Report on Beijing House

Shide Qiu

24/05/2020

1. Introduction

This project is related to the **IDV Learners Project** of the *HarvardX: PH125.9x Data Science: Capstone course.* I will use a sample of houses which have been sold in Beijing, China from 2011 to 2017 in this project. I chose this dataset because it has a significant number of features which can be explored. Also, I am currently in the process of looking at houses to buy myself and this real dataset will help me understand how different features will influence the sale price, which will be helpful for me to make a decision in the future.

Here, I will train a machine learning algorithm using the inputs in the training set to predict the house price in the test set. To compare different models and to see how well we're doing compared to a baseline, we will use **root mean squared error (RMSE) as our lose function.** RMSE is one of the most popular measure of accuracy to compare forecasting errors of different models for a particular dataset. If N is the number of sold housed in Beijing, y_i is the price for house i, and $\hat{y}_i$ is our prediction, then RMSE is defined as follows:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i}(\hat{y}_i - y_i)^2}$$

The best model is the one with lowest RMSE and will be used to predict the house price.

1.1. Used Dataset

The Housing price in Beijing dataset is provided and downloaded from Kaggle:

- https://www.kaggle.com/ruiqurm/lianjia/download

1.2. Used Libraries

The following libraries will be used in this report:

```
if(!require(tidyverse)) install.packages("tidyverse")
if(!require(tidyverse)) install.packages("caret")
if(!require(tidyverse)) install.packages("lubridate")
if(!require(tidyverse)) install.packages("data.table")
if(!require(tidyverse)) install.packages("DataExplorer")
if(!require(tidyverse)) install.packages("corrplot")
if(!require(tidyverse)) install.packages("gridExtra")
if(!require(tidyverse)) install.packages("PerformanceAnalytics")
```

```
if(!require(tidyverse)) install.packages("ggthemes")
if(!require(tidyverse)) install.packages("psych")
if(!require(tidyverse)) install.packages("kernlab")
if(!require(tidyverse)) install.packages("e1071")
library(tidyverse)
library(caret)
library(lubridate)
library(data.table)
library(DataExplorer)
library(corrplot)
library(gridExtra)
library(PerformanceAnalytics)
library(ggthemes)
library(psych)
library(glmnet)
library(kernlab)
library(e1071)
```

2. Data Preparation

Concerning the features, we have a very complete description of them as shown below:

- *url*: the url which fetches the data
- *id*: the id of transaction
- *Lng*: and Lat coordinates, using the BD09 protocol.
- *Cid*: community id
- *tradeTime*: the time of transaction
- *DOM*: active days on market. Know more in https://en.wikipedia.org/wiki/Days_on_market
- *followers*: the number of people follow the transaction.
- *totalPrice*: the total price
- *price*: the average price by square
- *square*: the square of house
- *livingRoom*: the number of living rooms
- *drawingRoom*: the number of drawing rooms
- *kitchen*: the number of kitchen
- *bathroom* the number of bathroom
- *floor*: the height of the house. I will turn the Chinese characters to English in the next version.
- *buildingType*: including tower(1) , bungalow(2)， combination of plate and tower(3), plate(4).
- *constructionTime*: the time of construction
- *renovationCondition*: including other(1), rough(2),Simplicity(3), hardcover(4)

- *buildingStructure*: including unknow(1), mixed(2), brick and wood(3), brick and concrete(4),steel(5) and steel-concrete composite (6).
- *ladderRatio*: the proportion between number of residents on the same floor and number of elevator of ladder. It describes how many ladders a resident has on average.
- *elevator* have (1) or not have elevator(0)
- *fiveYearsProperty*: if the owner has the property for less than 5 years

2.1. Data Loading

```
house <- read_csv('beijing.csv')
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   url = col_character(),
##   tradeTime = col_date(format = ""),
##   floor = col_character(),
##   constructionTime = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
## Warning: 226701 parsing failures.
##   row col expected       actual          file
## 92217  id a double BJ0000614981 'beijing.csv'
## 92218  id a double BJ0000614985 'beijing.csv'
## 92219  id a double BJ0000614991 'beijing.csv'
## 92220  id a double BJ0000614992 'beijing.csv'
## 92221  id a double BJ0000614999 'beijing.csv'
## ..... ... ........ ............ ............
## See problems(...) for more details.
```

2.2. Data Cleaning

According to the features description, let's remove some unnecessary features including *url*, *id*, and *Cid*.

```
house <- house %>% mutate(year = as.numeric(format(house$tradeTime, '%Y')))
house <- house %>% mutate(month = as.numeric(format(house$tradeTime, '%m')))
house <- house %>% select(-c(url, id, Cid, tradeTime))
house <- house %>% mutate(constructionTime = as.numeric(constructionTime))
```

```
## Warning: NAs introduced by coercion
```

Let's look at some of the general properties of the data to better understand the challenge. We check the first several rows of the dataset. Each row represents a single house with certain features.

```
head(house)
```

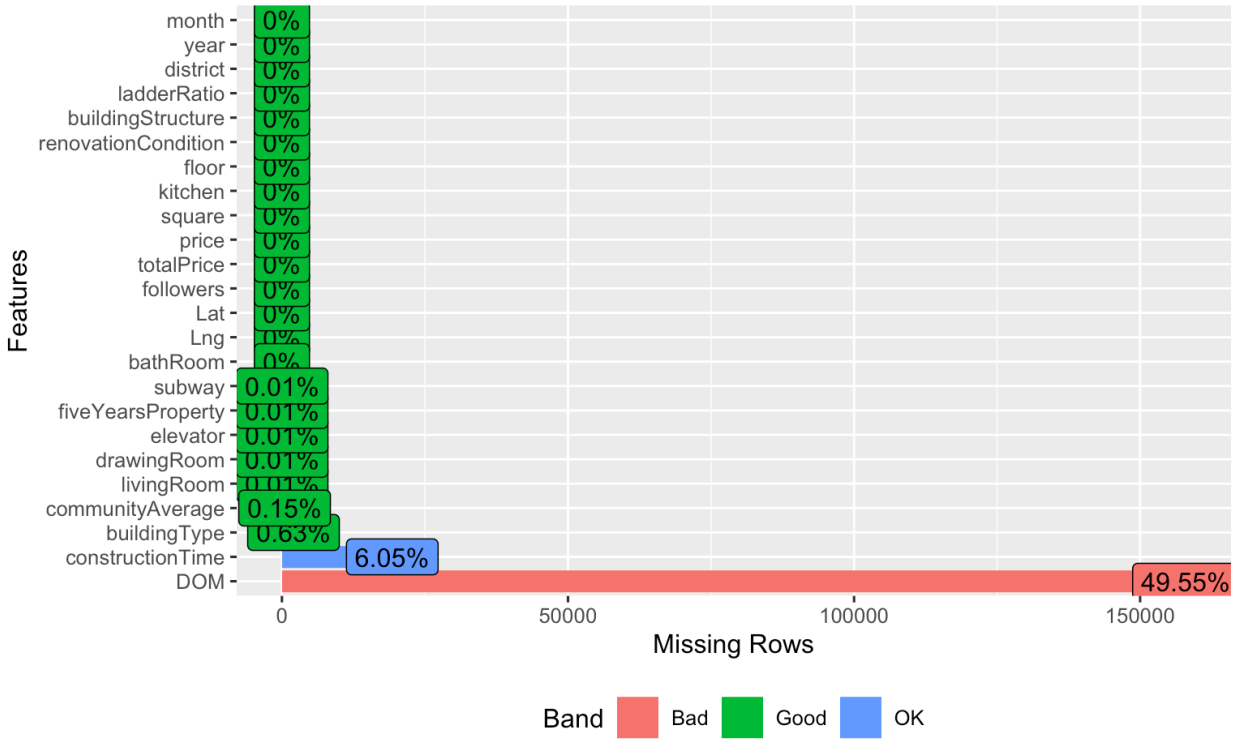
```
## # A tibble: 6 x 24
##     Lng   Lat   DOM followers totalPrice price square livingRoom drawingRo
om
##   <dbl> <dbl> <dbl>     <dbl>      <dbl> <dbl>  <dbl>      <dbl>       <db
l>
## 1  116.  40.0  1464       106       415  31680   131           2
 1
## 2  116.  39.9   903       126       575  43436   132.          2
 2
## 3  117.  39.9  1271        48      1030  52021   198           3
 2
## 4  116.  40.1   965       138       298. 22202   134           3
 1
## 5  116.  39.9   927       286       392  48396    81           2
 1
## 6  116.  40.0   861        57       276. 52000    53           1
 0
## # … with 15 more variables: kitchen <dbl>, bathRoom <dbl>, floor <chr>,
## #   buildingType <dbl>, constructionTime <dbl>, renovationCondition <dbl>,
## #   buildingStructure <dbl>, ladderRatio <dbl>, elevator <dbl>,
## #   fiveYearsProperty <dbl>, subway <dbl>, district <dbl>,
## #   communityAverage <dbl>, year <dbl>, month <dbl>
```

