Pricing Model for Residential Properties in King County, Washington

HARVARD EXTENSION SCHOOL

John Hur Kacper Lewtak German Parades Lauren Briese Michael Lefkoe
Rajesh Jain

17 December 2023

Abstract

This group homework is part of Harvard Extesion course CSCI E-106 Section 1 (Data Modelling) class. This study examines the data of homes sold between May 2014 and May 2015 in King County, USA and proposes a Polynomial Regression Model to enable future predictions and decision making in the real estate market.

Contents

House Sales in King County, USA data to be used in the Final Project

Variable	Description
id	**Unique ID for each home sold (it is not a predictor)**
date	*Date of the home sale*
price	*Price of each home sold*
bedrooms	*Number of bedrooms*
bathrooms	*Number of bathrooms, where ".5" accounts for a bathroom with a toilet but no shower*
sqft_living	*Square footage of the apartment interior living space*
sqft_lot	*Square footage of the land space*
floors	*Number of floors*
waterfront	*A dummy variable for whether the apartment was overlooking the waterfront or not*
view	*An index from 0 to 4 of how good the view of the property was*
condition	*An index from 1 to 5 on the condition of the apartment,*
grade	*An index from 1 to 13, where 1-3 falls short of building construction and design, 7 has an average level of construction and design, and 11-13 has a high-quality level of construction and design.*
sqft_above	*The square footage of the interior housing space that is above ground level*
sqft_basement	*The square footage of the interior housing space that is below ground level*
yr_built	*The year the house was initially built*
yr_renovated	*The year of the house's last renovation*
zipcode	*What zipcode area the house is in*
lat	*Latitude*
long	*Longitude*
sqft_living15	*The square footage of interior housing living space for the nearest 15 neighbors*
sqft_lot15	*The square footage of the land lots of the nearest 15 neighbors*

Executive Summary

A study was conducted on the house sale prices between May 2014 to May 2015 in King County, USA. The purpose of the study was to build a predictive model to enable decision making and to reduce risk during our automated underwriting process that relies on automated valuation of properties. Through this study, we analyzed data on more than 21,000 houses sold during the year, and across 17 variables that included, among others, the number of bedrooms, the number of bathrooms, the square footage of the house, zip code. We propose a Polynomial Regression Model in this study.

The model is intended to be used as a tool to help make decisions and to reduce risk and enable faster pre-approvals, and is not intended to be used as a replacement for a professional appraisal as it has several limitations including inconsistencies in data that need further investigation. The model is also based on a limited geographical area and may not be applicable to other areas outside of King County, USA. Because the model is based on a limited time frame, it should be continually monitored and updated with the latest pricing.

I. Introduction (5 points)

I.I Problem Statement

Our current pre-approval process relies on property value based on just 6 buckets corresponding to the conforming, jumbo and subprime loan limits based on Zip Code. This leads to sub-optimal decision making that increases the risk of default, and at the same time, leads to lost opportunities for the bank by denying a pre-approval to a person who is otherwise highly credit worthy.

I.II Purpose and Scope

The goal of this study is to build a model that can be used to predict the value of a property more accurately based on the characteristics of the property. The scope of this study is limited to the King County, USA area and is based on the actual sales price of over 21,613 houses sold between May 2014 and May 2015 and is based on 17 characteristics of a typical home.

I.III Methods

The predictive model was developed using Statistical Analysis using multiple regression models. We employed rigorous exploratory analysis of data, built multiple models including Linear, Polynomial, Weighted Least Square and Regression Tree models, and used a variety of statistical tests to ensure that the model is robust and reliable.

We started with extensive exploratory analysis to understand the data and to identify any missing values, outliers and inconsistencies. Through this analysis, we excluded variables that were either not relevant to the model as they were not a characteristic of a property, or were highly correlated to avoid overfitting. Next we split the data and utilized 70% of the observations to train a model, and the rest 30% to test the performance of the models.

We built multiple candidate Linear models, and utilized statistical techniques, to identify and retain the characteristics of the property which were most significant in explaining the price of a house. We also conducted a variety of both visual and statistical tests to ensure that the model is compliant with the assumptions of Normality, Linearity, Homoscedasticity and Independence.This helped narrow down to a "champion" model, which formed our baseline for further analysis.

Next, we built some non-linear "challenger" models, including Polynomial, Weighted Least Square and Regression Tree to compare the robustness, reliability and performance of the models, the most promising of which were a Polynomial model and the Regression Tree model. We again conducted numerical statistical tests by using the 30% test data to understand a models predictive power. We concluded by picking a Polynomial model, but also suggesting Regression Tree could be an alternate because the performance of the models was relatively close.

I.IV Limitations

Lastly, we ensured we documented the limitations and risks of such a model given that it was based on a limited geographical area and a limited time frame. We also identified some inconsistencies in the data that need further investigation and could result in incorrect predictions.

II. Description of the data and quality (15 points)

II.I Data Load and Inspection

```
# read in data
kc_house_sales = read.csv('KC_House_Sales.csv')

# get number of N/A values in each column
colSums(is.na(kc_house_sales))
```

```
##            id          date         price      bedrooms     bathrooms
##             0             0             0             0             0
##   sqft_living      sqft_lot        floors    waterfront          view
##             0             0             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
```

```
# get column names, data types, and first few values for each column
str(kc_house_sales)
```

```
## 'data.frame':    21613 obs. of  21 variables:
##  $ id           : num  7129300520 6414100192 5631500400 2487200875 1954400510 ...
##  $ date         : chr  "20141013T000000" "20141209T000000" "20150225T000000" "20141209T000000" ...
##  $ price        : chr  " $221,900.00 " " $538,000.00 " " $180,000.00 " " $604,000.00 " ...
##  $ bedrooms     : int  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  : int  1180 2570 770 1960 1680 5420 1715 1060 1780 1890 ...
##  $ sqft_lot     : int  5650 7242 10000 5000 8080 101930 6819 9711 7470 6560 ...
##  $ 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 ...
```

The data to be analyzed in this report is a set of housing sales data from King County, USA in the 2014-2015 time frame. King County is the most populous county of Washington, and the county seat is Seattle [1]. The original data set contains 21 columns with 21613 observations. There are no N/A values in the data set that need to be removed or addressed. Although there are some questionable values in the data set (for example, a house with 33 bedrooms, houses with 0 bathrooms and 0 bedrooms), there are few enough and without additional context or the ability to get clarification on these possible outliers, we did not drop any observations before developing the models.

II.II Data Cleanup

We took the following steps to prepare data prior to developing the models.

```
# convert the price to a numeric column
kc_house_sales$price = as.numeric(gsub('\\$|,','',kc_house_sales$price))
```

We updated the price column to numeric as it is the response variable which we built the models to predict.

```
# create n_yrs_ago_built
kc_house_sales$date = as.numeric(substr(kc_house_sales$date, start = 1, stop = 4))
kc_house_sales$n_yrs_ago_built = max(kc_house_sales$date) - kc_house_sales$yr_built
```

We used the difference of the maximum year from the sales date and the year built to create a new variable **n_yrs_ago_built** to determine how many years ago the house was built.

```
# convert zip code to characters so it will create dummy variables
kc_house_sales$zipcode = as.character(kc_house_sales$zipcode)

# convert waterfront to categorical
kc_house_sales$waterfront = as.factor(kc_house_sales$waterfront)
```

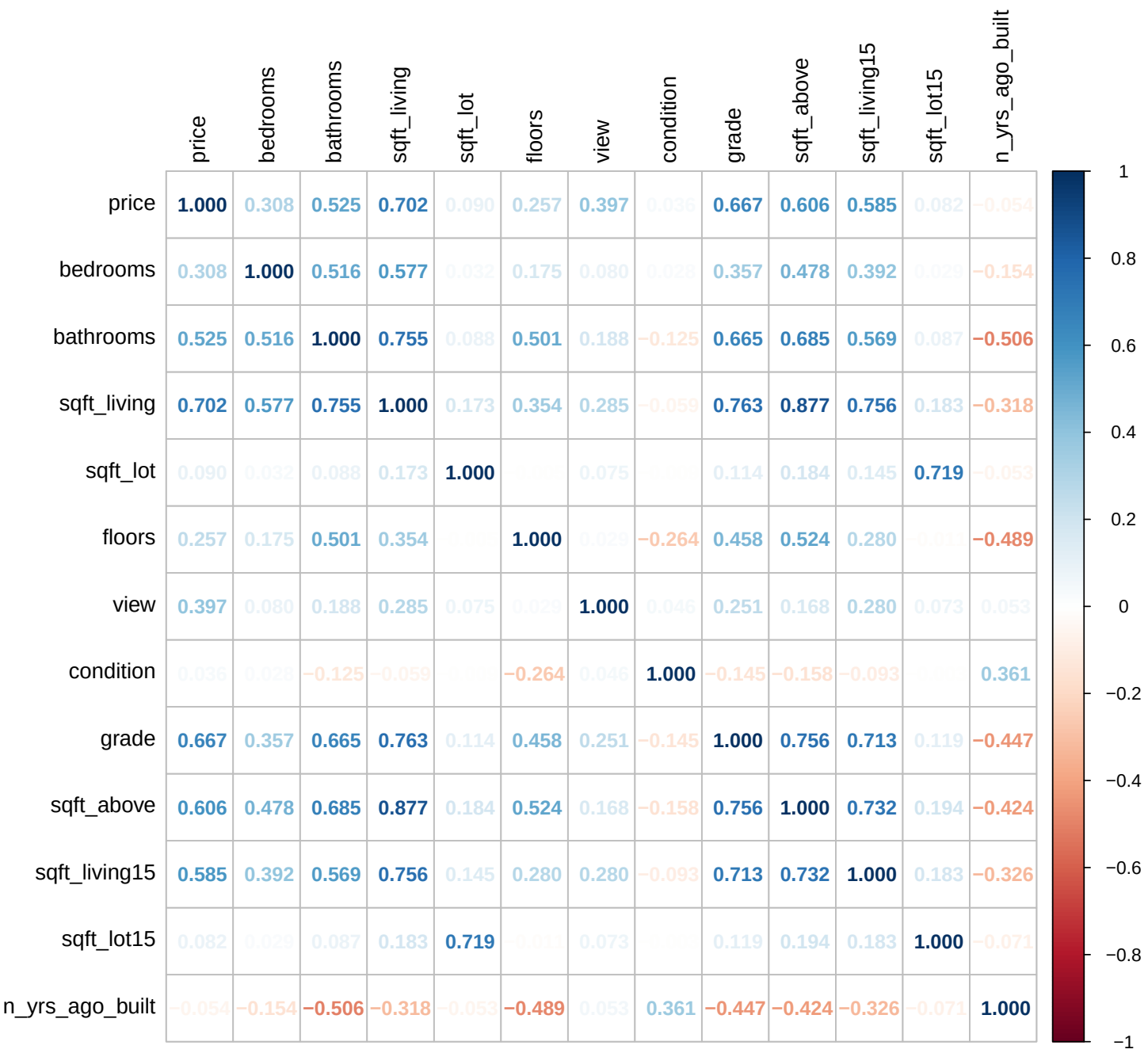
We converted zip code to a character column and treated it as a categorical variable. Although it appears numeric, zip codes are used as identifiers. In addition, waterfront is categorical per the description from the original data set, and was converted to a factor. The date field is categorical, but we chose to drop it.

```
kc_house_sales = subset(kc_house_sales, select = -c(id, lat, long, yr_built,
                                                    yr_renovated, sqft_basement, date))
```

We dropped the following columns prior to beginning model development:
- id: This is not a predictor or relevant to the sales data beyond being an identifier for each row.
- lat: This level of detail is not necessary, and the zipcode column also approximates location.
- long: This level of detail is not necessary, and the zipcode column also approximates location. - date: This was removed since all house sales were within the same 13-month period. - yr_built: Removed in favor of the calculated n_yrs_ago_built. - yr_renovated: Removed in favor of the calculated n_yrs_ago_built. - sqft_basement: This column is equal to sqft_living - sqft_above, so it does not provide new information and its inclusion would lead to multicollinearity.

II.III Correlation for Numeric Variables

```
# get correlation of numeric data, rounded for ease of review)
kc_house_sales_numeric = subset(kc_house_sales, select = -c(waterfront, zipcode))

#correlation plot
cor_matrix <- round(cor(kc_house_sales_numeric),3)
corrplot(cor_matrix, method = "number", title = "Correlation Matrix",
         tl.col = 'black', mar=c(0,0,1,0), number.digits = 3, number.cex=0.9)
```



	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	view	condition	grade	sqft_above	sqft_living15	sqft_lot15	n_yrs_ago_built
price	1.000	0.308	0.525	0.702	0.090	0.257	0.397	0.036	0.667	0.606	0.585	0.082	−0.054
bedrooms	0.308	1.000	0.516	0.577	0.032	0.175	0.080	0.028	0.357	0.478	0.392	0.029	−0.154
bathrooms	0.525	0.516	1.000	0.755	0.088	0.501	0.188	−0.125	0.665	0.685	0.569	0.087	−0.506
sqft_living	0.702	0.577	0.755	1.000	0.173	0.354	0.285	−0.059	0.763	0.877	0.756	0.183	−0.318
sqft_lot	0.090	0.032	0.088	0.173	1.000	−0.005	0.075	−0.009	0.114	0.184	0.145	0.719	−0.053
floors	0.257	0.175	0.501	0.354	−0.005	1.000	0.029	−0.264	0.458	0.524	0.280	−0.011	−0.489
view	0.397	0.080	0.188	0.285	0.075	0.029	1.000	0.046	0.251	0.168	0.280	0.073	0.053
condition	0.036	0.028	−0.125	−0.059	−0.009	−0.264	0.046	1.000	−0.145	−0.158	−0.093	−0.003	0.361
grade	0.667	0.357	0.665	0.763	0.114	0.458	0.251	−0.145	1.000	0.756	0.713	0.119	−0.447
sqft_above	0.606	0.478	0.685	0.877	0.184	0.524	0.168	−0.158	0.756	1.000	0.732	0.194	−0.424
sqft_living15	0.585	0.392	0.569	0.756	0.145	0.280	0.280	−0.093	0.713	0.732	1.000	0.183	−0.326
sqft_lot15	0.082	0.029	0.087	0.183	0.719	−0.011	0.073	−0.003	0.119	0.194	0.183	1.000	−0.071
n_yrs_ago_built	−0.054	−0.154	−0.506	−0.318	−0.053	−0.489	0.053	0.361	−0.447	−0.424	−0.326	−0.071	1.000

Looking at the correlation matrix above for the numeric columns in the data set, price appears to be most correlated with the fields sqft_living (0.702), grade (0.667), and sqft_above (0.606). **Note that the categorical variables waterfront and zipcode are not included in the correlation matrix.**

The sqft_living and sqft_above variables have a high correlation; it may be worth including only one of these in the model. For houses without basements, the sqft_living and sqft_above values should be equal.

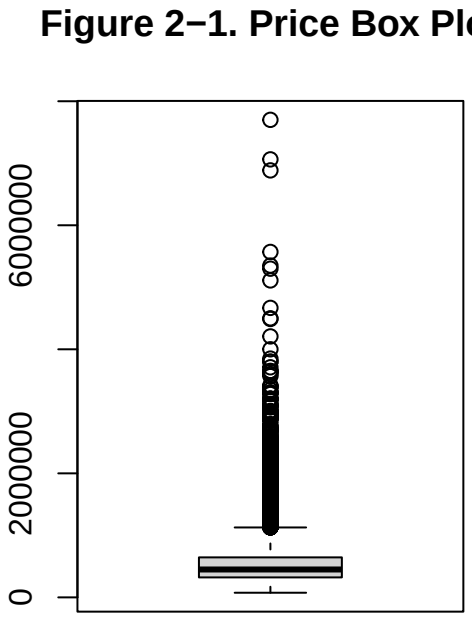
II.IV Exploratory Analysis

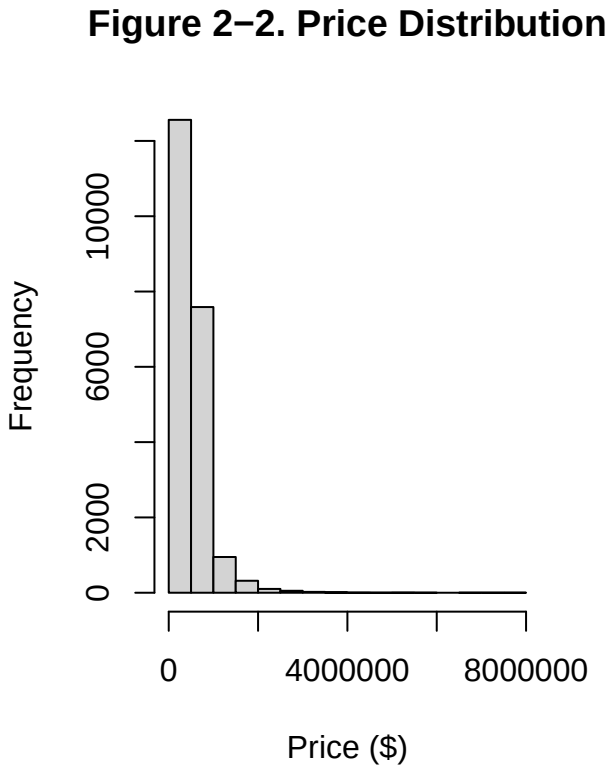
The following sections describe the response variable (price) and the predictor variables that we ultimately included in our models. Appendix C contains additional plots and discussion of predictor variables that were not included in the final models.

```
# price box plot and histogram
par(mfrow=c(1,2))
```

```
boxplot(kc_house_sales$price, main = "Figure 2-1. Price Box Plot")
hist(kc_house_sales$price, main = "Figure 2-2. Price Distribution", xlab="Price ($)")
```

