Airbnb

Objective:

Airbnb is a home-sharing platform that allows house owners and renters ('hosts') to put their properties ('listings') online, so that guests can pay to stay in them. Hosts set their own prices for their listings, and although Airbnb and other sites provide some general guidance, there are currently no free and accurate services which help hosts price their properties using a wide range of data points.

It's really important to get pricing right on Airbnb, especially in big cities like London where there is a lot of competition. In this project I have analysed the Airbnb data the main concentrations on "is reviews affecting the price". exploring the listing data to extract interesting and useful insights, in order to help the users can book in good house and good locations.

The dataset:

The dataset used for this project comes from Insideairbnb.com, an anti-Airbnb lobby group that scrapes Airbnb listings, reviews and calendar data from multiple cities around the world. The dataset was scraped on 9 April 2019 and contains information on all London Airbnb listings that were live on the site on that date (about 80,000). A shape file of London borough boundaries was also downloaded.

The data is quite messy and has some limitations. The major one is that it only includes the advertised price (sometimes called the 'sticker' price). The sticker price is the overall nightly price that is advertised to potential guests, rather than the actual average amount paid per night by previous guests. The advertised prices can be set to any arbitrary amount by the host, and hosts that are less experienced with Airbnb will often set these to very low.

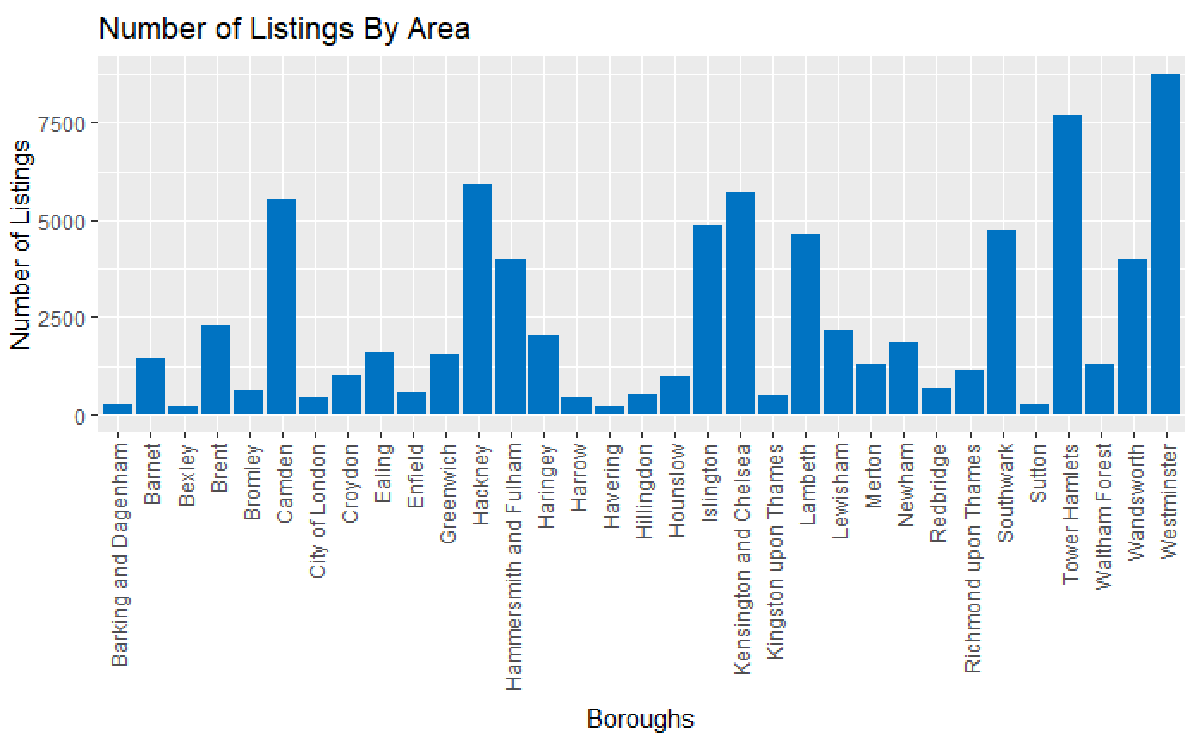
Nevertheless, this dataset can still be used as a proof of concept. A more accurate version could be built using data on the actual average nightly rates paid by guests, e.g. from sites like AirDNA that scrape and sell higher quality Airbnb data.

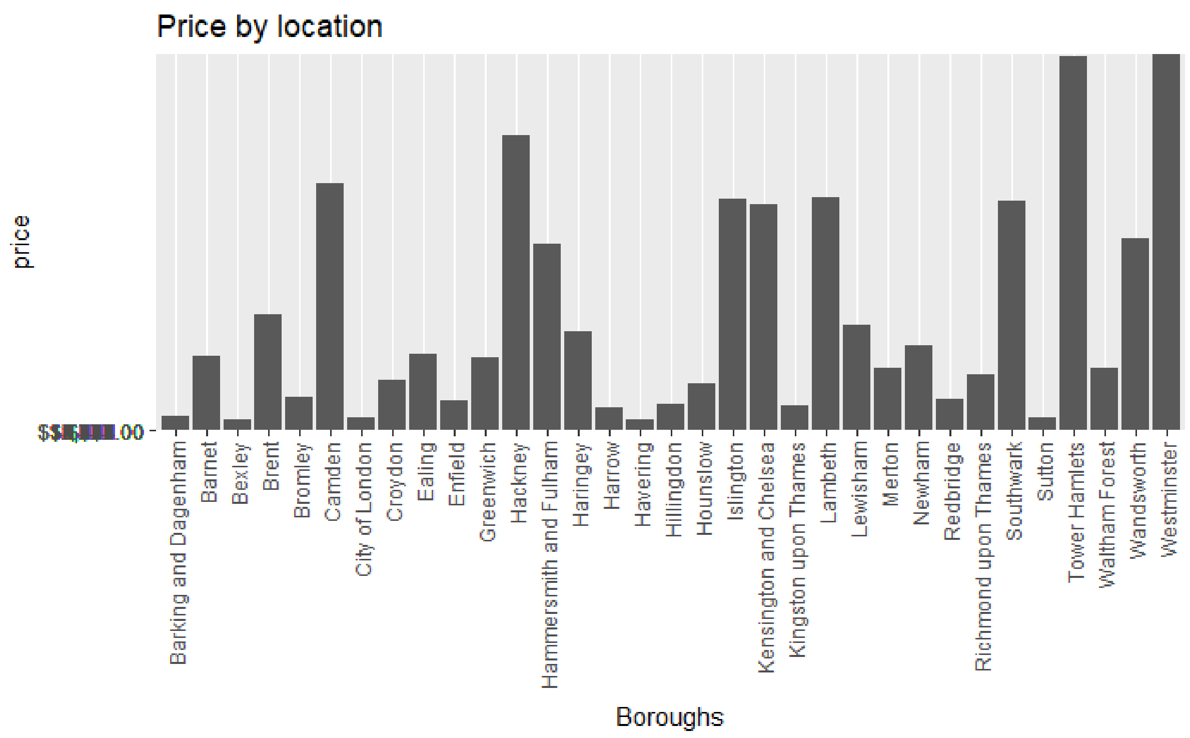
Exploring the data for insights:

Location, location, location:

which areas of London have the most Airbnb properties, and which are the most expensive?

Westminster (west of the City) has the most Airbnb properties, followed by Tower Hamlets (east of the City). Inner London boroughs have significantly more listings than outer London boroughs. However the pattern with prices is slightly different. Kensington and Chelsea (to the west of Westminster) is the most expensive area. This is a famously expensive area to live, with some of the highest house prices in the world. Although inner London is generally more expensive than outer London, there are also some more expensive listings spread out to the west of the city along the Thames (which has some very beautiful areas).There are very few listings in the very centre of London (as there is relatively little residential property here), but as a result they are very expensive.

