Exploratory Data Analysis and Data Cleaning for Housing Sales Dataset

Gerard Corrales

```r
library(dplyr)
library(tidyverse)
library(tidymodels)
library(mosaic)
library(cluster)
library(factoextra)
library(lubridate)
```

1. Data Importing and Pre-processing

1.1.1 Importing 'house_sales.csv' data set to our project using read_csv() function.The file type .csv is a popular tabular data type.

```r
housing_sales<- read.csv("house_sales.csv")
```

1.1.2 Checking the dimensions of the dataset

```r
dimensions <- dim(housing_sales)

print(paste("Total Number of Rows:", dimensions[1],
            ", Total Number of Columns:", dimensions[2]))
```

```
## [1] "Total Number of Rows: 21613 , Total Number of Columns: 21"
```

The dataset contains a total of 21,613 rows and 21 columns.

1.1.3 Checking data types in the data set

```r
str(housing_sales)
```

```
## 'data.frame':    21613 obs. of  21 variables:
##  $ id           : num  7.13e+09 6.41e+09 5.63e+09 2.49e+09 1.95e+09 ...
##  $ date         : chr  "20141013T000000" "20141209T000000" "20150225T000000" "20141209T000000" ...
##  $ price        : num  221900 538000 180000 604000 510000 ...
##  $ bedrooms     : num  3 3 2 4 3 4 3 3 3 3 ...
##  $ bathrooms    : num  1 2.25 1 3 2 4.5 2.25 1.5 1 2.5 ...
##  $ sqft_living  : num  1180 2570 770 1960 1680 ...
##  $ sqft_lot     : num  5650 7242 10000 5000 8080 ...
##  $ floors       : num  1 2 1 1 1 1 2 1 1 2 ...
##  $ waterfront   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ view         : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
##  $ condition    : int  3 3 3 5 3 3 3 3 3 3 ...
##  $ grade        : int  7 7 6 7 8 11 7 7 7 7 ...
##  $ sqft_above   : int  1180 2170 770 1050 1680 3890 1715 1060 1050 1890 ...
##  $ sqft_basement: int  0 400 0 910 0 1530 0 0 730 0 ...
##  $ yr_built     : int  1955 1951 1933 1965 1987 2001 1995 1963 1960 2003 ...
##  $ yr_renovated : int  0 1991 0 0 0 0 0 0 0 0 ...
##  $ zipcode      : int  98178 98125 98028 98136 98074 98053 98003 98198 98146 98038 ...
##  $ lat          : num  47.5 47.7 47.7 47.5 47.6 ...
##  $ long         : num  -122 -122 -122 -122 -122 ...
##  $ sqft_living15: int  1340 1690 2720 1360 1800 4760 2238 1650 1780 2390 ...
##  $ sqft_lot15   : int  5650 7639 8062 5000 7503 101930 6819 9711 8113 7570 ...
```

Data types found in the data frame:

id: Numeric

date: Character

price, bedrooms, bathrooms, sqft_living, sqft_lot, floors, lat, long: Numeric (these are continuous variables)

waterfront, view, condition, grade, yr_built, yr_renovated, zipcode, sqft_living15, sqft_lot15: Integer (ordinal variables)

sqft_above, sqft_basement: Integer (whole numbers)

All columns appear to be correct except for the 'date' column, which appears to be of character type. We will proceed to change the 'date' column to an appropriate Date type.

Fixing the date column using lubridate library:

```r
housing_sales[2] <- housing_sales[2] %>%
  mutate(date = as.character(date)) %>%
  mutate(date = str_remove(date, "T000000")) %>%
  mutate(date = ymd(date))

str(housing_sales)
```

```
## 'data.frame':    21613 obs. of  21 variables:
##  $ id           : num  7.13e+09 6.41e+09 5.63e+09 2.49e+09 1.95e+09 ...
##  $ date         : Date, format: "2014-10-13" "2014-12-09" ...
##  $ price        : num  221900 538000 180000 604000 510000 ...
##  $ bedrooms     : num  3 3 2 4 3 4 3 3 3 3 ...
##  $ bathrooms    : num  1 2.25 1 3 2 4.5 2.25 1.5 1 2.5 ...
##  $ sqft_living  : num  1180 2570 770 1960 1680 ...
##  $ sqft_lot     : num  5650 7242 10000 5000 8080 ...
##  $ floors       : num  1 2 1 1 1 1 2 1 1 2 ...
##  $ waterfront   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ view         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ condition    : int  3 3 3 5 3 3 3 3 3 3 ...
##  $ grade        : int  7 7 6 7 8 11 7 7 7 7 ...
##  $ sqft_above   : int  1180 2170 770 1050 1680 3890 1715 1060 1050 1890 ...
##  $ sqft_basement: int  0 400 0 910 0 1530 0 0 730 0 ...
##  $ yr_built     : int  1955 1951 1933 1965 1987 2001 1995 1963 1960 2003 ...
##  $ yr_renovated : int  0 1991 0 0 0 0 0 0 0 0 ...
##  $ zipcode      : int  98178 98125 98028 98136 98074 98053 98003 98198 98146 98038 ...
##  $ lat          : num  47.5 47.7 47.7 47.5 47.6 ...
##  $ long         : num  -122 -122 -122 -122 -122 ...
##  $ sqft_living15: int  1340 1690 2720 1360 1800 4760 2238 1650 1780 2390 ...
```

```
##  $ sqft_lot15   : int  5650 7639 8062 5000 7503 101930 6819 9711 8113 7570 ...
```

Date column has been appropiately changed the type to Date format

1.2 Clean, wrangle, and handling missing data

1.2.1 Checking which columns contains missing data

```
missing_data_house_sales <- colSums(is.na(housing_sales))
missing_data_house_sales
```

```
##            id          date         price      bedrooms     bathrooms
##             0             0             0          1134          1068
##   sqft_living      sqft_lot        floors    waterfront          view
##          1110          1044             0             0             0
##     condition         grade    sqft_above sqft_basement      yr_built
##             0             0             0             0             0
##  yr_renovated       zipcode           lat          long sqft_living15
##             0             0             0             0             0
##    sqft_lot15
##             0
```

```
## Columns with missing data:
##  bedrooms:  1134
##  bathrooms:  1068
##  sqft_living:  1110
##  sqft_lot:  1044
```

We decided to eliminate rows with missing values in the "price" column.

```
housing_clean<- housing_sales %>%
  filter(price != is.na(price))
```

By removing all rows containing missing values, we ended up losing 3,995 rows, which accounts for approximately 18.5% of the total dataset.

