STAT 444/844 Winter 2024 Final Project Report

Chase(Daolin) An

UW ID: 20885166

Contents

1 Data description

The original data contains 6000 observations, and it contains 27 variables with the response being housing prices in a specific area(probably around New York). The remaining variables contain housing-related information such as the number of rooms, location, neighborhood ID, and condition of houses. Both numeric and categorical values are present in the data, and some columns have varying sizes of missing values.

The Response:

- price: the price of a house

Predictors:

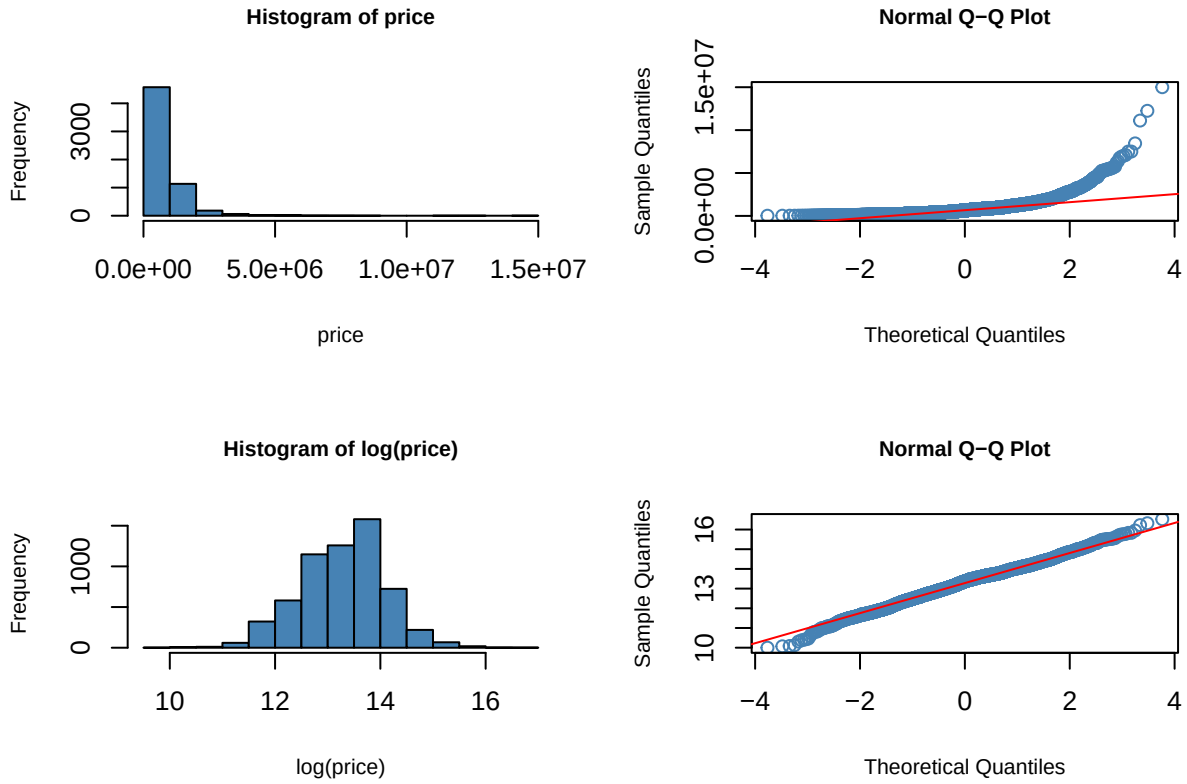
- bathrm: Number of bathrooms, (numeric)
- hf_bathrm: Number of half bathrooms, (numeric)
- heat: Heating, (categorical)
- ac: Air conditioning, (categorical)
- rooms: Number of rooms, (numeric)
- bedrm: Number of bedrooms, (numeric)
- ayb: The earliest time the main portion of the building was built, (Date)
- yr_rmdl: Year structure was remodelled, (Date)
- eyb: The year an improvement was built, (Date)
- stories: Number of stories in primary dwelling, (numeric)
- saledate: Date of sale, (Date)
- gba: Gross building area in square feet, (numeric)
- style: Style of a house, (categorical)
- grade: Evaluation of a house, (categorical)
- cndtn: Condition of a house, (categorical)
- extwall: Exterior wall type, (categorical)
- roof: Roof type, (categorical)
- intwall: Interior wall type, (categorical)
- kitchens: Number of kitchens, (numeric)
- fireplaces: Number of fireplaces, (numeric)
- landarea: Land area of property in square feet, (numeric)
- latitude: Latitude of a house, (numeric)
- longitude: Longitude of a house, (numeric)
- nbhd: Neighborhood ID, (categorical)
- ward: Ward ID, (categorical)
- quadrant: Division of an area, (categorical)