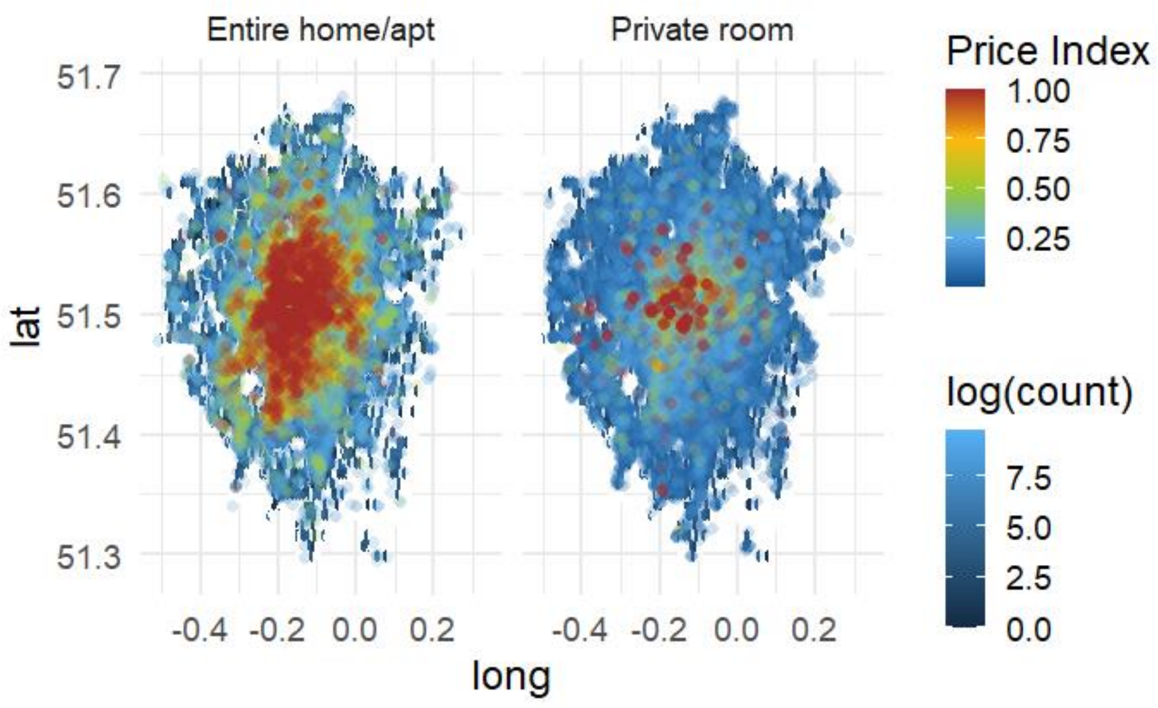




Assessing the competition

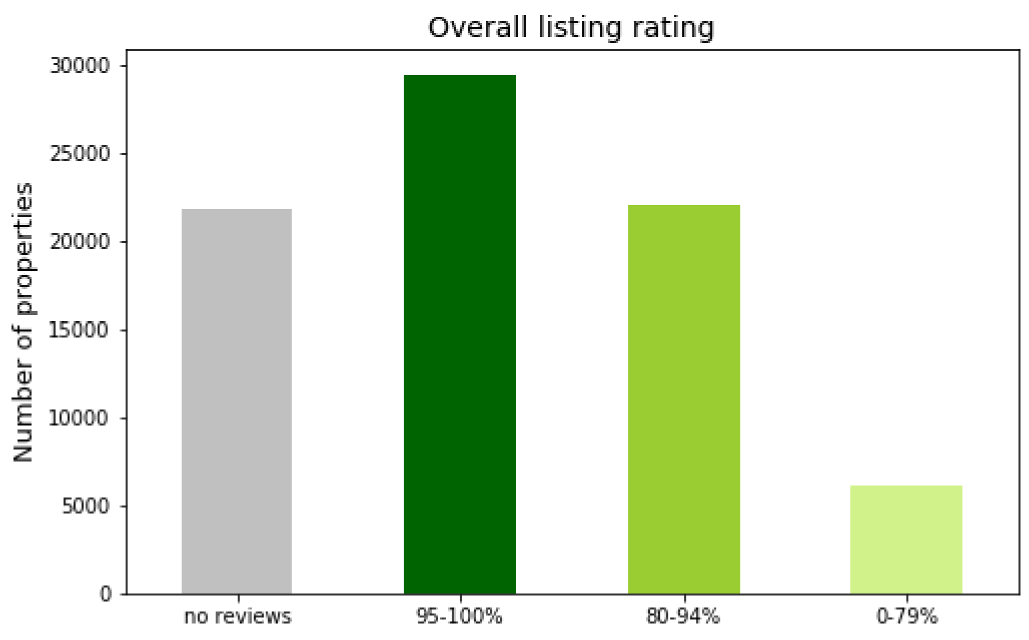
Question: what are the most common property and room types?

Answer: about 70% of properties are apartments. The remainder are houses or more uncommon property types. About 55% of listings are entire homes (i.e. you are renting the entire property on your own). Most of the remainder are private rooms (i.e. you are renting a bedroom and possibly also a bathroom, but there will be other people in the property). Fewer than 1% are shared rooms (i.e. you are sharing a room with either the property owner or other guests).



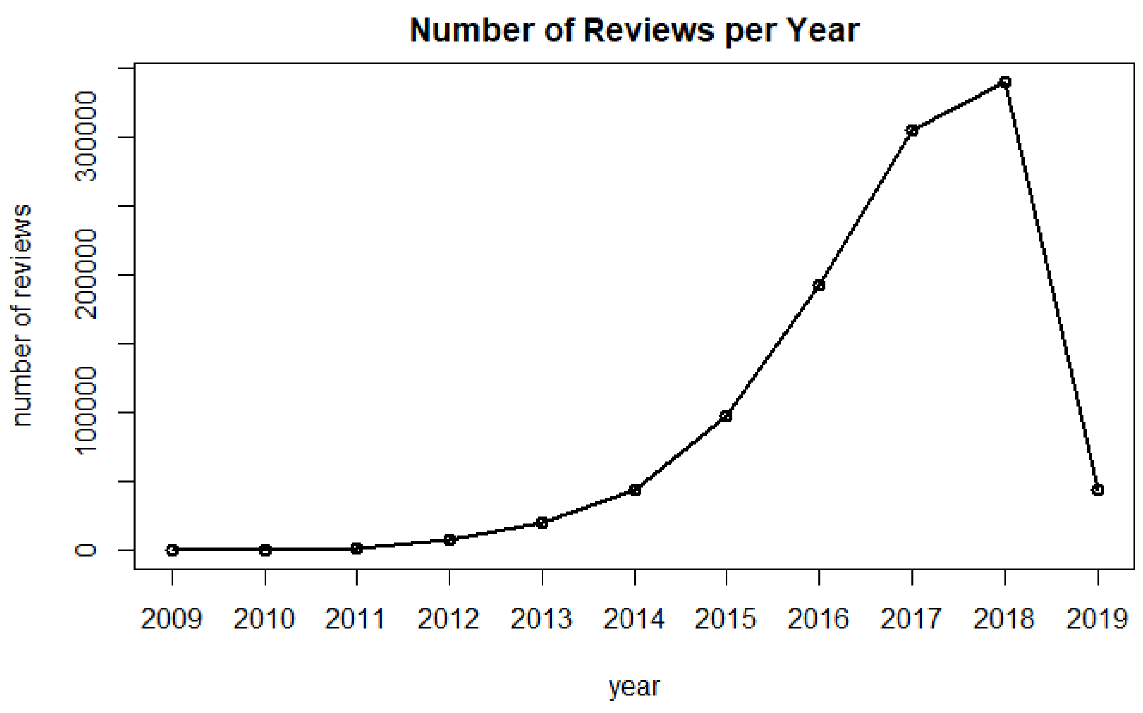
Question: what is the distribution of reviews like?

people love their Airbnbs. For every review category (accuracy, cleanliness, check-in, communication, location, value and overall), the majority of listings that have had a review received a 10/10 rating for that category (or 95–100/100 overall). So, probably best to make sure you’re in this category. Ratings or 8 or below are relatively rare. Guests seem to be most positive about communication, check-ins and accuracy. About a quarter of listings have not yet been reviewed. These will mostly be properties which have not yet had a completed stay, many of which may be ‘inactive’ properties — i.e. those which, although listed on the site, do not have any dates available to book.



Question: Number of reviews by year?

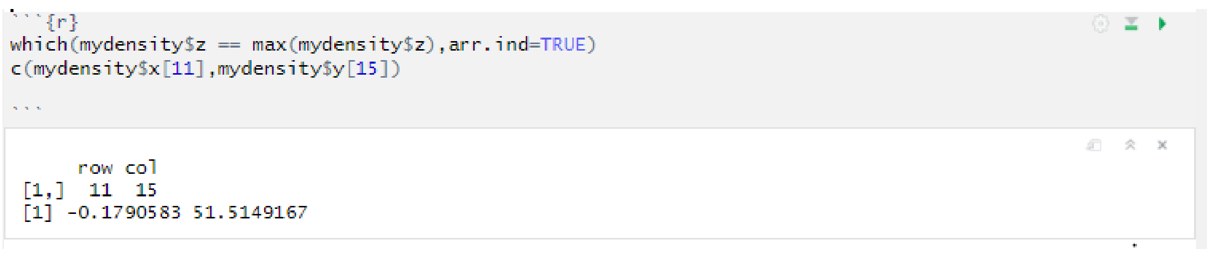
The number of reviews by year is increased with full speed from 2014 to 2018. In the year 2018 it crossed 30000 reviews in year and in the year 2019 we can see reviews are less than 5000. Because we have data only until April.



Model:

By making the centre point four different centrality are created with longitude and latitude points. central points as the middle of the map is pointed and average coordinate are pointed to find the which location has more reviews after find the number of reviews per locations density of reviews are plotted on map. and compare with other locations to check whether reviews are affecting the prices.

