R Notebook

Step II: Prediction

I.

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.0     ✔ readr     2.1.4
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ ggplot2   3.4.1     ✔ tibble    3.2.1
## ✔ lubridate 1.9.2     ✔ tidyr     1.3.0
## ✔ purrr     1.0.1
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(forecast)
```

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

```
hk1 <- read.csv("hong_kong_cleaned.csv", header = TRUE)
```

A. Process description

Step 1: Data exploration and partitioning

Before partitioning, we will simply do some variable adjustment to facilitate the analysis to address potential issues we noticed that could end up in the training set and mess up our predictive model without us noticing:

Removing outliers

First we remove some outliers using the 99% quantile

```
library(tidyverse)
hk <- filter(hk1, hk1$price < quantile(hk1$price, 0.99))
hk <- filter(hk, hk$minimum_nights < quantile(hk1$minimum_nights, 0.99))
hk <- filter(hk, hk$beds < quantile(hk1$beds, 0.99))
hk <- filter(hk, hk$number_of_reviews < quantile(hk1$number_of_reviews, 0.99))
hk <- filter(hk, hk$host_listings_count < quantile(hk1$host_listings_count, 0.99))
hk <- filter(hk, hk$reviews_per_month < quantile(hk1$reviews_per_month, 0.99))
```

Adjusting some variables

First, let's adjust the *"bathroom_type"* variable.

```
hk <- hk %>%
  mutate(bathroom_type = ifelse(bathroom_type == "half-bath", "shared", bathroom_type))
```

We simply needed to adjust it because the half-bath has a unique value in the category "half-bath", which could cause overfitting issues and overall prediction issues. Since a half-bath usually mean that the shower is shared, we will assign it to the "private" category which it looks closer to.

Second, let's adjust the *"room_type"* variable:

```
hk <- hk %>%
  mutate(room_type = ifelse(room_type == "Hotel room", "Private room", room_type))
```

We noticed the same issue for this variable, which is why we will assign the “Hotel room” unique data point to “Private room” since it seems the most suitable.

Let’s now move on.

Before any further analysis, let’s partition the data.

```
set.seed(699)

train.index <- sample(c(1:nrow(hk)), nrow(hk)*0.6)
train.df <- hk[train.index, ]
valid.df <- hk[-train.index, ]
```

Step 2: Variable selection

First, let’s distinguish between numeric and categorical variables since they will not receive the exact same treatement.

Categorical variables:

```
categ_var <-  c("description","X","neighborhood_overview", "host_id", "host_since","host_location", "hos
t_response_time","host_is_superhost","host_neighbourhood","host_verifications","host_has_profile_pic",
"host_identity_verified", "property_type",
              "room_type", "bathroom_type", "amenities", "amenities_categ","has_availability",
              "instant_bookable","got_reviewed")
```

Firstly, let’s remove categorical variable with entirely unique values or redundant arbitrary values.

These variables don’t have any pattern, so we need to remove them from the start.

In particular, *“X”* appears to be an arbitrary index variable.

“host_id” also represents an arbitrary number, unique for each host, that could have been transformed to be indicative of the number of host listings but we already have this information in other variables, which is why it is not needed.

```
train.df <- train.df %>% select(-X,-host_id)
valid.df <- valid.df %>% select(-X,-host_id)
```

Secondly, let’s remove redundant, overly specific and irrelevant categorical variables.

```
categ_var2 <-  c("host_response_time","host_is_superhost","host_has_profile_pic", "host_identity_verifie
d",
              "room_type", "bathroom_type", "has_availability",
              "instant_bookable","got_reviewed")
```

“neighborhood_overview”: most rentals don’t have an overview. Moreover, by nature, it seems unique to a listing (higher counts than one might correspond to listings from the same host maybe). It is impossible to generalize since it is ultra specific textual data, that contains information that can be provided by other variables (e.g.: room type) and contains unnecessary details details on the Wai Chai neighborhood which is our only focus here anyways. It could not work in our MLR model so we will remove them.

- *“description”*: similar situation.
- *“host_verifications”*: does not seem to add much (level of detail seems unnecessarily specfic -> every host provides different means of contact that do not seem meaningful in the context of our model to predict pricing

Also, does not compare to the more generalized binary variable *“host_identity_verified”* that acts as a more official safety and trustworthiness proof (throught verifying government IDs, email addresses, and phone numbers).

The different verification types (email, phone, work email) don’t have an obvious direct relationship with price. In fact, it could be prone to overfitting by the model since the relationship is vague.It would also unnecessary noise and collinearity and would be redundant since other indicators of host reputation are kept.

- Remove *“host_has_profile_pic”* -> ? still hesitant about it.

Very few “No” with higher prices (see box plotthe count below show that this variable scam?

- Remove *“property_type”* and keep *“room_type”*

```
hk %>%
  group_by(hk$property_type) %>%
  summarize(count = n()) %>%
  arrange(desc(count))
```

hk$property_type <chr>	count <int>
Private room in rental unit	723
Entire rental unit	295
Shared room in rental unit	49
Entire condo	48
Entire serviced apartment	47
Private room in condo	30
Private room in serviced apartment	15
Private room in kezhan	11
Private room in home	10
Room in hotel	10

1-10 of 26 rows Previous **1** 2 3 Next

```
unique(train.df$property_type)
```

```
##  [1] "Entire rental unit"                 "Private room in rental unit"
##  [3] "Shared room in rental unit"         "Entire condo"
##  [5] "Entire serviced apartment"          "Private room"
##  [7] "Private room in serviced apartment" "Private room in bed and breakfast"
##  [9] "Private room in home"               "Private room in condo"
## [11] "Private room in kezhan"             "Shared room in serviced apartment"
## [13] "Entire home"                        "Shared room in condo"
## [15] "Entire guesthouse"                  "Private room in guest suite"
## [17] "Private room in loft"               "Room in hostel"
## [19] "Entire guest suite"                 "Room in hotel"
## [21] "Private room in guesthouse"
```

It has too many categories and room type keeps the main information from property type.

Therefore, we consider this information loss negligible compared to the issues the variable could cause, and *"room_type"* actually looks like it contains the appropriate amount of information without needing to bin it further.

- Keep *"bathroom_type"*

Since it is common for an accomodation - especially on Airbnb during shorter stays - to have a shared room and shared bathroom or a private room and a shared bathroom, the two do not correlate that much, and utilizing our domain knowledge, it is a popular option when renting in a shared house for example since it its usually less expensive than having a private bathroom. It is relevant to the model and we expect it to have a relationship with price, which is why we decide to keep it.

- Keep *"has_availability"*

```
train.df %>%
  group_by(train.df$has_availability) %>%
  summarize(count = n()) %>%
  arrange(desc(count))
```

train.df$has_availability <lgl>	count <int>
TRUE	731
FALSE	28

2 rows

- Keep *“instant_bookable”*

```
train.df %>%
  group_by(train.df$instant_bookable) %>%
  summarize(count = n()) %>%
  arrange(desc(count))
```

train.df$instant_bookable	count
<lgl>	<int>
FALSE	645
TRUE	114

2 rows

- Keep *“got_reviewed”*

```
train.df %>%
  group_by(train.df$got_reviewed) %>%
  summarize(count = n()) %>%
  arrange(desc(count))
```

train.df$got_reviewed	count
<int>	<int>
0	528
1	231

2 rows

We expect this variable to likely have a strong relationship, a high correlation with the three numeric variables *“number_of_reviews”*, *reviews_ltm* and *“reviews_l30d”*, because it the number of reviews being 0 would correspond to its outcome “FALSE”, and any other number would correspond to its outcome “TRUE”, which is why we will only keep *“got_reviewed”* among them. Since it is a binary classification, it loses nuance/information on the demand/popularity, but this information is already available in the *“availability_”* variables.

- Remove *“host_location”* and *“host_neighborhood”*

The model is only meant to focus on Hong Kong listings from the Wan Chai neighborhood, so the location variation is not very relevant and would likely have minimal impact on price variations (compared to more specific location data within the neighborhood).

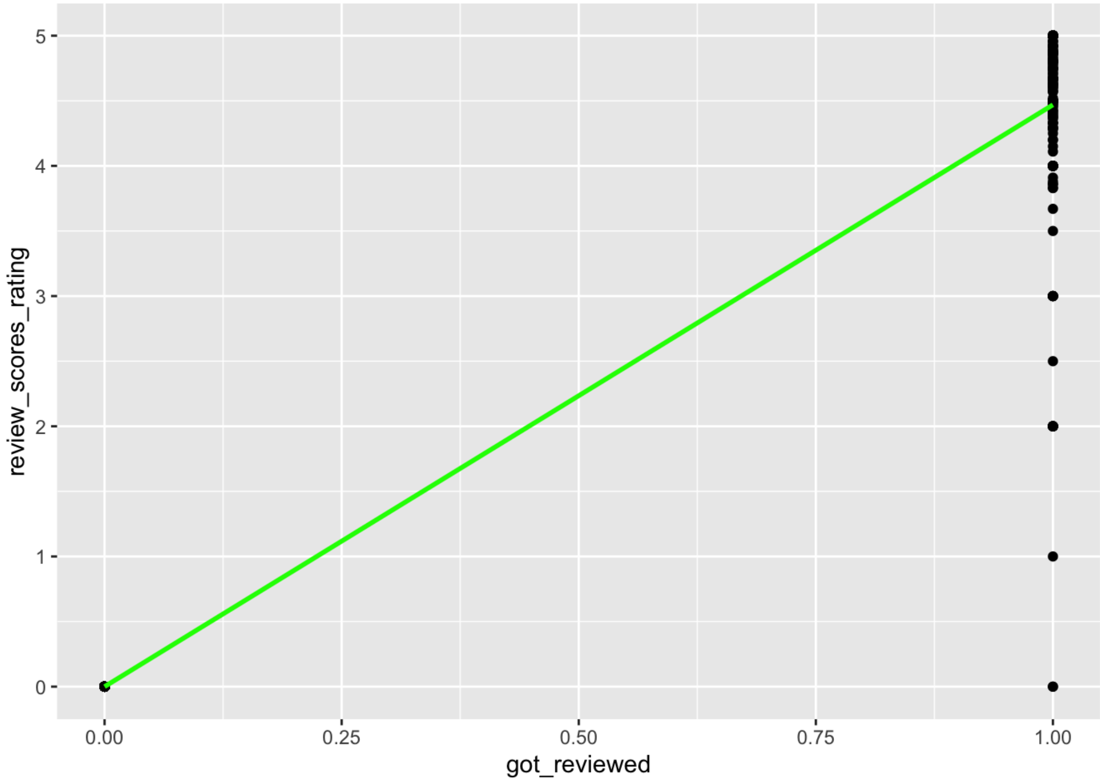
For *“host_location”*, the majority is Hong Kong, with some very few specific abroad locations. Even if we binned it into two outcome classes “Hong Kong” and “abroad”, there would still be a big risk of overfitting since the model would likely not be able to detect pattern to generalize with the limited information available in the “abroad” class. To avoid any risk for a variable that logically does not seem to closely correlate to pricing, we also will remove it.

- Remove *“got_reviewed”*:

As shown below, it seems that this variable is likely to be highly correlated with the numeric review scores variables, that account in most cases for no review by assigning a 0 rating, and all other ratings correspond to a rating as shown below while giving more useful detail on the perceived value of the listing based on the customer satisfaction.

