Final Report

Tian

Contents

1 Data description

After pre-processing, the data frame we are working with consist of 38 meaningful variables (note `fold` is included in the data frame for convenience, we don't use it in any modeling) and 5995 observations.

- 13 factors:
  - heat: type of heat used in the house
  - ac: whether the house has air conditioning or not
  - style: describes the number of stories and/or structure of the house
  - grade: overall rating of the house
  - cndtn: condition of the house
  - extwall: material used for exterior wall
  - roof: type of roof
  - intwall: material used for interior wall
  - nbhd: ID of the neighborhood the house belongs to
  - ward: ID of the ward the house belongs to
  - quadrant: quadrant the house belongs to
  - if_rmdl: whether the house has been re-modeled ever
  - buy_first: indicator variable that has value of 1 if the house was bought before build
- 6 integers:
  - rooms: total number of rooms
  - bathrm: number of full bathrooms (shower + toilet)
  - bedrm: number of bedrooms
  - eyb: the year an improvement was built
  - kitchens: number of kitchens
  - fireplaces: number of fireplaces
- 19 numerical:
  - ayb: the earliest time the main portion of the building was built
  - stories: number of stories in primary dwelling
  - saledate: date of sale as numerical values
  - **price (response)**: price of the house
  - gba: gross building area in square feet
  - landarea: land area of property in square feet
  - latitude: latitude of the house
  - longitude: longitude of the house
  - saleyear: year the house sold
  - rmdl_diff: difference between the sale year and the re-model year, if re-model is done after sale, then the value is 0
  - avg_room_size: average size of the room in sqre feet
  - build_age: how long the house has been built
  - total_bath: total number of full bathrooms and half bathrooms
  - inter1: interaction between latitude and saledate (used in random forest only)
  - inter2: interaction between longitude and saledate (used in random forest only)
  - inter3: interaction between gba and saledate (used in random forest only)
  - inter4: interaction between landarea and longitude (used in random forest only)
  - inter5: interaction between eyb and ayb (used in random forest only)
  - inter6: interaction between latitude and build_age (used in random forest only)

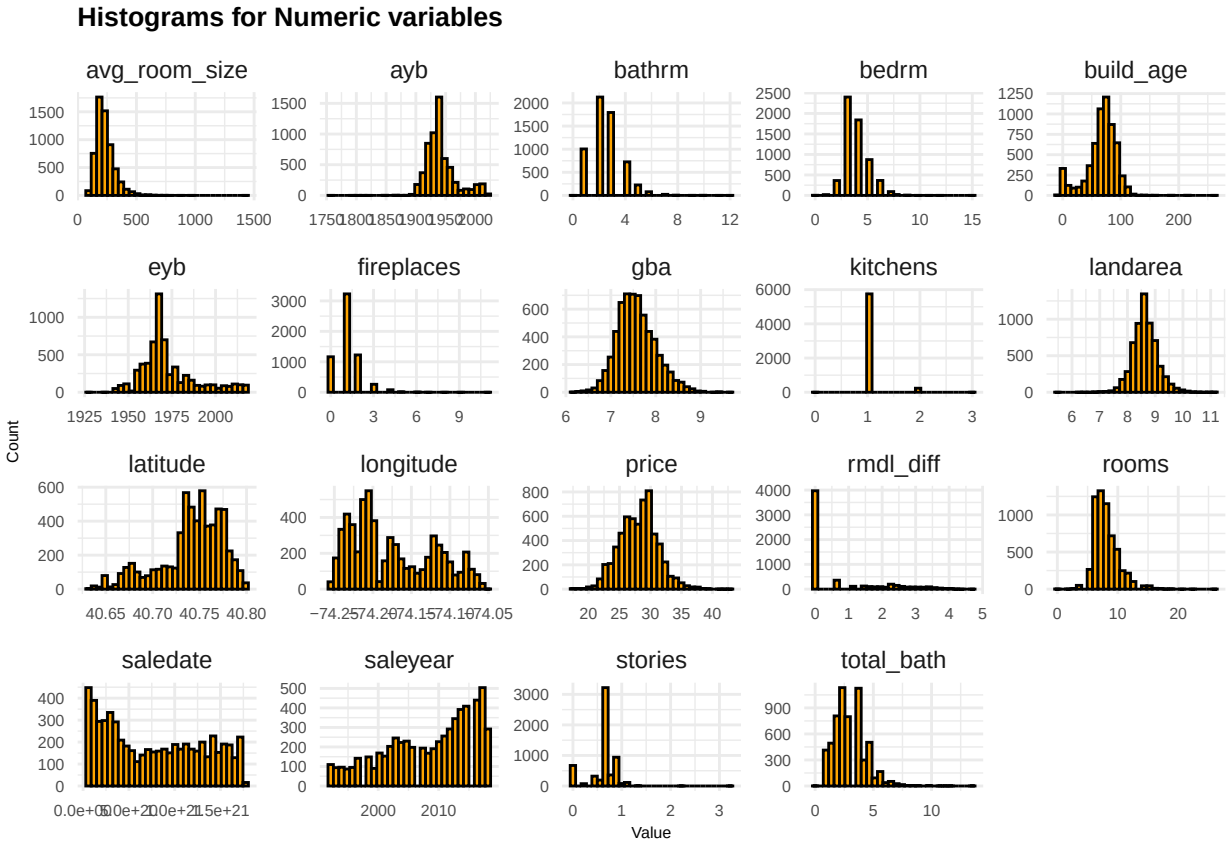
1.1 Variable Types

```
df_tmp <- subset(dat_full, select = -fold)
types <- sapply(df_tmp, class)
type_counts <- table(types)
type_table <- as.data.frame(type_counts)
colnames(type_table) <- c("Variable Type", "Count")

kable_output <- kable(type_table, format = "simple",
                      caption = "Summary of Variable Types (after pre-processing)")
print(kable_output)
```

```
##
##
## Table: Summary of Variable Types (after pre-processing)
##
## Variable Type    Count
## --------------  ------
## factor              13
## integer              6
## numeric             19
```

