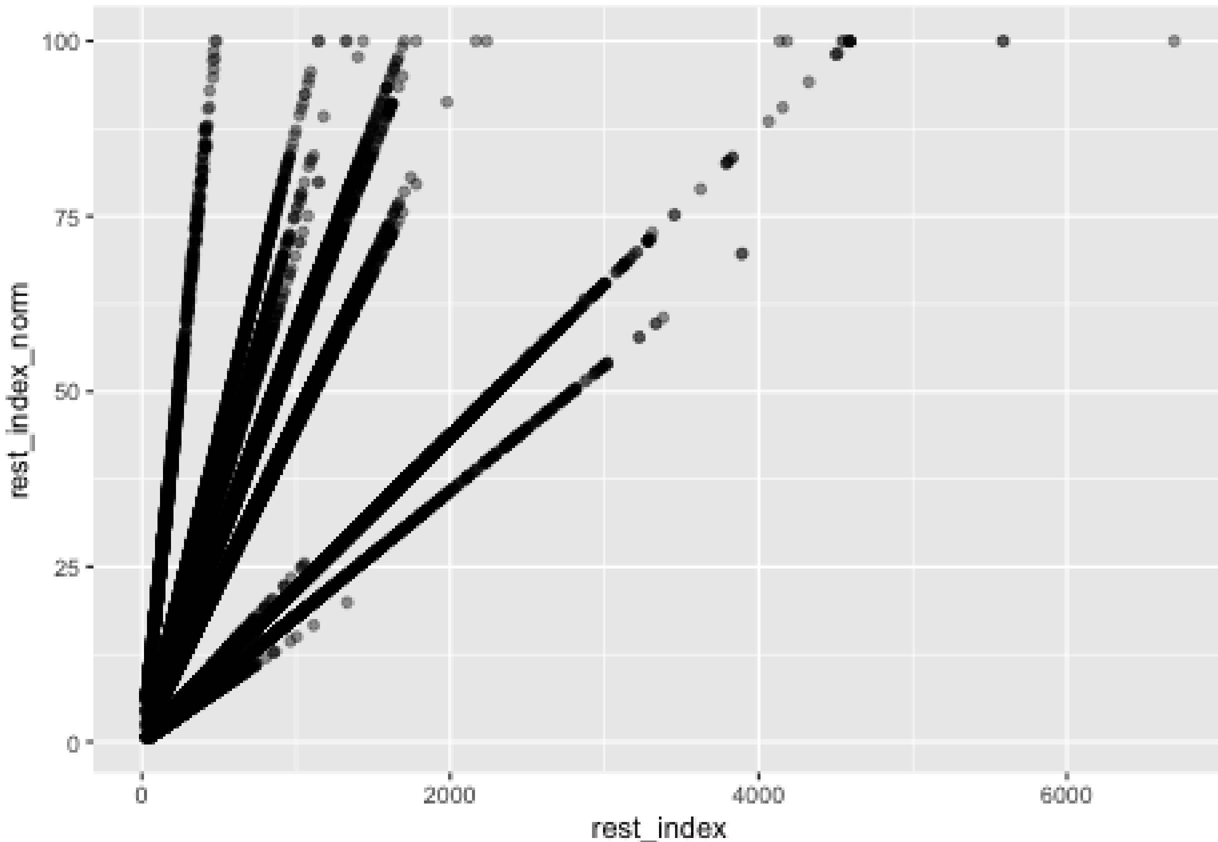



```
ggplot() + geom_point(data = my_data, aes(x = attr_index, y = attr_index_norm),
    alpha = 0.4) + facet_wrap(~city_day)
```

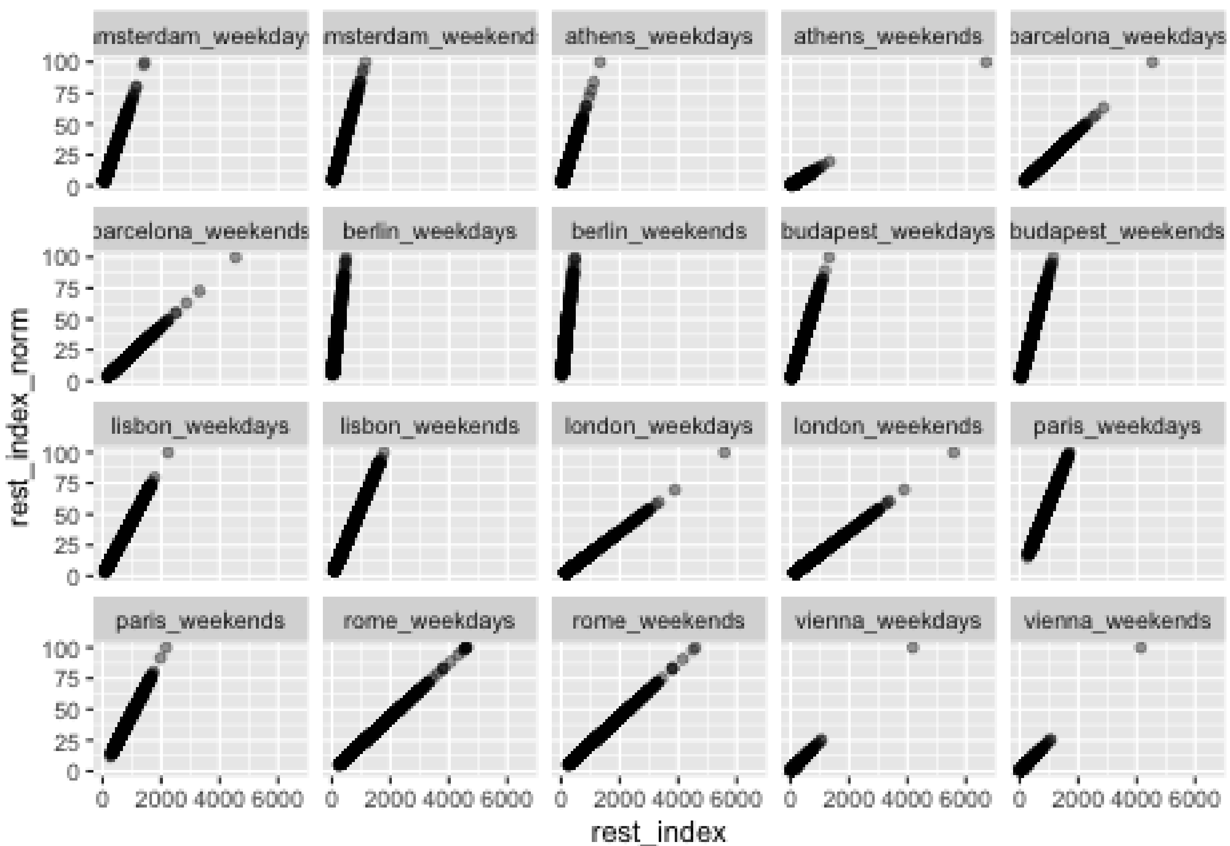


attr_index and attr_index_norm are same, attr_index_norm is just normalized attr_index

```
ggplot() + geom_point(data = my_data, aes(x = rest_index, y = rest_index_norm),
    alpha = 0.4)
```




```
ggplot() + geom_point(data = my_data, aes(x = rest_index, y = rest_index_norm),
    alpha = 0.4) + facet_wrap(~city_day)
```



rest__index and rest__index__norm are same, rest__index__norm is just normalized rest__index.

removing attr__index and rest__index

```
my_data = select(my_data, -c(attr_index, rest_index))
head(my_data)
```

Outliers using IQR Range

Filtering out the Outliers from Data Out of IQR Ranges

```
# Initialize an empty list to store the outliers
outliers_list <- list()

# Initialize an empty list to store the filtered data
# frames
df_list_filtered <- list()

# Loop through each file and read it into a data frame
# after removing outliers
for (i in seq_along(file_list)) {
    df_filtered <- read.csv(file_list[i])

    # Add a new column with the city_day
    df_filtered$city_day <- gsub("\\.csv", "", basename(file_list[i]))
```

```
    iqr_var1 <- IQR(df_filtered$realSum)

    # Calculate the upper and lower bounds for each
    # variable
    upper_var1 <- quantile(df_filtered$realSum, 0.75) + 1.5 *
        iqr_var1
    lower_var1 <- quantile(df_filtered$realSum, 0.25) - 1.5 *
        iqr_var1

    # Filter the data based on the upper and lower bounds
    # for each variable
    filtered_data <- filter(df_filtered, realSum > lower_var1 &
        realSum < upper_var1)

    # Append the filtered data frame to the list
    df_list_filtered[[i]] <- filtered_data

    # Get the rows that were removed while filtering
    outliers <- anti_join(df_filtered, filtered_data)

    # Append the outliers to the list
    outliers_list[[i]] <- outliers
}


# Combine all the filtered data frames into a single
# dataset
my_data_filtered <- bind_rows(df_list_filtered)

# Removing the .csv ext
my_data_filtered$city_day <- gsub("\\.csv", "", my_data_filtered$city_day)

summary(my_data_filtered)
```

```
##        X           realSum          room_type          room_shared
##  Min.   :   0   Min.   :  34.78   Length:48970       Length:48970
##  1st Qu.: 645   1st Qu.: 145.23   Class :character   Class :character
##  Median :1340   Median : 204.27   Mode  :character   Mode  :character
##  Mean   :1621   Mean   : 244.35
##  3rd Qu.:2385   3rd Qu.: 295.27
##  Max.   :5378   Max.   :1229.11
##  room_private       person_capacity host_is_superhost      multi
##  Length:48970       Min.   :2.00    Length:48970       Min.   :0.0000
##  Class :character   1st Qu.:2.00    Class :character   1st Qu.:0.0000
##  Mode  :character   Median :3.00    Mode  :character   Median :0.0000
##                     Mean   :3.08                       Mean   :0.2953
##                     3rd Qu.:4.00                       3rd Qu.:1.0000
##                     Max.   :6.00                       Max.   :1.0000
##       biz         cleanliness_rating guest_satisfaction_overall    bedrooms
##  Min.   :0.000   Min.   : 2.000     Min.   : 20.00             Min.   : 0.000
##  1st Qu.:0.000   1st Qu.: 9.000     1st Qu.: 90.00             1st Qu.: 1.000
##  Median :0.000   Median :10.000     Median : 95.00             Median : 1.000
##  Mean   :0.342   Mean   : 9.384     Mean   : 92.57             Mean   : 1.118
```

```
##  3rd Qu.:1.000   3rd Qu.:10.000     3rd Qu.: 98.00             3rd Qu.: 1.000
##  Max.   :1.000   Max.   :10.000     Max.   :100.00             Max.   :10.000
##       dist            metro_dist          attr_index      attr_index_norm
##  Min.   : 0.01506   Min.   : 0.002301   Min.   :  15.15   Min.   :  0.9263
##  1st Qu.: 1.48598   1st Qu.: 0.250718   1st Qu.: 133.75   1st Qu.:  6.2341
##  Median : 2.66962   Median : 0.416955   Median : 228.54   Median : 11.1929
##  Mean   : 3.24072   Mean   : 0.691774   Mean   : 285.15   Mean   : 13.0064
##  3rd Qu.: 4.31533   3rd Qu.: 0.749700   3rd Qu.: 374.37   3rd Qu.: 16.9444
##  Max.   :25.28456   Max.   :14.273577   Max.   :4513.56   Max.   :100.0000
##    rest_index      rest_index_norm         lng                lat
##  Min.   :  19.58   Min.   :  0.5928   Min.   :-9.22634   Min.   :37.95
##  1st Qu.: 245.42   1st Qu.:  8.5601   1st Qu.:-0.07277   1st Qu.:41.40
##  Median : 512.42   Median : 17.1799   Median : 4.87234   Median :47.51
##  Mean   : 611.32   Mean   : 22.2861   Mean   : 7.40027   Mean   :45.66
##  3rd Qu.: 818.44   3rd Qu.: 32.0321   3rd Qu.:13.52350   3rd Qu.:51.47
##  Max.   :6696.16   Max.   :100.0000   Max.   :23.78602   Max.   :52.64
##    city_day
##  Length:48970
##  Class :character
##  Mode  :character
##
##
##
```

```
# Combine all the outliers into a single dataset
my_outliers <- bind_rows(outliers_list)

# Removing the .csv ext
my_outliers$city_day <- gsub("\\.csv", "", my_outliers$city_day)

summary(my_outliers)
```

```
##        X           realSum         room_type          room_shared
##  Min.   :   0   Min.   :  279.4   Length:2737        Length:2737
##  1st Qu.: 666   1st Qu.:  469.2   Class :character   Class :character
##  Median :1237   Median :  691.9   Mode  :character   Mode  :character
##  Mean   :1614   Mean   :  915.5
##  3rd Qu.:2310   3rd Qu.:  996.3
##  Max.   :5374   Max.   :18545.5
##  room_private       person_capacity host_is_superhost      multi
##  Length:2737        Min.   :2.000   Length:2737        Min.   :0.000
##  Class :character   1st Qu.:4.000   Class :character   1st Qu.:0.000
##  Mode  :character   Median :5.000   Mode  :character   Median :0.000
##                     Mean   :4.628                      Mean   :0.221
##                     3rd Qu.:6.000                      3rd Qu.:0.000
##                     Max.   :6.000                      Max.   :1.000
##       biz          cleanliness_rating guest_satisfaction_overall    bedrooms
##  Min.   :0.0000   Min.   : 2.000     Min.   : 20.00             Min.   :0.000
##  1st Qu.:0.0000   1st Qu.: 9.000     1st Qu.: 91.00             1st Qu.:1.000
##  Median :0.0000   Median :10.000     Median : 97.00             Median :2.000
##  Mean   :0.4965   Mean   : 9.509     Mean   : 93.65             Mean   :1.886
##  3rd Qu.:1.0000   3rd Qu.:10.000     3rd Qu.:100.00             3rd Qu.:2.000
##  Max.   :1.0000   Max.   :10.000     Max.   :100.00             Max.   :6.000
##       dist            metro_dist         attr_index     attr_index_norm
```

```
##  Min.   : 0.01504   Min.   :0.006171   Min.   :  20.5   Min.   :  1.468
##  1st Qu.: 1.04119   1st Qu.:0.218081   1st Qu.: 225.1   1st Qu.: 11.719
##  Median : 1.89579   Median :0.352339   Median : 385.0   Median : 17.958
##  Mean   : 2.30674   Mean   :0.498426   Mean   : 456.2   Mean   : 20.892
##  3rd Qu.: 3.00820   3rd Qu.:0.576430   3rd Qu.: 610.6   3rd Qu.: 25.953
##  Max.   :21.29515   Max.   :8.918036   Max.   :2040.4   Max.   :100.000
##    rest_index     rest_index_norm        lng                lat
##  Min.   :  27.9   Min.   :  0.667   Min.   :-9.22476   Min.   :37.96
##  1st Qu.: 408.5   1st Qu.: 14.187   1st Qu.:-0.06677   1st Qu.:41.41
##  Median : 739.9   Median : 30.001   Median : 4.88384   Median :47.51
##  Mean   : 904.9   Mean   : 31.734   Mean   : 7.88764   Mean   :45.93
##  3rd Qu.:1269.7   3rd Qu.: 45.426   3rd Qu.:13.44666   3rd Qu.:51.50
##  Max.   :4183.1   Max.   :100.000   Max.   :23.75400   Max.   :52.58
##    city_day
##  Length:2737
##  Class :character
##  Mode  :character
##
##
##
```

Percentage of Outliers outside of IQR range.

```r
# Create empty table
outliers_table <- data.frame(City_day = character(), Data_Length = numeric(),
    Percent_Outliers = numeric(), stringsAsFactors = FALSE)

# Loop through city_data and fill in table
for (city_day in unique(my_data$city_day)) {
    x = my_data[my_data$city_day == city_day, ]$realSum
    q1 <- quantile(x, 0.25)
    q3 <- quantile(x, 0.75)
    iqr <- IQR(x)
    upper_bound <- q3 + 1.5 * iqr
    lower_bound <- q1 - 1.5 * iqr
    x_no_outliers <- x[x >= lower_bound & x <= upper_bound]
    percent_outliers <- ((length(x) - length(x_no_outliers))/length(x)) *
        100

    # Add row to table
    outliers_table <- rbind(outliers_table, data.frame(City_day = city_day,
        Data_Length = length(x), Percent_Outliers = percent_outliers))
}

# Format table using kable
kable(outliers_table, format = "markdown")
```

City_day	Data_Length	Percent_Outliers
amsterdam_weekdays	1103	5.077063
amsterdam_weekends	977	5.629478
athens_weekdays	2653	5.767056

City_day	Data_Length	Percent_Outliers
athens_weekends	2627	5.405405
barcelona_weekdays	1555	7.524116
barcelona_weekends	1278	8.059468
berlin_weekdays	1284	6.308411
berlin_weekends	1200	6.166667
budapest_weekdays	2074	5.930569
budapest_weekends	1948	5.544148
lisbon_weekdays	2857	3.360168
lisbon_weekends	2906	3.475568
london_weekdays	4614	5.353273
london_weekends	5379	5.521472
paris_weekdays	3130	6.134185
paris_weekends	3558	5.368184
rome_weekdays	4492	5.031167
rome_weekends	4535	5.005513
vienna_weekdays	1738	4.257767
vienna_weekends	1799	4.113396

Spilt Training and Testing Data

```
set.seed(123456789)
my_data_train <- my_data[sample(nrow(my_data), 0.7 * nrow(my_data)),
    ]
my_data_test <- my_data[setdiff(1:nrow(my_data), rownames(my_data_train)),
    ]
dim(my_data_train)
```

```
## [1] 36194    17
```

```
dim(my_data_test)
```

```
## [1] 15513    17
```

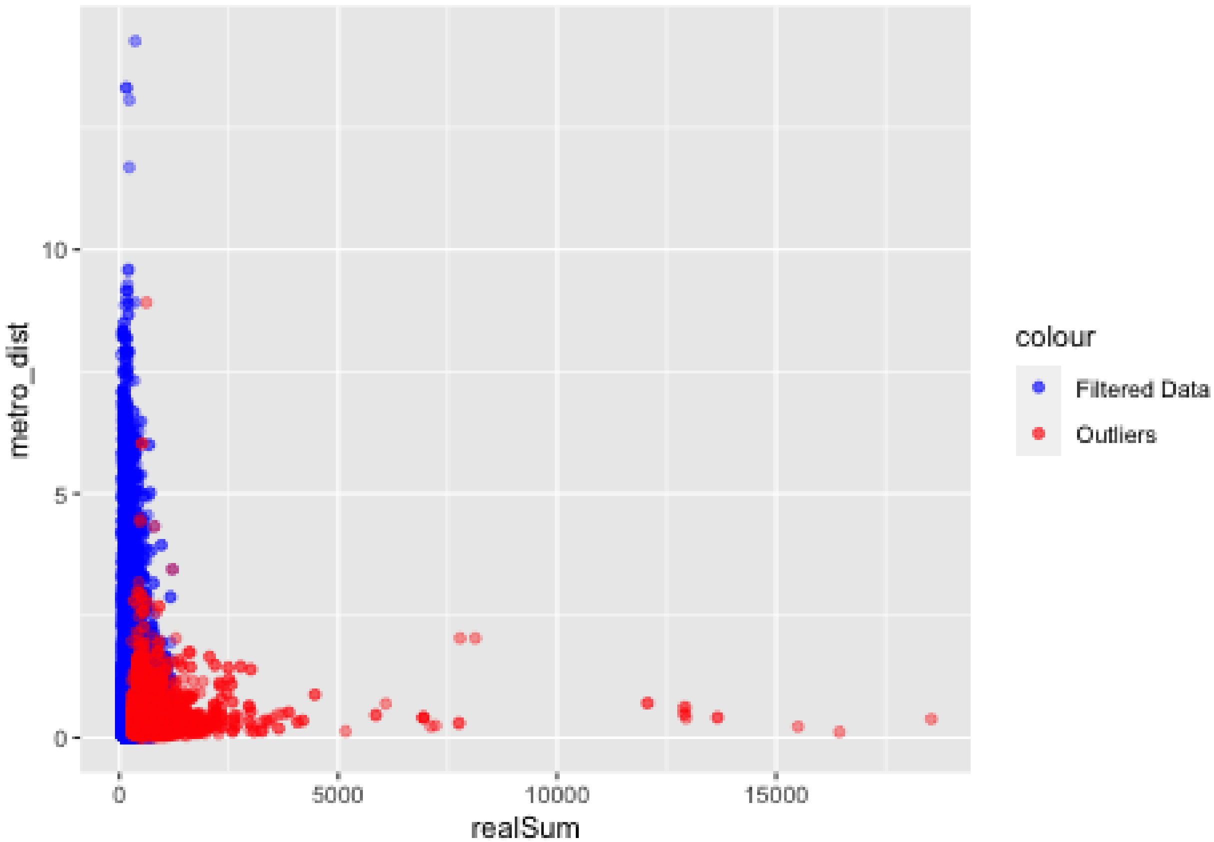
Exploratory Data Analysis

Outlier Analysis

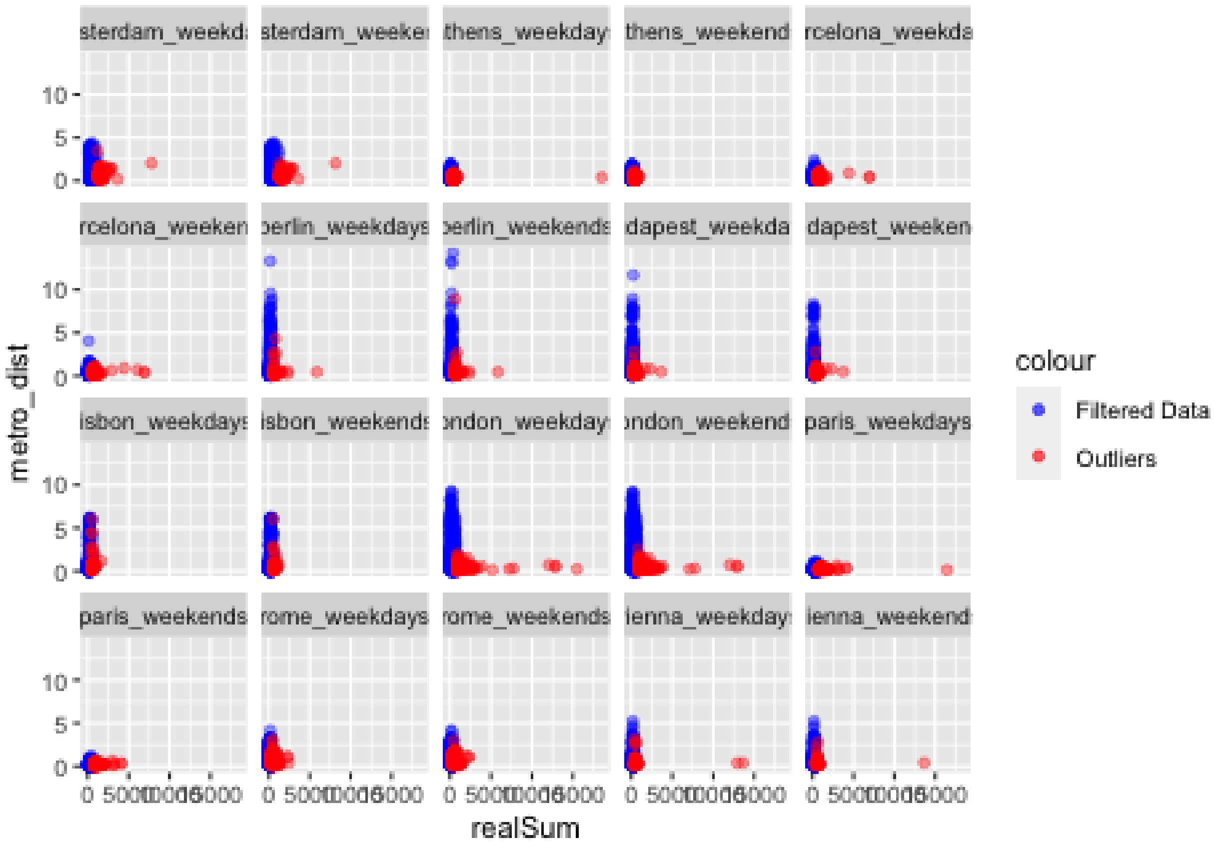
Metro Dist vs Real Sum

We have planned to analyse the filtered data along with outlier data. Here outlier data represents the hotel rooms with high prices.

```
ggplot() + geom_point(data = my_data_filtered, aes(x = realSum,
    y = metro_dist, color = "Filtered Data"), alpha = 0.4) +
    geom_point(data = my_outliers, aes(x = realSum, y = metro_dist,
        color = "Outliers"), alpha = 0.4) + scale_color_manual(values = c(`Filtered Data` = "blue",
    Outliers = "red"))
```



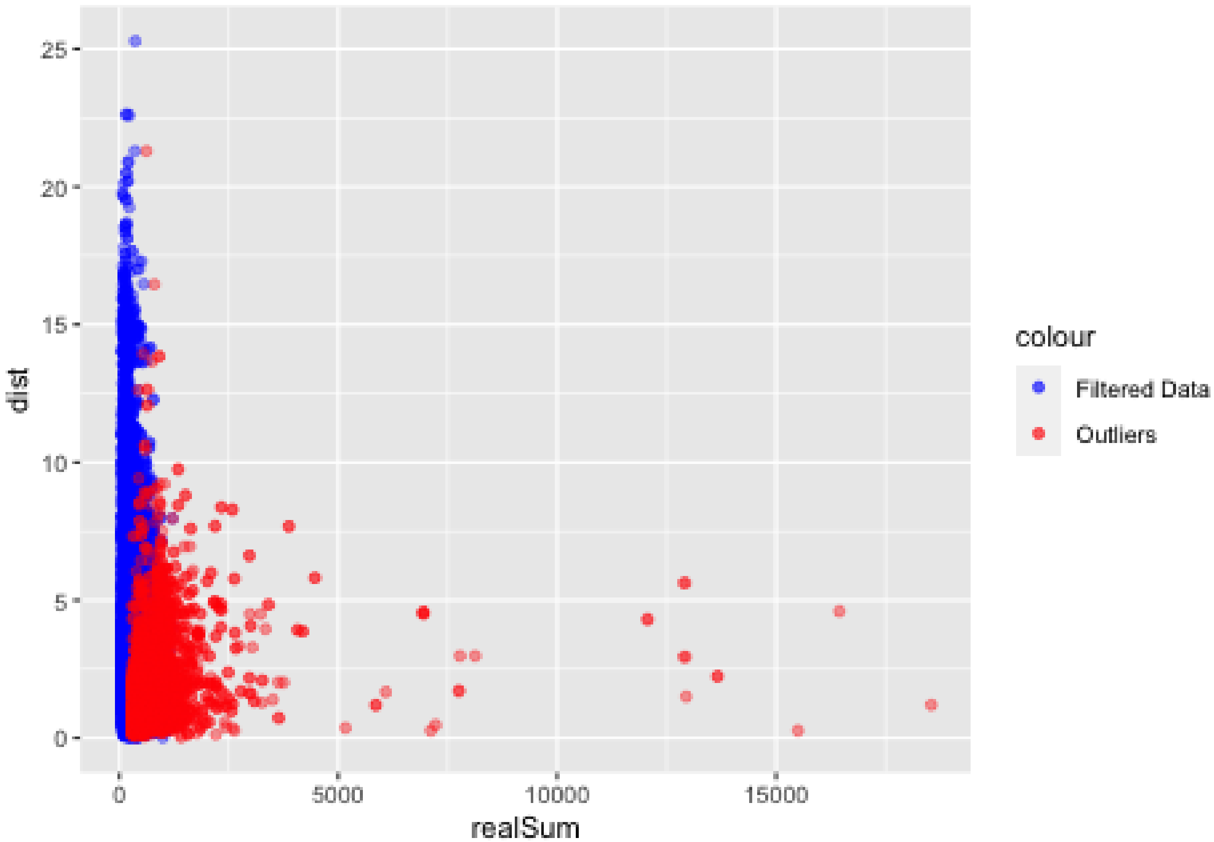

```
ggplot() + geom_point(data = my_data_filtered, aes(x = realSum,
    y = metro_dist, color = "Filtered Data"), alpha = 0.4) +
    geom_point(data = my_outliers, aes(x = realSum, y = metro_dist,
        color = "Outliers"), alpha = 0.4) + scale_color_manual(values = c(`Filtered Data` = "blue",
    Outliers = "red")) + facet_wrap(~city_day)
```



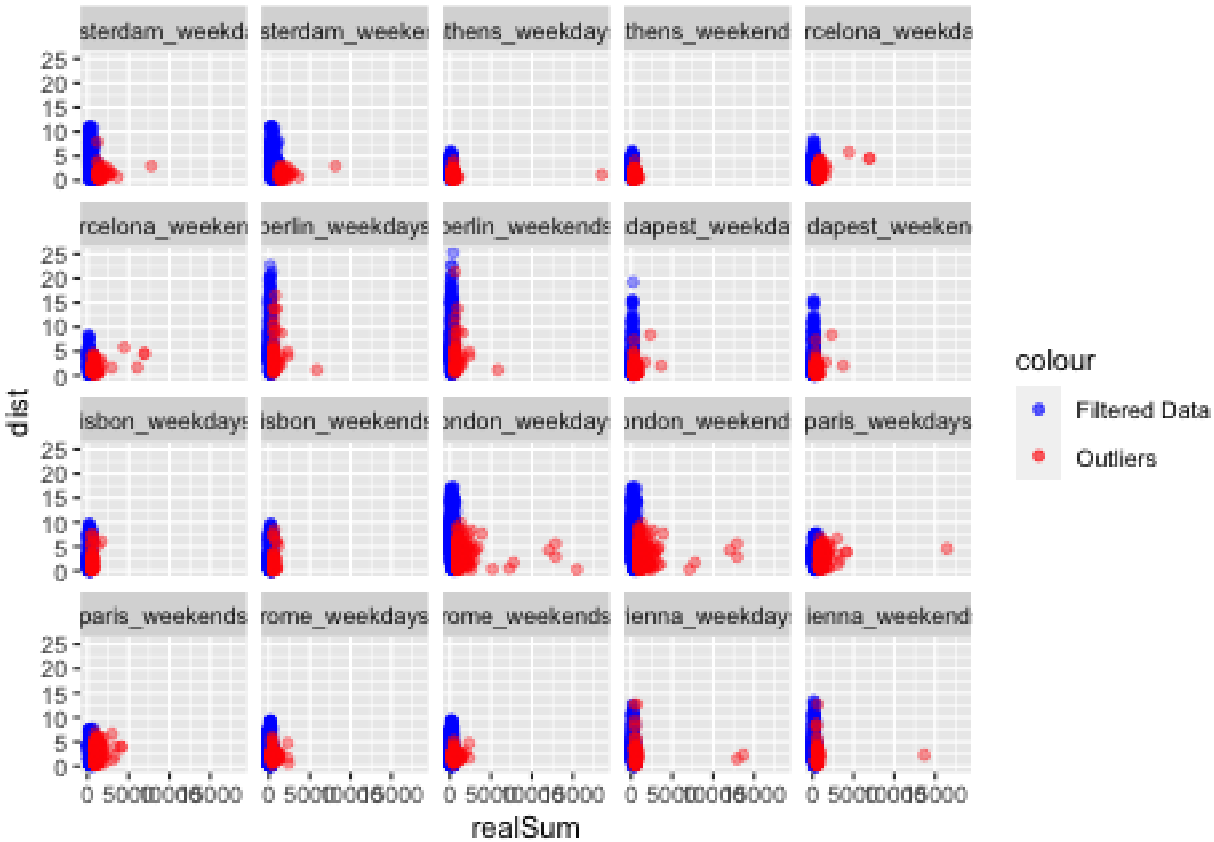
In general the rooms that are closer to metro have comparatively higher prices. But, in Rome city the distance to metro is almost same for both categories of price.