1.3 Transforming Data

- **Normalization/Rescale:** We divided some variables by a constant value, for example:
  - **"Price":** was divided by 10,000 to create a new variable **"price_10000"**
  - **"sqft_living", "sqft_lot", "sqft_above", and "sqft_basement":** were divided by 100 to create variables like **"sqft_living100", "sqft_lot100", "sqft_above100", and "sqft_basement100"**
- **Feature Construction:** We created a new variable called **"renovated"** that checked if **"yr_renovated"** was true or not. And with this information we could know whether a house has been renovated or not.
- **Aggregation:** We used the aggregation function "case_when" for two variables:
  - **Bedrooms:** Were grouped into categories depending on their square footage. If the square footage is less than 1000, it's categorized as 1 bedroom, and so on, up to 7 bedrooms for square footage greater than 3000.
  - **Bathrooms:** Were grouped into categories ranging from 1 to 5 depending on their square footage.

```
housing_clean[15871,4]<- 3
```

```r
housing_clean <- housing_clean %>%
  mutate(price_10000 = price / 10000,
         sqft_living = ifelse(is.na(sqft_living), sqft_living15, sqft_living),
         sqft_living100 = sqft_living / 100,
         sqft_lot = ifelse(is.na(sqft_lot), sqft_lot15, sqft_lot),
         sqft_lot100 = sqft_lot / 100,
         sqft_above100 = sqft_above / 100,
         sqft_basement100 = sqft_basement / 100,
         bedrooms = case_when(
           is.na(bedrooms) & sqft_living < 1000 ~ 1,
           is.na(bedrooms) & sqft_living < 1500 ~ 2,
           is.na(bedrooms) & sqft_living < 2000 ~ 3,
           is.na(bedrooms) & sqft_living < 2600 ~ 4,
           is.na(bedrooms) & sqft_living < 2900 ~ 5,
           is.na(bedrooms) & sqft_living <= 3000 ~ 6,
           is.na(bedrooms) & sqft_living > 3000 ~ 7,
           .default = bedrooms
         ),
         bathrooms = case_when(
           is.na(bathrooms) & sqft_living < 1000 ~ 1,
           is.na(bathrooms) & sqft_living < 1500 ~ 2,
           is.na(bathrooms) & sqft_living < 3000 ~ 3,
           is.na(bathrooms) & sqft_living < 4000 ~ 4,
           is.na(bathrooms) & sqft_living >= 4000 ~ 5,
           .default = bathrooms
         ),
         zipcode=as.factor(zipcode),
         condition_f = as.factor(condition),
         waterfront_f = as.factor(waterfront),
         view_f = as.factor(view),
         grade_f = as.factor(grade),
         renovated = ifelse(yr_renovated != 0, 1, 0),
         renovated = as.factor(renovated)) %>%
  filter(bedrooms != 0)

colSums(is.na(housing_clean))
```

```
##               id             date            price         bedrooms
##                0                0                0                0
##        bathrooms      sqft_living         sqft_lot           floors
##                0                0                0                0
##       waterfront             view        condition            grade
##                0                0                0                0
##       sqft_above    sqft_basement         yr_built     yr_renovated
##                0                0                0                0
##          zipcode              lat             long    sqft_living15
##                0                0                0                0
##       sqft_lot15      price_10000   sqft_living100      sqft_lot100
##                0                0                0                0
##    sqft_above100 sqft_basement100      condition_f     waterfront_f
##                0                0                0                0
##           view_f          grade_f        renovated
##                0                0                0
```

1.4 Reducing Redundant Data and Performing Discretization

- **Discretization:** We converted the continuous data in **bedrooms** and **bathrooms** to discrete intervals.
  - **bedrooms:**

```
housing_clean<- housing_clean %>%
  filter(id != 6306400140 & id != 1453602309 & id != 6896300380 & id != 2954400190 &
           id != 2569500210 & id != 2310060040 & id != 3374500520 & id != 7849202190 &
           id != 7849202299 & id != 9543000205 & id != 1222029077)

housing_clean<- housing_clean %>%
  mutate(bed_fact = as.factor(bedrooms),
         bath_char = as.character(bathrooms),
         bath_fact = fct_collapse(bath_char,
    "0 to 1" = c("0", "0.5", "0.75", "1"),
    "1.25 to 2" = c("1.25", "1.5", "1.75", "2"),
    "2.25-3" = c("2.25", "2.5", "2.75", "3") ,
    "3.25-4" = c("3.25", "3.5", "3.75", "4"),
    "4.25-5" = c("4.25", "4.5", "4.75", "5"),
    "5.25 and up" = c("5.25", "5.5", "5.75", "6", "6.25", "6.5", "6.75",
                      "7.5", "7.75", "8")
  ))


housing_clean<- housing_clean %>%
  mutate(bedrooms = case_when(
    is.na(bedrooms) & sqft_living < 1000 ~ 1,
    is.na(bedrooms) & sqft_living < 1500 ~ 2,
    is.na(bedrooms) & sqft_living < 2000 ~ 3,
    is.na(bedrooms) & sqft_living < 2600 ~ 4,
    is.na(bedrooms) & sqft_living < 2900 ~ 5,
    is.na(bedrooms) & sqft_living <= 3000 ~ 6,
    is.na(bedrooms) & sqft_living > 3000 ~ 7,
    .default = bedrooms
  ))
```

- **Bathrooms:**

```
housing_clean<- housing_clean %>%
  mutate(bathrooms = case_when(
    is.na(bathrooms) & sqft_living < 1000 ~ 1,
    is.na(bathrooms) & sqft_living < 1500 ~ 2,
    is.na(bathrooms) & sqft_living < 3000 ~ 3,
    is.na(bathrooms) & sqft_living < 4000 ~ 4,
    is.na(bathrooms) & sqft_living > 4000 ~ 5,
    .default = bathrooms
  ))
```

2. Data Analysis and Visualization

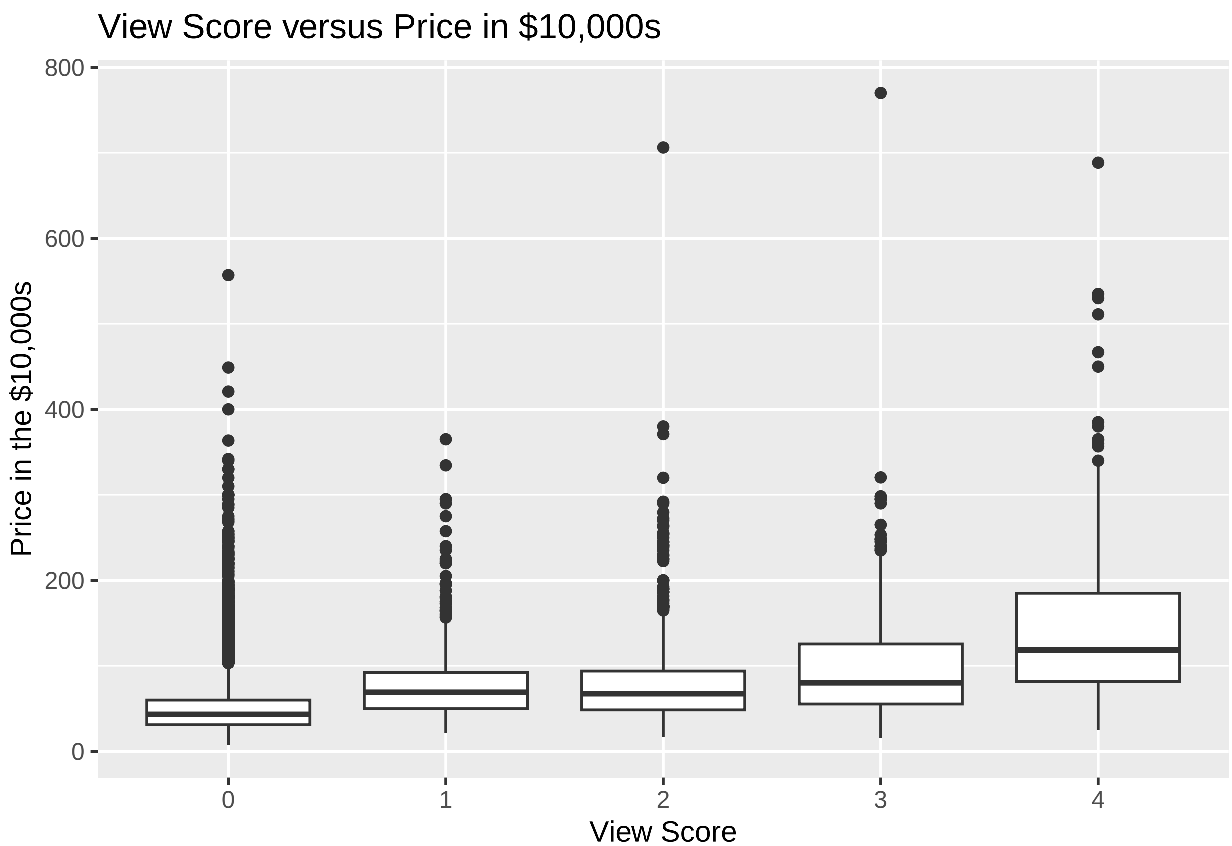
2.1 Identify categorical, ordinal, and numerical variables within data

- Categorical Variables:
  - **"zipcode":** Consist of discrete categories corresponding to different geographic locations.