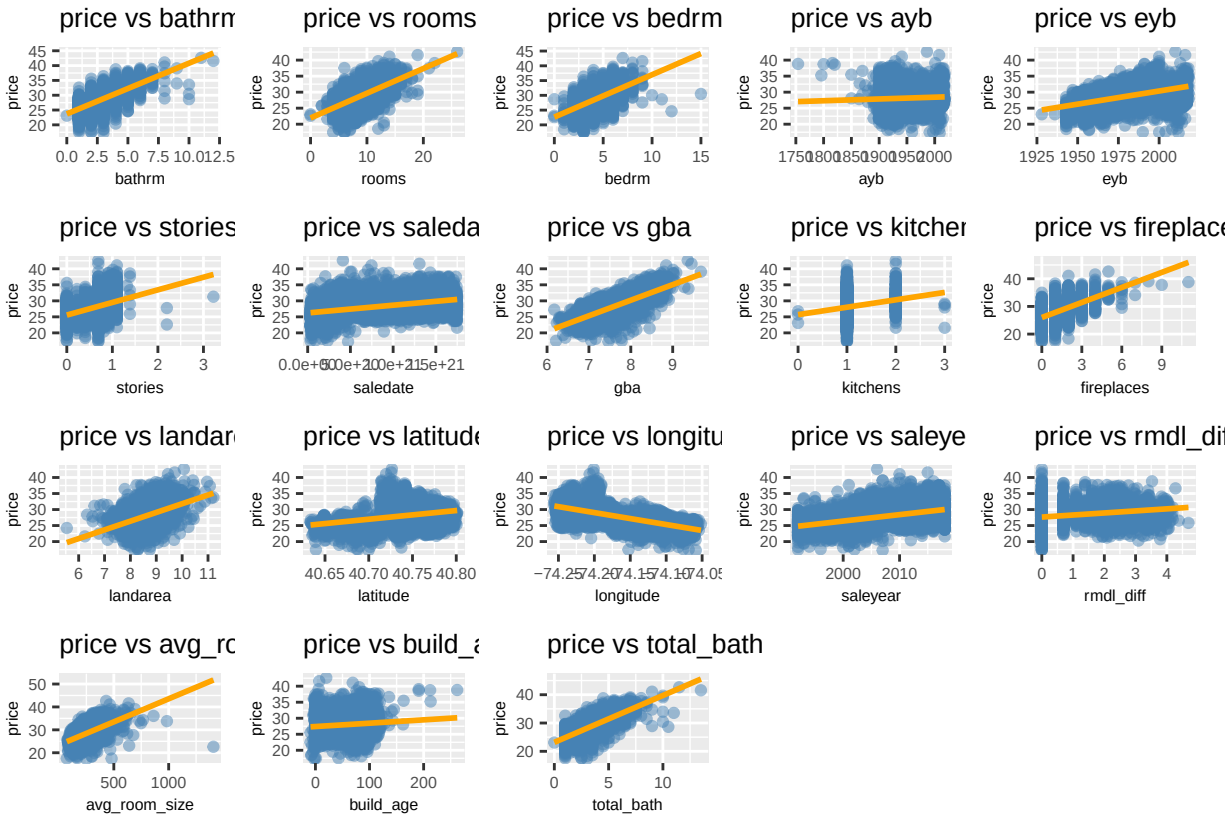
1.2 numerical variables

```
excluded_vars <- c("inter1", "inter2", "inter3", "inter4",
                   "inter5", "inter6", "fold")

plot_histograms <- function(df, exclude_vars = NULL,
                            bin_count = 30) {
  if (!is.null(exclude_vars)) {
    df <- select(df, -all_of(exclude_vars))
  }
  numeric_df <- df[sapply(df, is.numeric)]

  long_df <- pivot_longer(numeric_df, cols = everything(),
                          names_to = "Column", values_to = "Value")

  p <- ggplot(long_df, aes(x = Value)) +
    geom_histogram(bins = bin_count, fill = "orange", color = "black") +
    facet_wrap(~ Column, scales = "free") +
    theme_minimal() +
    theme(plot.title = element_text(size = 10, face = "bold"),
          axis.text = element_text(size = 6),
          axis.title = element_text(size = 6)) +
    labs(title = "Histograms for Numeric variables", x = "Value", y = "Count")

  return(p)
}

plot_histograms(dat_full, exclude_vars = excluded_vars)
```




```
plot_numeric <- function(data, target_var, exclude_vars) {
  numeric_vars <- sapply(data, is.numeric)
  numeric_vars[exclude_vars] <- FALSE
  plots <- list()
  for (var in names(numeric_vars)[numeric_vars]) {
    if (var != target_var) {
      p <- ggplot(data, aes_string(x = var, y = target_var)) +
        geom_point(alpha = 0.5, col = "steelblue") +
        geom_smooth(method = "lm", color = "orange") +
        labs(title = paste( target_var, "vs", var),
             x = var,
             y = target_var) +
        theme(plot.title = element_text(size = 10),
              axis.text = element_text(size = 6),
              axis.title = element_text(size = 6))

      plots[[var]] <- p
    }
  }

  plot_layout <- Reduce(`+`, plots) +
                 plot_layout(guides = 'collect')
  print(plot_layout)
}

excluded_vars <- c("inter1", "inter2", "inter3", "inter4", "inter5", "inter6", "fold")
```

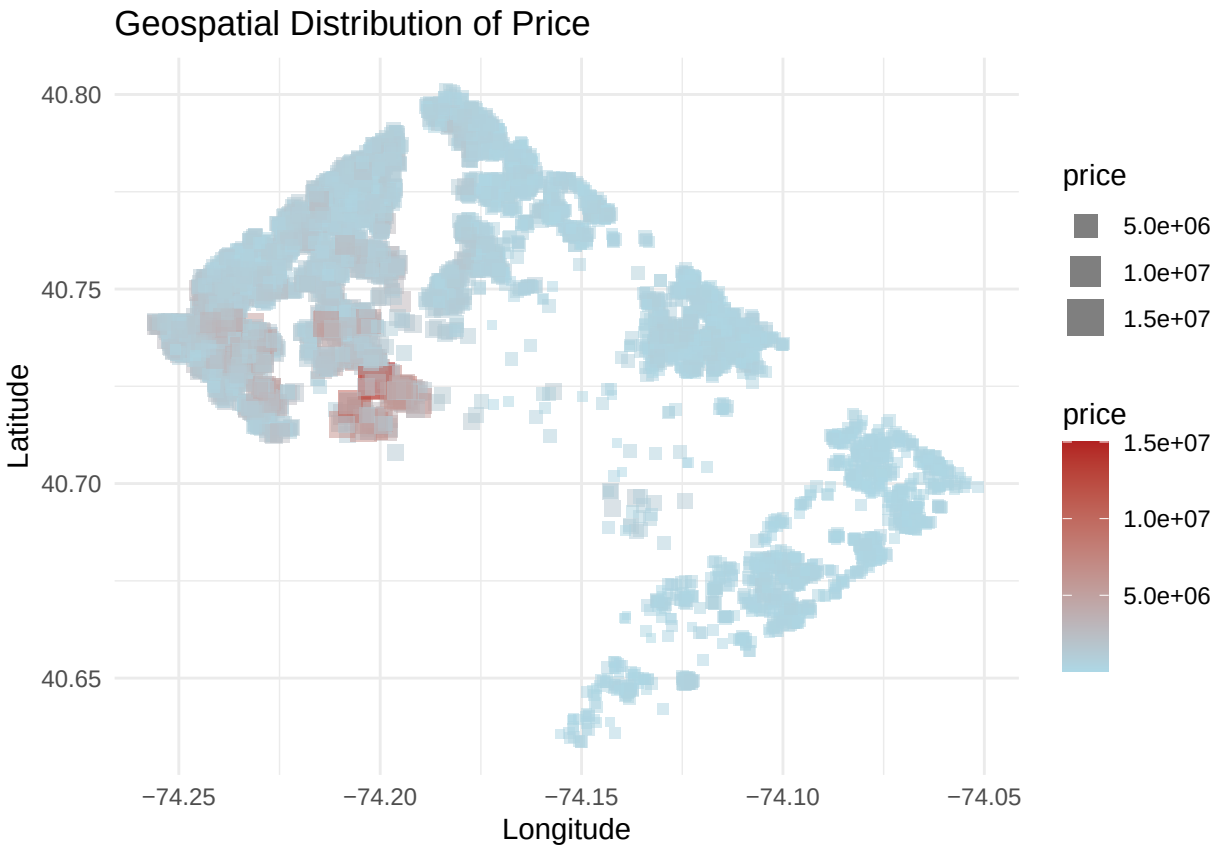
```
plot_numeric(dat_full, "price", excluded_vars)
```



All the numerical variables other than `longtitude` has a positive relationship with `price`.

1.3 geographic

```
ggplot(dat_ori, aes(x = longitude, y = latitude, color = price, size = price)) +
  geom_point(alpha = 0.5, shape = 15) +
  scale_color_gradient(low = "lightblue", high = "firebrick") +
  ggtitle("Geospatial Distribution of Price") +
  xlab("Longitude") +
  ylab("Latitude") +
  theme_minimal()
```