II.IV.I Price






```
mean(kc_house_sales$price)
```

```
## [1] 540088.1
```

```
median(kc_house_sales$price)
```

```
## [1] 450000
```

We updated the price column to a numeric column since it could not be analyzed with the character type. The median sales price for the observations was $450,000, and the average was $540,088.10. Per Figure 2-1, it appears as though there are many outliers in the data on the higher end of the price range. Figure 2-2 confirms this assumption, showing that the distribution of sales prices has a much higher frequency at the lower end of the range.

```
# sqft_living box plot, histogram, scatter plot vs. price
par(mfrow=c(1,2))
boxplot(kc_house_sales$sqft_living, main = "Figure 2-3. Sqft_living Box Plot", xlab="Sqft_living")
hist(kc_house_sales$sqft_living, main = "Figure 2-4. Distribution of Sqft_living", xlab = "Sqft_living")
```

II.IV.II Square Footage - Living

Figure 2−3. Sqft_living Box Plot

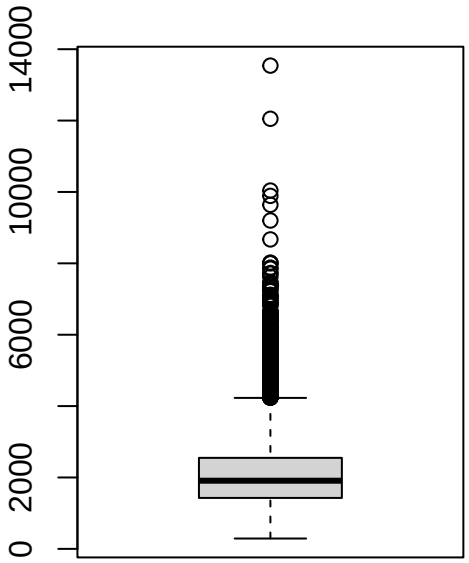
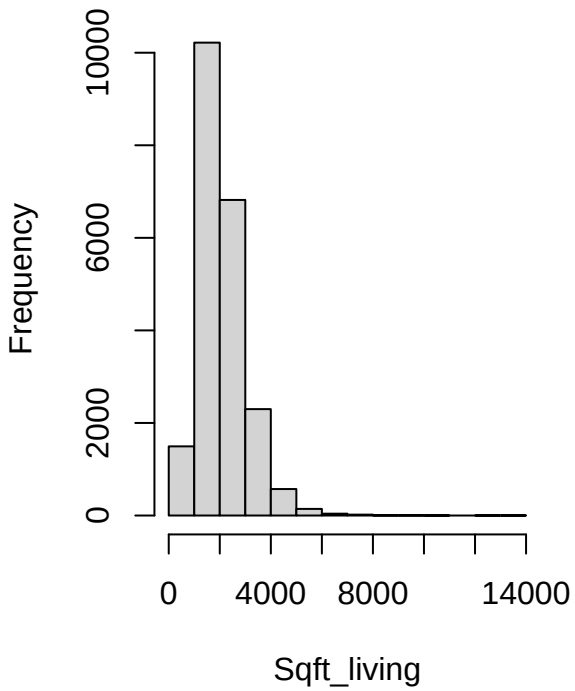
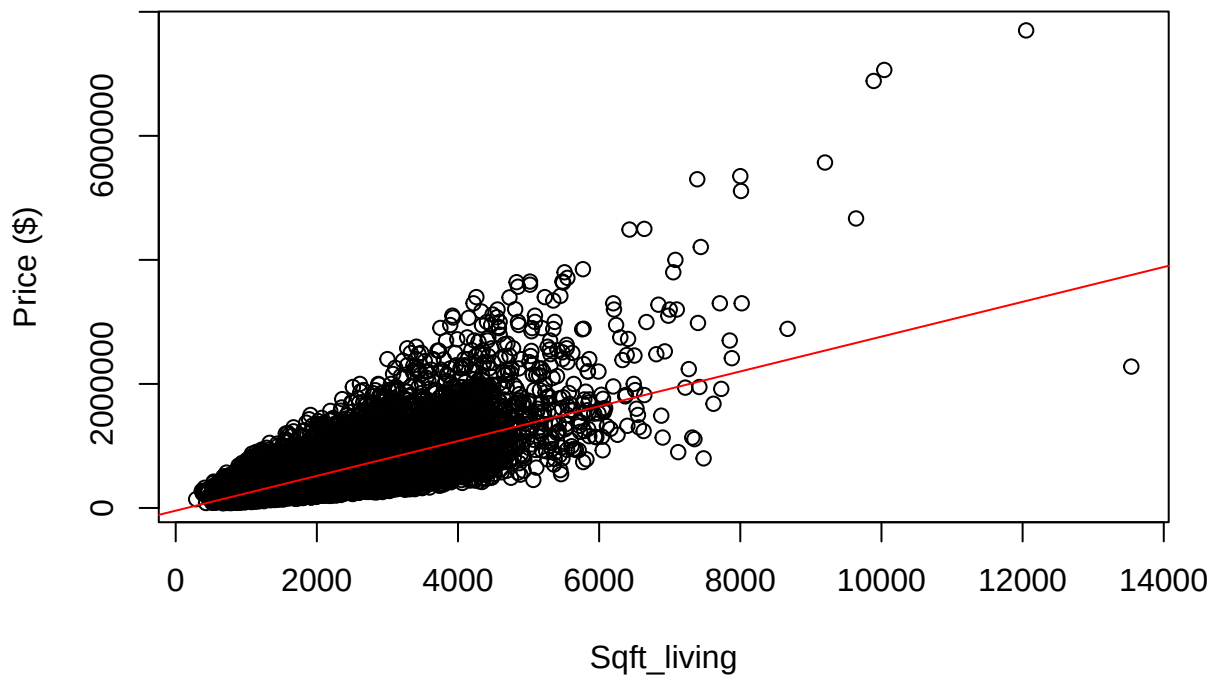


Figure 2−4. Distribution of Sqft_liv




```
par(mfrow=c(1,1))
plot(kc_house_sales$sqft_living, kc_house_sales$price,  main = "Figure 2-5. Sqft_living Scatter Plot",
     xlab="Sqft_living", ylab="Price ($)");
abline(lm(kc_house_sales$price ~ kc_house_sales$sqft_living), col = "red")
```

Figure 2−5. Sqft_living Scatter Plot



Per Figure 2-3 and 2-4, the median square footage of living space appears to be around 2000 sqft, with outliers present at the higher end of the range.

Per Figure 2-5, there appears to be a linear relationship between square feet of living space and price, with price increasing as the square footage increases.

```r
# view box plot, histogram, scatter plot vs. price
par(mfrow=c(1,2))
boxplot(kc_house_sales$view, main = "Figure 2-5. View Box Plot")
hist(kc_house_sales$view, main = "Figure 2-6. View Distribution", xlab = "View")
```

II.IV.III View

Figure 2−5. View Box Plot

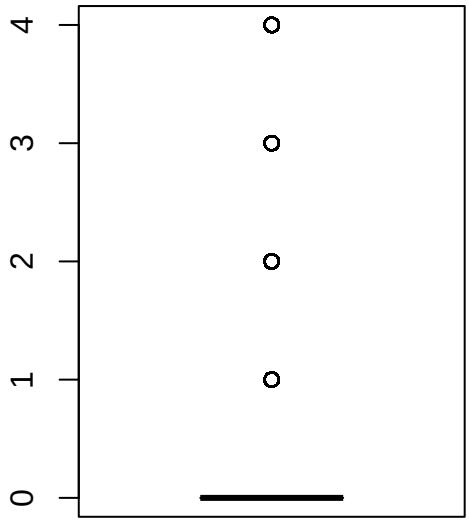
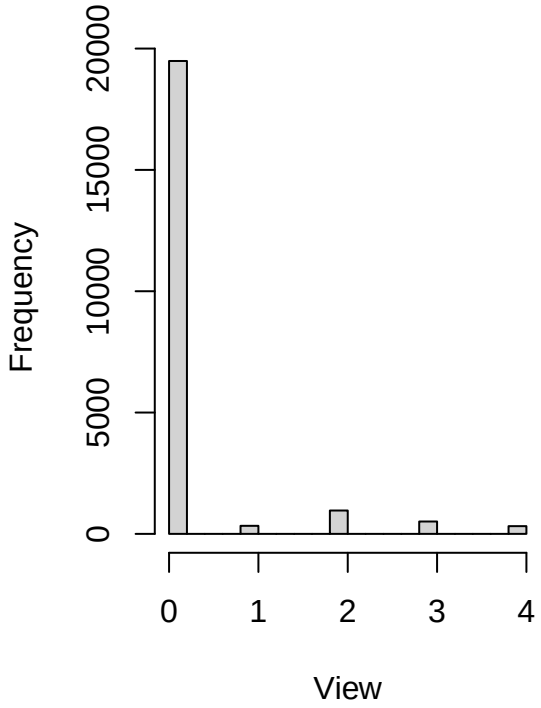
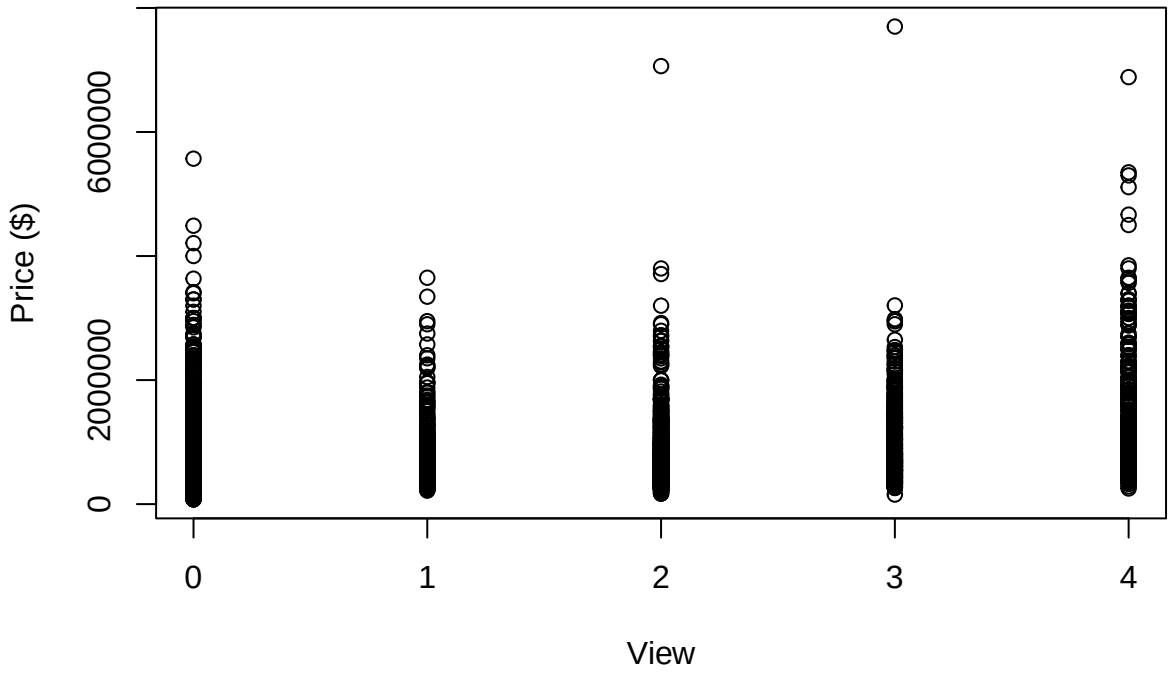


Figure 2−6. View Distribution




```r
par(mfrow=c(1,1))
plot(kc_house_sales$view, kc_house_sales$price,  main = "Figure 2-7. View Scatter Plot",
     xlab = "View", ylab = "Price ($)")
```

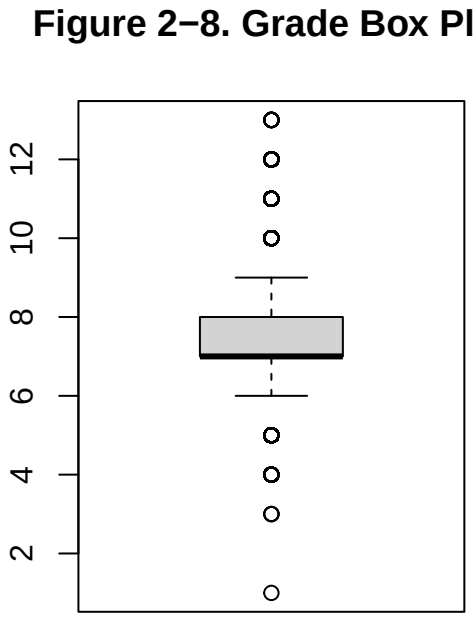
Figure 2−7. View Scatter Plot

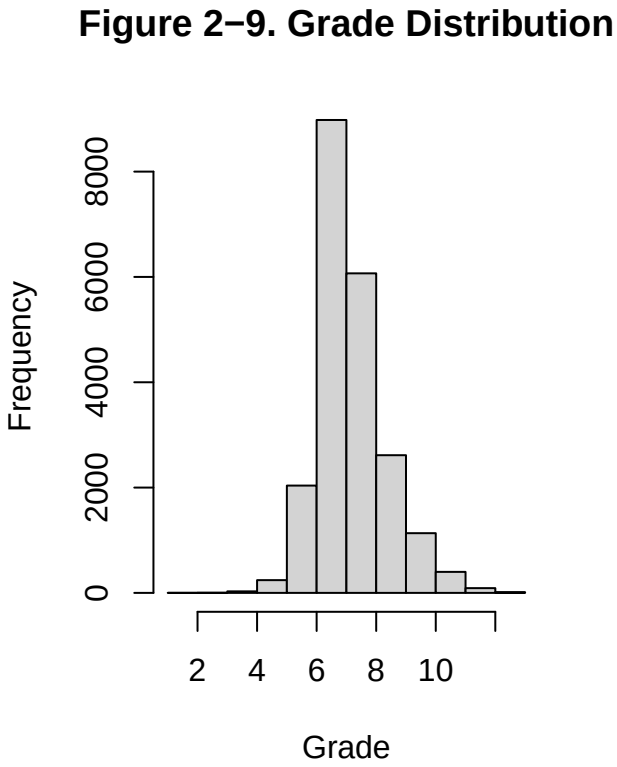


Looking at Figure 2-5 and Figure 2-6, most houses do not have a view; according to the box plot, any house with a view appears to be an outlier. Per Figure 2-7, the price does not seem to increase or decrease much depending on the house's view.

```
# grade box plot, histogram, scatter plot vs. price
par(mfrow=c(1,2))
boxplot(kc_house_sales$grade, main = "Figure 2-8. Grade Box Plot")
hist(kc_house_sales$grade, main = "Figure 2-9. Grade Distribution", xlab = "Grade")
```

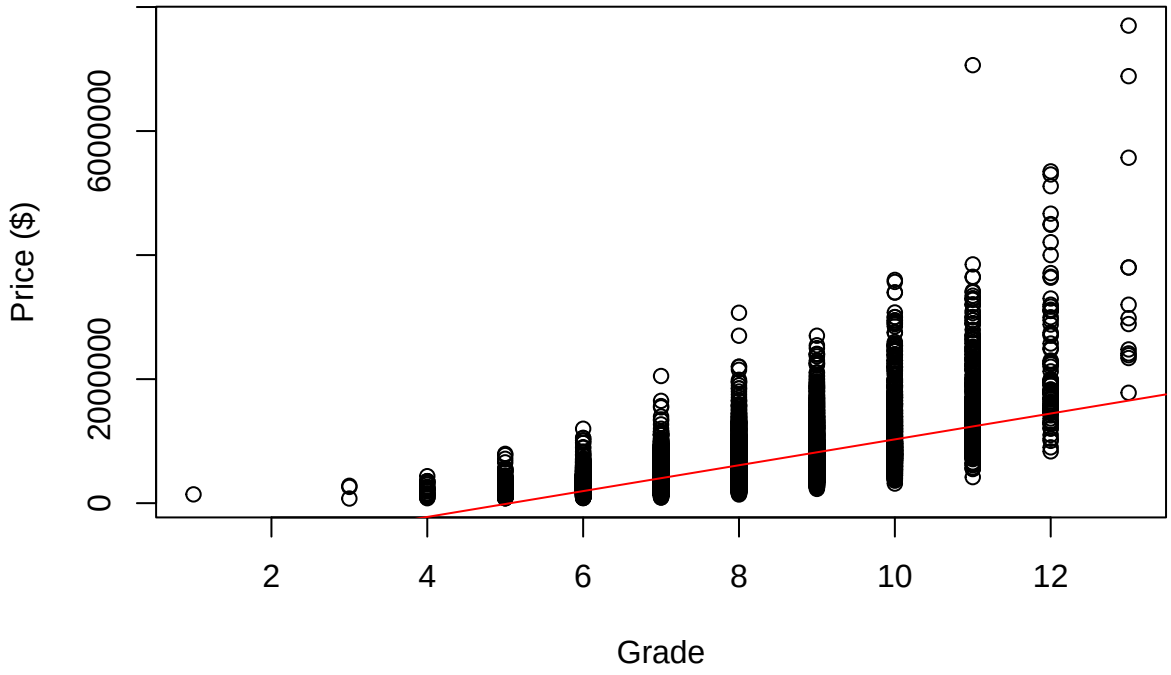
II.IV.IV Grade






```
par(mfrow=c(1,1))
plot(kc_house_sales$grade, kc_house_sales$price,  main = "Figure 2-10. Grade Scatter Plot",
     xlab = "Grade", ylab = "Price ($)");
abline(lm(kc_house_sales$price ~ kc_house_sales$grade), col = "red")
```