```
ggplot(train.df, aes(x=got_reviewed, y=review_scores_rating)) + geom_point() + geom_smooth(method="lm",
se=FALSE, color="green")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



- Remove *“amenities”

We would have to dive it into too many categories and it is exploited in the naive bayes model later on.

Numeric variables

```
num_var <- c("host_response_rate", "host_acceptance_rate","host_listings_count",
             "host_total_listings_count",
"latitude", "longitude", "accommodates", "bathroom_nb", "bedrooms", "beds",
"minimum_nights", "maximum_nights", "availability_30", "availability_60", "availability_90",
"availability_365", "number_of_reviews", "number_of_reviews_ltm", "number_of_reviews_l30d",
"review_scores_rating", "review_scores_accuracy", "review_scores_cleanliness",
"review_scores_checkin", "review_scores_communication", "review_scores_location",
"review_scores_value", "calculated_host_listings_count_entire_homes",
"calculated_host_listings_count_private_rooms", "calculated_host_listings_count_shared_rooms",
"reviews_per_month")
```

Remove highly correlated numeric variables

Some variables, by definition and mathemathically, appear to be highly correlated.

In order to avoid a potential multicolinearity issue and to make sure not to forget to remove a variable from a very high correlation pair, I built a correlation table in R that depicts the correlations among all of the numerical variables that we might use as predictors.

```
cm<- cor(hk[num_var])
```

Identify correlations higher or equal than 0.65 (our selected treshold since there are a lot of variables), while not including correlations equal to one corresponding to the perfect correlation of a variable with itself

```
inds <- which(cm >= 0.65 & cm < 1, arr.ind = TRUE)

# Extract variable names
vars <- rownames(cm)

# Create dataframe with pairs respecting the correlation constraint
pairs <- data.frame(var1 = vars[inds[,1]],
                    var2 = vars[inds[,2]],
                    corr = cm[inds])

# Remove duplicate rows
pairs <- pairs[!duplicated(t(apply(pairs, 1, sort))),]

print(pairs[order(-pairs$corr),])
```

```
##                                           var1                        var2
## 22                      review_scores_accuracy        review_scores_rating
## 50                      review_scores_location review_scores_communication
## 43                 review_scores_communication       review_scores_checkin
## 27                         review_scores_value        review_scores_rating
## 23                   review_scores_cleanliness        review_scores_rating
## 29                   review_scores_cleanliness      review_scores_accuracy
## 44                      review_scores_location       review_scores_checkin
## 30                       review_scores_checkin      review_scores_accuracy
## 39                         review_scores_value   review_scores_cleanliness
## 33                         review_scores_value      review_scores_accuracy
## 24                       review_scores_checkin        review_scores_rating
## 25                 review_scores_communication        review_scores_rating
## 31                 review_scores_communication      review_scores_accuracy
## 26                      review_scores_location        review_scores_rating
## 57                         review_scores_value      review_scores_location
## 32                      review_scores_location      review_scores_accuracy
## 51                         review_scores_value review_scores_communication
## 45                         review_scores_value       review_scores_checkin
## 36                       review_scores_checkin   review_scores_cleanliness
## 37                 review_scores_communication   review_scores_cleanliness
## 38                      review_scores_location   review_scores_cleanliness
## 13                             availability_90             availability_60
## 1                    host_total_listings_count         host_listings_count
## 9                              availability_60             availability_30
## 2  calculated_host_listings_count_private_rooms         host_listings_count
## 10                             availability_90             availability_30
## 6                                         beds                    bedrooms
## 4  calculated_host_listings_count_private_rooms   host_total_listings_count
## 17                            availability_365             availability_90
## 14                            availability_365             availability_60
## 11                            availability_365             availability_30
## 21                           reviews_per_month       number_of_reviews_ltm
## 5                                         beds                accommodates
##         corr
## 22 0.9962765
## 50 0.9958293
## 43 0.9948129
## 27 0.9936544
## 23 0.9933421
## 29 0.9929259
## 44 0.9922291
## 30 0.9912574
## 39 0.9912273
## 33 0.9911551
## 24 0.9908994
## 25 0.9907638
## 31 0.9903485
## 26 0.9889134
## 57 0.9887867
## 32 0.9883201
## 51 0.9881961
## 45 0.9872488
## 36 0.9863205
## 37 0.9851732
## 38 0.9838184
## 13 0.9768761
## 1  0.9698732
## 9  0.9656859
## 2  0.9594477
## 10 0.9301947
## 6  0.8908878
## 4  0.8890440
## 17 0.7873846
## 14 0.7388964
## 11 0.7094833
```

```
## 21 0.6821257
## 5  0.6656825
```

Unsurprisingly, a lot of variables are redundant and we think it is likely to assume they would cause multicolinearity issues.

- Remove all *“review_scores”* variables

“review_scores_rating”, “review_scores_accuracy”, “review_scores_cleanliness”, “review_scores_checkin”, “review_scores_communication”, “review_scores_location” and “review_scores_value”

Based on Aribnb ratings, similar to Uber -> 5 stars are very frequent, and a lot of customers rate five stars for all categories when they are satisfied. the categorical variable “*got_reviewed*” is more relevant here, as explained further below.

- Similarly, remove all *“number_of_reviews_”* variables

Similarly and additionally for *“number_of_reviews”, “number_of_reviews_ltm” and “number_of_reviews_l30d”*, which have relatively high correlation coefficients with the *“review_score_”* variables.

We can argue that is is also redundant in the sense that it’s another indicator of occupancy/demand signals like the *“availaility_”* variables.

- *“availability_30”, “availability_60”, “availability_90” and “availability_365”*: we will only keep *“availability_365”*
- Only keep *“host_listings_count”*

Among
host_listings_count,host_total_listings_count,calculated_host_listings_count_entire_homes,calculated_host_listings_count_shared_rooms and calculated_host_listings_count_private_rooms

We will only keep the variable *“host_listings_count”*, since it provides an overall grouping and the detailed subdivision counts (variables starting by “calculated_host_listings_count_”) doesn’t seem relevant for our model.

- Keep *“accomodates”* and *“bedrooms”*, remove *“beds”*

Since “beds” and “bedrooms” have a high correlation coefficient according to the correlation table, and bedrooms has a lower correlation with accomodates (which doesn’t have a high correlation with any other variable), we decided to only remove the most correlated variable: “beds” to avoid multicolinearity.

Logarithmic transformation of the response variable price

Since we have identified in the data cleaning step that the price variable had a significant number of outliers, let’s also try creating a model using log(price) to see if it helps mitigate the impact of these extreme outliers that can skew the results of our model. The logarithmic transformation should be able to shrink the effect of extreme values, making the regression model more robust.

```
train.df$log_price = log(train.df$price)
valid.df$log_price = log(valid.df$price)
```

This will also be needed for visualization purposes since otherwise, the distribution is unreadable since the plots appear completely “skewed” or “squeezed” towards the bottom due to the presence of outliers in the price data.

Remove variables with less significant predictive power in relation to the variable price

Visualizations to further assess relevancy

To further assess relevancy, we will make visualizations for both categorical and numeric variables not excluded from the model yet and look into their relationship with the price variable.

