Basic Property Pricing Model

Subhendu Mahapatro

2021-07-03

Contents

1 Introduction

Asset pricing is an important aspect of any financial project, especially the projects involving large transactions. Assets in this context can be something with intrinsic value that has the potential to generate future cash flows. There are various types of assets and asset pricing concepts that are used by professionals.

For this project we are focusing on real estate assets and property sales data based in New York City. The data has been downloaded manually from Kaggle. before being loaded into the R environment for analysis. Please see link to the source here. Our goal here is to build a model to predict the selling prices of the real estate assets (buildings) as per the dataset.

2 Analysis

As we have downloaded the dataset into the working directory, we can start by loading the dataset into the R environment.

```r
# loading data into the environment and saving in Rda format for future reference
nyc_rolling_sales <- read.csv("nyc-rolling-sales.csv")
save(nyc_rolling_sales, file = "nyc_rolling_sales.Rda")

str(nyc_rolling_sales)
```

```
## 'data.frame':    84548 obs. of  22 variables:
##  $ X                             : int  4 5 6 7 8 9 10 11 12 13 ...
##  $ BOROUGH                       : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ NEIGHBORHOOD                  : chr  "ALPHABET CITY" "ALPHABET CITY" "ALPHABET CITY" "ALPHABET CI
##  $ BUILDING.CLASS.CATEGORY       : chr  "07 RENTALS - WALKUP APARTMENTS             " "07 RENTALS - 
##  $ TAX.CLASS.AT.PRESENT          : chr  "2A" "2" "2" "2B" ...
##  $ BLOCK                         : int  392 399 399 402 404 405 406 407 379 387 ...
##  $ LOT                           : int  6 26 39 21 55 16 32 18 34 153 ...
##  $ EASE.MENT                     : logi  NA NA NA NA NA NA ...
##  $ BUILDING.CLASS.AT.PRESENT     : chr  "C2" "C7" "C7" "C4" ...
##  $ ADDRESS                       : chr  "153 AVENUE B" "234 EAST 4TH   STREET" "197 EAST 3RD   STREE
##  $ APARTMENT.NUMBER              : chr  " " " " " " " " ...
##  $ ZIP.CODE                      : int  10009 10009 10009 10009 10009 10009 10009 10009 10009 10009 
##  $ RESIDENTIAL.UNITS             : int  5 28 16 10 6 20 8 44 15 24 ...
##  $ COMMERCIAL.UNITS              : int  0 3 1 0 0 0 0 2 0 0 ...
##  $ TOTAL.UNITS                   : int  5 31 17 10 6 20 8 46 15 24 ...
##  $ LAND.SQUARE.FEET              : chr  "1633" "4616" "2212" "2272" ...
##  $ GROSS.SQUARE.FEET             : chr  "6440" "18690" "7803" "6794" ...
##  $ YEAR.BUILT                    : int  1900 1900 1900 1913 1900 1900 1920 1900 1920 1920 ...
##  $ TAX.CLASS.AT.TIME.OF.SALE     : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ BUILDING.CLASS.AT.TIME.OF.SALE: chr  "C2" "C7" "C7" "C4" ...
##  $ SALE.PRICE                    : chr  "6625000" " -  " " -  " "3936272" ...
##  $ SALE.DATE                     : chr  "2017-07-19 00:00:00" "2016-12-14 00:00:00" "2016-12-09 00:0
```

```r
dim(nyc_rolling_sales)
```

```
## [1] 84548    22
```

Upon exploring the dataset, we note that certain variables' classes are not in the suitable formats. We start cleaning the data with reformatting the names of the variables and then changing the class of certain variables.