Here, we noticed that there are Chinese characters in *floor* features. Thus, we will convert the features to numeric.

```
house <- house %>% mutate(floor = parse_number(iconv(enc2utf8(house$floor),su
b="byte")))
```

Before we take a look at some of the data in any details, we take a first look at the missing features.

```
DataExplorer::plot_missing(house)
```

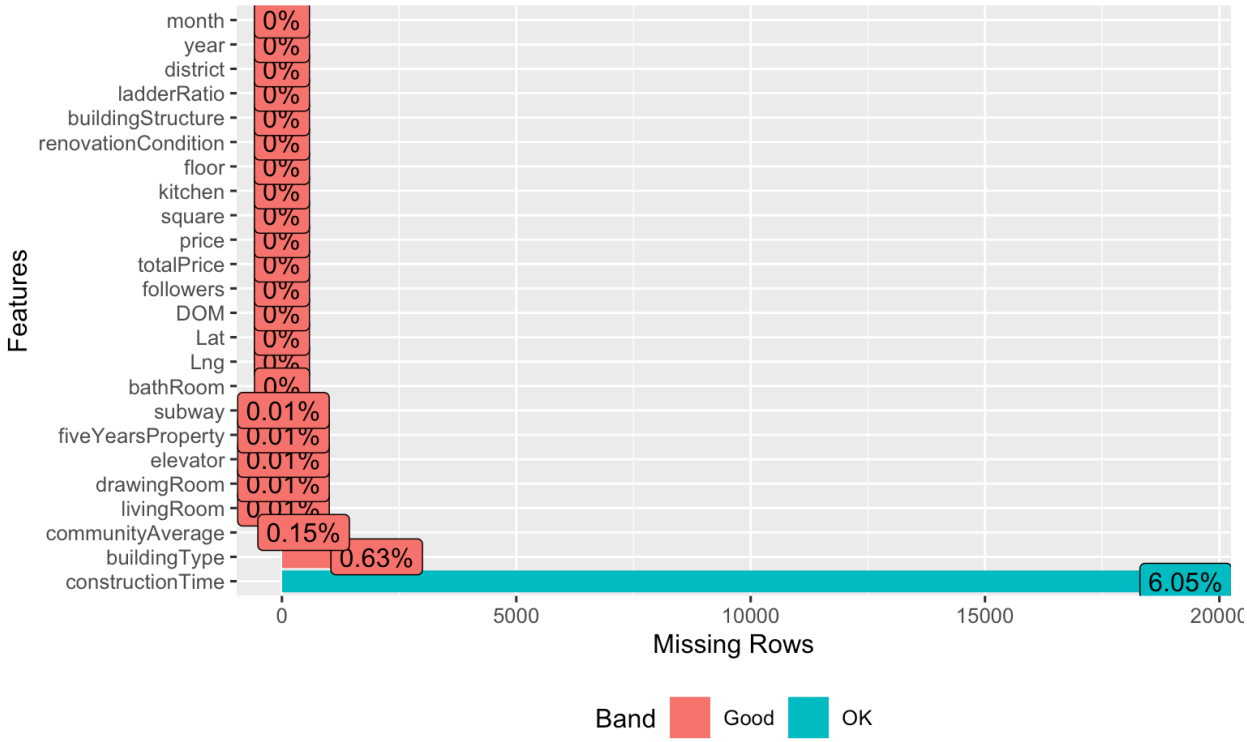


Since we miss 49.55% of DOM (active days on market) and we don't know any information about it, we decided to replace the na value with its median.

```
house$DOM <- ifelse(is.na(house$DOM), median(house$DOM, na.rm = TRUE), house$
DOM)
```

After removing the missing data of DOM, we check again to see if there is any other missing data in the dataset.

```
DataExplorer::plot_missing(house)
```