Real Sum vs Distance

```
ggplot() + geom_point(data = my_data_filtered, aes(x = realSum,
    y = dist, color = "Filtered Data"), alpha = 0.4) + geom_point(data = my_outliers,
    aes(x = realSum, y = dist, color = "Outliers"), alpha = 0.4) +
    scale_color_manual(values = c(`Filtered Data` = "blue", Outliers = "red"))
```



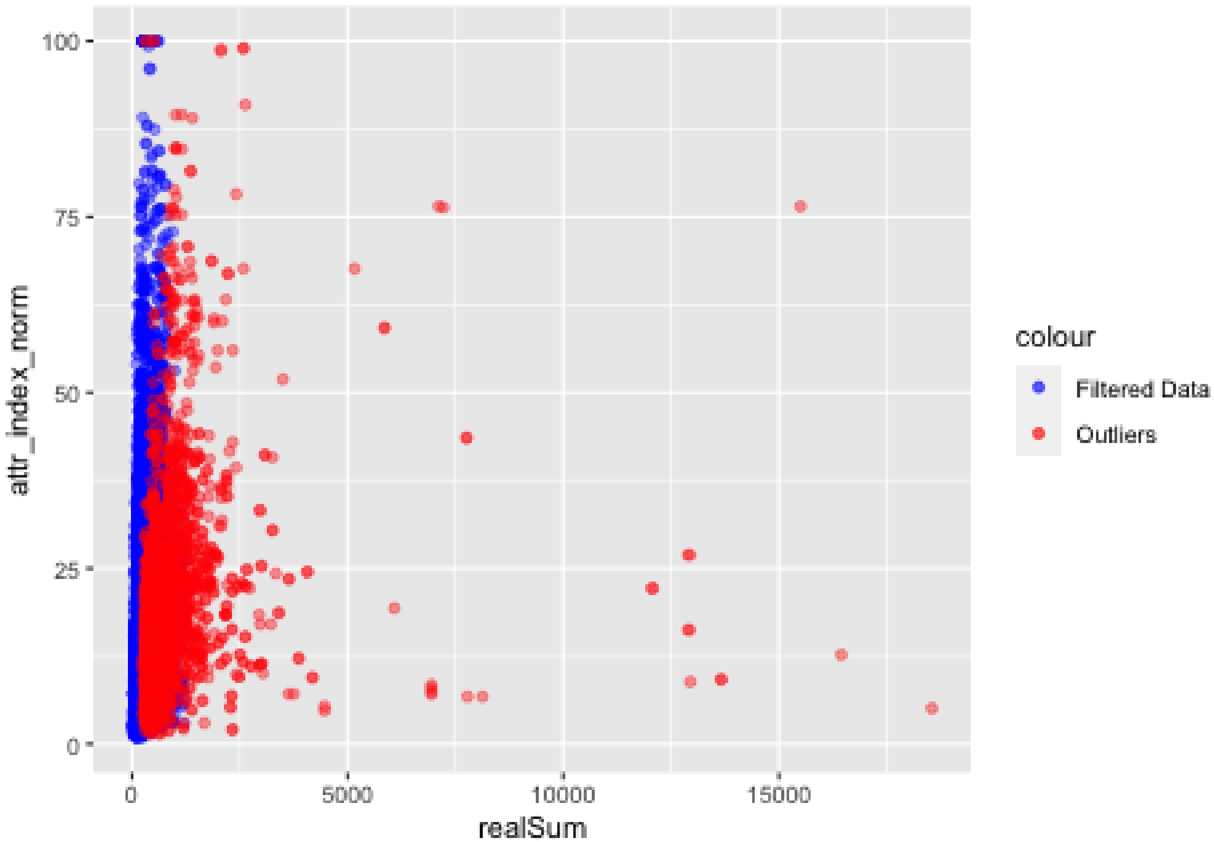

```
ggplot() + geom_point(data = my_data_filtered, aes(x = realSum,
    y = dist, color = "Filtered Data"), alpha = 0.4) + geom_point(data = my_outliers,
    aes(x = realSum, y = dist, color = "Outliers"), alpha = 0.4) +
    scale_color_manual(values = c(`Filtered Data` = "blue", Outliers = "red")) +
    facet_wrap(~city_day)
```



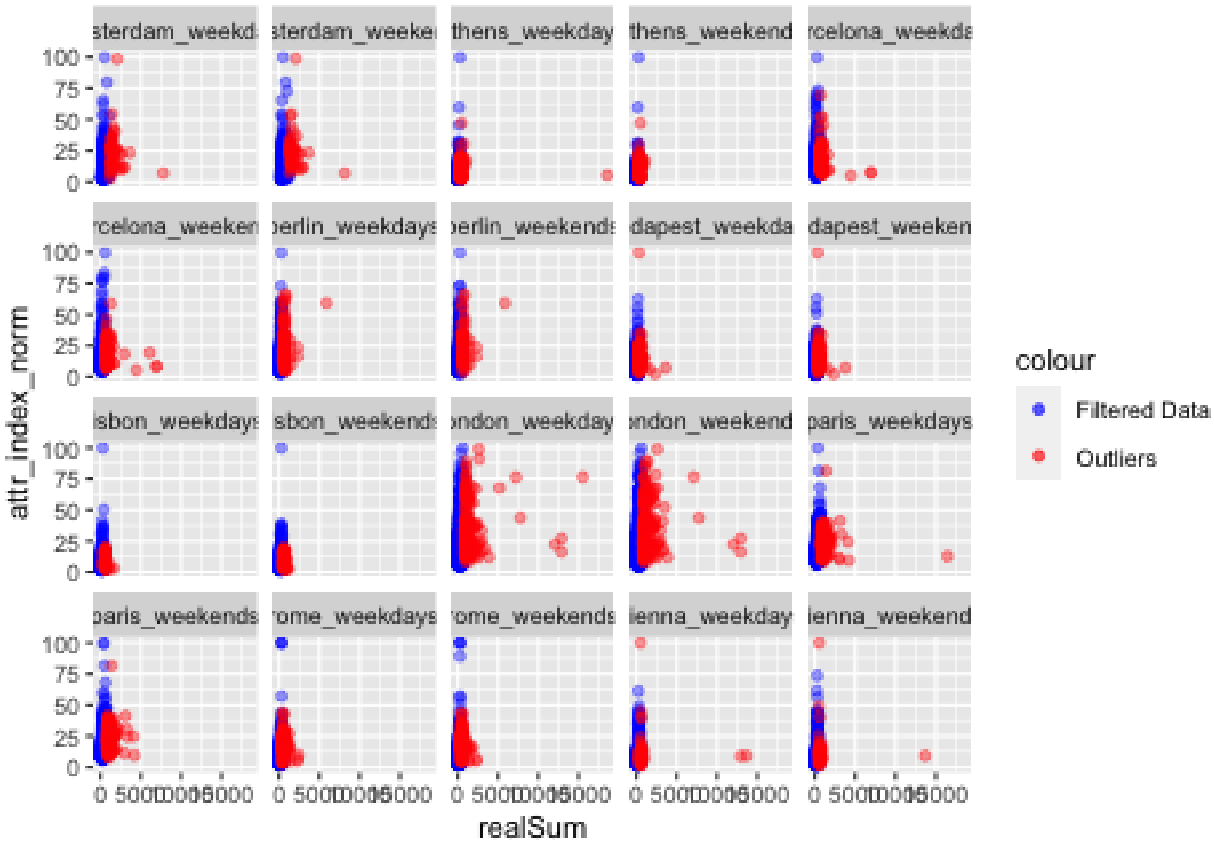
In general the pricey rooms are near to the centre of the city.

Real Sum vs Attraction Index Normal

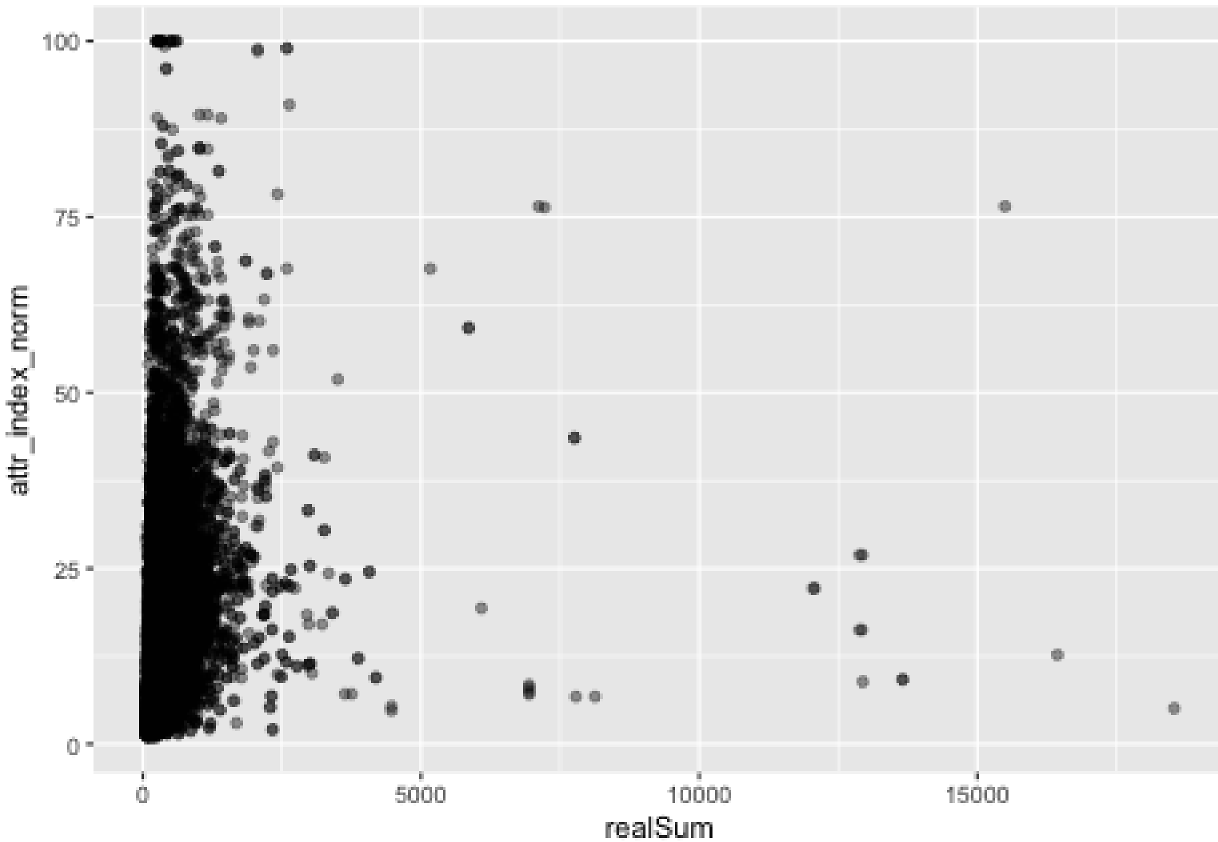
```
ggplot() + geom_point(data = my_data_filtered, aes(x = realSum,
    y = attr_index_norm, color = "Filtered Data"), alpha = 0.4) +
    geom_point(data = my_outliers, aes(x = realSum, y = attr_index_norm,
        color = "Outliers"), alpha = 0.4) + scale_color_manual(values = c(`Filtered Data` = "blue",
    Outliers = "red"))
```




```
ggplot() + geom_point(data = my_data_filtered, aes(x = realSum,
    y = attr_index_norm, color = "Filtered Data"), alpha = 0.4) +
    geom_point(data = my_outliers, aes(x = realSum, y = attr_index_norm,
        color = "Outliers"), alpha = 0.4) + scale_color_manual(values = c(`Filtered Data` = "blue",
    Outliers = "red")) + facet_wrap(~city_day)
```



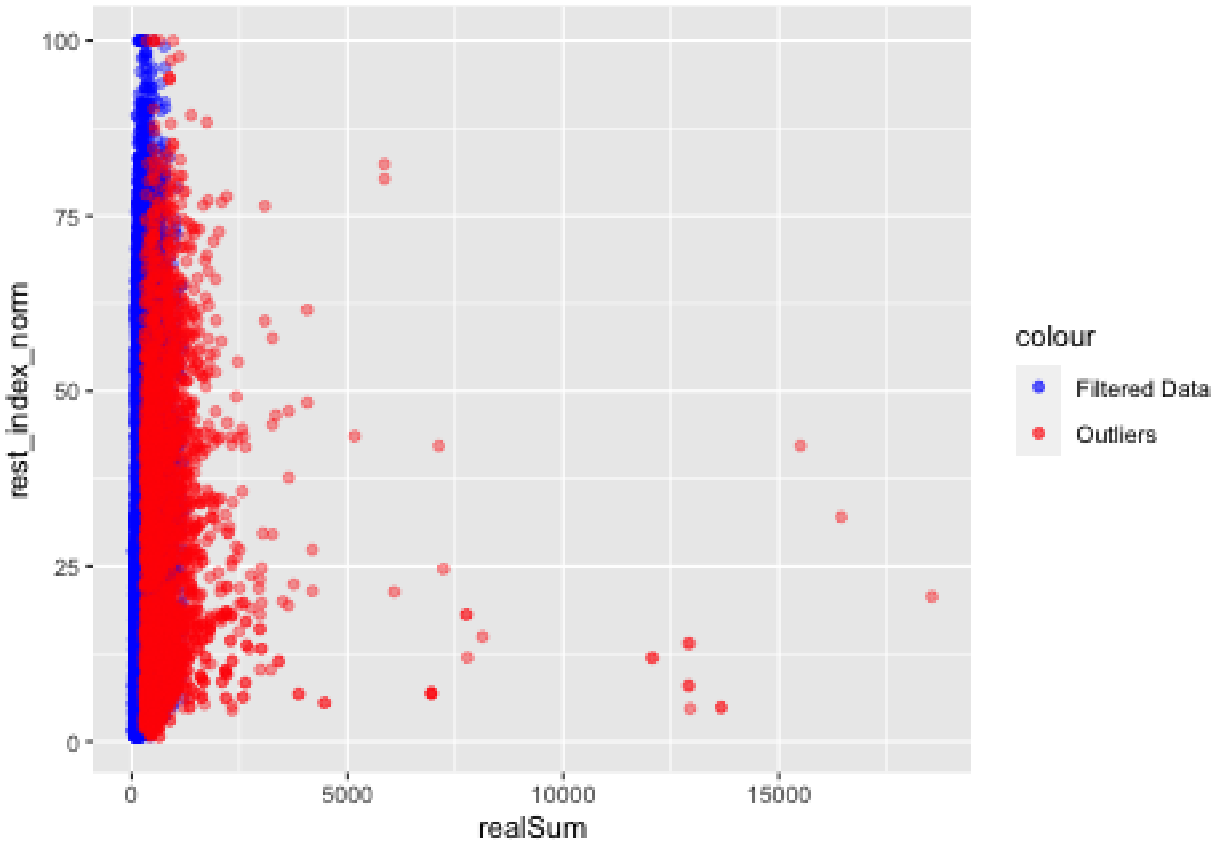

```r
ggplot() + geom_point(data = my_data_filtered, aes(x = realSum,
    y = attr_index_norm), alpha = 0.4) + geom_point(data = my_outliers,
    aes(x = realSum, y = attr_index_norm), alpha = 0.4)
```



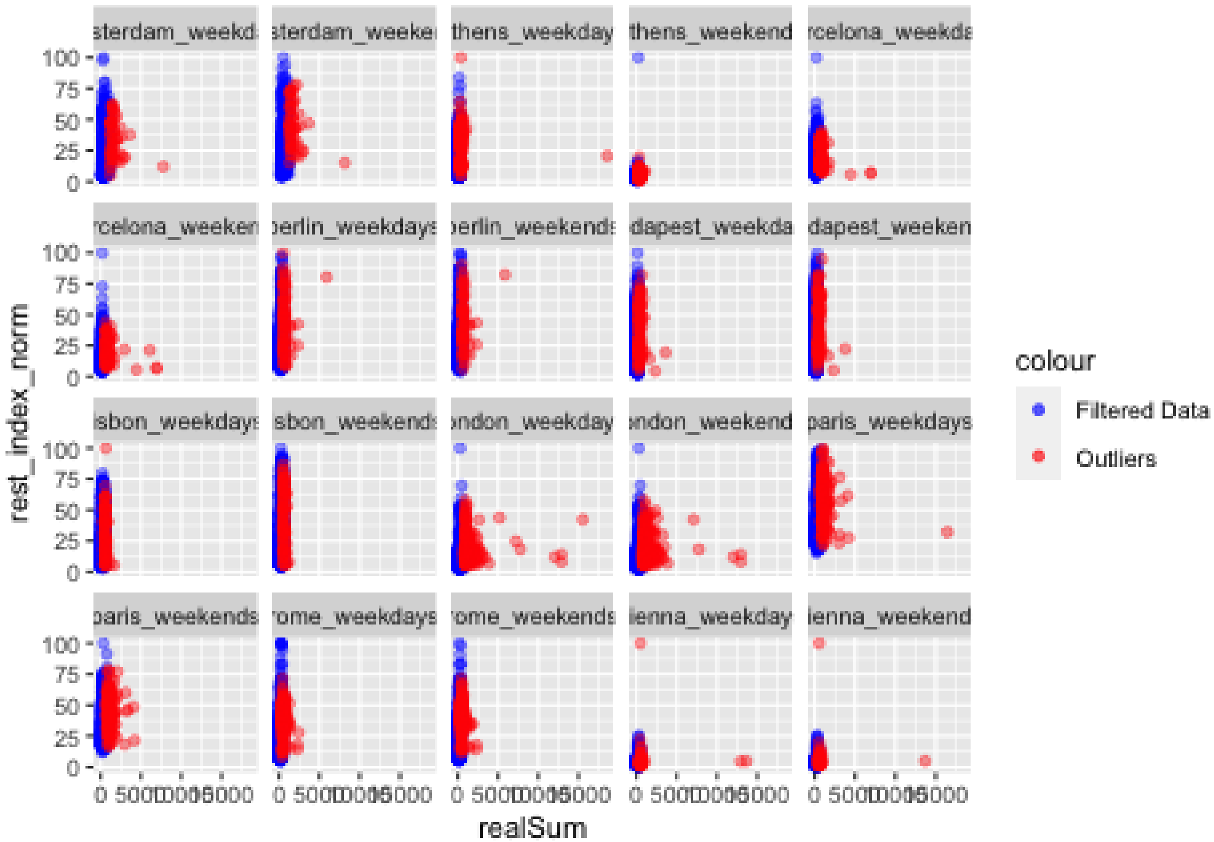
The range of values falling b/w outliers and normal data is almost same . So there isn't a relationship b/w attr_index and realSum.

Real Sum vs Restaurant Index Normal

```
ggplot() + geom_point(data = my_data_filtered, aes(x = realSum,
    y = rest_index_norm, color = "Filtered Data"), alpha = 0.4) +
    geom_point(data = my_outliers, aes(x = realSum, y = rest_index_norm,
        color = "Outliers"), alpha = 0.4) + scale_color_manual(values = c(`Filtered Data` = "blue",
    Outliers = "red"))
```



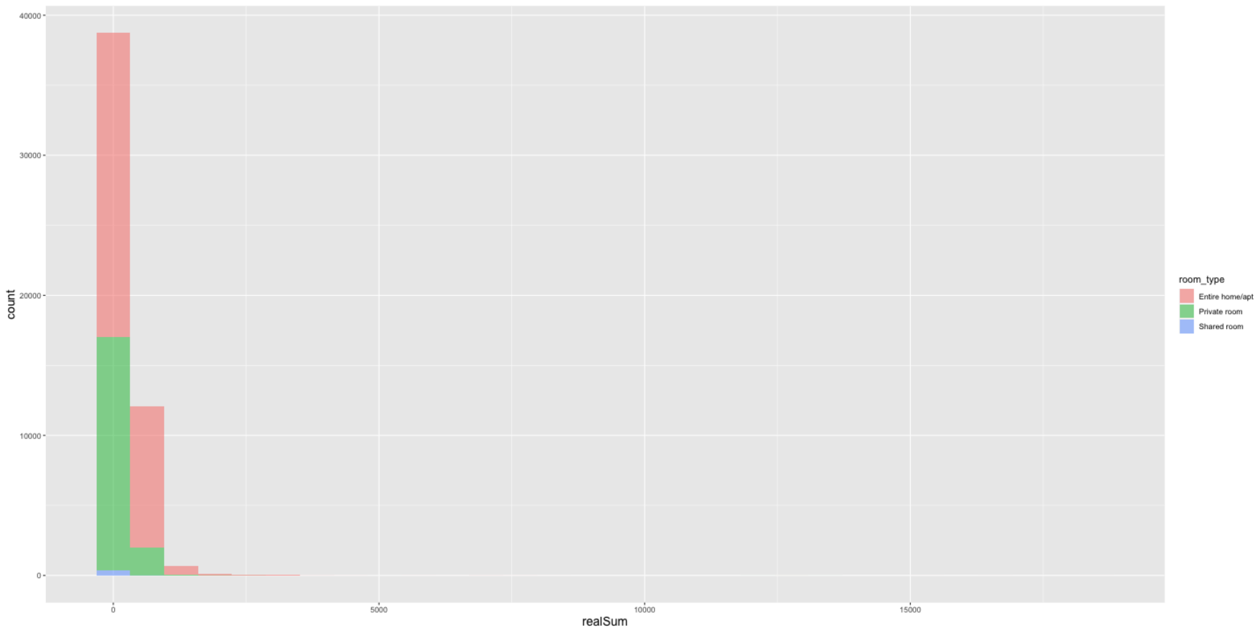

```
ggplot() + geom_point(data = my_data_filtered, aes(x = realSum,
    y = rest_index_norm, color = "Filtered Data"), alpha = 0.4) +
    geom_point(data = my_outliers, aes(x = realSum, y = rest_index_norm,
        color = "Outliers"), alpha = 0.4) + scale_color_manual(values = c(`Filtered Data` = "blue",
    Outliers = "red")) + facet_wrap(~city_day)
```



There is no relationship between outliers and rest_index

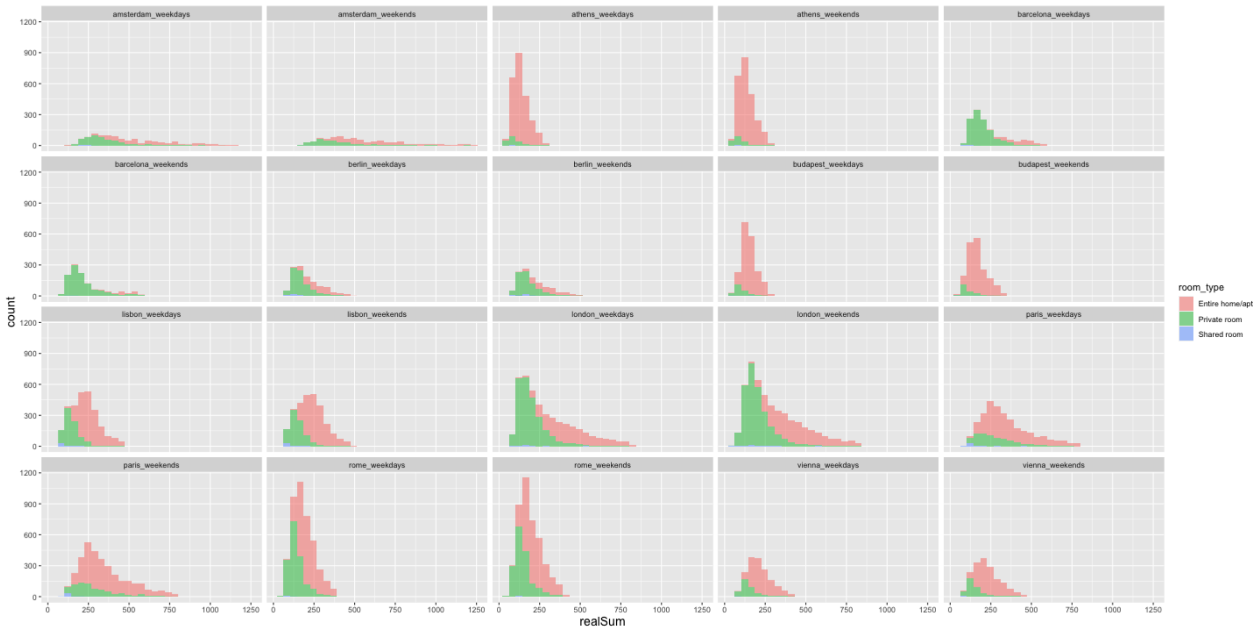
Room Type Vs Real Sum

```
ggplot(my_data, aes(x = realSum, fill = room_type, group = room_type)) +
    geom_histogram(alpha = 0.5, nbins = 20) + theme(axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14))
```




```
ggplot(my_data_filtered, aes(x = realSum, fill = room_type, group = room_type)) +
    geom_histogram(alpha = 0.5, nbins = 20) + theme(axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14))
```