1.1 Number of missing values

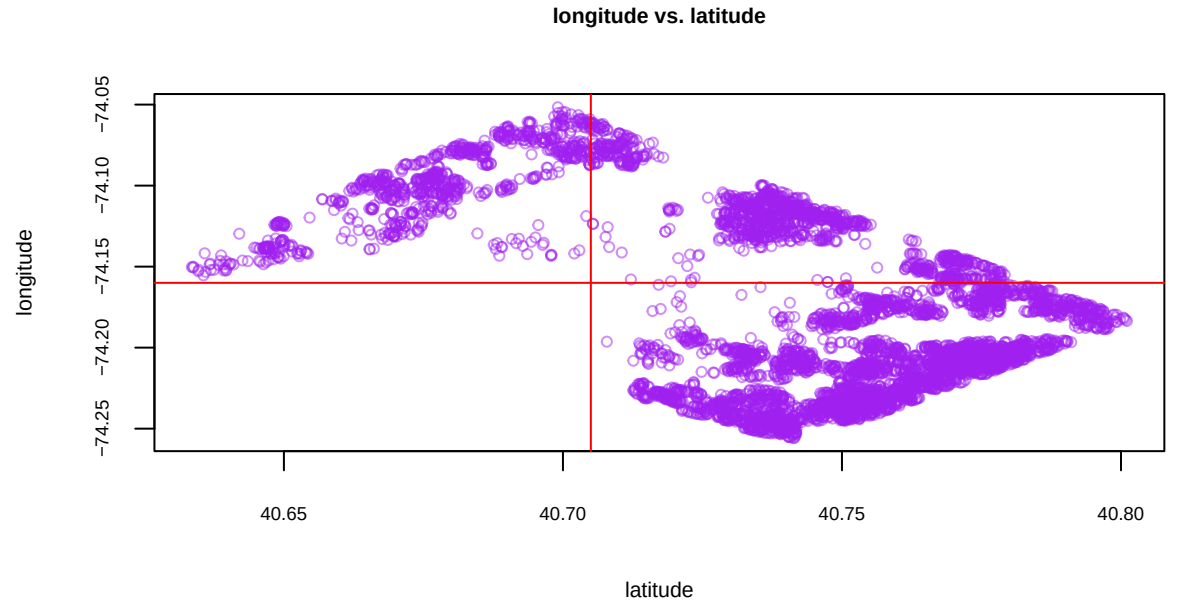
Table 1: Missing values

Predictor	Quantity
kitchens	1
stories	4
ayb	17
quadrant	32
yr_rmdl	2410

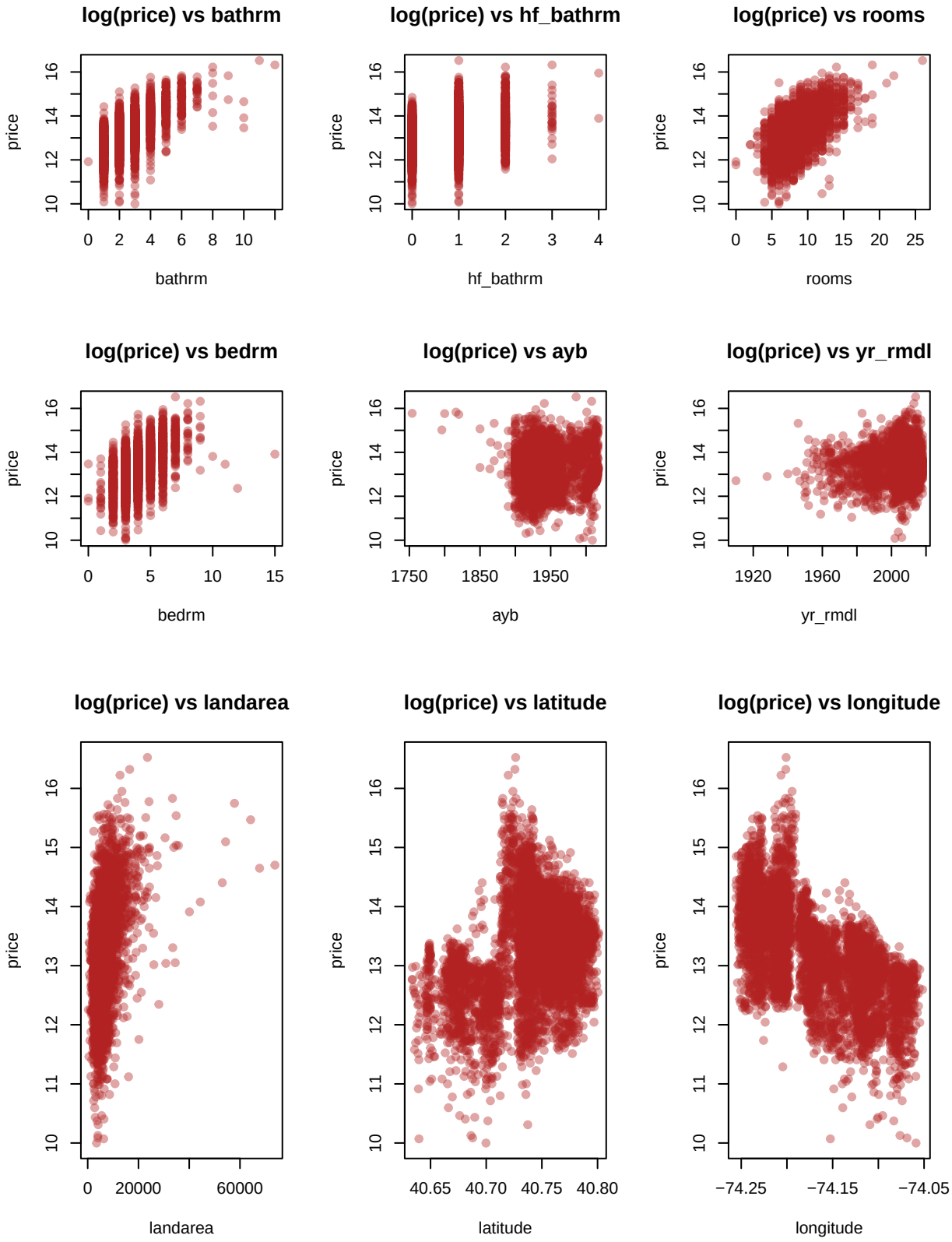
1.2 Histograms and Q-Q plots of the response and log(response)