Figure 2−10. Grade Scatter Plot

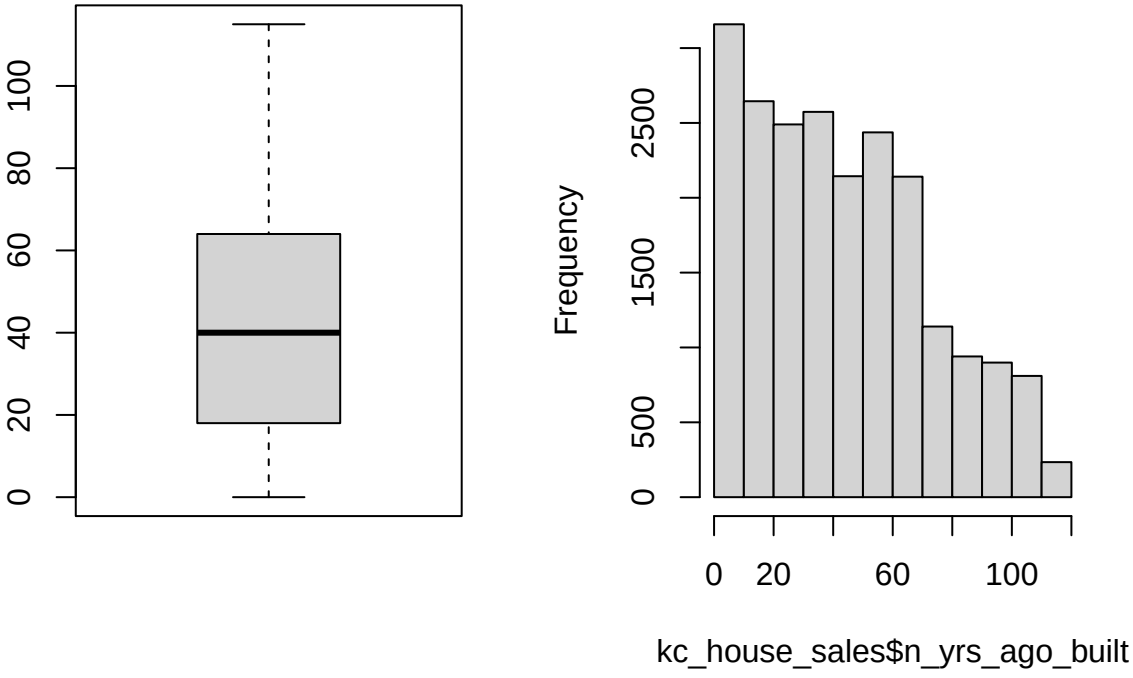


According to the box plot and histogram in Figure 2-8 and 2-9, the median grade appears to be 7, which is an "average" quality, and outliers are any grade outside the range of 6 to 9. The scatter plot in Figure 2-10 indicates that grade appears to have a linear relationship with prices, with the price increasing as the grade of the house increases.

```
# grade box plot, histogram, scatter plot vs. price
par(mfrow=c(1,2))
boxplot(kc_house_sales$n_yrs_ago_built, main = "Figure 2-11. N_yrs_ago_built Box Plot")
hist(kc_house_sales$n_yrs_ago_built, main = "Figure 2-12. N_yrs_ago_built Distribution")
```

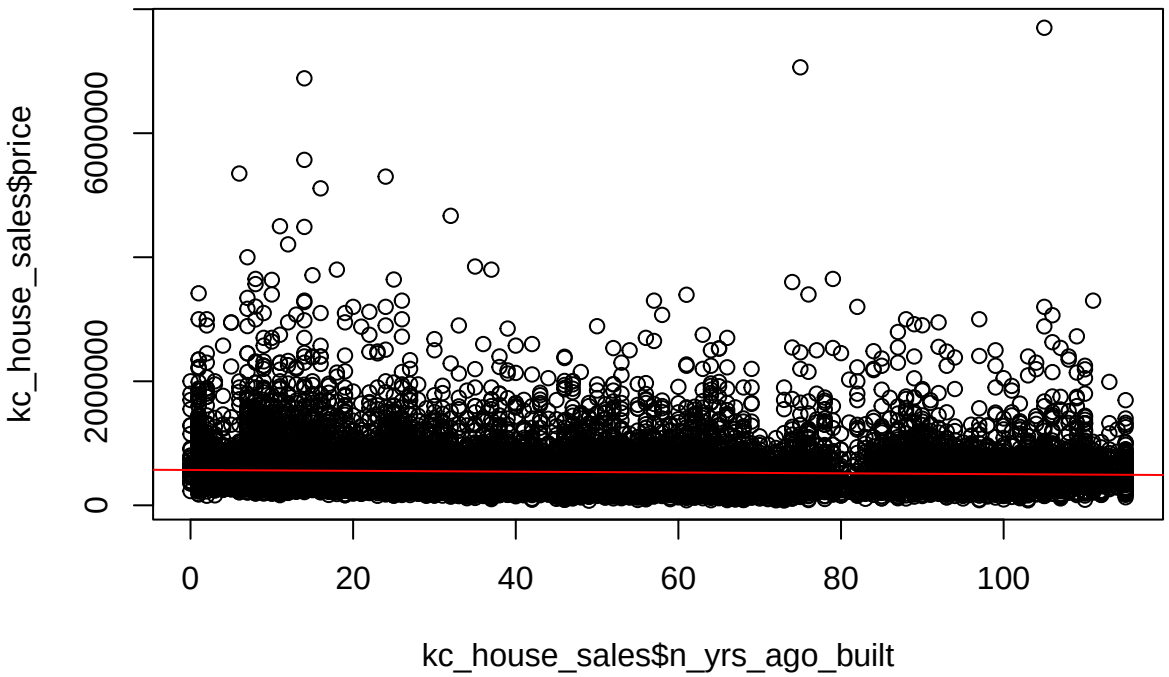
II.IV.V Years Ago Built

Figure 2−11. N_yrs_ago_built Box Figure 2−12. N_yrs_ago_built Distrib



```
par(mfrow=c(1,1))
plot(kc_house_sales$n_yrs_ago_built, kc_house_sales$price,
     main = "Figure 2-13. N_yrs_ago_built Scatter Plot");
abline(lm(kc_house_sales$price ~ kc_house_sales$n_yrs_ago_built), col="red")
```

Figure 2−13. N_yrs_ago_built Scatter Plot

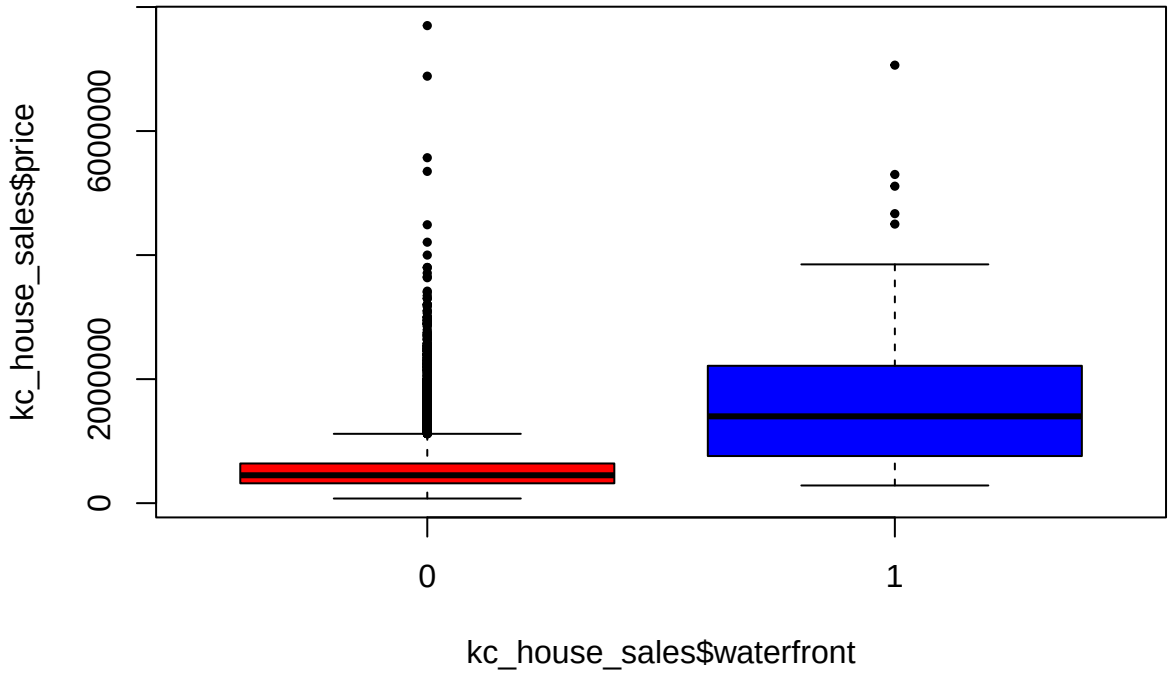


As discussed above, we create a new variable, n_yrs_ago_built by subtracting the year the house was built from 2015, the maximum sales date from the data set. The box plot in Figure 2-11 shows that the median age of houses sold was 40 years, with no outliers in the data set. The distribution of houses sold appears to skew towards newer houses per Figure 2-12. Looking at the scatter plot, the number of years ago built does not appear to have a strong relationship to the price of the house. It would be interesting to look at this variable in combination with other factors, such as zip code (e.g. was one desirable neighborhood/zip code built at a certain time and does that affect the price?).

```
boxplot(kc_house_sales$price ~ kc_house_sales$waterfront , main = "Figure 2-14 Waterfront Box Plot"
        , col = c("red", "blue"), pch=19, cex=0.5)
```

II.IV.VI Waterfront

Figure 2−14 Waterfront Box Plot



Waterfront is a categorical variable, with houses either on the waterfront (1) or not (0). From the box plot in Figure 2-14, it appears as though having a waterfront view may lead to higher prices; the median price of houses with a waterfront view is much higher.

II.IV.VII Zip Code As discussed above, we chose to use zip code as a categorical variable since it is used as an identifier rather than a number whose value has any meaning. There are 70 unique zip codes in the data set. The box plot in Figure 2-15 shows a wide range in the median value of house prices in certain zip codes, with several outliers for most of the zip codes. At first glance, it seems as though the oft-repeated "location, location, location" phrase may be true for this data set.

III. Model Development Process (15 points)

```
# Import house price data
kc_house_sales <- read_csv('KC_House_Sales.csv') %>%
  mutate(price = as.numeric(str_remove_all(price, '[$,]'))) %>%
  dplyr::select(-id, -lat, -long)

yr_max <- year(max(kc_house_sales$date))

# Transform variables where needed (NOTE: be sure to discuss in data quality
# section why we dropped sq ft of basement and year renovated)

kc_house_sales_transf <- kc_house_sales %>%
  mutate(n_yrs_ago_built = yr_max - yr_built,
         zipcode = as.character(zipcode)) %>%
  dplyr::select(-date, -yr_built, -yr_renovated, -sqft_basement)
```

```
# Split into train (70%) and test sets
set.seed(1023)
n <- nrow(kc_house_sales_transf)
train_i <- sample(n, size = 0.7*n)

train_df <- kc_house_sales_transf[train_i,]
test_df <- kc_house_sales_transf[-train_i,]
```

III.I Estimation of Full Model

Prior to conducting any variable selection or variable transformations, we estimate the linear regression model using all predictors. Note that the zipcode variable is a categorical variable with 70 unique zipcodes. In our model, we consider this to be one variable with 70 degrees of freedom, i.e., there will be a separate dummy variable for 69 of the 70 zipcodes.

The results from the full model estimation are shown below. Every variable is significant at the 95% level of significance, except for square feet of living space for the nearest 15 neighbors. Furthermore, the adjusted R-squared is .803, meaning that the predictors explain about 80.3% of the variation in the response variable.

The fact that almost every variable is significant suggests that if we were to do backward or forward stepwise selection process based on p-values alone, we would most likely end up with nearly all of the variables still in the model. Thus, it makes more sense to consider a performance metric like AIC or adjusted R-squared and include only the variables that provide material lift to those metrics, rather than all significant variables.

```
# Estimate the full model using training data (zipcode dummy vars created automatically)
full_model <- lm(price ~ ., data = train_df)

# Display results
tidy(full_model) %>%
  dplyr::select(Term = term, Coefficient = estimate, std.error, p.value) %>%
  mutate(Coefficient = round(Coefficient, 3),
         std.error = round(std.error, 4),
         p.value = round(p.value, 4)) %>%
  filter(!str_detect(Term, 'zipcode')) %>%
  kable(caption = "Full OLS Model - All Predictors (zip code not shown, but predictor is included in model)",
        booktabs = T,) %>%
  kable_styling(full_width = F,
                latex_options = "HOLD_position")
```

Table 2: Full OLS Model - All Predictors (zip code not shown, but predictor is included in model)

Term	Coefficient	std.error	p.value
(Intercept)	-565542.113	19167.9015	0.0000
bedrooms	-25073.411	1794.5200	0.0000
bathrooms	24125.154	3073.7739	0.0000
sqft_living	130.190	4.2249	0.0000
sqft_lot	0.259	0.0484	0.0000
floors	-39034.995	3778.3440	0.0000
waterfront	623462.792	17623.6730	0.0000
view	58020.758	2112.6314	0.0000
condition	23921.933	2253.9728	0.0000
grade	57853.582	2161.6682	0.0000
sqft_above	70.782	4.3343	0.0000
sqft_living15	15.903	3.4653	0.0000
sqft_lot15	-0.079	0.0718	0.2688
n_yrs_ago_built	837.218	73.2400	0.0000

III.II Forward Stepwise Selection (AIC)

Using the AIC as the selection criteria for forward stepwise selection, we can find that most variables drive a reduction in the AIC and adjusted R-squared, as shown in the table below. However, only a small handful of variables provide a material amount of additional lift, as the others barely move the needle on AIC and adjusted R-squared.

The two variables that absolutely must be in the model are sq ft of living space and zip code, as the model with just those two variables gets us most of the way there to the levels of AIC and adjusted R-squared seen when all variables are included. This makes sense from an intuitive standpoint, as location is well-known to be one of the most important things people consider when buying a home and rich people typically live in large homes.

We can also consider including waterfront, view, grade, and age of the home, as they provide a mild amount of additional lift. But after these variables, the remaining variables, despite being significant, provide minimal impact to the adjusted R-squared or AIC.

```
# Use AIC forward stepwise selection
ols_step_forward_aic(full_model)
```

```
##
##                                 Selection Summary
## ------------------------------------------------------------------------------------------------
## Variable              AIC              Sum Sq                  RSS              R-Sq      Adj. R-Sq
## ------------------------------------------------------------------------------------------------
## sqft_living        419922.764    975433357849843.750    1003472712546089.250    0.49292      0.49288
## zipcode            410522.699    1444704686460297.500    534201383935632.438    0.73005      0.72880
## waterfront         408449.379    1513179877155945.250    465726193239988.188    0.76465      0.76354
## view               407530.142    1540692783551635.750    438213286844293.438    0.77856      0.77750
## grade              406687.693    1564482248578371.000    414423821817561.125    0.79058      0.78956
## n_yrs_ago_built    406327.220    1574293338460845.250    404612731935085.312    0.79554      0.79453
## bedrooms           406145.714    1579171423862179.000    399734646533752.125    0.79800      0.79700
## sqft_above         406000.600    1583039578867751.750    395866491528179.062    0.79996      0.79895
## condition          405884.517    1586117344860477.000    392788725535452.625    0.80151      0.80050
## floors             405795.244    1588479879969471.000    390426190426462.750    0.80271      0.80168
## bathrooms          405737.677    1590014068757112.000    388892001638817.312    0.80348      0.80245
## sqft_lot           405703.901    1590932628352137.500    387973442043793.625    0.80395      0.80290
## sqft_living15      405685.191    1591463338256051.000    387442732139879.000    0.80421      0.80316
## ------------------------------------------------------------------------------------------------
```

III.III Best Subsets Variable Selection

This part will only consider sqft_living, zipcode, waterfront, view, grade, and age as potential variables, following the discussion in the previous section. To identify the best combination of these six variables to include in the final model, we

consider all possible subsets of the variables, of which there are 63 total subsets. In the table below, we present the best subset for each value of p, where p is the number of variables in the model.

To identify the best model, we can use the "elbow method", which is a useful rule of thumb that suggests choosing the value for p for which subsequent values provide minimal additional lift. Using the elbow method, it appears that "model 2", which includes only sqft_living and zipcode, is the best model. This is our first candidate for the final regression model.

The second candidate model is the one that includes all six variables considered in this section. The intuition behind doing so is because not only do the additional four variables provide an additional 6.8% boost to the R-squared, but the second candidate is also more intuitive and easier to explain to stakeholders, who may find it undesirable to include just two variables in a house pricing model.