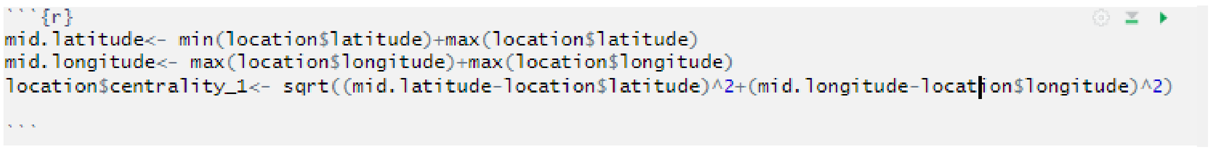
get the x and y values for the densest location to set it as our point of centrality.



~~~
```{r}
which(mydensity$z == max(mydensity$z),arr.ind=TRUE)
c(mydensity$x[11],mydensity$y[15])

```
~~~

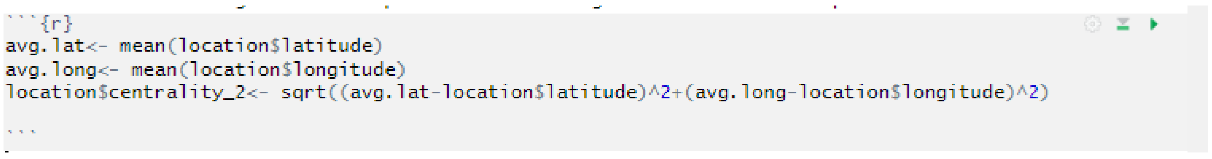
```
     row col
[1,]  11  15
[1] -0.1790583 51.5149167
```

CENTRALITY 1: basic centrality, setting the central point as the middle of the map.



~~~
```{r}
mid.latitude<- min(location$latitude)+max(location$latitude)
mid.longitude<- max(location$longitude)+max(location$longitude)
location$centrality_1<- sqrt((mid.latitude-location$latitude)^2+(mid.longitude-location$longitude)^2)

```
~~~

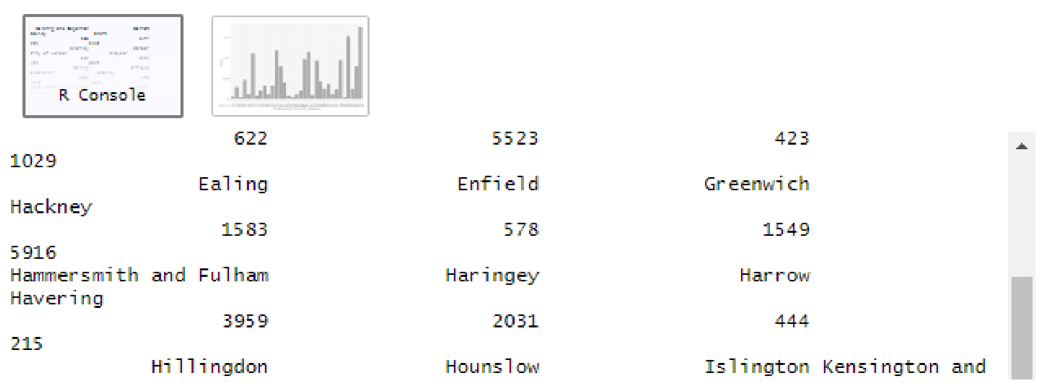
CENTRALITY 2: setting the central point as the average coordinate of all points



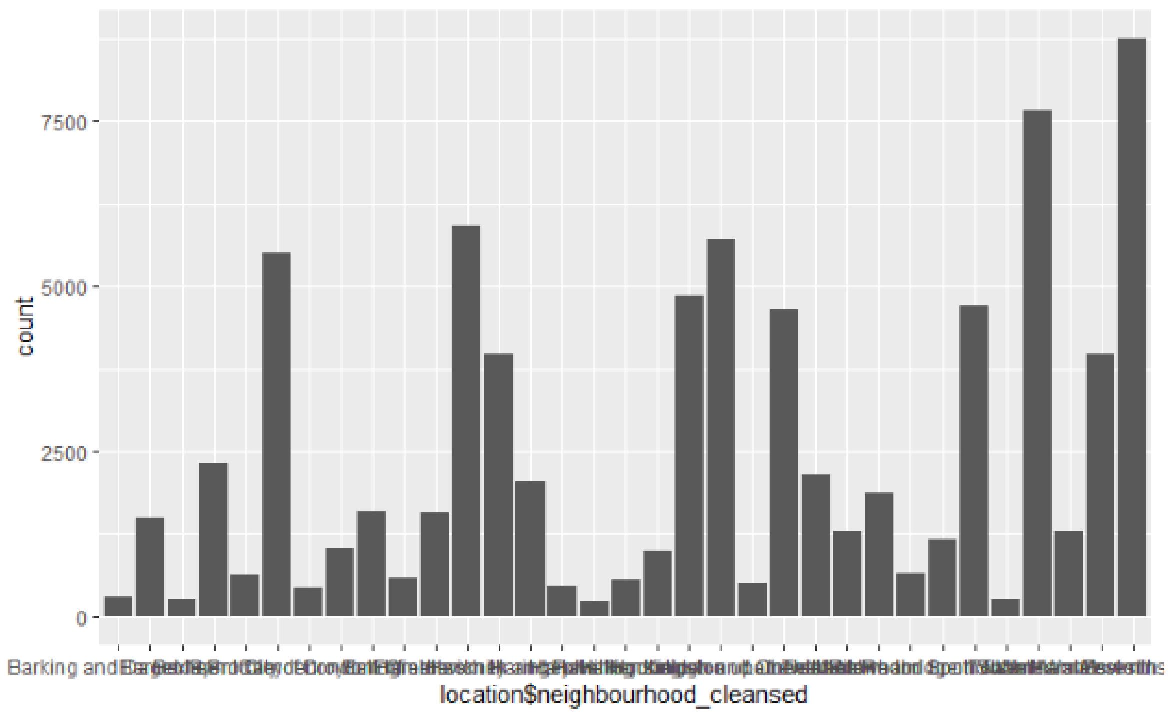
~~~
```{r}
avg.lat<- mean(location$latitude)
avg.long<- mean(location$longitude)
location$centrality_2<- sqrt((avg.lat-location$latitude)^2+(avg.long-location$longitude)^2)

```
~~~

CENTRALITY 3: setting the central point as the average coordinate of points in Westminster finding the borough with most listings using a frequency table

~~~
```{r}
table(location$neighbourhood_cleansed)
ggplot(data=location, aes(x=location$neighbourhood_cleansed))+geom_bar()

```
~~~



```
                      622                  5523                 423
1029
                   Ealing               Enfield           Greenwich
Hackney
                     1583                   578                1549
5916
Hammersmith and Fulham                Haringey              Harrow
Havering
                     3959                  2031                 444
215
               Hillingdon              Hounslow           Islington Kensington and
```

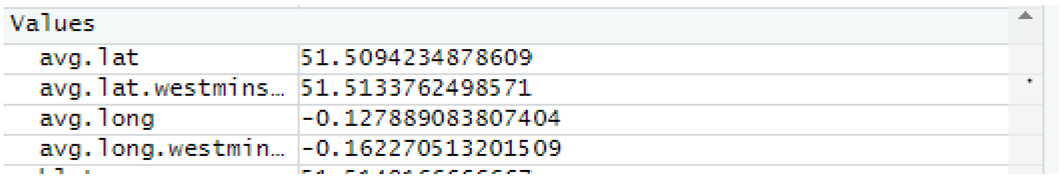


By, the centrality we found that Westminster that is 8749 has more count of reviews and Sutton has less review with 247. and now we check if reviews are affecting prices are not.

now that we know Westminster is has the highest frequency, we will use the average point in Westminster as the central point.

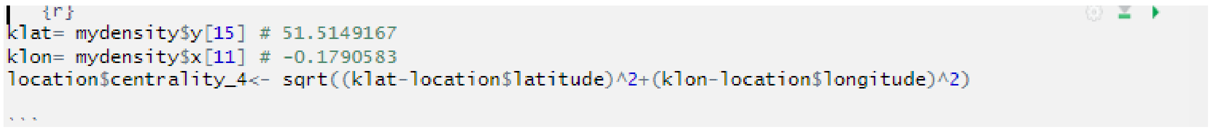
~~~
```{r}
westminster.plots<-location[location$neighbourhood_cleansed=="Westminster",]
avg.lat.westminster<- mean(westminster.plots$latitude)
avg.long.westminster<- mean(westminster.plots$longitude)
location$centrality_3<- sqrt((avg.lat.westminster-location$latitude)^2+(avg.long.westminster-location$
longitude)^2)

```
~~~