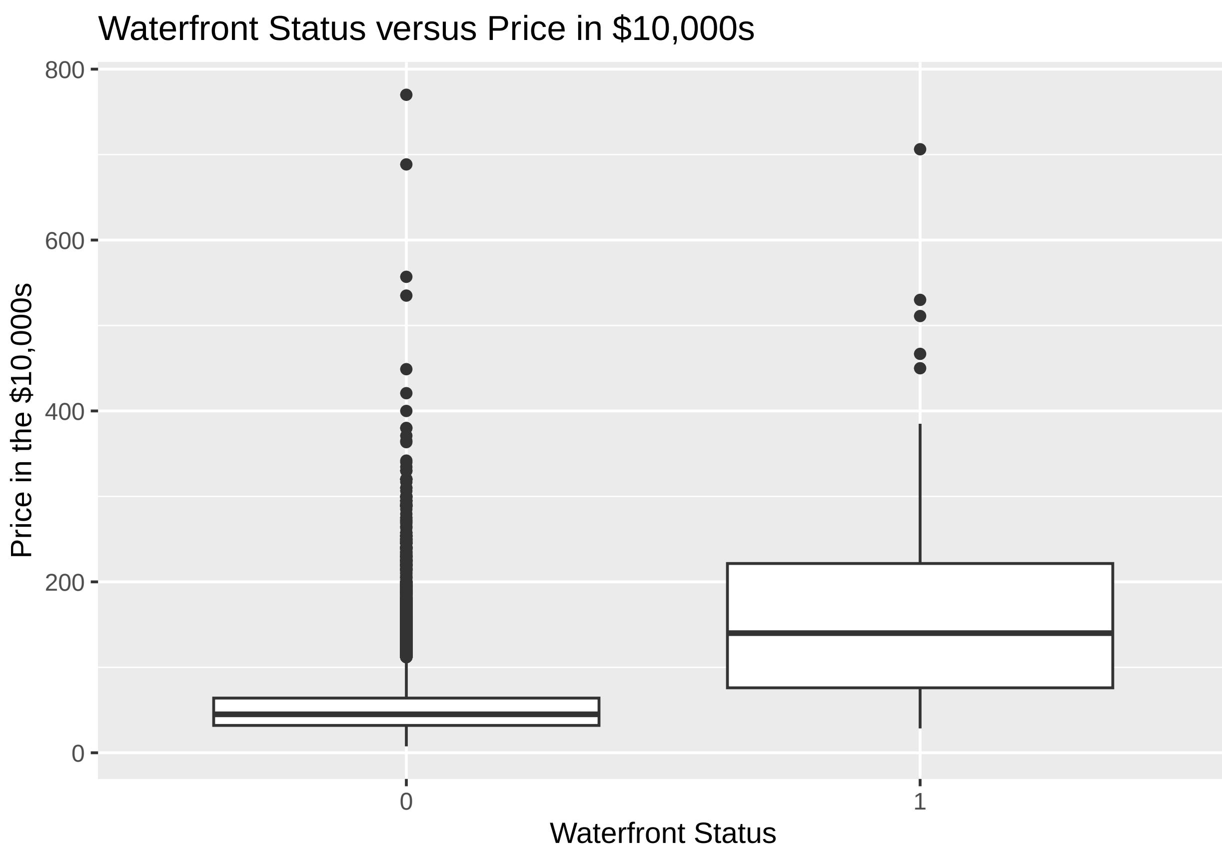
- Ordinal Variables:
  - **"waterfront", "view", "condition", "grade":** They all have ordered categories indicating if exists waterfront or not, how good is the view, the overall condition of the property, and the quality of the construction.
- Numerical Variables:
  - **"price", "sqft_living", "sqft_lot", "bedrooms", "bathrooms":** All these variables are continuous or discrete values.

2.2 Measures of centrality and distribution with visualizations

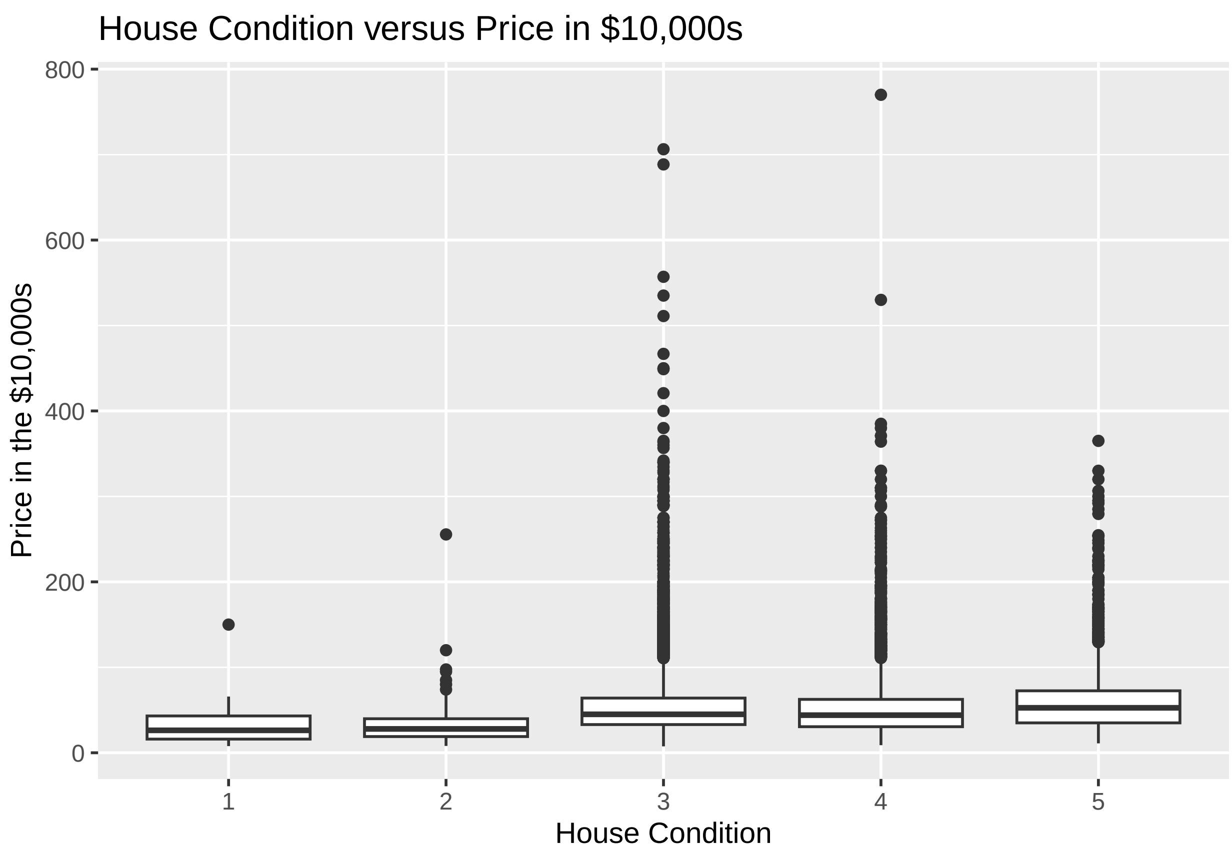
```
ggplot(data = housing_clean) +
  geom_boxplot(aes(x = view_f, y=price_10000)) +
  labs(title = "View Score versus Price in $10,000s") +
  xlab("View Score") +
  ylab("Price in the $10,000s")
```