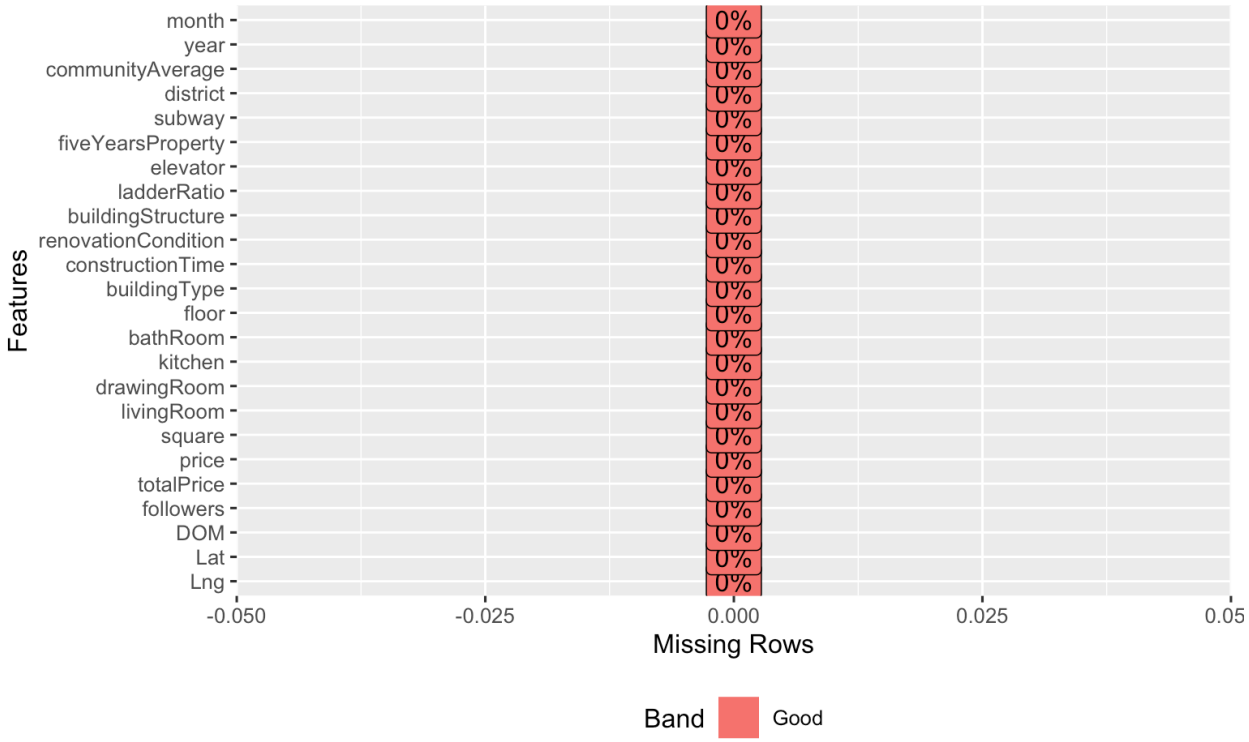


Only very few missing data in our dataset which will not influence our results. Thus, we decide to remove all the missing data.

```
house <- na.omit(house)
```

We plot the figure again to confirm there is no more missing data in the house dataset.

```
DataExplorer::plot_missing(house)
```



3. Data Exploration

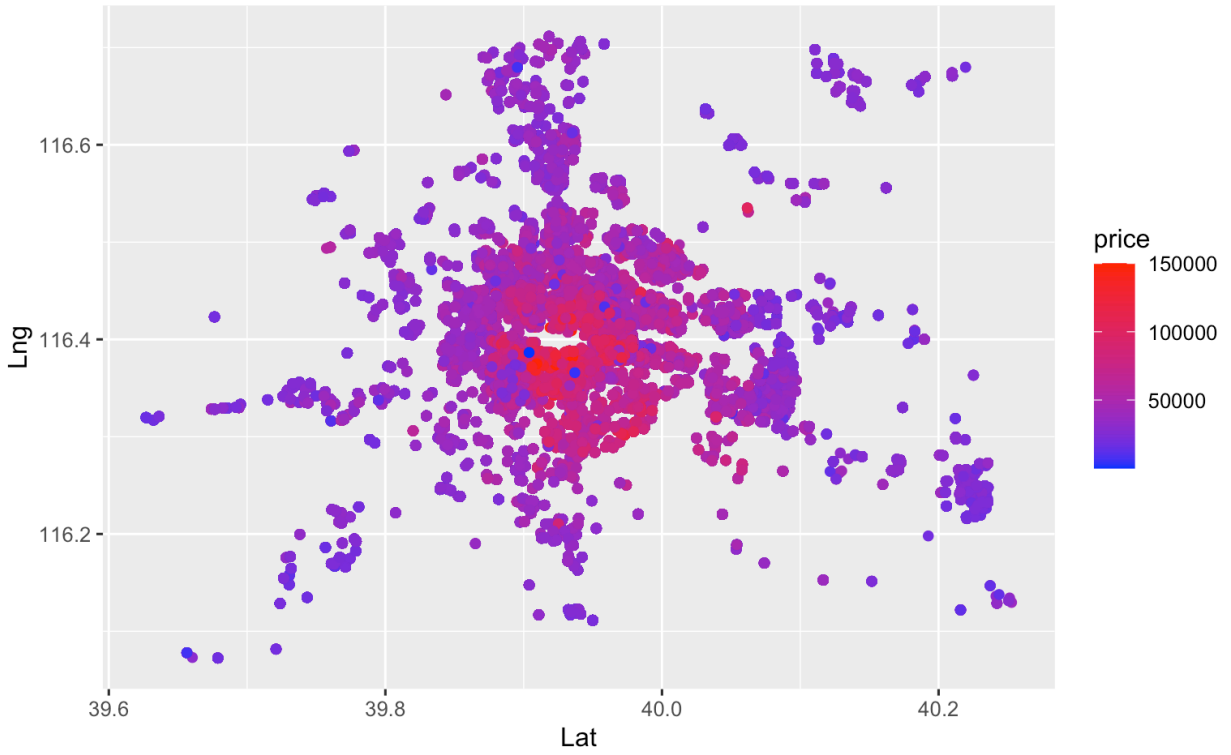
Since there are categorical and numerical features, we will explore more details with different techniques. We are mainly focusing on *price* (price of unit) and *totalPrice* (in million yuan). *totalPrice* will be the target variable of the regression model.

3.1 Categorical Features

3.1.1 Map

The houses are getting more expensive in the city center while more affordable in the suburbs.

```
house %>% ggplot(aes(Lat, Lng)) +
  geom_point(aes(color = price)) +
  scale_color_gradient(low="blue", high="red")
```



3.1.2 Building Type

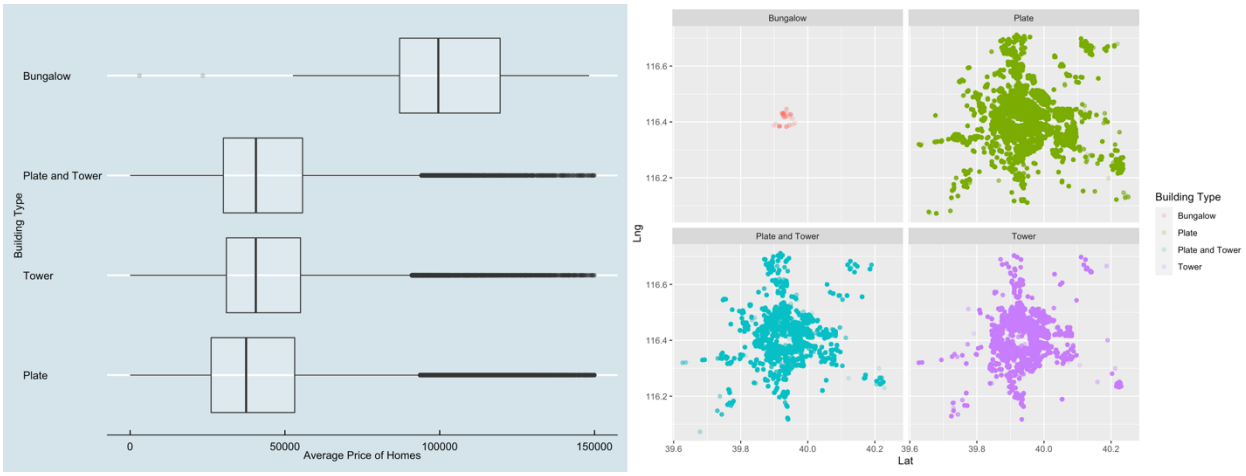
Bungalow are the most expensive homes in Beijing and are mostly located in the city center. These types of homes are probably the historical type of residence called "Siheyuan" which has historical and cultural meanings. And that's why there are very few Bungalow in the city with super high price.

```
box1 <- house %>% mutate(buildingType = case_when(buildingType == 1 ~ 'Tower'
,
                                                  buildingType == 2 ~ 'B
ungalow',
                                                  buildingType == 3 ~ 'P
late and Tower',
                                                  buildingType == 4 ~ 'P
late')) %>%
  mutate(buildingType = reorder(buildingType, price, FUN = median)) %>%
  ggplot(aes(price, buildingType)) +
  geom_boxplot(alpha = 0.2) +
  xlab('Average Price of Homes') +
  ylab('Building Type') +
  theme_economist()

map1 <- house %>% mutate(buildingType = case_when(buildingType == 1 ~ 'Tower'
,
                                                  buildingType == 2 ~ 'Bungalow',
                                                  buildingType == 3 ~ 'Plate and Towe
```

```
r',
                                                buildingType == 4 ~ 'Plate')) %>%
  ggplot(aes(Lat, Lng)) +
  geom_point(aes(color = buildingType), alpha = 0.2) +
  scale_color_discrete(name = 'Building Type') +
  facet_wrap(~buildingType)

grid.arrange(box1, map1, ncol = 2)
```