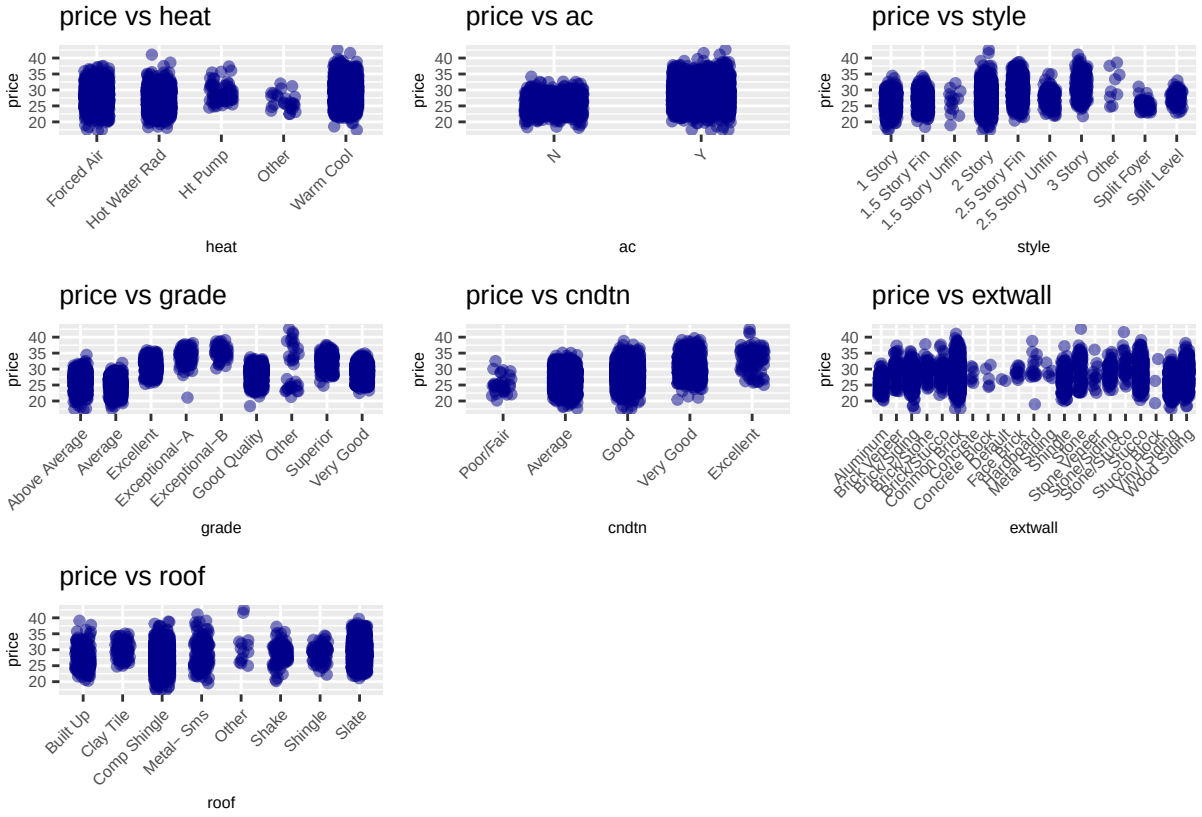


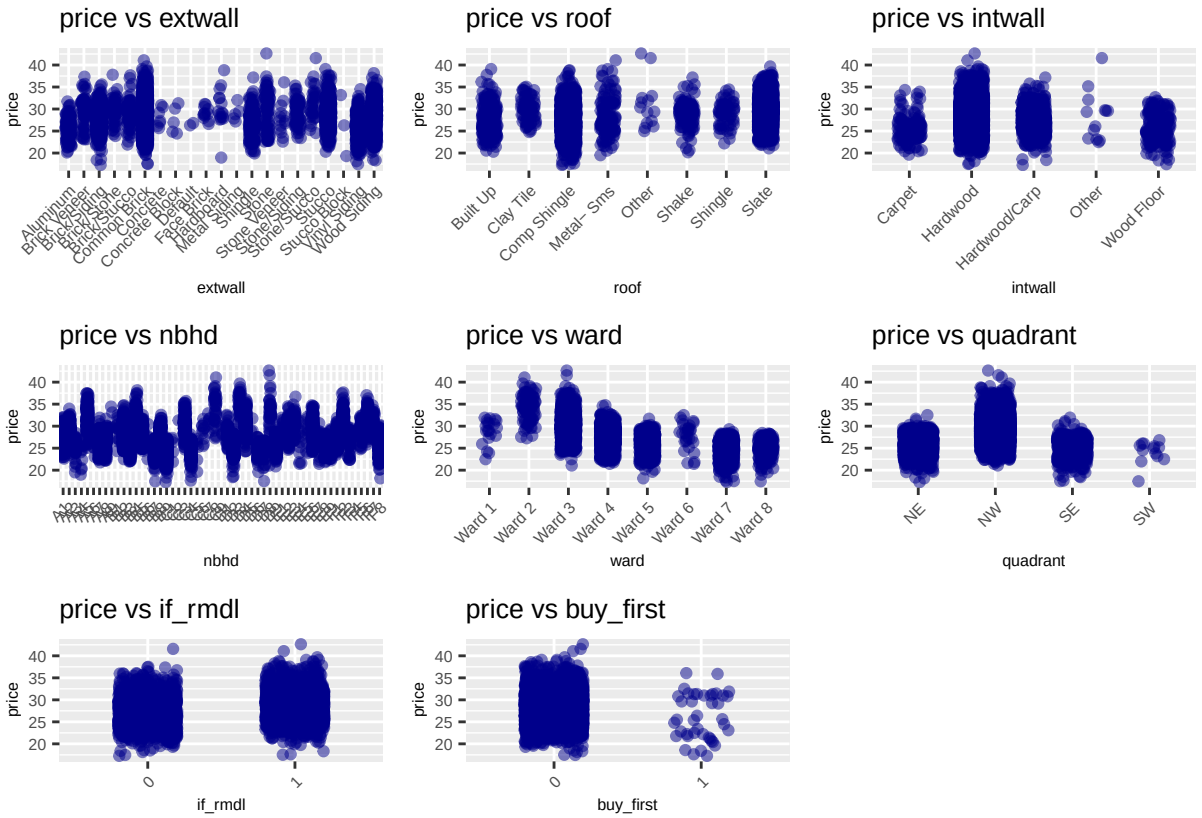
Using latitude and longitude values, we see area around longitude = -74.2 and latitude = 40.725 has higher price, which indicates location is an importance factor in house price.

1.4 categorical variables

```
plot_cate <- function(data, target_var) {
  factor_vars <- sapply(data, is.factor)
  plots <- list()

  for (var in names(factor_vars)[factor_vars]) {
      p <- ggplot(data, aes_string(x = var, y = target_var)) +
          geom_jitter(width = 0.2, alpha = 0.5, color = "darkblue") +
          labs(title = paste(target_var, "vs", var),
               x = var, y = target_var) +
          theme(plot.title = element_text(size = 10),
                axis.text = element_text(size = 6),
                axis.title = element_text(size = 6),
                axis.text.x = element_text(angle = 45, hjust = 1))
      plots[[var]] <- p
  }

  group1 <- plots[1:min(7, length(plots))]
  group2 <- if (length(plots) > 5) plots[6:length(plots)] else NULL
```

```
  if (!is.null(group1)) {
    plot_group1 <- wrap_plots(group1)
    print(plot_group1)
  }
  if (!is.null(group2)) {
    plot_group2 <- wrap_plots(group2)
    print(plot_group2)
  }
}

plot_cate(data = dat_full, target_var = "price")
```





It can be seen that some variables has obvious different impact on prices based on their levels. Such variables are

- `cndtn`: the better the condition, the higher the price.
- `grade`: the better the rating, the higher the price.
- `ward`: houses located in ward 2 and 3 have higher prices while house located in ward 7 and 8 have the lowest prices.
- `quadrant`: houses located in north west are tend to have higher prices.

2 Statistical analysis

2.1 The Models for Each Method

2.1.1 Smoothing

The final model is:

```
begin_sm <- Sys.time()
fit.sm <- mgcv::gam(price ~ s(rooms)
             +s(total_bath)+s(rmdl_diff)
             +s(bedrm)+s(ayb, k = 20, by = cndtn)+s(eyb)+s(saledate)+s(gba)
             +fireplaces
             +s(landarea)+s(latitude)+s(longitude)
             + heat+ac+style+grade+cndtn+roof+kitchens+ward
             +if_rmdl+buy_first
             + ti(eyb, ayb) + ti(gba,landarea)+ti(longitude,gba)
             + ti(longitude, ayb)
             +ti(longitude, eyb)+ti(saledate,latitude)
             ,data = dat_full)
end_sm <- Sys.time()
```

2.1.2 Random Forest

The final model is:

```
begin_rf <- Sys.time()
fit.rf <- ranger::ranger(price ~ . - fold,
                data = dat_full,
                mtry = 37, splitrule = "extratrees",
                min.node.size = 5)
end_rf <- Sys.time()
```

2.2 Comparison Summary

```
compare_df <- data.frame(
  model = c("smoothing spline", "random forest"),
  running_time = c(end_sm - begin_sm, end_rf - begin_rf),
  training_time = c("10 hrs", "5 hrs"),
  prediction_error = c(0.1908, 0.1959)
)

kable(compare_df, format = "latex", booktabs = TRUE, align = "c",
      caption = "Comparison Summary") %>%
  kable_styling(latex_options = c("striped", "scale_down")) %>%
  column_spec(1, bold = TRUE, color = "black") %>%
  row_spec(0, bold = TRUE, background = "#D3D3D3")
```

Table 1: Comparison Summary

model	running_time	training_time	prediction_error
smoothing spline	36.926018 secs	10 hrs	0.1908
random forest	7.573599 secs	5 hrs	0.1959