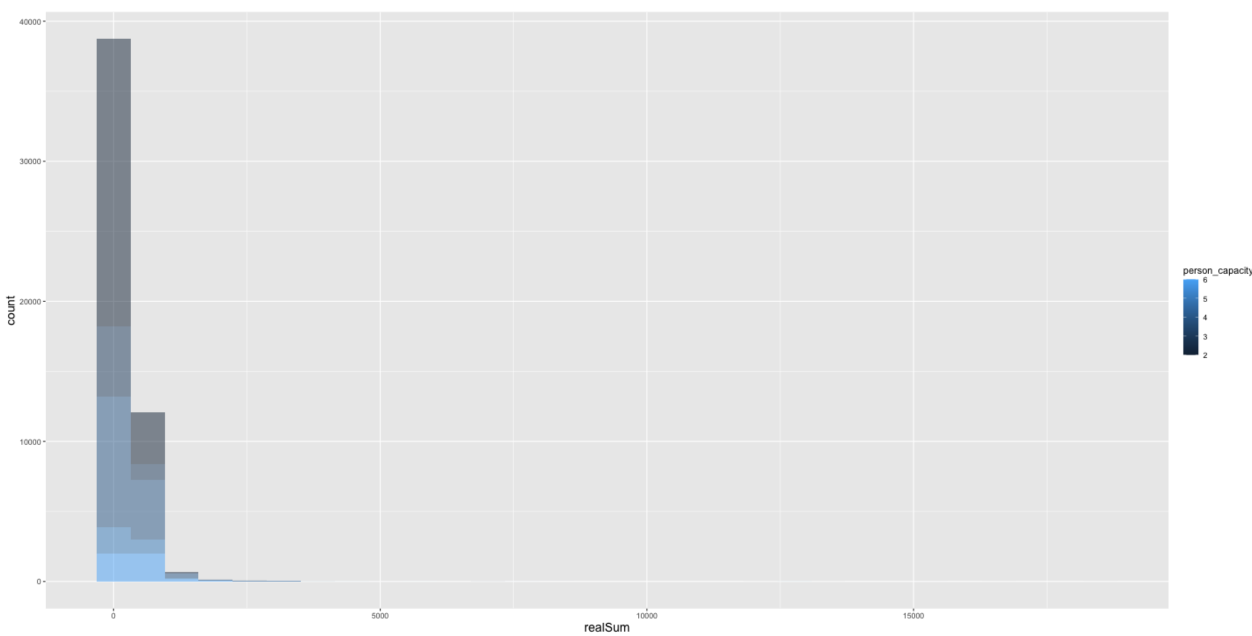

```
ggplot(my_data_filtered, aes(x = realSum, fill = room_type, group = room_type)) +
    geom_histogram(alpha = 0.5, nbins = 20) + theme(axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14)) + facet_wrap(~city_day)
```



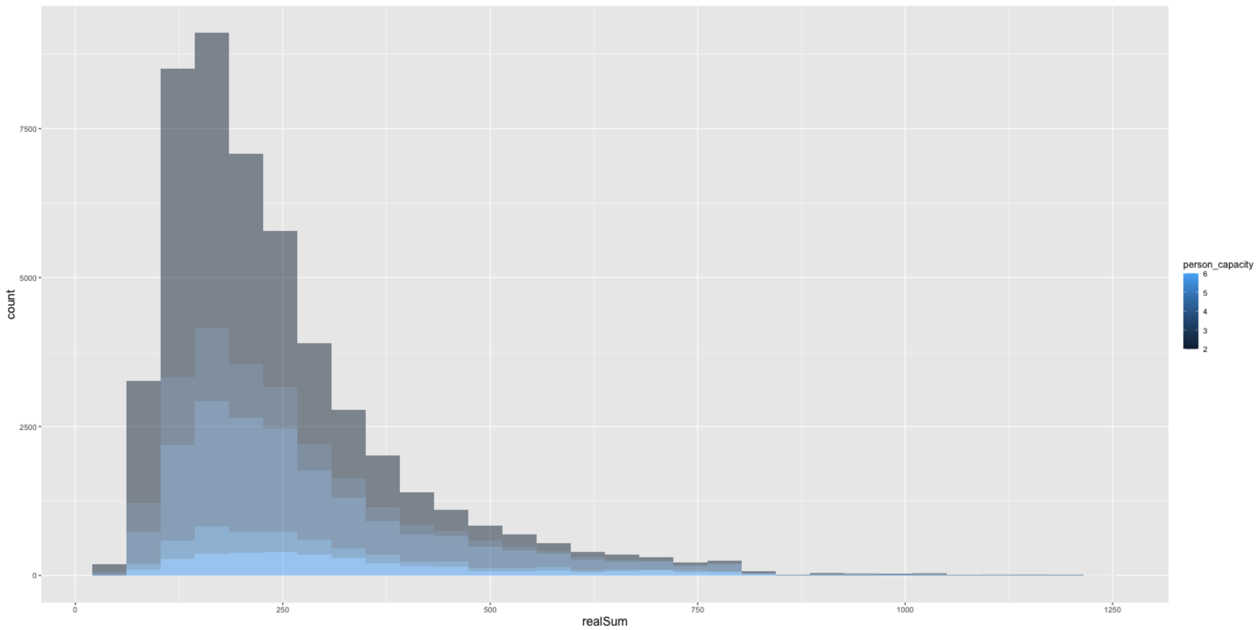
The price of entire home/apt tend to be higher compared to other two categories. And the count of entire home /apt is also more.

Room Type Vs Person Capacity

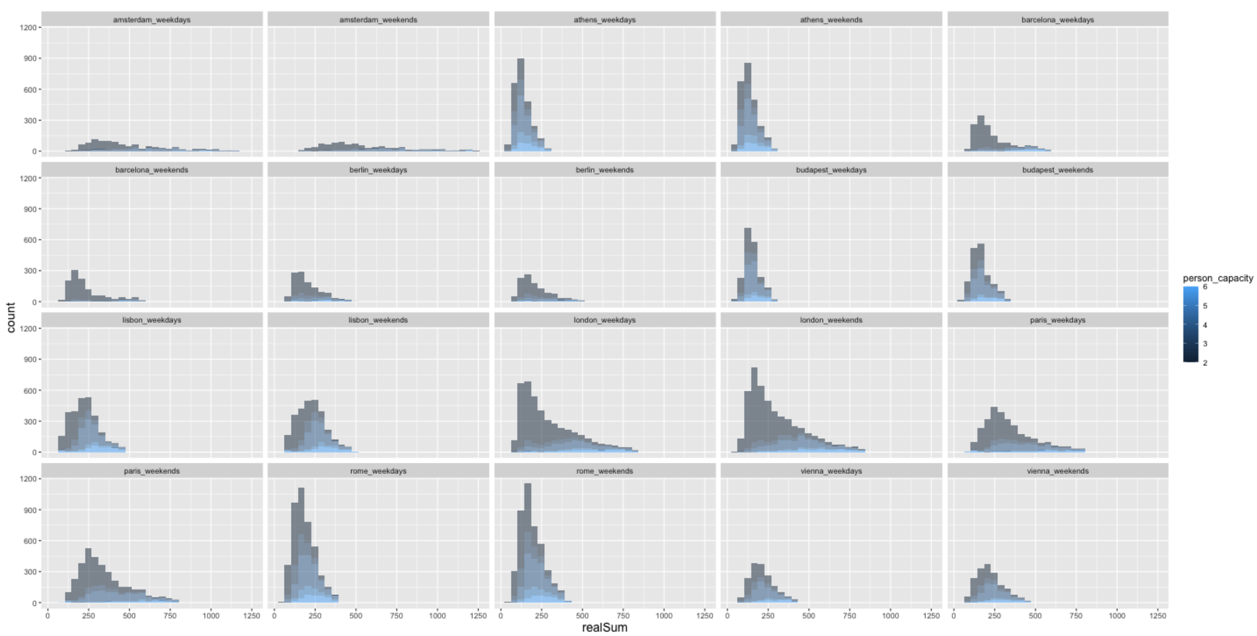
```
ggplot(my_data, aes(x = realSum, fill = person_capacity, group = person_capacity)) +
    geom_histogram(alpha = 0.5, nbins = 20) + theme(axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14))
```




```
ggplot(my_data_filtered, aes(x = realSum, fill = person_capacity,
    group = person_capacity)) + geom_histogram(alpha = 0.5, nbins = 20) +
    theme(axis.title.x = element_text(size = 14), axis.title.y = element_text(size = 14))
```



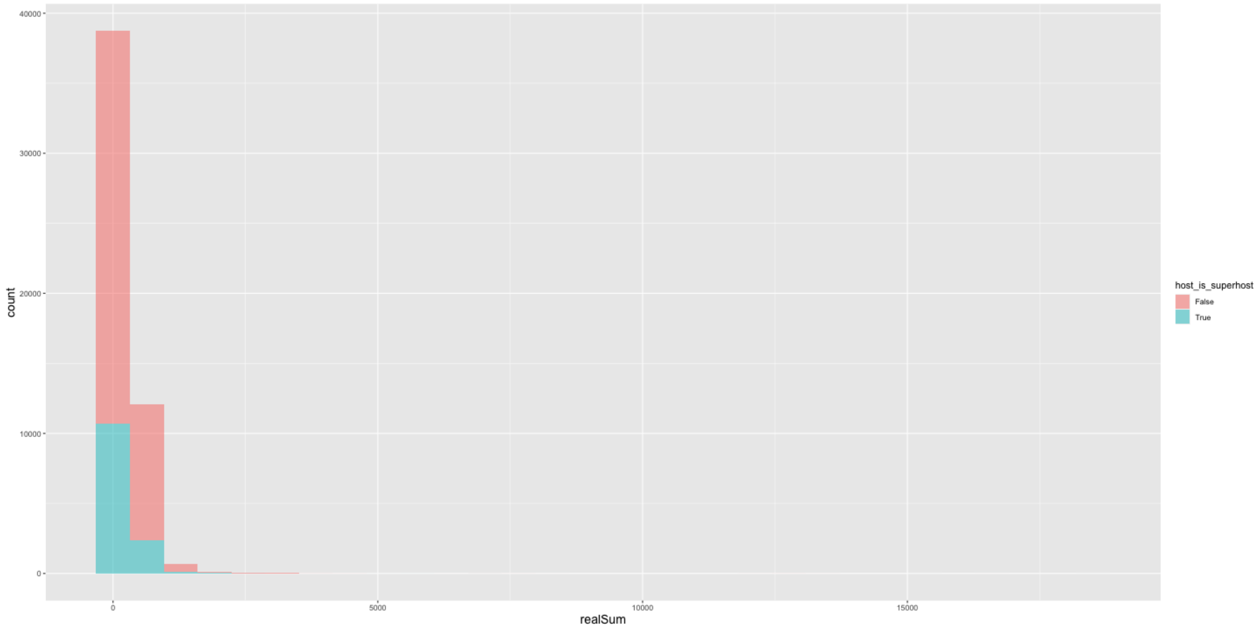

```r
ggplot(my_data_filtered, aes(x = realSum, fill = person_capacity,
    group = person_capacity)) + geom_histogram(alpha = 0.5, nbins = 20) +
    theme(axis.title.x = element_text(size = 14), axis.title.y = element_text(size = 14)) +
    facet_wrap(~city_day)
```



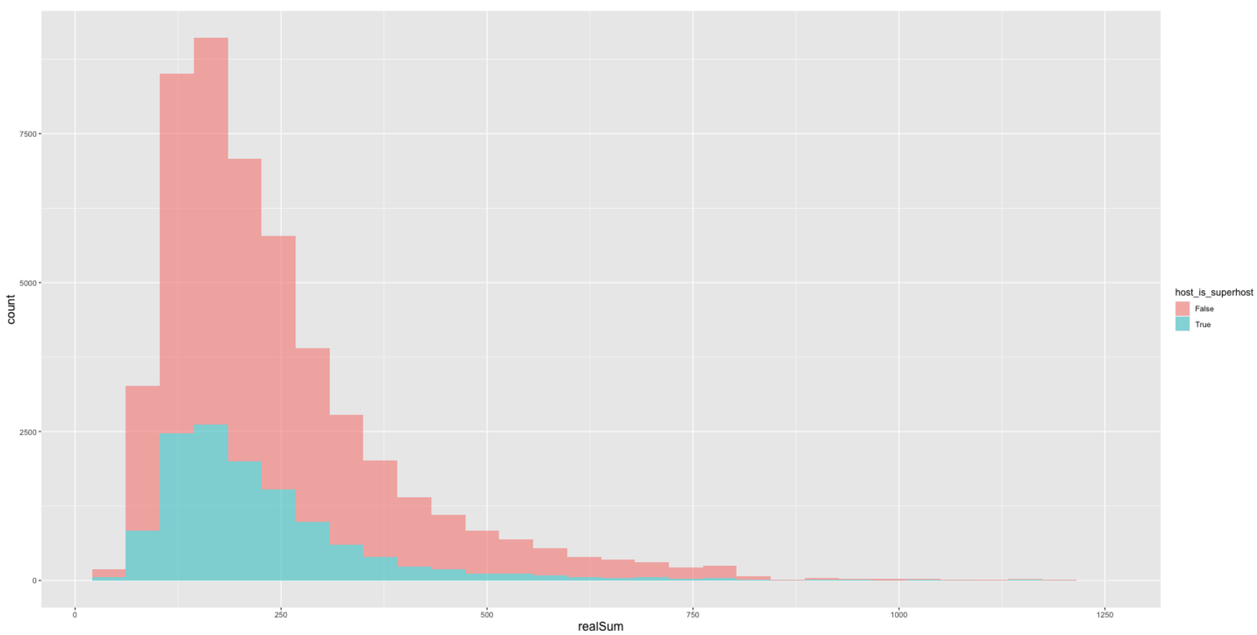
The overall price is distributed similarly across the spectrum irrespective of person_capacity. But for some cities like london, london_weekdays, lisbon the price is higher with person capacity. So, person capacity along with city will be an important variable for determining price.

Real Sum Vs host_is_superhost

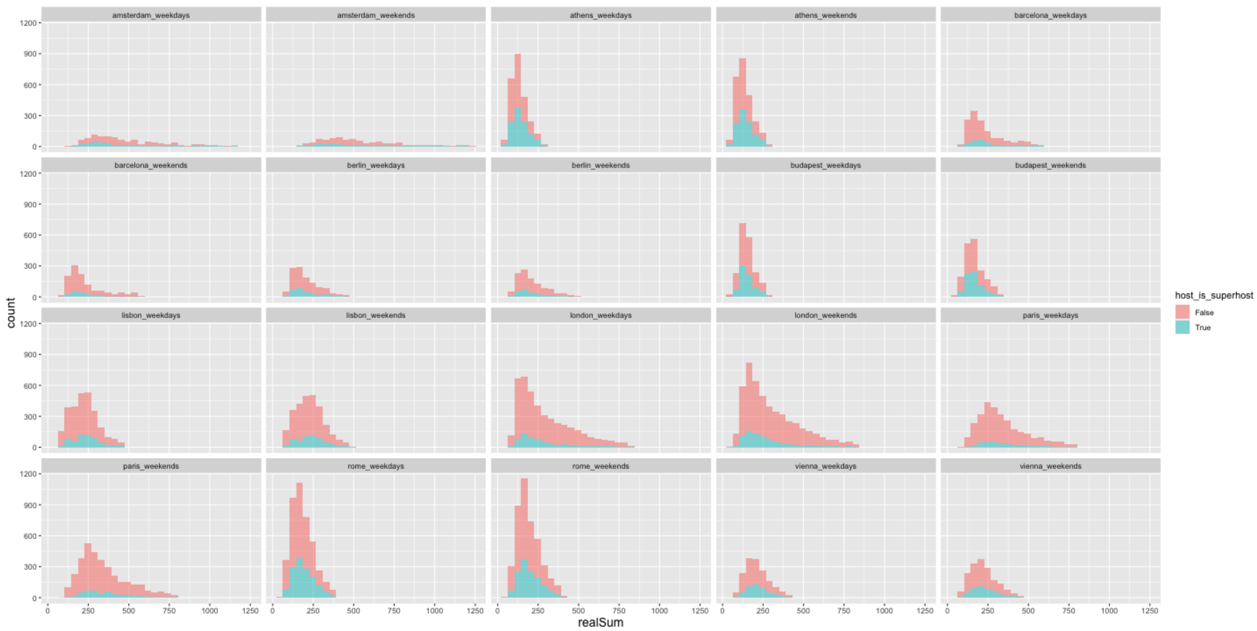
```
ggplot(my_data, aes(x = realSum, fill = host_is_superhost, group = host_is_superhost)) +
    geom_histogram(alpha = 0.5, nbins = 20) + theme(axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14))
```




```
ggplot(my_data_filtered, aes(x = realSum, fill = host_is_superhost,
    group = host_is_superhost)) + geom_histogram(alpha = 0.5,
    nbins = 20) + theme(axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14))
```



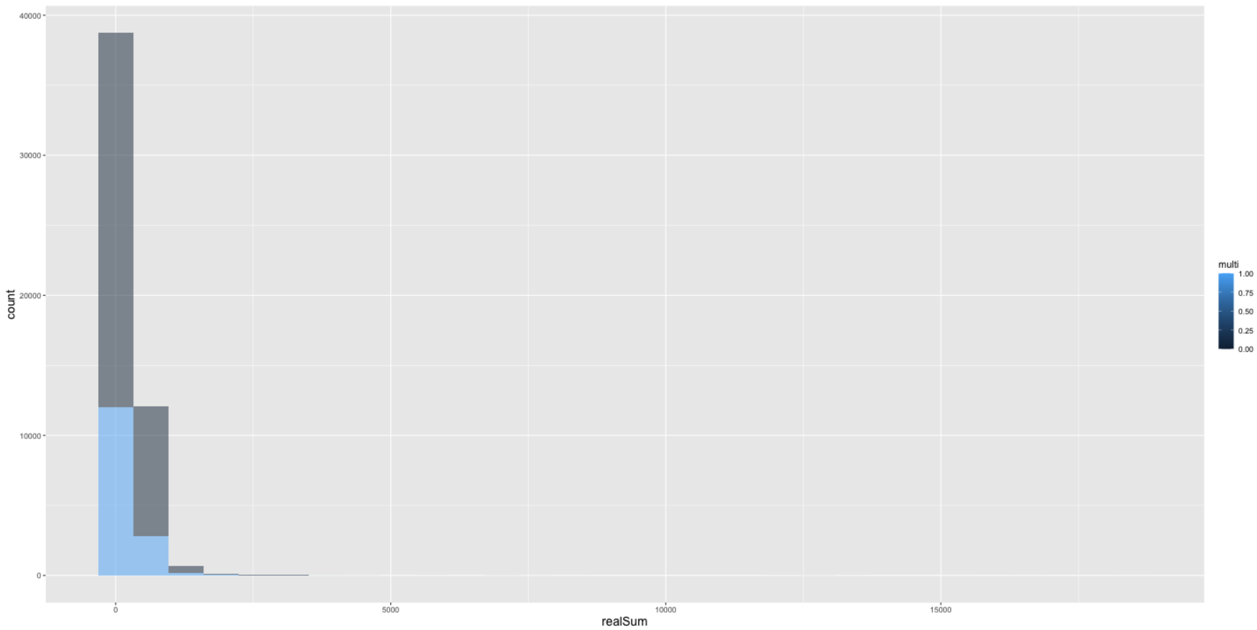
```r
ggplot(my_data_filtered, aes(x = realSum, fill = host_is_superhost,
    group = host_is_superhost)) + geom_histogram(alpha = 0.5,
    nbins = 20) + theme(axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14)) + facet_wrap(~city_day)
```



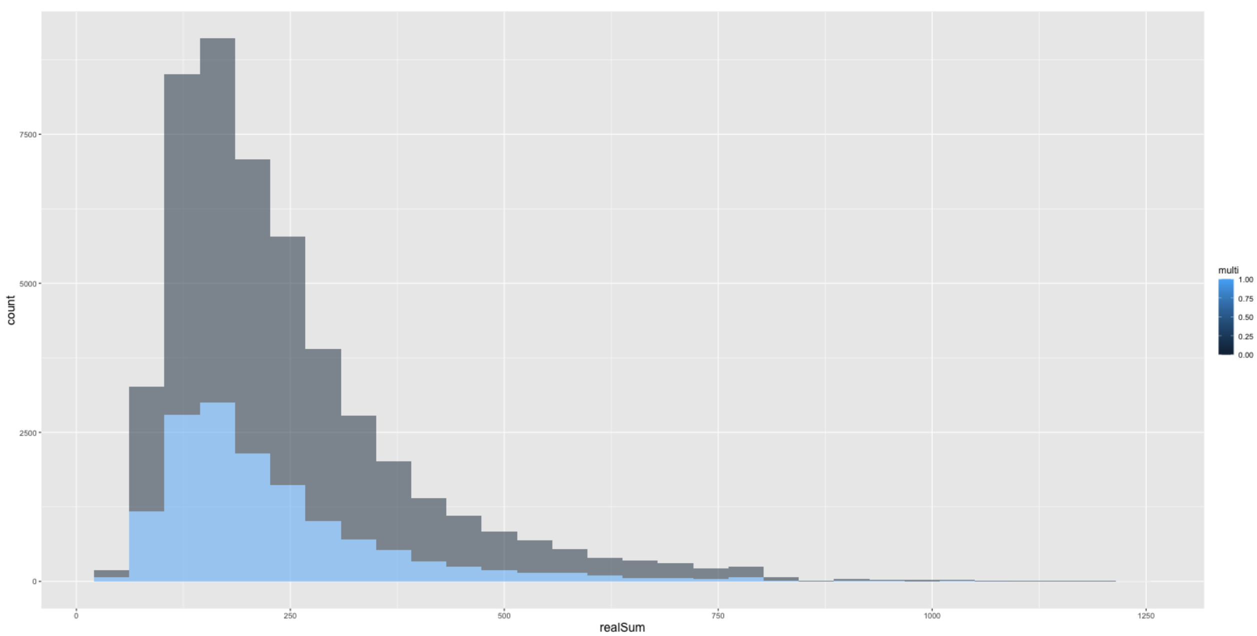
The prices are spread across all the spectrum irrespective of super_host or not.

Real Sum Vs multi

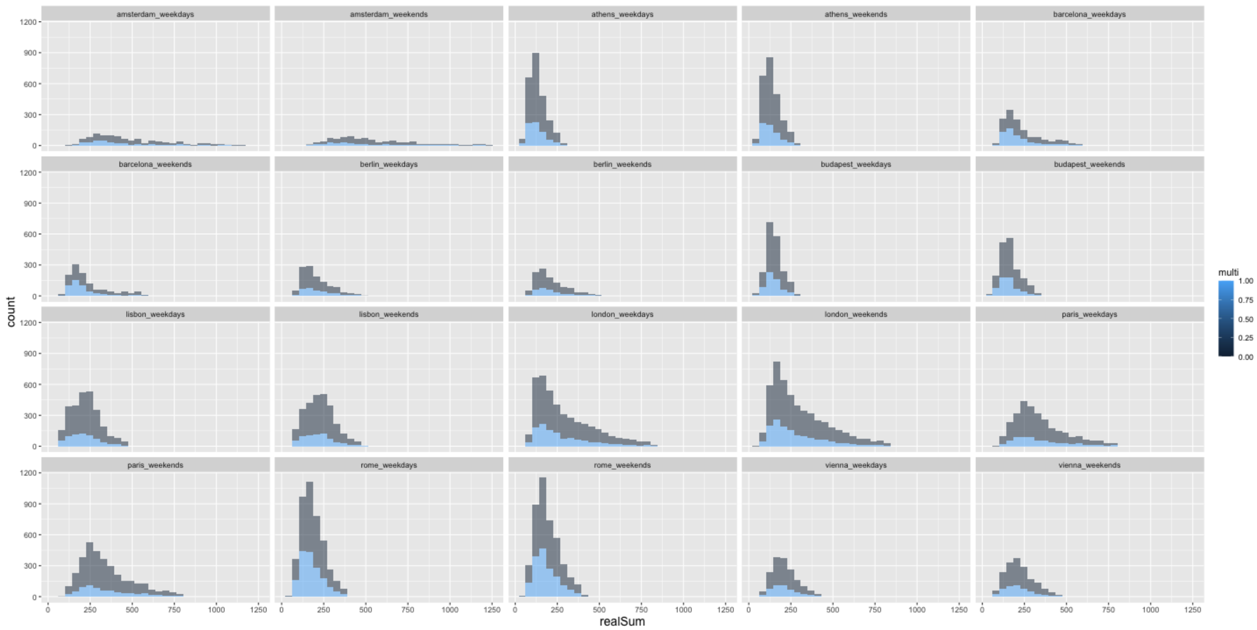
```r
ggplot(my_data, aes(x = realSum, fill = multi, group = multi)) +
    geom_histogram(alpha = 0.5, nbins = 20) + theme(axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14))
```




```r
ggplot(my_data_filtered, aes(x = realSum, fill = multi, group = multi)) +
    geom_histogram(alpha = 0.5, nbins = 20) + theme(axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14))
```



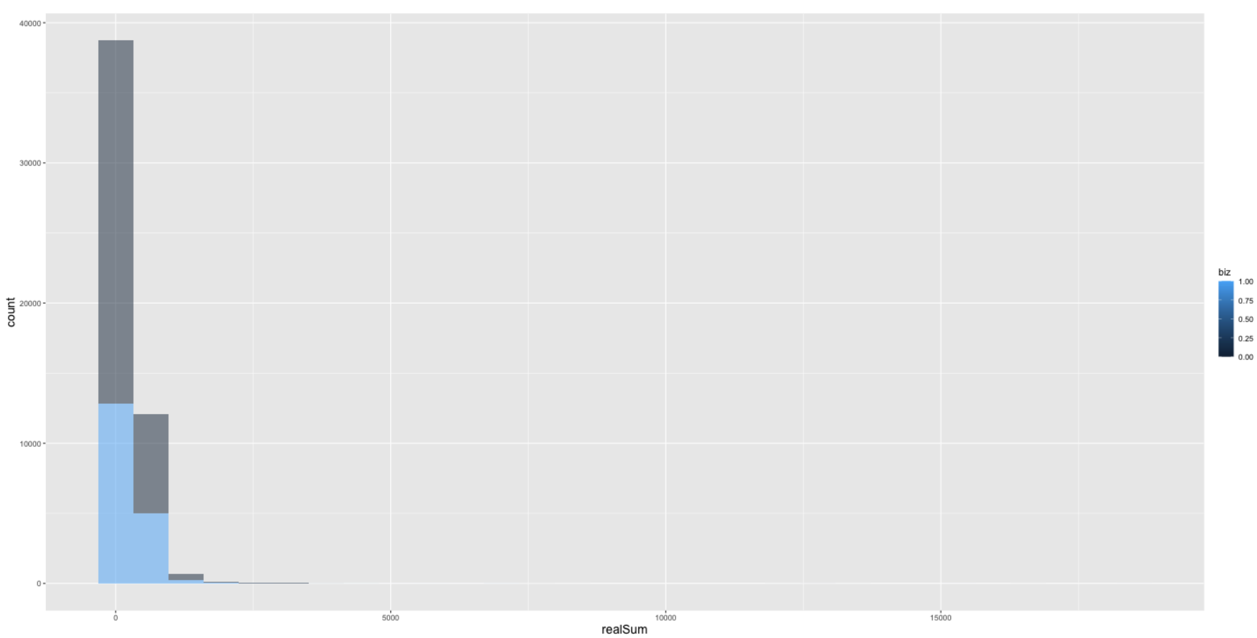

```r
ggplot(my_data_filtered, aes(x = realSum, fill = multi, group = multi)) +
    geom_histogram(alpha = 0.5, nbins = 20) + theme(axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14)) + facet_wrap(~city_day)
```



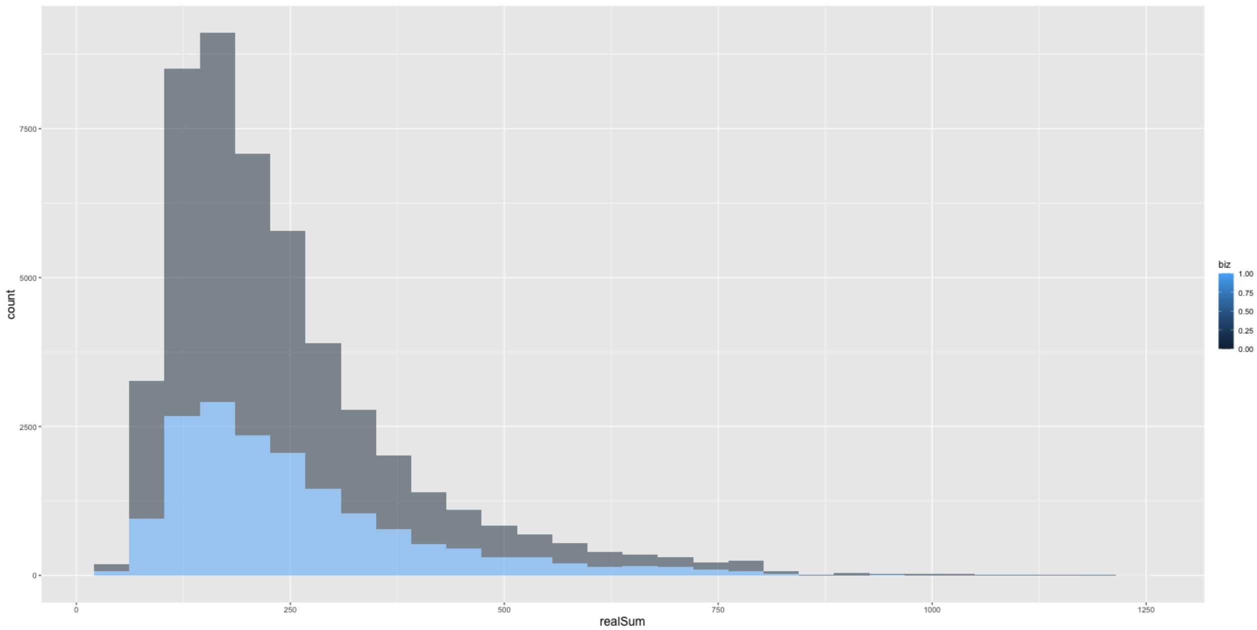
The prices are similar irrespective of multi or not.