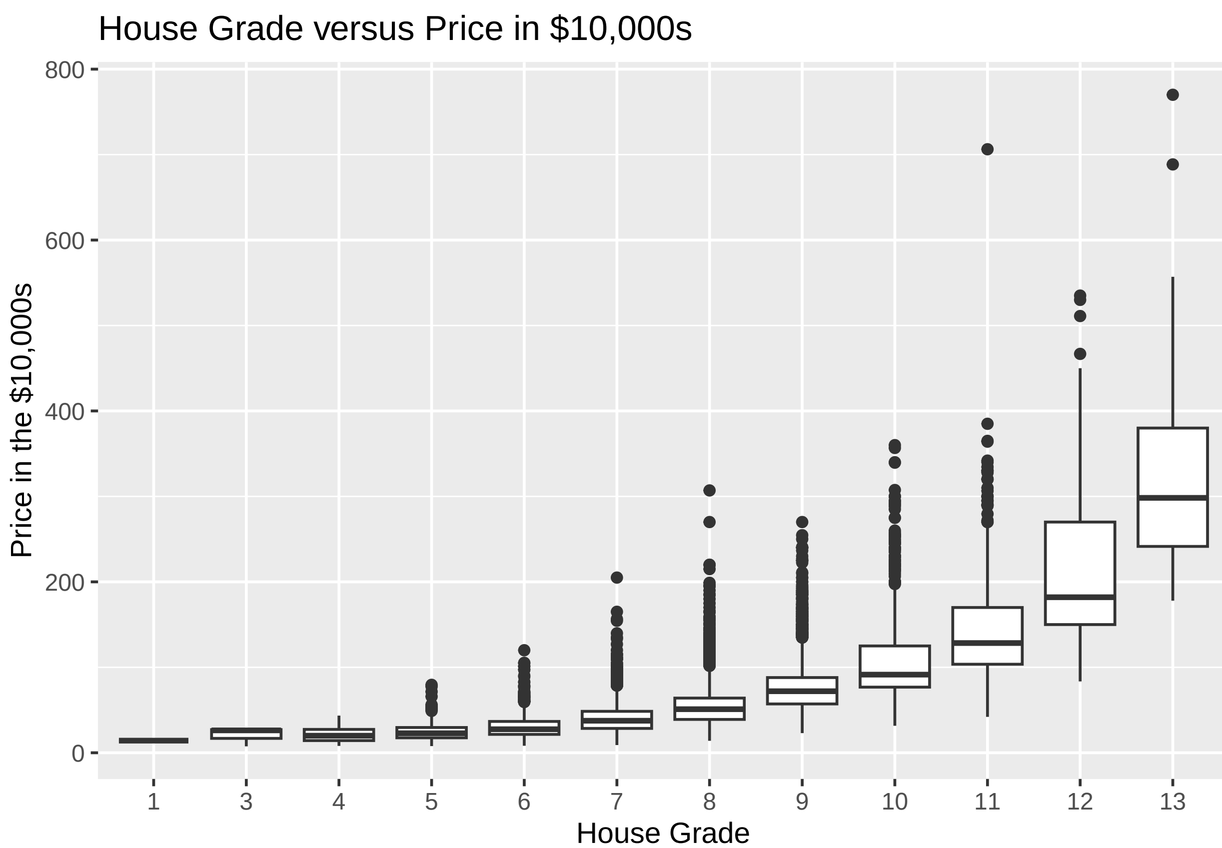

```
ggplot(data = housing_clean) +
  geom_boxplot(aes(x = waterfront_f, y=price_10000)) +
  labs(title = "Waterfront Status versus Price in $10,000s") +
  xlab("Waterfront Status") +
  ylab("Price in the $10,000s")
```



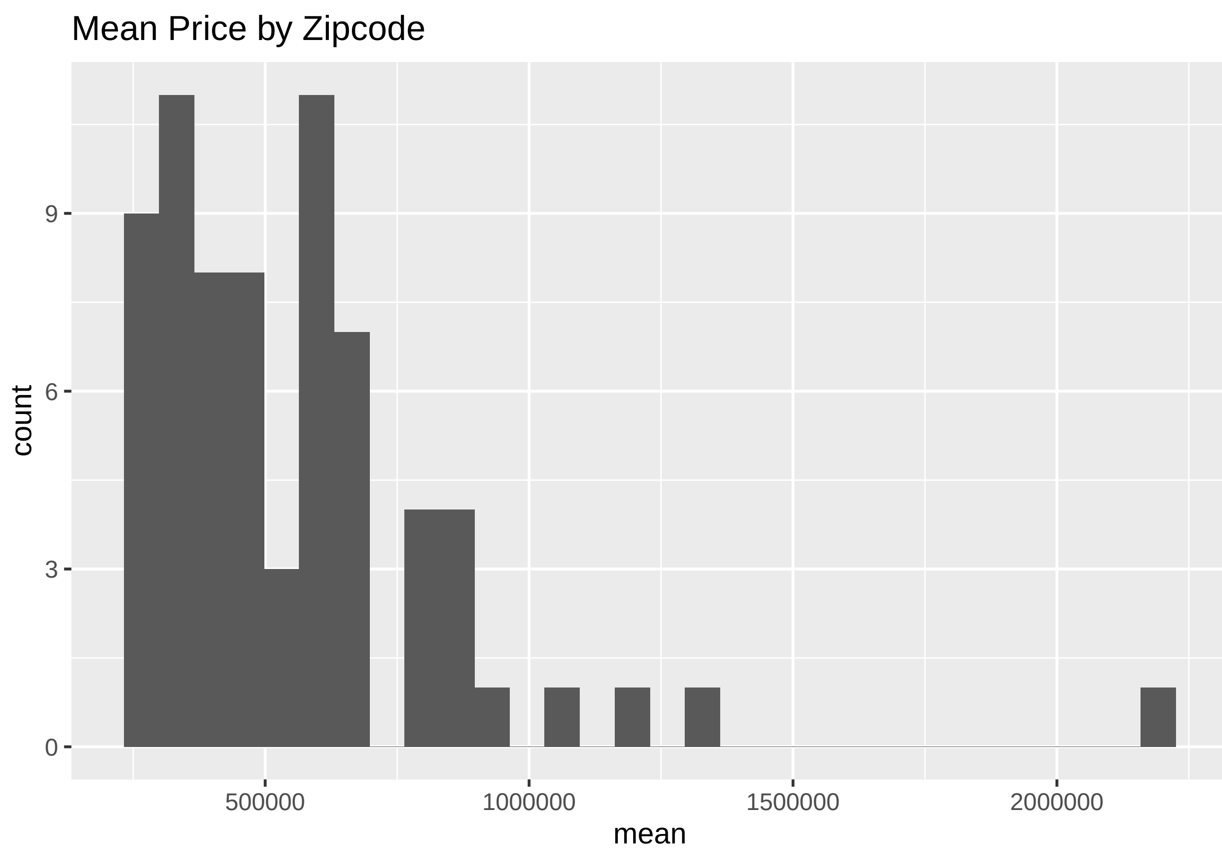

```
ggplot(data = housing_clean) +
  geom_boxplot(aes(x = condition_f, y=price_10000)) +
  labs(title = "House Condition versus Price in $10,000s") +
  xlab("House Condition") +
  ylab("Price in the $10,000s")
```




```r
ggplot(data = housing_clean) +
  geom_boxplot(aes(x = grade_f, y=price_10000)) +
  labs(title = "House Grade versus Price in $10,000s") +
  xlab("House Grade") +
  ylab("Price in the $10,000s")
```




```
housing_clean %>%
  group_by(zipcode) %>%
  summarise(mean = mean(price)) %>%
  ggplot() +
  geom_histogram((aes(x = mean))) +
  labs(title = "Mean Price by Zipcode")
```



2.3 Diagnose for correlations between variables and determine independent and dependent variables

```
# Correlation Analysis
correlation_matrix <- cor(housing_clean[, c("price", "bedrooms",
                                             "bathrooms", "sqft_living",
                                             "sqft_lot")])
print(head(correlation_matrix, 1))
```

```
##       price  bedrooms bathrooms sqft_living   sqft_lot
## price     1 0.3372049 0.5184271   0.6948486 0.08857508
```

These correlations shows that **"sqft_living" and "bathrooms"** have the strongest influence on the price, while the number of **"bedrooms"** has a weaker correlation. The **"sqft_lot"** has no correlation.