2.3 Predictive accuracy

The cross-validation function for smoothing and random forest:

```
cv_sm <- function(data) {

  errors_sm <- numeric(5)

  for (i in 1:5) {
    train_data <- data[fold != i, ]
    test_data <- data[fold == i, ]

    sm_model <- gam(price ~ s(rooms)
              +s(total_bath)+s(rmdl_diff)
              +s(bedrm)+s(ayb, k = 20, by = cndtn)+s(eyb)+s(saledate)+s(gba)
              +fireplaces
              +s(landarea)+s(latitude)+s(longitude)
              + heat+ac+style+grade+cndtn+roof+kitchens+ward
              +if_rmdl+buy_first
              + ti(eyb, ayb) + ti(gba,landarea)+ti(longitude,gba)
              + ti(longitude, ayb)
              +ti(longitude, eyb)+ti(saledate,latitude)
              ,data = train_data)

    pred <- predict(sm_model, newdata = test_data)
    pred_reverse <- (pred * 0.101 + 1)^(1/0.101)
    actual_responses <- test_data[['price']]
    actual_responses <- (actual_responses * 0.101 + 1)^(1/0.101)

    errors_sm[i] <-  sqrt(mean((log(pred_reverse)-
                                log(actual_responses))^2))
  }
  return(errors_sm)
}
```

```
cv_rf <- function(data) {
  errors_rf <- numeric(5)

  for (i in 1:5) {
    train_data <- data[fold != i, ]
    test_data <- data[fold == i, ]

    rf_model <- ranger(price ~ . - fold,
                data = train_data,
                mtry = 37, splitrule = "extratrees",
                min.node.size = 5)
```

```
    pred <- predict(rf_model, data = test_data)$predictions
    pred_reverse <- (pred * 0.101 + 1)^(1/0.101)
    actual_responses <- test_data[['price']]
    actual_responses <- (actual_responses * 0.101 + 1)^(1/0.101)

    errors_rf[i] <- sqrt(mean((log(pred_reverse)-
                               log(actual_responses))^2))
  }
  return(errors_rf)
}
```

The 5 fold cross-validation score for `smoothing`:

```
set.seed(42)
cv_result_sm <- cv_sm(dat_full)
print(paste("5-fold cross-validatoin:" ,cv_result_sm))
```

```
## [1] "5-fold cross-validatoin: 0.16781614017129"
## [2] "5-fold cross-validatoin: 0.197484661024805"
## [3] "5-fold cross-validatoin: 0.192162056601297"
## [4] "5-fold cross-validatoin: 0.208260761794989"
## [5] "5-fold cross-validatoin: 0.188157206556455"
```

```
print(paste("Average: ", mean(cv_result_sm)))
```

```
## [1] "Average:  0.190776165229767"
```

The 5 fold cross-validation score for `random forest`:

```
set.seed(42)
cv_result_rf <- cv_rf(dat_full)
print(paste("5-fold cross-validatoin:" ,cv_result_rf))
```

```
## [1] "5-fold cross-validatoin: 0.181719803189755"
## [2] "5-fold cross-validatoin: 0.206810603266541"
## [3] "5-fold cross-validatoin: 0.200137545836101"
## [4] "5-fold cross-validatoin: 0.198220889590503"
## [5] "5-fold cross-validatoin: 0.192456821531735"
```

```
print(paste("Average: ", mean(cv_result_rf)))
```

```
## [1] "Average:  0.195869132682927"
```

We see from 5-fold cross-validation that the predictor error of smoothing model is much lower than that of the random forest model. So from the perspective of prediction accuracy, smoothing spline model is better than random forest.

2.4 Computational complexity and runtime

The running time for smoothing is 36.926017999649, and the running time for random forest is 7.57359886169434. So the random forest model runs almost **7** times faster than the smoothing spline model. Random forest is more computational efficient.

2.5 Ease of use/model building

It takes a longer time to build the smoothing spline model since deciding whether to add a predictor/interaction term or not is a time-consuming process (plus the running time for smoothing spline is longer). Excluding the pre-processing part, I spent about 10 hours building the smoothing spline model while for the random forest model I only spent about 5 hours. Because there aren't many tuning parameters for random forest and the running time is shorter.

2.6 Interpretation

- Random Forest : although variable importance measures can provide insights into which variables are most influential, understanding the specific nature of the relationships and interactions is challenging since we don't have direct access to them. Also, we can't visualize random forest.
- Smoothing Spline: each component of the model (each predictor and interaction) has a clear, visualizable effect on the response variable, allowing us to understand how changes in predictors affect the response. Also, the effects of the predictors are modeled as smooth curves, which can be plotted and directly interpreted (see the plot below).
- Summary: smoothing spline is more interpretable than random forest.

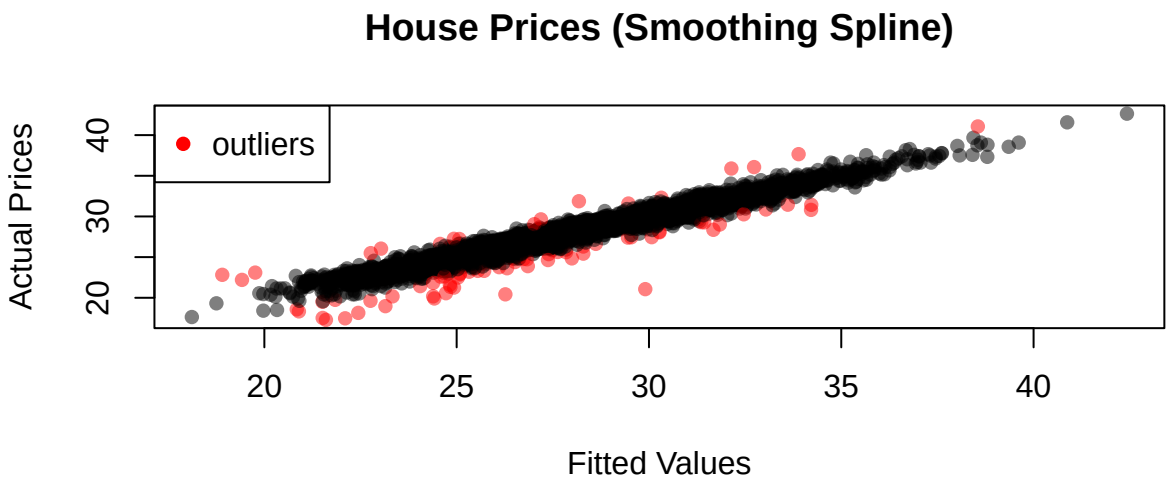
2.7 Sensitivity to outliers

2.7.1 Outliers Identification

For each fitted model, we calculate it's residuals and set the outlier threshold to be **3 times the standard deviation of the residuals**. That is, any observation with residuals 3 times the standard deviation of the residuals is labelled as outlier.

For smoothing:

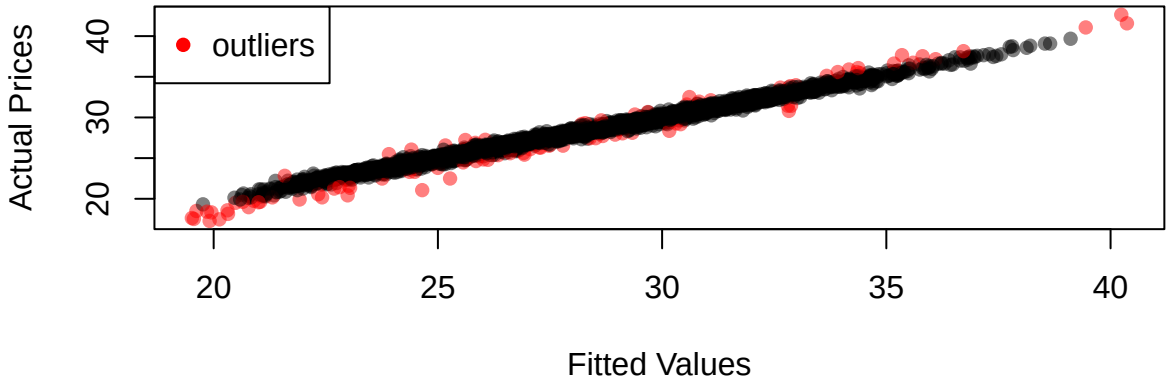
```
res1 <- residuals(fit.sm)
outlier_cutoff1 <- 3 * sd(res1)
outliers_indices1 <- which(abs(res1) > outlier_cutoff1)
is_outlier1 <- abs(res1) > outlier_cutoff1

plot(fitted(fit.sm), dat_full$price,
     xlab = "Fitted Values", ylab = "Actual Prices",
     main = "House Prices (Smoothing Spline)",
     col = ifelse(is_outlier1, rgb(1, 0, 0, 0.5), rgb(0, 0, 0, 0.5)),
     pch = 16)

legend("topleft", legend = "outliers", col = "red", pch = 16)
```



There are 74 outliers.

For random forest:

```
pred <- predict(fit.rf, dat_full)$predictions
res2 <- dat_full$price - pred
outlier_cutoff2 <- 3 * sd(res2)
outliers_indices2 <- which(abs(res2) > outlier_cutoff2)
is_outlier2 <- abs(res2) > outlier_cutoff2

plot(pred, dat_full$price,
     xlab = "Fitted Values", ylab = "Actual Prices",
     main = "House Prices (random forest)",
     col = ifelse(is_outlier2, rgb(1, 0, 0, 0.5), rgb(0, 0, 0, 0.5)),
     pch = 16)

legend("topleft", legend = "outliers", col = "red", pch = 16)
```



There are 89 outliers.