Real Sum Vs biz

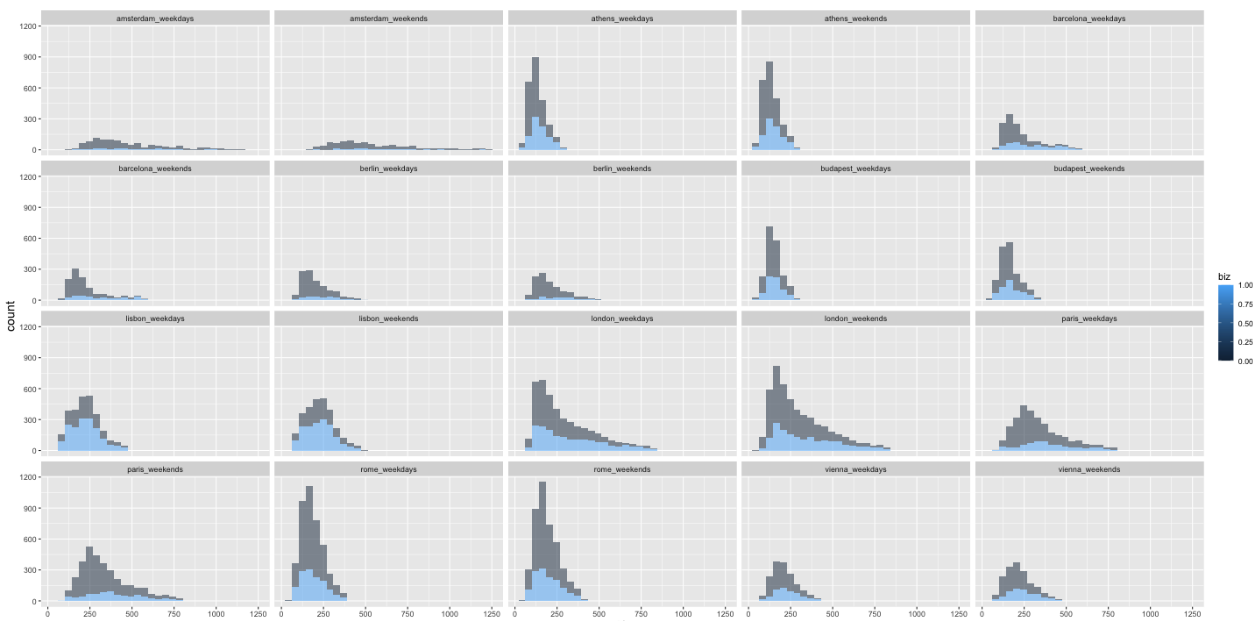
```r
ggplot(my_data, aes(x = realSum, fill = biz, group = biz)) +
    geom_histogram(alpha = 0.5, nbins = 20) + theme(axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14))
```




```r
ggplot(my_data_filtered, aes(x = realSum, fill = biz, group = biz)) +
    geom_histogram(alpha = 0.5, nbins = 20) + theme(axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14))
```



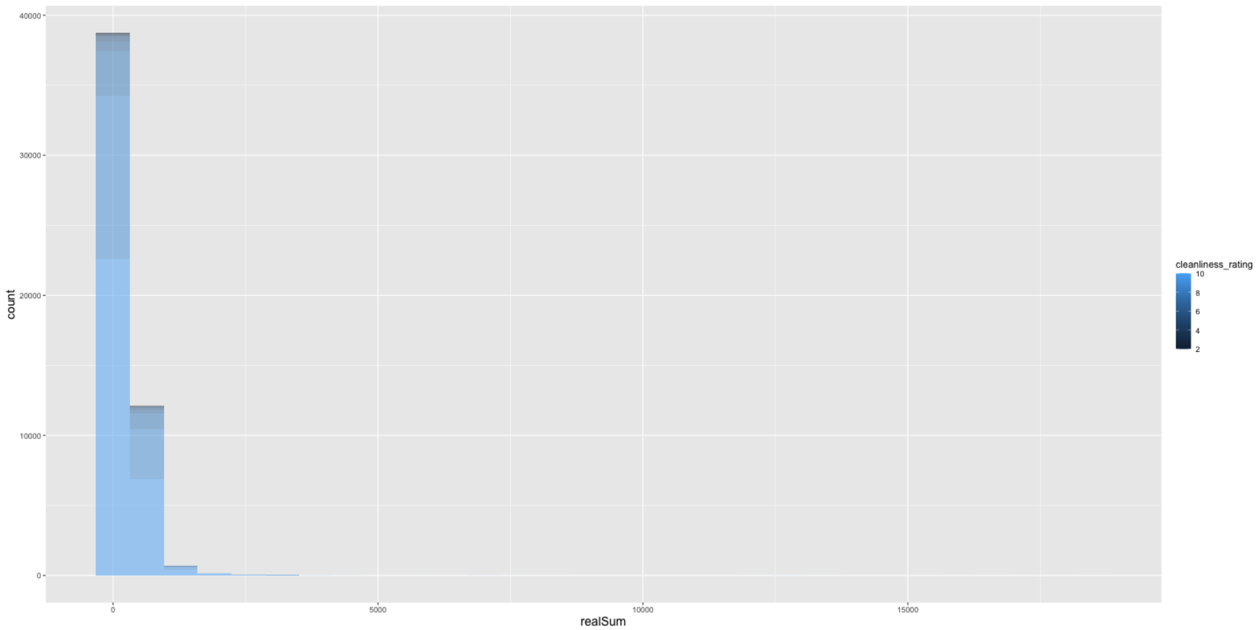

```
ggplot(my_data_filtered, aes(x = realSum, fill = biz, group = biz)) +
    geom_histogram(alpha = 0.5, nbins = 20) + theme(axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14)) + facet_wrap(~city_day)
```



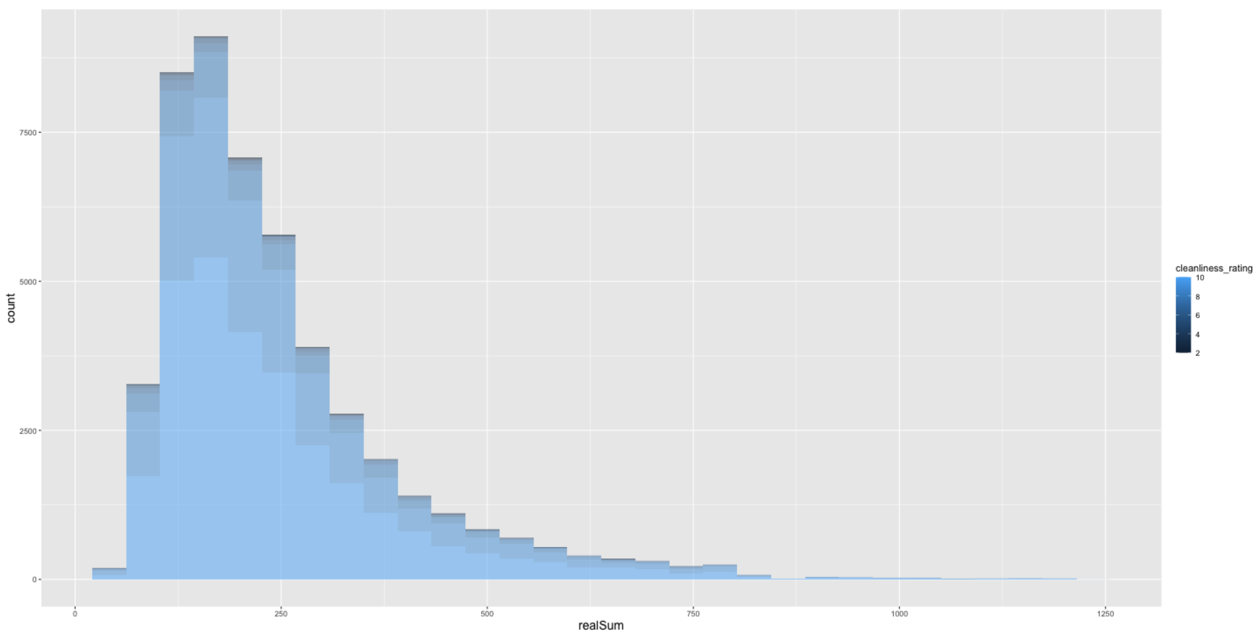
The prices are similar irrespective of biz or not.

Real Sum vs Cleanliness

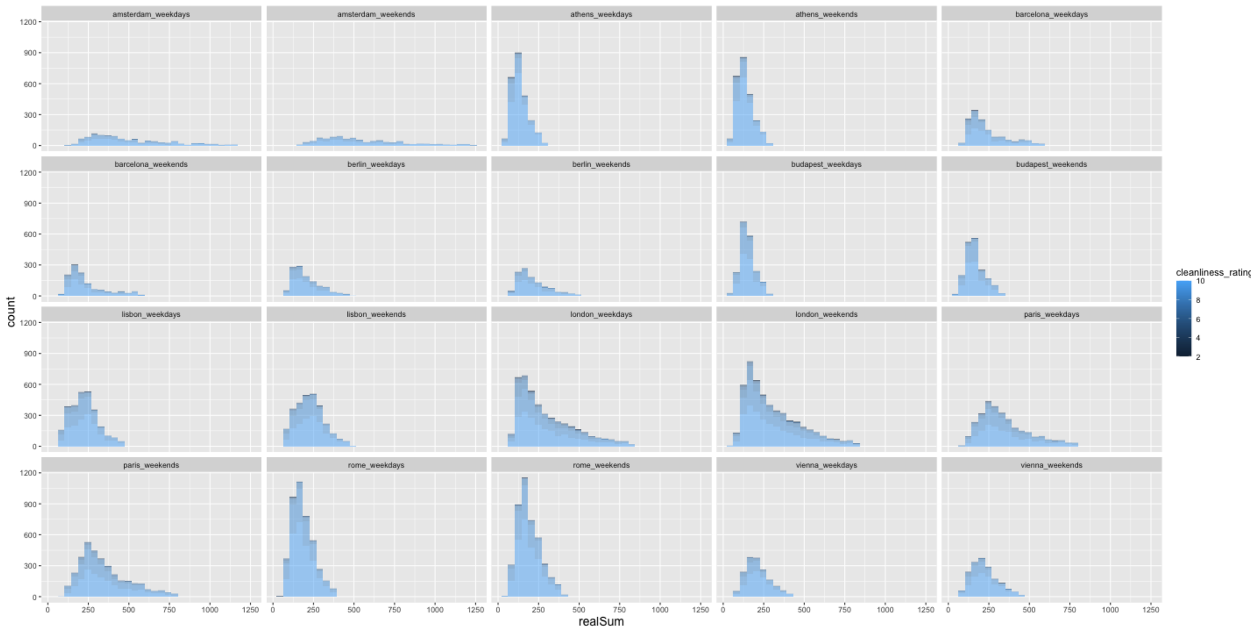
```
ggplot(my_data, aes(x = realSum, fill = cleanliness_rating, group = cleanliness_rating)) +
    geom_histogram(alpha = 0.5, nbins = 20) + theme(axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14))
```




```
ggplot(my_data_filtered, aes(x = realSum, fill = cleanliness_rating,
    group = cleanliness_rating)) + geom_histogram(alpha = 0.5,
    nbins = 20) + theme(axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14))
```



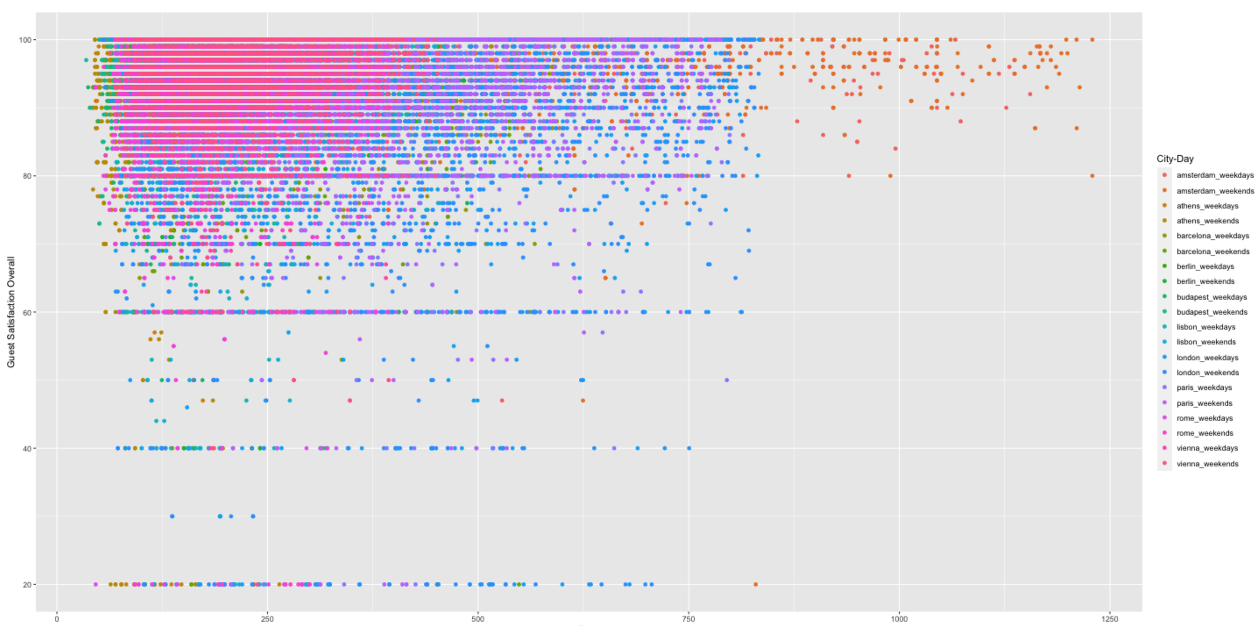

```
ggplot(my_data_filtered, aes(x = realSum, fill = cleanliness_rating,
    group = cleanliness_rating)) + geom_histogram(alpha = 0.5,
    nbins = 20) + theme(axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14)) + facet_wrap(~city_day)
```



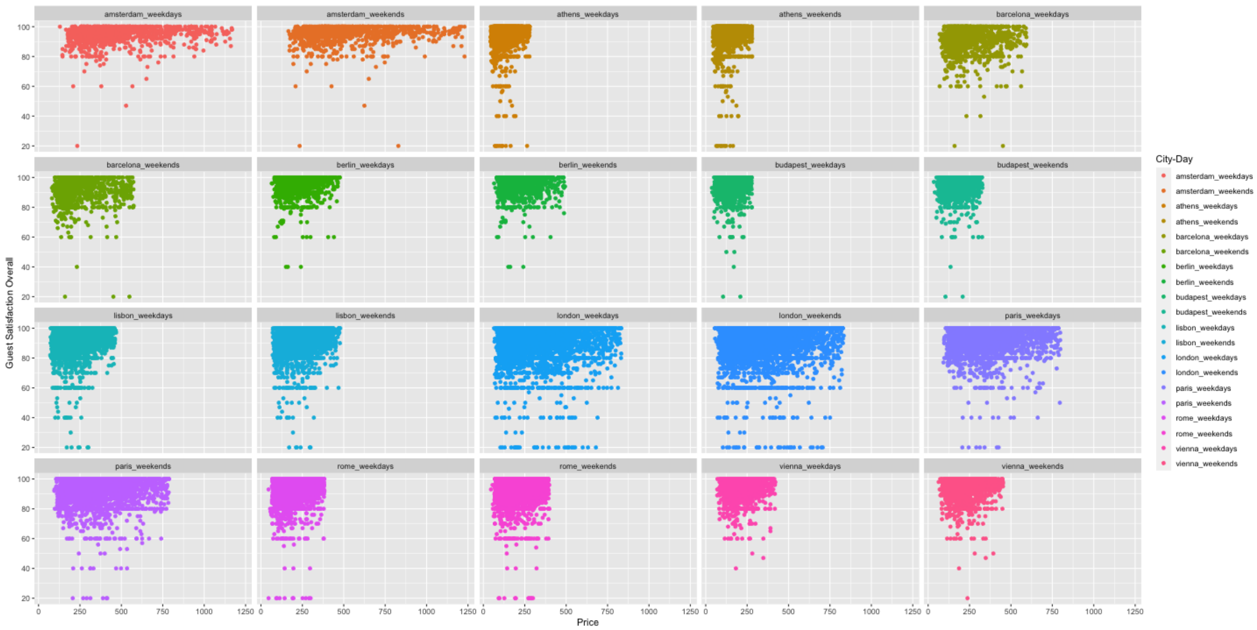
The cleanliness rating doesn't really have an effect on price

Scatterplot of Price vs Guest Satisfaction filtered by city

```
ggplot(my_data_filtered, aes(x = realSum, y = guest_satisfaction_overall,
    color = city_day)) + geom_point() + xlab("Price") + ylab("Guest Satisfaction Overall") +
    scale_color_discrete(name = "City-Day")
```



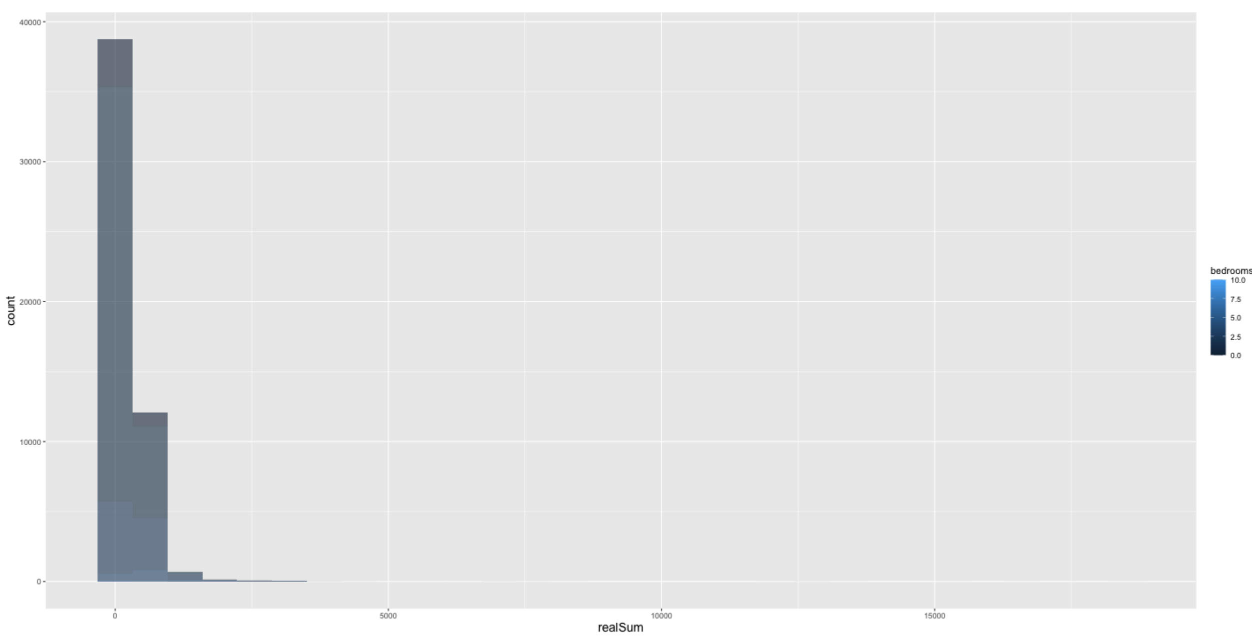
```
ggplot(my_data_filtered, aes(x = realSum, y = guest_satisfaction_overall,
    color = city_day)) + geom_point() + xlab("Price") + ylab("Guest Satisfaction Overall") +
    scale_color_discrete(name = "City-Day") + facet_wrap(~city_day)
```



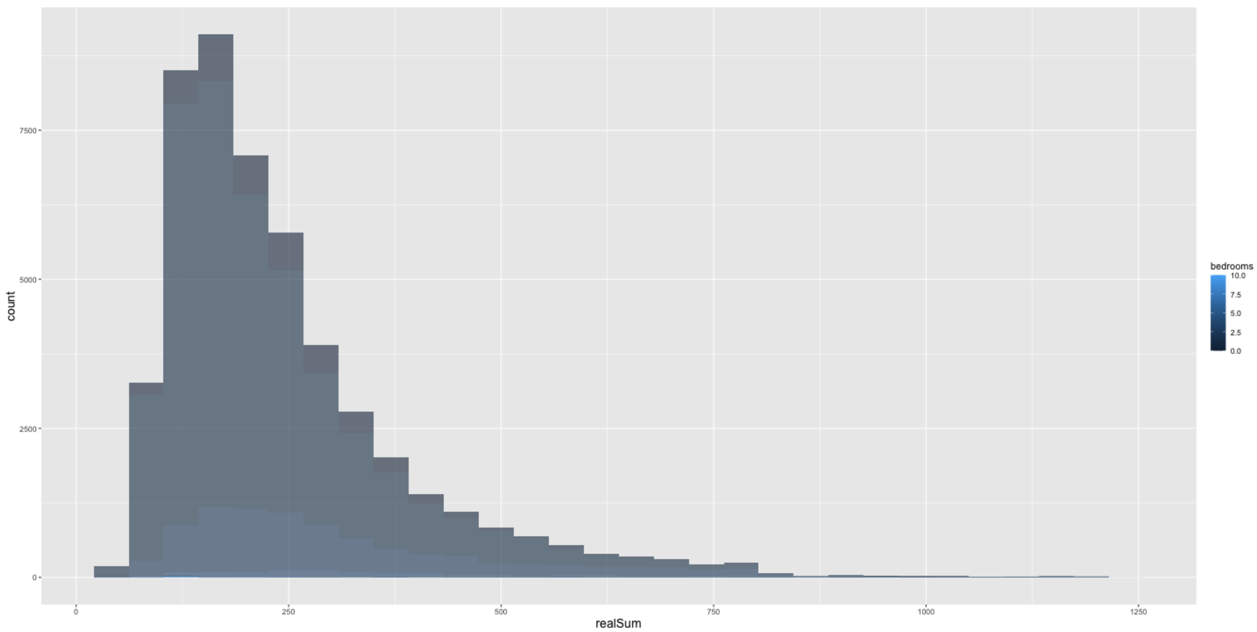
The plot depicts that there is no correlation of price with guest satisfaction, good satisfaction rate is found across all the prices. In some cities like london, we can see a group of reviews with low guest satisfaction.