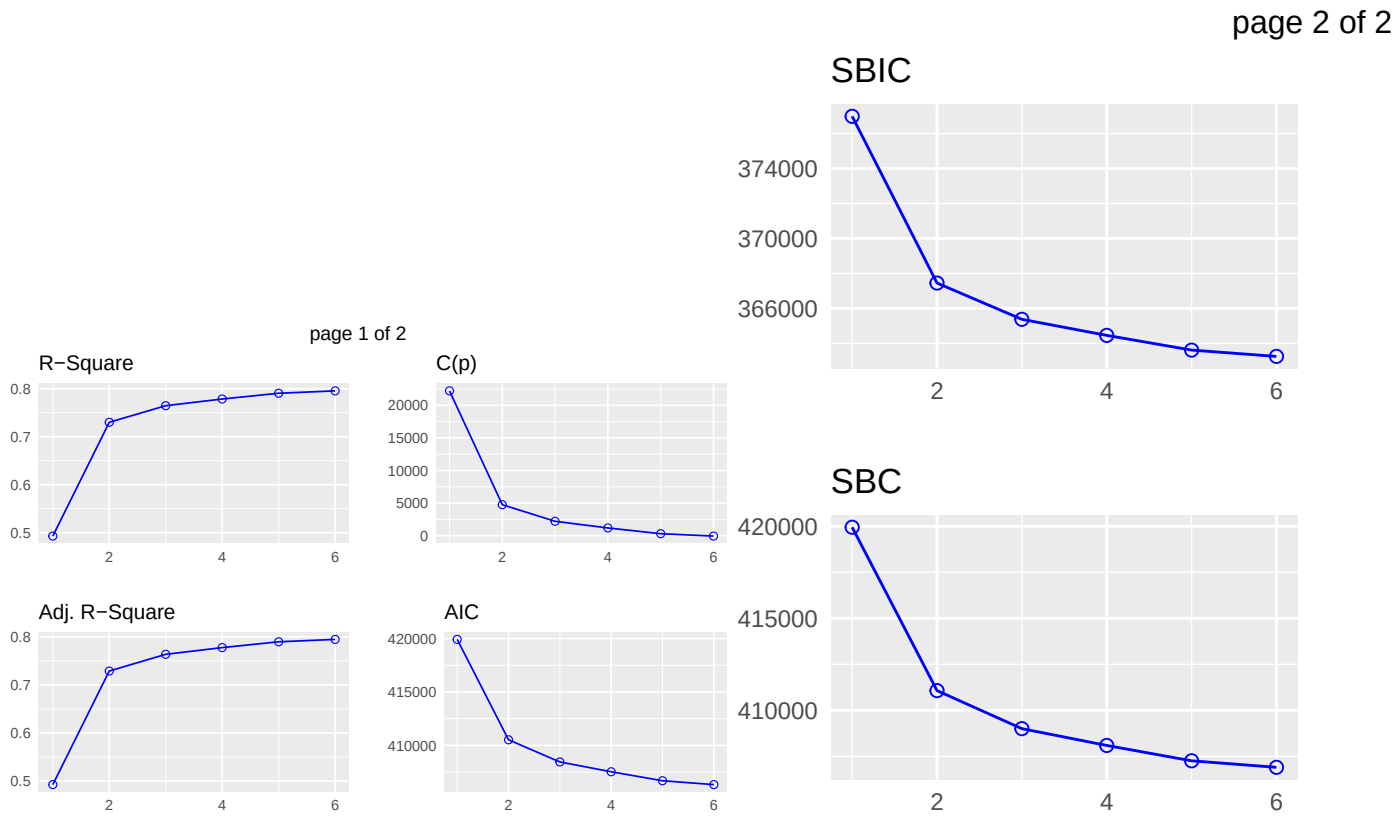
To summarize, the two final candidate models for the linear regression model are as follows:

Candidate 1: Price = sqft_living + zipcode

Candidate 2: Price = sqft_living + zipcode + waterfront + view + grade + n_yrs_ago_built

```
# Construct the reduced model
reduced_model <- lm(price ~ sqft_living + waterfront + view + grade + n_yrs_ago_built +
                      zipcode, data = train_df)

reduced_forward <- ols_step_best_subset(reduced_model)

reduced_forward
```

```
plot(reduced_forward)
```



IV. Model Performance Testing (15 points)

Use the test data set to assess the model performances. Here, build the best multiple linear models by using the stepwise both ways selection method. Compare the performance of the best two linear models. Make sure that model assumption(s) are checked for the final linear model. Apply remedy measures (transformation, etc.) that helps satisfy the assumptions. In particular you must deeply investigate unequal variances and multicollinearity. If necessary, apply remedial methods (WLS, Ridge, Elastic Net, Lasso, etc.).

In this section, we examine the performance of the two best models found in the section above along multiple dimensions. First, we check the assumptions of the linear regression model and apply remediation measures as needed, such as the addition of polynomial terms, weighted least squares, robust regression, and variable transformation.

IV.I Performance of Candidate OLS Models

Two OLS models were identified as candidate models in the previous section, referred to as Candidate 1 and Candidate 2. To determine which of the two models to examine and refine further, we must compare the performance of the two models on the test set using two metrics: adjusted R-squared and root mean squared error (RMSE). The results are shown below.

Clearly, Candidate 2 is the better model, as it performs much better than candidate 1 on the test set with regards to both the adjusted R-squared and RMSE. Thus, for the remainder of this section, we choose to refine the second candidate model, which has the following six variables:

- Square feet of living space
- Zip Code
- Waterfront
- Grade
- View
- Age of Home

```
# Estimate best models
best_ols_cand1 <- lm(price ~ sqft_living + zipcode, data = train_df)
best_ols_cand2 <- lm(price ~ sqft_living + waterfront + view + grade +
                       n_yrs_ago_built + zipcode, data = train_df)

# Calculate RMSE for each model

# Get predictions on test set
cand1_yhat <- predict(best_ols_cand1, newdata = test_df)
cand2_yhat <- predict(best_ols_cand2, newdata = test_df)

# Get observed values for response
y_actual <- test_df %>% pull(price)

# Calculate RMSE
rmse_ols1 <- sqrt(sum((y_actual - cand1_yhat)^2) / nrow(test_df))
rmse_ols2 <- sqrt(sum((y_actual - cand2_yhat)^2) / nrow(test_df))

# Function to calculate adjusted R-squared
calc_rsq_adj <- function(n, p, sse, sst) {
  r_sq <- 1 - sse / sst
  adj_r_sq <- 1 - (1 - r_sq) * (n - 1) / (n - p - 1)
  return(adj_r_sq)
}

# SST and SSE
sst_test <- sum((y_actual - mean(y_actual))^2)
sse_cand1_test <- sum((y_actual - cand1_yhat)^2)
sse_cand2_test <- sum((y_actual - cand2_yhat)^2)

# Calculate adjusted R-squared for each model
n <- nrow(test_df)
adj_rsq_cand1 <- calc_rsq_adj(n, best_ols_cand1$rank - 1, sse_cand1_test, sst_test)
adj_rsq_cand2 <- calc_rsq_adj(n, best_ols_cand2$rank - 1, sse_cand2_test, sst_test)
```

```r
# Put results into a table
results_df <- tibble(Model = c('OLS Candidate 1', 'OLS Candidate 2'),
                         RMSE = c(rmse_ols1, rmse_ols2),
                         `Adjusted R-squared` = c(adj_rsq_cand1, adj_rsq_cand2))

kable(results_df,
      format.args = list(big.mark = ","),
      booktabs = T,
      caption = ' Root Mean Squared Error (RMSE) and Adjusted R-squared on the Test Set for the Two Candidate
  kable_styling(full_width = F,
                latex_options = "HOLD_position")
```

Table 3: Root Mean Squared Error (RMSE) and Adjusted R-squared on the Test Set for the Two Candidate Linear Regression Models

Model	RMSE	Adjusted R-squared
OLS Candidate 1	198,347.0	0.7238776
OLS Candidate 2	168,695.3	0.8001395

IV.II Estimation of Candidate Model 2

The results from the estimation of candidate model number 2 are shown below. Signs on each of the coefficients follow intuition, as a unit increase in each of the variables (except for zip code) are intuitively expected to increase price. This is consistent with positive signs on each of those variables.

```r
# Show summary of candidate 2
tidy(best_ols_cand2) %>%
  dplyr::select(Term = term, Coefficient = estimate, std.error, p.value) %>%
  mutate(Coefficient = round(Coefficient, 3),
         std.error = round(std.error, 4),
         p.value = round(p.value, 4)) %>%
  filter(!str_detect(Term, 'zipcode')) %>%
  kable(caption = "Candidate OLS Model 2 (zip code not shown, but predictor is included in model)",
        booktabs = T) %>%
  kable_styling(full_width = F,
                latex_options = "HOLD_position")
```

Table 4: Candidate OLS Model 2 (zip code not shown, but predictor is included in model)

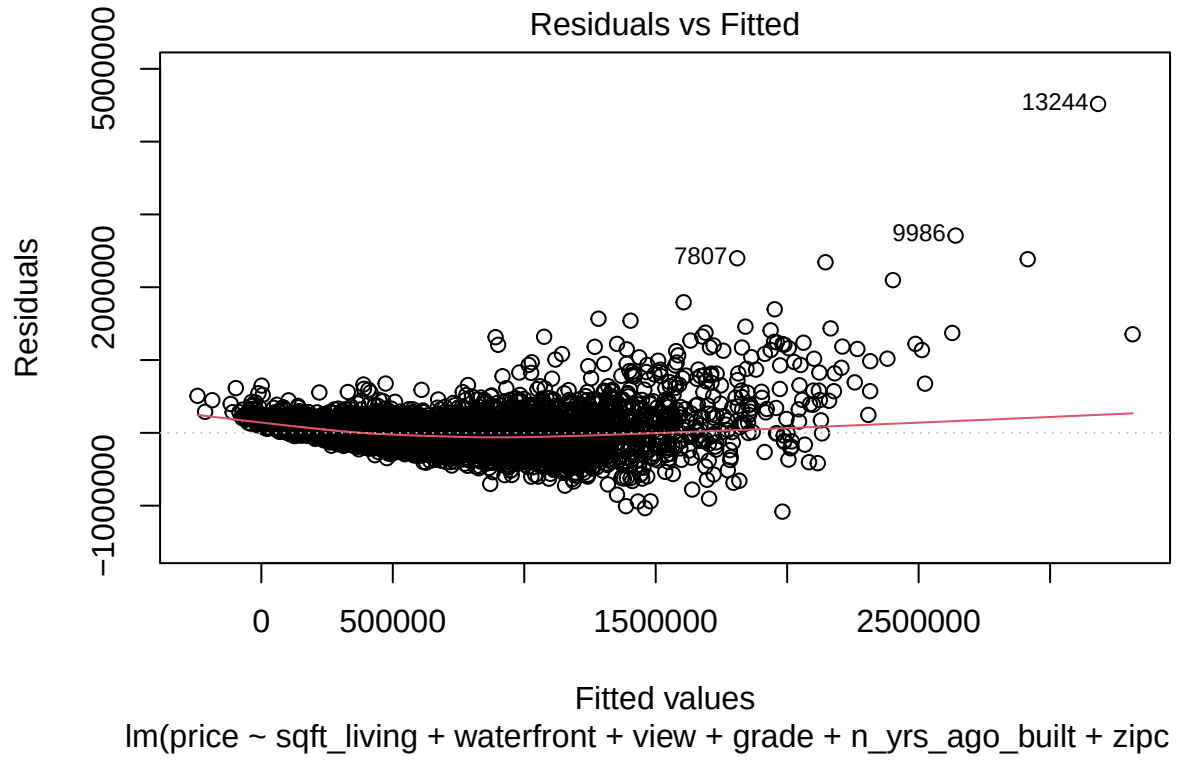
Term	Coefficient	std.error	p.value
(Intercept)	-616850.80	16817.3122	0
sqft_living	173.26	2.3703	0
waterfront	628281.75	17952.8962	0
view	58019.26	2092.7790	0
grade	71571.02	2045.2100	0
n_yrs_ago_built	1157.30	60.5733	0

IV.III Model Diagnostics

In this section, the model assumptions to check are: Linearity, Homoskedasticity, Error Normality, and no multicollinearity. From here on, the term "model" refers to candidate 2 above.

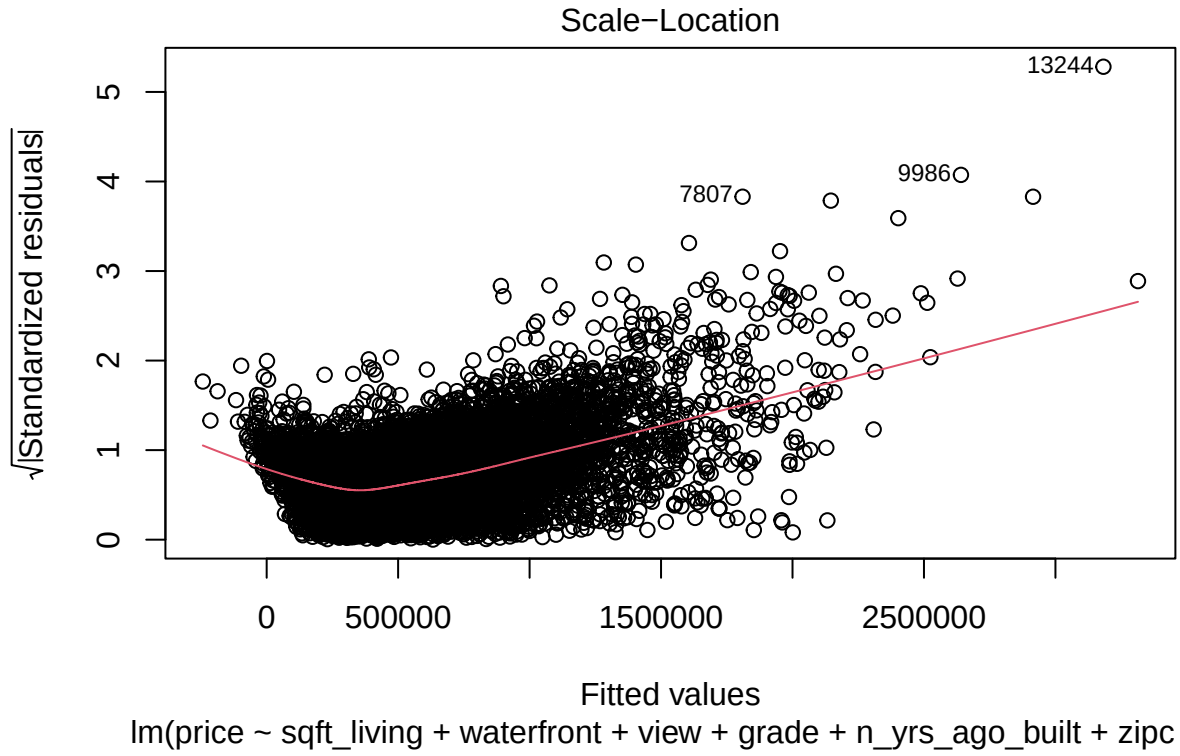
IV.III.I Linearity Assumption We can test the linearity assumption by plotting the residuals against the fitted values. From the plot below, it seems that the linearity assumption is valid on most of the data, except for the most expensive homes, where the model consistently underestimates price. This suggests that a polynomial term may be useful to help better predict the more expensive properties.

```r
# Plot residuals against fitted values
plot(best_ols_cand2, which = 1)
```



IV.III.II Homoskedasticity Assumption To test the equal variances assumption, we can examine a scale-location plot. From the plot below, it's very clear that this assumption is violated. We can confirm this with the Breusch-Pagan test, which has a null hypothesis of equal variances and an alternative hypothesis of unequal variances. The result of this test is also shown below. As the p-value is nearly zero, we can very confidently reject the null of equal variances.

```r
# Plot scale-location
plot(best_ols_cand2, which = 3)
```



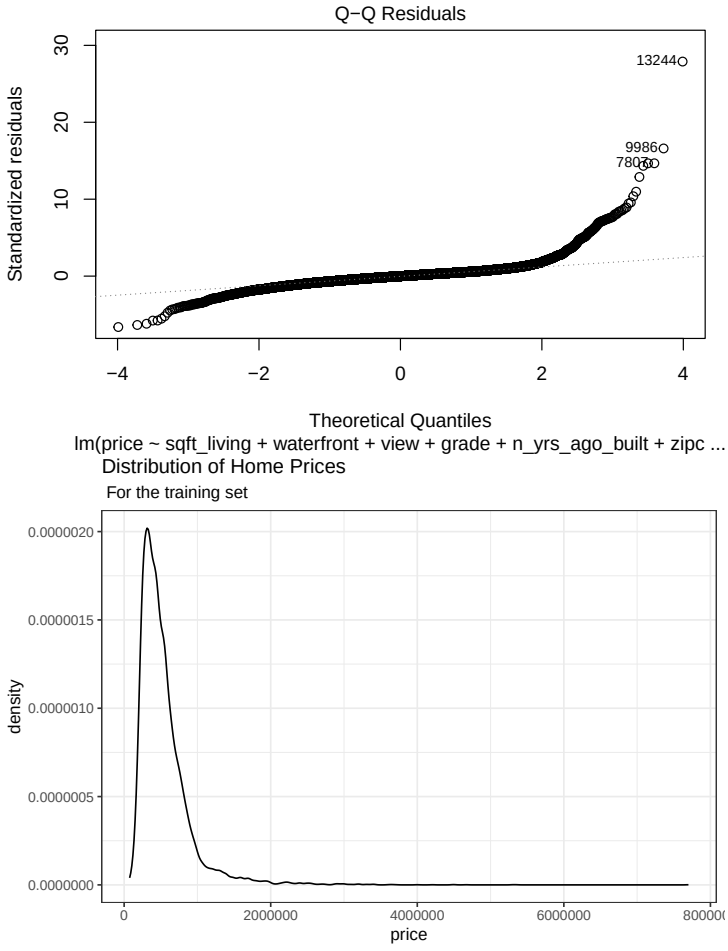
```r
# Breusch-Pagan test
bptest(best_ols_cand2, studentize = FALSE)
```

```
##
##  Breusch-Pagan test
##
## data:  best_ols_cand2
## BP = 58590, df = 74, p-value < 0.00000000000000022
```

IV.III.III Error Normality Assumption Next, we check the error normality assumption. This can be done by plotting the standardized residuals against the theoretical residuals, as shown below. Interestingly, the residuals follow normality for most of the range, except there are strong departures from normality in the tails. This model has very thick tails, particularly the right tail. This indicates that the model is a good fit for a wide range of properties; however, for a decent number of properties the model is a poor fit.