2.7.2 Sensitivity Comparison:

To evaluate model's sensitivity to outliers, we lower the weight of outliers (set them to be 0.5) and re-run cross-validation for both models.

```
# Smoothing
weights1 <- rep(1, nrow(dat_full))
weights1[outliers_indices1] <- 0.5  # Set weights of outliers to 0.5

cv_sm_weight <- function(data) {

  errors_sm <- numeric(5)

  for (i in 1:5) {
    train_indices <- which(data$fold != i)
    train_data <- data[fold != i, ]
    test_data <- data[fold == i, ]
    train_weights <- weights1[train_indices]

    sm_model <- gam(price ~ s(rooms)
              +s(total_bath)+s(rmdl_diff)
              +s(bedrm)+s(ayb, k = 20, by = cndtn)+s(eyb)+s(saledate)+s(gba)
              +fireplaces
              +s(landarea)+s(latitude)+s(longitude)
              + heat+ac+style+grade+cndtn+roof+kitchens+ward
              +if_rmdl+buy_first
              + ti(eyb, ayb) + ti(gba,landarea)+ti(longitude,gba)
              + ti(longitude, ayb)
              +ti(longitude, eyb)+ti(saledate,latitude)
              ,data = train_data, weights = train_weights)

    pred <- predict(sm_model, newdata = test_data)
    pred_reverse <- (pred * 0.101 + 1)^(1/0.101)
    actual_responses <- test_data[['price']]
    actual_responses <- (actual_responses * 0.101 + 1)^(1/0.101)

    errors_sm[i] <-  sqrt(mean((log(pred_reverse)-
                                log(actual_responses))^2))
  }
  return(errors_sm)
}

set.seed(42)
cv_sm_weight <- cv_sm_weight(dat_full)
```

```
# Random Forest
weights2 <- rep(1, nrow(dat_full))
weights2[outliers_indices2] <- 0.5  # Set weights of outliers to 0.5

cv_rf_weight <- function(data) {
  errors_rf <- numeric(5)
```

```
  for (i in 1:5) {
    train_indices <- which(data$fold != i)
    test_indices <- which(data$fold == i)

    train_data <- data[train_indices, ]
    test_data <- data[test_indices, ]

    train_weights <- weights2[train_indices]

    rf_model <- ranger(
      formula = price ~ . -fold,
      data = train_data,
      case.weights = train_weights,
      mtry = 37,
      splitrule = "extratrees",
      min.node.size = 5,
      num.trees = 500
    )

    pred <- predict(rf_model, data = test_data)$predictions
    pred_reverse <- (pred * 0.101 + 1)^(1/0.101)
    actual_responses <- test_data[['price']]
    actual_responses <- (actual_responses * 0.101 + 1)^(1/0.101)

    errors_rf[i] <- sqrt(mean((log(pred_reverse) -
                               log(actual_responses))^2))
  }
  return(errors_rf)
}

set.seed(42)
cv_rf_weight <- cv_rf_weight(dat_full)
```

```
print(paste("5-fold cross-validatoin:" ,cv_sm_weight))
```

```
## [1] "5-fold cross-validatoin: 0.165661222626796"
## [2] "5-fold cross-validatoin: 0.195562889987725"
## [3] "5-fold cross-validatoin: 0.192009089284346"
## [4] "5-fold cross-validatoin: 0.207466343173008"
## [5] "5-fold cross-validatoin: 0.185867945521133"
```

```
print(paste("Average: ", mean(cv_sm_weight)))
```

```
## [1] "Average:  0.189313498118601"
```

```
print(paste("5-fold cross-validatoin:" ,cv_rf_weight))
```

```
## [1] "5-fold cross-validatoin: 0.185375623395775"
## [2] "5-fold cross-validatoin: 0.208842084979092"
## [3] "5-fold cross-validatoin: 0.201095176385844"
## [4] "5-fold cross-validatoin: 0.200004522043545"
## [5] "5-fold cross-validatoin: 0.194597993378015"
```

```r
print(paste("Average: ", mean(cv_rf_weight)))
```

```
## [1] "Average:  0.197983080036454"
```

- The average cross-validation error for smoothing spline reduced from 0.1907762 to 0.1893135
- The average cross-validation error for random forest increased from 0.1958691 to 0.1979831. One explanation could be the random forest model is good at handling the outliers or the outliers are influential variables that helps setting the decision boundary in the model, hence excluding them might be bad for the model (less observations).
- Above results indicate random forest is more robust to outliers than smoothing spline.

2.8 Insights

2.8.1 Important interaction

```r
summary(fit.sm)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## price ~ s(rooms) + s(total_bath) + s(rmdl_diff) + s(bedrm) +
##     s(ayb, k = 20, by = cndtn) + s(eyb) + s(saledate) + s(gba) +
##     fireplaces + s(landarea) + s(latitude) + s(longitude) + heat +
##     ac + style + grade + cndtn + roof + kitchens + ward + if_rmdl +
##     buy_first + ti(eyb, ayb) + ti(gba, landarea) + ti(longitude,
##     gba) + ti(longitude, ayb) + ti(longitude, eyb) + ti(saledate,
##     latitude)
##
## Parametric coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)           27.063324   0.571247  47.376  < 2e-16 ***
## fireplaces             0.089248   0.014191   6.289 3.43e-10 ***
## heatHot Water Rad     -0.029973   0.028595  -1.048 0.294595
## heatHt Pump            0.034923   0.086639   0.403 0.686896
## heatOther             -0.185386   0.138152  -1.342 0.179681
## heatWarm Cool          0.006356   0.023846   0.267 0.789842
## acY                    0.114309   0.033657   3.396 0.000688 ***
## style1.5 Story Fin     0.028691   0.048061   0.597 0.550550
## style1.5 Story Unfin -0.316990   0.182687  -1.735 0.082767 .
## style2 Story          -0.030821   0.038615  -0.798 0.424819
## style2.5 Story Fin     0.068907   0.046690   1.476 0.140046
## style2.5 Story Unfin   0.048542   0.073944   0.656 0.511549
## style3 Story           0.070563   0.070158   1.006 0.314565
## styleOther             0.109707   0.235781   0.465 0.641740
## styleSplit Foyer       0.409198   0.118044   3.466 0.000531 ***
## styleSplit Level       0.049339   0.093328   0.529 0.597058
## gradeAverage          -0.150916   0.037015  -4.077 4.62e-05 ***
## gradeExcellent         0.467667   0.059483   7.862 4.47e-15 ***
```

```
## gradeExceptional-A    1.071374   0.107950   9.925  < 2e-16 ***
## gradeExceptional-B    1.599542   0.133337  11.996  < 2e-16 ***
## gradeGood Quality     0.094680   0.036225   2.614 0.008980 **
## gradeOther            1.112533   0.145164   7.664 2.10e-14 ***
## gradeSuperior         0.744130   0.075749   9.824  < 2e-16 ***
## gradeVery Good        0.214509   0.043308   4.953 7.51e-07 ***
## cndtnAverage          0.568881   0.539778   1.054 0.291966
## cndtnGood             0.863449   0.539953   1.599 0.109848
## cndtnVery Good        1.311959   0.540850   2.426 0.015308 *
## cndtnExcellent        2.238033   0.557107   4.017 5.96e-05 ***
## roofClay Tile         0.278364   0.094045   2.960 0.003090 **
## roofComp Shingle      0.004034   0.050795   0.079 0.936703
## roofMetal- Sms       -0.093336   0.080018  -1.166 0.243484
## roofOther             0.595871   0.201567   2.956 0.003127 **
## roofShake            -0.016176   0.083382  -0.194 0.846186
## roofShingle           0.100639   0.099026   1.016 0.309534
## roofSlate             0.052327   0.053980   0.969 0.332399
## kitchens              0.036325   0.044716   0.812 0.416618
## wardWard 2            1.796371   0.196848   9.126  < 2e-16 ***
## wardWard 3            0.093662   0.175152   0.535 0.592845
## wardWard 4           -0.263032   0.170467  -1.543 0.122882
## wardWard 5           -0.494687   0.208618  -2.371 0.017760 *
## wardWard 6            1.811549   0.258782   7.000 2.84e-12 ***
## wardWard 7           -0.522814   0.253992  -2.058 0.039598 *
## wardWard 8           -1.586421   0.257871  -6.152 8.16e-10 ***
## if_rmdl1             -0.038622   0.026959  -1.433 0.152026
## buy_first1           -3.557897   0.118336 -30.066  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                            edf Ref.df        F  p-value
## s(rooms)                 8.390  8.836    4.487 2.86e-05 ***
## s(total_bath)            1.917  2.500   57.926  < 2e-16 ***
## s(rmdl_diff)             6.067  6.986   20.481  < 2e-16 ***
## s(bedrm)                 7.893  8.484    4.828 3.71e-06 ***
## s(ayb):cndtnPoor/Fair   10.581 10.973    2.078 0.015577 *
## s(ayb):cndtnAverage      2.706  3.546    3.603 0.010480 *
## s(ayb):cndtnGood        12.014 13.904    3.364 2.06e-05 ***
## s(ayb):cndtnVery Good    1.002  1.003   22.427 2.24e-06 ***
## s(ayb):cndtnExcellent    2.051  2.410    3.191 0.026599 *
## s(eyb)                   4.260  5.275    9.226  < 2e-16 ***
## s(saledate)              8.987  9.000 2363.668  < 2e-16 ***
## s(gba)                   1.003  1.006  226.741  < 2e-16 ***
## s(landarea)              7.924  8.713   19.460  < 2e-16 ***
## s(latitude)              8.711  8.973   45.595  < 2e-16 ***
## s(longitude)             8.375  8.894  124.275  < 2e-16 ***
## ti(eyb,ayb)              2.809  3.736    3.489 0.017660 *
## ti(gba,landarea)        11.158 12.307    5.639  < 2e-16 ***
## ti(longitude,gba)       11.324 12.946    6.473  < 2e-16 ***
## ti(longitude,ayb)        8.755  9.890    4.346 5.93e-06 ***
## ti(longitude,eyb)       10.870 12.788    2.912 0.000736 ***
## ti(saledate,latitude)   10.161 12.443    7.803  < 2e-16 ***
## ---
```