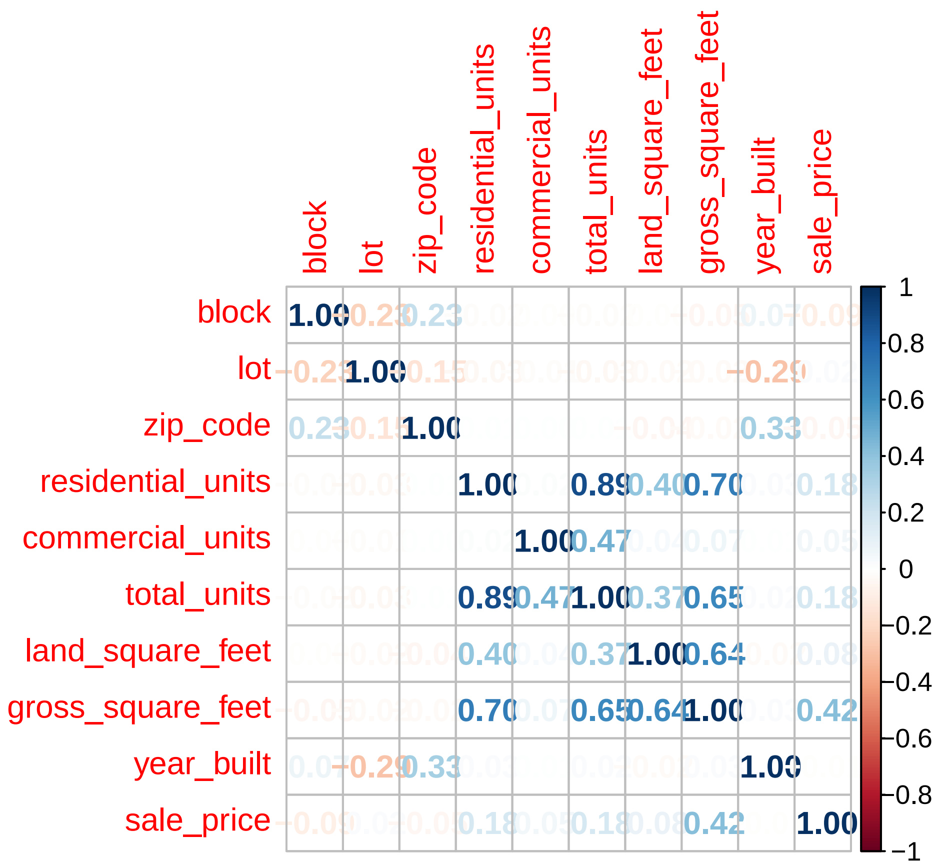
```
names(nyc_rolling_sales) <- tolower(str_replace_all(names(nyc_rolling_sales), "\\.", "_"))
names(nyc_rolling_sales)
```

```
##  [1] "x"                              "borough"
##  [3] "neighborhood"                   "building_class_category"
##  [5] "tax_class_at_present"           "block"
##  [7] "lot"                            "ease_ment"
##  [9] "building_class_at_present"      "address"
## [11] "apartment_number"               "zip_code"
## [13] "residential_units"              "commercial_units"
## [15] "total_units"                    "land_square_feet"
## [17] "gross_square_feet"              "year_built"
## [19] "tax_class_at_time_of_sale"      "building_class_at_time_of_sale"
## [21] "sale_price"                     "sale_date"
```

```
nyc_rolling_sales <- nyc_rolling_sales %>%
  mutate(x = NULL,
         borough = as.factor(borough),
         neighborhood = as.factor(neighborhood),
         building_class_category = as.factor(building_class_category),
         tax_class_at_present = as.factor(tax_class_at_present),
         ease_ment = NULL,
         building_class_at_present = as.factor(building_class_at_present),
         address = NULL,
         apartment_number = NULL,
         zip_code = as.integer(zip_code),
         land_square_feet = as.integer(land_square_feet),
         gross_square_feet = as.integer(gross_square_feet),
         tax_class_at_time_of_sale = as.factor(tax_class_at_time_of_sale),
         building_class_at_time_of_sale = as.factor(building_class_at_time_of_sale),
         sale_price = as.integer(sale_price),
         sale_date = as.Date(sale_date))
```

Correlation analysis and linear regression are widely used in finance, and compliment & serve the concept of relative valuation. We look at the correlation among the numeric variables.