Real Sum Vs Bedroom Count

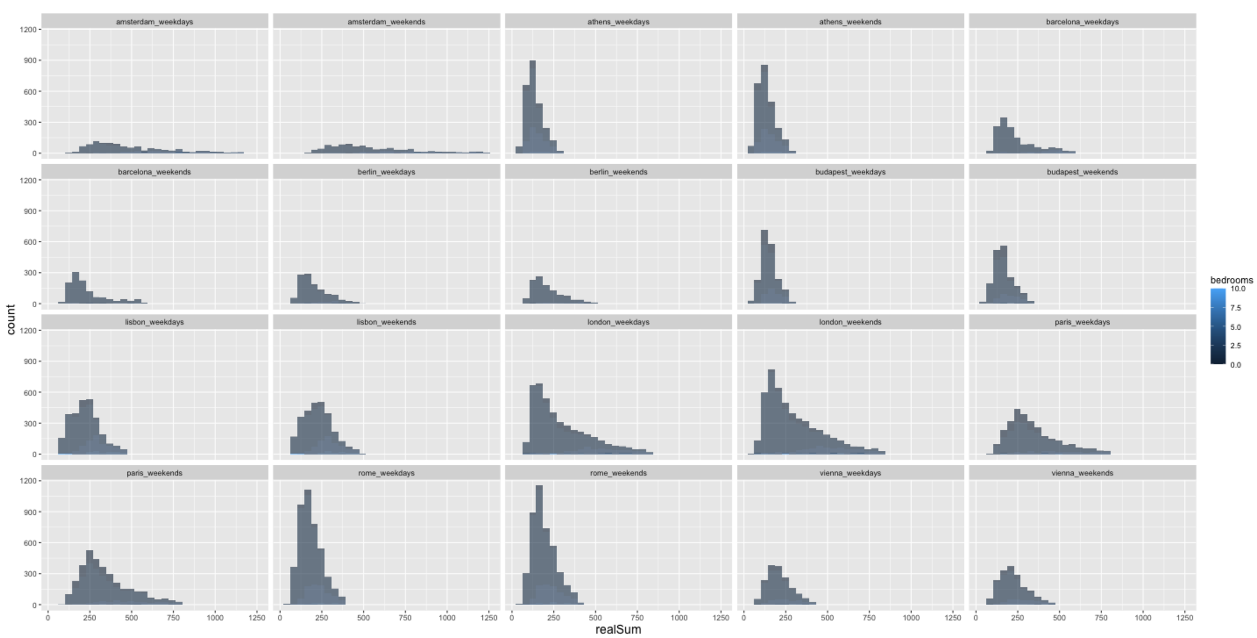
```
ggplot(my_data, aes(x = realSum, fill = bedrooms, group = bedrooms)) +
    geom_histogram(alpha = 0.6) + theme(axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14))
```



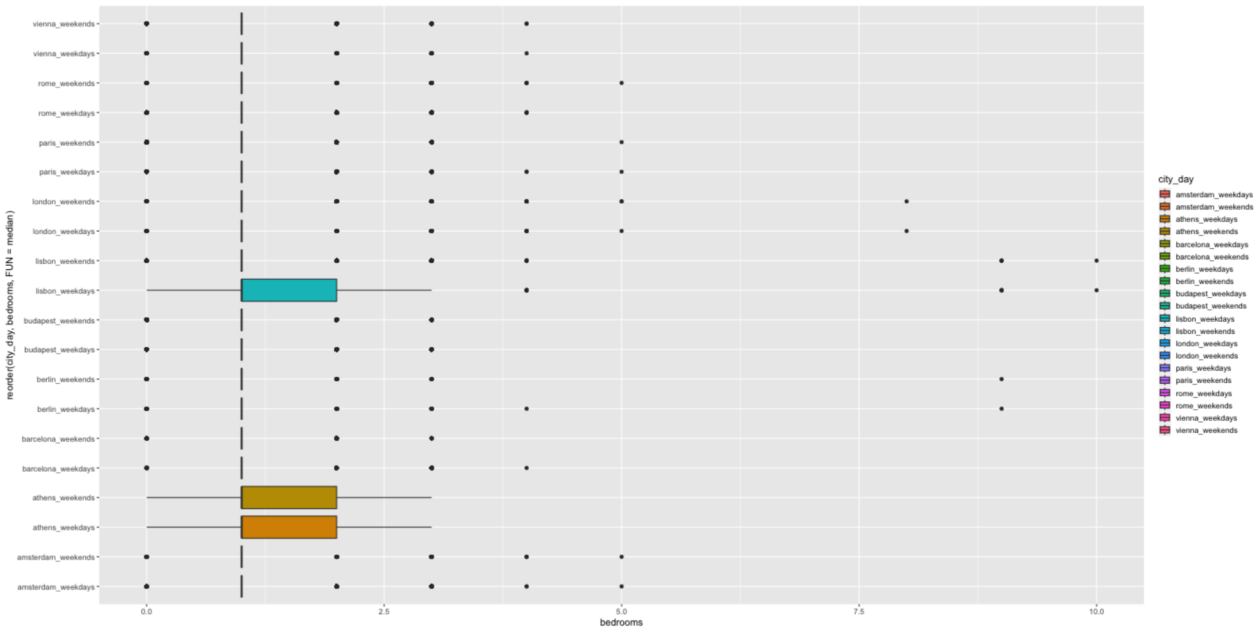

```
ggplot(my_data_filtered, aes(x = realSum, fill = bedrooms, group = bedrooms)) +
    geom_histogram(alpha = 0.6) + theme(axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14))
```




```
ggplot(my_data_filtered, aes(x = realSum, fill = bedrooms, group = bedrooms)) +
    geom_histogram(alpha = 0.6) + theme(axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14)) + facet_wrap(~city_day)
```




```
ggplot(my_data_filtered, aes(x = reorder(city_day, bedrooms,
    FUN = median), y = bedrooms, fill = city_day)) + geom_boxplot() +
    coord_flip() + theme(legend.key.height = unit(0.5, "cm"),
    legend.key.size = unit(1, "lines"))
```



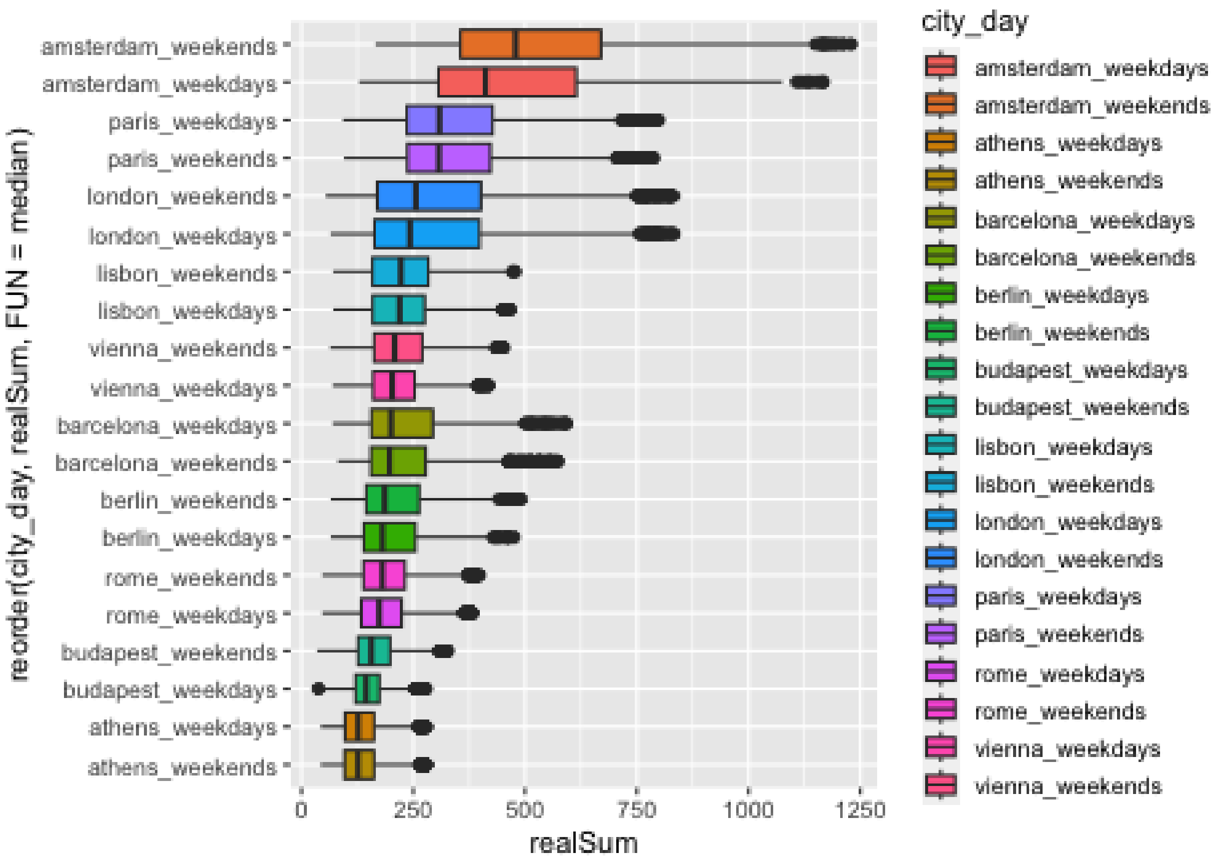

```
cor(as.numeric(factor(my_data$multi)), as.numeric(factor(my_data$biz)))
```

```
## [1] -0.4707248
```

Non Outlier Analysis

Boxplot of Price Vs City

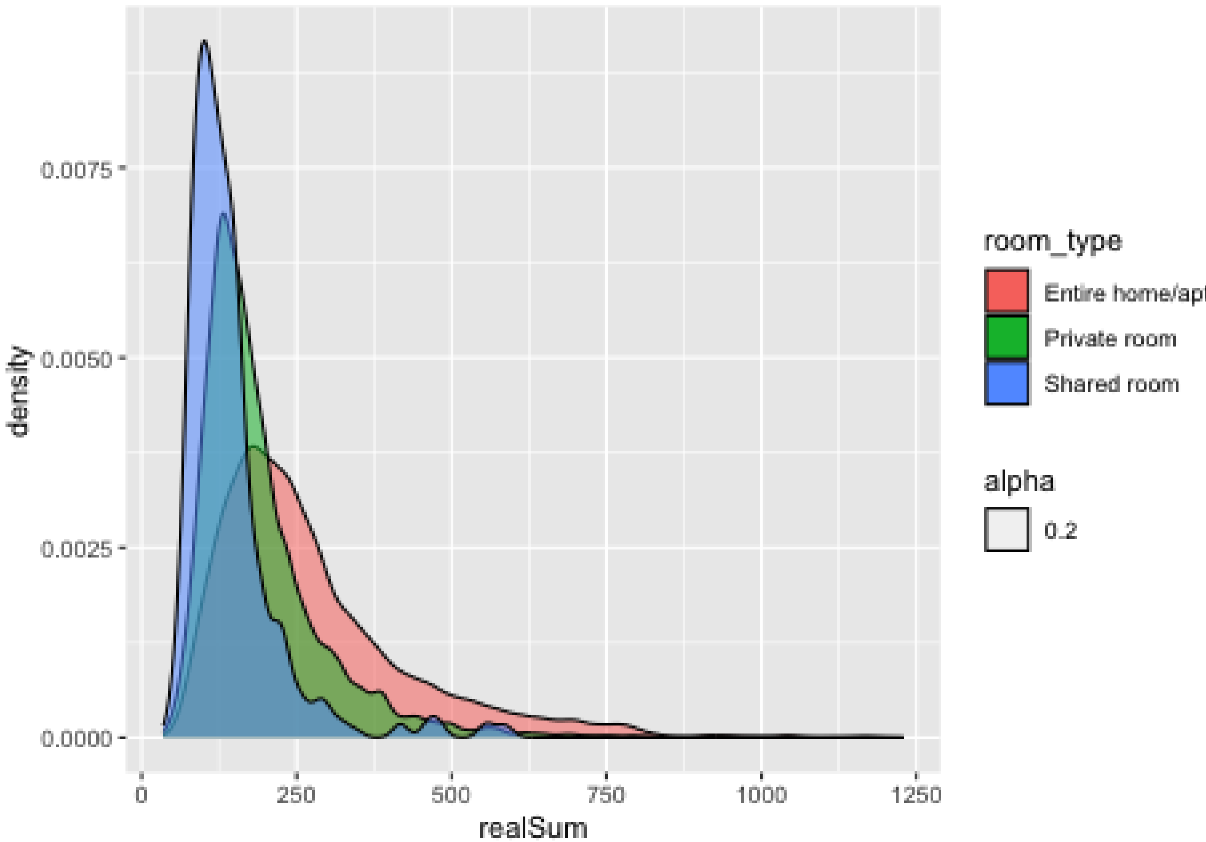
```
ggplot(my_data_filtered, aes(x = reorder(city_day, realSum, FUN = median),
    y = realSum, fill = city_day)) + geom_boxplot() + coord_flip() +
    theme(legend.key.height = unit(0.5, "cm"), legend.key.size = unit(1,
        "lines"))
```



The highest prices in Europe are found in Amsterdam.

Density plot of Price vs Room type

```
ggplot(my_data_filtered, aes(x = realSum, group = room_type,
    fill = room_type, alpha = 0.2)) + geom_density()
```



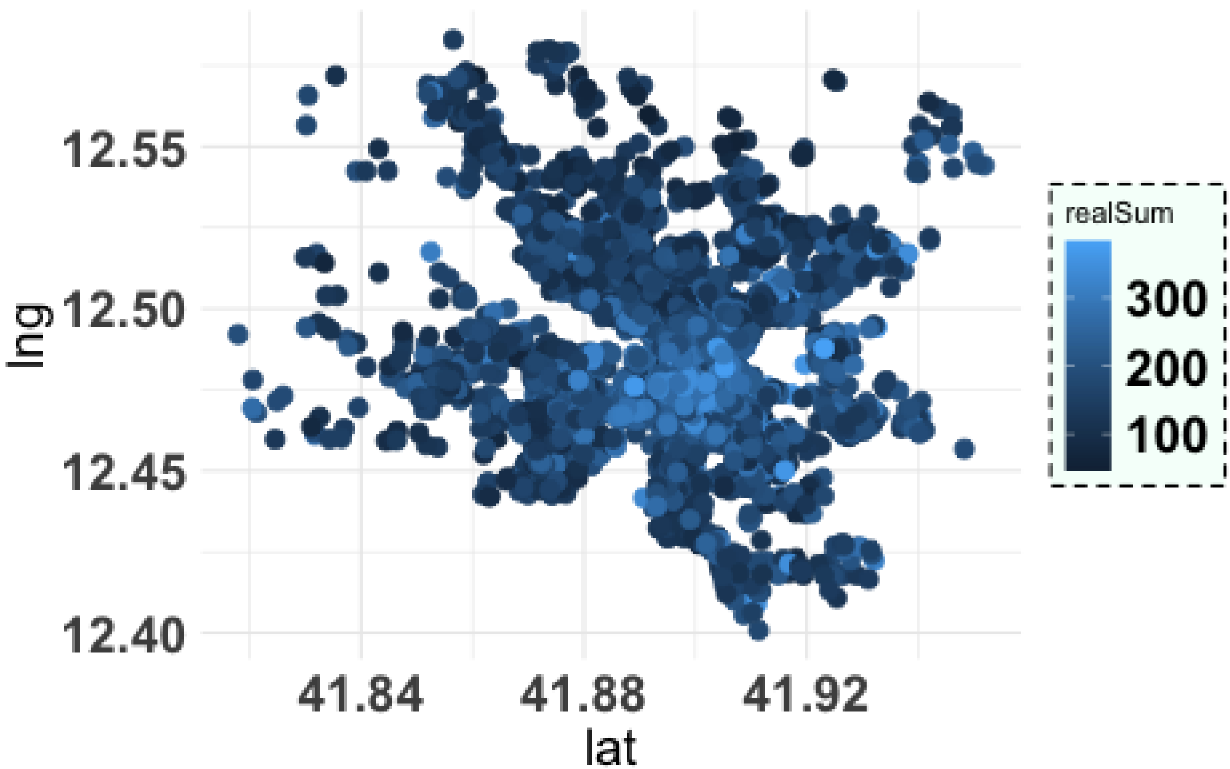
The prices of entire home are high comparatively

Scatterplot of Prices in Rome w.r.t Latitude and Longitude during weekdays

```
tema <- theme(plot.title = element_text(size = 23, hjust = 0.5),
    axis.text.x = element_text(size = 19, face = "bold"), axis.text.y = element_text(size = 19,
        face = "bold"), axis.title.x = element_text(size = 19),
    axis.title.y = element_text(size = 19), legend.text = element_text(colour = "black",
        size = 19, face = "bold"), legend.background = element_rect(fill = "#F5FFFA",
        size = 0.5, linetype = "dashed", colour = "black"))

rome_data <- my_data_filtered %>%
    subset(city_day == "rome_weekdays")

ggplot(data = rome_data, mapping = aes(x = lat, y = lng)) + theme_minimal() +
    scale_fill_identity() + geom_point(mapping = aes(color = realSum),
    size = 3) + ggtitle("") + tema
```

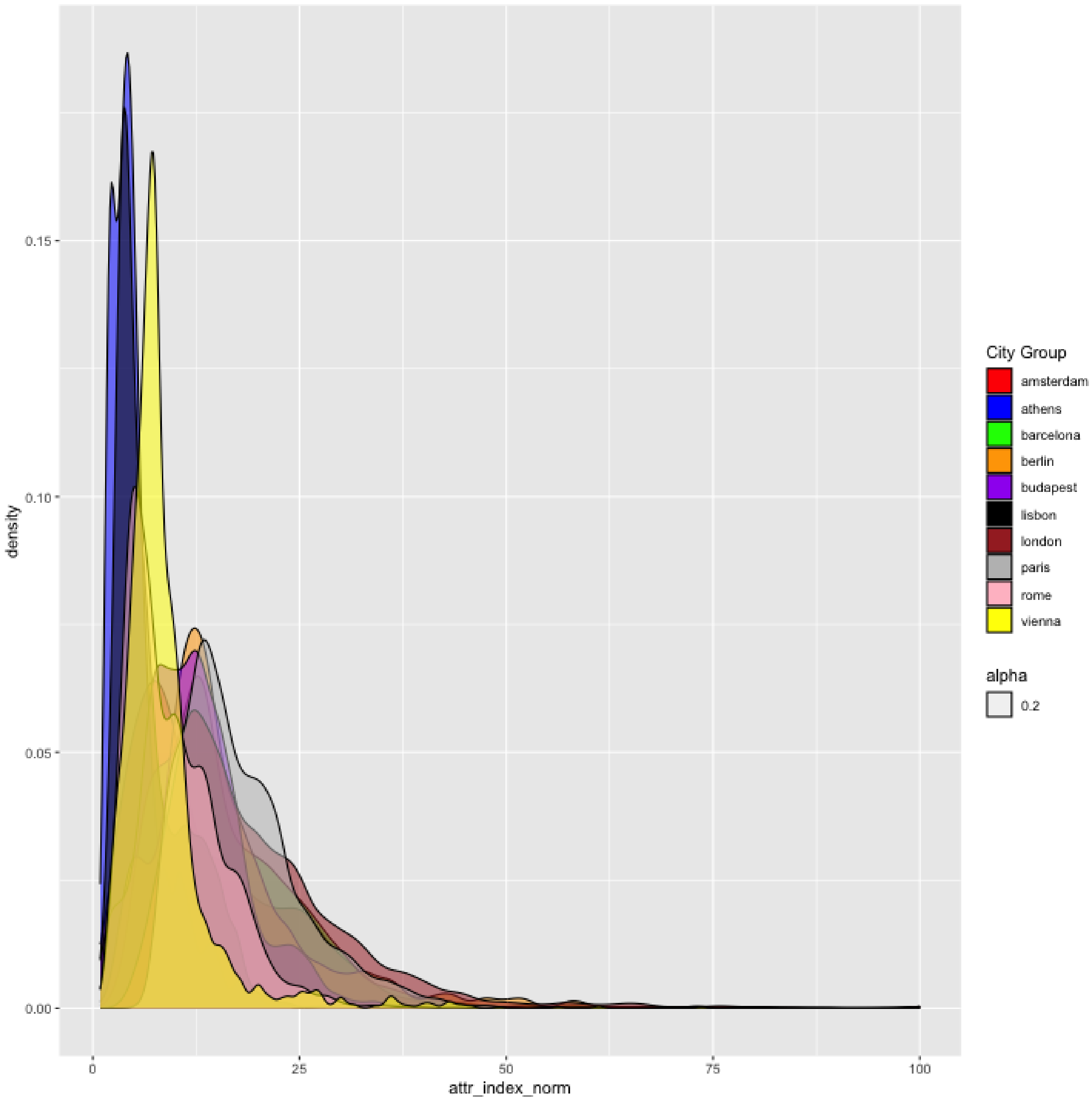


This plot is within expectations of general trends, which suggests similar types of establishments (price and hospitality) tend be in clusters.

Attraction Index in all Cities

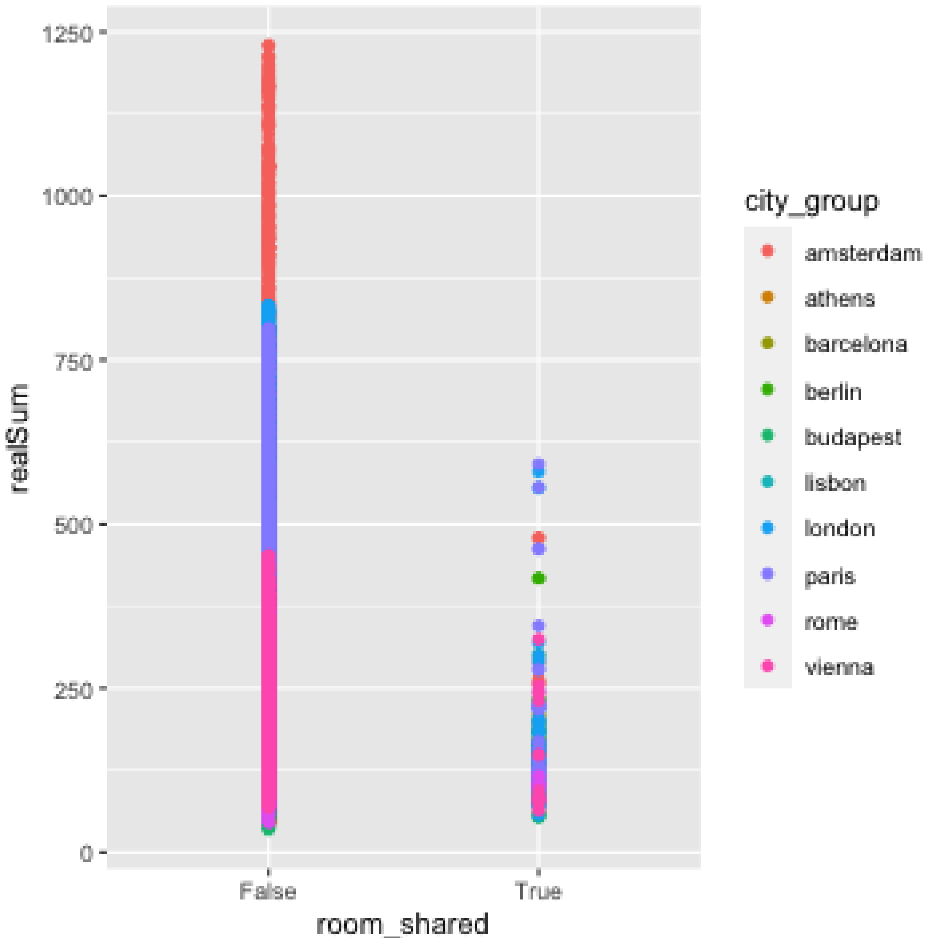
```
# create a new column that groups the cities
my_data_filtered$city_group <- ifelse(my_data_filtered$city_day %in%
    c("amsterdam_weekdays", "amsterdam_weekends"), "amsterdam",
    ifelse(my_data_filtered$city_day %in% c("athens_weekdays",
        "athens_weekends"), "athens", ifelse(my_data_filtered$city_day %in%
        c("barcelona_weekdays", "barcelona_weekends"), "barcelona",
        ifelse(my_data_filtered$city_day %in% c("berlin_weekdays",
            "berlin_weekends"), "berlin", ifelse(my_data_filtered$city_day %in%
            c("budapest_weekdays", "budapest_weekends"), "budapest",
            ifelse(my_data_filtered$city_day %in% c("lisbon_weekdays",
                "lisbon_weekends"), "lisbon", ifelse(my_data_filtered$city_day %in%
                c("london_weekdays", "london_weekends"), "london",
                ifelse(my_data_filtered$city_day %in% c("paris_weekdays",
                  "paris_weekends"), "paris", ifelse(my_data_filtered$city_day %in%
                  c("rome_weekdays", "rome_weekends"), "rome",
                  "vienna")))))))))

# plot the density plot with the new groupings
ggplot(my_data_filtered, aes(x = attr_index_norm, fill = city_group,
    alpha = 0.2)) + geom_density() + scale_fill_manual(values = c(amsterdam = "red",
    athens = "blue", barcelona = "green", berlin = "orange",
    budapest = "purple", lisbon = "black", london = "brown",
    paris = "grey", rome = "pink", vienna = "yellow")) + labs(fill = "City Group")
```



Real Sum vs Room Shared for all Cities

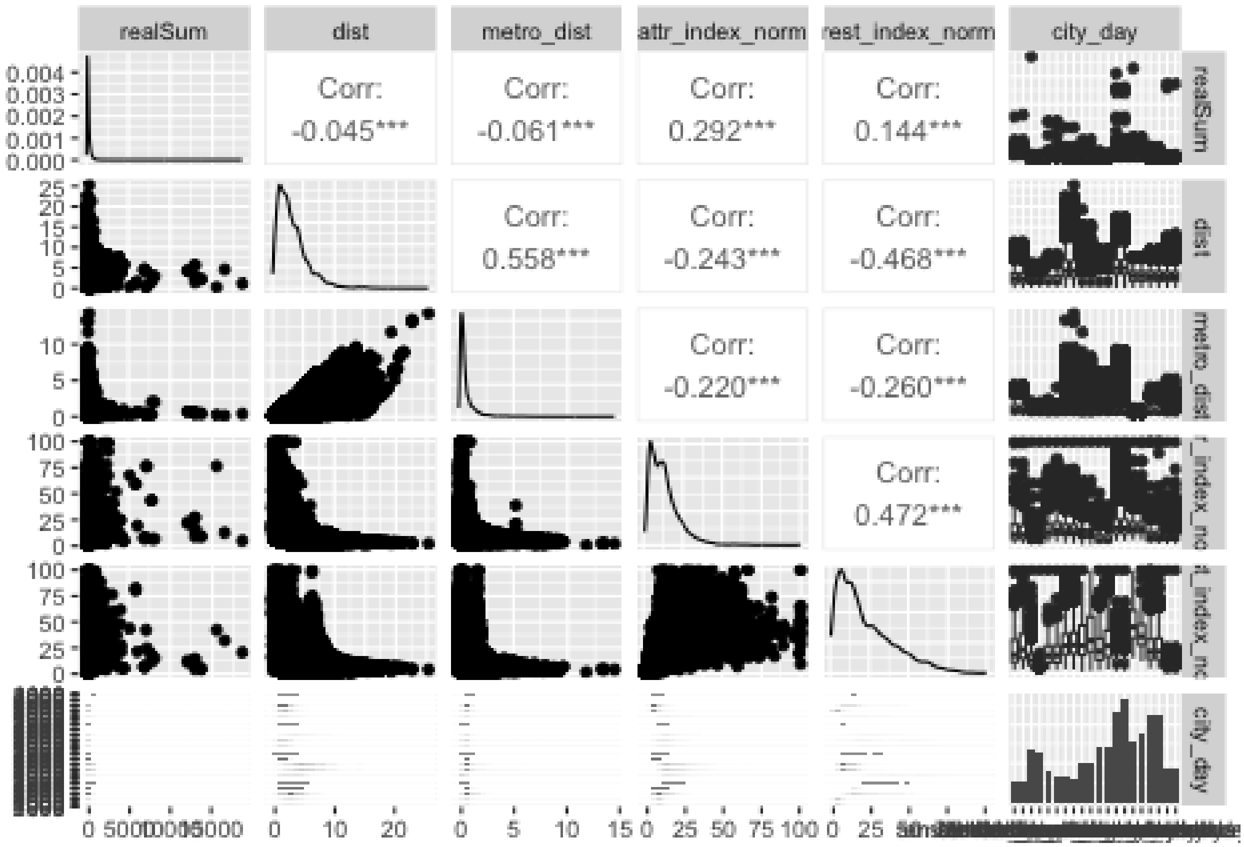
```
ggplot(my_data_filtered, aes(x = room_shared, y = realSum, color = city_group),
    alpha = 0.001) + geom_point() + scale_fill_manual(values = c(amsterdam = "red",
    athens = "blue", barcelona = "green", berlin = "orange",
    budapest = "purple", lisbon = "black", london = "brown",
    paris = "grey", rome = "pink", vienna = "yellow")) + labs(fill = "City Group")
```