Categorical variables

- Remove *“host_since”*

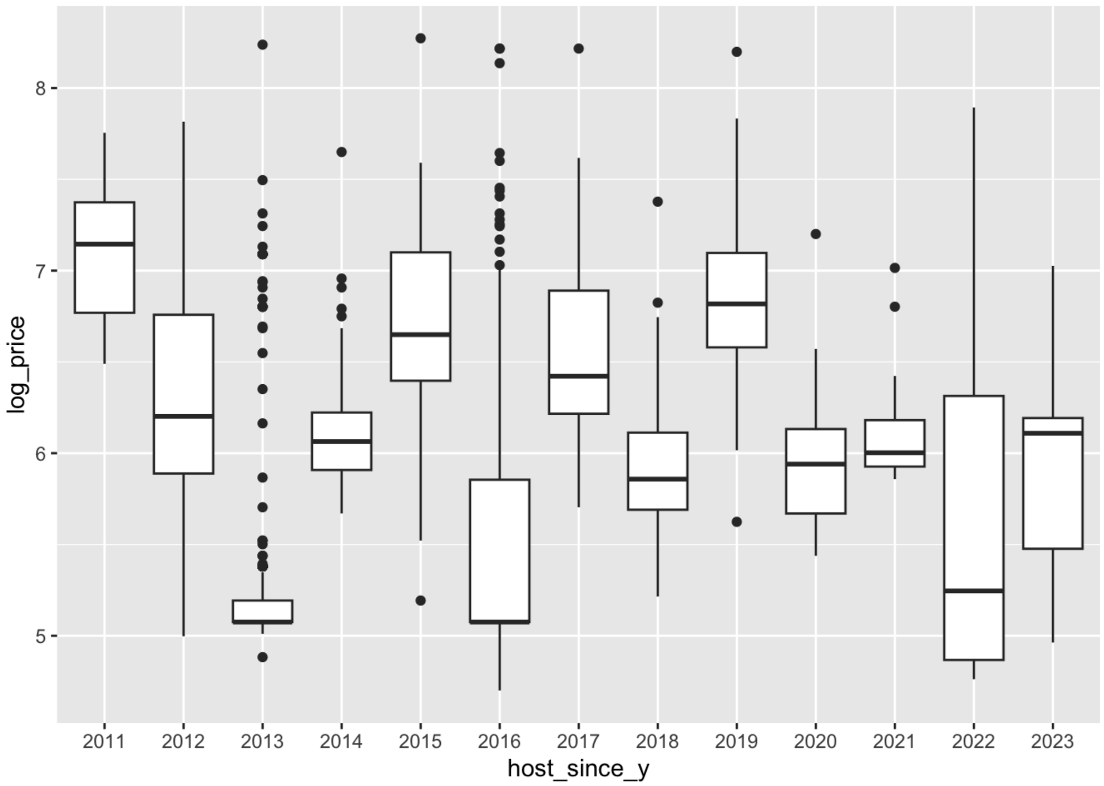
“host_since” has a unique specific date value for each host, which would not be viable in the model by itself. We considered converting it to proper dates format and only extracting the year, however, we assume that the host experience in terms of years, which most likely correspond to the date of registration as an Airbnb host, is less relevant than the Superhost status for example when accounting for the experience, recognition and reputation of a host. This is why to avoid a clear risk of overfitting and to avoid adding a less relevant variable that does not show a clear relationship with price, we chose to remove host_since. When we look at the box plots, it seems that even collapsing the levels or grouping them would not be relevant as a predictor for price.

```
library(lubridate)
```

```
train.df$host_since_y<-ymd(train.df$host_since)
```

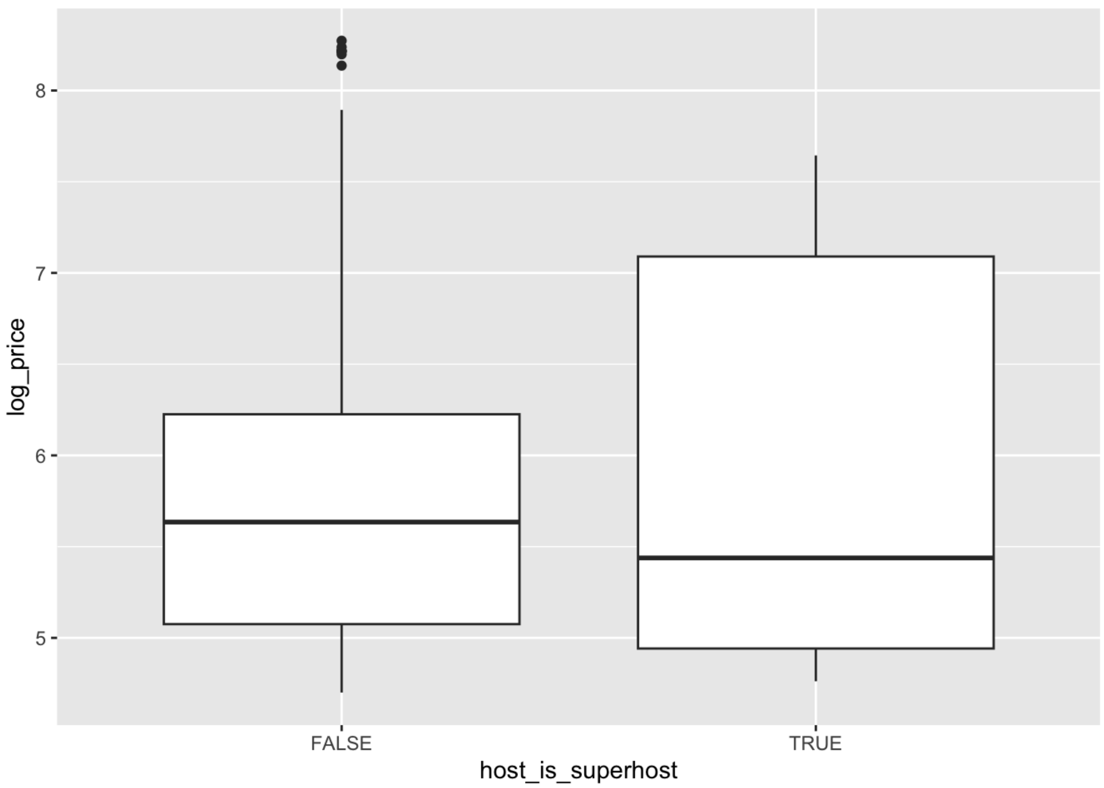
```
train.df <- train.df %>%
  mutate(host_since_y = factor(year(host_since)))
```

```
ggplot(train.df, aes(x = host_since_y, y = log_price)) +
  geom_boxplot()
```

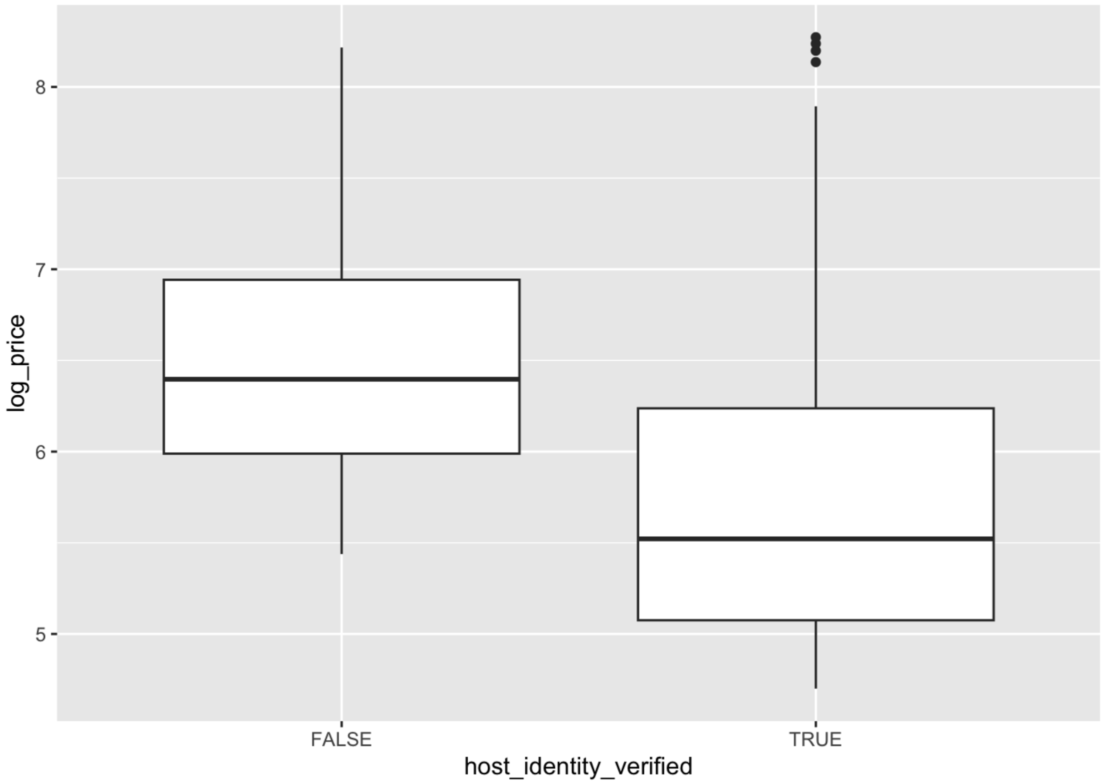



```
train.df <- train.df %>% select(-host_since_y)
```