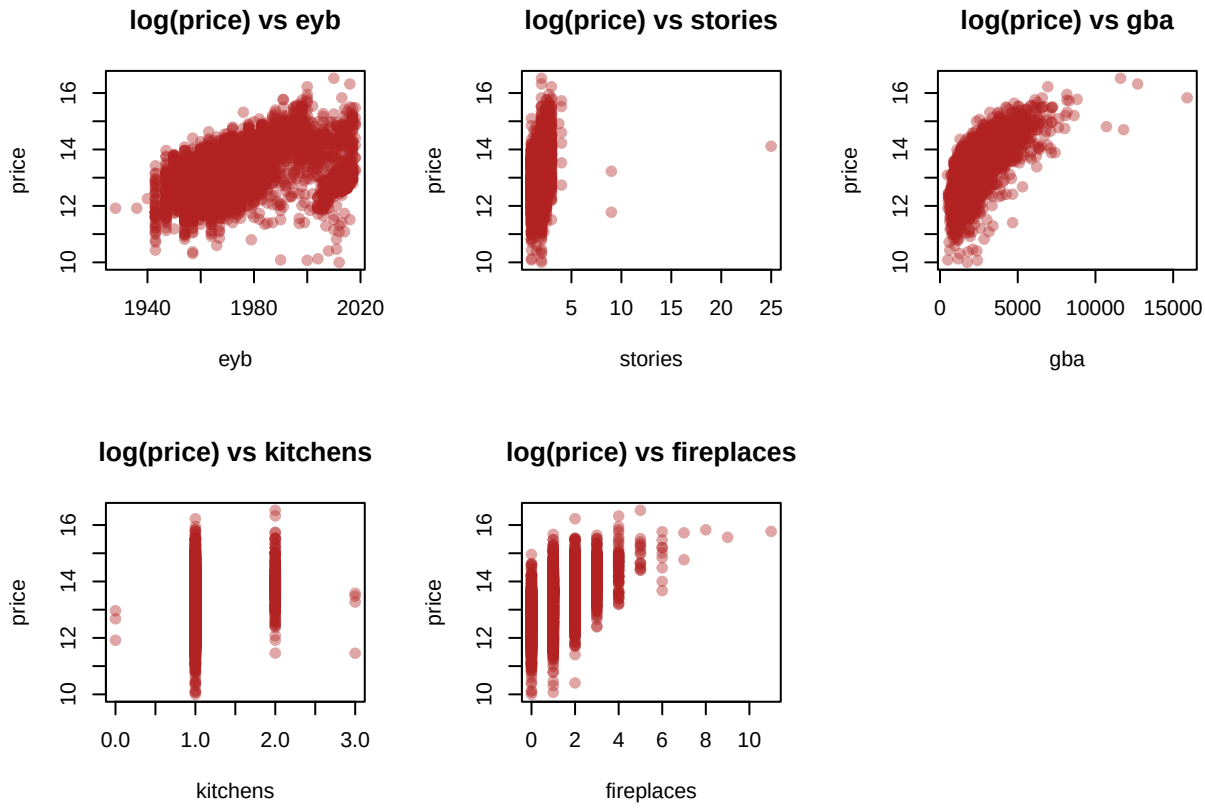


1.3 Location: latitude and longitude

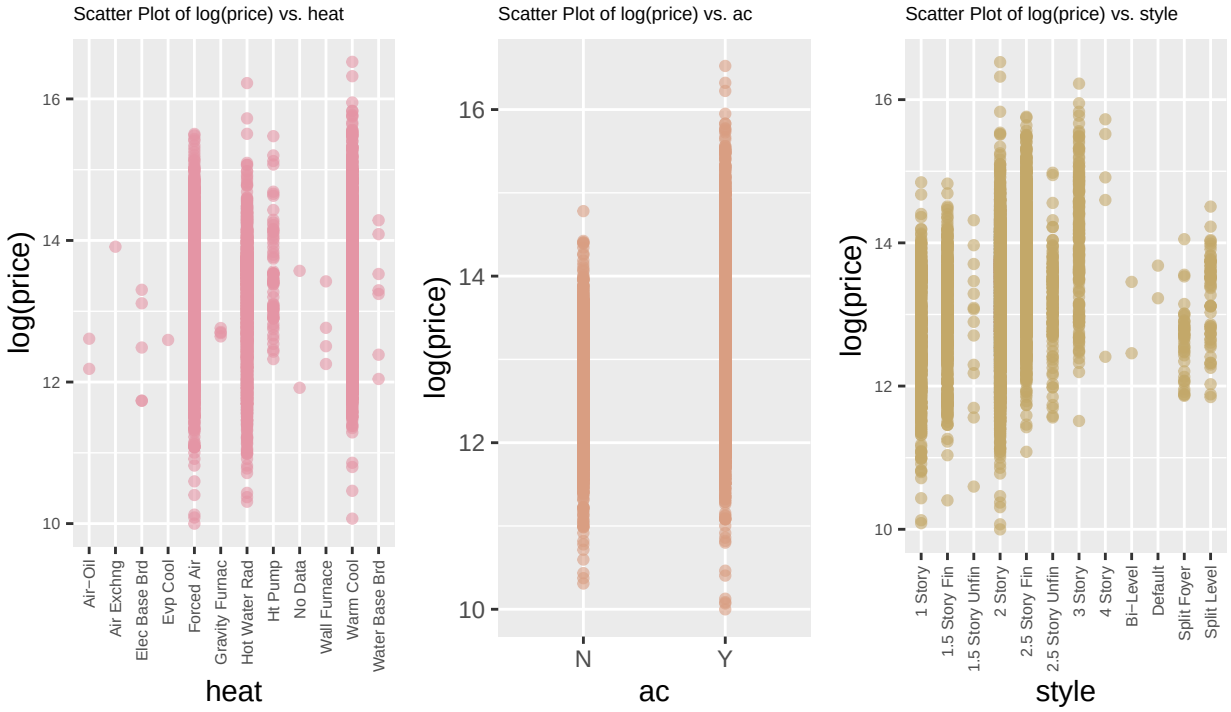


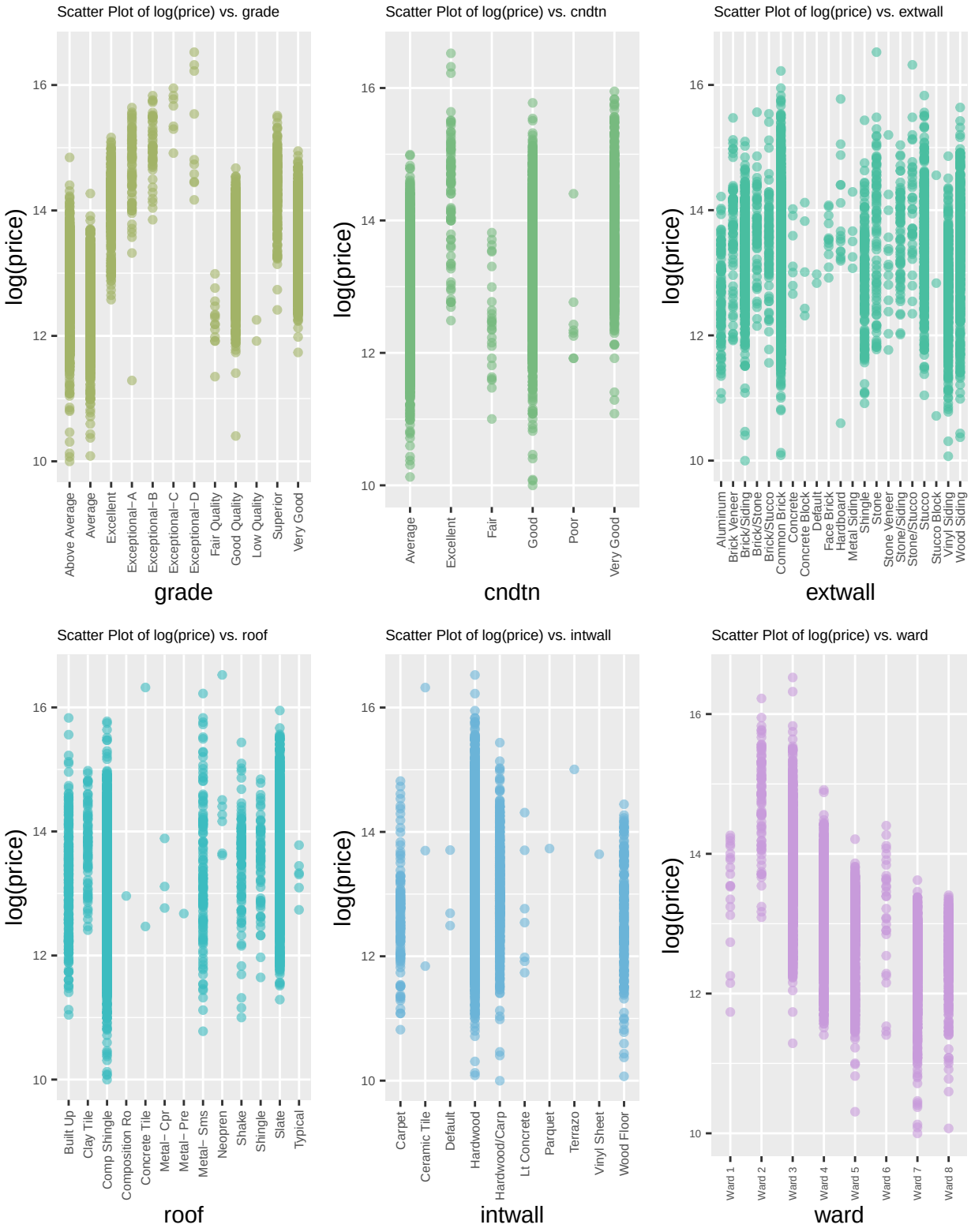
1.4 log(Response) vs. Numeric predictors scatterplots

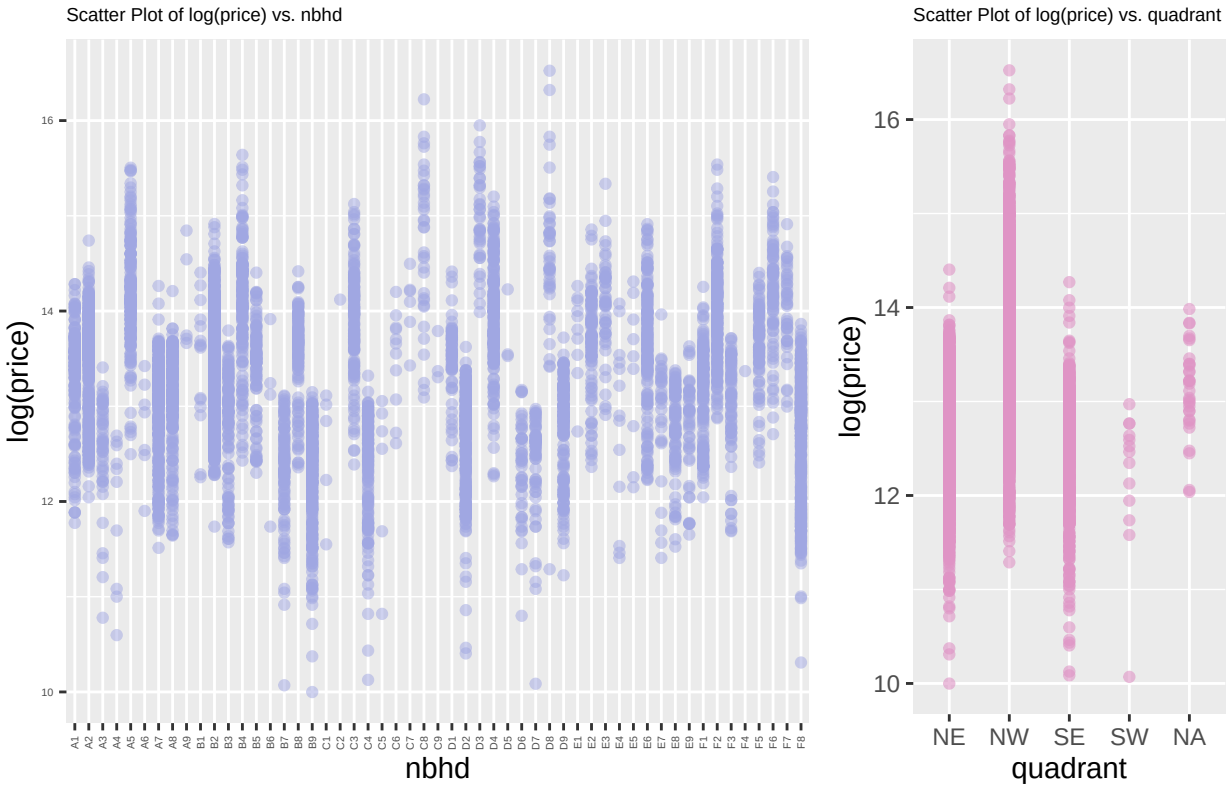




1.5 Plots for price vs categorical predictors




Scatter Plot of log(price) vs. grade
log(price)
16
14
12
10
Above Average
Average
Excellent
Exceptional−A
Exceptional−B
Exceptional−C
Exceptional−D
Fair Quality
Good Quality
Low Quality
Superior
Very Good
grade
Scatter Plot of log(price) vs. cndtn
log(price)
16
14
12
10
Average
Excellent
Fair
Good
Poor
Very Good
cndtn
Scatter Plot of log(price) vs. extwall
log(price)
16
14
12
10
Aluminum
Brick Veneer
Brick/Siding
Brick/Stone
Brick/Stucco
Common Brick
Concrete
Concrete Block
Default
Face Brick
Hardboard
Metal Siding
Shingle
Stone
Stone Veneer
Stone/Siding
Stone/Stucco
Stucco
Stucco Block
Vinyl Siding
Wood Siding
extwall
Scatter Plot of log(price) vs. roof
log(price)
16
14
12
10
Built Up
Clay Tile
Comp Shingle
Composition Ro
Concrete Tile
Metal− Cpr
Metal− Pre
Metal− Sms
Neopren
Shake
Shingle
Slate
Typical
roof
Scatter Plot of log(price) vs. intwall
log(price)
16
14
12
10
Carpet
Ceramic Tile
Default
Hardwood
Hardwood/Carp
Lt Concrete
Parquet
Terrazo
Vinyl Sheet
Wood Floor
intwall
Scatter Plot of log(price) vs. ward
log(price)
16
14
12
10
Ward 1
Ward 2
Ward 3
Ward 4
Ward 5
Ward 6
Ward 7
Ward 8
ward