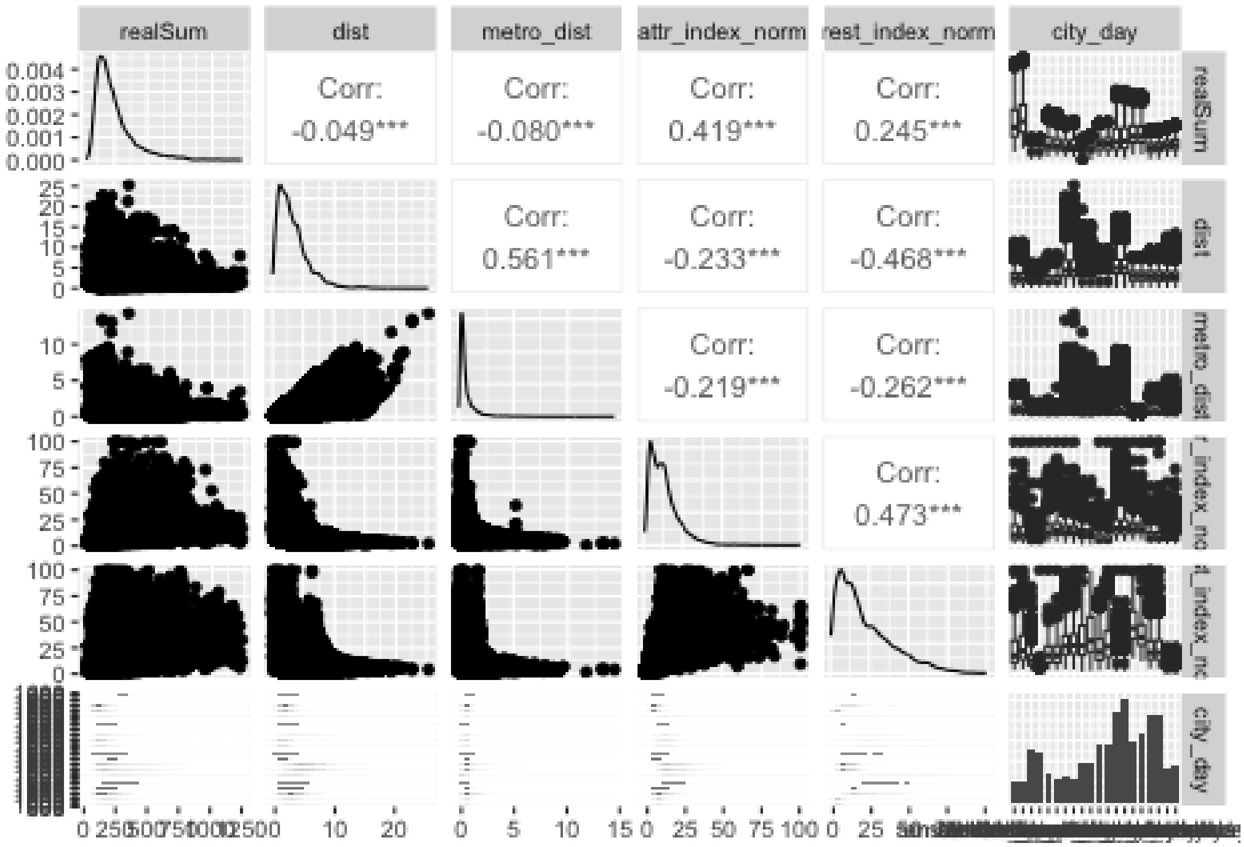
Different Model Selection and Training

Checking for correlations between different attributes

```
ggpairs(my_data[c("realSum", "dist", "metro_dist", "attr_index_norm",
    "rest_index_norm", "city_day")], cardinality = 20, cardinality_threshold = 999)
```



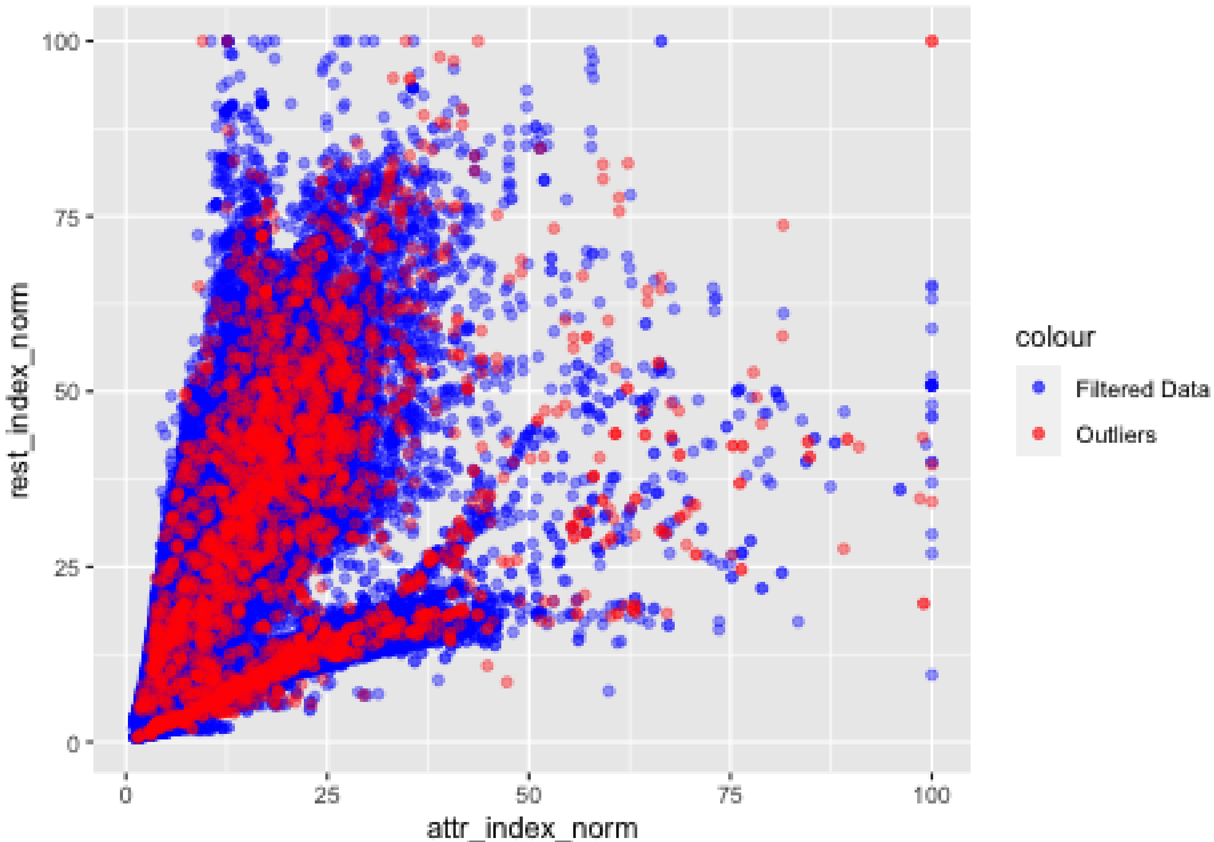

```r
ggpairs(my_data_filtered[c("realSum", "dist", "metro_dist", "attr_index_norm",
    "rest_index_norm", "city_day")], cardinality = 20, cardinality_threshold = 999)
```



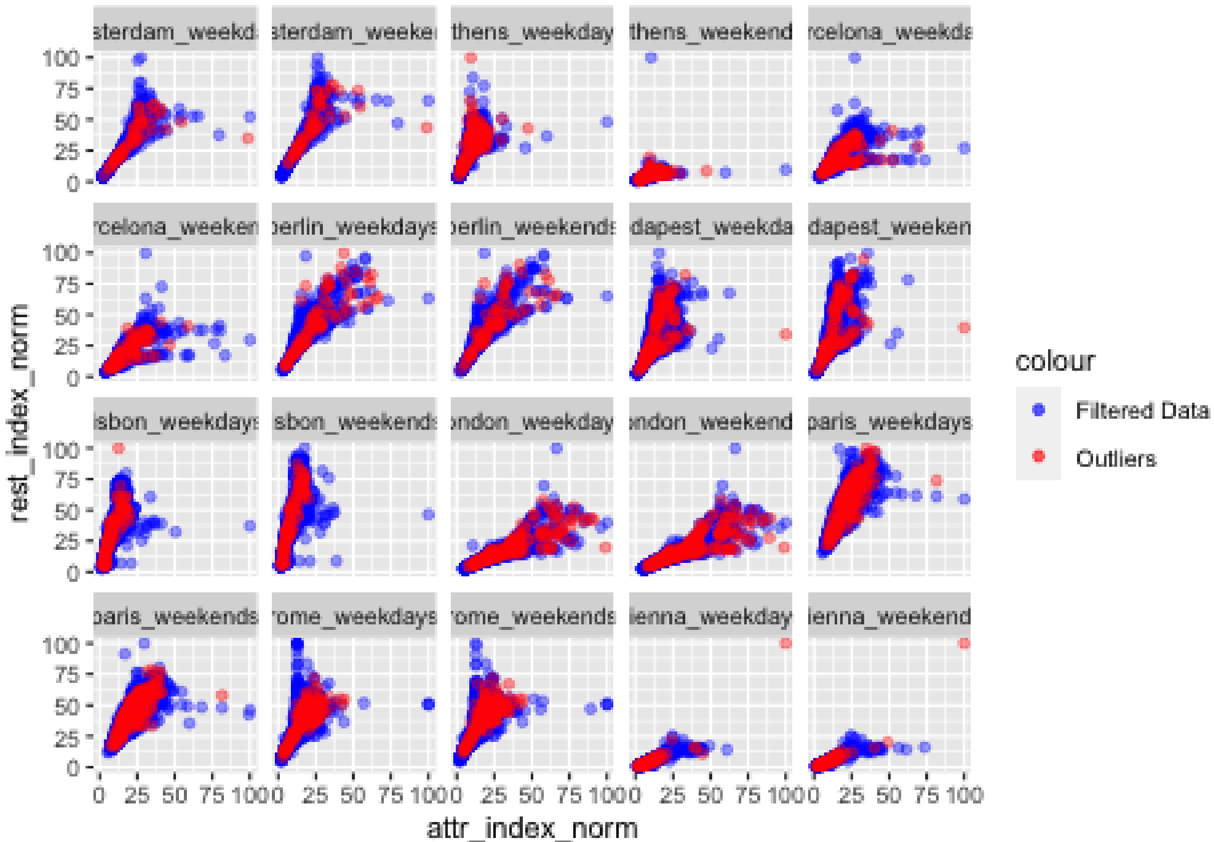

```
cor(my_data$attr_index, my_data$rest_index)
```

```
## [1] 0.4721427
```

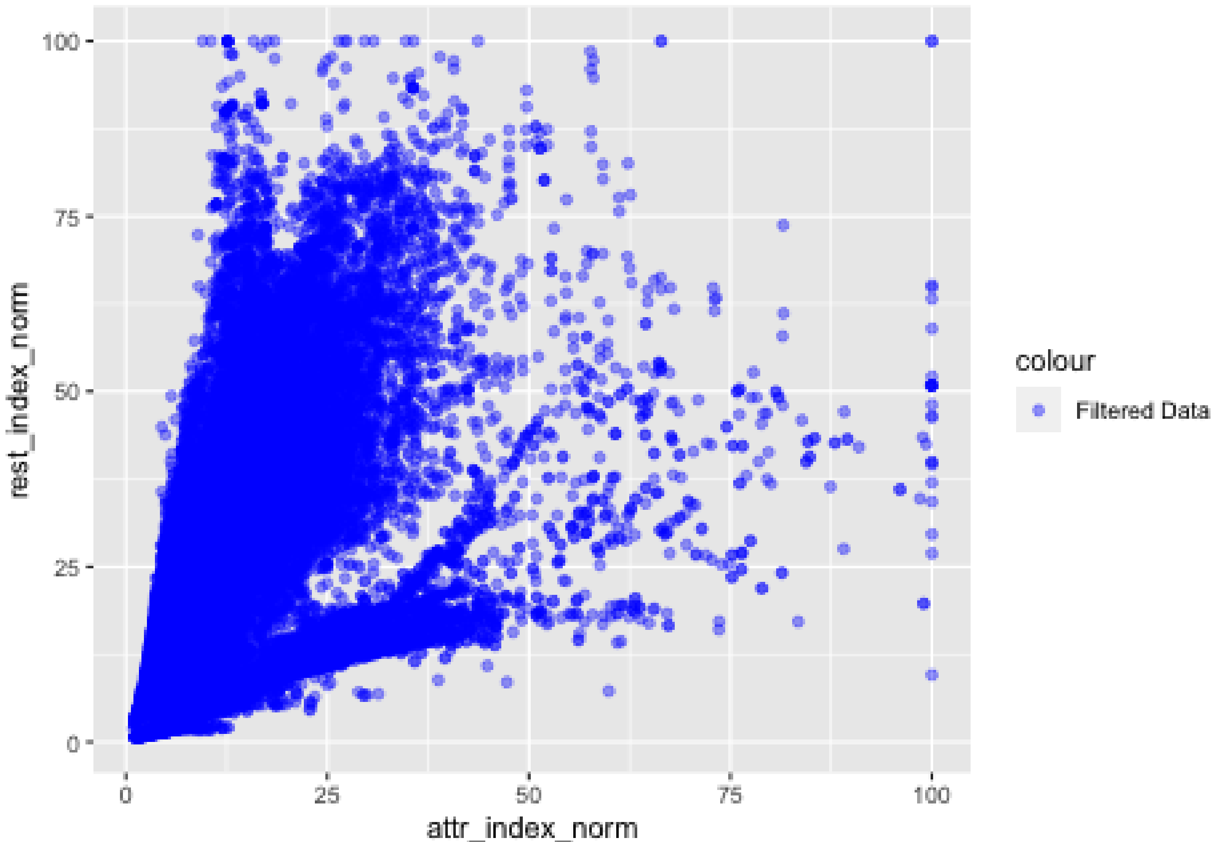
```
ggplot() + geom_point(data = my_data_filtered, aes(x = attr_index_norm,
    y = rest_index_norm, color = "Filtered Data"), alpha = 0.4) +
    geom_point(data = my_outliers, aes(x = attr_index_norm, y = rest_index_norm,
        color = "Outliers"), alpha = 0.4) + scale_color_manual(values = c(`Filtered Data` = "blue",
    Outliers = "red"))
```



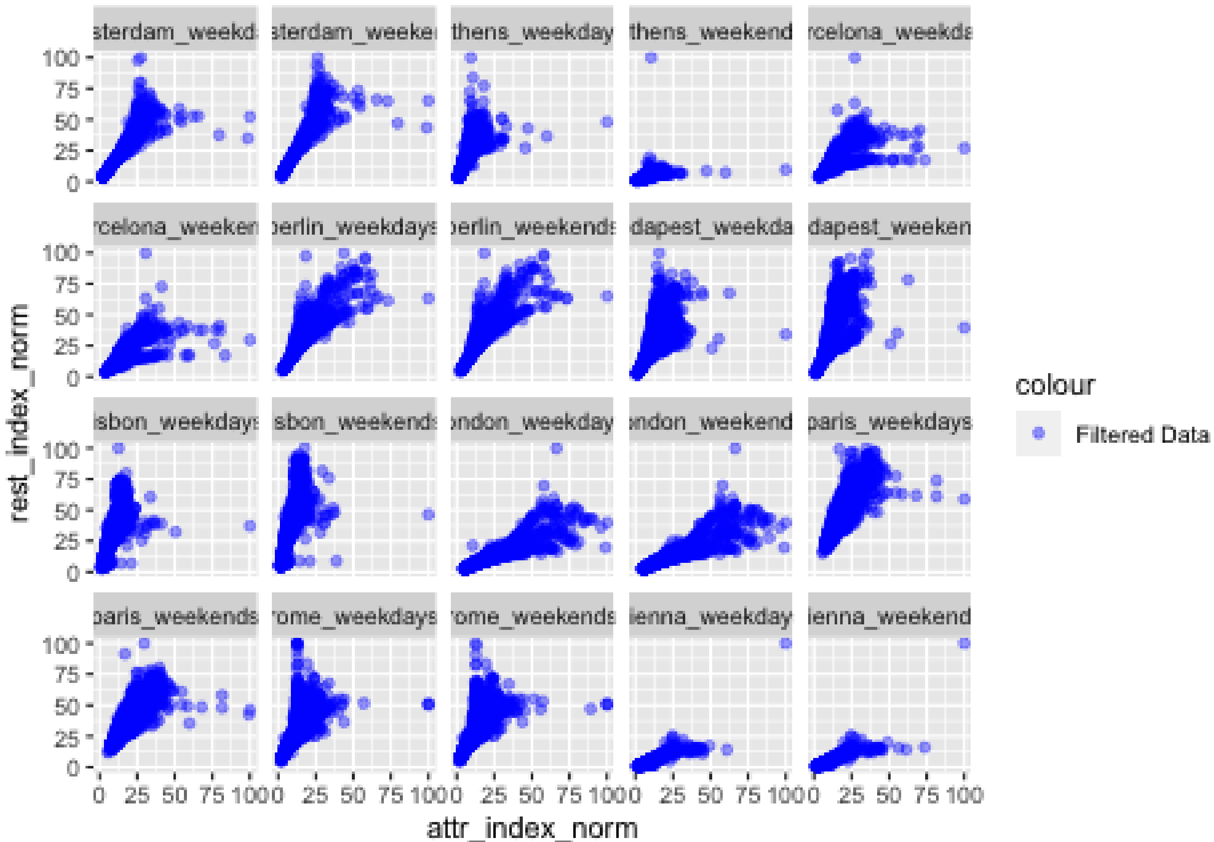

```
ggplot() + geom_point(data = my_data_filtered, aes(x = attr_index_norm,
    y = rest_index_norm, color = "Filtered Data"), alpha = 0.4) +
    geom_point(data = my_outliers, aes(x = attr_index_norm, y = rest_index_norm,
        color = "Outliers"), alpha = 0.4) + scale_color_manual(values = c(`Filtered Data` = "blue",
    Outliers = "red")) + facet_wrap(~city_day)
```




```
ggplot() + geom_point(data = my_data, aes(x = attr_index_norm,
    y = rest_index_norm, color = "Filtered Data"), alpha = 0.4) +
    scale_color_manual(values = c(`Filtered Data` = "blue"))
```




```
ggplot() + geom_point(data = my_data, aes(x = attr_index_norm,
    y = rest_index_norm, color = "Filtered Data"), alpha = 0.4) +
    scale_color_manual(values = c(`Filtered Data` = "blue")) +
    facet_wrap(~city_day)
```




```
cor(my_data$attr_index_norm, my_data$rest_index_norm)
```

```
## [1] 0.4721427
```

Linear, Polynomial and Step Regression

MLR Seperated by City Day

```
temp_data <- subset(my_data_train, city_day == "amsterdam_weekends" |
    city_day == "amsterdam_weekdays")

M_0 <- lm(realSum ~ . - realSum - X, data = temp_data)
temp_data <- subset(my_data_train, city_day == "athens_weekdays" |
    city_day == "athens_weekends", )

M_1 <- lm(realSum ~ . - realSum - X - city_day, data = temp_data)
temp_data <- subset(my_data_train, city_day == "barcelona_weekdays" |
    city_day == "barcelona_weekends", )

M_2 <- lm(realSum ~ . - realSum - X - city_day, data = temp_data)
temp_data <- subset(my_data_train, city_day == "berlin_weekdays" |
    city_day == "berlin_weekends", )

M_3 <- lm(realSum ~ . - realSum - X - city_day, data = temp_data)
```

```r
temp_data <- subset(my_data_train, city_day == "budapest_weekdays" |
    city_day == "budapest_weekends", )

M_4 <- lm(realSum ~ . - realSum - X - city_day, data = temp_data)
temp_data <- subset(my_data_train, city_day == "lisbon_weekdays" |
    city_day == "lisbon_weekends", )

M_5 <- lm(realSum ~ . - realSum - X - city_day, data = temp_data)
temp_data <- subset(my_data_train, city_day == "london_weekdays" |
    city_day == "london_weekends", )

M_6 <- lm(realSum ~ . - realSum - X - city_day, data = temp_data)
temp_data <- subset(my_data_train, city_day == "paris_weekdays" |
    city_day == "paris_weekends", )

M_7 <- lm(realSum ~ . - realSum - X - city_day, data = temp_data)
temp_data <- subset(my_data_train, city_day == "rome_weekdays" |
    city_day == "rome_weekends", )

M_8 <- lm(realSum ~ . - realSum - X - city_day, data = temp_data)
temp_data <- subset(my_data_train, city_day == "vienna_weekdays" |
    city_day == "vienna_weekends", )

M_9 <- lm(realSum ~ . - realSum - X - city_day, data = temp_data)
```

```r
coefs <- tidy(M_0)
coefs[order(coefs$estimate, decreasing = TRUE), ]
coefs <- tidy(M_1)
coefs[order(coefs$estimate, decreasing = TRUE), ]
coefs <- tidy(M_2)
coefs[order(coefs$estimate, decreasing = TRUE), ]
coefs <- tidy(M_3)
coefs[order(coefs$estimate, decreasing = TRUE), ]
coefs <- tidy(M_4)
coefs[order(coefs$estimate, decreasing = TRUE), ]
coefs <- tidy(M_5)
coefs[order(coefs$estimate, decreasing = TRUE), ]
coefs <- tidy(M_6)
coefs[order(coefs$estimate, decreasing = TRUE), ]
coefs <- tidy(M_7)
coefs[order(coefs$estimate, decreasing = TRUE), ]
coefs <- tidy(M_8)
coefs[order(coefs$estimate, decreasing = TRUE), ]
coefs <- tidy(M_9)
```

```r
coefs[order(coefs$estimate, decreasing = TRUE), ]
```

MLR

```r
M1 <- lm(realSum ~ . - realSum - X, data = my_data_train)
```

```
summary(M1)
```

```
##
## Call:
## lm(formula = realSum ~ . - realSum - X, data = my_data_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
##  -758.3   -84.0   -21.0    42.9 18422.4
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                -4954.9076  3996.1824  -1.240 0.215017
## room_typePrivate room       -114.3655     4.2823 -26.707  < 2e-16 ***
## room_typeShared room        -204.1842    18.9348 -10.784  < 2e-16 ***
## person_capacity               23.9626     1.7645  13.581  < 2e-16 ***
## host_is_superhostTrue          1.0749     3.9344   0.273 0.784700
## multi                          9.6004     4.1324   2.323 0.020173 *
## biz                           33.2806     4.1885   7.946 1.99e-15 ***
## cleanliness_rating             5.0383     2.4153   2.086 0.036987 *
## guest_satisfaction_overall     0.7760     0.2615   2.968 0.002999 **
## bedrooms                      86.0154     3.1888  26.974  < 2e-16 ***
## dist                          -1.5330     1.2628  -1.214 0.224761
## metro_dist                    -3.9967     2.5025  -1.597 0.110262
## attr_index_norm                6.3705     0.2946  21.627  < 2e-16 ***
## rest_index_norm               -0.1837     0.1774  -1.036 0.300215
## lng                         -262.8909    40.1931  -6.541 6.20e-11 ***
## lat                          123.2117    76.5228   1.610 0.107378
## city_dayamsterdam_weekends    67.9410    16.0017   4.246 2.18e-05 ***
## city_dayathens_weekdays     6315.0906  1388.5351   4.548 5.43e-06 ***
## city_dayathens_weekends     6303.5311  1388.6527   4.539 5.66e-06 ***
## city_daybarcelona_weekdays   411.7717   837.7909   0.491 0.623078
## city_daybarcelona_weekends   429.6529   837.8109   0.513 0.608075
## city_dayberlin_weekdays     1949.0424   342.1245   5.697 1.23e-08 ***
## city_dayberlin_weekends     1958.8844   342.0401   5.727 1.03e-08 ***
## city_daybudapest_weekdays   3902.3511   706.8806   5.521 3.40e-08 ***
## city_daybudapest_weekends   3929.6734   706.8440   5.559 2.73e-08 ***
## city_daylisbon_weekdays    -2312.9309  1143.8977  -2.022 0.043186 *
## city_daylisbon_weekends    -2304.0067  1143.8126  -2.014 0.043983 *
## city_daylondon_weekdays    -1409.2997   206.1046  -6.838 8.17e-12 ***
## city_daylondon_weekends    -1410.9328   206.1223  -6.845 7.76e-12 ***
## city_dayparis_weekdays      -403.0289   278.4819  -1.447 0.147840
## city_dayparis_weekends      -422.1389   278.6437  -1.515 0.129787
## city_dayrome_weekdays       2947.6852   881.3984   3.344 0.000826 ***
## city_dayrome_weekends       2952.5352   881.4297   3.350 0.000810 ***
## city_dayvienna_weekdays     3231.8185   582.7092   5.546 2.94e-08 ***
## city_dayvienna_weekends     3230.2680   582.7972   5.543 3.00e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 305.1 on 36159 degrees of freedom
## Multiple R-squared:  0.215,  Adjusted R-squared:  0.2143
## F-statistic: 291.3 on 34 and 36159 DF,  p-value: < 2.2e-16
```

```r
# Create summary table with coefficients and p-values
table <- summary(M1)$coefficients[, c(1, 4)]
```

```r
# Calculate R-squared and multiple R-squared
y_train_pred <- predict(M1, my_data_train)
y_train_mean <- mean(my_data_train$realSum)
SST <- sum((my_data_train$realSum - y_train_mean)^2)
SSR <- sum((my_data_train$realSum - y_train_pred)^2)
R_squared <- 1 - SSR/SST
n <- length(my_data_train$realSum)
p <- ncol(my_data_train)
adj_R_squared <- 1 - (SSR/(n - p - 1))/(SST/(n - 1))
RMSE = sqrt(mean((my_data_train$realSum - y_train_pred)^2))


# Print the R-squared and multiple R-squared values
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.2150414
```

```r
cat("Adjusted R-squared:", adj_R_squared, "\n")
```

```
## Adjusted R-squared: 0.2146725
```

```r
cat("RMSE:", RMSE, "\n")
```

```
## RMSE: 304.9175
```

The r^2 and adjusted r^2 values are too low for the Linear regression model to be considered a competent one in this case.

```r
# Calculate R-squared and multiple R-squared
y_train_pred <- predict(M1, my_data_test)
y_train_mean <- mean(my_data_test$realSum)
SST <- sum((my_data_test$realSum - y_train_mean)^2)
SSR <- sum((my_data_test$realSum - y_train_pred)^2)
R_squared <- 1 - SSR/SST
n <- length(my_data_test$realSum)
p <- ncol(my_data_test)
adj_R_squared <- 1 - (SSR/(n - p - 1))/(SST/(n - 1))
RMSE = sqrt(mean((my_data_test$realSum - y_train_pred)^2))


# Print the R-squared and multiple R-squared values
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.33744
```

```r
cat("Adjusted R-squared:", adj_R_squared, "\n")
```

```
## Adjusted R-squared: 0.3367131
```

```r
cat("RMSE:", RMSE, "\n")
```

```
## RMSE: 233.2618
```

```r
M1_step = step(M1, direction = "backward")
```

```r
# Calculate R-squared and multiple R-squared
y_train_pred <- predict(M1_step, my_data_train)
y_train_mean <- mean(my_data_train$realSum)
SST <- sum((my_data_train$realSum - y_train_mean)^2)
SSR <- sum((my_data_train$realSum - y_train_pred)^2)
R_squared <- 1 - SSR/SST
n <- length(my_data_train$realSum)
p <- ncol(my_data_train)
adj_R_squared <- 1 - (SSR/(n - p - 1))/(SST/(n - 1))
RMSE = sqrt(mean((my_data_train$realSum - y_train_pred)^2))


# Print the R-squared and multiple R-squared values
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.2149964
```

```r
cat("Adjusted R-squared:", adj_R_squared, "\n")
```

```
## Adjusted R-squared: 0.2146275
```

```r
cat("RMSE:", RMSE, "\n")
```

```
## RMSE: 304.9262
```

```r
# Calculate R-squared and multiple R-squared
y_train_pred <- predict(M1_step, my_data_test)
y_train_mean <- mean(my_data_test$realSum)
SST <- sum((my_data_test$realSum - y_train_mean)^2)
SSR <- sum((my_data_test$realSum - y_train_pred)^2)
R_squared <- 1 - SSR/SST
n <- length(my_data_test$realSum)
p <- ncol(my_data_test)
adj_R_squared <- 1 - (SSR/(n - p - 1))/(SST/(n - 1))
RMSE = sqrt(mean((my_data_test$realSum - y_train_pred)^2))


# Print the R-squared and multiple R-squared values
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.3373441
```

```r
cat("Adjusted R-squared:", adj_R_squared, "\n")
```