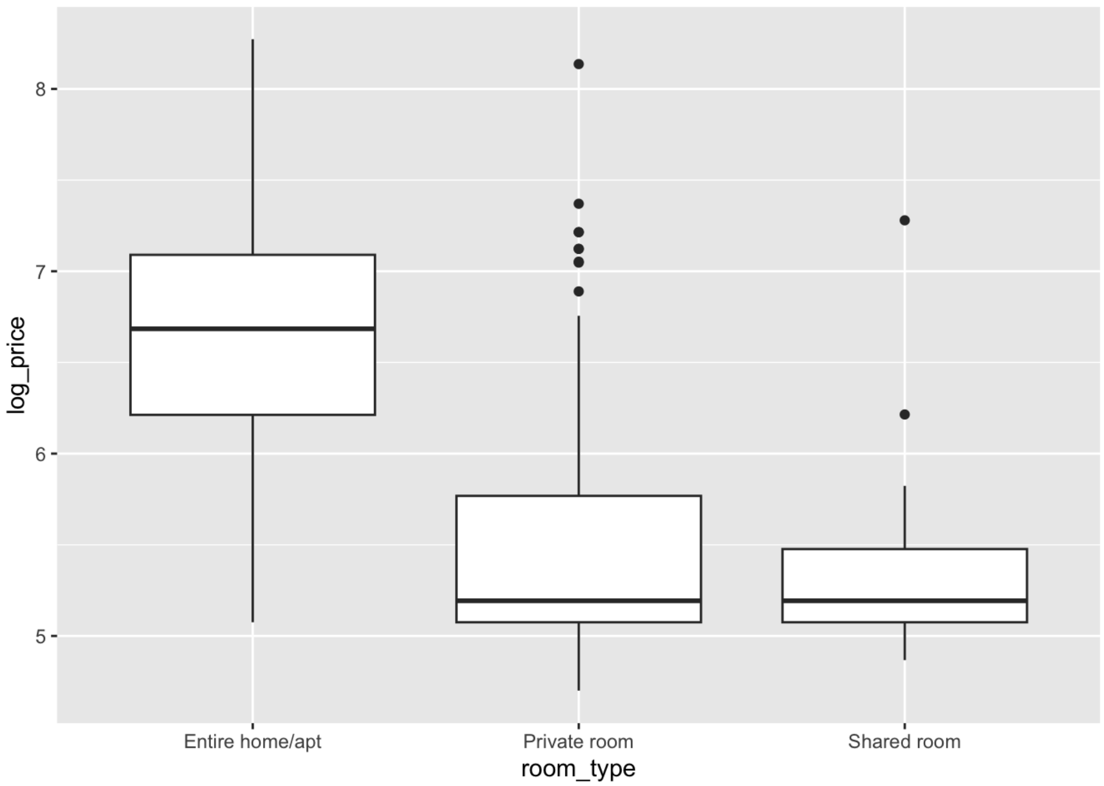
```
ggplot(train.df, aes(x = host_is_superhost, y = log_price)) +
  geom_boxplot()
```



```
ggplot(train.df, aes(x = host_identity_verified, y = log_price)) +
  geom_boxplot()
```




```
ggplot(train.df, aes(x = room_type, y = log_price)) +
  geom_boxplot()
```

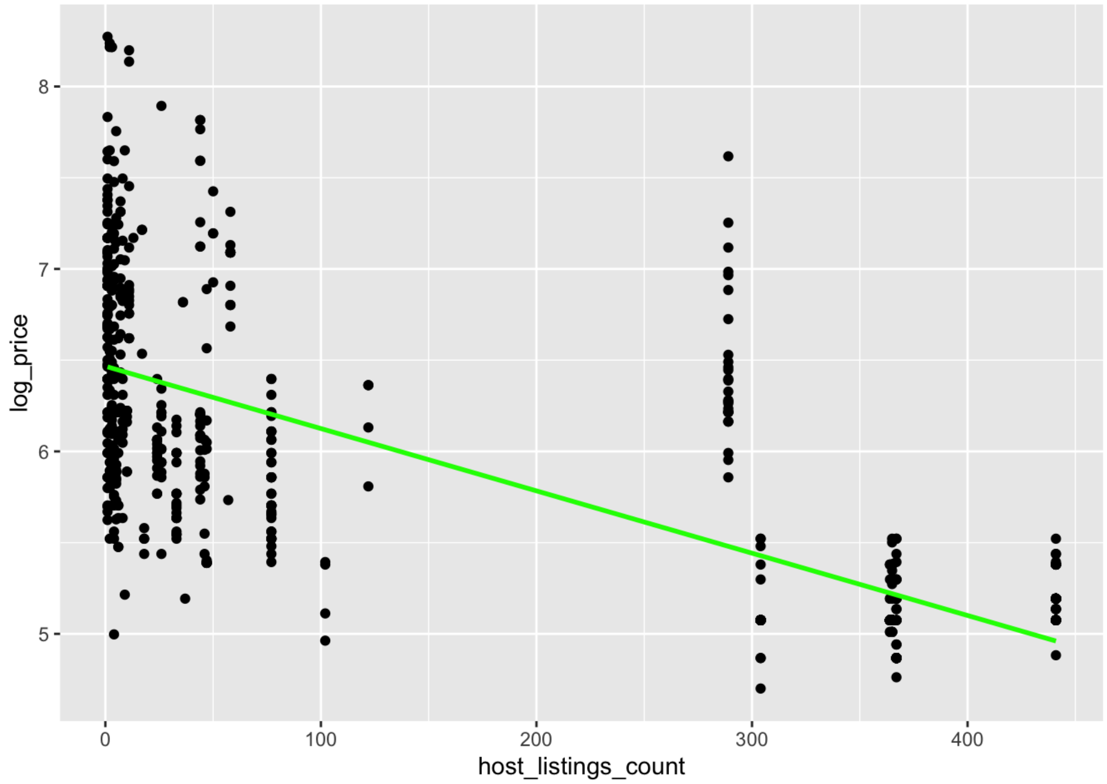


We can clearly see the price is influenced by those categories. We should keep them, we will try a model that can remove the unnecessary ones later

Numeric variables

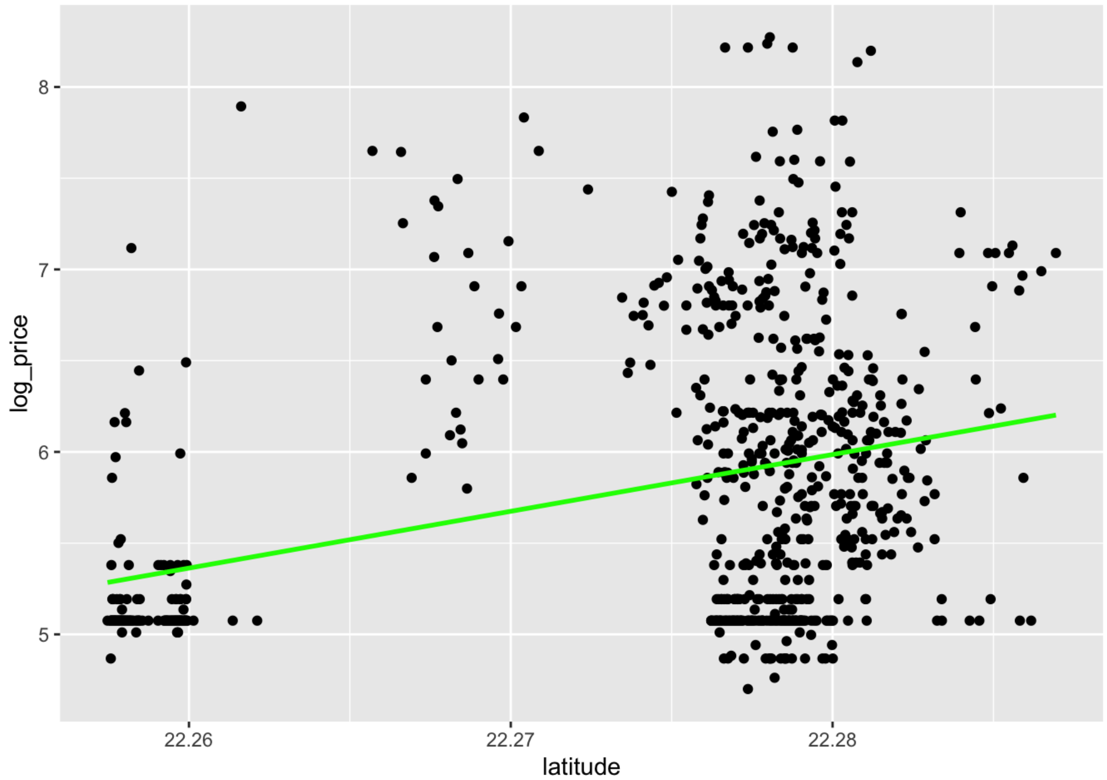
```
ggplot(train.df, aes(x=host_listings_count, y=log_price)) + geom_point() + geom_smooth(method="lm", se=F
ALSE, color="green")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



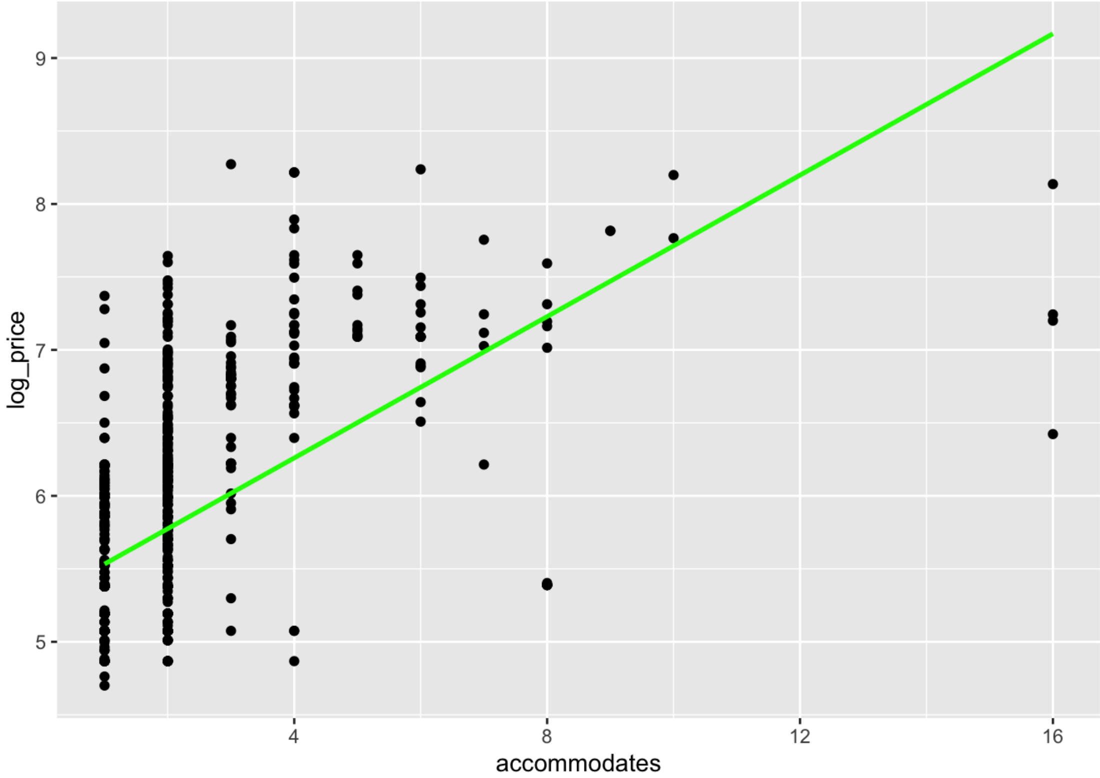