1.6 Tables

Table 2: Number of bathrm and hf_bathrm combinations

	0	1	2	3	4
0	1	0	0	0	0
1	414	494	95	2	0
2	713	1134	275	8	0
3	524	1120	147	4	0
4	152	500	70	4	0
5	23	165	35	4	1
6	4	49	26	0	0
7	0	16	5	2	0
8	0	2	3	0	1
9	0	0	2	0	0
10	1	1	1	0	0
11	0	1	0	0	0
12	0	0	0	1	0

Table 3: Number of rooms and bedrm combinations

	0	1	2	3	4	5	6	7	8	9	10	11	12	15
0	2	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	2	0	0	0	0	0	0	0	0	0	0	0
3	0	4	0	3	0	0	0	0	0	0	0	0	0	0
4	0	9	30	8	4	0	0	0	0	0	0	0	0	0
5	0	3	150	85	23	5	0	0	0	0	0	0	0	0
6	0	2	116	904	196	23	5	0	0	0	0	0	0	0
7	0	1	50	745	433	90	7	1	0	0	0	0	0	0
8	0	0	9	403	545	166	30	0	0	0	0	0	0	0
9	0	0	5	164	317	161	59	12	1	0	0	0	0	0
10	0	0	1	65	194	194	67	14	2	0	0	0	0	0
11	0	0	0	17	68	95	51	14	2	2	0	0	0	0
12	1	0	0	6	48	78	60	20	2	2	0	0	0	0
13	0	0	0	1	10	36	41	12	3	1	0	0	0	0
14	0	0	0	0	4	12	17	8	3	0	0	0	1	0
15	0	0	0	1	3	12	17	6	4	2	0	0	0	0
16	0	0	0	0	0	3	6	1	1	0	0	0	0	0
17	0	0	0	0	0	0	3	3	1	0	1	1	0	0
18	0	0	0	0	0	1	0	1	1	0	0	0	0	0
19	0	0	0	0	0	0	2	0	0	1	0	0	0	1
21	0	0	0	0	0	0	0	0	1	0	0	0	0	0
22	0	0	0	0	0	0	0	0	1	0	0	0	0	0
26	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Table 4: Number of stories and style combinations

	1 Story	1.5 Story Fin	1.5 Story Unfin	2 Story	2.5 Story Fin	2.5 Story Unfin	3 Story	4 Story	Bi-Level	Default	Split Foyer	Split Level
1	595	7	0	5	1	0	2	0	1	0	33	28
1.25	22	50	4	0	0	0	0	0	0	0	0	0
1.5	0	301	10	7	1	0	0	0	0	0	0	4
1.75	0	17	0	176	1	0	0	0	0	0	0	0
2	6	5	0	3150	14	2	4	1	1	0	10	28
2.2	0	0	0	0	1	0	0	0	0	0	0	0
2.25	0	0	0	65	235	58	2	0	0	0	0	0
2.5	0	0	0	24	860	55	2	0	0	1	0	0
2.7	0	0	0	0	1	0	1	0	0	0	0	0
2.75	0	0	0	0	18	1	60	0	0	0	0	0
3	0	0	0	7	2	0	107	0	0	0	0	0
3.75	0	0	0	0	0	0	0	1	0	0	0	0
4	0	0	0	2	0	0	1	3	0	0	0	0
9	0	0	0	1	0	0	0	0	0	1	0	0
25	0	0	0	0	1	0	0	0	0	0	0	0

The response is not normally distributed as can be seen from 1.2 but the log of it is. Also notice there is no sale for longitude smaller than -74.16 and latitude smaller than 40.705. Most of the houses are built within the 110 year since they were build. The avegrage housing price in nbhd “A5” and quadrant “NW” seem to be higher than the average price others.

2 Statistical analysis

2.1 Analysis on the data

2.1.1 Adding new variables

- Since column yr_rmdled contains almost half of missing values, an indicator variable, rmdled, is added to indicate if the yr_rmdled on the same row is NA.
- Based on the plot above(1.3), four areas can be identified using the red solid lines. A categorical variable, Location, is used to identify the four areas.

```
set.seed(20885166) # For later use
dat$rmdled <- ifelse(is.na(dat$yr_rmdl), "N", "Y")
dat$location[which(dat$latitude >= 40.705 & dat$longitude >= -74.16)] <- "A"
dat$location[which(dat$latitude <= 40.705 & dat$longitude >= -74.16)] <- "S"
dat$location[which(dat$latitude <= 40.705 & dat$longitude <= -74.16)] <- "T"
dat$location[which(dat$latitude >= 40.705 & dat$longitude <= -74.16)] <- "C"
```

2.1.2 Changing the format of dates(years -> days)

- The original data contains dates that are not in the same format so all of the dates are changed to numeric values in days.

```
dat$ayb <- as.numeric(as.Date(as.character(dat$ayb), format = "%Y"))
dat$yr_rmdl <- as.numeric(as.Date(as.character(dat$yr_rmdl), format = "%Y"))
dat$eyb <- as.numeric(as.Date(as.character(dat$eyb), format = "%Y"))
dat$saledate <- as.numeric(as.Date(dat$saledate))
```

2.1.3 Missing data handling

- Numeric columns, ayb, stories, and kitchens: all have a small number of missing values, so a simple imputation method, mean of the non-NA values of their corresponding column, is used.
- Categorical column, quadrant: also has a small number of missing values, and its average cannot be determined. Therefore, mode is used for imputation.
- Special case, yr_rmdl: NA values of this column could indicate two things – the house is never remodeled, or the information is lost. Using mean to impute this column is somewhat inappropriate, as remodeling date cannot be larger than the date the house was first built. So to lessen its effect, the ayb value at the same row of NAs in yr_rmdl column is used as replacement.

```
dat$ayb[which(is.na(dat$ayb))] <- mean(dat$ayb, na.rm = TRUE)
dat$stories[which(is.na(dat$stories))] <- mean(dat$stories, na.rm = TRUE)
dat$kitchens[which(is.na(dat$kitchens))] <- mean(dat$kitchens, na.rm = TRUE)
quadrant.mode <- names(sort(table(dat$quadrant), decreasing = TRUE))[1]
dat$quadrant[which(is.na(dat$quadrant))] <- quadrant.mode
dat$yr_rmdl[which(is.na(dat$yr_rmdl))] <- dat$ayb[is.na(dat$yr_rmdl)]
```

2.1.4 Again new variables are added

- It is commonly understood that saledate has a significant impact on the price of a house. Now saledates are available, but they do not represent the "age" of a house. Three variables, dsRM, dsIM, and dsBT, are created to represent the "ages" of the houses.

```
dat$dsRM <- dat$saledate - dat$yr_rmdl
dat$dsIM <- dat$saledate - dat$eyb
dat$dsBT <- dat$saledate - dat$ayb
```

2.1.5 Transformations

- Apparently from plot 1.2, the histogram of the response is heavily right skewed, indicating normality is violated while the histogram of the log response is kind of symmetric. A logarithmic transformation on the response is taken.
- Also notice the log(price) vs landarea plot from 1.4. There is no observable pattern in landarea so its values are square rooted.

```
dat$price <- log(dat$price)
dat$landarea <- sqrt(dat$landarea)
```

2.1.6 Extreme values handling

- From 1.4 log(price) vs kitchens plot, there are a few rare kitchens values.
- From 1.4 log(price) vs fireplaces plot, there are five rare fireplaces values.
- From Table 2, there are three rare combinations of bathrm and hf_bathrm.
- From Table 3, there are a few rare combinations of rooms and bedrm.
- From Table 4, there are nine rare combinations of stories and style.