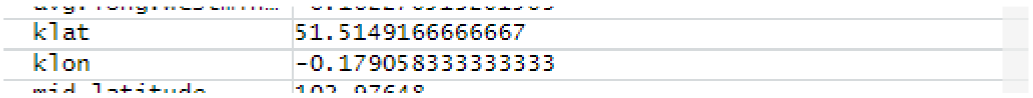
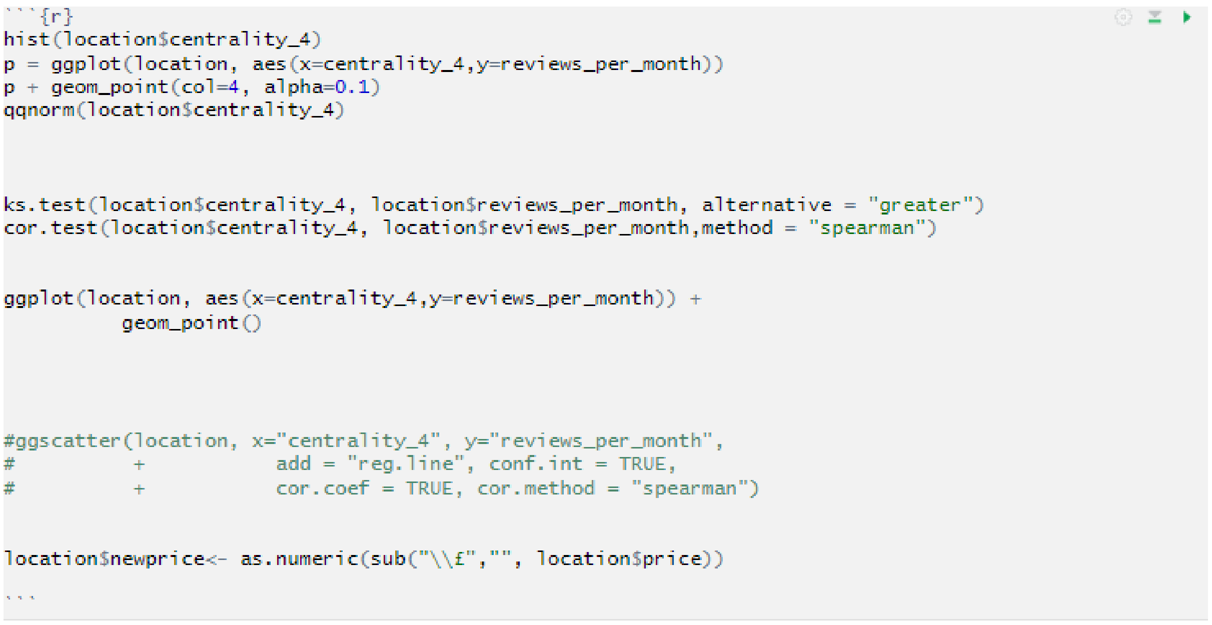
Output:

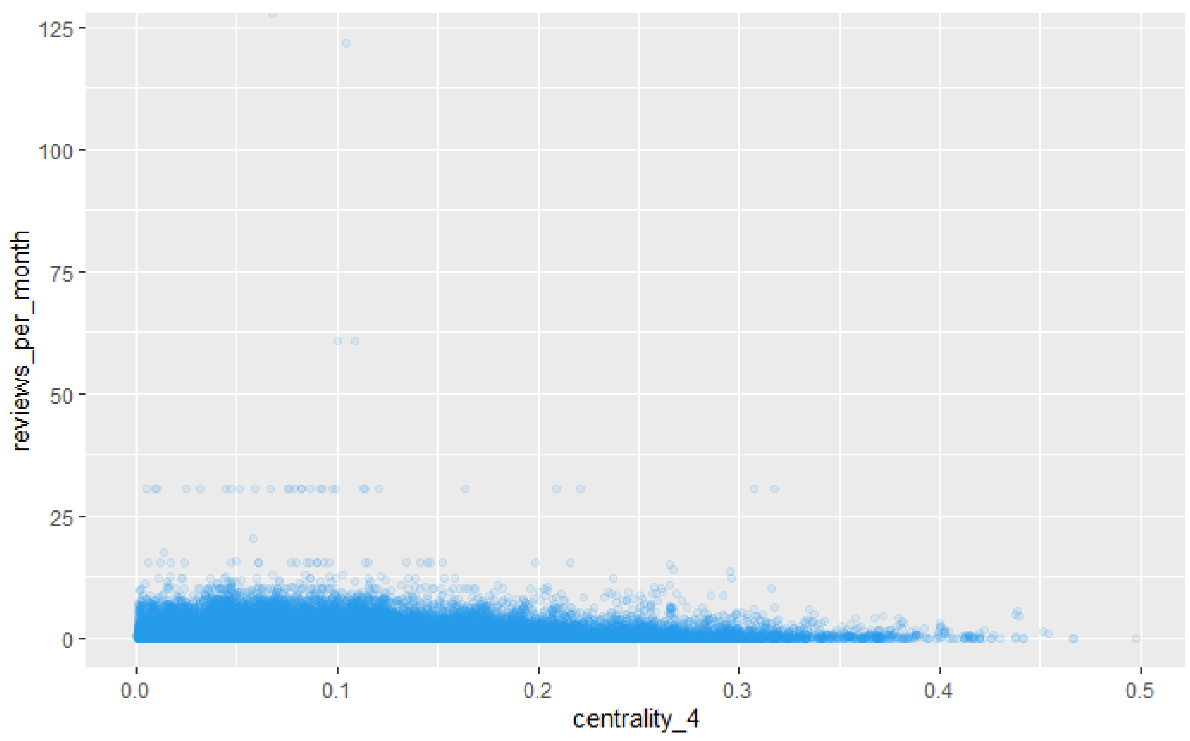


```
Values
  avg.lat               51.5094234878609
  avg.lat.westmins…     51.5133762498571
  avg.long              -0.127889083807404
  avg.long.westmin…     -0.162270513201509
```

CENTRALITY 4: setting the central point as the area with highest kernel density



```
 {r}
klat= mydensity$y[15] # 51.5149167
klon= mydensity$x[11] # -0.1790583
location$centrality_4<- sqrt((klat-location$latitude)^2+(klon-location$longitude)^2)

```
```

Output:



```
  klat      51.5149166666667
  klon      -0.179058333333333
```

Now, we plot the reviews per month by taking the centrality_4.



````
```{r}
hist(location$centrality_4)
p = ggplot(location, aes(x=centrality_4,y=reviews_per_month))
p + geom_point(col=4, alpha=0.1)
qqnorm(location$centrality_4)



ks.test(location$centrality_4, location$reviews_per_month, alternative = "greater")
cor.test(location$centrality_4, location$reviews_per_month,method = "spearman")


ggplot(location, aes(x=centrality_4,y=reviews_per_month)) +
          geom_point()




#ggscatter(location, x="centrality_4", y="reviews_per_month",
#          +           add = "reg.line", conf.int = TRUE,
#          +           cor.coef = TRUE, cor.method = "spearman")


location$newprice<- as.numeric(sub("\\£","", location$price))

```
````

Output:
````

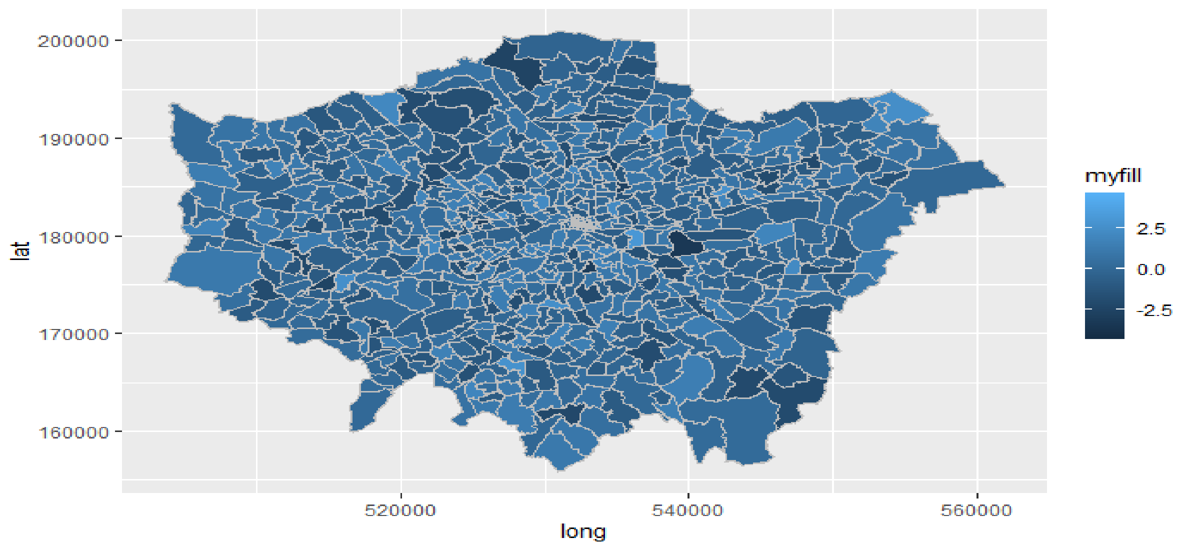


Now the shapefile can be plotted as either a geom_path or a geom_polygon.

Paths handle clipping better. Polygons can be filled.

You need the aesthetics long, lat, and group.