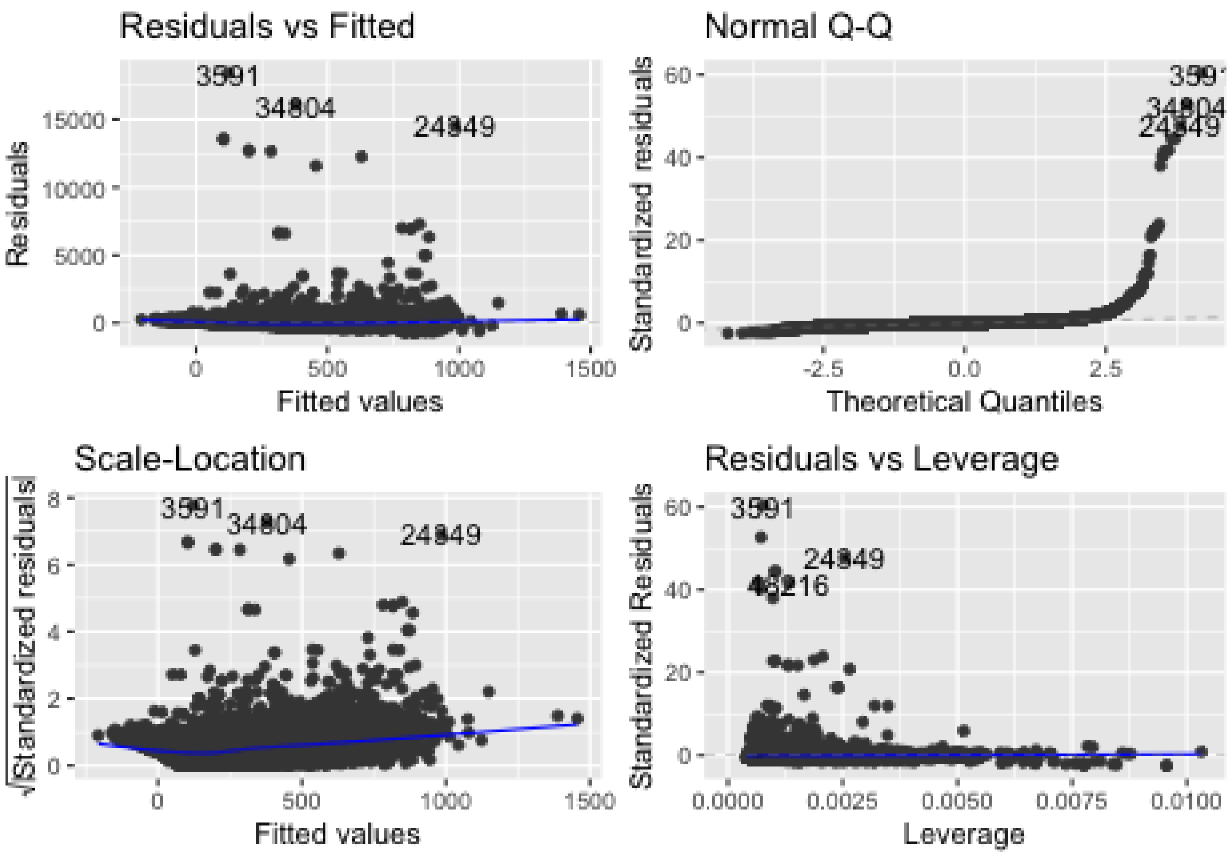
```
## Adjusted R-squared: 0.336617
```

```r
cat("RMSE:", RMSE, "\n")
```

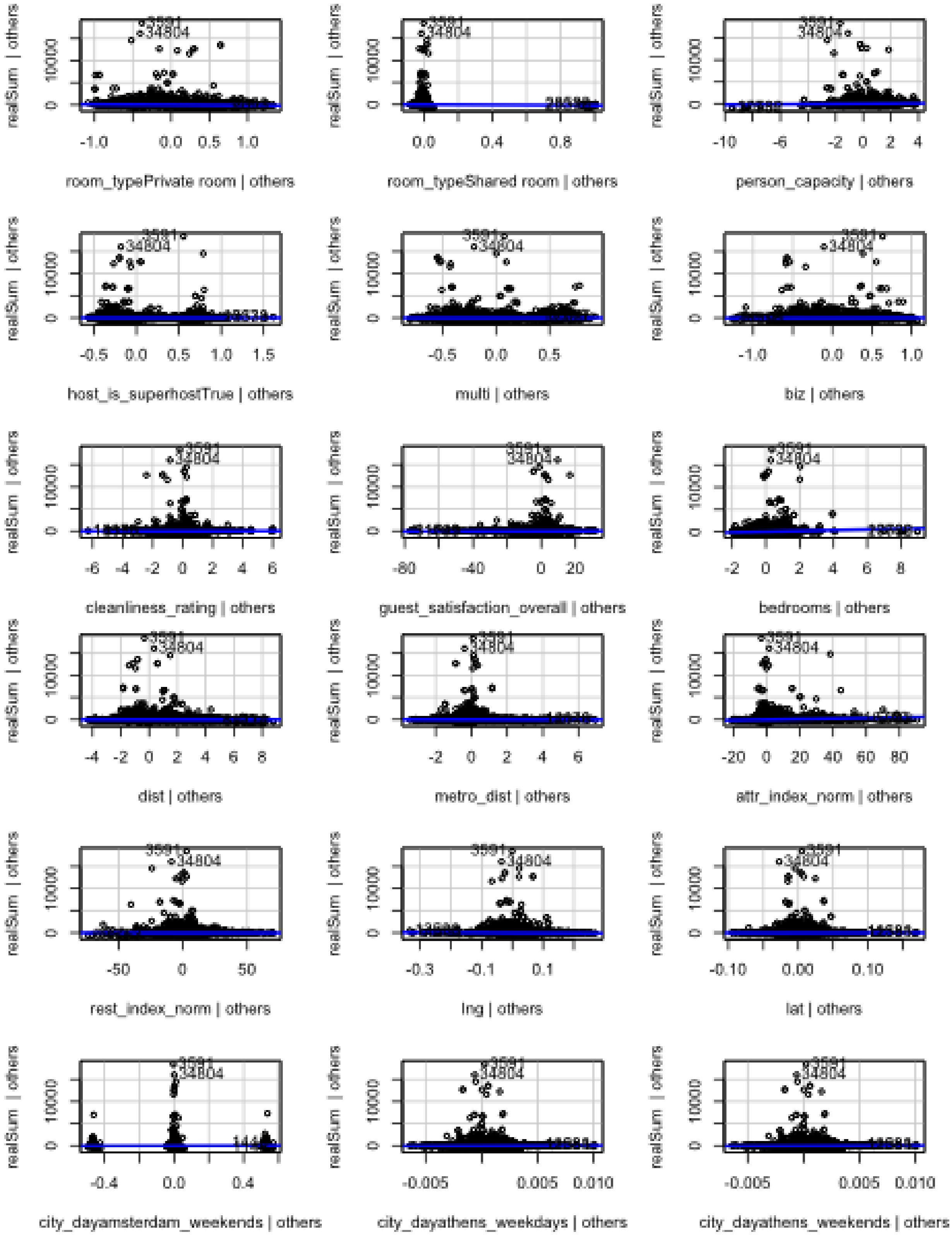
```
## RMSE: 233.2787
```

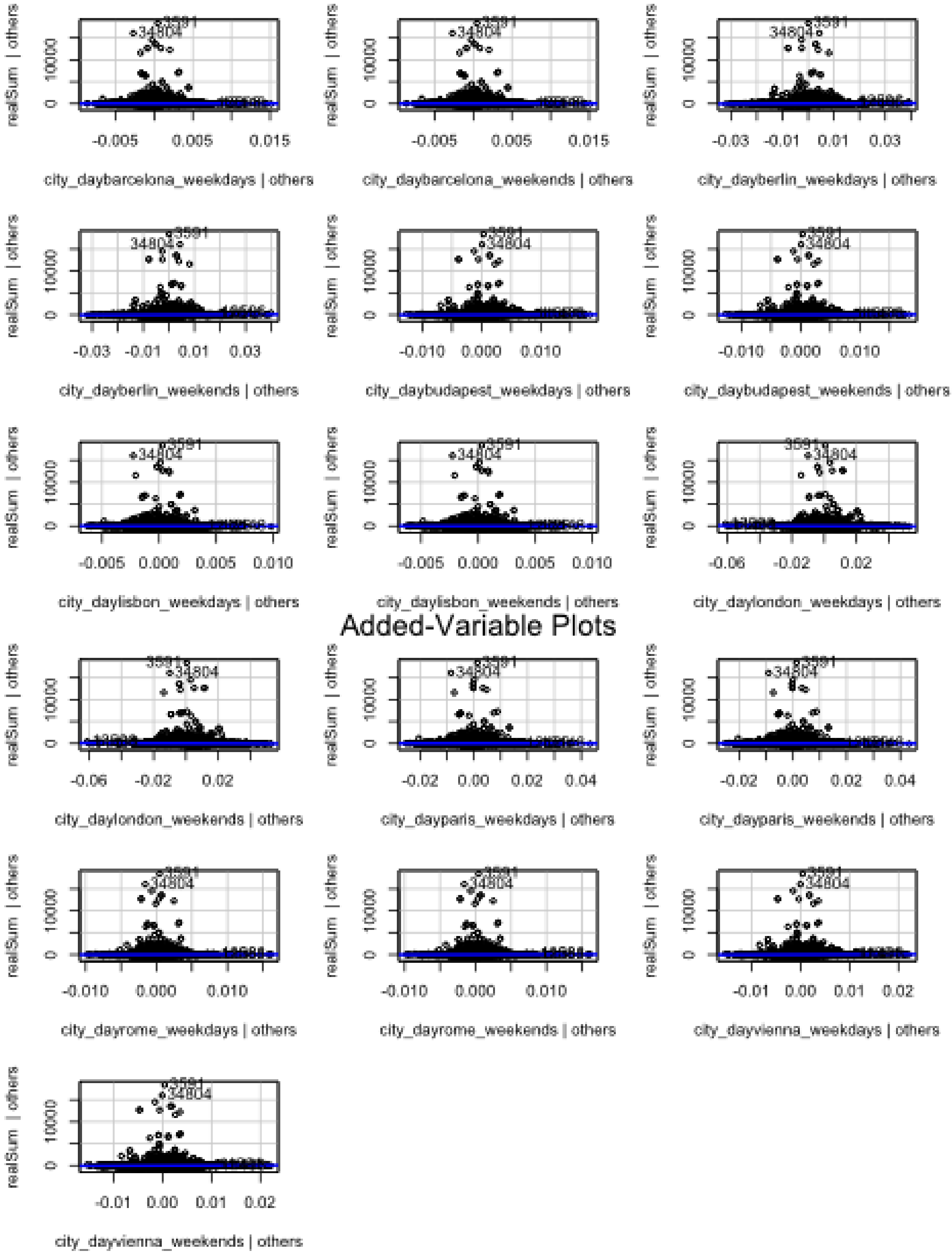
M1_step Diagnostics

```r
library(car)
autoplot(M1)
```




```r
avPlots(M1)
```


realSum | others
room_typePrivate room | others
room_typeShared room | others
person_capacity | others
host_is_superhostTrue | others
multi | others
biz | others
cleanliness_rating | others
guest_satisfaction_overall | others
bedrooms | others
dist | others
metro_dist | others
attr_index_norm | others
rest_index_norm | others
lng | others
lat | others
city_dayamsterdam_weekends | others
city_dayathens_weekdays | others
city_dayathens_weekends | others
34804


Added-Variable Plots
realSum | others
city_daybarcelona_weekdays | others
city_daybarcelona_weekends | others
city_dayberlin_weekdays | others
city_dayberlin_weekends | others
city_daybudapest_weekdays | others
city_daybudapest_weekends | others
city_daylisbon_weekdays | others
city_daylisbon_weekends | others
city_daylondon_weekdays | others
city_daylondon_weekends | others
city_dayparis_weekdays | others
city_dayparis_weekends | others
city_dayrome_weekdays | others
city_dayrome_weekends | others
city_dayvienna_weekdays | others
city_dayvienna_weekends | others

```r
my_data_train[34804, ]
```

```
##           X  realSum       room_type person_capacity host_is_superhost multi biz
## 49907 1736 637.6364 Entire home/apt               2             False     0   0
##       cleanliness_rating guest_satisfaction_overall bedrooms      dist
## 49907                 10                         93        1 0.9940386
##       metro_dist attr_index_norm rest_index_norm      lng     lat
## 49907  0.2025371        12.10842        6.748565 16.38568 48.2046
##              city_day
## 49907 vienna_weekdays
```

```r
my_data_train[3591, ]
```

```
##        X  realSum       room_type person_capacity host_is_superhost multi biz
## 4917 183 156.3049 Entire home/apt               6             False     0   1
##      cleanliness_rating guest_satisfaction_overall bedrooms     dist metro_dist
## 4917                  9                         94        2 2.841432 0.04608507
##      attr_index_norm rest_index_norm      lng      lat        city_day
## 4917        2.366625         1.39056 23.72258 37.99906 athens_weekends
```

```r
my_data_train[24349, ]
```

```
##           X realSum    room_type person_capacity host_is_superhost multi biz
## 21461 1904 108.818 Private room               2             False     0   1
##       cleanliness_rating guest_satisfaction_overall bedrooms     dist
## 21461                  8                         80        1 3.216239
##       metro_dist attr_index_norm rest_index_norm      lng      lat
## 21461  0.3290175        3.115481        14.47335 -9.14292 38.74124
##              city_day
## 21461 lisbon_weekends
```

MLR with IVs

```r
M1IV <- lm(realSum ~ room_type + host_is_superhost + multi +
    biz + city_day + person_capacity + cleanliness_rating + guest_satisfaction_overall +
    bedrooms + dist + metro_dist + attr_index_norm + rest_index_norm +
    lng + lat + metro_dist:dist + attr_index_norm:dist + attr_index_norm:metro_dist +
    rest_index_norm:dist + rest_index_norm:metro_dist + rest_index_norm:attr_index_norm,
    data = my_data_train)
```

```r
# Calculate R-squared and multiple R-squared
y_train_pred <- predict(M1IV, my_data_train)
y_train_mean <- mean(my_data_train$realSum)
SST <- sum((my_data_train$realSum - y_train_mean)^2)
SSR <- sum((my_data_train$realSum - y_train_pred)^2)
R_squared <- 1 - SSR/SST
n <- length(my_data_train$realSum)
p <- ncol(my_data_train)
adj_R_squared <- 1 - (SSR/(n - p - 1))/(SST/(n - 1))
```

```r
RMSE = sqrt(mean((my_data_train$realSum - y_train_pred)^2))


# Print the R-squared and multiple R-squared values
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.2159815
```

```r
cat("Adjusted R-squared:", adj_R_squared, "\n")
```

```
## Adjusted R-squared: 0.215613
```

```r
cat("RMSE:", RMSE, "\n")
```

```
## RMSE: 304.7348
```

```r
# Calculate R-squared and multiple R-squared
y_train_pred <- predict(M1IV, my_data_test)
y_train_mean <- mean(my_data_test$realSum)
SST <- sum((my_data_test$realSum - y_train_mean)^2)
SSR <- sum((my_data_test$realSum - y_train_pred)^2)
R_squared <- 1 - SSR/SST
n <- length(my_data_test$realSum)
p <- ncol(my_data_test)
adj_R_squared <- 1 - (SSR/(n - p - 1))/(SST/(n - 1))
RMSE = sqrt(mean((my_data_test$realSum - y_train_pred)^2))


# Print the R-squared and multiple R-squared values
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.3379444
```

```r
cat("Adjusted R-squared:", adj_R_squared, "\n")
```

```
## Adjusted R-squared: 0.337218
```

```r
cat("RMSE:", RMSE, "\n")
```

```
## RMSE: 233.173
```

```r
M1stepIV = step(M1IV, direction = "backward")
```

```r
# Calculate R-squared and multiple R-squared
y_train_pred <- predict(M1stepIV, my_data_train)
y_train_mean <- mean(my_data_train$realSum)
SST <- sum((my_data_train$realSum - y_train_mean)^2)
SSR <- sum((my_data_train$realSum - y_train_pred)^2)
```

```r
R_squared <- 1 - SSR/SST
n <- length(my_data_train$realSum)
p <- ncol(my_data_train)
adj_R_squared <- 1 - (SSR/(n - p - 1))/(SST/(n - 1))
RMSE = sqrt(mean((my_data_train$realSum - y_train_pred)^2))


# Print the R-squared and multiple R-squared values
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.2159435
```

```r
cat("Adjusted R-squared:", adj_R_squared, "\n")
```

```
## Adjusted R-squared: 0.215575
```

```r
cat("RMSE:", RMSE, "\n")
```

```
## RMSE: 304.7422
```

```r
# Calculate R-squared and multiple R-squared
y_train_pred <- predict(M1stepIV, my_data_test)
y_train_mean <- mean(my_data_test$realSum)
SST <- sum((my_data_test$realSum - y_train_mean)^2)
SSR <- sum((my_data_test$realSum - y_train_pred)^2)
R_squared <- 1 - SSR/SST
n <- length(my_data_test$realSum)
p <- ncol(my_data_test)
adj_R_squared <- 1 - (SSR/(n - p - 1))/(SST/(n - 1))
RMSE = sqrt(mean((my_data_test$realSum - y_train_pred)^2))


# Print the R-squared and multiple R-squared values
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.3380392
```

```r
cat("Adjusted R-squared:", adj_R_squared, "\n")
```

```
## Adjusted R-squared: 0.3373129
```

```r
cat("RMSE:", RMSE, "\n")
```

```
## RMSE: 233.1563
```

Second Order Polynomial

```r
poly2 <- lm(realSum ~ room_type + host_is_superhost + multi +
    biz + city_day + person_capacity + cleanliness_rating + guest_satisfaction_overall +
    bedrooms + poly(dist, 2) + poly(metro_dist, 2) + poly(attr_index_norm,
    2) + poly(rest_index_norm, 2) + lng + lat, data = my_data_train)
```

```r
# Calculate R-squared and multiple R-squared
y_train_pred <- predict(poly2, my_data_train)
y_train_mean <- mean(my_data_train$realSum)
SST <- sum((my_data_train$realSum - y_train_mean)^2)
SSR <- sum((my_data_train$realSum - y_train_pred)^2)
R_squared <- 1 - SSR/SST
n <- length(my_data_train$realSum)
p <- ncol(my_data_train)
adj_R_squared <- 1 - (SSR/(n - p - 1))/(SST/(n - 1))
RMSE = sqrt(mean((my_data_train$realSum - y_train_pred)^2))


# Print the R-squared and multiple R-squared values
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.2154699
```

```r
cat("Adjusted R-squared:", adj_R_squared, "\n")
```

```
## Adjusted R-squared: 0.2151012
```

```r
cat("RMSE:", RMSE, "\n")
```

```
## RMSE: 304.8342
```

```r
# Calculate R-squared and multiple R-squared
y_train_pred <- predict(poly2, my_data_test)
y_train_mean <- mean(my_data_test$realSum)
SST <- sum((my_data_test$realSum - y_train_mean)^2)
SSR <- sum((my_data_test$realSum - y_train_pred)^2)
R_squared <- 1 - SSR/SST
n <- length(my_data_test$realSum)
p <- ncol(my_data_test)
adj_R_squared <- 1 - (SSR/(n - p - 1))/(SST/(n - 1))
RMSE = sqrt(mean((my_data_test$realSum - y_train_pred)^2))


# Print the R-squared and multiple R-squared values
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.3373538
```

```r
cat("Adjusted R-squared:", adj_R_squared, "\n")
```

```
## Adjusted R-squared: 0.3366268
```

```r
cat("RMSE:", RMSE, "\n")
```

```
## RMSE: 233.277
```

```r
poly2step = step(poly2, direction = "backward")
```

```r
# Calculate R-squared and multiple R-squared
y_train_pred <- predict(poly2step, my_data_train)
y_train_mean <- mean(my_data_train$realSum)
SST <- sum((my_data_train$realSum - y_train_mean)^2)
SSR <- sum((my_data_train$realSum - y_train_pred)^2)
R_squared <- 1 - SSR/SST
n <- length(my_data_train$realSum)
p <- ncol(my_data_train)
adj_R_squared <- 1 - (SSR/(n - p - 1))/(SST/(n - 1))
RMSE = sqrt(mean((my_data_train$realSum - y_train_pred)^2))


# Print the R-squared and multiple R-squared values
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.2154289
```

```r
cat("Adjusted R-squared:", adj_R_squared, "\n")
```

```
## Adjusted R-squared: 0.2150602
```

```r
cat("RMSE:", RMSE, "\n")
```

```
## RMSE: 304.8422
```

```r
# Calculate R-squared and multiple R-squared
y_train_pred <- predict(poly2step, my_data_test)
y_train_mean <- mean(my_data_test$realSum)
SST <- sum((my_data_test$realSum - y_train_mean)^2)
SSR <- sum((my_data_test$realSum - y_train_pred)^2)
R_squared <- 1 - SSR/SST
n <- length(my_data_test$realSum)
p <- ncol(my_data_test)
adj_R_squared <- 1 - (SSR/(n - p - 1))/(SST/(n - 1))
RMSE = sqrt(mean((my_data_test$realSum - y_train_pred)^2))


# Print the R-squared and multiple R-squared values
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.3373204
```

```r
cat("Adjusted R-squared:", adj_R_squared, "\n")
```

```
## Adjusted R-squared: 0.3365933
```

```r
cat("RMSE:", RMSE, "\n")
```

```
## RMSE: 233.2829
```

Second Order Polynomial with IVs

```r
poly2IV <- lm(realSum ~ room_type + host_is_superhost + multi +
    biz + city_day + person_capacity + cleanliness_rating + guest_satisfaction_overall +
    bedrooms + poly(dist, 2) * poly(metro_dist, 2) * poly(attr_index_norm,
    2) * poly(rest_index_norm, 2) + lng + lat, data = my_data_train)
```

```r
# Calculate R-squared and multiple R-squared
y_train_pred <- predict(poly2IV, my_data_train)
y_train_mean <- mean(my_data_train$realSum)
SST <- sum((my_data_train$realSum - y_train_mean)^2)
SSR <- sum((my_data_train$realSum - y_train_pred)^2)
R_squared <- 1 - SSR/SST
n <- length(my_data_train$realSum)
p <- ncol(my_data_train)
adj_R_squared <- 1 - (SSR/(n - p - 1))/(SST/(n - 1))
RMSE = sqrt(mean((my_data_train$realSum - y_train_pred)^2))


# Print the R-squared and multiple R-squared values
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.22214
```

```r
cat("Adjusted R-squared:", adj_R_squared, "\n")
```

```
## Adjusted R-squared: 0.2217744
```

```r
cat("RMSE:", RMSE, "\n")
```

```
## RMSE: 303.5356
```

```r
# Calculate R-squared and multiple R-squared
y_train_pred <- predict(poly2IV, my_data_test)
y_train_mean <- mean(my_data_test$realSum)
SST <- sum((my_data_test$realSum - y_train_mean)^2)
SSR <- sum((my_data_test$realSum - y_train_pred)^2)
R_squared <- 1 - SSR/SST
n <- length(my_data_test$realSum)
p <- ncol(my_data_test)
```

```r
adj_R_squared <- 1 - (SSR/(n - p - 1))/(SST/(n - 1))
RMSE = sqrt(mean((my_data_test$realSum - y_train_pred)^2))


# Print the R-squared and multiple R-squared values
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.334993
```

```r
cat("Adjusted R-squared:", adj_R_squared, "\n")
```

```
## Adjusted R-squared: 0.3342634
```

```r
cat("RMSE:", RMSE, "\n")
```

```
## RMSE: 233.6922
```

```r
poly2stepIV = step(poly2IV, direction = "backward")
```

```r
# Calculate R-squared and multiple R-squared
y_train_pred <- predict(poly2stepIV, my_data_train)
y_train_mean <- mean(my_data_train$realSum)
SST <- sum((my_data_train$realSum - y_train_mean)^2)
SSR <- sum((my_data_train$realSum - y_train_pred)^2)
R_squared <- 1 - SSR/SST
n <- length(my_data_train$realSum)
p <- ncol(my_data_train)
adj_R_squared <- 1 - (SSR/(n - p - 1))/(SST/(n - 1))
RMSE = sqrt(mean((my_data_train$realSum - y_train_pred)^2))


# Print the R-squared and multiple R-squared values
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.2221344
```

```r
cat("Adjusted R-squared:", adj_R_squared, "\n")
```

```
## Adjusted R-squared: 0.2217689
```

```r
cat("RMSE:", RMSE, "\n")
```

```
## RMSE: 303.5367
```

```r
# Calculate R-squared and multiple R-squared
y_train_pred <- predict(poly2stepIV, my_data_test)
y_train_mean <- mean(my_data_test$realSum)
SST <- sum((my_data_test$realSum - y_train_mean)^2)
```

```r
SSR <- sum((my_data_test$realSum - y_train_pred)^2)
R_squared <- 1 - SSR/SST
n <- length(my_data_test$realSum)
p <- ncol(my_data_test)
adj_R_squared <- 1 - (SSR/(n - p - 1))/(SST/(n - 1))
RMSE = sqrt(mean((my_data_test$realSum - y_train_pred)^2))


# Print the R-squared and multiple R-squared values
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.3350694
```

```r
cat("Adjusted R-squared:", adj_R_squared, "\n")
```

```
## Adjusted R-squared: 0.3343399
```

```r
cat("RMSE:", RMSE, "\n")
```

```
## RMSE: 233.6788
```

Third Order Polynomial

```r
poly3 <- lm(realSum ~ room_type + host_is_superhost + multi +
    biz + city_day + person_capacity + cleanliness_rating + guest_satisfaction_overall +
    bedrooms + poly(dist, 3) + poly(metro_dist, 3) + poly(attr_index_norm,
    3) + poly(rest_index_norm, 3) + lng + lat, data = my_data_train)
```

```r
# Calculate R-squared and multiple R-squared
y_train_pred <- predict(poly3, my_data_train)
y_train_mean <- mean(my_data_train$realSum)
SST <- sum((my_data_train$realSum - y_train_mean)^2)
SSR <- sum((my_data_train$realSum - y_train_pred)^2)
R_squared <- 1 - SSR/SST
n <- length(my_data_train$realSum)
p <- ncol(my_data_train)
adj_R_squared <- 1 - (SSR/(n - p - 1))/(SST/(n - 1))
RMSE = sqrt(mean((my_data_train$realSum - y_train_pred)^2))


# Print the R-squared and multiple R-squared values
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.2160624
```

```r
cat("Adjusted R-squared:", adj_R_squared, "\n")
```

```
## Adjusted R-squared: 0.215694
```

```
cat("RMSE:", RMSE, "\n")
```

```
## RMSE: 304.7191
```

```
# Calculate R-squared and multiple R-squared
y_train_pred <- predict(poly3, my_data_test)
y_train_mean <- mean(my_data_test$realSum)
SST <- sum((my_data_test$realSum - y_train_mean)^2)
SSR <- sum((my_data_test$realSum - y_train_pred)^2)
R_squared <- 1 - SSR/SST
n <- length(my_data_test$realSum)
p <- ncol(my_data_test)
adj_R_squared <- 1 - (SSR/(n - p - 1))/(SST/(n - 1))
RMSE = sqrt(mean((my_data_test$realSum - y_train_pred)^2))


# Print the R-squared and multiple R-squared values
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.3375855
```

```
cat("Adjusted R-squared:", adj_R_squared, "\n")
```

```
## Adjusted R-squared: 0.3368588
```

```
cat("RMSE:", RMSE, "\n")
```

```
## RMSE: 233.2362
```

```
poly3step = step(poly3, direction = "backward")
```

```
# Calculate R-squared and multiple R-squared
y_train_pred <- predict(poly3step, my_data_train)
y_train_mean <- mean(my_data_train$realSum)
SST <- sum((my_data_train$realSum - y_train_mean)^2)
SSR <- sum((my_data_train$realSum - y_train_pred)^2)
R_squared <- 1 - SSR/SST
n <- length(my_data_train$realSum)
p <- ncol(my_data_train)
adj_R_squared <- 1 - (SSR/(n - p - 1))/(SST/(n - 1))
RMSE = sqrt(mean((my_data_train$realSum - y_train_pred)^2))


# Print the R-squared and multiple R-squared values
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.2160337
```

```
cat("Adjusted R-squared:", adj_R_squared, "\n")
```

```
## Adjusted R-squared: 0.2156653
```

```
cat("RMSE:", RMSE, "\n")
```

```
## RMSE: 304.7247
```

```
# Calculate R-squared and multiple R-squared
y_train_pred <- predict(poly3step, my_data_test)
y_train_mean <- mean(my_data_test$realSum)
SST <- sum((my_data_test$realSum - y_train_mean)^2)
SSR <- sum((my_data_test$realSum - y_train_pred)^2)
R_squared <- 1 - SSR/SST
n <- length(my_data_test$realSum)
p <- ncol(my_data_test)
adj_R_squared <- 1 - (SSR/(n - p - 1))/(SST/(n - 1))
RMSE = sqrt(mean((my_data_test$realSum - y_train_pred)^2))


# Print the R-squared and multiple R-squared values
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.3376519
```

```
cat("Adjusted R-squared:", adj_R_squared, "\n")
```

```
## Adjusted R-squared: 0.3369253
```

```
cat("RMSE:", RMSE, "\n")
```

```
## RMSE: 233.2245
```