3.1.3 Renovation Condition

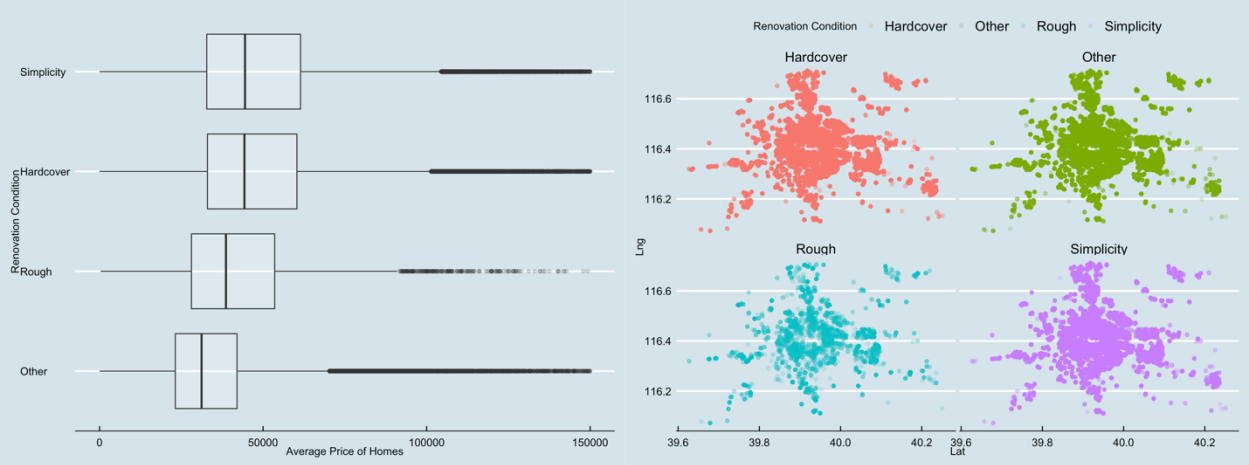
Simplicity and Hardcover renovations are more expensive.

```
box2 <- house %>% mutate(renovationCondition = case_when(renovationCondition
== 1 ~ 'Other',
                                                         renovationCondi
tion == 2 ~ 'Rough',
                                                         renovationCondi
tion == 3 ~ 'Simplicity',
                                                         renovationCondi
tion == 4 ~ 'Hardcover')) %>%
  mutate(renovationCondition = reorder(renovationCondition, price, FUN = medi
an)) %>%
  ggplot(aes(price, renovationCondition)) +
  geom_boxplot(alpha = 0.2) +
  xlab('Average Price of Homes') +
  ylab('Renovation Condition') +
  theme_economist()

map2 <- house %>% mutate(renovationCondition = case_when(renovationCondition
== 1 ~ 'Other',
                                                         renovationCondition
== 2 ~ 'Rough',
                                                         renovationCondition
== 3 ~ 'Simplicity',
                                                         renovationCondition
```

```
== 4 ~ 'Hardcover')) %>%
  ggplot(aes(Lat, Lng)) +
  geom_point(aes(color = renovationCondition), alpha = 0.2) +
  facet_wrap(~renovationCondition) +
  scale_color_discrete(name = 'Renovation Condition') +
  theme_economist()

grid.arrange(box2, map2, ncol = 2)
```



3.1.4 Building Structure

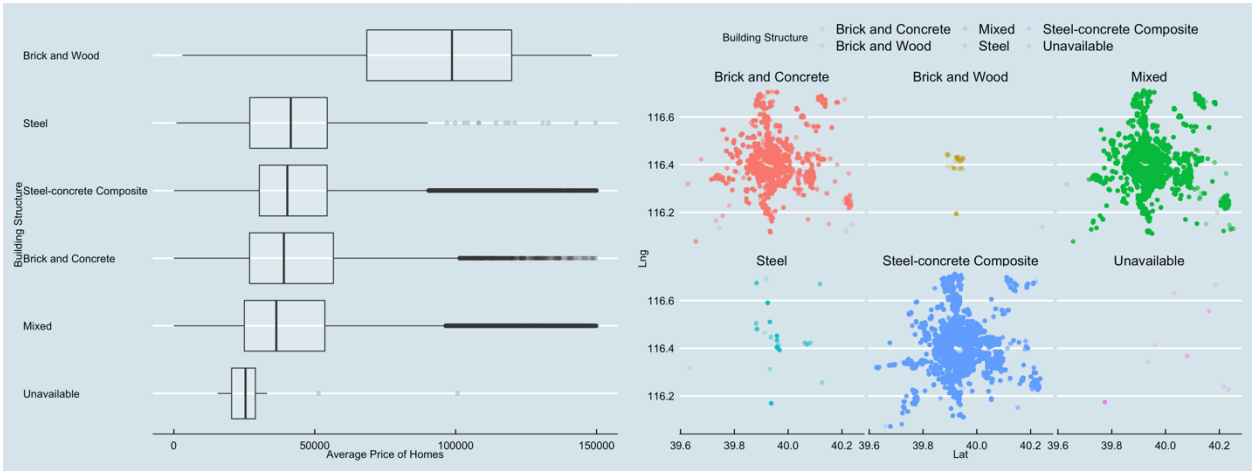
Brick and wood type homes are the most expensive which are almost twice as the others.

```
box3 <- house %>% mutate(buildingStructure = case_when(buildingStructure == 1
 ~ 'Unavailable',
                                                        buildingStructure
 == 2 ~ 'Mixed',
                                                        buildingStructure
 == 3 ~ 'Brick and Wood',
                                                        buildingStructure
 == 4 ~ 'Brick and Concrete',
                                                        buildingStructure
 == 5 ~ 'Steel',
                                                        buildingStructure
 == 6 ~ 'Steel-concrete Composite')) %>%
  mutate(buildingStructure = reorder(buildingStructure, price, FUN = median))
 %>%
  ggplot(aes(price, buildingStructure)) +
  geom_boxplot(alpha = 0.2) +
  xlab('Average Price of Homes') +
  ylab('Building Structure') +
  theme_economist()

map3 <- house %>% mutate(buildingStructure = case_when(buildingStructure == 1
 ~ 'Unavailable',
                                                    buildingStructure == 2 ~ 'Mixe
```

```
d',
                                                    buildingStructure == 3 ~ 'Bric
k and Wood',
                                                    buildingStructure == 4 ~ 'Bric
k and Concrete',
                                                    buildingStructure == 5 ~ 'Stee
l',
                                                    buildingStructure == 6 ~ 'Stee
l-concrete Composite')) %>%
  ggplot(aes(Lat, Lng)) +
  geom_point(aes(color = buildingStructure), alpha = 0.2) +
  facet_wrap(~buildingStructure) +
  scale_color_discrete(name = 'Building Structure') +
  theme_economist()

grid.arrange(box3, map3, ncol = 2)
```