Therefore, the dependent variable will be **"price"**, and the predictors will be **"bedrooms", "bathrooms", and "sqft_living"**.

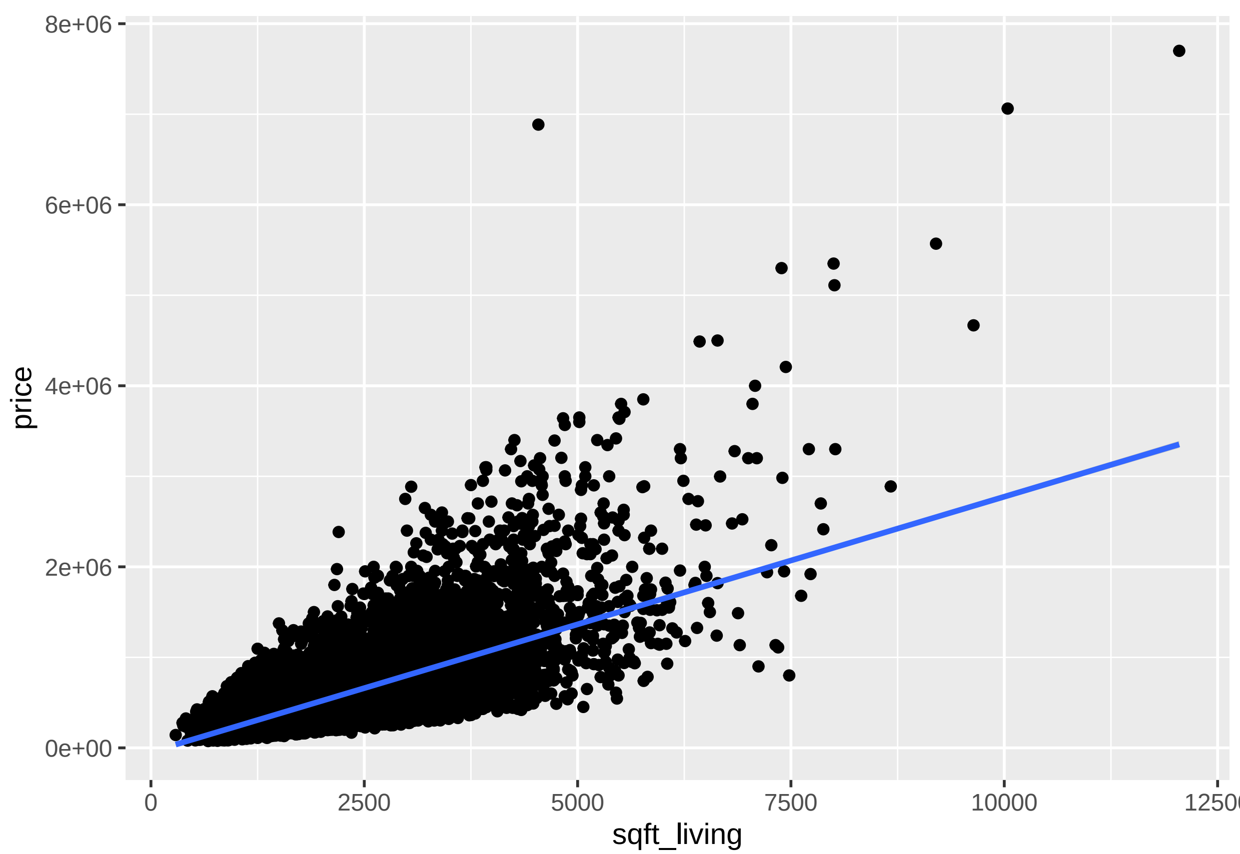
```
# Linear Regression
lm_model <- lm(price ~ bedrooms + bathrooms + sqft_living + sqft_lot,
               data = housing_clean)
summary(lm_model)
```

```
##
## Call:
## lm(formula = price ~ bedrooms + bathrooms + sqft_living + sqft_lot,
##     data = housing_clean)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

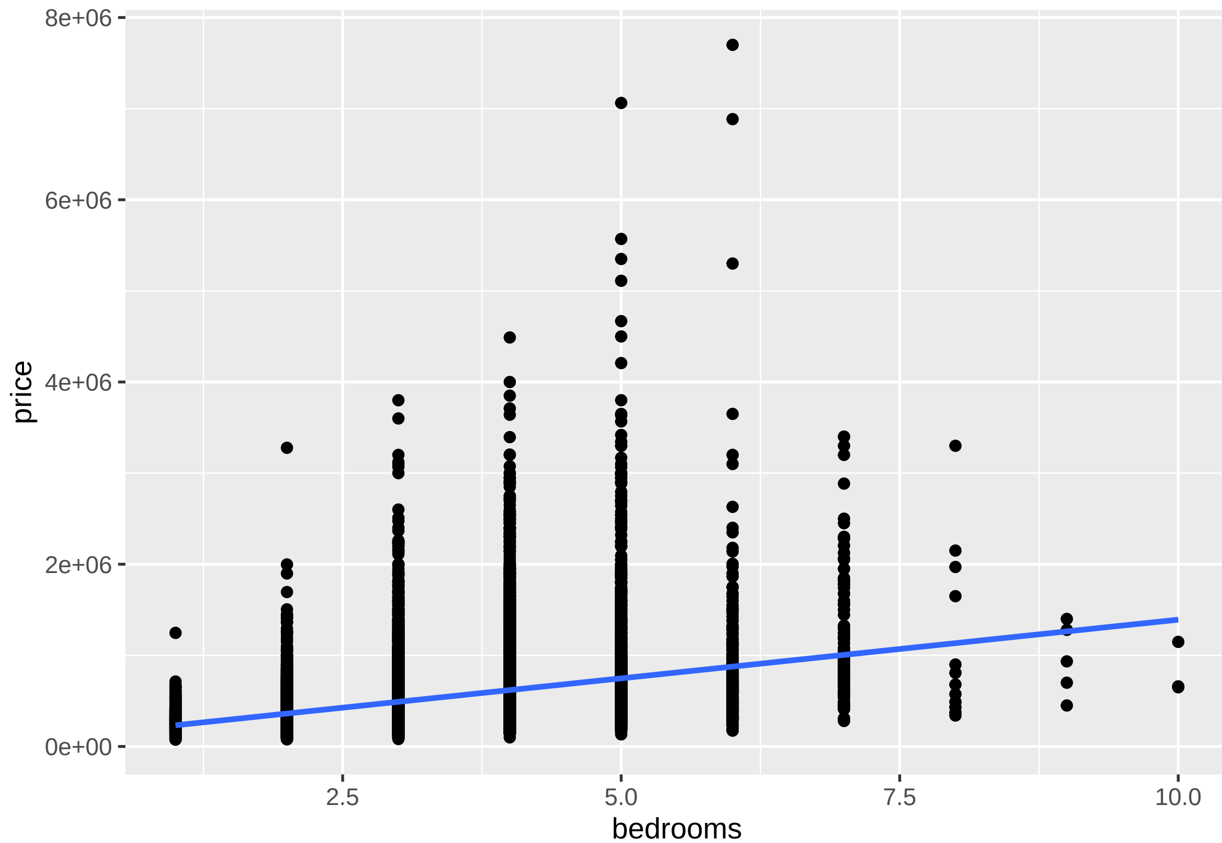
```
## -1252106  -145970   -23722   102558  5645216
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.149e+04  6.832e+03   7.537 5.02e-14 ***
## bedrooms    -5.299e+04  2.350e+03 -22.550  < 2e-16 ***
## bathrooms    1.459e+04  3.388e+03   4.308 1.66e-05 ***
## sqft_living  3.091e+02  3.191e+00  96.861  < 2e-16 ***
## sqft_lot    -3.289e-01  4.436e-02  -7.415 1.26e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 260800 on 21597 degrees of freedom
## Multiple R-squared:  0.4955, Adjusted R-squared:  0.4954
## F-statistic:  5303 on 4 and 21597 DF,  p-value: < 2.2e-16
```

- **Formula:** price = (5.149e+04) + (-5.299e+04 x bedrooms) + (1.459e+04 x bathrooms) + (3.091e+02 x sqft_living)

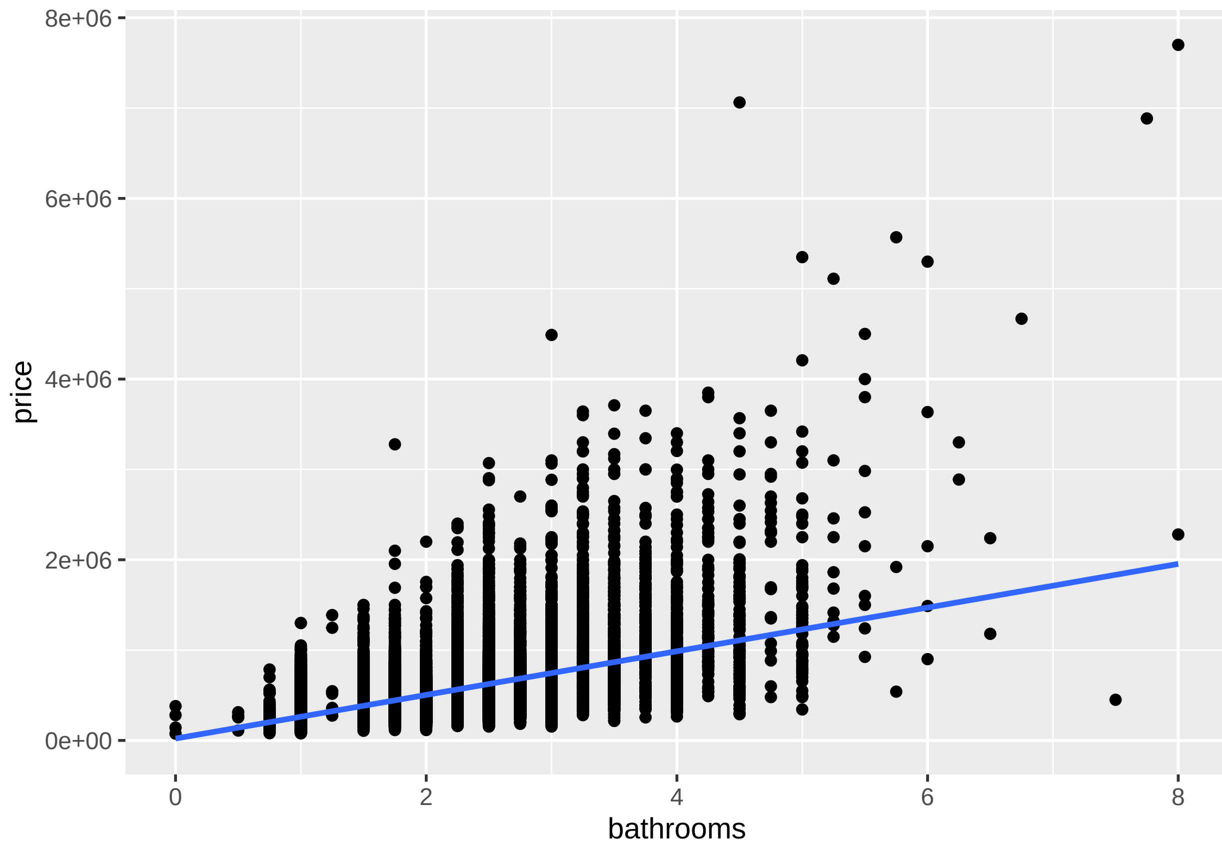
```
ggplot(data = housing_clean, aes(x=sqft_living, y=price)) +
  geom_point() +
  geom_smooth(method = "lm")
```