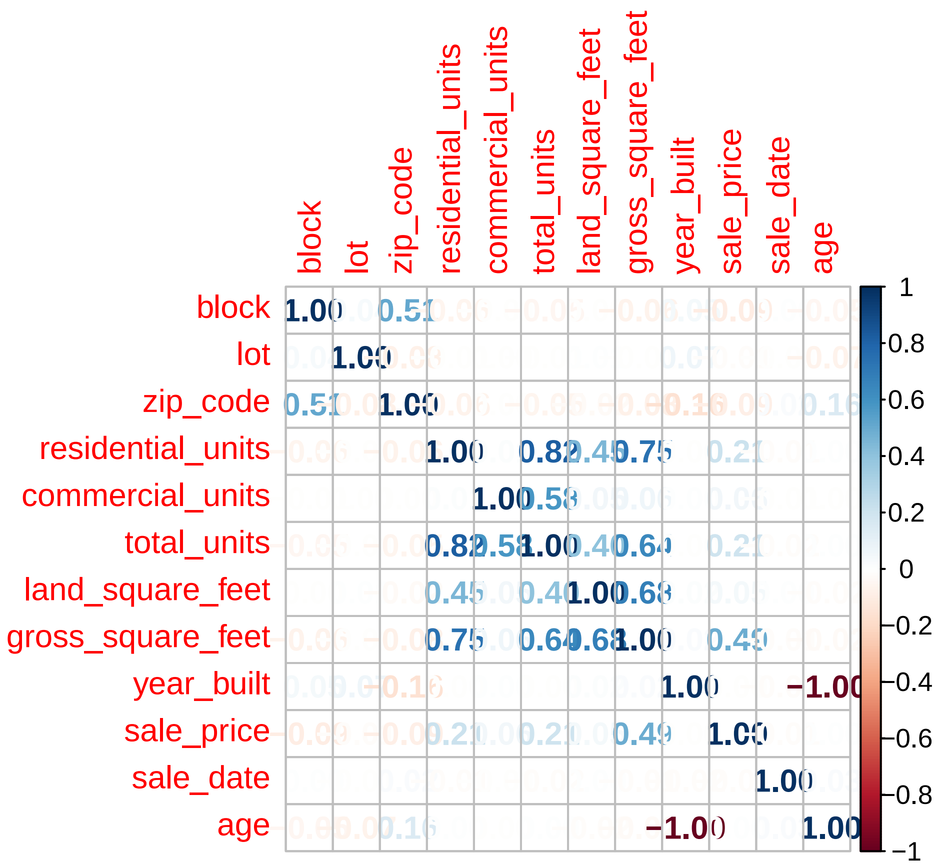
```
nyc_rolling_sales %>% select_if(is.numeric) %>%
  cor(use = "pairwise.complete.obs") %>% corrplot(method = "number")
```



The variables with notable correlation with selling price are, as we note from the plot, gross area, number of residential units and total number of units (residencial and commercial).

As NAs and odd data entries can affect the results, we remove NAs and only consider entries with gross area, total land area and selling price values of more than 100. We also introduce the age of the buildings as a variable, which is calculated as the difference in years between the year built and the date of sale. We also introduce the year of sale as a numeric variable for this plot.

```
nyc_rolling_sales_new <- nyc_rolling_sales %>%
  filter(!is.na(land_square_feet) & !is.na(gross_square_feet) & ! is.na(sale_price))
nyc_rolling_sales_new <- nyc_rolling_sales_new %>%
  filter(sale_price > 100 & land_square_feet > 100 & gross_square_feet > 100
         & zip_code > 0)
nyc_rolling_sales_new %>%
  mutate(age = as.integer(format(sale_date,"%Y")) - year_built,
         sale_date = as.integer(format(sale_date,"%Y"))) %>%
  select_if(is.numeric) %>%
  cor(use = "pairwise.complete.obs") %>% corrplot(method = "number")
```