This is likely due to the fact that the distribution of house prices has a long right tail, as most houses have a price in the range of two-hundred thousand to one million dollars, but a small subset of houses have prices spread across the wide range of one million to eight million dollars. We are probably having trouble fitting these most expensive homes, and one way to address this issue is with robust regression, which places less weight on influential outliers.

```r
# Generate Q-Q plot
plot(best_ols_cand2, which = 2)

# Show distribution of home prices
ggplot(train_df) +
  geom_density(aes(x = price)) +
  labs(title = "Distribution of Home Prices",
       subtitle = " For the training set") +
  theme_bw()
```



IV.III.IV Multicollinearity Lastly, we check for multicollinearity in the candidate model. Variance inflation factors (VIF) are shown below for each of the predictors. Note that because the zip code term has more than one degree of freedom, the generalized VIF (GVIF) is calculated. None of the variables show signs of multicollinearity, as the VIFs are all below 5. Thus, we don't need to consider methods to remediate multicollinearity, such as Ridge and Lasso regressions.

```
# Check VIF
car::vif(best_ols_cand2)
```

```
##                     GVIF Df GVIF^(1/(2*Df))
## sqft_living     2.632796  1        1.622589
## waterfront      1.226738  1        1.107582
## view            1.409191  1        1.187094
## grade           3.237639  1        1.799344
## n_yrs_ago_built 1.787531  1        1.336986
## zipcode         2.234802 69        1.005844
```

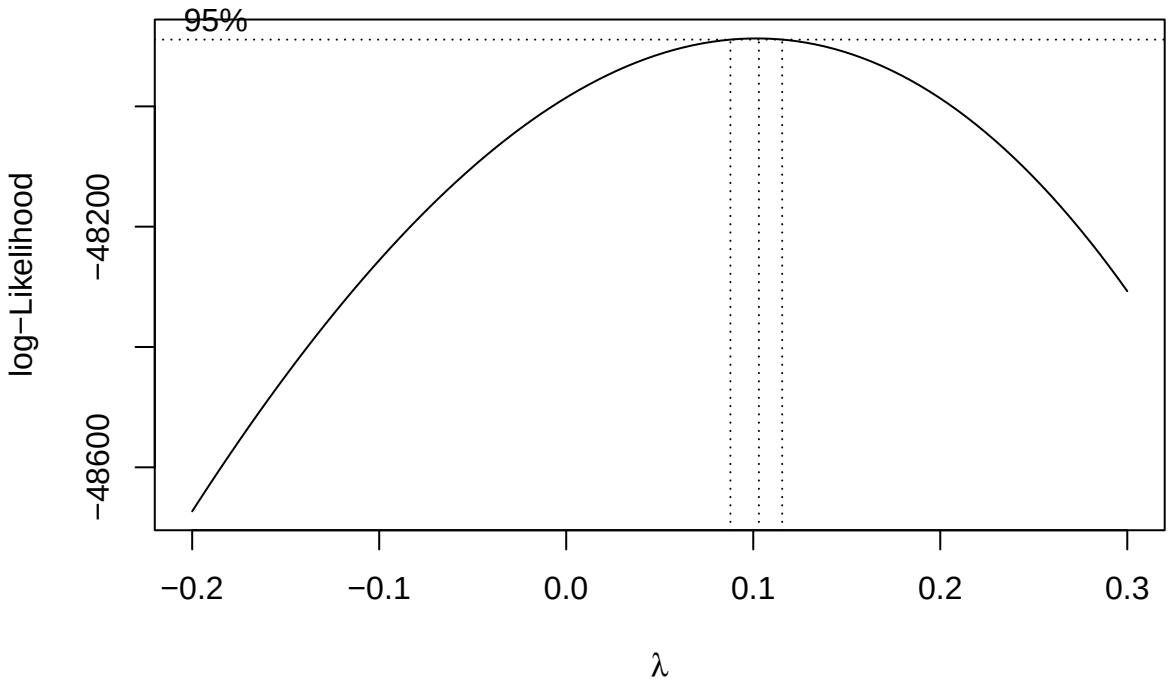
IV.IV Remediation

To address the issues described above, we can consider several remedial measures, including the inclusion of polynomial terms, transformation of the response variables, weighted least squares, and robust regression.

IV.IV.I Variable Transformation We can consider the possibility of a transformation of the response variable, price. To find the optimal transformation, we consider the Box-Cox method, which seeks to find the power transformation that maximizes log-likelihood. From the plot below, it appears that a transformation of 0.1 is optimal. Note that none of zero, one-half, and one are in the 95% confidence interval.

However, as raising the response to the power of 0.1 sounds strange and can be difficult to communicate to stakheholders, we will not pursue this transformation, as there may be better ways to remediate the issues with the model.

```
# Find optimal transformation
boxcox(best_ols_cand2, lambda = seq(-.2,.3,.01))
```



IV.IV.II Polynomial Terms We next consider adding a second order term to the model for the continuous variables, namely square feet of living space. Note that we must center this variable first to prevent multicollinearity. The estimated model, shown below, indicates that the square term is highly significant.

Furthermore, inclusion of the second order term provides a boost of several percentage points to the adjusted R-squared, which is now 82.6%.

```
# Add square terms, but center them first
train_poly_df <- train_df %>%
```

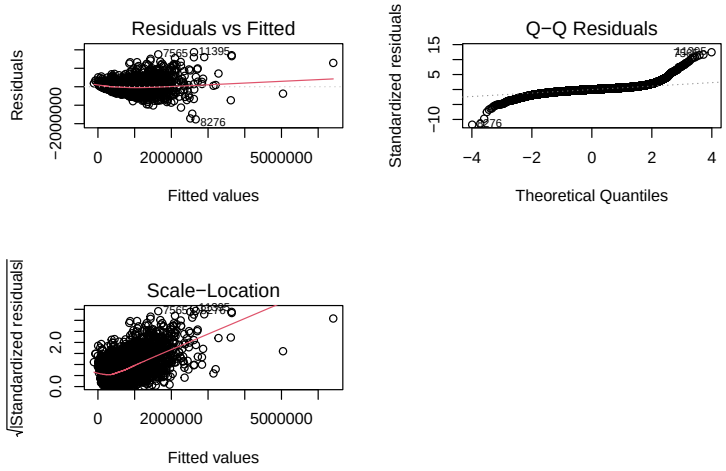
```
  dplyr::select(price, sqft_living, waterfront, view, grade, n_yrs_ago_built, zipcode) %>%
  mutate(across(.cols = c(sqft_living), ~.x - mean(.x)),
         across(.cols = c(sqft_living), ~ .x^2, .names = "{.col}_2")) %>%
  dplyr::select(price:n_yrs_ago_built, sqft_living_2, zipcode)

# Train model with poly terms
ols_poly_model <- lm(price ~ ., data = train_poly_df)

# Display results
tidy(ols_poly_model) %>%
  dplyr::select(Term = term, Coefficient = estimate, std.error, p.value) %>%
  mutate(Coefficient = round(Coefficient, 3),
         std.error = round(std.error, 4),
         p.value = round(p.value, 4)) %>%
  slice(1:7) %>%
  kable(caption = "OLS Model with Second Order Term (zip code not shown, but predictor is included in model)",
        booktabs = T) %>%
  kable_styling(full_width = F,
                latex_options = "HOLD_position")
```

Table 5: OLS Model with Second Order Term (zip code not shown, but predictor is included in model)

Term	Coefficient	std.error	p.value
(Intercept)	-298870.665	17588.6693	0
sqft_living	119.029	2.4230	0
waterfront	591591.047	16559.2787	0
view	55113.879	1929.3696	0
grade	74757.448	1885.7224	0
n_yrs_ago_built	956.134	55.9553	0
sqft_living_2	0.039	0.0007	0

IV.IV.II.I Polynomial Model Diagnostics We next must test the model assumptions on the second order polynomial model, the results of which are shown below. The good news is that the plot showing residuals vs fitted values is looking much better, suggesting that the linearity assumption is somewhat more satisfied (the red line is mostly horizontal until the end, most likely due to a couple of outlying price observations). However, no progress has been made on the other two assumptions, as error variances are still very unequal and errors still have very thick tails. In fact, heteroskedasticity seems to be an even greater issue with this new model.

```
# Plot all three
par(mfrow = c(2,2))
plot(ols_poly_model, which = 1)
plot(ols_poly_model, which = 2)
plot(ols_poly_model, which = 3)
```



IV.IV.II.II Multicollinearity - Second Order Model There are **no issues** with multicollinearity in the second order model. From the table below, all the VIFs are well under 5. Thus, we do not need to consider Lasso or Ridge regressions.

```
# VIF
car::vif(ols_poly_model)
```

```
##                      GVIF Df GVIF^(1/(2*Df))
## sqft_living      3.239683  1        1.799912
## waterfront       1.228994  1        1.108600
## view             1.410387  1        1.187597
## grade            3.241100  1        1.800306
## n_yrs_ago_built  1.796212  1        1.340228
## sqft_living_2    1.531980  1        1.237732
## zipcode          2.293810 69        1.006034
```

IV.III.I Robust Regression To address potential outlier cases that have an large influence on the model, we can use robust regression, which puts lower weights on the influential points. This is expected to have a meaningful impact on the model's performance, since the model has been shown to struggle with fitting the more expensive properties. We use Huber weights for the robust regression.

The results from the robust regression are shown below. All variables are significant (to conserve space, zip code is not shown, but it is included in the model). Interestingly, the AIC increased over the polynomial somewhat from 403,855 to 404,905. This is not a meaningful change in AIC, but it does suggest that the robust regression doesn't provide that much of a lift to model performance, contrary to expectations.

Lastly, as expected the standard errors on each of the estimators decreased when compared with the initial second order model.

```
# Use Huber weights
robust_model <- rlm(price ~ ., data = train_poly_df, psi = psi.huber)

# Display results
tidy(robust_model) %>%
  dplyr::select(Term = term, Coefficient = estimate, std.error, statistic) %>%
  mutate(Coefficient = round(Coefficient, 3),
         std.error = round(std.error, 4)) %>%
  slice(1:7) %>%
  kable(caption = "Robust Regression (zip code not shown, but predictor is included in model)",
        booktabs = T) %>%
  kable_styling(full_width = F,
                latex_options = "HOLD_position")
```

Table 6: Robust Regression (zip code not shown, but predictor is included in model)

Term	Coefficient	std.error	statistic
(Intercept)	-164978.362	10479.3139	-15.74324
sqft_living	121.694	1.4436	84.29724
waterfront	539742.619	9866.0039	54.70732
view	45955.039	1149.5168	39.97770
grade	58493.781	1123.5118	52.06335
n_yrs_ago_built	767.509	33.3381	23.02198
sqft_living_2	0.029	0.0004	64.63631

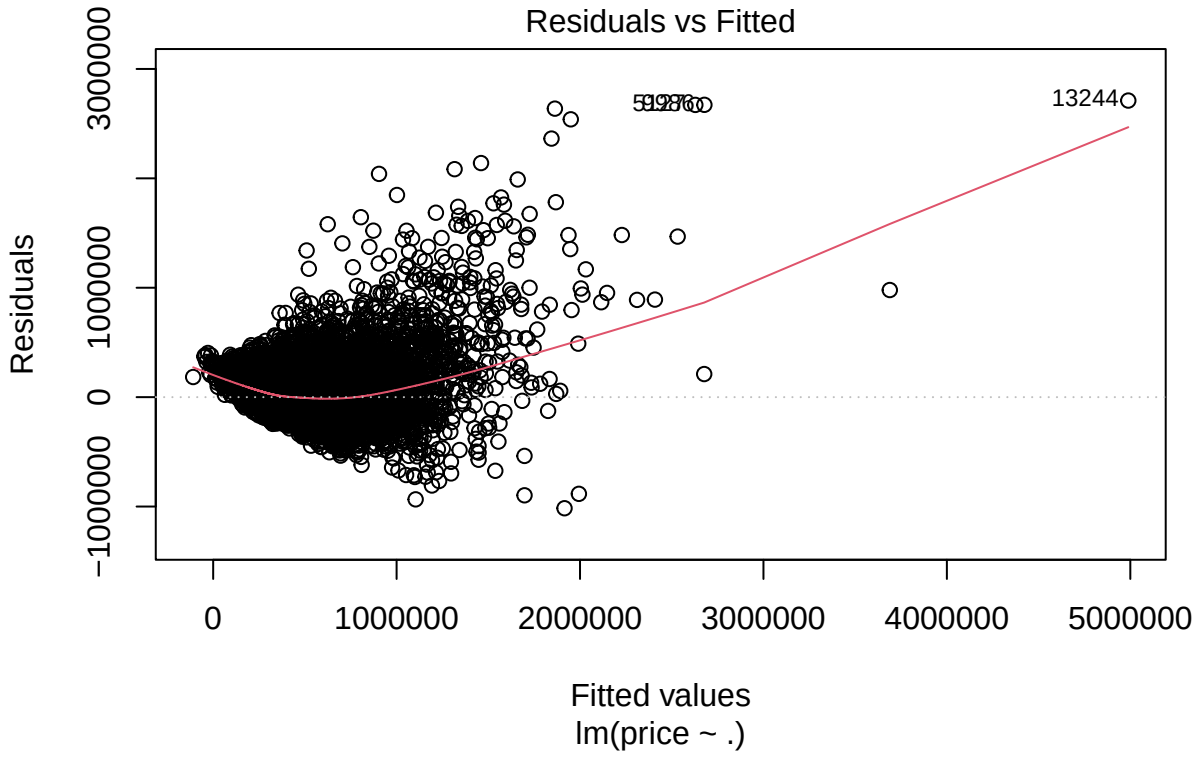
IV.III.II Weighted Least Squares Weighted Least Squares (WLS) may address issues with heteroskedasticity by putting less weight on observations with higher error variance. We apply WLS to the second order model by regressing absolute residuals on fitted values, and then we set the weights equal to the inverse of the square of the fitted absolute residuals. This has the effect of putting less weight on those observations that have higher error variance.

The results are shown below. Note that adjusted R-squared drops sharply to 61.2%, whereas before it was 82.6% (in the original second order model).

The WLS model also does not do much to address unequal variances, as shown by the plot of fitted values vs residuals and the Breusch-Pagan test. The plot continues to show a cone shape. Likewise, the p-value of the Breusch-Pagan test is nearly zero, which means we reject the null hypothesis of equal variances.

```
# Get absolute residuals and fitted values from the second order model
resids_poly <- abs(residuals(ols_poly_model))
yhat_poly <- predict(ols_poly_model)

# Regress absolute residuals on fitted values
wts_model <- lm(resids_poly ~ yhat_poly)
wts_wls <- 1 / (predict(wts_model)^2)

# Estimate WLS model
wls_model <- lm(price ~., data = train_poly_df, weights = wts_wls)
glance(wls_model)[1:9]
```

```
plot(wls_model, which = 1)
```




```
bptest(wls_model, studentize = F)
```

```
##
##  Breusch-Pagan test
##
## data:  wls_model
## BP = 906769420346, df = 75, p-value < 0.00000000000000022
```

IV.V Performance Evaluation Using Test Set

Four models have been considered in this section: the linear model with six predictors, a second order model, robust regression, and weighted least squares. To compare the performance, we apply the estimated models to the test set and calculate RMSE and adjusted R-squared for each model.

The results are shown below. OLS Candidate 1 refers to the model with only two predictors (sq ft of living space and zip code) whereas OLS Candidate 2 refers to the model with six predictors (additional variables include waterfront, view, age of home, and grade). The polynomial model is built off the second candidate OLS model. Likewise, the robust and WLS models are built off the polynomial model.