```
ggplot(data = housing_clean, aes(x=bedrooms, y=price)) +
  geom_point() +
  geom_smooth(method = "lm")
```




```
ggplot(data = housing_clean, aes(x=bathrooms, y=price)) +
  geom_point() +
  geom_smooth(method = "lm")
```



3. Data Analytics

3.1 Determine the need for a supervised or unsupervised learning method and identify dependent and independent variables

Based on our research of predicting **"price"** through **"bedrooms", "bathrooms", and "sqft_living"** as independent variables, we will need a supervised learning method.

The target variable is **"price"**, and the predictors are **"bedrooms", "bathrooms", and "sqft_living"**.

- **Formula:** price = (5.149e+04) + (-5.299e+04 x bedrooms) + (1.459e+04 x bathrooms) + (3.091e+02 x sqft_living)

3.2 Train, test, and provide accuracy and evaluation metrics for model results

```
set.seed(123456)
housing_split <- initial_split(housing_clean, prop=0.5)
housing_train <- training(housing_split)
housing_test <- testing(housing_split)
```

Building Linear Regression models with training data- PREDICTING PRICE

```
model0<- lm(price ~ 1, data = housing_train)
summary(model0)
```

```
##
## Call:
```

```
## lm(formula = price ~ 1, data = housing_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -460420 -214420  -88420  105580 7161580
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   538420       3545   151.9   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 368500 on 10800 degrees of freedom
```

```
model0_all<- lm(price ~ ., data = housing_train)
#summary(model0_all)
```

```
model1<- lm(price ~ sqft_living, data = housing_train)
summary(model1)
```

```
##
## Call:
## lm(formula = price ~ sqft_living, data = housing_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1268714  -148471   -22816   106707  4339216
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -46095.061   6376.966  -7.228 5.22e-13 ***
## sqft_living    282.729      2.826 100.036  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 265500 on 10799 degrees of freedom
## Multiple R-squared:  0.481,  Adjusted R-squared:  0.4809
## F-statistic: 1.001e+04 on 1 and 10799 DF,  p-value: < 2.2e-16
```

```
model2<- lm(price ~ sqft_living + bedrooms + bathrooms, data = housing_train)
summary(model2)
```

```
##
## Call:
## lm(formula = price ~ sqft_living + bedrooms + bathrooms, data = housing_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1255052  -145220   -24826   101581  4137991
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  58284.970   9674.403   6.025 1.75e-09 ***
## sqft_living    311.748      4.478  69.620  < 2e-16 ***
## bedrooms    -54487.221   3327.853 -16.373  < 2e-16 ***
## bathrooms     9259.872   4818.920   1.922   0.0547 .
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 262300 on 10797 degrees of freedom
## Multiple R-squared:  0.4936, Adjusted R-squared:  0.4934
## F-statistic:  3507 on 3 and 10797 DF,  p-value: < 2.2e-16
```

```r
model3<- lm(price ~ sqft_living + grade, data = housing_train)
summary(model3)
```

```
##
## Call:
## lm(formula = price ~ sqft_living + grade, data = housing_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1085695  -139336   -25267   101072  4754093
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.718e+05  1.925e+04  -29.70   <2e-16 ***
## sqft_living  1.911e+02  4.187e+00   45.64   <2e-16 ***
## grade        9.346e+04  3.243e+03   28.82   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 255800 on 10798 degrees of freedom
## Multiple R-squared:  0.518,  Adjusted R-squared:  0.5179
## F-statistic:  5803 on 2 and 10798 DF,  p-value: < 2.2e-16
```

```r
model4<- lm(price ~ condition + waterfront + view + grade, data = housing_train)
summary(model4)
```

```
##
## Call:
## lm(formula = price ~ condition + waterfront + view + grade, data = housing_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1658871  -133889   -25773    94397  5860133
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1198577      22651  -52.92   <2e-16 ***
## condition      66740       3771   17.70   <2e-16 ***
## waterfront    764534      31401   24.35   <2e-16 ***
## view           83190       3604   23.08   <2e-16 ***
## grade         193993       2184   88.84   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 253800 on 10796 degrees of freedom
## Multiple R-squared:  0.5258, Adjusted R-squared:  0.5256
## F-statistic:  2993 on 4 and 10796 DF,  p-value: < 2.2e-16
```