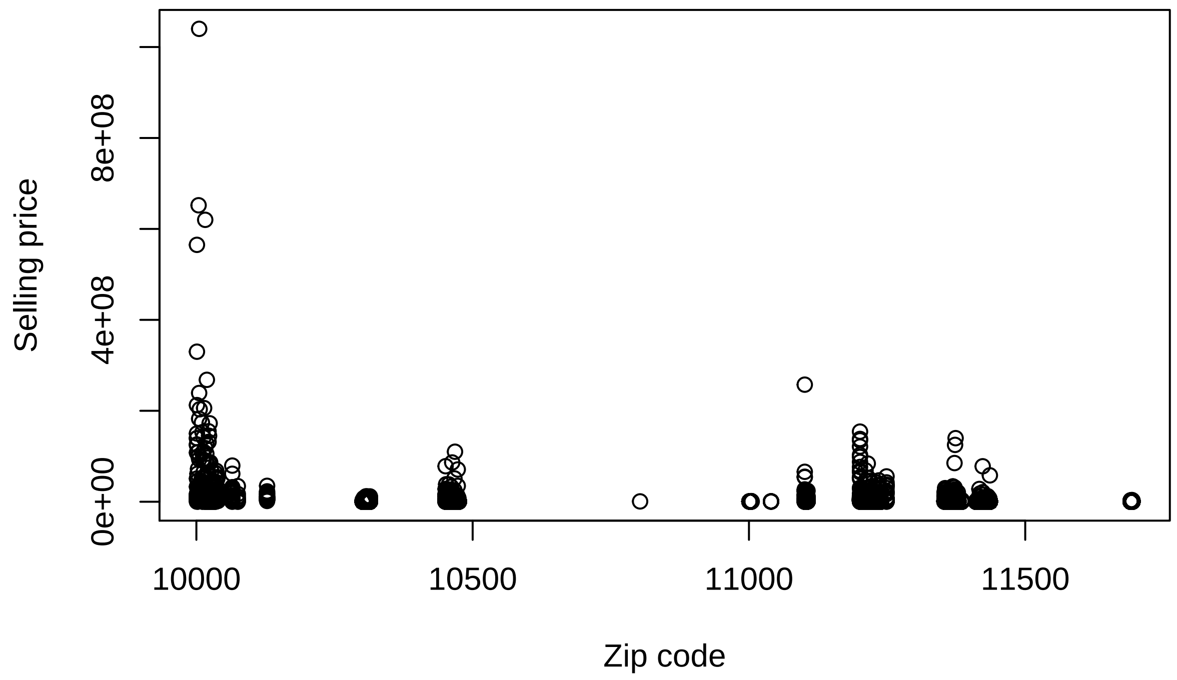


We can see that the correlation values have improved giving us a clearer picture. We note that number of residencial units, total number of units and gross area are still the ones with oconsiderable effect on selling price, with gross area having the highest correlation. We also notice the presence of correlation among the predictor variables.

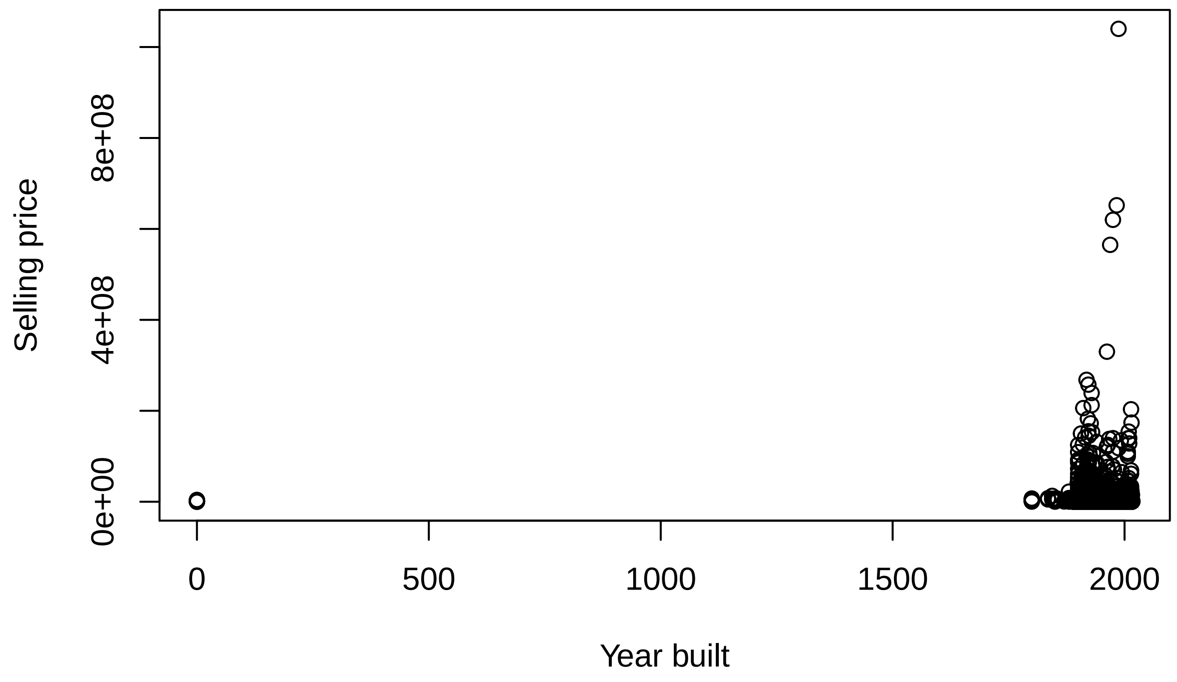
For the purpose of linear regression, if we consider the numeric predictors to be used, we have 3: residencial units, total number of units and gross area. However both residencial units, total number of units are highly correlated to gross area, so we will not be considering them for further analysis.