3.1.5 Elevator

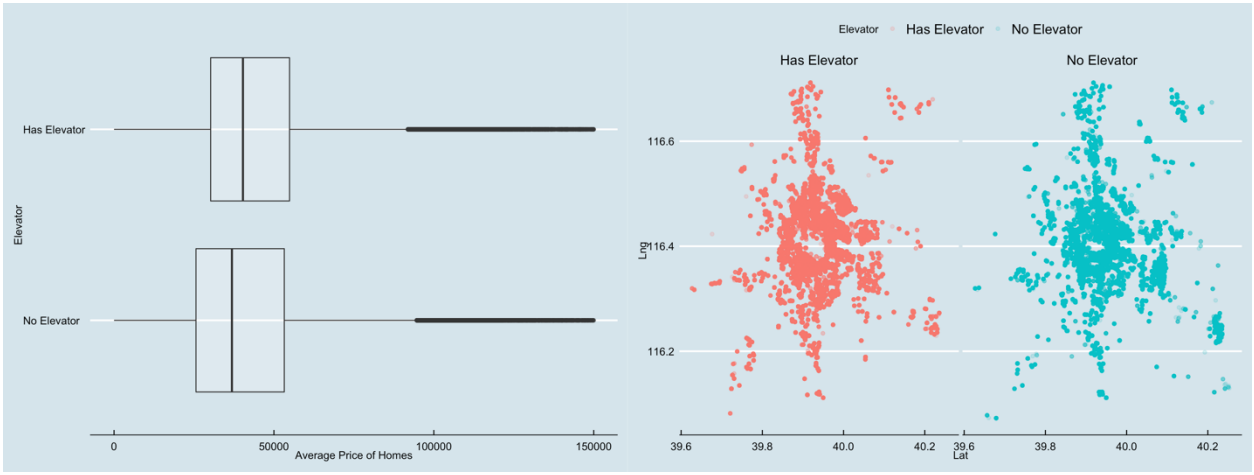
The elevator doesn't have a significant influence on home prices.

```
box4 <- house %>% mutate(elevator = case_when(elevator == 1 ~ 'Has Elevator',
                                              elevator == 0 ~ 'No Elevat
or')) %>%
  mutate(elevator = reorder(elevator, price, FUN = median)) %>%
  ggplot(aes(price, elevator)) +
  geom_boxplot(alpha = 0.2) +
  xlab('Average Price of Homes') +
  ylab('Elevator') +
  theme_economist()

map4 <- house %>% mutate(elevator = case_when(elevator == 1 ~ 'Has Elevator',
                                    elevator == 0 ~ 'No Elevator')) %>%
  ggplot(aes(Lat, Lng)) +
  geom_point(aes(color = elevator), alpha = 0.2) +
  facet_wrap(~elevator) +
```

```
  scale_color_discrete(name = 'Elevator') +
  theme_economist()

grid.arrange(box4, map4, ncol = 2)
```



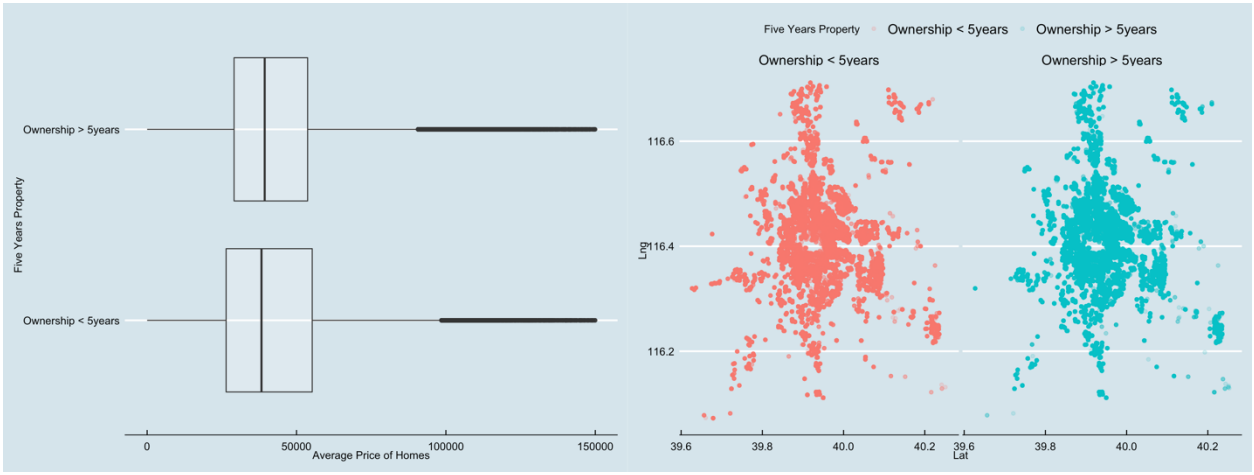
3.1.6 Five Years Property

The dependence of the price with respect the owner having the property for more than 5 years is really small.

```
box5 <- house %>% mutate(fiveYearsProperty = case_when(fiveYearsProperty == 1
 ~ 'Ownership > 5years',
                                                        fiveYearsProperty
 == 0 ~ 'Ownership < 5years')) %>%
  mutate(fiveYearsProperty = reorder(fiveYearsProperty, price, FUN = median))
 %>%
  ggplot(aes(price, fiveYearsProperty)) +
  geom_boxplot(alpha = 0.2) +
  xlab('Average Price of Homes') +
  ylab('Five Years Property') +
  theme_economist()

map5 <- house %>% mutate(fiveYearsProperty = case_when(fiveYearsProperty == 1
 ~ 'Ownership > 5years',
                                                       fiveYearsProperty == 0 ~ 'Owne
rship < 5years')) %>%
  ggplot(aes(Lat, Lng)) +
  geom_point(aes(color = fiveYearsProperty), alpha = 0.2) +
  facet_wrap(~fiveYearsProperty) +
  scale_color_discrete(name = 'Five Years Property') +
  theme_economist()

grid.arrange(box5, map5, ncol = 2)
```