```
ggplot(train.df, aes(x=latitude, y=log_price)) + geom_point() + geom_smooth(method="lm", se=FALSE, color
="green")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
ggplot(train.df, aes(x=accommodates, y=log_price)) + geom_point() + geom_smooth(method="lm", se=FALSE, c
olor="green")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Step 3: Data preparation

```
sapply(train.df[categ_var2], is.factor)
```

```
##     host_response_time      host_is_superhost   host_has_profile_pic
##                  FALSE                  FALSE                  FALSE
## host_identity_verified              room_type          bathroom_type
##                  FALSE                  FALSE                  FALSE
##       has_availability       instant_bookable           got_reviewed
##                  FALSE                  FALSE                  FALSE
```

```
train.df[categ_var] <- lapply(train.df[categ_var2], as.factor)
valid.df[categ_var] <- lapply(valid.df[categ_var2], as.factor)
```

```
sapply(train.df[categ_var2], is.factor)
```

```
##     host_response_time      host_is_superhost   host_has_profile_pic
##                   TRUE                   TRUE                   TRUE
## host_identity_verified              room_type          bathroom_type
##                   TRUE                   TRUE                   TRUE
##       has_availability       instant_bookable           got_reviewed
##                   TRUE                   TRUE                   TRUE
```

Let's now clean up our training and validation sets, by removing the eliminated variables from this step, additionally to "X" and "host_id".

```
train.df <- train.df %>% select( -description,-neighborhood_overview,-host_since,-host_neighbourhood,
                                 -host_location,-property_type,-amenities,-host_total_listings_count,
                                 -host_verifications,-availability_30,-availability_60,-availability_90,
                                 -number_of_reviews,-number_of_reviews_ltm,-number_of_reviews_l30d,
                                 -review_scores_rating,-review_scores_accuracy,
                                 -review_scores_cleanliness,-review_scores_checkin,
                                 -review_scores_communication,-review_scores_location,
                                 -review_scores_value,-calculated_host_listings_count_entire_homes,
                                 -calculated_host_listings_count_private_rooms,
                                 -calculated_host_listings_count_shared_rooms,-reviews_per_month,-beds,
-X,
                                 -log_price, -host_id, -amenities_categ)
```

```
valid.df <- valid.df %>% select( -description,-neighborhood_overview,-host_since,-host_neighbourhood,
                                 -host_location,-property_type,-amenities,-host_total_listings_count,
                                 -host_verifications,-availability_30,-availability_60,-availability_90,
                                 -number_of_reviews,-number_of_reviews_ltm,-number_of_reviews_l30d,
                                 -review_scores_rating,-review_scores_accuracy,
                                 -review_scores_cleanliness,-review_scores_checkin,
                                 -review_scores_communication,-review_scores_location,
                                 -review_scores_value,-calculated_host_listings_count_entire_homes,
                                 -calculated_host_listings_count_private_rooms,
                                 -calculated_host_listings_count_shared_rooms,-reviews_per_month,-beds,
-X,
                                 -log_price, -host_id, -amenities_categ)
```

Step 4: Model building

MLR Model with price as a response variable

Using backward elimination, build the mutliple linear regression model with the data in our training set, to predict the price variable.

```
library(stats)
```