Third Order Polynomial with IVs

```
poly3IV <- lm(realSum ~ room_type + host_is_superhost + multi +
    biz + city_day + person_capacity + cleanliness_rating + guest_satisfaction_overall +
    bedrooms + poly(dist, 3) * poly(metro_dist, 3) * poly(attr_index_norm,
    3) * poly(rest_index_norm, 3) + lng + lat, data = my_data_train)
```

```
# Calculate R-squared and multiple R-squared
y_train_pred <- predict(poly3IV, my_data_train)
y_train_mean <- mean(my_data_train$realSum)
SST <- sum((my_data_train$realSum - y_train_mean)^2)
SSR <- sum((my_data_train$realSum - y_train_pred)^2)
R_squared <- 1 - SSR/SST
n <- length(my_data_train$realSum)
p <- ncol(my_data_train)
```

```r
adj_R_squared <- 1 - (SSR/(n - p - 1))/(SST/(n - 1))
RMSE = sqrt(mean((my_data_train$realSum - y_train_pred)^2))


# Print the R-squared and multiple R-squared values
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.2330663
```

```r
cat("Adjusted R-squared:", adj_R_squared, "\n")
```

```
## Adjusted R-squared: 0.2327059
```

```r
cat("RMSE:", RMSE, "\n")
```

```
## RMSE: 301.3962
```

```r
# Calculate R-squared and multiple R-squared
y_train_pred <- predict(poly3IV, my_data_test)
y_train_mean <- mean(my_data_test$realSum)
SST <- sum((my_data_test$realSum - y_train_mean)^2)
SSR <- sum((my_data_test$realSum - y_train_pred)^2)
R_squared <- 1 - SSR/SST
n <- length(my_data_test$realSum)
p <- ncol(my_data_test)
adj_R_squared <- 1 - (SSR/(n - p - 1))/(SST/(n - 1))
RMSE = sqrt(mean((my_data_test$realSum - y_train_pred)^2))


# Print the R-squared and multiple R-squared values
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.1901115
```

```r
cat("Adjusted R-squared:", adj_R_squared, "\n")
```

```
## Adjusted R-squared: 0.189223
```

```r
cat("RMSE:", RMSE, "\n")
```

```
## RMSE: 257.8955
```

```r
poly3stepIV = step(poly3IV, direction = "backward")
```

```r
# Calculate R-squared and multiple R-squared
y_train_pred <- predict(poly3stepIV, my_data_train)
y_train_mean <- mean(my_data_train$realSum)
SST <- sum((my_data_train$realSum - y_train_mean)^2)
```

```r
SSR <- sum((my_data_train$realSum - y_train_pred)^2)
R_squared <- 1 - SSR/SST
n <- length(my_data_train$realSum)
p <- ncol(my_data_train)
adj_R_squared <- 1 - (SSR/(n - p - 1))/(SST/(n - 1))
RMSE = sqrt(mean((my_data_train$realSum - y_train_pred)^2))


# Print the R-squared and multiple R-squared values
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.2285384
```

```r
cat("Adjusted R-squared:", adj_R_squared, "\n")
```

```
## Adjusted R-squared: 0.2281759
```

```r
cat("RMSE:", RMSE, "\n")
```

```
## RMSE: 302.2846
```

```r
# Calculate R-squared and multiple R-squared
y_train_pred <- predict(poly3stepIV, my_data_test)
y_train_mean <- mean(my_data_test$realSum)
SST <- sum((my_data_test$realSum - y_train_mean)^2)
SSR <- sum((my_data_test$realSum - y_train_pred)^2)
R_squared <- 1 - SSR/SST
n <- length(my_data_test$realSum)
p <- ncol(my_data_test)
adj_R_squared <- 1 - (SSR/(n - p - 1))/(SST/(n - 1))
RMSE = sqrt(mean((my_data_test$realSum - y_train_pred)^2))


# Print the R-squared and multiple R-squared values
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.3281442
```

```r
cat("Adjusted R-squared:", adj_R_squared, "\n")
```

```
## Adjusted R-squared: 0.327407
```

```r
cat("RMSE:", RMSE, "\n")
```

```
## RMSE: 234.8925
```

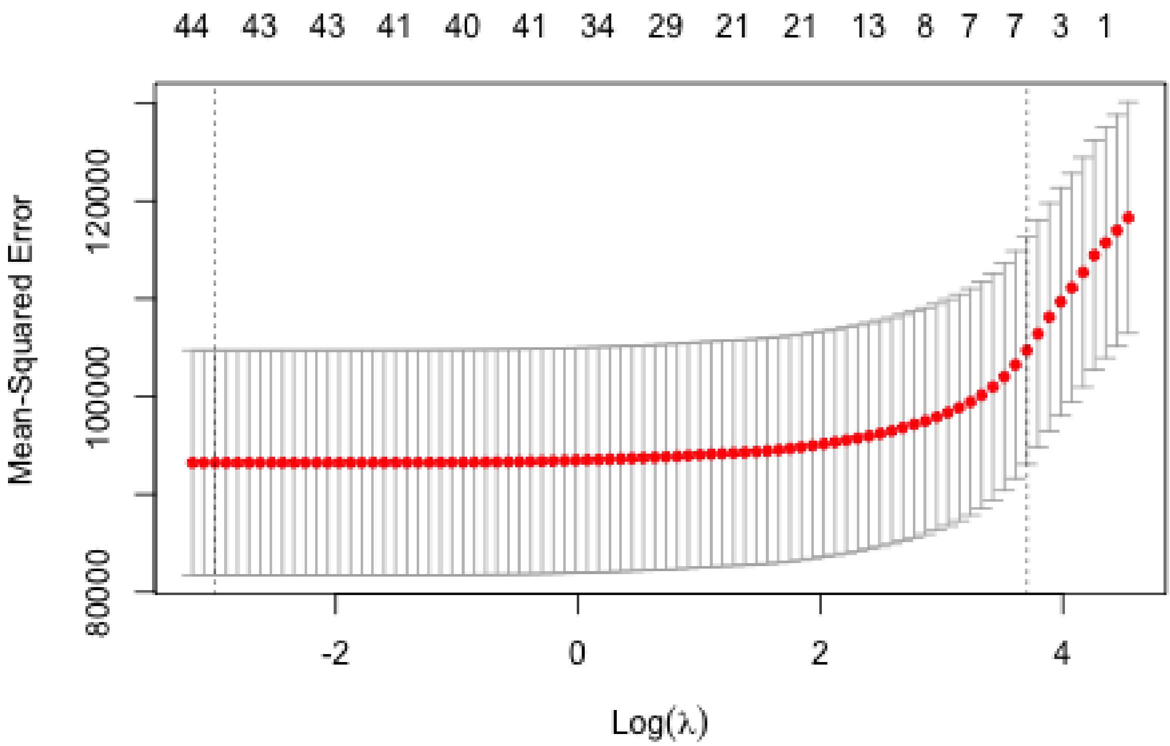
Lasso Regression

```r
# Prepare the predictors and response variable
x_train <- model.matrix(realSum ~ room_type + person_capacity +
    host_is_superhost + multi + biz + cleanliness_rating + guest_satisfaction_overall +
    bedrooms + dist * metro_dist * attr_index_norm * rest_index_norm +
    lng + lat + city_day, data = my_data_train)[, -1]
y_train <- my_data_train$realSum
y_test <- my_data_test$realSum

# Fit a Lasso regression model
lasso_model <- glmnet(x_train, y_train, alpha = 1)

# Select the best lambda value using cross-validation
cv_model <- cv.glmnet(x_train, y_train, alpha = 1, nfolds = 5)

# Plot the cross-validation results
plot(cv_model)
```




```r
# Select the lambda value that minimizes the mean
# cross-validation error
best_lambda <- cv_model$lambda.min

# Fit a Lasso regression model with the selected lambda
# value
lasso_model_best <- glmnet(x_train, y_train, alpha = 1, lambda = best_lambda)

# Calculate R-squared and multiple R-squared
y_train_pred <- predict(lasso_model_best, newx = x_train)
y_train_mean <- mean(y_train)
SST <- sum((y_train - y_train_mean)^2)
```

```r
SSR <- sum((y_train - y_train_pred)^2)
R_squared <- 1 - SSR/SST
multiple_R_squared <- cor(y_train_pred, y_train)^2
n <- length(y_train)
p <- ncol(x_train)
adj_R_squared <- 1 - (SSR/(n - p - 1))/(SST/(n - 1))


# Print the R-squared and multiple R-squared values
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.2157005
```

```r
cat("Adjusted R-squared:", adj_R_squared, "\n")
```

```
## Adjusted R-squared: 0.2147241
```

```r
# Evaluate the model performance
rmse <- sqrt(mean((my_data_test$realSum - y_train_pred)^2))
cat("RMSE on train set:", rmse, "\n")
```

```
## RMSE on train set: 325.8081
```

```r
x_test <- model.matrix(realSum ~ room_type + person_capacity +
    host_is_superhost + multi + biz + cleanliness_rating + guest_satisfaction_overall +
    bedrooms + dist * metro_dist * attr_index_norm * rest_index_norm +
    lng + lat + city_day, data = my_data_test)[, -1]

# Calculate R-squared and multiple R-squared
y_test_pred <- predict(lasso_model_best, newx = x_test)
y_test_mean <- mean(y_test)
SST <- sum((y_test - y_test_mean)^2)
SSR <- sum((y_test - y_test_pred)^2)
R_squared <- 1 - SSR/SST
n <- length(y_test)
p <- ncol(x_test)
adj_R_squared <- 1 - (SSR/(n - p - 1))/(SST/(n - 1))


# Print the R-squared and multiple R-squared values
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.3372477
```

```r
cat("Adjusted R-squared:", adj_R_squared, "\n")
```

```
## Adjusted R-squared: 0.3353195
```

```r
# Evaluate the model performance
rmse <- sqrt(mean((my_data_test$realSum - y_test_pred)^2))
cat("RMSE on test set:", rmse, "\n")
```

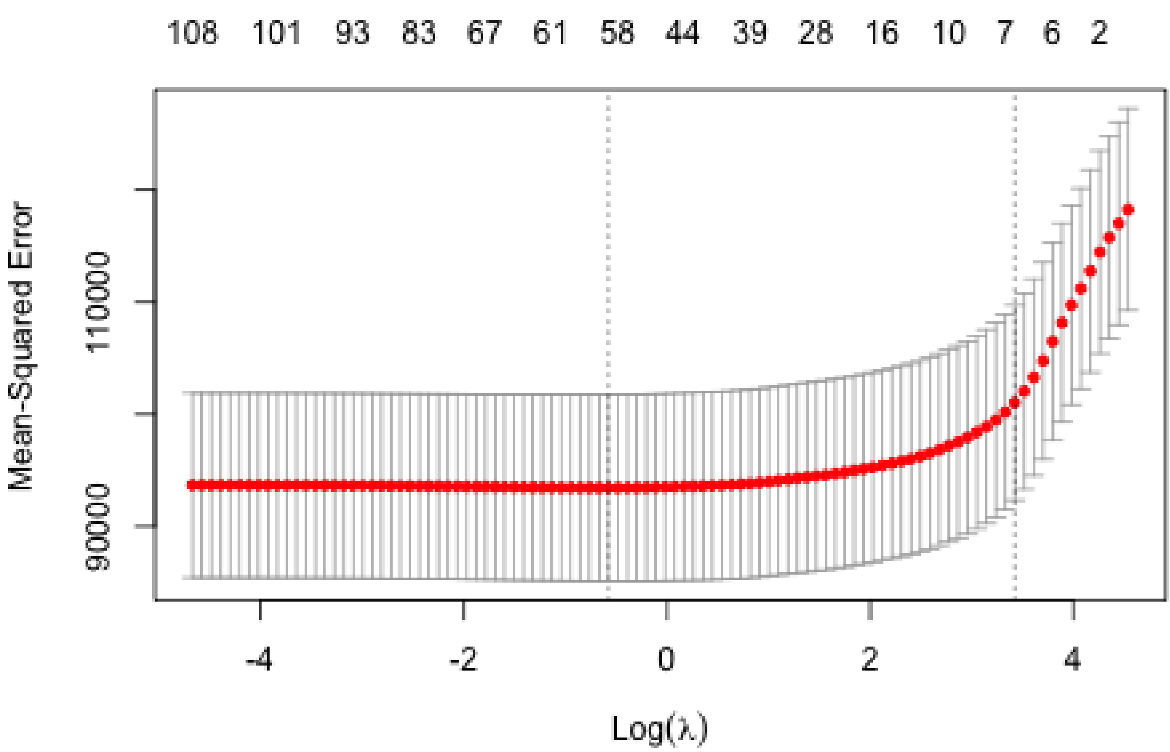
```
## RMSE on test set: 233.2957
```

```r
# Prepare the predictors and response variable
x_train <- model.matrix(realSum ~ room_type + person_capacity +
    host_is_superhost + multi + biz + cleanliness_rating + guest_satisfaction_overall +
    bedrooms + poly(dist, 2) * poly(metro_dist, 2) * poly(attr_index_norm,
    2) * poly(rest_index_norm, 2) + lng + lat + city_day, data = my_data_train)[,
    -1]
y_train <- my_data_train$realSum
y_test <- my_data_test$realSum

# Fit a Lasso regression model
lasso_model <- glmnet(x_train, y_train, alpha = 1)

# Select the best lambda value using cross-validation
cv_model <- cv.glmnet(x_train, y_train, alpha = 1, nfolds = 5)

# Plot the cross-validation results
plot(cv_model)
```




```r
# Select the lambda value that minimizes the mean
# cross-validation error
best_lambda <- cv_model$lambda.min

# Fit a Lasso regression model with the selected lambda
```

```r
# value
lasso_model_best <- glmnet(x_train, y_train, alpha = 1, lambda = best_lambda)

# Calculate R-squared and multiple R-squared
y_train_pred <- predict(lasso_model_best, newx = x_train)
y_train_mean <- mean(y_train)
SST <- sum((y_train - y_train_mean)^2)
SSR <- sum((y_train - y_train_pred)^2)
R_squared <- 1 - SSR/SST
multiple_R_squared <- cor(y_train_pred, y_train)^2
n <- length(y_train)
p <- ncol(x_train)
adj_R_squared <- 1 - (SSR/(n - p - 1))/(SST/(n - 1))


# Print the R-squared and multiple R-squared values
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.2176971
```

```r
cat("Adjusted R-squared:", adj_R_squared, "\n")
```

```
## Adjusted R-squared: 0.2153122
```

```r
# Evaluate the model performance
rmse <- sqrt(mean((my_data_test$realSum - y_train_pred)^2))
cat("RMSE on train set:", rmse, "\n")
```

```
## RMSE on train set: 324.7053
```

```r
x_test <- model.matrix(realSum ~ room_type + person_capacity +
    host_is_superhost + multi + biz + cleanliness_rating + guest_satisfaction_overall +
    bedrooms + poly(dist, 2) * poly(metro_dist, 2) * poly(attr_index_norm,
    2) * poly(rest_index_norm, 2) + lng + lat + city_day, data = my_data_test)[,
    -1]

# Calculate R-squared and multiple R-squared
y_test_pred <- predict(lasso_model_best, newx = x_test)
y_test_mean <- mean(y_test)
SST <- sum((y_test - y_test_mean)^2)
SSR <- sum((y_test - y_test_pred)^2)
R_squared <- 1 - SSR/SST
n <- length(y_test)
p <- ncol(x_test)
adj_R_squared <- 1 - (SSR/(n - p - 1))/(SST/(n - 1))


# Print the R-squared and multiple R-squared values
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.2821967
```

```r
cat("Adjusted R-squared:", adj_R_squared, "\n")
```

```
## Adjusted R-squared: 0.2770702
```

```r
# Evaluate the model performance
rmse <- sqrt(mean((my_data_test$realSum - y_test_pred)^2))
cat("RMSE on test set:", rmse, "\n")
```

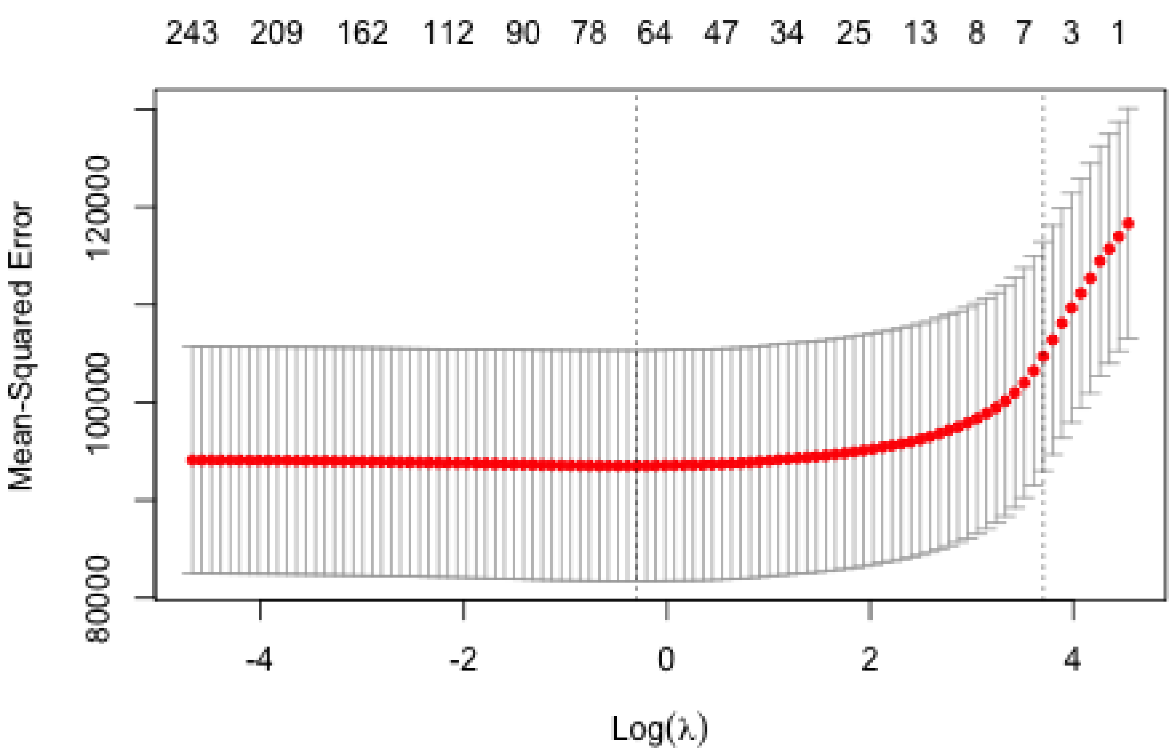
```
## RMSE on test set: 242.7917
```

```r
# Prepare the predictors and response variable
x_train <- model.matrix(realSum ~ room_type + person_capacity +
    host_is_superhost + multi + biz + cleanliness_rating + guest_satisfaction_overall +
    bedrooms + poly(dist, 3) * poly(metro_dist, 3) * poly(attr_index_norm,
    3) * poly(rest_index_norm, 3) + lng + lat + city_day, data = my_data_train)[,
    -1]
y_train <- my_data_train$realSum
y_test <- my_data_test$realSum

# Fit a Lasso regression model
lasso_model <- glmnet(x_train, y_train, alpha = 1)

# Select the best lambda value using cross-validation
cv_model <- cv.glmnet(x_train, y_train, alpha = 1, nfolds = 5)

# Plot the cross-validation results
plot(cv_model)
```



```r
# Select the lambda value that minimizes the mean
# cross-validation error
best_lambda <- cv_model$lambda.min

# Fit a Lasso regression model with the selected lambda
# value
lasso_model_best <- glmnet(x_train, y_train, alpha = 1, lambda = best_lambda)

# Calculate R-squared and multiple R-squared
y_train_pred <- predict(lasso_model_best, newx = x_train)
y_train_mean <- mean(y_train)
SST <- sum((y_train - y_train_mean)^2)
SSR <- sum((y_train - y_train_pred)^2)
R_squared <- 1 - SSR/SST
multiple_R_squared <- cor(y_train_pred, y_train)^2
n <- length(y_train)
p <- ncol(x_train)
adj_R_squared <- 1 - (SSR/(n - p - 1))/(SST/(n - 1))


# Print the R-squared and multiple R-squared values
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.2188426
```

```r
cat("Adjusted R-squared:", adj_R_squared, "\n")
```

```
## Adjusted R-squared: 0.2126426
```

```r
# Evaluate the model performance
rmse <- sqrt(mean((my_data_test$realSum - y_train_pred)^2))
cat("RMSE on train set:", rmse, "\n")
```

```
## RMSE on train set: 324.4427
```

```r
x_test <- model.matrix(realSum ~ room_type + person_capacity +
    host_is_superhost + multi + biz + cleanliness_rating + guest_satisfaction_overall +
    bedrooms + poly(dist, 3) * poly(metro_dist, 3) * poly(attr_index_norm,
    3) * poly(rest_index_norm, 3) + lng + lat + city_day, data = my_data_test)[,
    -1]

# Calculate R-squared and multiple R-squared
y_test_pred <- predict(lasso_model_best, newx = x_test)
y_test_mean <- mean(y_test)
SST <- sum((y_test - y_test_mean)^2)
SSR <- sum((y_test - y_test_pred)^2)
R_squared <- 1 - SSR/SST
n <- length(y_test)
p <- ncol(x_test)
adj_R_squared <- 1 - (SSR/(n - p - 1))/(SST/(n - 1))
```

```r
# Print the R-squared and multiple R-squared values
cat("R-squared:", R_squared, "\n")
```

```
## R-squared: 0.2587531
```

```r
cat("Adjusted R-squared:", adj_R_squared, "\n")
```

```
## Adjusted R-squared: 0.2448794
```

```r
# Evaluate the model performance
rmse <- sqrt(mean((my_data_test$realSum - y_test_pred)^2))
cat("RMSE on test set:", rmse, "\n")
```

```
## RMSE on test set: 246.7246
```

Even step regression is not good because of extremely low value of R^2 even in polynomial model of power 2 and 3.

Random Forest Regression

```r
rf_model <- randomForest(realSum ~ room_type + host_is_superhost +
    multi + biz + city_day + person_capacity + cleanliness_rating +
    guest_satisfaction_overall + bedrooms + dist + metro_dist +
    attr_index_norm + rest_index_norm + lng + lat, data = my_data_train)
```

```r
predictions <- predict(rf_model, my_data_train)
```

```r
rmse <- sqrt(mean((my_data_train$realSum - predictions)^2))
rmse
```

```
## [1] 120.6731
```

```r
r_squared <- 1 - (sum((my_data_train$realSum - predictions)^2)/sum((my_data_train$realSum -
    mean(my_data_train$realSum))^2))
r_squared
```

```
## [1] 0.8770571
```

```r
adj_R_squared <- 1 - ((1 - r_squared) * (n - 1)/(n - p - 1))
adj_R_squared
```

```
## [1] 0.874756
```

```r
predictions <- predict(rf_model, my_data_test)
```

```r
rmse <- sqrt(mean((my_data_test$realSum - predictions)^2))
rmse
```

```
## [1] 140.1627
```

```r
r_squared <- 1 - (sum((my_data_test$realSum - predictions)^2)/sum((my_data_test$realSum -
    mean(my_data_test$realSum))^2))
r_squared
```

```
## [1] 0.7607771
```

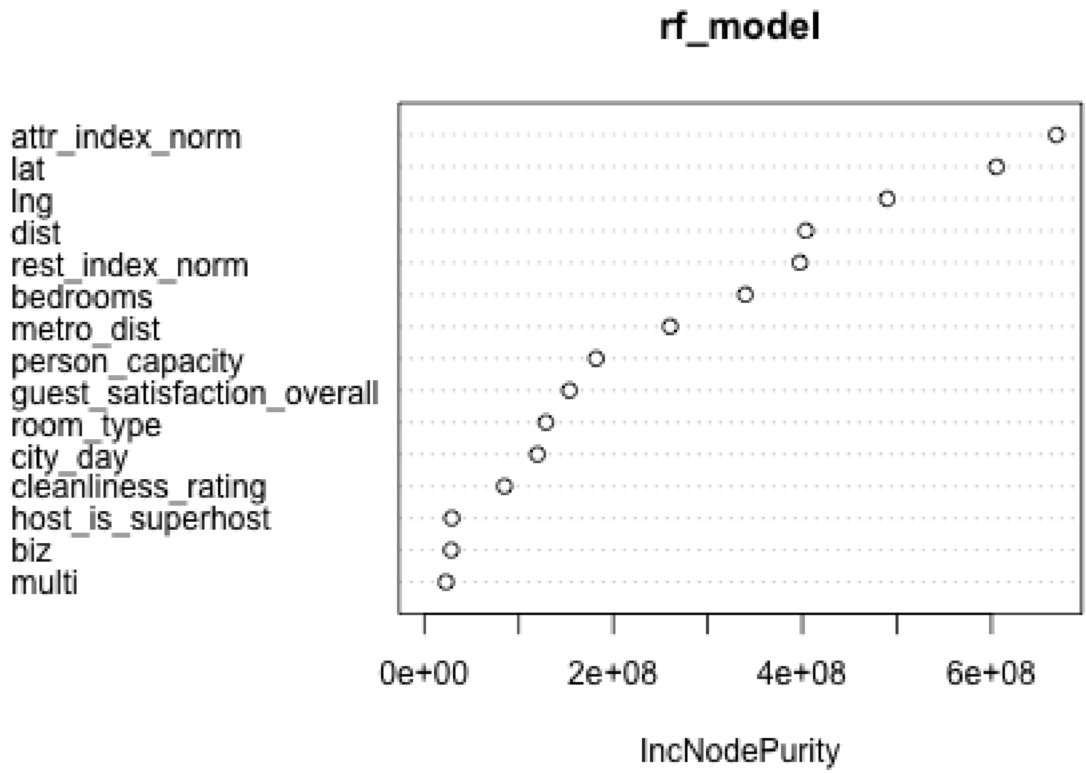
```r
adj_R_squared <- 1 - ((1 - r_squared) * (n - 1)/(n - p - 1))
adj_R_squared
```

```
## [1] 0.7562996
```

```r
importance(rf_model)
```

```
##                            IncNodePurity
## room_type                      128820124
## host_is_superhost               29206740
## multi                           23032640
## biz                             28483381
## city_day                       119817452
## person_capacity                181945316
## cleanliness_rating              84896646
## guest_satisfaction_overall     153697848
## bedrooms                       339811264
## dist                           403842463
## metro_dist                     260251950
## attr_index_norm                668769759
## rest_index_norm                397578038
## lng                            489702071
## lat                            605397008
```

```r
varImpPlot(rf_model)
```


rf_model
attr_index_norm
lat
lng
dist
rest_index_norm
bedrooms
metro_dist
person_capacity
guest_satisfaction_overall
room_type
city_day
cleanliness_rating
host_is_superhost
biz
multi
0e+00
2e+08
4e+08
6e+08
IncNodePurity