For cases with unreasonable values which could be mis-recorded, closest matches of their values are used to replace their original value.

```
dat$kitchens[which(dat$kitchens < 1)] <- 1
dat$kitchens[which(dat$kitchens > 2)] <- 2
dat$fireplaces[which(dat$fireplaces > 6)] <- 6
dat$bathrm[which(dat$bathrm==11 | dat$bathrm==12)] <- 10
dat$hf_bathrm[which(dat$hf_bathrm==4)] <- 3
dat$rooms[which(dat$rooms > 17)] <- 17
dat$rooms[which(dat$rooms < 3)] <- 3
dat$bedrm[which(dat$bedrm > 9)] <- 9
dat$bedrm[which(dat$bedrm < 1)] <- 1
dat$stories[which(dat$stories > 3)] <- 3
```

2.2 The Models for Each Method

2.2.1 Smoothing

The final model is:

```
finalSM.formula <- formula(price ~ s(bathrm, k=10) + s(rooms, k=10) +
                                   s(ayb, k=20) + s(yr_rmdl, k=20) +
                                   s(eyb, k=65) + s(saledate, k=24) +
                                   s(gba, k=10) + s(landarea) + s(latitude) +
                                   s(longitude, k=10) + s(dsRM) + s(dsIM) +
                                   s(dsBT) + hf_bathrm + bedrm + fireplaces +
                                   heat + ac + style + grade + cndtn + roof +
                                   intwall + nbhd + ward + quadrant +
                                   location + te(saledate, ayb, k=9) +
                                   te(saledate, eyb, k=8) +
                                   te(saledate, yr_rmdl, k=9) +
                                   te(landarea, longitude) +
                                   te(landarea, latitude))
finalSM <- mgcv::gam(finalSM.formula, data=dat)
```

This smoothing model is quite different from the one I used in Project Smoothing. **Firstly**, the method used in data pre-processing, and the training data are different. **Secondly**, smooth terms s() are added to every

variables that can be added with the optimal k determined by grid-searching. **Lastly**, different interactions terms are added.

2.2.2 Random Forest

The final model is:

```
finalRF.formula <- formula(price ~ heat + ac + style + grade + cndtn + extwall +
                                   roof + intwall + nbhd + ward + quadrant +
                                   location + rmdled + bathrm + hf_bathrm +
                                   rooms + bedrm + ayb + yr_rmdl + eyb +
                                   stories + saledate + gba + fireplaces +
                                   landarea + latitude + longitude +
                                   dsRM + dsIM + dsBT)
finalRF <- randomForest::randomForest(finalRF.formula, mtry=16,
                                      nodesize=1, ntree=400, data=dat)
```

This RF model is very similar to the RF model that was used in Project Random Forest, and the model building process is similar. However both optimal nodesizes and mtrys are different, since **ntree** parameter is also tuned this time. The other disparities lie within how 5-fold CVs are performed, and how the data is pre-processed.

Main difference in 5-fold CVs:

- **Project Random Forest**: Random folds were used.
- **Project Final**: Fixed folds were used.

Main difference in methods used for pre-processing the data:

- **Project Random Forest**: Outliers were removed.
- **Project Final**: Outliers were replaced with closest matches.

2.2.3 Boosting

gbm::gbm() function requires categorical variables to be treated as factors explicitly, so a copy of dat is made and the categorical variables are treated as factors manually. No other changes are made.

```
dat.copy <- dat
dat.copy$heat <- as.factor(dat.copy$heat)
dat.copy$ac <- as.factor(dat.copy$ac)
dat.copy$style <- as.factor(dat.copy$style)
dat.copy$grade <- as.factor(dat.copy$grade)
dat.copy$cndtn <- as.factor(dat.copy$cndtn)
dat.copy$extwall <- as.factor(dat.copy$extwall)
dat.copy$roof <- as.factor(dat.copy$roof)
dat.copy$intwall <- as.factor(dat.copy$intwall)
dat.copy$nbhd <- as.factor(dat.copy$nbhd)
dat.copy$ward <- as.factor(dat.copy$ward)
dat.copy$quadrant <- as.factor(dat.copy$quadrant)
dat.copy$rmdled <- as.factor(dat.copy$rmdled)
dat.copy$location <- as.factor(dat.copy$location)
```

The final Boosting model is then:

```
finalBT.formula <- formula(price ~ heat + ac + style + grade + cndtn +
                                   extwall + roof + intwall + nbhd + ward +
                                   quadrant + location + rmdled + bathrm +
                                   hf_bathrm + rooms + bedrm + ayb +
                                   yr_rmdl + eyb + stories + saledate + gba +
```

```
                                    kitchens + fireplaces + landarea +
                                    latitude + longitude + dsRM + dsIM + dsBT)
finalBT <- gbm::gbm(formula=finalBT.formula, distribution="gaussian",
                    bag.fraction=1, shrinkage=0.08,
                    interaction.depth=10, n.trees=500, data=dat.copy)
```

2.3 Comparison summary

Table 5: Comparisons

	Running time	Rough training time	5-Fold CV score
Smoothing	122.14	18000	0.1921476
Random Forest	91.51	5400	0.1980850
Boosting	7.00	570	0.1752821

- Ranking models based on their predictive accuracy(most accurate to least accurate): **Boosting** > **Smoothing** > **Random Forest**, as can be seen from Table 5.
- Ranking models based on their running time(most time-consuming to least time-consuming): **Smoothing** > **Random Forest** > **Boosting**, as can be seen from Table 5.
- Ranking models based on their ease of use(easiest to hardest) in this case: **Boosting** > **Random Forest** > **Smoothing**.
- Ranking models based on their sensitivity of outliers(most sensitive to least sensitive): **Smoothing** > **Random Forest** > **Boosting**.

2.4 Predictive accuracy

Let $RMLSE_{avg}$ be the 5-Fold CV score, p_{t_i} be the natural log of real housing price of testing set of fold i where i = 1, 2, ..., 5, and p_{e_i} be the natural log of estimated housing price of testing set in the same fold. Moreover, let $p_{t_{ij}}$ be the j-th real log housing price, and $p_{e_{ij}}$ be the j-th estimated log housing price.

Then the 5-Fold CV score is calculated by:

$$RMLSE_{avg} = \sqrt{\frac{1}{5}\sum_{i=1}^{5}\sum_{j=1}^{1200}(p_{t_{ij}} - p_{e_{ij}})^2}$$

It is clear that the 5-Fold CV score of the final smoothing model is a bit lower than that of the final random forest model(table 5), indicating that the predictions using the final smoothing model is more accurate than those using the final random forest model, while the boosting model outperforms the smoothing model.

2.5 Computational complexity and runtime

- Note when estimating training time, time to perform 5-Fold CV is taken into account.
- **Smoothing model**: The computational complexity increases significantly as more smooth terms and interaction terms are added to the model. It is even more time-consuming grid-searching on optimal k values of smooth terms and interaction terms. Non-default optimal k values for terms: **ayb**:20, **yr_rmdl**:20, **eyb**:65, **saledate**:24, **saledate & ayb**:9, **saledate & yr_rmdl**:8, **saledate & yr_rmdl**:9.
- **Random Forest model**: The computational complexity for random forest models is affected by the number of leaf nodes, so the default random forest model is more complex than the default smoothing model. However, it is also very time-consuming to determine the optimal mtry and nodesize. Optimal values: **mtry** = 16, **nodesize** = 1, **ntree** = 400.

- **Boosting model**: During modelling, it requires the least time among the three models to train though multiple parameter were tuned. Optimal values: **shrinkage** = 0.08, **n.trees** = 500, **interaction.depth** = 10. Note **bag.fraction** is set to 1 to avoid randomness.