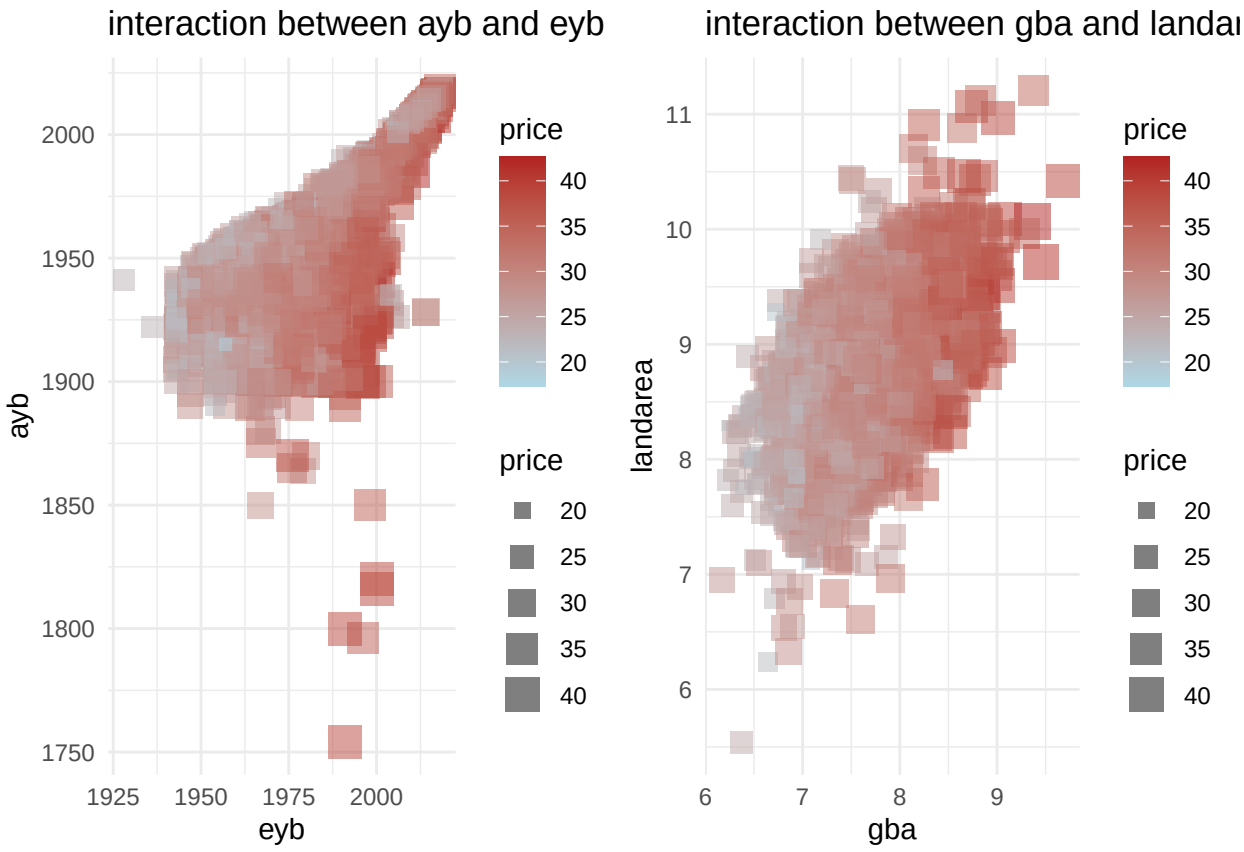
```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.953   Deviance explained = 95.4%
## GCV = 0.45872  Scale est. = 0.44403   n = 5995
```

In our final smoothing model there are 7 interaction terms, which are `eyb*ayb`, `gba*landarea`, `longitude*ayb`, `longitude*gba`, `longitude*eyb`, `saledate*latitude` and `ayb*cndtn` (specified using `by`). And all of them has p-value less than 0.05, which suggests they are significant.

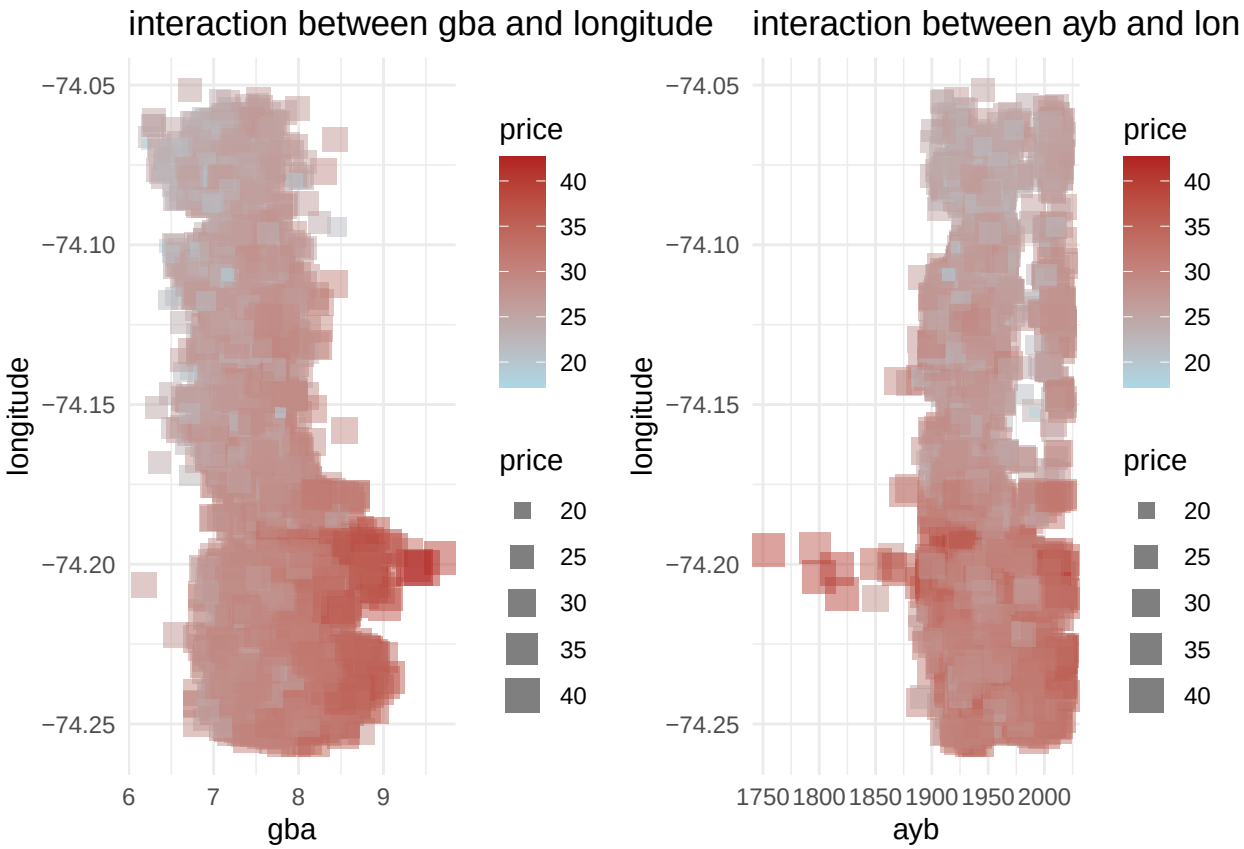
The following plots illustrate how the effect of one predictor on house prices varies across different values of another predictor.

```
g1 <- ggplot(dat_full, aes(x = eyb, y = ayb, color = price, size = price)) +
        geom_point(alpha = 0.5, shape = 15) +
        scale_color_gradient(low = "lightblue", high = "firebrick") +
        ggtitle("interaction between ayb and eyb") +
        xlab("eyb") +
        ylab("ayb") +
        theme_minimal()

g2 <- ggplot(dat_full, aes(x = gba, y = landarea, color = price, size = price)) +
  geom_point(alpha = 0.5, shape = 15) +
  scale_color_gradient(low = "lightblue", high = "firebrick") +
  ggtitle("interaction between gba and landarea") +
  xlab("gba") +
  ylab("landarea") +
  theme_minimal()


g3 <- ggplot(dat_full, aes(x = gba, y = longitude, color = price, size = price)) +
  geom_point(alpha = 0.5, shape = 15) +
  scale_color_gradient(low = "lightblue", high = "firebrick") +
  ggtitle("interaction between gba and longitude") +
  xlab("gba") +
  ylab("longitude") +
  theme_minimal()


g4 <- ggplot(dat_full, aes(x = ayb, y = longitude, color = price, size = price)) +
  geom_point(alpha = 0.5, shape = 15) +
  scale_color_gradient(low = "lightblue", high = "firebrick") +
  ggtitle("interaction between ayb and longitude") +
  xlab("ayb") +
  ylab("longitude") +
  theme_minimal()

g5 <- ggplot(dat_full, aes(x = eyb, y = longitude, color = price, size = price)) +
  geom_point(alpha = 0.5, shape = 15) +
  scale_color_gradient(low = "lightblue", high = "firebrick") +
  ggtitle("interaction between ayb and longitude") +
  xlab("ayb") +
  ylab("longitude") +
  theme_minimal()
```