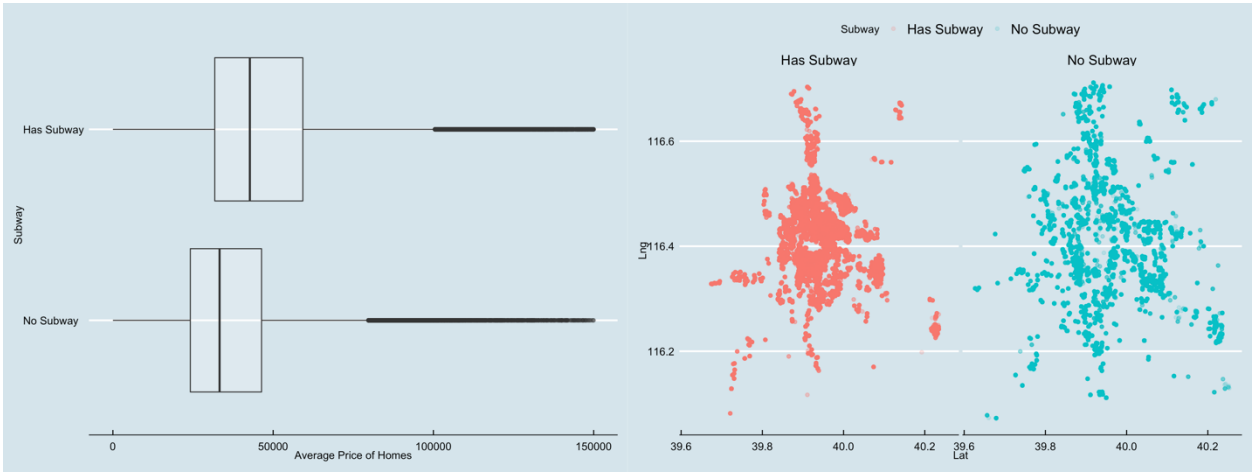
3.1.7 Subway

The nearby subway doesn't have a significant influence on home prices.

```
box6 <- house %>% mutate(subway = case_when(subway == 1 ~ 'Has Subway',
                                            subway == 0 ~ 'No Subway')) %>
%
  mutate(subway = reorder(subway, price, FUN = median)) %>%
  ggplot(aes(price, subway)) +
  geom_boxplot(alpha = 0.2) +
  xlab('Average Price of Homes') +
  ylab('Subway') +
  theme_economist()

map6 <- house %>% mutate(subway = case_when(subway == 1 ~ 'Has Subway',
                                      subway == 0 ~ 'No Subway')) %>%
  ggplot(aes(Lat, Lng)) +
  geom_point(aes(color = subway), alpha = 0.2) +
  facet_wrap(~subway) +
  scale_color_discrete(name = 'Subway') +
  theme_economist()

grid.arrange(box6, map6, ncol = 2)
```



3.1.8 District

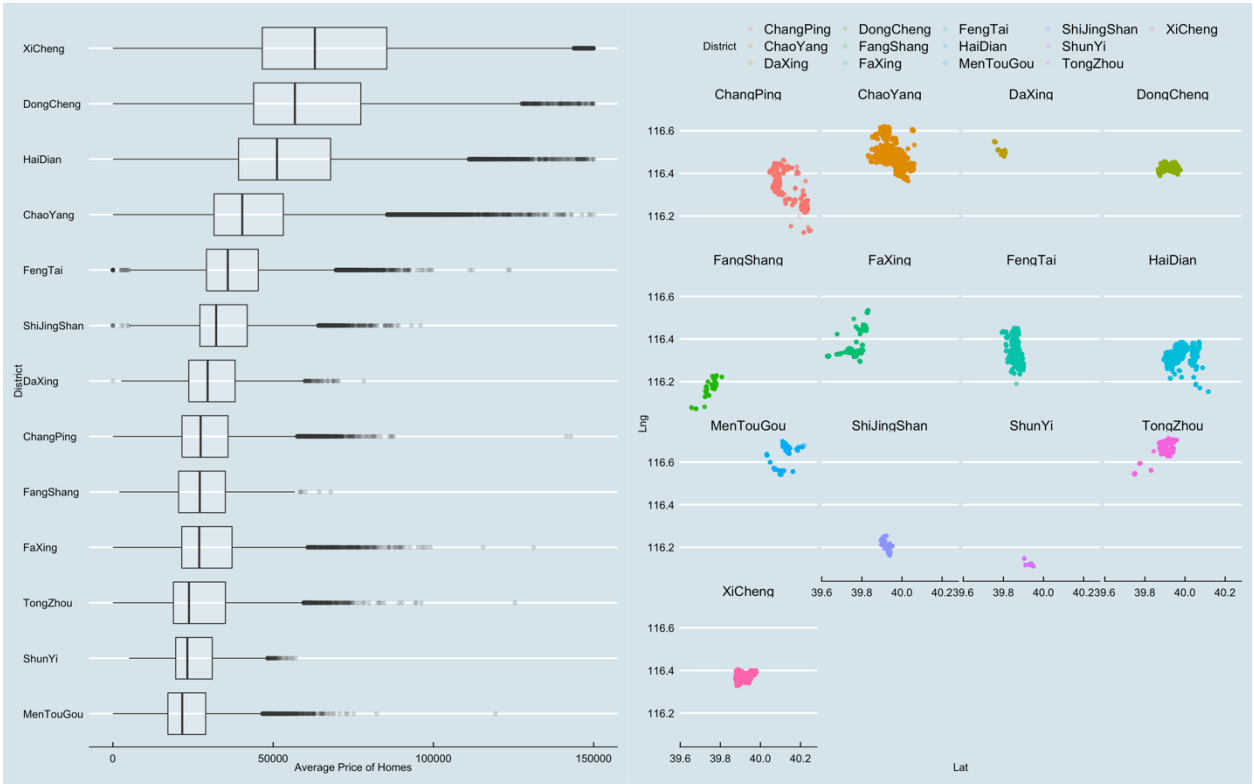
Homes located in Xicheng District are the most expensive because they are in the city center.

```
box7 <- house %>% mutate(district = case_when(district == 1 ~ 'DongCheng',
                                              district == 2 ~ 'FengTai',
                                              district == 3 ~ 'DaXing',
                                              district == 4 ~ 'FaXing',
                                              district == 5 ~ 'FangShang
',
                                              district == 6 ~ 'ChangPing
',
                                              district == 7 ~ 'ChaoYang'
,
                                              district == 8 ~ 'HaiDian',
                                              district == 9 ~ 'ShiJingSh
an',
                                              district == 10 ~ 'XiCheng'
,
                                              district == 11 ~ 'TongZhou
',
                                              district == 12 ~ 'ShunYi',
                                              district == 13 ~ 'MenTouGo
u')) %>%
  mutate(district = reorder(district, price, FUN = median)) %>%
  ggplot(aes(price, district)) +
  geom_boxplot(alpha = 0.2) +
  xlab('Average Price of Homes') +
  ylab('District') +
  theme_economist()

map7 <- house %>% mutate(district = case_when(district == 1 ~ 'DongCheng',
                                              district == 2 ~ 'FengTai',
                                              district == 3 ~ 'DaXing',
```

```
                                    district == 4 ~ 'FaXing',
                                    district == 5 ~ 'FangShang',
                                    district == 6 ~ 'ChangPing',
                                    district == 7 ~ 'ChaoYang',
                                    district == 8 ~ 'HaiDian',
                                    district == 9 ~ 'ShiJingShan',
                                    district == 10 ~ 'XiCheng',
                                    district == 11 ~ 'TongZhou',
                                    district == 12 ~ 'ShunYi',
                                    district == 13 ~ 'MenTouGou')) %>%
  ggplot(aes(Lat, Lng)) +
  geom_point(aes(color = district), alpha = 0.2) +
  facet_wrap(~district) +
  scale_color_discrete(name = 'District') +
  theme_economist()

grid.arrange(box7, map7, ncol = 2)
```