2.6 Ease of use/model building

- The data used to train the models are the same and same pre-processing methods(Section 2.1) are used except that the categorical values are treated explicitly as factors for Boosting.
- The 5-Fold CV function for the smoothing model is more complex to write than those for the random forest model and boosting model since level-mismatches need to be handled manually.(See Appendix)
- Other than the 5-Fold CV functions, the random forest model is easier to use than the smoothing model since it requires less parameters to be tuned. Boosting model is just as simple as random forest model but the speed of tuning parameters is much faster in our case.
- Pre-processing is more necessarily needed when building a smoothing model compared to the other models(though, in my case, same pre-processing techniques are used).
- The time required to train the smoothing model(around 100s) exceeds the time required to train the random forest model(around 90s) and the time for the boosting model(around 10s).

Model building details can be found in 20885166-Supp.pdf or 20885166 Supp.Rmd.

2.7 Interpretation

- Given that a log transformation is applied on the price, predictions using the predict() function on new data set will also be in log form. Use exp(predict()) to get the estimated prices in their original scale.
- **Smoothing model**: The smoothing model contains several smooth terms, indicating there exist non-linear relationships between several predictors and the response. Their relationships can be summarized by smooth functions determined by the gam() function. Moreover, there are interaction terms meaning some predictor combinations jointly influence the estimation of price more than they do individually.
- **Random Forest model**: It is hard to determine the relationship between predictors and the response for the random forest model since it involves multiple decision trees. However when the forest includes 400 trees and each tree makes 16 splits at each step, the random forest model yields more accurate predictions compared to its counterpart without such configuration.
- **Boosting model**: The model performs well with a learning rate 0.08, and a tree number of 500. Learning rate being 0.08 implies that each tree in the forest makes a small contribution to the forest(final result), and the forest consist of 500 trees. Moreover, the setting of interaction.depth=10 suggests the presence of significant interaction effects among predictors.

Both the smoothing model and boosting model suggest there exist strong relationships among predictors, especially among housing configuration variates and location/date variates. Besides, saledate and nbhd are two most important individual factors that affect the housing prices in this area.

2.8 Sensitivity to outliers

The inherent feature of tree-based methods, like random forest and boosting, is that it is generally more robust to outliers compared to other models, such as smoothing models.

Table 6: Comparison of CV scores(with/without outliers)

	Without outliers	With outliers
Smoothing	0.1921476	0.1929056
Random Forest	0.1980850	0.1983868
Boosting	0.1752821	0.1752821

As depicted in Table 6, the increase in 5-Fold CV scores is more pronounced in the smoothing model compared to the random forest model. Interestingly, the boosting model appears to be unaffected by outliers.

2.9 Insights

2.9.1 Important interactions

- **saledate & eyb**
- **saledate & yr_rmdl**
- **saledate & ayb**
- **landarea & longitude**
- **landarea & latitude**

The above terms contribute most to the increment of predictive power of my models. Note all significant interaction term combinations are related to date(saledate, eyb, yr_rmdl, ayb) and location(latitude/longitude), wihch matches our common sense in real life.

2.9.2 Variable importance

Significant variables in the **Smoothing model**:

- Non-smooth variables: hf_bathrm, bedrm, fireplaces, heat, ac, style, grade, cndtn, roof, intwall, nbhd, and ward.
- Smooth variables: bathrm, rooms, eyb, saledate, gba, landarea, latitude, longitude, dsRM, and dsIM.
- Interaction variables: saledate & ayb, saledate & eyb, saledate & yr_rmdl, landarea & longitude, and landarea & latitude.

Significant variables(compared to others) in the **Random Forest model**: saledate, nbhd, longitude, gba, grade, ward, bathrm, eyb, yr_rmdl, latitude, dsBT, dsRM, dsIM, and landarea.

Significant variables(compared to others) in the **Boosting model**: saledate, nbhd, longitude, gba, grade, dsBT, cndtn, dsRM, bedrm, extwall, and latitude.

As demonstrated in Appendix B, the significant predictors can be ranked in terms of importance as follows: saledate > nbhd > longitude > gba > grade > (others). The rest of the variables seems not as significant as the previous ones, but they still contribute to the overall predictive power of the models to some extent. They may provide useful interaction effects when they are combined with other terms, such as **landarea & latitude**.

In summary, variables related to **location** and **date** significant influence the predictive power of the models so they are the most significant. On the other hand, variables that indicate housing configurations, like kitchens and extwall, have relatively less importance.

2.9.3 Outliers

Notable outliers in predictors:

- There are only about 7 houses constructed before 1850.
- Only 2 houses have 4 half bathrooms.
- There are about 10 houses with 17 more rooms.
- Only 4 houses have more than 10 bedrooms.
- Only 3 houses were remodeled before 1940.
- There are about 8 houses with over 40000 square foot land area.
- Only 3 houses have more than 4 stories.
- It is rather unusual that there is only 1 house each in both neighborhood IDs C2 and F4.

2.9.4 Exploratory Data Analysis (EDA)

From a seller's viewpoint, is it advisable to renovate the house before selling it?

```
mean(exp(dat$price)) # Average housing price
```

```
## [1] 803309.6
```

```r
N <- mean(exp(dat$price[dat$rmdled=="N"])); N # Average housing price - Not Remodeled
```

```
## [1] 648998.9
```

```r
Y <- mean(exp(dat$price[dat$rmdled=="Y"])); Y # Average housing price - Remodeled
```

```
## [1] 906899.7
```

```r
(Y-N)/N # Higher in percentage
```

```
## [1] 0.3973825
```

The results show that, on average, the price of houses that are remodeled is significantly, about 40 percent, higher than that of non-remodeled houses. This significant difference marks the potential benefits of renovating a house before selling it, but renovating costs. As housing prices are affected by multiple factors, other factors likes location and land area should be also taken into account to comprehensively understand the local housing market.

From a buyer's point of view, it is advised to buy a house earlier as houses lately are more expensive on average than houses bought earlier.

3 Conclusions

To summarize, location and date are the two most important factors that affect the housing price in this area. Generally, the average house prices of houses located at northwest of this specific area are higher than the prices of houses at the other areas. The housing prices show an increasing trend at this area during the period the data was recorded since the average prices of houses sold later is higher than those of houses sold earlier. Also the smoothing model indicates that some factors together, like the combination of location and land area, also influence the prices to a large extent. Moreover, though the models I have built may provide some insights to the housing prices at this area, they may not capture all the crucial aspects. Lastly, it surprised me a lot that the boosting model is so efficient that it not only outperforms the other two models, but requires the least training time as well in this case.

4 Appendices

4.1 Appendix A: 5-Fold Cross-Validation Function For Each Method

4.1.1 For Smoothing

```
CV.SM <- function(formula) {
  RMLSEs <- numeric(5)
  for(i in 1:5) {
    testing <- dat[which(fold==i), ]
    training <- dat[which(fold!=i), ]
    if (i == 1) {
      testing$roof[testing$roof=="Composition Ro"] <- "Comp Shingle"
      testing$intwall[testing$intwall=="Terrazo"] <- "Parquet"
      testing$intwall[testing$intwall=="Vinyl Sheet"] <- "Parquet"
    } else if (i == 2) {
      testing$heat[testing$heat=="Air Exchng"] <- "Ht Pump"
      testing$nbhd[testing$nbhd=="C2"] <- "C3"
    } else if (i == 3) {
      testing$style[testing$style=="Default"] <- "2 Story"
      testing$roof[testing$roof=="Metal- Pre"] <- "Composition Ro"
    } else if (i == 4) {
      testing$extwall[testing$extwall=="Stucco Block"] <- "Aluminum"
      testing$intwall[testing$intwall=="Parquet"] <- "Default"
      testing$nbhd[testing$nbhd=="C2"] <- "C3"
      testing$nbhd[testing$nbhd=="F4"] <- "F5"
    } else {
      testing$heat[testing$heat=="Evp Cool"] <- "Elec Base Brd"
    }
    model <- mgcv::gam(formula, data=training)
    pred <- predict(model, newdata=testing)
    RMLSEs[i] <- sqrt(mean((testing$price - pred)^2))
  }
  mean(RMLSEs)
}
```

4.1.2 For Random Forest

```
CV.RF <- function(formula, mtry=10, nodesize=1, ntree=500) {
  RMLSEs <- numeric(5)
  for (i in 1:5) {
    testing <- dat[which(fold==i), ]
    training <- dat[which(fold!=i), ]
    model <- randomForest::randomForest(formula, mtry=mtry, ntree=ntree,
                                        nodesize=nodesize, data=training)
    pred <- predict(model, newdata=testing)
    RMLSEs[i] <- sqrt(mean((testing$price - pred)^2))
  }
  mean(RMLSEs)
}
```