The two best performing models are the robust and polynomial models, as they each have the lowest RMSE and highest adjusted R-squared values. Since the robust model is built off the polynomial model and provides no additional lift, we believe the polynomial model is the best model.

```
# Prepare the test set by adding poly terms
test_poly_df <- test_df %>%
  mutate(sqft_living = sqft_living - mean(sqft_living),
         sqft_living_2 = sqft_living^2) %>%
  dplyr::select(price, sqft_living, sqft_living_2, waterfront, view, grade, n_yrs_ago_built, zipcode)

# Make predictions for each model using the test sample
model1_yhat <- predict(best_ols_cand1, newdata = test_df)
model2_yhat <- predict(best_ols_cand2, newdata = test_df)
model2poly_yhat <- predict(ols_poly_model, newdata = test_poly_df)
rlm_yhat <- predict(robust_model, newdata = test_poly_df)
wls_yhat <- predict(wls_model, newdata = test_poly_df)

# Calculate RMSE for each model
y_actual <- test_poly_df %>% pull(price)
```

```r
n_test <- nrow(test_poly_df)
rmse_ols1 <- sqrt(sum((y_actual - model1_yhat)^2) / n_test)
rmse_ols2 <- sqrt(sum((y_actual - model2_yhat)^2) / n_test)
rmse_poly <- sqrt(sum((y_actual - model2poly_yhat)^2) / n_test)
rmse_rlm <- sqrt(sum((y_actual - rlm_yhat)^2) / n_test)
rmse_wls <- sqrt(sum((y_actual - wls_yhat)^2) / n_test)

# SST and SSE
sst_test <- sum((y_actual - mean(y_actual))^2)
sse_cand1_test <- sum((y_actual - model1_yhat)^2)
sse_cand2_test <- sum((y_actual - model2_yhat)^2)
sse_poly_test <- sum((y_actual - model2poly_yhat)^2)
sse_rlm_test <- sum((y_actual - rlm_yhat)^2)
sse_wls_test <- sum((y_actual - wls_yhat)^2)

# Calculate adjusted R-squared for each model
n <- nrow(test_df)
adj_rsq_cand1 <- calc_rsq_adj(n, best_ols_cand1$rank - 1, sse_cand1_test, sst_test)
adj_rsq_cand2 <- calc_rsq_adj(n, best_ols_cand2$rank - 1, sse_cand2_test, sst_test)
adj_rsq_poly <- calc_rsq_adj(n, ols_poly_model$rank - 1, sse_poly_test, sst_test)
adj_rsq_rlm <- calc_rsq_adj(n, ols_poly_model$rank - 1, sse_rlm_test, sst_test)
adj_rsq_wls <- calc_rsq_adj(n, ols_poly_model$rank - 1, sse_wls_test, sst_test)


rmse_df <- tibble(Model = c('OLS Candidate 1', 'OLS Candidate 2', 'Polynomial', 'Robust', 'WLS'),
                  RMSE = c(rmse_ols1, rmse_ols2, rmse_poly, rmse_rlm, rmse_wls),
                  `Adj. R-Squared` = c(adj_rsq_cand1, adj_rsq_cand2, adj_rsq_poly,
                                       adj_rsq_rlm, adj_rsq_wls))

kable(rmse_df,
      format.args = list(big.mark = ","),
      caption = ' Root Mean Squared Error (RMSE) and Adjusted R-squared on the Test Set',
      booktabs = T) %>%
  kable_styling(full_width = F,
                latex_options = "HOLD_position")
```

Table 7: Root Mean Squared Error (RMSE) and Adjusted R-squared on the Test Set

Model	RMSE	Adj. R-Squared
OLS Candidate 1	198,347.0	0.7238776
OLS Candidate 2	168,695.3	0.8001395
Polynomial	163,999.0	0.8110829
Robust	164,455.0	0.8100307
WLS	223,572.5	0.6489040

Thus, the final model contains the following predictors:

Price = sqft_living + sqft_living_2 + waterfront + view + grade + n_yrs_ago_built + zipcode

V. Challenger Models (15 points)

```
# Train and Test Data already prepared during OLS phase, just copying dataframes
train_data <- train_df
test_data <- test_df
```

V.I Regression Tree

```
# Create Regression Tree Based on Predictors in Section III

library(rpart)
reg_tree <- rpart(price ~ ., data = train_data, cp=0.001)
reg_tree
```

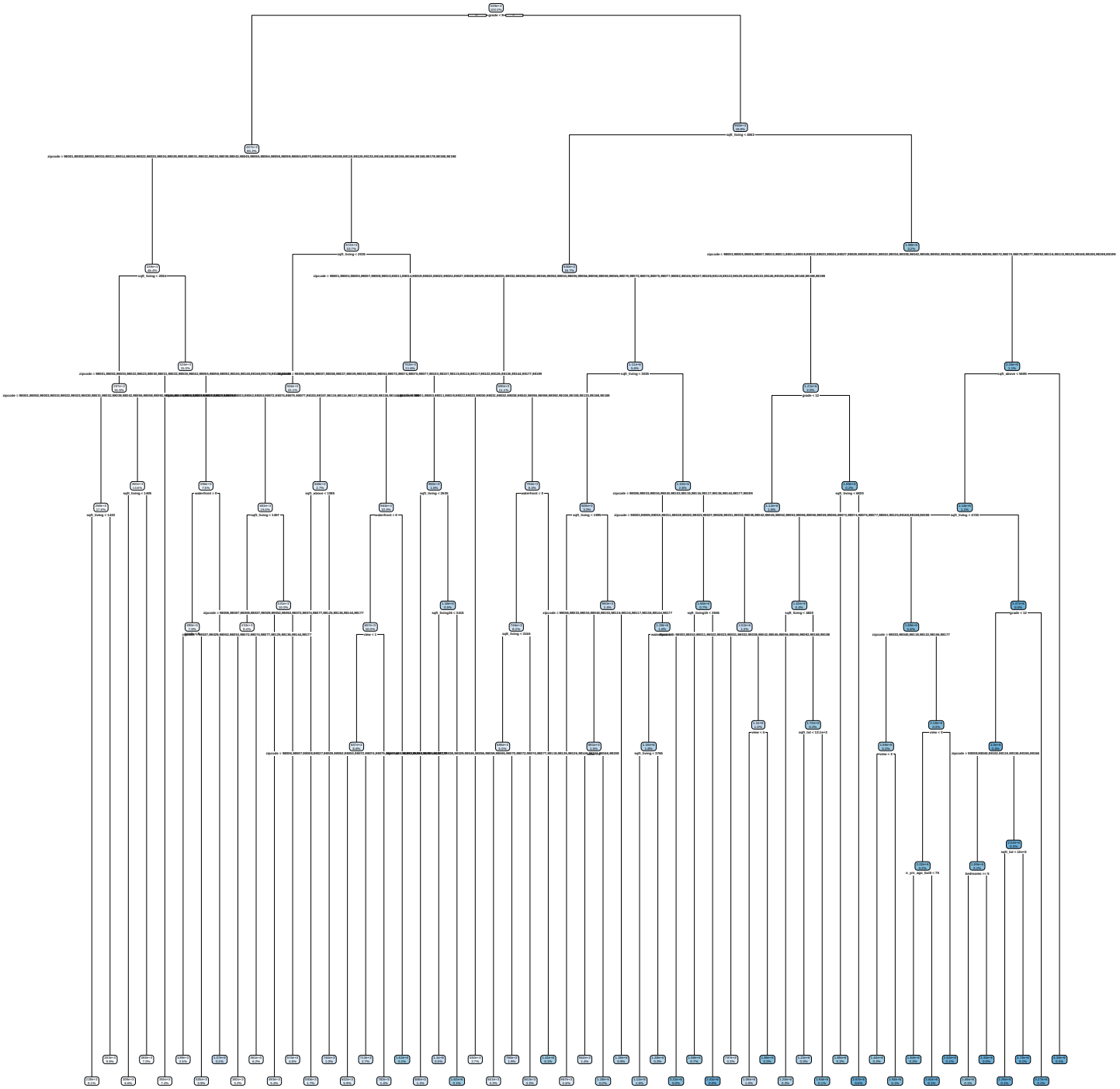
```
## n= 15129
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##   1) root 15129 1978906000000000  538713.5
##     2) grade< 8.5 12131  460103700000000  437022.5
##       4) zipcode=98001,98002,98003,98010,98011,98014,98019,98022,98023,98024,98028,98030,98031,98032,98034,
##         8) sqft_living< 2004 4760   39738220000000  297358.7
##          16) zipcode=98001,98002,98003,98010,98022,98023,98030,98031,98032,98038,98042,98055,98058,98092,98
##            32) sqft_living< 1432.5 1232    4825540000000  227613.9 *
##            33) sqft_living>=1432.5 1478    5366830000000  282682.3 *
##          17) zipcode=98011,98014,98019,98024,98028,98034,98045,98056,98059,98065,98070,98108,98118,98126,98
##            34) sqft_living< 1405 991    5988624000000  308826.8 *
##            35) sqft_living>=1405 1059    8367327000000  388248.5 *
##         9) sqft_living>=2004 2265   42879360000000  425881.6
##          18) zipcode=98001,98002,98003,98022,98023,98030,98031,98032,98038,98042,98055,98058,98092,98106,98
##          19) zipcode=98010,98011,98014,98019,98024,98028,98034,98045,98056,98059,98065,98070,98108,98118,98
##            38) waterfront< 0.5 1128   15377200000000  489581.3
##              76) grade< 7.5 533    4154025000000  438864.0 *
##              77) grade>=7.5 595    8624019000000  535013.8 *
##            39) waterfront>=0.5 13    2908849000000 1069881.0 *
##       5) zipcode=98004,98005,98006,98007,98008,98027,98029,98033,98039,98040,98052,98053,98072,98074,98075,
##        10) sqft_living< 2035 3344   62323190000000  503509.4
##          20) zipcode=98006,98007,98008,98027,98029,98033,98052,98053,98072,98074,98075,98077,98103,98107,98
##            40) sqft_living< 1456.5 1422   15698060000000  432953.7
##              80) zipcode=98027,98029,98052,98053,98072,98074,98077,98125,98136,98144,98177 491    298039400
##              81) zipcode=98006,98007,98008,98033,98075,98103,98107,98115,98116,98117,98122,98199 931   1064
##            41) sqft_living>=1456.5 1512   18217790000000  530796.0
##              82) zipcode=98006,98007,98008,98027,98029,98052,98053,98072,98074,98077,98125,98136,98144,9817
##              83) zipcode=98033,98075,98103,98107,98115,98116,98117,98122,98199 701    8546364000000  574020
##          21) zipcode=98004,98005,98039,98040,98102,98105,98109,98112,98119 410   11691490000000  647589.4
##            42) sqft_above< 1385 261    5040823000000  592674.1 *
##            43) sqft_above>=1385 149    4484836000000  743783.4 *
##        11) sqft_living>=2035 1762   83105520000000  702460.3
##          22) zipcode=98005,98006,98007,98008,98027,98029,98033,98052,98053,98072,98074,98075,98077,98103,98
##            44) waterfront< 0.5 1510   35527290000000  657086.2
##              88) view< 0.5 1300   25410330000000  636929.3
##                176) zipcode=98006,98007,98008,98027,98029,98052,98053,98072,98074,98075,98077,98116,98125,98
##                177) zipcode=98005,98033,98103,98107,98115,98117,98122,98199 411    9533065000000  712783.3 *
##              89) view>=0.5 210    6319024000000  781866.7 *
##            45) waterfront>=0.5 9    1352884000000 1619556.0 *
##          23) zipcode=98004,98039,98040,98102,98105,98109,98112,98119 243   20602990000000  950448.0
##            46) sqft_living< 2635 161    5302402000000  845865.2 *
##            47) sqft_living>=2635 82   10082170000000 1155787.0
##              94) sqft_living15< 3415 73    6810131000000 1098205.0 *
```

```
##              95) sqft_living15>=3415 9    1066707000000 1622844.0 *
##     3) grade>=8.5 2998  885748100000000  950192.7
##       6) sqft_living< 4062.5 2519  315900300000000  830227.4
##        12) zipcode=98001,98003,98005,98007,98008,98010,98011,98014,98019,98022,98023,98024,98027,98028,9802
##          24) zipcode=98001,98003,98011,98019,98022,98023,98030,98031,98032,98038,98042,98055,98058,98092,98
##          25) zipcode=98005,98007,98008,98010,98014,98024,98027,98028,98029,98045,98052,98053,98056,98059,98
##            50) waterfront< 0.5 1248   35111450000000  746044.1
##             100) sqft_living< 3155 764   15385680000000  685187.8
##               200) zipcode=98010,98014,98024,98027,98028,98029,98045,98056,98059,98065,98070,98072,98074,98
##               201) zipcode=98005,98007,98008,98052,98053,98075,98107,98122 282    5632550000000  759948.6 *
##             101) sqft_living>=3155 484   12429960000000  842106.5 *
##            51) waterfront>=0.5 19    7643001000000 1407666.0 *
##        13) zipcode=98004,98006,98033,98034,98039,98040,98102,98103,98105,98109,98112,98115,98116,98117,9811
##          26) sqft_living< 3035 455   37599550000000  929355.2
##            52) sqft_living< 1995 86    2224901000000  667063.4 *
##            53) sqft_living>=1995 369   28079180000000  990485.5
##             106) zipcode=98006,98033,98034,98040,98103,98115,98116,98117,98136,98144,98177 232    996119600
##               212) view< 2.5 206    5706803000000  843802.0 *
##               213) view>=2.5 26    1700702000000 1176391.0 *
##             107) zipcode=98004,98039,98102,98105,98109,98112,98119,98199 137   10637790000000 1175765.0 *
##          27) sqft_living>=3035 391   67990870000000 1316688.0
##            54) zipcode=98006,98033,98034,98040,98103,98115,98116,98117,98136,98144,98177,98199 278   343779
##             108) waterfront< 0.5 271   26110580000000 1160745.0
##               216) sqft_living< 3765 221   16933840000000 1111687.0 *
##               217) sqft_living>=3765 50    6293992000000 1377580.0 *
##             109) waterfront>=0.5 7     756902900000 2209857.0 *
##            55) zipcode=98004,98039,98102,98105,98109,98112,98119,98178 113   17474460000000 1635347.0
##             110) sqft_living15< 3945 106   11903050000000 1583865.0 *
##             111) sqft_living15>=3945 7    1036240000000 2414929.0 *
##       7) sqft_living>=4062.5 479  342947100000000 1581075.0
##        14) zipcode=98003,98005,98006,98007,98010,98011,98014,98019,98022,98023,98024,98027,98028,98029,9803
##          28) grade< 11.5 269   45517550000000 1117753.0
##            56) zipcode=98003,98005,98010,98011,98019,98022,98023,98027,98028,98031,98032,98038,98042,98045,
##             112) zipcode=98003,98010,98011,98022,98023,98031,98032,98038,98042,98045,98056,98065,98092,9816
##             113) zipcode=98005,98019,98027,98028,98052,98053,98058,98059,98072,98074,98075,98077,98125,9819
##               226) view< 3.5 144    6761045000000 1061039.0 *
##               227) view>=3.5 7    5006705000000 1857921.0 *
##            57) zipcode=98006,98007,98014,98024,98029,98034,98116,98118,98199 66   16010280000000 1415593.0
##             114) sqft_living< 4820 43    3858485000000 1254541.0 *
##             115) sqft_living>=4820 23    8951305000000 1716690.0
##               230) sqft_lot< 12119 7     509772500000 1214500.0 *
##               231) sqft_lot>=12119 16    5903823000000 1936398.0 *
##          29) grade>=11.5 37   18321200000000 1875497.0
##            58) sqft_living< 6820 30    6028977000000 1690450.0 *
##            59) sqft_living>=6820 7    6862354000000 2668556.0 *
##        15) zipcode=98004,98008,98033,98039,98040,98102,98105,98107,98109,98112,98115,98119,98122,98136,9814
##          30) sqft_above< 5685 165   87269400000000 2136728.0
##            60) sqft_living< 4725 95   32646850000000 1891728.0
##             120) zipcode=98033,98040,98119,98122,98166,98177 47   10683810000000 1594429.0
##               240) view< 2.5 32    4252696000000 1440304.0 *
##               241) view>=2.5 15    4049322000000 1923230.0 *
##             121) zipcode=98004,98008,98039,98102,98107,98109,98112,98115,98144 48   13741250000000 2182833.
##               242) view< 2.5 35    7950692000000 2019136.0
##                 484) n_yrs_ago_built< 75 27    2822467000000 1843731.0 *
##                 485) n_yrs_ago_built>=75 8    1493915000000 2611125.0 *
##               243) view>=2.5 13    2327568000000 2623558.0 *
##            61) sqft_living>=4725 70   41181310000000 2469227.0
##             122) grade< 11.5 58   22459540000000 2303523.0
##               244) zipcode=98008,98040,98102,98115,98136,98155,98166 20    6097079000000 1893073.0
##                 488) bedrooms>=4.5 13    1767188000000 1658808.0 *
```

```
##                 489) bedrooms< 4.5 7    2291483000000 2328137.0 *
##               245) zipcode=98004,98033,98039,98105,98109,98112,98119,98144 38   11219710000000 2519550.0
##                 490) sqft_lot< 12041.5 17    1894113000000 2256029.0 *
##                 491) sqft_lot>=12041.5 21    7189391000000 2732876.0 *
##             123) grade>=11.5 12    9431973000000 3270125.0 *
##          31) sqft_above>=5685 8   19126300000000 4338250.0 *
```

```r
# plot tree
library(rpart.plot)
rpart.plot(reg_tree, digits = 3)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



V.II Regression Tree Performance Evaluation

```
# IN SAMPLE TEST (Train Data)

price_predict <- predict(reg_tree, train_data)
summary(price_predict) #summary of predicted values
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  227614  354507  460676  538714  601861 4338250
```

```
summary(train_data$price) #summary of actual values
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   75000  320000  450000  538714  641250 7700000
```

```
corr_insample <- cor(price_predict, train_data$price)

# Measuring performance with the mean absolute error
MAE <- function(actual, predicted) {mean(abs(actual - predicted))}
mae_in_sample <- MAE(train_data$price, price_predict)

# Calculate RMSE
rmse_reg_tree_in_sample <- sqrt(sum((train_data$price - price_predict)^2) / nrow(train_data))

# Measuring performance with the SSE
SSE <- function(actual, predicted) {sum((actual - predicted)^2)}
sse_in_sample <- SSE(train_data$price, price_predict)

# Measuring performance with the R-square
R2 <- function(actual, predicted) {sum((actual - predicted)^2)/((length(actual)-1)*var(actual))}
r2_tree_train <- 1 - R2(train_data$price, price_predict)
```

```
# OUT OF SAMPLE TEST (Test Data)

price_predict <- predict(reg_tree, test_data)
summary(price_predict) #summary of predicted values
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  227614  354507  460676  540073  601861 4338250
```

```
summary(test_data$price) #summary of actual values
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   81000  325000  450000  543295  648000 7062500
```

```
cor_test <- cor(price_predict, test_data$price)

# Measuring performance with the mean absolute error
MAE <- function(actual, predicted) {mean(abs(actual - predicted))}
mae_test <- MAE(test_data$price, price_predict)

# Calculate RMSE
rmse_reg_tree_test <- sqrt(sum((test_data$price - price_predict)^2) / nrow(test_data))


# Measuring performance with the SSE
SSE <- function(actual, predicted) {sum((actual - predicted)^2)}
reg_sse_test <- SSE(test_data$price, price_predict)

# Measuring performance with the R-square
R2 <- function(actual, predicted) {sum((actual - predicted)^2)/((length(actual)-1)*var(actual))}
r2_tree_test <- 1 - R2(test_data$price, price_predict)
```

```r
# Present Results
rmse_df_in_out_sample <- tibble(Test = c('In Sample (Train)', 'Out of Sample (Test)'),
                  `Correlation` = c(corr_insample, cor_test),
                  MAE = c(mae_in_sample, mae_test),
                  SSE = c(sse_in_sample, reg_sse_test),
                  RMSE = c(rmse_reg_tree_in_sample, rmse_reg_tree_test),
                  "$R^2$" = c(r2_tree_train, r2_tree_test))

kable(rmse_df_in_out_sample,
      format.args = list(big.mark = ","),
      caption = 'In-Sample versus Out-of-Sample comparison',
      booktabs = T) %>%
  kable_styling(full_width = F,
                latex_options = "HOLD_position")
```

Table 8: In-Sample versus Out-of-Sample comparison

Test	Correlation	MAE	SSE	RMSE	R^2
In Sample (Train)	0.9152820	90,145.88	321,094,872,218,829	145,683.9	0.8377412
Out of Sample (Test)	0.8846356	97,456.69	203,126,842,795,038	176,995.5	0.7824997

V.II Conclusion

Based on the results for both the out-of-sample and in-sample tests for the model, we believe this regression tree model is fit for purpose as its results are similar to the results obtained from the regression models in earlier sections. When compared to the polynomial model developed in *Section IV.IV.II*, the regression tree model developed in this section performs slightly worse when comparing its out-of-sample results. However, when looking at its in-sample results, the regression tree performed better than the polynomial model in both its R-squared and RMSE metrics.

VI. Model Limitation and Assumptions (15 points)

Based on the test sample results for all of our models, we choose the Polynomial model as our champion model. Its prediction performance is the best out of all models tested, with both the lowest SSE + RMSE values and highest Adjusted R-Squared. Looking at the Q-Q plot of the residuals we can see that once again the polynomial model shows the best distribution of residuals [Appendix A].