B

```
mlr_model<-lm(price ~ ., data = train.df)
mlr_model.step <- step(mlr_model, direction = "backward")
```

```
## Start:  AIC=8897.91
## price ~ host_response_time + host_response_rate + host_acceptance_rate + 
##     host_is_superhost + host_listings_count + host_has_profile_pic + 
##     host_identity_verified + latitude + longitude + room_type + 
##     accommodates + bathroom_nb + bathroom_type + bedrooms + minimum_nights + 
##     maximum_nights + has_availability + availability_365 + instant_bookable + 
##     got_reviewed
## 
## 
## Step:  AIC=8897.91
## price ~ host_response_time + host_response_rate + host_acceptance_rate + 
##     host_is_superhost + host_listings_count + host_has_profile_pic + 
##     host_identity_verified + latitude + longitude + room_type + 
##     accommodates + bathroom_nb + bathroom_type + bedrooms + minimum_nights + 
##     maximum_nights + has_availability + availability_365 + instant_bookable
## 
##                          Df Sum of Sq       RSS    AIC
## - bathroom_type           2     51323  87972979 8894.3
## - host_acceptance_rate    1      4564  87926220 8895.9
## - host_has_profile_pic    1      6888  87928544 8896.0
## - longitude               1      7196  87928852 8896.0
## - latitude                1     25999  87947655 8896.1
## - host_response_rate      1     57056  87978711 8896.4
## - host_is_superhost       1     64422  87986077 8896.5
## - instant_bookable        3    666840  88588495 8897.6
## <none>                                 87921656 8897.9
## - availability_365        1    312478  88234134 8898.6
## - maximum_nights          1    423028  88344684 8899.5
## - has_availability        1    505648  88427304 8900.3
## - bedrooms                1    666358  88588014 8901.6
## - bathroom_nb             1    726979  88648635 8902.2
## - host_identity_verified  1    801428  88723083 8902.8
## - host_listings_count     1   1254591  89176246 8906.7
## - host_response_time      1   2316943  90238599 8915.6
## - minimum_nights          1   2334091  90255746 8915.8
## - accommodates            1   9774063  97695718 8975.9
## - room_type               2  13676074 101597730 9003.6
## 
## Step:  AIC=8894.35
## price ~ host_response_time + host_response_rate + host_acceptance_rate + 
##     host_is_superhost + host_listings_count + host_has_profile_pic + 
##     host_identity_verified + latitude + longitude + room_type + 
##     accommodates + bathroom_nb + bedrooms + minimum_nights + 
##     maximum_nights + has_availability + availability_365 + instant_bookable
## 
##                          Df Sum of Sq       RSS    AIC
## - longitude               1      3280  87976259 8892.4
## - host_acceptance_rate    1     10159  87983138 8892.4
## - latitude                1     29424  88002403 8892.6
## - host_has_profile_pic    1     34295  88007274 8892.6
## - host_is_superhost       1     55361  88028340 8892.8
## - host_response_rate      1     63775  88036754 8892.9
## - instant_bookable        3    657852  88630831 8894.0
## <none>                                 87972979 8894.3
## - availability_365        1    273459  88246439 8894.7
## - maximum_nights          1    440341  88413320 8896.1
## - has_availability        1    506008  88478987 8896.7
## - bedrooms                1    667422  88640401 8898.1
## - bathroom_nb             1    692018  88664997 8898.3
## - host_identity_verified  1    816084  88789063 8899.4
## - host_listings_count     1   1415318  89388297 8904.5
## - host_response_time      1   2318221  90291200 8912.1
## - minimum_nights          1   2320201  90293181 8912.1
## - accommodates            1   9858525  97831504 8973.0
## - room_type               2  20739165 108712144 9051.0
## 
## Step:  AIC=8892.38
```

```
## price ~ host_response_time + host_response_rate + host_acceptance_rate +
##     host_is_superhost + host_listings_count + host_has_profile_pic +
##     host_identity_verified + latitude + room_type + accommodates +
##     bathroom_nb + bedrooms + minimum_nights + maximum_nights +
##     has_availability + availability_365 + instant_bookable
##
##                          Df Sum of Sq       RSS    AIC
## - host_acceptance_rate    1      9927  87986186 8890.5
## - host_has_profile_pic    1     33610  88009870 8890.7
## - latitude                1     46865  88023124 8890.8
## - host_is_superhost       1     54535  88030794 8890.8
## - host_response_rate      1     63028  88039288 8890.9
## - instant_bookable        3    657408  88633667 8892.0
## <none>                               87976259 8892.4
## - availability_365        1    275849  88252109 8892.8
## - maximum_nights          1    438573  88414833 8894.2
## - has_availability        1    502777  88479036 8894.7
## - bedrooms                1    679103  88655362 8896.2
## - bathroom_nb             1    692328  88668588 8896.3
## - host_identity_verified  1    813363  88789622 8897.4
## - host_listings_count     1   1437615  89413875 8902.7
## - host_response_time      1   2315312  90291571 8910.1
## - minimum_nights          1   2320595  90296854 8910.1
## - accommodates            1   9861223  97837482 8971.0
## - room_type               2  20737461 108713720 9049.0
##
## Step:  AIC=8890.46
## price ~ host_response_time + host_response_rate + host_is_superhost +
##     host_listings_count + host_has_profile_pic + host_identity_verified +
##     latitude + room_type + accommodates + bathroom_nb + bedrooms +
##     minimum_nights + maximum_nights + has_availability + availability_365 +
##     instant_bookable
##
##                          Df Sum of Sq       RSS    AIC
## - host_has_profile_pic    1     32756  88018942 8888.7
## - host_is_superhost       1     63440  88049626 8889.0
## - latitude                1     67958  88054144 8889.0
## - host_response_rate      1     80086  88066273 8889.2
## - instant_bookable        3    654926  88641112 8890.1
## <none>                               87986186 8890.5
## - availability_365        1    272779  88258965 8890.8
## - maximum_nights          1    444764  88430950 8892.3
## - has_availability        1    509950  88496136 8892.8
## - bedrooms                1    690534  88676721 8894.4
## - bathroom_nb             1    694420  88680606 8894.4
## - host_identity_verified  1    822699  88808885 8895.5
## - host_listings_count     1   1595503  89581689 8902.1
## - host_response_time      1   2323313  90309499 8908.2
## - minimum_nights          1   2327499  90313685 8908.3
## - accommodates            1   9906644  97892830 8969.4
## - room_type               2  20967506 108953693 9048.7
##
## Step:  AIC=8888.75
## price ~ host_response_time + host_response_rate + host_is_superhost +
##     host_listings_count + host_identity_verified + latitude +
##     room_type + accommodates + bathroom_nb + bedrooms + minimum_nights +
##     maximum_nights + has_availability + availability_365 + instant_bookable
##
##                          Df Sum of Sq       RSS    AIC
## - host_response_rate      1     66542  88085483 8887.3
## - host_is_superhost       1     70938  88089880 8887.4
## - latitude                1     93869  88112811 8887.6
## - instant_bookable        3    694133  88713075 8888.7
## <none>                               88018942 8888.7
## - availability_365        1    283303  88302245 8889.2
## - has_availability        1    500105  88519047 8891.0
## - maximum_nights          1    500552  88519494 8891.0
```

```
## - bathroom_nb             1    682065  88701006 8892.6
## - bedrooms                1    698098  88717040 8892.7
## - host_identity_verified  1    821846  88840788 8893.8
## - host_listings_count     1   1567850  89586792 8900.1
## - minimum_nights          1   2326874  90345815 8906.5
## - host_response_time      1   2419767  90438708 8907.3
## - accommodates            1   9986149  98005091 8968.3
## - room_type               2  21465164 109484106 9050.4
##
## Step:  AIC=8887.32
## price ~ host_response_time + host_is_superhost + host_listings_count +
##     host_identity_verified + latitude + room_type + accommodates +
##     bathroom_nb + bedrooms + minimum_nights + maximum_nights +
##     has_availability + availability_365 + instant_bookable
##
##                          Df Sum of Sq       RSS    AIC
## - host_is_superhost       1     63710  88149194 8885.9
## - latitude                1     99308  88184792 8886.2
## <none>                                 88085483 8887.3
## - availability_365        1    280832  88366315 8887.7
## - has_availability        1    481040  88566523 8889.5
## - maximum_nights          1    510534  88596017 8889.7
## - bathroom_nb             1    684966  88770449 8891.2
## - bedrooms                1    700019  88785503 8891.3
## - host_identity_verified  1    766560  88852043 8891.9
## - host_listings_count     1   1511527  89597011 8898.2
## - instant_bookable        3   2344425  90429909 8901.3
## - minimum_nights          1   2348214  90433697 8905.3
## - host_response_time      1   2641208  90726692 8907.7
## - accommodates            1   9968216  98053699 8966.7
## - room_type               2  21620930 109706413 9049.9
##
## Step:  AIC=8885.87
## price ~ host_response_time + host_listings_count + host_identity_verified +
##     latitude + room_type + accommodates + bathroom_nb + bedrooms +
##     minimum_nights + maximum_nights + has_availability + availability_365 +
##     instant_bookable
##
##                          Df Sum of Sq       RSS    AIC
## - latitude                1     97560  88246753 8884.7
## <none>                                 88149194 8885.9
## - availability_365        1    257085  88406279 8886.1
## - maximum_nights          1    495440  88644634 8888.1
## - has_availability        1    505853  88655047 8888.2
## - bathroom_nb             1    681681  88830874 8889.7
## - bedrooms                1    707102  88856296 8889.9
## - host_identity_verified  1    742234  88891427 8890.2
## - host_listings_count     1   1457758  89606951 8896.3
## - instant_bookable        3   2429046  90578239 8900.5
## - minimum_nights          1   2306107  90455301 8903.5
## - host_response_time      1   2724757  90873951 8907.0
## - accommodates            1   9905221  98054415 8964.7
## - room_type               2  22476184 110625378 9054.3
##
## Step:  AIC=8884.71
## price ~ host_response_time + host_listings_count + host_identity_verified +
##     room_type + accommodates + bathroom_nb + bedrooms + minimum_nights +
##     maximum_nights + has_availability + availability_365 + instant_bookable
##
##                          Df Sum of Sq       RSS    AIC
## - availability_365        1    218699  88465452 8884.6
## <none>                                 88246753 8884.7
## - maximum_nights          1    465319  88712073 8886.7
## - has_availability        1    486452  88733205 8886.9
## - bathroom_nb             1    635114  88881867 8888.2
## - bedrooms                1    715915  88962669 8888.8
## - host_identity_verified  1    745727  88992480 8889.1
```

```
## - host_listings_count     1   1571611  89818365 8896.1
## - instant_bookable        3   2431771  90678524 8899.3
## - minimum_nights          1   2268395  90515148 8902.0
## - host_response_time      1   2662154  90908907 8905.3
## - accommodates            1   9870135  98116888 8963.2
## - room_type               2  22388870 110635623 9052.3
##
## Step:  AIC=8884.59
## price ~ host_response_time + host_listings_count + host_identity_verified +
##     room_type + accommodates + bathroom_nb + bedrooms + minimum_nights +
##     maximum_nights + has_availability + instant_bookable
##
##                          Df Sum of Sq       RSS    AIC
## <none>                                 88465452 8884.6
## - maximum_nights          1    468853  88934305 8886.6
## - has_availability        1    539203  89004655 8887.2
## - bathroom_nb             1    686879  89152331 8888.5
## - host_identity_verified  1    690722  89156175 8888.5
## - bedrooms                1    703736  89169188 8888.6
## - host_listings_count     1   1353231  89818683 8894.1
## - instant_bookable        3   2602378  91067830 8900.6
## - minimum_nights          1   2177184  90642636 8901.0
## - host_response_time      1   2443585  90909037 8903.3
## - accommodates            1  10068267  98533719 8964.4
## - room_type               2  22170996 110636448 9050.3
```

Summary of the MLR model with price

```
summary(mlr_model.step)
```

```
##
## Call:
## lm(formula = price ~ host_response_time + host_listings_count +
##     host_identity_verified + room_type + accommodates + bathroom_nb +
##     bedrooms + minimum_nights + maximum_nights + has_availability +
##     instant_bookable, data = train.df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1545.80  -129.43   -16.57    79.10  3001.45
##
## Coefficients:
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      1037.49661  167.39047   6.198 9.47e-10 ***
## host_response_timeTRUE           -333.79632   73.63227  -4.533 6.77e-06 ***
## host_listings_count                -0.48330    0.14326  -3.374 0.000781 ***
## host_identity_verifiedTRUE        322.01618  133.60610   2.410 0.016186 *
## room_typePrivate room            -415.73633   34.36134 -12.099  < 2e-16 ***
## room_typeShared room             -691.84693   69.93191  -9.893  < 2e-16 ***
## accommodates                       92.17829   10.01732   9.202  < 2e-16 ***
## bathroom_nb                        87.21490   36.28700   2.403 0.016484 *
## bedrooms                           52.81967   21.71157   2.433 0.015218 *
## minimum_nights                     -4.30617    1.00634  -4.279 2.12e-05 ***
## maximum_nights                      0.07114    0.03583   1.986 0.047431 *
## has_availability1                 -73.00139   34.28115  -2.129 0.033541 *
## instant_bookablewithin a day     -454.54156  114.78286  -3.960 8.21e-05 ***
## instant_bookablewithin a few hours -487.32971  104.66800  -4.656 3.82e-06 ***
## instant_bookablewithin an hour   -439.55404  103.01766  -4.267 2.24e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 344.8 on 744 degrees of freedom
## Multiple R-squared:  0.5896, Adjusted R-squared:  0.5819
## F-statistic: 76.36 on 14 and 744 DF,  p-value: < 2.2e-16
```

Now let us try with log it could improve the model we've seen previously that on visualizations the relationship is more linear with log price.