4.1.3 For Boosting

```
CV.BT <- function(formula, shrinkage=0.1, n.trees=100, interaction.depth=1) {
  RMLSEs <- numeric(5)
  for(i in 1:5) {
    testing <- dat[which(fold==i), ]
    training <- dat[which(fold!=i), ]
    training$heat <- as.factor(training$heat)
    training$ac <- as.factor(training$ac)
    training$style <- as.factor(training$style)
    training$grade <- as.factor(training$grade)
    training$cndtn <- as.factor(training$cndtn)
    training$extwall <- as.factor(training$extwall)
    training$roof <- as.factor(training$roof)
    training$intwall <- as.factor(training$intwall)
    training$nbhd <- as.factor(training$nbhd)
    training$ward <- as.factor(training$ward)
    training$quadrant <- as.factor(training$quadrant)
    training$rmdled <- as.factor(training$rmdled)
    training$location <- as.factor(training$location)
    testing$heat <- as.factor(testing$heat)
    testing$ac <- as.factor(testing$ac)
    testing$style <- as.factor(testing$style)
    testing$grade <- as.factor(testing$grade)
    testing$cndtn <- as.factor(testing$cndtn)
    testing$extwall <- as.factor(testing$extwall)
    testing$roof <- as.factor(testing$roof)
    testing$intwall <- as.factor(testing$intwall)
    testing$nbhd <- as.factor(testing$nbhd)
    testing$ward <- as.factor(testing$ward)
    testing$quadrant <- as.factor(testing$quadrant)
    testing$rmdled <- as.factor(testing$rmdled)
    testing$location <- as.factor(testing$location)
    model <- gbm::gbm(formula, data=training, distribution="gaussian",
                      shrinkage=shrinkage, n.trees=n.trees,
                      interaction.depth=interaction.depth,
                      bag.fraction=1)
    pred <- predict(model, newdata=testing)
    RMLSEs[i] <- sqrt(mean((testing$price - pred)^2))
  }
  mean(RMLSEs)
}
```

4.2 Appendix B: Variable Importance

4.2.1 of Smoothing Model

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## price ~ s(bathrm, k = 10) + s(rooms, k = 10) + s(ayb, k = 20) +
##     s(yr_rmdl, k = 20) + s(eyb, k = 65) + s(saledate, k = 24) +
```

```
##     s(gba, k = 10) + s(landarea) + s(latitude) + s(longitude,
##     k = 10) + s(dsRM) + s(dsIM) + s(dsBT) + hf_bathrm + bedrm +
##     fireplaces + heat + ac + style + grade + cndtn + roof + intwall +
##     nbhd + ward + quadrant + location + te(saledate, ayb, k = 9) +
##     te(saledate, eyb, k = 8) + te(saledate, yr_rmdl, k = 9) +
##     te(landarea, longitude) + te(landarea, latitude)
##
## Parametric coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)          12.9899229  0.2849493  45.587  < 2e-16 ***
## hf_bathrm             0.0253363  0.0044849   5.649 1.69e-08 ***
## bedrm                 0.0157834  0.0034936   4.518 6.38e-06 ***
## fireplaces            0.0273406  0.0039243   6.967 3.60e-12 ***
## heatAir Exchng       -0.2288359  0.2245972  -1.019 0.308307
## heatElec Base Brd     0.0522140  0.1553488   0.336 0.736802
## heatEvp Cool         -0.8818344  0.2275380  -3.876 0.000108 ***
## heatForced Air        0.1253797  0.1300871   0.964 0.335181
## heatGravity Furnac    0.2149137  0.1615432   1.330 0.183447
## heatHot Water Rad     0.1129311  0.1301365   0.868 0.385546
## heatHt Pump           0.1459668  0.1321360   1.105 0.269349
## heatNo Data           0.1334395  0.2208972   0.604 0.545815
## heatWall Furnace      0.2192614  0.1624382   1.350 0.177130
## heatWarm Cool         0.1196164  0.1300784   0.920 0.357836
## heatWater Base Brd    0.1234468  0.1472333   0.838 0.401817
## acY                   0.0385109  0.0091731   4.198 2.73e-05 ***
## style1.5 Story Fin    0.0135289  0.0130632   1.036 0.300411
## style1.5 Story Unfin -0.0614268  0.0494717  -1.242 0.214415
## style2 Story         -0.0093005  0.0107479  -0.865 0.386890
## style2.5 Story Fin    0.0197147  0.0130056   1.516 0.129611
## style2.5 Story Unfin  0.0181774  0.0204825   0.887 0.374869
## style3 Story          0.0117824  0.0193462   0.609 0.542530
## style4 Story         -0.0927627  0.0910463  -1.019 0.308317
## styleBi-Level         0.0832190  0.1298760   0.641 0.521706
## styleDefault          0.1349634  0.1304349   1.035 0.300845
## styleSplit Foyer      0.1088354  0.0311234   3.497 0.000474 ***
## styleSplit Level      0.0096579  0.0252818   0.382 0.702469
## gradeAverage         -0.0417913  0.0127760  -3.271 0.001078 **
## gradeExcellent        0.1491190  0.0187850   7.938 2.45e-15 ***
## gradeExceptional-A    0.2665503  0.0319890   8.333  < 2e-16 ***
## gradeExceptional-B    0.4511607  0.0388183  11.622  < 2e-16 ***
## gradeExceptional-C    0.6925932  0.0880967   7.862 4.50e-15 ***
## gradeExceptional-D    0.7435792  0.0743704   9.998  < 2e-16 ***
## gradeFair Quality    -0.1363625  0.0546022  -2.497 0.012540 *
## gradeGood Quality     0.0393865  0.0119540   3.295 0.000991 ***
## gradeLow Quality     -1.0746872  0.2271466  -4.731 2.29e-06 ***
## gradeSuperior         0.2307629  0.0224510  10.279  < 2e-16 ***
## gradeVery Good        0.0800925  0.0144306   5.550 2.98e-08 ***
## cndtnExcellent        0.2748080  0.0281690   9.756  < 2e-16 ***
## cndtnFair            -0.1240552  0.0360330  -3.443 0.000580 ***
## cndtnGood             0.0743449  0.0063914  11.632  < 2e-16 ***
## cndtnPoor            -0.2404551  0.0821443  -2.927 0.003434 **
## cndtnVery Good        0.1913188  0.0104686  18.275  < 2e-16 ***
## roofClay Tile         0.0616782  0.0255142   2.417 0.015663 *
## roofComp Shingle      0.0052462  0.0139951   0.375 0.707779
```