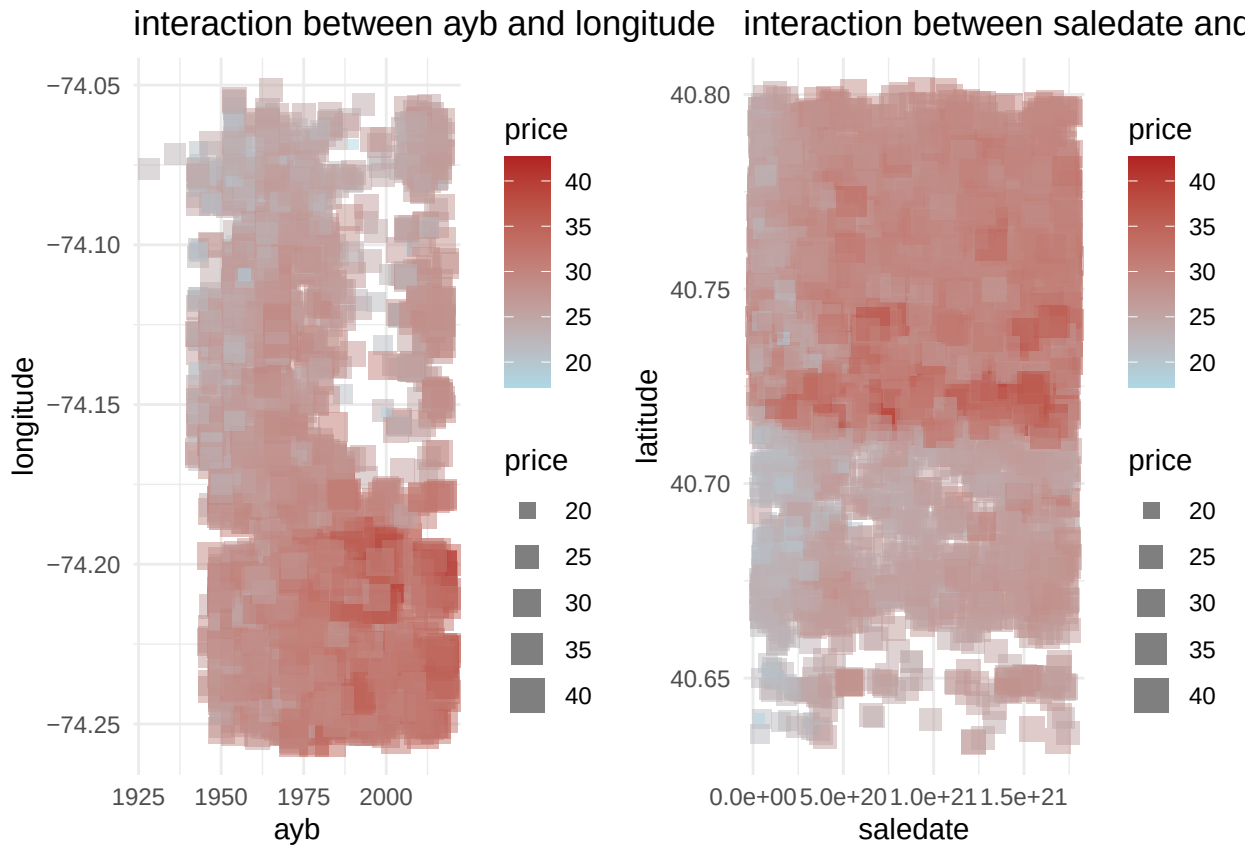
```
g6 <- ggplot(dat_full, aes(x = saledate, y = latitude,
                           color = price, size = price)) +
        geom_point(alpha = 0.5, shape = 15) +
        scale_color_gradient(low = "lightblue", high = "firebrick") +
        ggtitle("interaction between saledate and latitude") +
        xlab("saledate") +
        ylab("latitude") +
        theme_minimal()

grid.arrange(g1, g2, ncol=2, nrow=1)
```



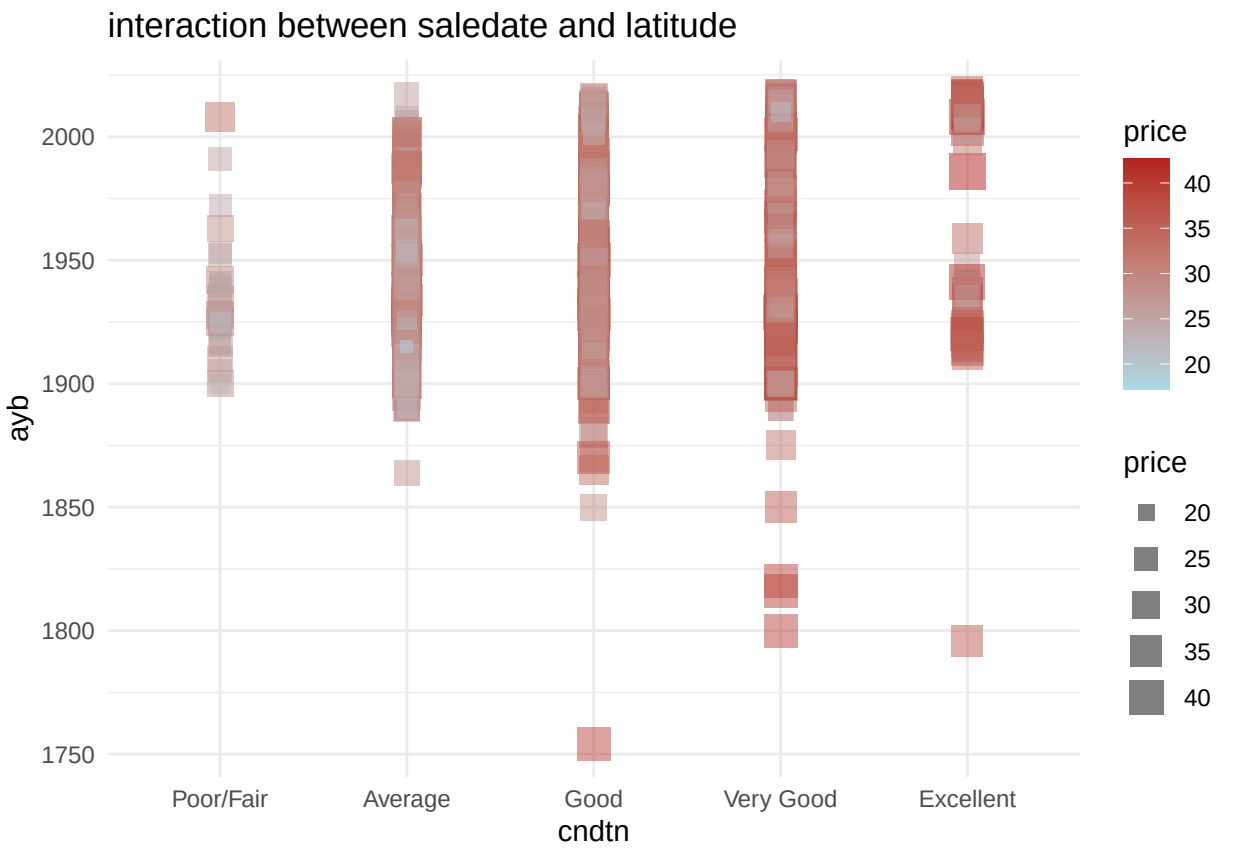

```
grid.arrange(g3, g4, ncol=2, nrow=1)
```




```
grid.arrange(g5, g6, ncol=2, nrow=1)
```