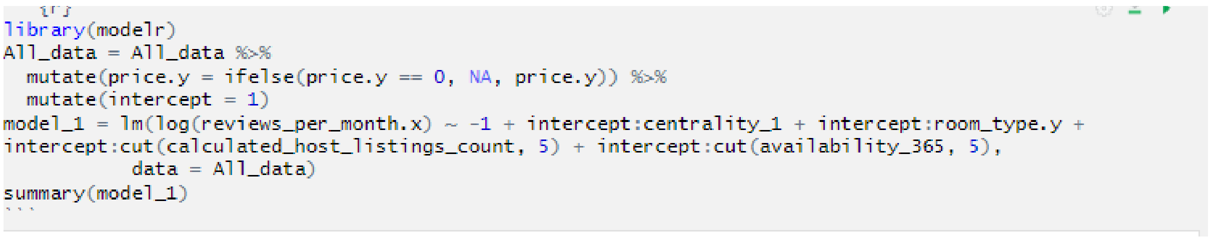
Output:

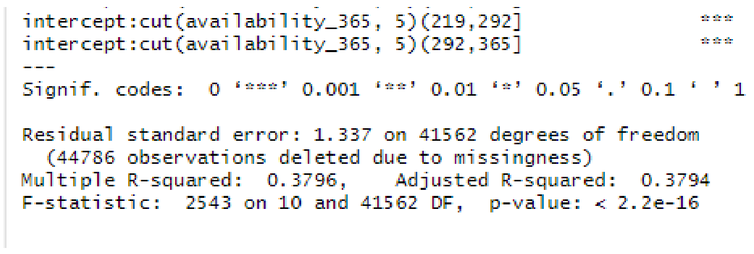


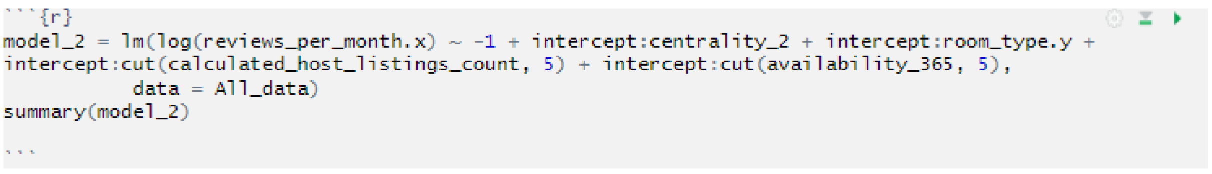
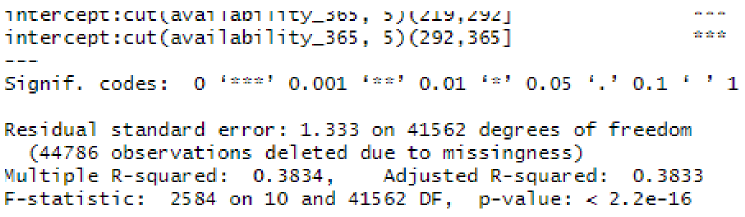
Linear regression:

By using the linear regression, we predict whether which centrality is best fit and which is bad fit as taking price, reviews and host count.

Model_1:

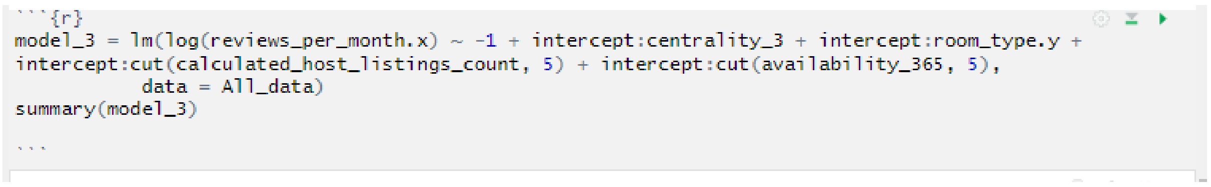


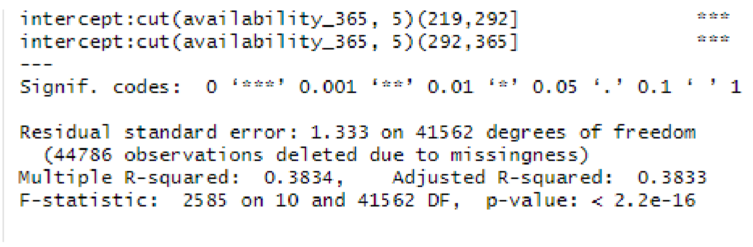
```
  {r}
library(modelr)
All_data = All_data %>%
  mutate(price.y = ifelse(price.y == 0, NA, price.y)) %>%
  mutate(intercept = 1)
model_1 = lm(log(reviews_per_month.x) ~ -1 + intercept:centrality_1 + intercept:room_type.y +
intercept:cut(calculated_host_listings_count, 5) + intercept:cut(availability_365, 5),
           data = All_data)
summary(model_1)
```
```

Output:



```
intercept:cut(availability_365, 5)(219,292]                 ***
intercept:cut(availability_365, 5)(292,365]                 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

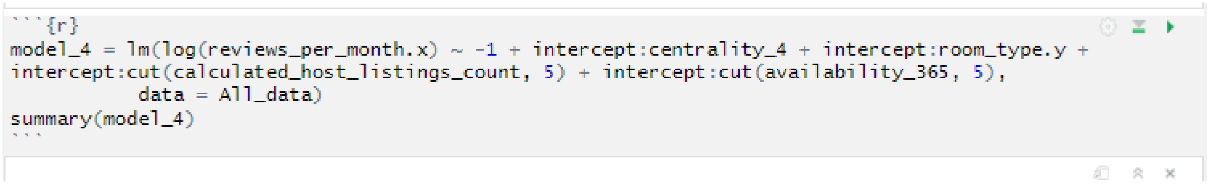
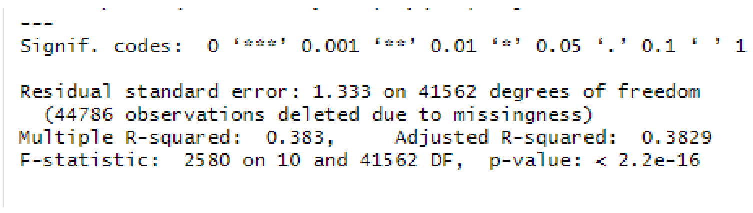
Residual standard error: 1.337 on 41562 degrees of freedom
  (44786 observations deleted due to missingness)
Multiple R-squared:  0.3796,    Adjusted R-squared:  0.3794
F-statistic:  2543 on 10 and 41562 DF,  p-value: < 2.2e-16
```

We can see that model_1 has 0.3794 adjected square and 1.337 Residual standard error.

Model_2:



```
```{r}
model_2 = lm(log(reviews_per_month.x) ~ -1 + intercept:centrality_2 + intercept:room_type.y +
intercept:cut(calculated_host_listings_count, 5) + intercept:cut(availability_365, 5),
           data = All_data)
summary(model_2)

```
```

Output:



```
intercept:cut(availability_365, 5)(219,292]                 ***
intercept:cut(availability_365, 5)(292,365]                 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

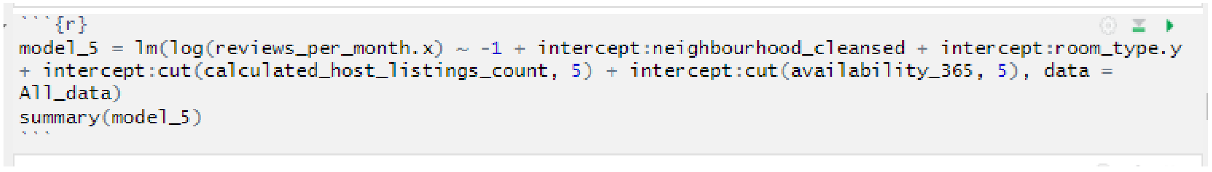
Residual standard error: 1.333 on 41562 degrees of freedom
  (44786 observations deleted due to missingness)
Multiple R-squared:  0.3834,    Adjusted R-squared:  0.3833
F-statistic:  2584 on 10 and 41562 DF,  p-value: < 2.2e-16
```

In the above output we can see that model_2 has 0.3833 and 1.333 residual standard error.
```

Model_3:

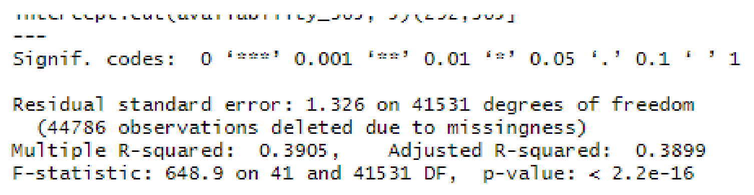


~~~r
```{r}
model_3 = lm(log(reviews_per_month.x) ~ -1 + intercept:centrality_3 + intercept:room_type.y +
intercept:cut(calculated_host_listings_count, 5) + intercept:cut(availability_365, 5),
            data = All_data)
summary(model_3)

~~~
```

Output:



```
intercept:cut(availability_365, 5)(219,292]              ***
intercept:cut(availability_365, 5)(292,365]              ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.333 on 41562 degrees of freedom
  (44786 observations deleted due to missingness)
Multiple R-squared:  0.3834,	Adjusted R-squared:  0.3833 
F-statistic:  2585 on 10 and 41562 DF,  p-value: < 2.2e-16
~~~

In model_3 output we can see that it has 0.3833 adjusted R-squared and 1.333 Residual error.