It is possible that although zip code did not have high correlation with selling price, the building in different parts of the city will be valued differently. We confirm that with a simple plot which shows us that selling price is not uniformly distributed acorss zip codes. We also note a similar case with selling price with respect to year built, as certain years may have some historical significance.

```r
plot(nyc_rolling_sales_new$zip_code, nyc_rolling_sales_new$sale_price,
     xlab = "Zip code", ylab = "Selling price")
```



```r
plot(nyc_rolling_sales_new$year_built, nyc_rolling_sales_new$sale_price,
     xlab = "Year built", ylab = "Selling price")
```



We move forward to building the models after selecting the necessary variables and converting selling price to price in millions.

```r
nyc_rolling_sales_new <- nyc_rolling_sales %>%
  filter(sale_price > 100 & gross_square_feet > 100
         & zip_code > 0) %>%
  mutate(sale_price = sale_price / 10^6,
         lot = NULL,
         block = NULL,
         zip_code = as.factor(zip_code),
         residential_units = NULL,
         commercial_units = NULL,
         total_units = NULL,
         land_square_feet = NULL,
         year_built = as.factor(year_built),
         sale_date = NULL)
dim(nyc_rolling_sales_new)
```

```
## [1] 28593    11
```

We have also converted zip code and year built to factor variables, as we saw the absence of uniform distribution of selling price with respect to these two predictors even though considerably high correlation was absent.

We start by creating train and test sets.

```
set.seed(1, sample.kind = "Rounding")
test_index <- caret::createDataPartition(y = nyc_rolling_sales_new$sale_price, times = 1, p = 0.2, list
train <- nyc_rolling_sales_new[-test_index,]

temp <- nyc_rolling_sales_new[test_index,]
test <- temp %>%
  semi_join(train, by = "borough") %>%
  semi_join(train, by = "neighborhood") %>%
  semi_join(train, by = "building_class_category") %>%
  semi_join(train, by = "tax_class_at_present") %>%
  semi_join(train, by = "building_class_at_present") %>%
  semi_join(train, by = "zip_code") %>%
  semi_join(train, by = "year_built") %>%
  semi_join(train, by = "tax_class_at_time_of_sale") %>%
  semi_join(train, by = "building_class_at_time_of_sale")

removed <- anti_join(temp, test)
```

```
## Joining, by = c("borough", "neighborhood", "building_class_category", "tax_class_at_present", "buildi
```

```
train <- rbind(train, removed)

rm(test_index, temp, removed)
```

We choose linear model (lm) and decision tree (rpart) for our task here.

```
lmfit <- lm(sale_price~., data = train)
rpartfit <- rpart(sale_price~., data = train)
```

We will also be using random forest (rf) for our modelling. However, as random forest cannot deal with factor valiables with levels of more than 53, we will be creating separate train and test sets here after removing certain factor valiables and converting zip code and year built into integers.

```
nyc_rolling_sales_rf <- nyc_rolling_sales_new %>%
  mutate(neighborhood = NULL,
         zip_code = as.integer(zip_code),
         year_built = as.integer(year_built),
         building_class_at_present = NULL,
         building_class_at_time_of_sale = NULL,
         borough = NULL,
         tax_class_at_present = NULL) %>%
  filter(zip_code != 0)

set.seed(1, sample.kind = "Rounding")
test_index <- caret::createDataPartition(y = nyc_rolling_sales_rf$sale_price, times = 1, p = 0.2, list =
train_rf <- nyc_rolling_sales_rf[-test_index,]
temp <- nyc_rolling_sales_rf[test_index,]

test_rf <- temp %>%
  semi_join(train_rf, by = "building_class_category") %>%
  semi_join(train_rf, by = "tax_class_at_time_of_sale")

removed <- anti_join(temp, test_rf)
```

```
## Joining, by = c("building_class_category", "zip_code", "gross_square_feet", "year_built", "tax_class_
```

```
train_rf <- rbind(train_rf, removed)

rm(test_index, temp, removed)

rffit <- randomForest(sale_price~., data = train_rf)
```

3 Results

Having built the models, we will now look R-squared value of the linear model.

```
summary(lmfit)$r.squared
```

```
## [1] 0.72
```

```
summary(lmfit)$adj.r.squared
```

```
## [1] 0.71
```

We also look at the RMSEs as well as RMSE as a multiple of standard deviation for all the models. The root mean square error, or RMSE, is a measure of deviation of the values predicted by a model from the observed values.