```r
model5<- lm(price ~ condition + waterfront + view + grade + sqft_living +
              bedrooms, data = housing_train)
summary(model5)
```

```
##
## Call:
## lm(formula = price ~ condition + waterfront + view + grade +
##     sqft_living + bedrooms, data = housing_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1301768  -125399   -16443    96179  4638356
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -7.054e+05  2.426e+04  -29.08   <2e-16 ***
## condition    6.061e+04  3.486e+03   17.39   <2e-16 ***
## waterfront   7.080e+05  2.901e+04   24.41   <2e-16 ***
## view         6.245e+04  3.361e+03   18.58   <2e-16 ***
## grade        9.587e+04  3.027e+03   31.68   <2e-16 ***
## sqft_living  1.894e+02  4.544e+00   41.68   <2e-16 ***
## bedrooms    -3.187e+04  2.982e+03  -10.69   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 234000 on 10794 degrees of freedom
## Multiple R-squared:  0.5968, Adjusted R-squared:  0.5965
## F-statistic:  2662 on 6 and 10794 DF,  p-value: < 2.2e-16
```

```r
model6<- lm(price ~ date + floors + condition + waterfront + view + grade +
              sqft_living + sqft_lot + bedrooms + bathrooms + yr_built +
              sqft_above + sqft_basement, data = housing_train)
summary(model6)
```

```
##
## Call:
## lm(formula = price ~ date + floors + condition + waterfront +
##     view + grade + sqft_living + sqft_lot + bedrooms + bathrooms +
##     yr_built + sqft_above + sqft_basement, data = housing_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1474384  -108765    -8697    89589  4230196
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    4.083e+06  3.588e+05  11.382  < 2e-16 ***
## date           1.219e+02  1.869e+01   6.522 7.23e-11 ***
## floors         3.441e+04  5.286e+03   6.509 7.90e-11 ***
## condition      2.118e+04  3.493e+03   6.063 1.38e-09 ***
## waterfront     7.057e+05  2.713e+04  26.013  < 2e-16 ***
## view           4.321e+04  3.228e+03  13.388  < 2e-16 ***
## grade          1.204e+05  3.105e+03  38.776  < 2e-16 ***
## sqft_living    4.517e+01  1.624e+01   2.781  0.00542 **
```

```
## sqft_lot      -2.618e-01  5.159e-02  -5.074 3.96e-07 ***
## bedrooms      -3.648e+04  2.855e+03 -12.779  < 2e-16 ***
## bathrooms      3.215e+04  4.635e+03   6.935 4.28e-12 ***
## yr_built      -3.510e+03  9.549e+01 -36.758  < 2e-16 ***
## sqft_above     1.351e+02  1.649e+01   8.190 2.92e-16 ***
## sqft_basement  1.432e+02  1.687e+01   8.492  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 218600 on 10787 degrees of freedom
## Multiple R-squared:  0.6483, Adjusted R-squared:  0.6479
## F-statistic:  1530 on 13 and 10787 DF,  p-value: < 2.2e-16
```

With a Multiple R^2 of 0.6483, that means that 64.83% of the variability in this dataset is explained by this model (model6).

Model 6: Top 5 Variables with highest absolute value of t-value 1. Grade: 38.776 2. yr_built: -36.758 3. waterfront: 26.013 4. view: 13.388 5. bedrooms:-12.779

Model 6 with transformed price, sqft_living, sqft_lot, sqft_above, and sqft_basement. This does NOT change any of the p-values, but it does make interpretation different.

Each increase in one unit of price_10000 = an increase of $10,000 Each increase in one unit of sqft_blank100 = an increase of 100 sqft

```r
model6_trans<- lm(price_10000 ~ date + floors + condition + waterfront + view +
                    grade + sqft_living100 + sqft_lot100 + bedrooms + bathrooms
                  + yr_built + sqft_above100 + sqft_basement100,
                  data = housing_train)
summary(model6_trans)
```

```
##
## Call:
## lm(formula = price_10000 ~ date + floors + condition + waterfront +
##     view + grade + sqft_living100 + sqft_lot100 + bedrooms +
##     bathrooms + yr_built + sqft_above100 + sqft_basement100,
##     data = housing_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -147.44  -10.88   -0.87    8.96  423.02
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       4.083e+02  3.588e+01  11.382  < 2e-16 ***
## date              1.219e-02  1.869e-03   6.522 7.23e-11 ***
## floors            3.441e+00  5.286e-01   6.509 7.90e-11 ***
## condition         2.118e+00  3.493e-01   6.063 1.38e-09 ***
## waterfront        7.057e+01  2.713e+00  26.013  < 2e-16 ***
## view              4.321e+00  3.228e-01  13.388  < 2e-16 ***
## grade             1.204e+01  3.105e-01  38.776  < 2e-16 ***
## sqft_living100    4.517e-01  1.624e-01   2.781  0.00542 **
## sqft_lot100      -2.618e-03  5.159e-04  -5.074 3.96e-07 ***
## bedrooms         -3.648e+00  2.855e-01 -12.779  < 2e-16 ***
## bathrooms         3.215e+00  4.635e-01   6.935 4.28e-12 ***
## yr_built         -3.510e-01  9.549e-03 -36.758  < 2e-16 ***
## sqft_above100     1.351e+00  1.649e-01   8.190 2.92e-16 ***
```

```
## sqft_basement100  1.432e+00  1.687e-01   8.492  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.86 on 10787 degrees of freedom
## Multiple R-squared:  0.6483, Adjusted R-squared:  0.6479
## F-statistic:  1530 on 13 and 10787 DF,  p-value: < 2.2e-16
```

What if we try to account for interactions?

```
model7<- lm(price_10000 ~ date + floors + condition + waterfront + view +
              grade + sqft_living100 + sqft_lot100 + bedrooms + bathrooms +
              yr_built + sqft_above100 + sqft_basement100 + bedrooms:bathrooms
            + sqft_living100:sqft_lot100, data = housing_train)
summary(model7)
```

```
##
## Call:
## lm(formula = price_10000 ~ date + floors + condition + waterfront +
##     view + grade + sqft_living100 + sqft_lot100 + bedrooms +
##     bathrooms + yr_built + sqft_above100 + sqft_basement100 +
##     bedrooms:bathrooms + sqft_living100:sqft_lot100, data = housing_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -148.78  -10.65   -0.77    8.81  385.53
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 3.875e+02  3.553e+01  10.907  < 2e-16 ***
## date                        1.230e-02  1.849e-03   6.653 3.01e-11 ***
## floors                      4.045e+00  5.246e-01   7.710 1.36e-14 ***
## condition                   2.614e+00  3.473e-01   7.526 5.65e-14 ***
## waterfront                  6.982e+01  2.685e+00  26.003  < 2e-16 ***
## view                        4.132e+00  3.199e-01  12.915  < 2e-16 ***
## grade                       1.258e+01  3.094e-01  40.653  < 2e-16 ***
## sqft_living100              4.541e-01  1.610e-01   2.821  0.00480 **
## sqft_lot100                 6.333e-04  1.098e-03   0.577  0.56401
## bedrooms                   -1.092e+01  5.579e-01 -19.567  < 2e-16 ***
## bathrooms                  -8.829e+00  9.248e-01  -9.547  < 2e-16 ***
## yr_built                   -3.313e-01  9.548e-03 -34.701  < 2e-16 ***
## sqft_above100               1.263e+00  1.633e-01   7.737 1.11e-14 ***
## sqft_basement100            1.416e+00  1.669e-01   8.485  < 2e-16 ***
## bedrooms:bathrooms          3.324e+00  2.214e-01  15.014  < 2e-16 ***
## sqft_living100:sqft_lot100 -1.121e-04  3.491e-05  -3.210  0.00133 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.64 on 10785 degrees of freedom
## Multiple R-squared:  0.6557, Adjusted R-squared:  0.6552
## F-statistic:  1369 on 15 and 10785 DF,  p-value: < 2.2e-16
```

Adding interactions between bedrooms and bathrooms, along with sqft_living100 and sqft_lot100 has rendered sqft_lot100 not significant. It has increased the t-value of grade by 2.

The R^2 value is 0.6557, which means that 65.57% of the variability in this dataset can be explained by this model.