Model_4:



```r
```{r}
model_4 = lm(log(reviews_per_month.x) ~ -1 + intercept:centrality_4 + intercept:room_type.y +
intercept:cut(calculated_host_listings_count, 5) + intercept:cut(availability_365, 5),
            data = All_data)
summary(model_4)
~~~
```

Output:



```
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.333 on 41562 degrees of freedom
  (44786 observations deleted due to missingness)
Multiple R-squared:  0.383,	Adjusted R-squared:  0.3829 
F-statistic:  2580 on 10 and 41562 DF,  p-value: < 2.2e-16
~~~

In the above output we can see that model_4 has 0.3829 adjusted R-squared and 1.333 standard error.

Model_5:



```r
```{r}
model_5 = lm(log(reviews_per_month.x) ~ -1 + intercept:neighbourhood_cleansed + intercept:room_type.y
+ intercept:cut(calculated_host_listings_count, 5) + intercept:cut(availability_365, 5), data =
All_data)
summary(model_5)
~~~
```
```

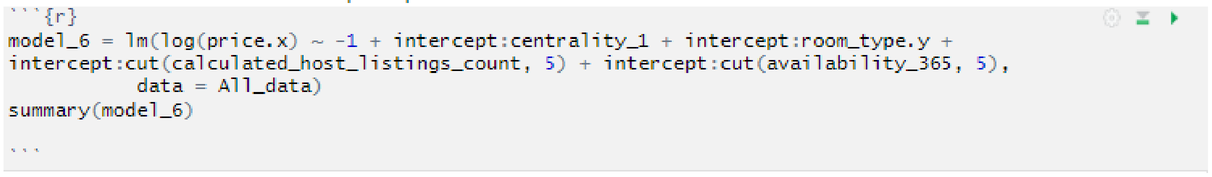
Output:



```
intercept:cut(availability_365, 5)(292,365]
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.326 on 41531 degrees of freedom
  (44786 observations deleted due to missingness)
Multiple R-squared:  0.3905,	Adjusted R-squared:  0.3899 
F-statistic: 648.9 on 41 and 41531 DF,  p-value: < 2.2e-16
```

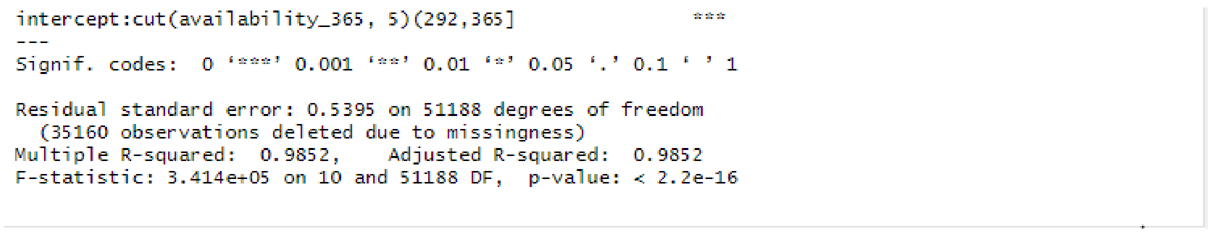
In the model_5 output we can see that Adjusted R-squared value is 0.3899 and residual standard error value is 1.326.

Model_6:



~~~
```{r}
model_6 = lm(log(price.x) ~ -1 + intercept:centrality_1 + intercept:room_type.y +
intercept:cut(calculated_host_listings_count, 5) + intercept:cut(availability_365, 5),
            data = All_data)
summary(model_6)

```
~~~

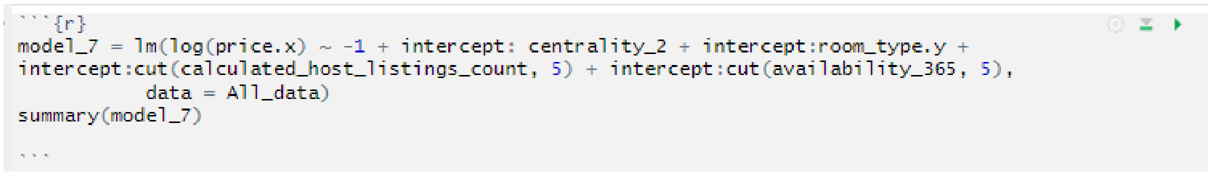
Output:



```
intercept:cut(availability_365, 5)(292,365]              ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5395 on 51188 degrees of freedom
  (35160 observations deleted due to missingness)
Multiple R-squared:  0.9852,	Adjusted R-squared:  0.9852 
F-statistic: 3.414e+05 on 10 and 51188 DF,  p-value: < 2.2e-16
```

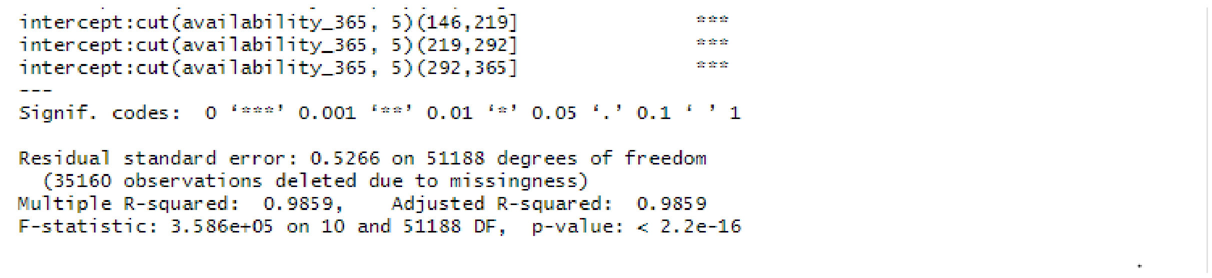
In the above model we can see that Adjusted R-squared value is 0.9852 and standard error is 0.5395.

Model_7:



~~~
```{r}
model_7 = lm(log(price.x) ~ -1 + intercept: centrality_2 + intercept:room_type.y +
intercept:cut(calculated_host_listings_count, 5) + intercept:cut(availability_365, 5),
            data = All_data)
summary(model_7)

```
~~~

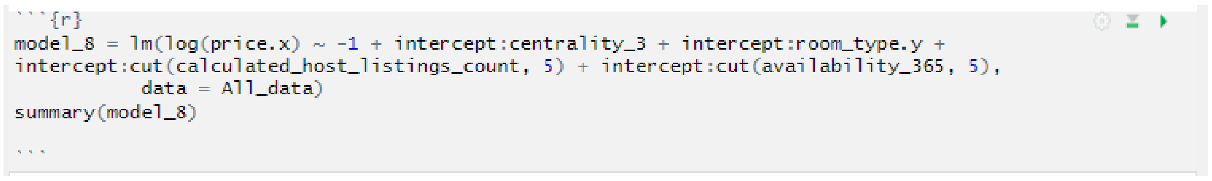
Output:



```
intercept:cut(availability_365, 5)(146,219]              ***
intercept:cut(availability_365, 5)(219,292]              ***
intercept:cut(availability_365, 5)(292,365]              ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5266 on 51188 degrees of freedom
  (35160 observations deleted due to missingness)
Multiple R-squared:  0.9859,	Adjusted R-squared:  0.9859 
F-statistic: 3.586e+05 on 10 and 51188 DF,  p-value: < 2.2e-16
```

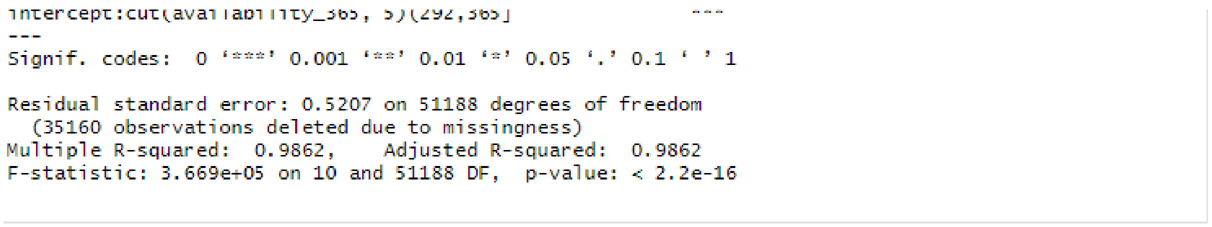
In model_7 we can see that it has 0.9859 adjusted R-squared and 0.5266 standard error.

Model_8:



~~~
```{r}
model_8 = lm(log(price.x) ~ -1 + intercept:centrality_3 + intercept:room_type.y +
intercept:cut(calculated_host_listings_count, 5) + intercept:cut(availability_365, 5),
                data = All_data)
summary(model_8)

```
~~~

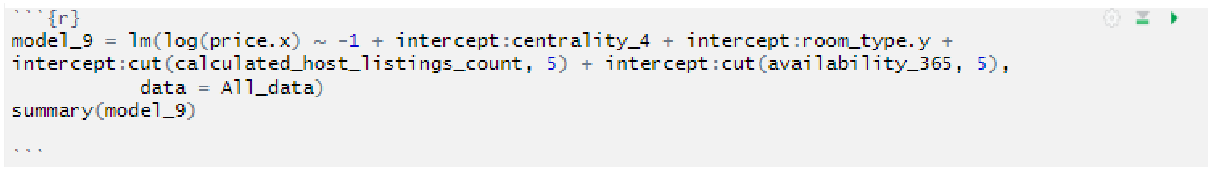
Output:



```
intercept:cut(availability_365, 5)(292,365]          ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5207 on 51188 degrees of freedom
  (35160 observations deleted due to missingness)
Multiple R-squared:  0.9862,	Adjusted R-squared:  0.9862 
F-statistic: 3.669e+05 on 10 and 51188 DF,  p-value: < 2.2e-16
```

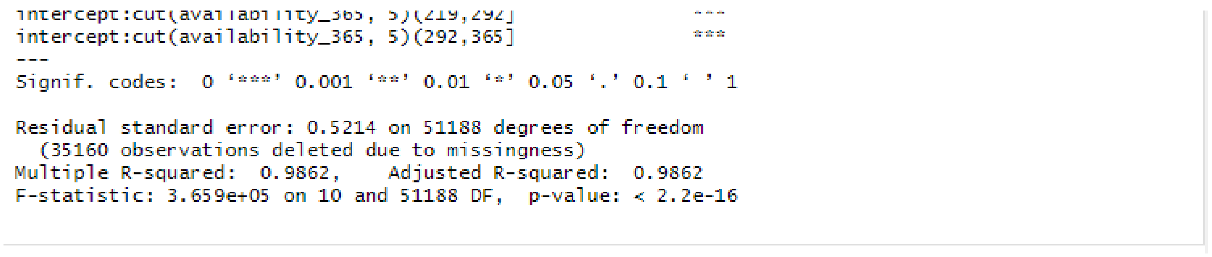
In model 8 we can see that it has 0.9862 adjusted R-squared value and 0.5207 standard error.

Model_9:



~~~
```{r}
model_9 = lm(log(price.x) ~ -1 + intercept:centrality_4 + intercept:room_type.y +
intercept:cut(calculated_host_listings_count, 5) + intercept:cut(availability_365, 5),
                data = All_data)
summary(model_9)

```
~~~

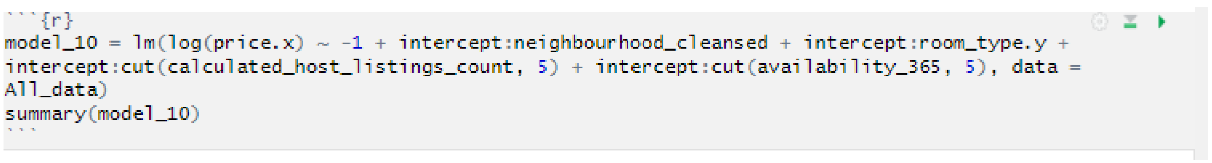
Output:



```
intercept:cut(availability_365, 5)(219,292]          ***
intercept:cut(availability_365, 5)(292,365]          ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5214 on 51188 degrees of freedom
  (35160 observations deleted due to missingness)
Multiple R-squared:  0.9862,	Adjusted R-squared:  0.9862 
F-statistic: 3.659e+05 on 10 and 51188 DF,  p-value: < 2.2e-16
```

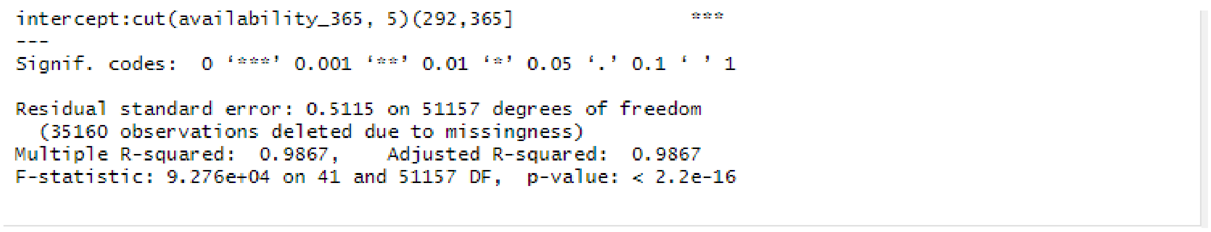
In model 9 adjusted R-squared is 0.9862 and residual standard error is 0.5214.

Model_10:



~~~
```{r}
model_10 = lm(log(price.x) ~ -1 + intercept:neighbourhood_cleansed + intercept:room_type.y +
intercept:cut(calculated_host_listings_count, 5) + intercept:cut(availability_365, 5), data =
All_data)
summary(model_10)
```
~~~

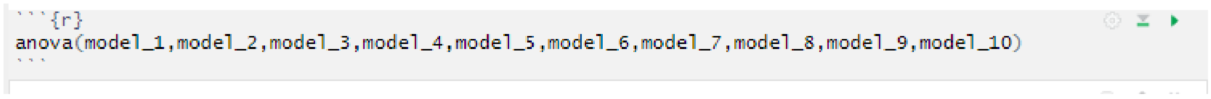
Output:



```
intercept:cut(availability_365, 5)(292,365]              ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5115 on 51157 degrees of freedom
  (35160 observations deleted due to missingness)
Multiple R-squared:  0.9867,    Adjusted R-squared:  0.9867
F-statistic: 9.276e+04 on 41 and 51157 DF,  p-value: < 2.2e-16
```

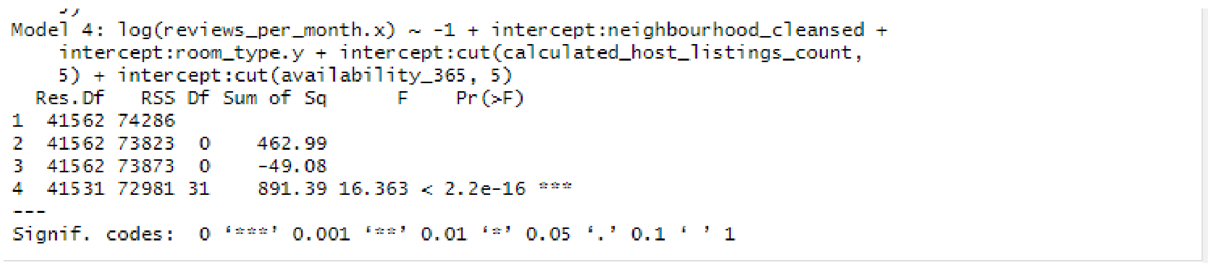
In the model 10 we can see that Adjusted R-squared value is 0.9867 and residual standard error is 0.5115.

Now, using the Anova function we can compare the one are more linear regressions model.



~~~
```{r}
anova(model_1,model_2,model_3,model_4,model_5,model_6,model_7,model_8,model_9,model_10)
```
~~~

Output:

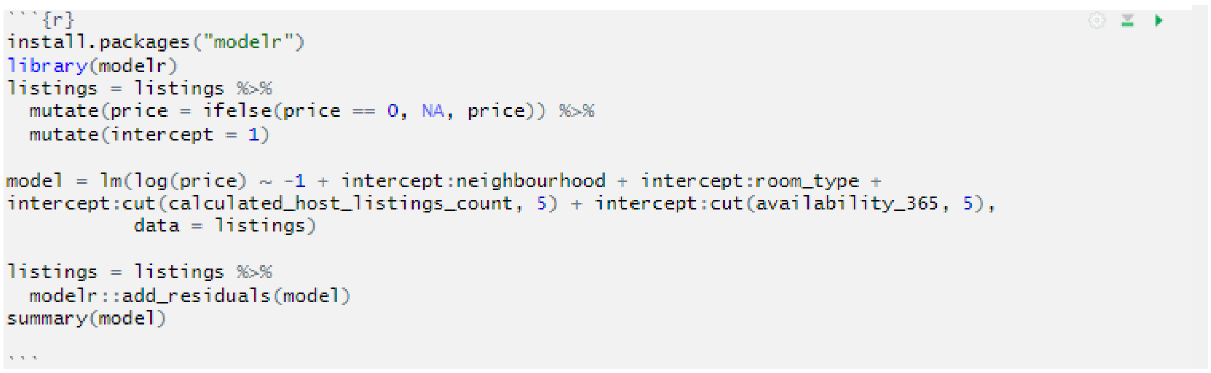


```
Model 4: log(reviews_per_month.x) ~ -1 + intercept:neighbourhood_cleansed +
    intercept:room_type.y + intercept:cut(calculated_host_listings_count,
    5) + intercept:cut(availability_365, 5)
  Res.Df   RSS Df Sum of Sq      F    Pr(>F)
1  41562 74286
2  41562 73823  0    462.99
3  41562 73873  0    -49.08
4  41531 72981 31    891.39 16.363 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Model	Adjusted R-squared	Standard error
Model_1	0.3794	1.337
Model_2	0.3833	1.333
Model_3	0.3833	1.333
Model_4	0.3829	1.333
Model_5	0.3899	1.326
Model_6	0.9852	0.5266
Model_7	0.9859	0.5266
Model_8	0.9862	0.5207
Model_9	0.9862	0.5214
Model_10	0.9867	0.5115

By, Comparing the all the 10 model’s model_10 has the best linear regression model with adjected R square is 0.9867 and standard error-0.5115. and model_4 has bad fit with adjected R square 0.3829 and standard error-1.333.