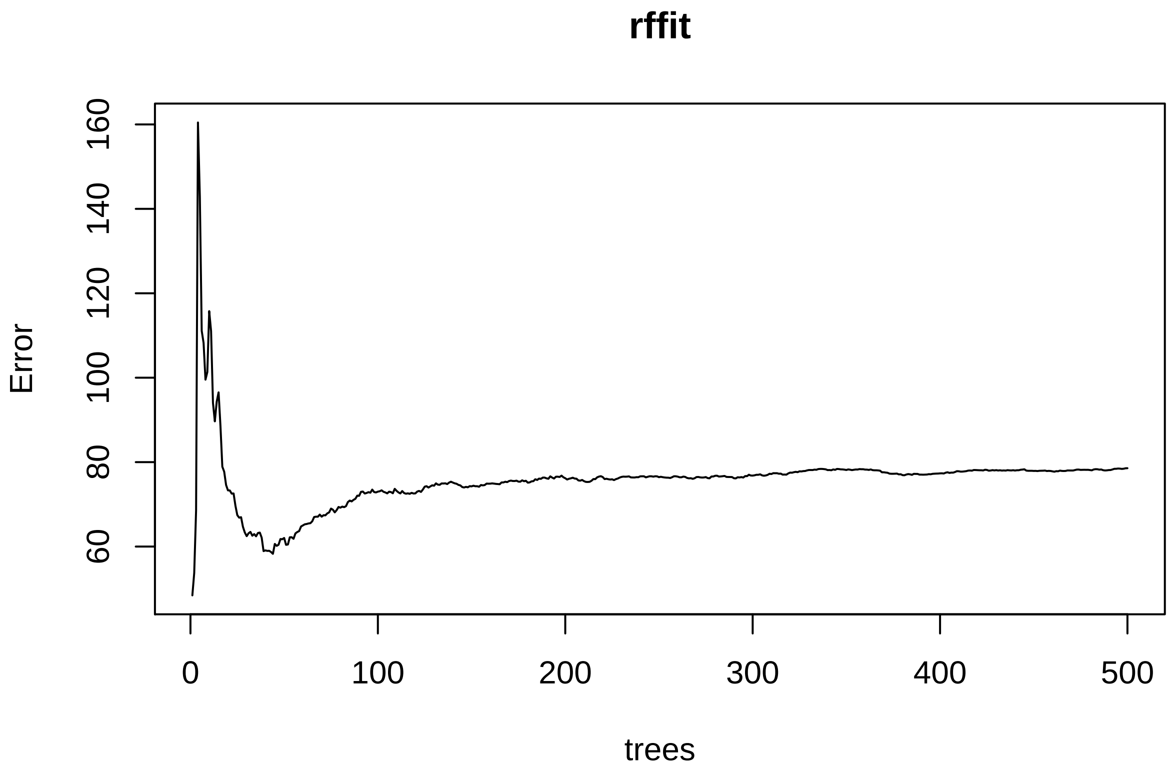
$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}$$

```
rmse <- function(pred, act){
  if(length(pred) == length(act)){
    sqrt(sum((pred - act)^2)/length(act))} else{
    print("Lengths of vectors do not match")
  }
}

rmse_lm <- rmse(predict(lmfit, test),test$sale_price)
rmse_rpart<- rmse(predict(rpartfit, test),test$sale_price)
rmse_rf <- rmse(predict(rffit, test_rf),test_rf$sale_price)

rmse_all <- tibble(RMSE = c(rmse_lm, rmse_rpart, rmse_rf))
rmse_all <- tibble(cbind(data.frame(model = c("lm", "rpart", "rf")),
                         rmse_all,
                         data.frame(RMSE_by_SD = c(
                                        rmse_all$RMSE[1]/sd(test$sale_price),
                                        rmse_all$RMSE[2]/sd(test$sale_price),
                                        rmse_all$RMSE[3]/sd(test_rf$sale_price)))))
rmse_all
```

```
## # A tibble: 3 x 3
##   model  RMSE RMSE_by_SD
##   <chr> <dbl>      <dbl>
## 1 lm     4.49      0.764
## 2 rpart  3.94      0.670
## 3 rf     3.68      0.625
```

We note that we have the least RMSE with random forest at as well as the least RMSE as a multiple of standard deviation. The evolution of errors with respect to the number of forests is as follows.

```r
plot(rffit)
```



4 Conclusion

We conclude that from the 3 basic models we have built, random forest can be chosen for application. However, certain codes could not be tested in this analysis, especially bootstraping, including more variables in the random forest model and KNN model. we note that testing these mothods may prove to be fruitful for better results.