MLR Model with log(price) as a response variable

```
log_mlr_model<-lm(log(price) ~ ., data = train.df)
log_mlr_model.step <- step(log_mlr_model, direction = "backward")
```

```
## Start:  AIC=-1485.31
## log(price) ~ host_response_time + host_response_rate + host_acceptance_rate +
##     host_is_superhost + host_listings_count + host_has_profile_pic +
##     host_identity_verified + latitude + longitude + room_type +
##     accommodates + bathroom_nb + bathroom_type + bedrooms + minimum_nights +
##     maximum_nights + has_availability + availability_365 + instant_bookable +
##     got_reviewed
##
##
## Step:  AIC=-1485.31
## log(price) ~ host_response_time + host_response_rate + host_acceptance_rate +
##     host_is_superhost + host_listings_count + host_has_profile_pic +
##     host_identity_verified + latitude + longitude + room_type +
##     accommodates + bathroom_nb + bathroom_type + bedrooms + minimum_nights +
##     maximum_nights + has_availability + availability_365 + instant_bookable
##
##                          Df Sum of Sq    RSS     AIC
## - bathroom_nb             1     0.008 100.67 -1487.3
## - latitude                1     0.016 100.68 -1487.2
## - host_has_profile_pic    1     0.020 100.69 -1487.2
## - host_is_superhost       1     0.040 100.71 -1487.0
## - bathroom_type           2     0.423 101.09 -1486.1
## - bedrooms                1     0.170 100.84 -1486.0
## - host_acceptance_rate    1     0.194 100.86 -1485.8
## - host_response_rate      1     0.217 100.89 -1485.7
## - has_availability        1     0.242 100.91 -1485.5
## - longitude               1     0.251 100.92 -1485.4
## <none>                                100.67 -1485.3
## - maximum_nights          1     0.448 101.12 -1483.9
## - availability_365        1     0.644 101.31 -1482.5
## - host_identity_verified  1     0.828 101.50 -1481.1
## - instant_bookable        3     1.577 102.24 -1479.5
## - host_response_time      1     1.236 101.90 -1478.0
## - minimum_nights          1     4.115 104.78 -1456.9
## - accommodates            1     9.428 110.10 -1419.4
## - host_listings_count     1    12.652 113.32 -1397.5
## - room_type               2    35.015 135.68 -1262.8
##
## Step:  AIC=-1487.26
## log(price) ~ host_response_time + host_response_rate + host_acceptance_rate +
##     host_is_superhost + host_listings_count + host_has_profile_pic +
##     host_identity_verified + latitude + longitude + room_type +
##     accommodates + bathroom_type + bedrooms + minimum_nights +
##     maximum_nights + has_availability + availability_365 + instant_bookable
##
##                          Df Sum of Sq    RSS     AIC
## - latitude                1     0.018 100.69 -1489.1
## - host_has_profile_pic    1     0.020 100.70 -1489.1
## - host_is_superhost       1     0.041 100.72 -1489.0
## - bedrooms                1     0.164 100.84 -1488.0
## - bathroom_type           2     0.439 101.11 -1488.0
## - host_acceptance_rate    1     0.192 100.87 -1487.8
## - host_response_rate      1     0.217 100.89 -1487.6
## - has_availability        1     0.246 100.92 -1487.4
## - longitude               1     0.251 100.93 -1487.4
## <none>                                100.67 -1487.3
## - maximum_nights          1     0.441 101.12 -1485.9
## - availability_365        1     0.642 101.32 -1484.4
## - host_identity_verified  1     0.823 101.50 -1483.1
## - instant_bookable        3     1.570 102.25 -1481.5
## - host_response_time      1     1.228 101.90 -1480.0
## - minimum_nights          1     4.111 104.79 -1458.9
## - accommodates            1     9.494 110.17 -1420.8
## - host_listings_count     1    12.657 113.33 -1399.4
## - room_type               2    36.029 136.70 -1259.1
##
## Step:  AIC=-1489.12
```

```
## log(price) ~ host_response_time + host_response_rate + host_acceptance_rate +
##     host_is_superhost + host_listings_count + host_has_profile_pic +
##     host_identity_verified + longitude + room_type + accommodates +
##     bathroom_type + bedrooms + minimum_nights + maximum_nights +
##     has_availability + availability_365 + instant_bookable
##
##                          Df Sum of Sq    RSS     AIC
## - host_has_profile_pic    1     0.015 100.71 -1491.0
## - host_is_superhost       1     0.038 100.73 -1490.8
## - bedrooms                1     0.159 100.85 -1489.9
## - bathroom_type           2     0.438 101.13 -1489.8
## - host_response_rate      1     0.218 100.91 -1489.5
## - host_acceptance_rate    1     0.243 100.94 -1489.3
## - has_availability        1     0.245 100.94 -1489.3
## <none>                                100.69 -1489.1
## - longitude               1     0.371 101.06 -1488.3
## - maximum_nights          1     0.426 101.12 -1487.9
## - availability_365        1     0.625 101.32 -1486.4
## - host_identity_verified  1     0.825 101.52 -1484.9
## - instant_bookable        3     1.702 102.39 -1482.4
## - host_response_time      1     1.220 101.91 -1482.0
## - minimum_nights          1     4.093 104.79 -1460.9
## - accommodates            1     9.507 110.20 -1422.6
## - host_listings_count     1    13.106 113.80 -1398.2
## - room_type               2    36.206 136.90 -1260.0
##
## Step:  AIC=-1491.01
## log(price) ~ host_response_time + host_response_rate + host_acceptance_rate +
##     host_is_superhost + host_listings_count + host_identity_verified +
##     longitude + room_type + accommodates + bathroom_type + bedrooms +
##     minimum_nights + maximum_nights + has_availability + availability_365 +
##     instant_bookable
##
##                          Df Sum of Sq    RSS     AIC
## - host_is_superhost       1     0.034 100.74 -1492.8
## - bedrooms                1     0.157 100.87 -1491.8
## - bathroom_type           2     0.437 101.14 -1491.7
## - host_response_rate      1     0.241 100.95 -1491.2
## - host_acceptance_rate    1     0.248 100.96 -1491.1
## - has_availability        1     0.250 100.96 -1491.1
## <none>                                100.71 -1491.0
## - longitude               1     0.361 101.07 -1490.3
## - maximum_nights          1     0.414 101.12 -1489.9
## - availability_365        1     0.612 101.32 -1488.4
## - host_identity_verified  1     0.829 101.54 -1486.8
## - instant_bookable        3     1.688 102.40 -1484.4
## - host_response_time      1     1.206 101.91 -1484.0
## - minimum_nights          1     4.091 104.80 -1462.8
## - accommodates            1     9.493 110.20 -1424.6
## - host_listings_count     1    13.473 114.18 -1397.7
## - room_type               2    38.024 138.73 -1251.9
##
## Step:  AIC=-1492.75
## log(price) ~ host_response_time + host_response_rate + host_acceptance_rate +
##     host_listings_count + host_identity_verified + longitude +
##     room_type + accommodates + bathroom_type + bedrooms + minimum_nights +
##     maximum_nights + has_availability + availability_365 + instant_bookable
##
##                          Df Sum of Sq    RSS     AIC
## - bedrooms                1     0.159 100.90 -1493.6
## - bathroom_type           2     0.427 101.17 -1493.5
## - host_response_rate      1     0.227 100.97 -1493.0
## - has_availability        1     0.260 101.00 -1492.8
## <none>                                100.74 -1492.8
## - host_acceptance_rate    1     0.284 101.03 -1492.6
## - longitude               1     0.350 101.09 -1492.1
## - maximum_nights          1     0.405 101.15 -1491.7
```

```
## - availability_365        1     0.589 101.33 -1490.3
## - host_identity_verified  1     0.808 101.55 -1488.7
## - instant_bookable        3     1.706 102.45 -1486.0
## - host_response_time      1     1.266 102.01 -1485.3
## - minimum_nights          1     4.058 104.80 -1464.8
## - accommodates            1     9.459 110.20 -1426.6
## - host_listings_count     1    13.719 114.46 -1397.8
## - room_type               2    38.558 139.30 -1250.8
##
## Step:  AIC=-1493.56
## log(price) ~ host_response_time + host_response_rate + host_acceptance_rate +
##     host_listings_count + host_identity_verified + longitude +
##     room_type + accommodates + bathroom_type + minimum_nights +
##     maximum_nights + has_availability + availability_365 + instant_bookable
##
##                          Df Sum of Sq    RSS     AIC
## - bathroom_type           2     0.424 101.33 -1494.4
## - host_response_rate      1     0.224 101.12 -1493.9
## <none>                                100.90 -1493.6
## - has_availability        1     0.295 101.20 -1493.3
## - host_acceptance_rate    1     0.307 101.21 -1493.2
## - longitude               1     0.383 101.28 -1492.7
## - maximum_nights          1     0.423 101.32 -1492.4
## - availability_365        1     0.587 101.49 -1491.2
## - host_identity_verified  1     0.797 101.70 -1489.6
## - host_response_time      1     1.280 102.18 -1486.0
## - instant_bookable        3     1.854 102.75 -1485.7
## - minimum_nights          1     4.171 105.07 -1464.8
## - host_listings_count     1    13.689 114.59 -1399.0
## - accommodates            1    15.778 116.68 -1385.3
## - room_type               2    38.407 139.31 -1252.7
##
## Step:  AIC=-1494.37
## log(price) ~ host_response_time + host_response_rate + host_acceptance_rate +
##     host_listings_count + host_identity_verified + longitude +
##     room_type + accommodates + minimum_nights + maximum_nights +
##     has_availability + availability_365 + instant_bookable
##
##                          Df Sum of Sq    RSS     AIC
## - host_response_rate      1     0.241 101.57 -1494.6
## <none>                                101.33 -1494.4
## - longitude               1     0.289 101.61 -1494.2
## - has_availability        1     0.292 101.62 -1494.2
## - host_acceptance_rate    1     0.408 101.73 -1493.3
## - availability_365        1     0.442 101.77 -1493.1
## - maximum_nights          1     0.464 101.79 -1492.9
## - host_identity_verified  1     0.824 102.15 -1490.2
## - host_response_time      1     1.294 102.62 -1486.7
## - instant_bookable        3     1.899 103.22 -1486.3
## - minimum_nights          1     3.992 105.32 -1467.0
## - host_listings_count     1    14.916 116.24 -1392.1
## - accommodates            1    16.101 117.43 -1384.4
## - room_type               2    57.530 158.85 -1157.1
##
## Step:  AIC=-1494.57
## log(price) ~ host_response_time + host_acceptance_rate + host_listings_count +
##     host_identity_verified + longitude + room_type + accommodates +
##     minimum_nights + maximum_nights + has_availability + availability_365 +
##     instant_bookable
##
##                          Df Sum of Sq    RSS     AIC
## - has_availability        1     0.256 101.82 -1494.7
## <none>                                101.57 -1494.6
## - longitude               1     0.270 101.84 -1494.5
## - availability_365        1     0.451 102.02 -1493.2
## - maximum_nights          1     0.479 102.05 -1493.0
## - host_acceptance_rate    1     0.603 102.17 -1492.1
```

```
## - host_identity_verified  1     0.700 102.27 -1491.4
## - host_response_time      1     1.584 103.15 -1484.8
## - instant_bookable        3     3.326 104.89 -1476.1
## - minimum_nights          1     4.050 105.62 -1466.9
## - host_listings_count     1    14.917 116.48 -1392.6
## - accommodates            1    16.179 117.75 -1384.4
## - room_type               2    57.615 159.18 -1157.5
##
## Step:  AIC=-1494.66
## log(price) ~ host_response_time + host_acceptance_rate + host_listings_count +
##     host_identity_verified + longitude + room_type + accommodates +
##     minimum_nights + maximum_nights + availability_365 + instant_bookable
##
##                          Df Sum of Sq    RSS     AIC
## - longitude               1     0.231 102.05 -1494.9
## <none>                                101.82 -1494.7
## - maximum_nights          1     0.437 102.26 -1493.4
## - availability_365        1     0.504 102.33 -1492.9
## - host_acceptance_rate    1     0.632 102.45 -1492.0
## - host_identity_verified  1     0.677 102.50 -1491.6
## - host_response_time      1     1.624 103.45 -1484.7
## - instant_bookable        3     3.538 105.36 -1474.7
## - minimum_nights          1     3.798 105.62 -1468.9
## - host_listings_count     1    14.715 116.54 -1394.2
## - accommodates            1    16.032 117.85 -1385.7
## - room_type               2    57.363 159.19 -1159.5
##
## Step:  AIC=-1494.94
## log(price) ~ host_response_time + host_acceptance_rate + host_listings_count +
##     host_identity_verified + room_type + accommodates + minimum_nights +
##     maximum_nights + availability_365 + instant_bookable
##
##                          Df Sum of Sq    RSS     AIC
## <none>                                102.05 -1494.9
## - maximum_nights          1     0.400 102.45 -1494.0
## - availability_365        1     0.486 102.54 -1493.3
## - host_identity_verified  1     0.658 102.71 -1492.1
## - host_acceptance_rate    1     0.730 102.78 -1491.5
## - host_response_time      1     1.537 103.59 -1485.6
## - instant_bookable        3     3.430 105.48 -1475.8
## - minimum_nights          1     3.801 105.85 -1469.2
## - accommodates            1    16.047 118.10 -1386.1
## - host_listings_count     1    16.259 118.31 -1384.7
## - room_type               2    57.177 159.23 -1161.3
```

```
summary(log_mlr_model.step)
```

```
##
## Call:
## lm(formula = log(price) ~ host_response_time + host_acceptance_rate +
##     host_listings_count + host_identity_verified + room_type +
##     accommodates + minimum_nights + maximum_nights + availability_365 +
##     instant_bookable, data = train.df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.47338 -0.18169 -0.02937  0.18063  1.73762
##
## Coefficients:
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                       6.925e+00  1.765e-01  39.228  < 2e-16 ***
## host_response_timeTRUE           -2.779e-01  8.295e-02  -3.350 0.000849 ***
## host_acceptance_rate             -1.603e-01  6.943e-02  -2.308 0.021258 *
## host_listings_count              -1.924e-03  1.766e-04 -10.894  < 2e-16 ***
## host_identity_verifiedTRUE        3.149e-01  1.437e-01   2.192 0.028713 *
## room_typePrivate room            -6.999e-01  3.679e-02 -19.023  < 2e-16 ***
## room_typeShared room             -9.898e-01  7.202e-02 -13.744  < 2e-16 ***
## accommodates                      9.679e-02  8.943e-03  10.824  < 2e-16 ***
## minimum_nights                   -5.538e-03  1.051e-03  -5.268 1.81e-07 ***
## maximum_nights                    6.535e-05  3.824e-05   1.709 0.087928 .
## availability_365                  2.513e-04  1.335e-04   1.883 0.060080 .
## instant_bookablewithin a day     -3.560e-01  1.238e-01  -2.875 0.004158 **
## instant_bookablewithin a few hours -4.646e-01  1.170e-01  -3.970 7.88e-05 ***
## instant_bookablewithin an hour   -3.029e-01  1.184e-01  -2.559 0.010693 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3701 on 745 degrees of freedom
## Multiple R-squared:  0.7875, Adjusted R-squared:  0.7838
## F-statistic: 212.4 on 13 and 745 DF,  p-value: < 2.2e-16
```