```
# Getting the SSE of our models
rmse_df <- tibble(
  Model = c('OLS Candidate 1', 'OLS Candidate 2', 'Polynomial', 'Robust', 'WLS', 'Regression Tree'),
  `SSE` = c(sse_cand1_test, sse_cand2_test, sse_poly_test, sse_rlm_test, sse_wls_test, reg_sse_test),
  RMSE = c(rmse_ols1, rmse_ols2, rmse_poly, rmse_rlm, rmse_wls, rmse_reg_tree_test),
  `Adj. R-Squared` = c(adj_rsq_cand1, adj_rsq_cand2, adj_rsq_poly, adj_rsq_rlm, adj_rsq_wls, r2_tree_test))

kable(rmse_df,
      format.args = list(big.mark = ","),
      caption = 'Sum Squared of Errors, Root Mean Squared Error (RMSE) and
      Adjusted R-squared on the Test Set',
      booktabs = T) %>%
  kable_styling(full_width = F,
                latex_options = "HOLD_position")
```

Table 9: Sum Squared of Errors, Root Mean Squared Error (RMSE) and Adjusted R-squared on the Test Set

Model	SSE	RMSE	Adj. R-Squared
OLS Candidate 1	255,090,526,233,546	198,347.0	0.7238776
OLS Candidate 2	184,522,252,253,861	168,695.3	0.8001395
Polynomial	174,391,452,412,820	163,999.0	0.8110829
Robust	175,362,783,884,148	164,455.0	0.8100307
WLS	324,100,620,402,634	223,572.5	0.6489040
Regression Tree	203,126,842,795,038	176,995.5	0.7824997

VI.I Assumptions

There are several assumptions that must be made in order to support the champion model. Foremost is that the data is not perfectly clean, with the existence of several large outliers which have larger than normal leverage over the whole model. We found that, while this was somewhat problematic, the benefits of adjusting the model to better explain the outliers ultimately did more harm than good. Likewise, we follow the rationale that most prices will tend to mean/median of the county. This is due to factors external to that which is measured by the dataset, such as fair market competition. As such, going forward we would expect there to be few additional outliers.

Likewise, one must assume that the heteroskedasticity of the models is at acceptable levels. VIF tests of our models show that multicollinearity is as acceptable levels. Even with BoxCox transformations, while the graphs may appear to look normal and heteroskedasticity solved, follow up Breusch-Pagan tests reveal that we still cannot reject the null hypothesis [Appendix B]. As a result we have decided that other remedial measures would be more acceptable and better explained to potential stakeholders.

VI.II Limitations

There are several limitations to the models presented here.

The first limitation is the time period over which the data has been collected. Given that the data only encompasses roughly 1 year of data collection, certain macro-economic effects can not be measured despite their real world influence. Aspects such as seasonality and year-over-year trends can not be extracted with the current data. Another source of influence that can not be measured within this dataset is the impact of market interest rates which, as recent years have shown, hold large influence on the price of housing and rent. We additionally have the limitation of this data being collected from a single county within the United States. The relatively small geographic concentration of this data consequently means that the use of the models outside of the local area would be meaningless.

VII. Ongoing Model Monitoring Plan (5 points)

There are several aspects which must be considered when monitoring ongoing performance of the existing model.

To start we must first address data inconsistencies and missing context information. In our initial data exploration phase we identified several variables that lacked clarity, such as the bathrooms, sqft_lot, and grade. For the bathrooms variable, it exists as a sum of several constituent components. It is recommended for future data collection to identify the individual components of the bathroom, as described in Section II, and tally that count for more precise data interpretation (For example, interpretation of 3.75). For sqft_lot, we identified some inconsistencies within the measurements that would have been better explained with more robust data clarification. As for the grade variable, this is a largely subjective entity. It exists as a matter of personal or arbitrary judgments on the part of the individual(s) assessing the property. No information is given as to the criterion used in determining the grade, nor is there any note of if those criteria were ever changed throughout the data collection process.

Therefore it is our overall recommendation, that in the ongoing monitoring of the model, to note any changes in the approach to data collection. More precise data collection would have substantial ramifications on the quality of the existing model. One such example would be if the factors determining the grade of homes were to change and all houses reassessed. This could potentially lead to changes in the predictive qualities of the variable and would force us to reconsider our models.

Next, in the OLS models, we note that the variables which have the largest influence over sale price mostly consist of factors external to the house itself. That is, factors which either require substantial development of the home or are inherent to the location of the house. Considering first the square feet of living space and the grade of the home, these are factors that, while they can be changed, are also subject to regulatory policies. To accurately maintain future performance of this model, our next suggestion is to closely monitor regulations involving housing developments and renovations. New legislation could change new housing developments or mandate that certain changes be made to existing ones. Any such changes would have to be noted and consequently measured over time to determine whether or not they had any influential impact on the model. Should changes to the model be observed, perhaps additional variables should then be considered for model performance.

Quantitative thresholds that should be considered are primarily centered around the existence of outliers and the RMSE, Root Mean Square Error, of the model. The best performance metric we were able to achieve with our models is $RMSE = 163,999.0$ with $Adj.R^2 = 0.8110829$ for polynomial regression and $RMSE = 176,995.5$ with $Adj.R^2 = 0.7824997$ for the regression tree. These out-of-sample test results should be used as benchmarks going forward, with future models attempting to either reduce RMSE or increase adjusted R-squared, preferably both. For data outliers, as can be seen in figure ###, the presence of said outliers presents a challenge in correctly fitting the model to the data. Should further data compound these issues then additional remedial measures must be taken to properly adjust the model for best fitment.

We can also look to existing recommendations given by the Federal Reserve in regards to model risk management. Per the supervisory guidance, we must first admit that "[our] model may have fundamental errors and may produce inaccurate outputs when viewed against the design objective and intended business uses"[2]. We believe this to be true given the nature of errors we identified in the model, such as high multicollinearity between several variables, heteroskedasticity issues, and some slight departures from normality. However we assume that despite these issues, the performance of the model remains robust and is viable for sale price prediction.

Likewise the National Association of Insurance Commissioners, a regulatory board overseeing matters of public and private insurance, recommends that model performance be assessed in combination with real world feedback from businesses and users of the model [3]. In their white paper, they discuss that only individuals with sufficient skills, knowledge and domain expertise should be responsible for the successful assessment of any particular model. This is due to complex statistical nature inherent to these models which some may find difficult to both understand and explain to other stakeholders their significance.

Finally, it should be noted that a "challenge from model users may be weak if the model does not materially affect their results"[2]. This is to say that if the model is taken with blind faith, there may be adverse business effects further down the line. This is why we present here several challenger models for price prediction, utilizing both traditional statistical methods of creating OLS models, as well as regression tree models. The ability to compare the performance of these models is crucial in understanding their individual strengths and weaknesses. Going forward we expect that the same approach should be had in order to maintain consistent and relevant model performance.

VIII. Conclusion (5 points)

```
rmse_df <- tibble(Model = c('Polynomial', 'Reg. Tree'),
                  RMSE = c(rmse_poly, rmse_reg_tree_test),
                  `Adj. R-Squared` = c(adj_rsq_poly,  r2_tree_test))

kable(rmse_df,
      format.args = list(big.mark = ","),
      caption = ' Root Mean Squared Error (RMSE) and Adjusted R-squared on the Test Set',
      booktabs = T) %>%
  kable_styling(full_width = F,
                latex_options = "HOLD_position")
```

Table 10: Root Mean Squared Error (RMSE) and Adjusted R-squared on the Test Set

Model	RMSE	Adj. R-Squared
Polynomial	163,999.0	0.8110829
Reg. Tree	176,995.5	0.7824997

VIII.I Champion Model

The results of the Regression Model in **Section V** showed that the Polynomial Model performed developed in **Section IV.IV.II** better than all the alternative models developed during the process, with the robust model having similar results but slightly worse. As shown in the table above, the polynomial model has an R-squared of 0.8110829 and an RMSE of 163,999.

VIII.II Challenger Model

As for our challenger model from section V, the **Regression Tree** did not performed poorly, but had slightly worse results than the polynomial model. As shown in the table above, the R-squared of the regression tree is 0.7824997, and its RMSE is 176,995.5.

VIII.III - Winner: Champion Model

Based on these results, the Polynomial Model from section III is the most ideal model for our data, as it provides the highest R-squared value and the lowest RMSE. In addition, when taking into account the assumptions and limitations mentioned in section VI, the polynomial model's performance is very high. However, if all those assumptions and limitations were to be addressed, the performance of both the polynomial model and the regression tree could either increase or decrease, and the regression tree could become the most ideal model since the current results are not too far away from each other. But as things stand right now, the polynomial model is the most ideal model based on the data available, our assumptions, and the limitations.

Bibliography (7 points)

Please include all references, articles and papers in this section.

[1] Wikipedia. *King County, Washington.* https://en.wikipedia.org/wiki/King_County,_Washington.

[2] Board of Governors of the Federal Reserve System. *SR Letter 11-7: Supervisory Guidance on Model Risk Management.* 2011. https://www.federalreserve.gov/supervisionreg/srletters/sr1107a1.pdf.

[3] National Association of Insurance Commissioners. *Casualty Actuarial and Statistical (C) Task Force Regulatory Review of Predictive Models White Paper.* August 27, 2020. https://portal.ct.gov/cid/-/media/CID/Regulatory-Review-of-Predictive-Models-White-Paper.pdf

[4] Yuko, Elizabeth. *The Actual Difference Between a Half, 3/4, and Full Bathroom.* April 10, 2012. https://lifehacker.com/the-actual-difference-between-a-half-3-4-and-full-bat-1848773483.

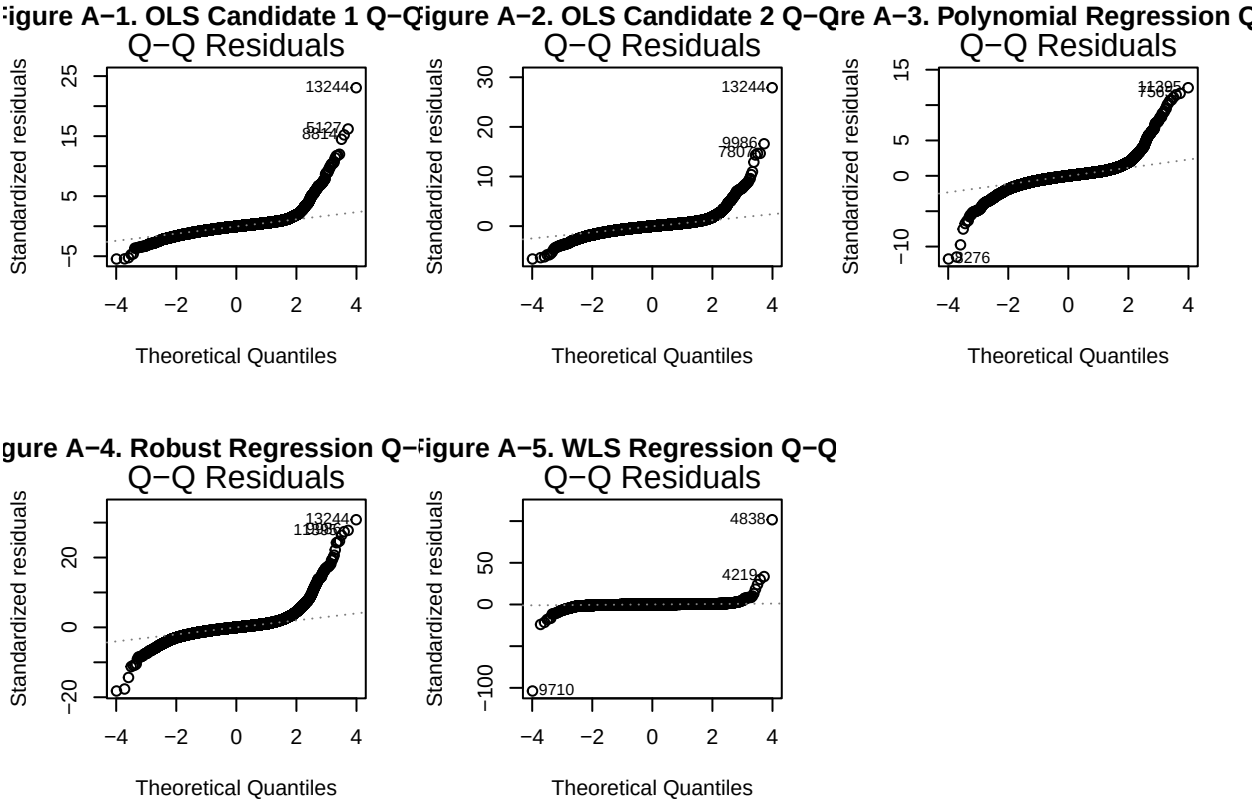
[5] Pacheco, Kaitlyn. *The 2022 U.S. Lot Size Index.* August 5, 2022. https://www.angi.com/articles/lot-size-index.htm.

[6] Kutner, Nachtsheim, Neter, Li. *Applied Linear Statistical Models, Fifth Edition.* 2005.

Appendix (3 points)

A. OLS Q-Q Plots

```
# Move to Appendix A
par(mfrow=c(2,3))
plot(best_ols_cand1, which=2, main = "Figure A-1. OLS Candidate 1 Q-Q Plot")
plot(best_ols_cand2, which=2, main = "Figure A-2. OLS Candidate 2 Q-Q Plot")
plot(ols_poly_model, which=2, main = "Figure A-3. Polynomial Regression Q-Q Plot")
plot(robust_model, which=2, main = "Figure A-4. Robust Regression Q-Q Plot")
plot(wls_model, which=2, main = "Figure A-5. WLS Regression Q-Q Plot")
```



B. Breush-Pagan Tests of OLS Models

Here we explore the results of Breusch-Pagan Tests as they relate to our OLS Models. We provide a Box-Cox transformation of our champion model to show that any such transformation does not provide meaningful changes to the quality of the model.