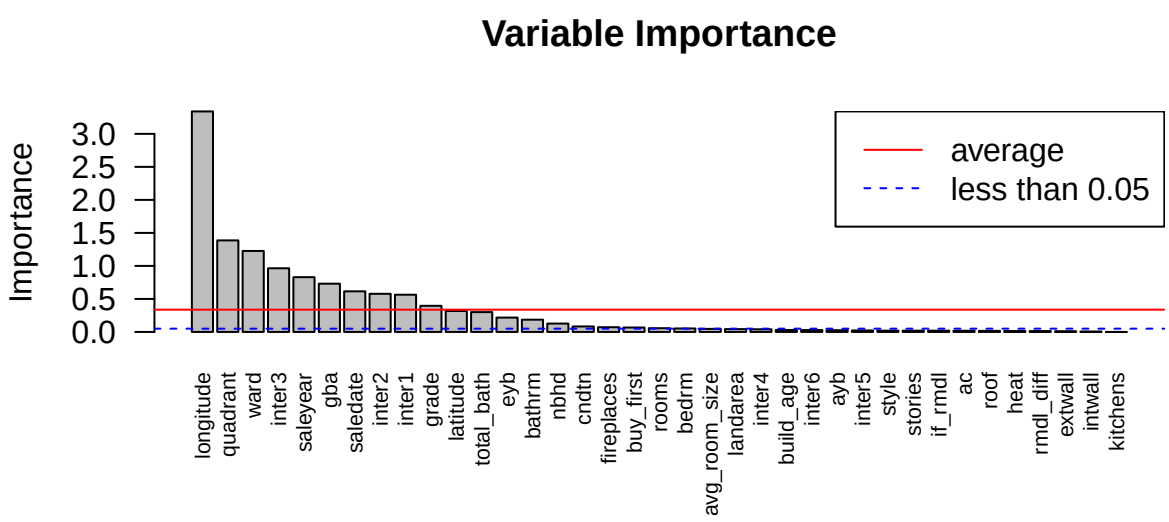

```
ggplot(dat_full, aes(x = cndtn, y = ayb,
                         color = price, size = price)) +
        geom_point(alpha = 0.5, shape = 15) +
        scale_color_gradient(low = "lightblue", high = "firebrick") +
        ggtitle("interaction between saledate and latitude") +
        xlab("cndtn") +
        ylab("ayb") +
        theme_minimal()
```



2.8.2 Variable importance

The variable importance is straightforward for random forest model. We use `importance = "permutation"`, which measures the decrease in mean squared error, to evaluate the variable importance in our final random forest model.

```
set.seed(444)
rf_model <- ranger(price ~ . - fold,
                   data = dat_full,
                   mtry = 37, splitrule = "extratrees",
                   min.node.size = 5,
                   importance = "permutation")
importance <-rf_model$variable.importance
importance_sort <- sort(importance, decreasing = TRUE)
barplot(importance_sort, names.arg = names(importance_sort), las = 2,
        main = "Variable Importance",
        ylab = "Importance", cex.names = 0.7)
legend("topright", col = c("red", "blue"),
       legend = c("average", "less than 0.05"),
       lty = c(1, 2))
avg <- mean(importance)
abline(h = avg, col = "red")
abline(h = 0.05, col = "blue", lty = 2)
```




```
importance_sort
```

```
##     longitude      quadrant          ward        inter3      saleyear
##  3.3398125862  1.3864306220  1.2263928317  0.9641290495  0.8297619000
##           gba      saledate        inter2        inter1         grade
##  0.7310381505  0.6152016295  0.5774422843  0.5639450617  0.3957296136
##      latitude    total_bath           eyb       bathrm          nbhd
##  0.3171097241  0.3003308108  0.2172932778  0.1857964952  0.1269971508
##         cndtn    fireplaces    buy_first         rooms        bedrm
##  0.0832391659  0.0715330890  0.0672272841  0.0564028555  0.0523123414
## avg_room_size      landarea        inter4    build_age        inter6
##  0.0464549060  0.0427187931  0.0402217834  0.0293724158  0.0280747970
##           ayb        inter5         style       stories       if_rmdl
##  0.0263309151  0.0226513313  0.0199765347  0.0198940112  0.0195779705
##            ac          roof          heat    rmdl_diff       extwall
##  0.0181662321  0.0159727351  0.0137693066  0.0129730530  0.0104667171
##       intwall      kitchens
##  0.0060605156 -0.0003117266
```

For variables that has importance greater than the average importance:

- There are 10 variables have importance larger than the average importance.
- We see the top 5 important variables are `longitude`, `quadrant`,`ward`, `inter3 (gba * saledate)`, and `saleyear`. Hence, we can conclude the most important two factor of the house price is location and the time it was sold.
- Other important variables include `gba`, `saledate`, `inter2 (longitude * saledate)`, `inter1(latitude * saledate)` and `grade`. Hence, we can conclude the gross building area, how the effect of the selling date on house prices varies across different locations, and the overall rating of the house itself are helpful in predicting house price.
- `longitude` is significantly more important than other variables.

For variables that has importance less than the average importance:

- There are 27 variables have importance less than the average importance and 16 variables that has importance less than 0.05.

- The least important 3 variables are `kitchens`, `intwall`, and `extwall`, which suggests the number of kitchens, the material of interior wall and exterior wall are not help in predicting house price.

2.8.3 Exploratory Data Analysis (EDA)

Using this dataset, there are some meaningful questions we can answer.

1. What factors are most predictive of house prices?

Answer:

- since longitude, quadrant and ward are highly important variable in our prediction model, location is the most important factor in predicting house prices. House located at city center have a higher prices than house located at rural areas.
- some other important factors include the time the house was sold and the gross area of the house, which align with our intuition. The bigger than area of the house, the higher the price and the earlier the time it was sold, the less the price.

2. How have house prices changed over time?

Answer: the prices increase over time.

3. How do renovations and remodeling affect house prices?

Answer: the variable importance from random forest suggest remodeling doesn't have a obvious impact on house price. Hence, from a seller's point of view, renovation is optional before selling the house.

4. How specific features like type of heating, air conditioning, and the presence of a fireplace influence house prices?

Answer:

- The most important features are number of bathrooms, number of bedrooms and number of fireplaces. The higher number the number of bathrooms, bedrooms, and fireplaces, the higher the price
- The least important features are number of kitchens, material of interior and exterior wall, and type of heat. In general, buyers don't care about those feature that much.

3 Conclusions

In this project, we built two models using random forest and smoothing spline to predict the house price. The findings can be divided into two aspects:

From the perspective of modelling:

- Random forest is more time-efficient/computational efficient and more robust to outliers
- Smoothing spline has higher interpretability and higher prediction accuracy.
- Suggestion: if one has enough time and need to do detailed data analysis, it's advisable to use the smoothing spline approach as a decent smoothing spline model will give you more information and higher prediction accuracy. However, if it's a urgent task, random forest will serve the purpose of making good prediction with less effort in tuning.

From the perspective of house price prediction:

- Location: the location of the house (captured by variables like longitude, ward, and quadrant) was found to be a critical predictor of house prices.
- Time influence: the selling time was also crucial, as the price increases over time.
- Property characteristics: physical attributes of properties, including gross building area, conditions, and number of bathrooms played substantial roles in influencing house prices.