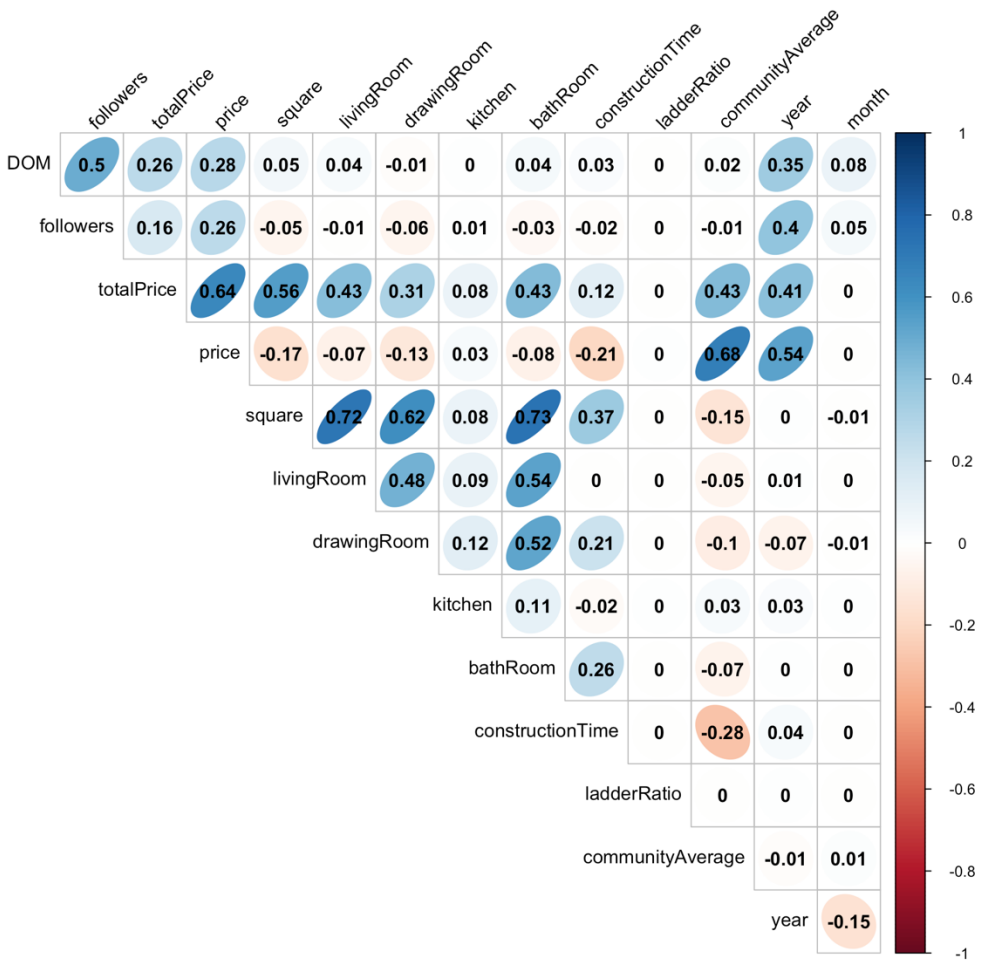


3.2 Numerical Features

Now let's explore more about numerical features.

```
numeric_cols <- names(house[,sapply(house,function(x) {is.numeric(x)})] %>% s
elect(-Lng, -Lat, -buildingType, -renovationCondition, -buildingStructure, -e
levator, -fiveYearsProperty, -subway, -district))
cor_numerics <- cor(house[,numeric_cols])
```

```
corrplot(cor_numerics,
         method = 'ellipse',
         type="upper",
         addCoef.col = "black", # Add coefficient of correlation
         tl.col="black", tl.srt=45, #Text label color and rotation
         # hide correlation coefficient on the principal diagonal
         diag=FALSE
)
```



totalPrice has a strong positive correlation with the *square* since if the home has a larger square area, the chance you pay for it is higher.

totalPrice has a positive correlation with *livingRoom* and *bathRoom*.

price has a strong positive correlation with *communityAverage* which means prices of homes are higher in dense population area. (maybe better public resources including education and medical service)

price has a strong positive correlation with *year* which means the homes in Beijing is getting more expensive during the time.

square has a strong positive correlation with *livingRoom*, *bathRoom*, and *drawingRoom* because the larger homes will probably have more rooms.

```
# write a function to analyze distributions
plot_hist = function(df, numcol,Color = "green2" ,bins = 15,Title = "")
{
  skew = skewness(df[,numcol])
  options(repr.plot.width=4, repr.plot.height=3) # Set the initial plot area
dimensions
  bw = (max(df[,numcol]) - min(df[,numcol]))/(bins + 1)
  p <- ggplot(df, aes_string(numcol)) +
    geom_histogram(alpha = 0.6, binwidth = bw,color = Color)+
    ggtitle(paste("Histogram of ",Title,numcol,sep = ''))+
    annotate(geom = "text",x = (max(df[,numcol]) + min(df[,numcol]))/2, y = 2
00000,label = paste("Skew:",round(skew,2)))+
    theme_minimal()
}
# check the distribution of totalprice, price
p1 <- plot_hist(house, 'totalPrice')
```

```
## Warning in mean.default(x): argument is not numeric or logical: returning
NA
```

```
p2 <- plot_hist(house, 'price')
```

```
## Warning in mean.default(x): argument is not numeric or logical: returning
NA
```

```
# check square distribution
p3 <- plot_hist(house, 'square')
```

```
## Warning in mean.default(x): argument is not numeric or logical: returning
NA
```