```
model7a<- lm(price_10000 ~ date + floors + condition + waterfront + view
             + grade + sqft_living100 + bedrooms + bathrooms + yr_built +
               sqft_above100 + sqft_basement100 + bedrooms:bathrooms +
               sqft_living100:sqft_lot100, data = housing_train)
summary(model7a)
```

```
##
## Call:
## lm(formula = price_10000 ~ date + floors + condition + waterfront +
##     view + grade + sqft_living100 + bedrooms + bathrooms + yr_built +
##     sqft_above100 + sqft_basement100 + bedrooms:bathrooms + sqft_living100:sqft_lot100,
##     data = housing_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -148.88  -10.66   -0.76    8.84  385.49
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 3.874e+02  3.553e+01  10.905  < 2e-16 ***
## date                        1.229e-02  1.849e-03   6.648 3.12e-11 ***
## floors                      4.028e+00  5.237e-01   7.690 1.59e-14 ***
## condition                   2.616e+00  3.473e-01   7.533 5.35e-14 ***
## waterfront                  6.981e+01  2.685e+00  26.001  < 2e-16 ***
## view                        4.143e+00  3.193e-01  12.975  < 2e-16 ***
## grade                       1.257e+01  3.093e-01  40.652  < 2e-16 ***
## sqft_living100              4.495e-01  1.608e-01   2.795  0.00519 **
## bedrooms                   -1.090e+01  5.568e-01 -19.569  < 2e-16 ***
## bathrooms                  -8.810e+00  9.242e-01  -9.532  < 2e-16 ***
## yr_built                   -3.312e-01  9.543e-03 -34.702  < 2e-16 ***
## sqft_above100               1.266e+00  1.632e-01   7.754 9.68e-15 ***
## sqft_basement100            1.417e+00  1.669e-01   8.488  < 2e-16 ***
## bedrooms:bathrooms          3.317e+00  2.211e-01  15.005  < 2e-16 ***
## sqft_living100:sqft_lot100 -9.425e-05  1.624e-05  -5.804 6.66e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.63 on 10786 degrees of freedom
## Multiple R-squared:  0.6557, Adjusted R-squared:  0.6552
## F-statistic:  1467 on 14 and 10786 DF,  p-value: < 2.2e-16
```

Removing the insignificant term does not change the R-Squared.

What if we removed the variables that are included in the interactions, and kept only the interactions?

```
model8<- lm(price_10000 ~ date + floors + condition + waterfront + view +
              grade + yr_built + sqft_above100 + sqft_basement100 +
              bedrooms:bathrooms + sqft_living100:sqft_lot100,
            data = housing_train)
summary(model8)
```

```
##
## Call:
## lm(formula = price_10000 ~ date + floors + condition + waterfront +
```

```
##     view + grade + yr_built + sqft_above100 + sqft_basement100 +
##     bedrooms:bathrooms + sqft_living100:sqft_lot100, data = housing_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -143.51  -11.23   -0.75    9.07  442.52
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 3.720e+02  3.589e+01  10.367  < 2e-16 ***
## date                        1.210e-02  1.885e-03   6.420 1.42e-10 ***
## floors                      4.097e+00  5.200e-01   7.879 3.63e-15 ***
## condition                   2.026e+00  3.513e-01   5.768 8.26e-09 ***
## waterfront                  7.236e+01  2.734e+00  26.469  < 2e-16 ***
## view                        4.633e+00  3.246e-01  14.274  < 2e-16 ***
## grade                       1.280e+01  3.098e-01  41.301  < 2e-16 ***
## yr_built                   -3.365e-01  9.295e-03 -36.205  < 2e-16 ***
## sqft_above100               1.623e+00  5.186e-02  31.301  < 2e-16 ***
## sqft_basement100            1.741e+00  6.604e-02  26.358  < 2e-16 ***
## bedrooms:bathrooms          9.126e-02  7.930e-02   1.151     0.25
## sqft_living100:sqft_lot100 -7.142e-05  1.651e-05  -4.326 1.53e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 22.06 on 10789 degrees of freedom
## Multiple R-squared:  0.6421, Adjusted R-squared:  0.6417
## F-statistic:  1759 on 11 and 10789 DF,  p-value: < 2.2e-16
```

Doing this renders the bedrooms and bathrooms interaction insignificant. We would suggest that we keep separate bedrooms and bathrooms terms.

What if we kept the separate bedrooms and bathrooms terms, plus their interaction, but removed the interaction for sqft_living100 and sqft_lot100?

```r
model9<- lm(price_10000 ~ date + bedrooms + bathrooms + floors + condition
            + waterfront + view + grade + yr_built + sqft_above100 +
              sqft_basement100 + sqft_living100 + sqft_lot100 +
              bedrooms:bathrooms , data = housing_train)
summary(model9)
```

```
##
## Call:
## lm(formula = price_10000 ~ date + bedrooms + bathrooms + floors +
##     condition + waterfront + view + grade + yr_built + sqft_above100 +
##     sqft_basement100 + sqft_living100 + sqft_lot100 + bedrooms:bathrooms,
##     data = housing_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -149.35  -10.64   -0.76    8.81  385.88
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         3.881e+02  3.554e+01  10.918  < 2e-16 ***
## date                1.221e-02  1.850e-03   6.602 4.23e-11 ***
## bedrooms           -1.077e+01  5.563e-01 -19.362  < 2e-16 ***
```

```
## bathrooms          -8.718e+00  9.246e-01  -9.429  < 2e-16 ***
## floors              4.021e+00  5.247e-01   7.663 1.97e-14 ***
## condition           2.622e+00  3.474e-01   7.546 4.84e-14 ***
## waterfront          6.982e+01  2.686e+00  25.992  < 2e-16 ***
## view                4.182e+00  3.197e-01  13.083  < 2e-16 ***
## grade               1.257e+01  3.095e-01  40.624  < 2e-16 ***
## yr_built           -3.308e-01  9.551e-03 -34.631  < 2e-16 ***
## sqft_above100       1.265e+00  1.634e-01   7.741 1.07e-14 ***
## sqft_basement100    1.415e+00  1.670e-01   8.476  < 2e-16 ***
## sqft_living100      4.242e-01  1.608e-01   2.638  0.00835 **
## sqft_lot100        -2.486e-03  5.108e-04  -4.867 1.15e-06 ***
## bedrooms:bathrooms  3.288e+00  2.212e-01  14.865  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.64 on 10786 degrees of freedom
## Multiple R-squared:  0.6554, Adjusted R-squared:  0.6549
## F-statistic:  1465 on 14 and 10786 DF,  p-value: < 2.2e-16
```

Doing so gives us an R^2 value of 0.6544, which means that 65.44% of the variability in the dataset can be explained by this model (model9)

A summary of the models created so far

Model	R^2
model0	NA
model1	0.481
model2	0.4936
model3	0.518
model4	0.5258
model5	0.5968
model6	0.6483
model7	0.6557
model7a	0.6557
model8	0.6421
model9	0.6554

From these models, we will move forward with model7a, as it has the highest R^2 value, and less predictor variables than model7.

Applying the model to the data

We are going to take the model7a and apply it to the testing data using the function "augment". This will add some columns to the housing_test tibble, which provides the estimates based on model7a.

```
fit_7a<- model7a %>%
  augment(housing_test)

fit_7a %>%
  rmse(price_10000, .fitted) %>%
     pull(.estimate)
```

```
## [1] 46.87993
```

```r
fit_9<- model9 %>%
  augment(housing_test)

fit_9 %>%
  rmse(price_10000, .fitted) %>%
     pull(.estimate)
```

```
## [1] 46.87624
```