The model with log price has higher adjusted r-squared so we keep this last one.

Its coefficients are written in the estimates. The equation is the following:

$\hat{Y}$= a+ b1 x X1 + b2 x X2 +…+ bp x Xp

where $\hat{Y}$ is the prediction, a is the intercept and b1 to bp are the coefficients in the estimate and X1 to Xp are the inputs of the predictors.

C. Analysis of other model metrics

After the using the log price instead of price and backwards elimination our adjusted R squared improved by over 0.2 it is now 0.78.

This could be explained by the fact that the distribution of the price is very skewed which would not be surprising because we have many outliers as seen previously. It could also be explained by the fact that the price became more linear after the log transformation.

Let us test the model with the validation set now to evaluate its performance. We have to transform the price of the validation to log first.

```
valid.df$price <- sapply(valid.df$price, log)
library(rsq)
library(Metrics)
```

```
##
## Attaching package: 'Metrics'
```

```
## The following object is masked from 'package:forecast':
##
##     accuracy
```

```
predicted_values <- predict(log_mlr_model.step, newdata = valid.df)
sse <- sum((valid.df$price - predicted_values)^2)
sst <- sum((valid.df$price - mean(valid.df$price))^2)
r_sq <- 1 - (sse/sst)
r_sq
```

```
## [1] 0.7465671
```

We get a R_squared of 0.74 which is close to the traning data set. This means the model is not overfitted. Finally let us look at the RMSE and min and max residuals. We have to bring back the values without the log transformation to be able to interprete it.

```
valid.df$price <- sapply(valid.df$price,exp)
predicted_values <- sapply(predicted_values,exp)

RMSE <- rmse(valid.df$price, predicted_values)
Min_Residual <- min(valid.df$price-predicted_values)
Max_Residual <- max(valid.df$price-predicted_values)

RMSE
```

```
## [1] 388.1555
```

```
Min_Residual
```

```
## [1] -2000.209
```

```
Max_Residual
```

```
## [1] 2027.676
```

The RMSE is of 388, on average the models makes a mistake of 388 which is consiquent considering that most apartments have a price around 600.

The minimum residual is around -2000 and maximum residual 2000 this represents the highest mistake the model can make.

Based on those performance metrics we can conclude that the model is fair at predicting price but it still makes mistakes by a margin.