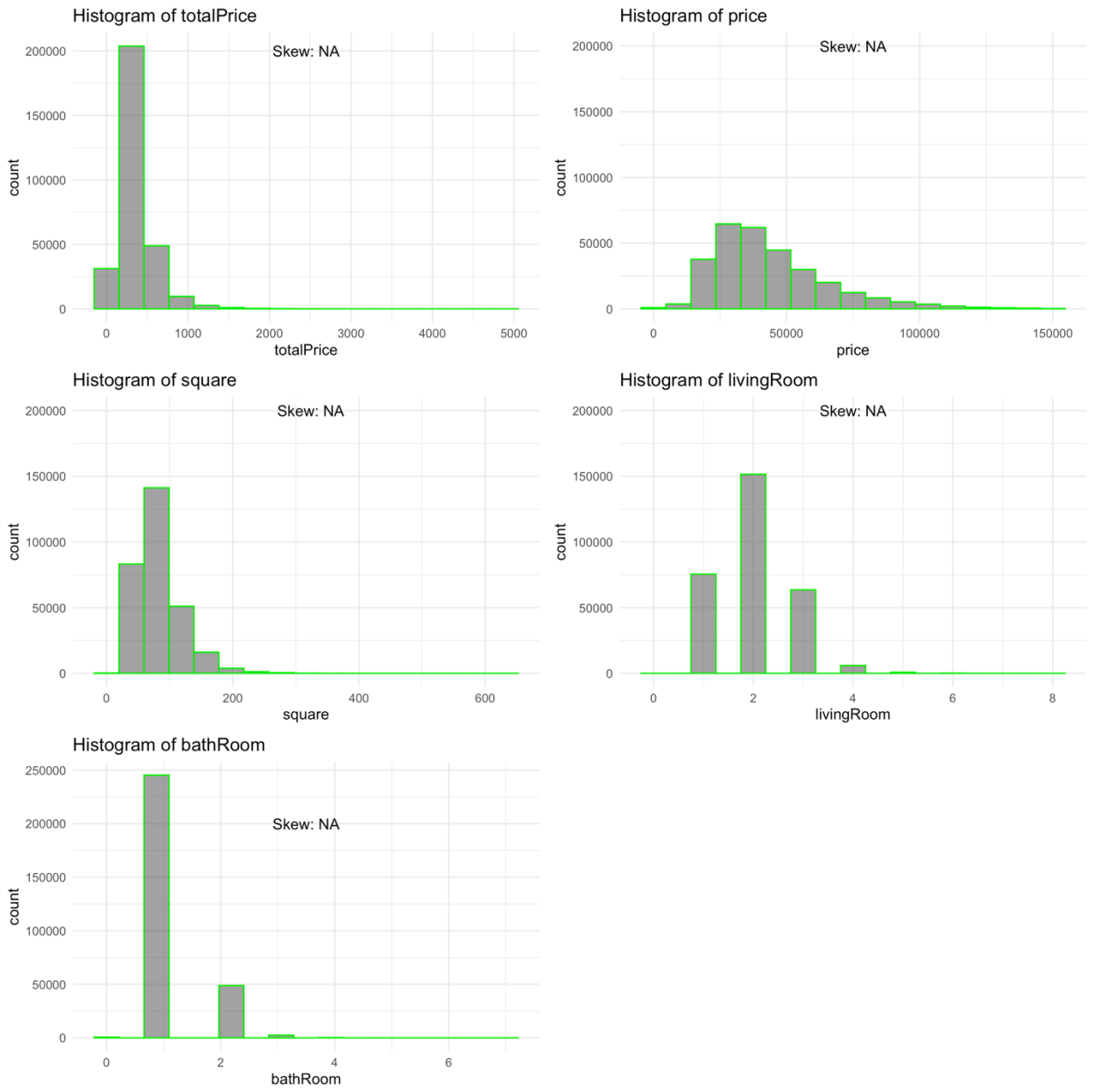
```
# check living room distribution
p4 <- plot_hist(house, 'livingRoom')
```

```
## Warning in mean.default(x): argument is not numeric or logical: returning
NA
```

```
# check bathroom distribution
p5 <- plot_hist(house, 'bathRoom')
```

```
## Warning in mean.default(x): argument is not numeric or logical: returning
NA
```

```
grid.arrange(p1, p2, p3, p4, p5)
```



From the distribution diagram, we decide to remove some outliers. My method of removing outliers is quite basic. I inspected the graphs and filtered the data below to remove the outliers about a visually identified constant.

```
house <- house %>% filter(totalPrice < 2500,
                          square < 400,
                          livingRoom < 8,
                          bathRoom < 6)
```

4. Regression Model

Firstly, we create the loss function RMSE as shown below:

```
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

4.1. Prepare Train/Test Samples

The *House* dataset will be splitted into 2 subsets: "house_train', a training subset containing 80% of the data to train the algorithm, and"house_test" subset containing 20% of the data for test.

```
set.seed(2019)
test_index <- createDataPartition(y = house$totalPrice, times = 1, p = 0.2, l
ist = FALSE)
house_train <- house[-test_index,]
house_test <- house[test_index,]
```

4.2 Simplest Model (Average)

We start with a model that assumes the same totalPrice for all houses in Beijing. In this case, the least squares of μ, the estimate that minimized the room mean squared error, is the average price of all houses in Beijing.

```
mu <- mean(house_train$totalPrice)
naive_rmse <- RMSE(house_test$totalPrice, mu)
rmse_results <- data_frame(method = 'Just the average',
                           RMSE = naive_rmse)

## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.

rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	215.6306

4.3 Generalized Linear Model

We build a Generalized Linear Model here to estimate the RMSE value.

```
temp_train <- as.data.frame(cbind(
  house_train %>% select_if(is.numeric) %>% select(-Lng, -Lat, -year, -month)
,
  'bldgType'= dummy.code(house_train$buildingType),
  'bldgStruc'= dummy.code(house_train$buildingStructure),
  'renovation'= dummy.code(house_train$renovationCondition),
  'hasElevator'= dummy.code(house_train$elevator),
```

```
  'hasSubway'= dummy.code(house_train$subway),
  'IsFiveYears'= dummy.code(house_train$fiveYearsProperty),
  'districtCat'= dummy.code(house_train$district)))

temp_test <- as.data.frame(cbind(
  house_test %>% select_if(is.numeric) %>% select(-Lng, -Lat, -year, -month),
  'bldgType'= dummy.code(house_test$buildingType),
  'bldgStruc'= dummy.code(house_test$buildingStructure),
  'renovation'= dummy.code(house_test$renovationCondition),
  'hasElevator'= dummy.code(house_test$elevator),
  'hasSubway'= dummy.code(house_test$subway),
  'IsFiveYears'= dummy.code(house_test$fiveYearsProperty),
  'districtCat'= dummy.code(house_test$district)))

# Train Model
my_control <-trainControl(method="cv", number=10)
glmGrid <- expand.grid(alpha = seq(0, 1, by = 0.1), lambda = seq(0.001,0.1,by
 = 0.0005))
glm_mod <- train(x= temp_train %>% select(-totalPrice), y=temp_train$totalPri
ce, method='glmnet', trControl= my_control, tuneGrid=glmGrid)

# Test Model
glm_preds <- predict(glm_mod,temp_test %>% select(-totalPrice))
glm_rmse <- RMSE(temp_test$totalPrice, glm_preds)

rmse_results <- bind_rows(rmse_results,
                          data_frame(method = 'Generalized Linear Model',
                                     RMSE = glm_rmse))
rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	215.63065
Generalized Linear Model	72.21111

4.4 Support Vector Machine

We build a Support Vector Machine Model here to estimate the RMSE value.

```
# Train Model
svm_model <- svm(totalPrice ~., data = temp_train)
svm_preds <- predict(svm_model, temp_test%>% select(-totalPrice))
svm_rmse <- RMSE(temp_test$totalPrice, svm_preds)

rmse_results <- bind_rows(rmse_results,
                          data_frame(method = 'SVM Model',
                                     RMSE = svm_rmse))
rmse_results %>% knitr::kable()
```

method	RMSE

Just the average	215.63065
Generalized Linear Model	72.21111
SVM Model	16.15483

5. Conclusion

The RMSE values for the used models are shown below:

```
rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	215.63065
Generalized Linear Model	72.21111
SVM Model	16.15483

We can confirm that we built a machine learning algorithm to predict house price with The House Price in Beijing dataset. The simplest model (Average) calculates the RMSE of 215.63. The Generalized Linear Model and SVM Model improve the accuracy by 200% and 1256%. In conclusion, the final RMSE is 16.15483 with an improvement over 12 times with respect to the simplest model.