Now, we predict the prices of location by using the linear regression model.



~~~
```{r}
install.packages("modelr")
library(modelr)
listings = listings %>%
  mutate(price = ifelse(price == 0, NA, price)) %>%
  mutate(intercept = 1)

model = lm(log(price) ~ -1 + intercept:neighbourhood + intercept:room_type +
intercept:cut(calculated_host_listings_count, 5) + intercept:cut(availability_365, 5),
           data = listings)

listings = listings %>%
  modelr::add_residuals(model)
summary(model)
```
~~~

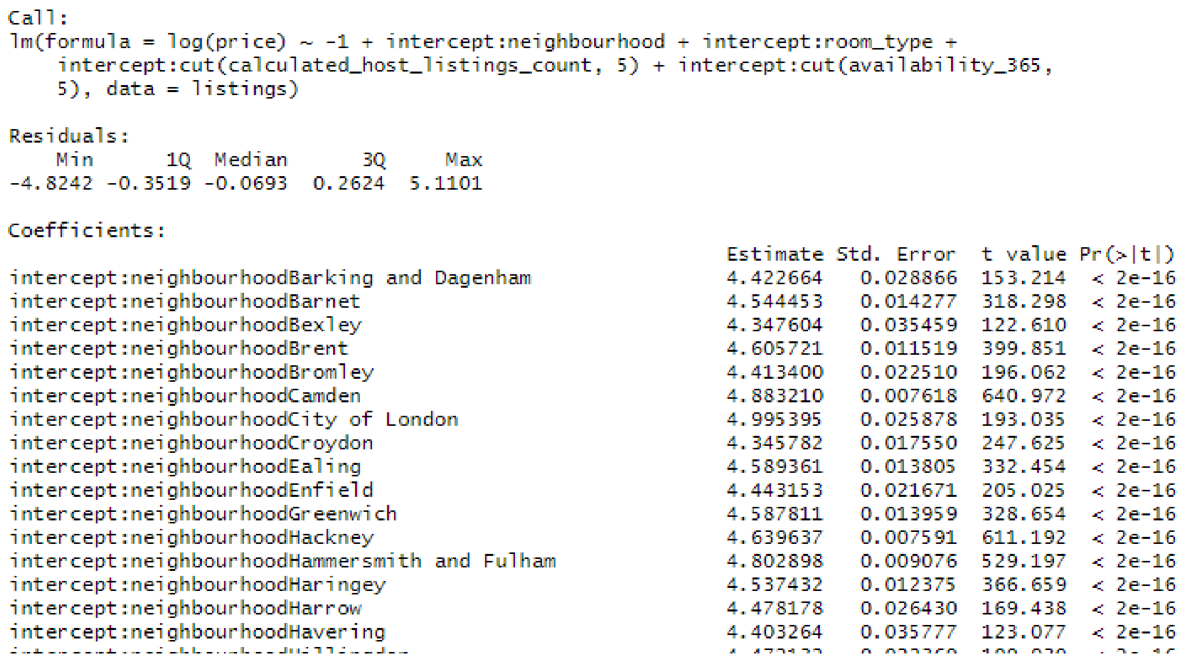
We see two parameters in this model:

- β0β0 or the intercept (29.75)
- β1β1 or the slope of price (12.35)

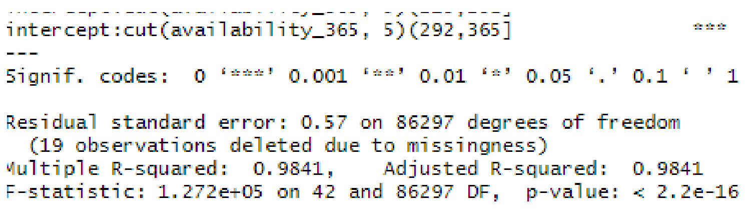
These parameters have the following interpretations. The intercept (β0β0) is the estimated price for an observation with price equal to zero. The slope (β1β1) is the estimated increase in price for every increase in price. This determines the steepness of the fitted regression line on the graph. So for a listing that has an overall satisfaction of, for example, 3.5, the estimated price is 29.75 + 3.5 ×× 12.35 = 72.98.

The slope is positive and significant. You could report this as follows: "There was a significantly positive relationship between price and overall satisfaction (ββ = 12.35, *t*(10585) = 6.63, *p* < .001)."
In the output, we also get information about the overall model. The model comes with an *F* value of 1.27 with 1 degree of freedom in the numerator and 10585 degrees of freedom in the denominator. This *F*-statistic tells us whether our model with one predictor (price) predicts the dependent variable (price) better than a model without predictors (which would simply predict the average of price for every level of overall satisfaction). The degrees of freedom allow us to find the corresponding *p*-value (< .001) of the *F*-statistic (1.272). The degrees of freedom in the numerator are equal to the number of predictors in our model. The degrees of freedom in the denominator are equal to the number of observations minus the number of predictors minus one. Remember that we have 10587 observations for which we have values for both price and price. The *p*-value is smaller than .05, so we reject the null hypothesis that our model does not predict the dependent variable better than a model without predictors. Note that in the case of simple linear regression, the *p*-value of the model corresponds to the *p*-value of the single predictor. For models with more predictors, there is no such correspondence.
Finally, also note the *R*-squared statistic of the model. This statistic is equal to 1.27. This statistic tells us how much of the variance in the dependent variable

is explained by our predictors. The more predictors you add to a model, the higher the *R*-squared will be.



```
Call:
lm(formula = log(price) ~ -1 + intercept:neighbourhood + intercept:room_type +
    intercept:cut(calculated_host_listings_count, 5) + intercept:cut(availability_365,
    5), data = listings)

Residuals:
    Min      1Q  Median      3Q     Max
-4.8242 -0.3519 -0.0693  0.2624  5.1101

Coefficients:
                                                     Estimate Std. Error  t value Pr(>|t|)
intercept:neighbourhoodBarking and Dagenham          4.422664   0.028866  153.214  < 2e-16
intercept:neighbourhoodBarnet                        4.544453   0.014277  318.298  < 2e-16
intercept:neighbourhoodBexley                        4.347604   0.035459  122.610  < 2e-16
intercept:neighbourhoodBrent                         4.605721   0.011519  399.851  < 2e-16
intercept:neighbourhoodBromley                       4.413400   0.022510  196.062  < 2e-16
intercept:neighbourhoodCamden                        4.883210   0.007618  640.972  < 2e-16
intercept:neighbourhoodCity of London                4.995395   0.025878  193.035  < 2e-16
intercept:neighbourhoodCroydon                       4.345782   0.017550  247.625  < 2e-16
intercept:neighbourhoodEaling                        4.589361   0.013805  332.454  < 2e-16
intercept:neighbourhoodEnfield                       4.443153   0.021671  205.025  < 2e-16
intercept:neighbourhoodGreenwich                     4.587811   0.013959  328.654  < 2e-16
intercept:neighbourhoodHackney                       4.639637   0.007591  611.192  < 2e-16
intercept:neighbourhoodHammersmith and Fulham        4.802898   0.009076  529.197  < 2e-16
intercept:neighbourhoodHaringey                      4.537432   0.012375  366.659  < 2e-16
intercept:neighbourhoodHarrow                        4.478178   0.026430  169.438  < 2e-16
intercept:neighbourhoodHavering                      4.403264   0.035777  123.077  < 2e-16
```



```
intercept:cut(availability_365, 5)(292,365]           ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.57 on 86297 degrees of freedom
  (19 observations deleted due to missingness)
Multiple R-squared:  0.9841,    Adjusted R-squared:  0.9841
F-statistic: 1.272e+05 on 42 and 86297 DF,  p-value: < 2.2e-16
```

Conclusion:

The exploratory analysis above highlights some interesting trends and patterns, as well as some factors that can increase an Airbnb listing’s price.

To sum up, in the framework, I analysed Airbnb listings from London, extracted from www.insideairbnb.com. Objective was to implement Linear or generalized linear models, so as to approximate prices of listings conditional to explanatory variables which would provide the optimal equilibrium between prices and locations and reviews by area.

And four centralities are created by taking the central location so, as to find the which location has more reviews and which has less reviews. and the listings having less reviews is affecting the price. and we can suggest the best match for user for booking in good place.

Towards this orientation, I cleaned and formulated the data so they would be apt for data analysis, next step was to plot the data. plotting enables us to distinguish relationships between reviews and location. and number of listings by location and compared by price range. the project was mostly focused on the price changes depending on the locations and is reviews affecting the prices. The procedure was conducted using linear algorithm transformation for the response variable, so as final model is the most efficient, unbiased and consistent.

Future Work:

- Use better quality/more accurate data which includes the actual average prices paid per night.
- Include a wider geographic area, e.g. the rest of the UK or other major cities around the world.
- Augment the model with natural language processing (NLP) of listing descriptions and/or reviews, e.g. for sentiment analysis or looking for keywords.
- Tailor the model more specifically to new listings in order to help hosts set prices for new properties, by removing features that would not be known at the time — e.g. other fees, availability and reviews.
- In addition to predicting base prices, a sequence model could be created to calculate daily rates using data on seasonality and occupancy, which would allow the creation of actual pricing software.