```
## roofComposition Ro    0.0760416  0.1879803   0.405 0.685847
## roofConcrete Tile     0.1000097  0.1525931   0.655 0.512236
## roofMetal- Cpr        0.2273855  0.1117358   2.035 0.041894 *
## roofMetal- Pre       -0.0291610  0.1848438  -0.158 0.874651
## roofMetal- Sms        0.0099751  0.0225880   0.442 0.658788
## roofNeopren           0.1166104  0.0818932   1.424 0.154521
## roofShake            -0.0004256  0.0226587  -0.019 0.985016
## roofShingle           0.0076784  0.0269999   0.284 0.776124
## roofSlate             0.0118470  0.0148607   0.797 0.425365
## roofTypical          -0.0881806  0.0758713  -1.162 0.245187
## intwallCeramic Tile  -0.0728001  0.1202004  -0.606 0.544768
## intwallDefault        0.0712171  0.1157717   0.615 0.538480
## intwallHardwood       0.0703086  0.0176359   3.987 6.78e-05 ***
## intwallHardwood/Carp  0.0472386  0.0184522   2.560 0.010491 *
## intwallLt Concrete    0.2702475  0.0848011   3.187 0.001446 **
## intwallParquet        0.0317459  0.1822029   0.174 0.861688
## intwallTerrazo        0.1201655  0.1888384   0.636 0.524580
## intwallVinyl Sheet    0.0505864  0.1824539   0.277 0.781594
## intwallWood Floor     0.0271429  0.0220235   1.232 0.217832
## nbhdA2                0.0889236  0.0650908   1.366 0.171947
## nbhdA3               -0.0851749  0.0604141  -1.410 0.158638
## nbhdA4               -0.4077504  0.0718506  -5.675 1.46e-08 ***
## nbhdA5                0.1361388  0.0756038   1.801 0.071805 .
## nbhdA6               -0.2150487  0.1239104  -1.736 0.082704 .
## nbhdA7               -0.0448182  0.0320127  -1.400 0.161564
## nbhdA8               -0.1080146  0.1016651  -1.062 0.288075
## nbhdA9                0.0876183  0.2507400   0.349 0.726773
## nbhdB1                0.2921679  0.2602054   1.123 0.261555
## nbhdB2                0.0880934  0.0604720   1.457 0.145237
## nbhdB3               -0.0743418  0.0565789  -1.314 0.188916
## nbhdB4                0.3147101  0.0684664   4.597 4.39e-06 ***
## nbhdB5                0.0652563  0.0498265   1.310 0.190361
## nbhdB6               -0.3667086  0.2571980  -1.426 0.153986
## nbhdB7               -0.1708383  0.0668457  -2.556 0.010623 *
## nbhdB8                0.1000872  0.0336448   2.975 0.002944 **
## nbhdB9               -0.1659280  0.0536787  -3.091 0.002004 **
## nbhdC1               -0.0574188  0.1316644  -0.436 0.662781
## nbhdC2                0.2099255  0.3040852   0.690 0.490002
## nbhdC3                0.1512136  0.0637079   2.374 0.017651 *
## nbhdC4               -0.1196579  0.0530991  -2.253 0.024267 *
## nbhdC5               -0.4642753  0.1306991  -3.552 0.000385 ***
## nbhdC6                0.1520487  0.0932147   1.631 0.102911
## nbhdC7                0.3758106  0.0994220   3.780 0.000158 ***
## nbhdC8                0.3161093  0.2374470   1.331 0.183149
## nbhdC9                0.2827821  0.1269469   2.228 0.025949 *
## nbhdD1               -0.0140639  0.0636126  -0.221 0.825033
## nbhdD2                0.0567238  0.0623775   0.909 0.363197
## nbhdD3                0.2443411  0.2323752   1.051 0.293077
## nbhdD4                0.0543905  0.0754086   0.721 0.470769
## nbhdD5                0.1303483  0.2735935   0.476 0.633786
## nbhdD6               -0.0375850  0.0571829  -0.657 0.511030
## nbhdD7               -0.2947839  0.0557242  -5.290 1.27e-07 ***
## nbhdD8                0.2349576  0.0766102   3.067 0.002173 **
## nbhdD9               -0.2110811  0.1025176  -2.059 0.039542 *
```

```
## nbhdE1                0.0362523  0.2455257   0.148 0.882623
## nbhdE2                0.1897470  0.0661920   2.867 0.004164 **
## nbhdE3                0.2515872  0.0737133   3.413 0.000647 ***
## nbhdE4               -0.1130102  0.2580821  -0.438 0.661487
## nbhdE5                0.3170685  0.2103513   1.507 0.131782
## nbhdE6                0.1198818  0.0772729   1.551 0.120860
## nbhdE7               -0.0624666  0.0429229  -1.455 0.145636
## nbhdE8               -0.0209736  0.0602955  -0.348 0.727968
## nbhdE9               -0.1067197  0.0739270  -1.444 0.148912
## nbhdF1                0.0407725  0.0364743   1.118 0.263682
## nbhdF2                0.1141537  0.0716876   1.592 0.111356
## nbhdF3               -0.0581379  0.0655560  -0.887 0.375201
## nbhdF4                0.1663785  0.2135076   0.779 0.435857
## nbhdF5                0.1204748  0.0656166   1.836 0.066404 .
## nbhdF6                0.1848193  0.0727529   2.540 0.011100 *
## nbhdF7                0.3095318  0.0742025   4.171 3.07e-05 ***
## nbhdF8               -0.2560476  0.1043139  -2.455 0.014135 *
## wardWard 2            0.2345808  0.1337907   1.753 0.079597 .
## wardWard 3           -0.1600695  0.2370324  -0.675 0.499509
## wardWard 4           -0.1889911  0.2364425  -0.799 0.424144
## wardWard 5           -0.1571897  0.2211457  -0.711 0.477239
## wardWard 6            0.1363283  0.1782038   0.765 0.444296
## wardWard 7           -0.5612311  0.2176807  -2.578 0.009956 **
## wardWard 8           -0.6847372  0.2113267  -3.240 0.001201 **
## quadrantNW            0.0546238  0.0292391   1.868 0.061789 .
## quadrantSE            0.0472130  0.0419574   1.125 0.260527
## quadrantSW           -0.0026900  0.0816886  -0.033 0.973732
## locationC             0.0392228  0.0458183   0.856 0.392006
## locationS            -0.0331628  0.0320719  -1.034 0.301175
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                            edf  Ref.df      F  p-value
## s(bathrm)               2.9021  3.6974 28.017  < 2e-16 ***
## s(rooms)                4.7863  5.8501  2.563  0.01669 *
## s(ayb)                  9.8282 11.0269  0.925  0.41400
## s(yr_rmdl)              0.4950  0.4950  0.074  0.84854
## s(eyb)                 33.5265 40.2413  2.144 3.78e-05 ***
## s(saledate)            13.8433 16.2351  3.469 9.48e-07 ***
## s(gba)                  8.3701  8.8777 42.861  < 2e-16 ***
## s(landarea)             5.0015  6.1731  2.589  0.01449 *
## s(latitude)             7.3247  8.1684  3.997  0.00527 **
## s(longitude)            7.8896  8.5609  4.755 9.40e-06 ***
## s(dsRM)                 8.5264  8.5264  4.393 1.13e-05 ***
## s(dsIM)                 0.8604  1.0775 14.788 7.74e-05 ***
## s(dsBT)                 0.5012  0.5012  2.318  0.28108
## te(saledate,ayb)       50.4992 55.7113  7.636  < 2e-16 ***
## te(saledate,eyb)       19.1158 60.0000  1.233  < 2e-16 ***
## te(saledate,yr_rmdl)   32.7537 72.0000  1.644  < 2e-16 ***
## te(landarea,longitude)  3.4336  3.9887  3.637  0.00629 **
## te(landarea,latitude)   3.8952 16.0000  1.389 1.62e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Rank: 581/586
## R-sq.(adj) =   0.95   Deviance explained = 95.3%
## GCV = 0.033861  Scale est. = 0.031945  n = 6000
```

4.2.2 of Random Forest Model

```
##             IncNodePurity
## heat             3.216825
## ac               2.560208
## style            4.108863
## grade          181.008935
## cndtn           25.943842
## extwall          7.014883
## roof             3.904450
## intwall          2.881901
## nbhd           949.923516
## ward           134.533098
## quadrant         1.523271
## location         4.477882
## rmdled           1.321754
## bathrm         124.106330
## hf_bathrm        4.075407
## rooms           14.755552
## bedrm           23.527929
## ayb             15.836298
## yr_rmdl         61.416026
## eyb            102.654048
## stories          5.174369
## saledate       975.092788
## gba            349.840434
## fireplaces       6.128691
## landarea        30.264865
## latitude        59.323950
## longitude      628.623776
## dsRM            38.825892
## dsIM            35.891000
## dsBT            44.209380
```

4.2.3 of Boosting Model

```
## n.trees not given. Using 500 trees.
```

```
##         heat           ac        style        grade        cndtn      extwall
##    3.1514687   15.8041312   13.4601634 1463.1149745  234.5050141   63.3738782
##         roof      intwall         nbhd         ward     quadrant     location
##   15.0277703    5.1602146 7144.3987529   35.8642377   15.4680648    1.1373979
##       rmdled       bathrm    hf_bathrm        rooms        bedrm          ayb
##    2.7879579  707.6819161   23.3625294   18.7319778   74.3001374   16.1500081
##      yr_rmdl          eyb      stories     saledate          gba     kitchens
##   43.1241139  220.9524031    2.1675890 7282.7549655 2128.8296496    0.1279562
##   fireplaces     landarea     latitude    longitude         dsRM         dsIM
##   33.7324087  162.8300445   44.6306468 4543.7836531  134.4497344   41.7614792
##         dsBT
##  234.7664465
```