```
# Move to Appendix B
bptest(best_ols_cand1, studentize = FALSE)
```

```
##
##  Breusch-Pagan test
##
## data:  best_ols_cand1
## BP = 42207, df = 70, p-value < 0.00000000000000022
```

```
bptest(best_ols_cand2, studentize = FALSE)
```

```
##
##  Breusch-Pagan test
##
## data:  best_ols_cand2
## BP = 58590, df = 74, p-value < 0.00000000000000022
```

```
bptest(ols_poly_model, studentize = FALSE)
```

```
##
##  Breusch-Pagan test
##
## data:  ols_poly_model
## BP = 52866, df = 75, p-value < 0.00000000000000022
```

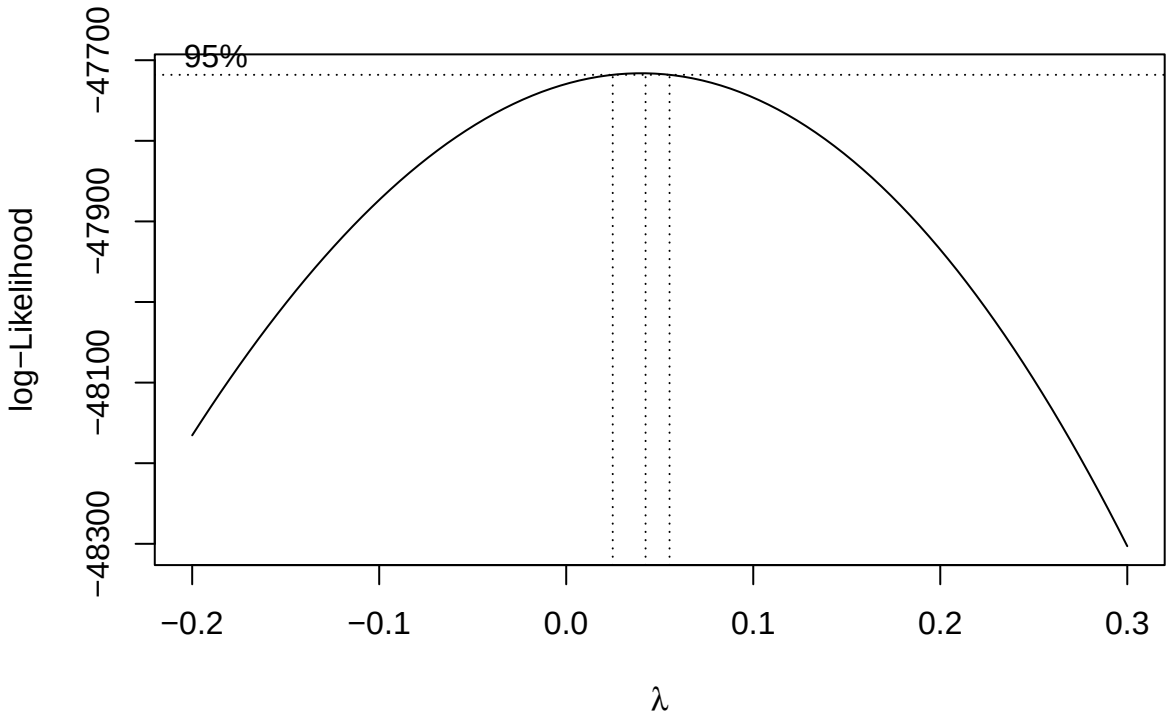
```
bptest(robust_model, studentize = FALSE)
```

```
##
##  Breusch-Pagan test
##
## data:  robust_model
## BP = 52866, df = 75, p-value < 0.00000000000000022
```

```
bptest(wls_model, studentize = FALSE)
```

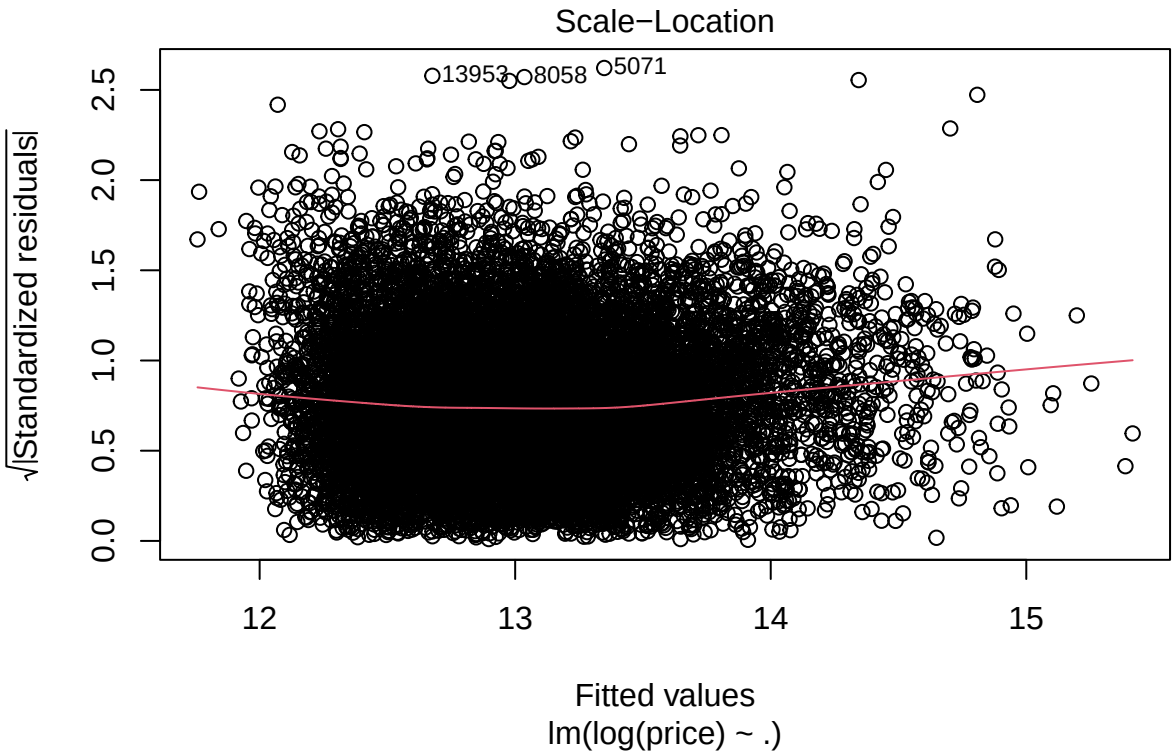
```
##
##  Breusch-Pagan test
##
## data:  wls_model
## BP = 906769420346, df = 75, p-value < 0.00000000000000022
```

```
boxcox(ols_poly_model, lambda = seq(-.2,.3,.01))
```



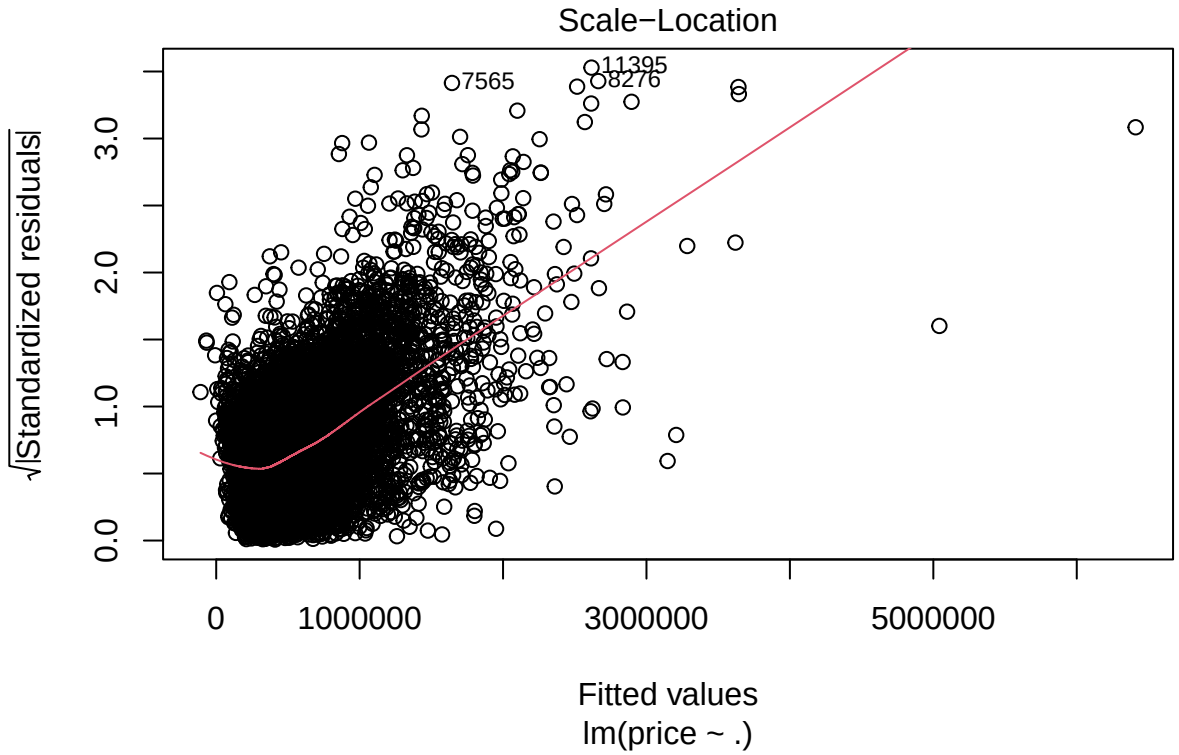

```
log_poly <- lm(log(price) ~ ., data = train_poly_df)
plot(log_poly, which=3, main = "Figure B-1. Log-Polynomial Scale-Location Plot")
```

Figure B−1. Log−Polynomial Scale−Location Plot




```
plot(ols_poly_model, which=3, main = "Figure B-2. Polynomial Regression Scale-Location Plot")
```

Figure B−2. Polynomial Regression Scale−Location Plot




```
tidy(log_poly) %>%
  dplyr::select(Term = term, Coefficient = estimate, std.error, p.value) %>%
  mutate(Coefficient = round(Coefficient, 3),
         std.error = round(std.error, 4),
         p.value = round(p.value, 4)) %>%
  slice(1:7) %>%
  kable(caption = "OLS Model with Second Order Term and Log Transformed Price",
        booktabs = T) %>%
  kable_styling(full_width = F,
                latex_options = "HOLD_position")
```

Table 11: OLS Model with Second Order Term and Log Transformed Price

Term	Coefficient	std.error	p.value
(Intercept)	11.688	0.0223	0
sqft_living	0.000	0.0000	0
waterfront	0.451	0.0209	0
view	0.067	0.0024	0
grade	0.111	0.0024	0
n_yrs_ago_built	0.001	0.0001	0
sqft_living_2	0.000	0.0000	0

```
bptest(log_poly, studentize=FALSE)
```

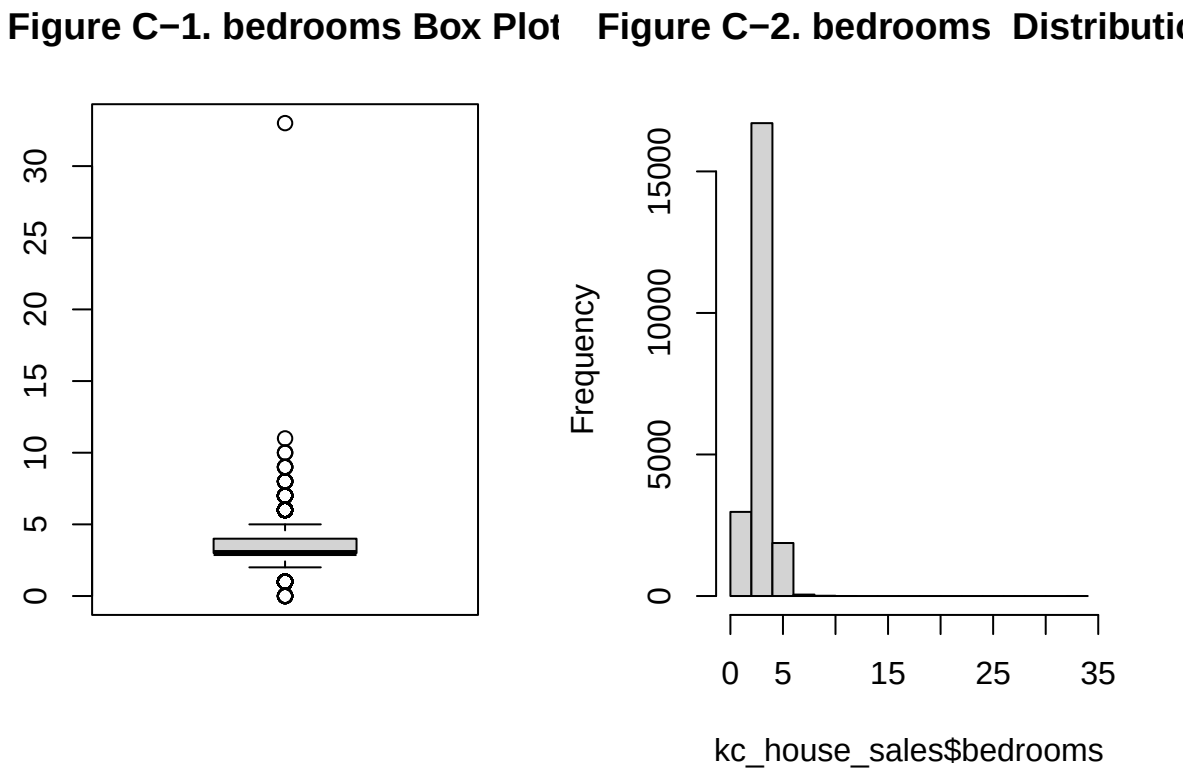
```
##
##  Breusch-Pagan test
##
## data:  log_poly
## BP = 2346.6, df = 75, p-value < 0.00000000000000022
```

C. Exploratory Analysis of Additional Variables

The following sections include additional plots of variables that were not included in the final models.

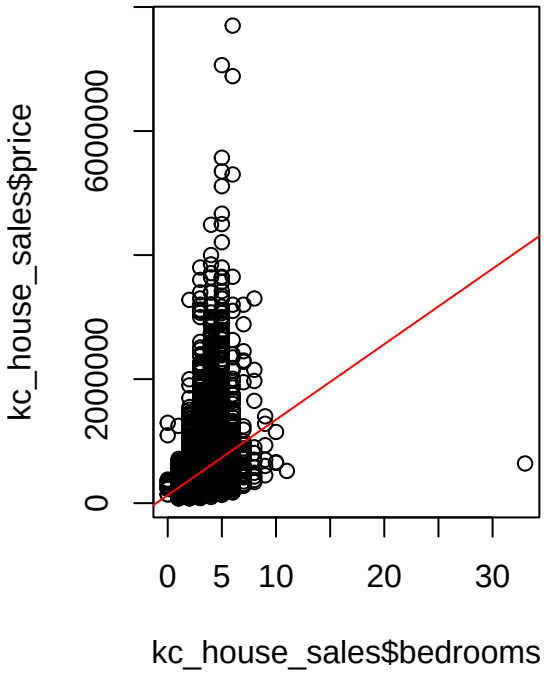
```
# bedrooms box plot, histogram, scatter plot vs. price
par(mfrow=c(1,2))
boxplot(kc_house_sales$bedrooms, main = "Figure C-1. bedrooms Box Plot")
hist(kc_house_sales$bedrooms, main = "Figure C-2. bedrooms  Distribution")
```

Bedrooms




```
par(mfrow=c(1,2))
plot(kc_house_sales$bedrooms, kc_house_sales$price,  main = "Figure C-3. bedrooms Scatter Plot");
abline(lm(price ~ bedrooms, data = kc_house_sales), col = "red")
```

Figure C−3. bedrooms Scatter Pl



Looking at the number of bedrooms and the box plot in Figure C-1, there are outliers on the lower and upper ends of the range of bedroom values, with one observation having more than 30 bedrooms. Per Figure C-2, it appears as though the vast majority of houses have between two and four bedrooms. It would be interesting to know the state of the "houses" with 0 bedrooms–are these studio apartments, cabins, plots of bare land, houses that have been gutted or otherwise have no interiors and thus no bedrooms, commercial properties accidentally included in the data set, typos, or is there another explanation?

Figure C-3 displays what appears to be a curvilinear relationship between the number of bedrooms and the house price, increasing until reaching five bedrooms and then falling again.

```r
# bathrooms box plot, histogram, scatter plot vs. price
par(mfrow=c(1,2))
boxplot(kc_house_sales$bathrooms, main = "Figure C-4. bathrooms Box Plot")
hist(kc_house_sales$bathrooms, main = "Figure C-5. bathrooms Distribution")
```

Bathrooms

Figure C−4. bathrooms Box Plo

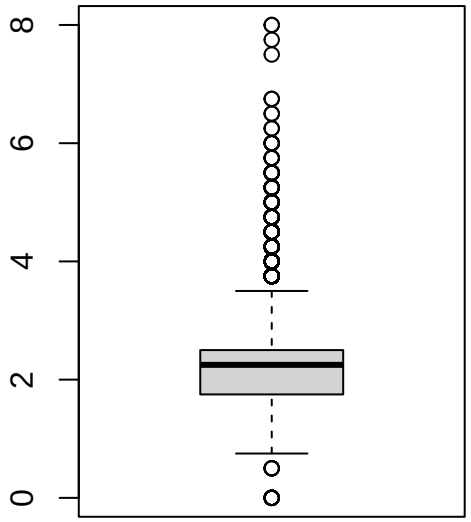
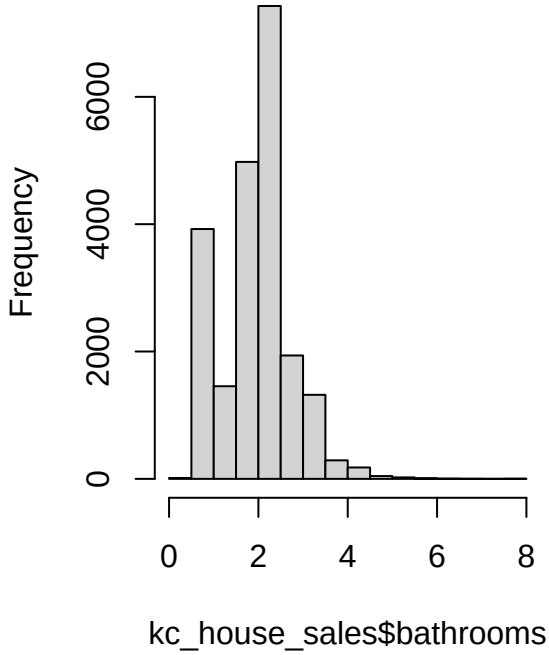
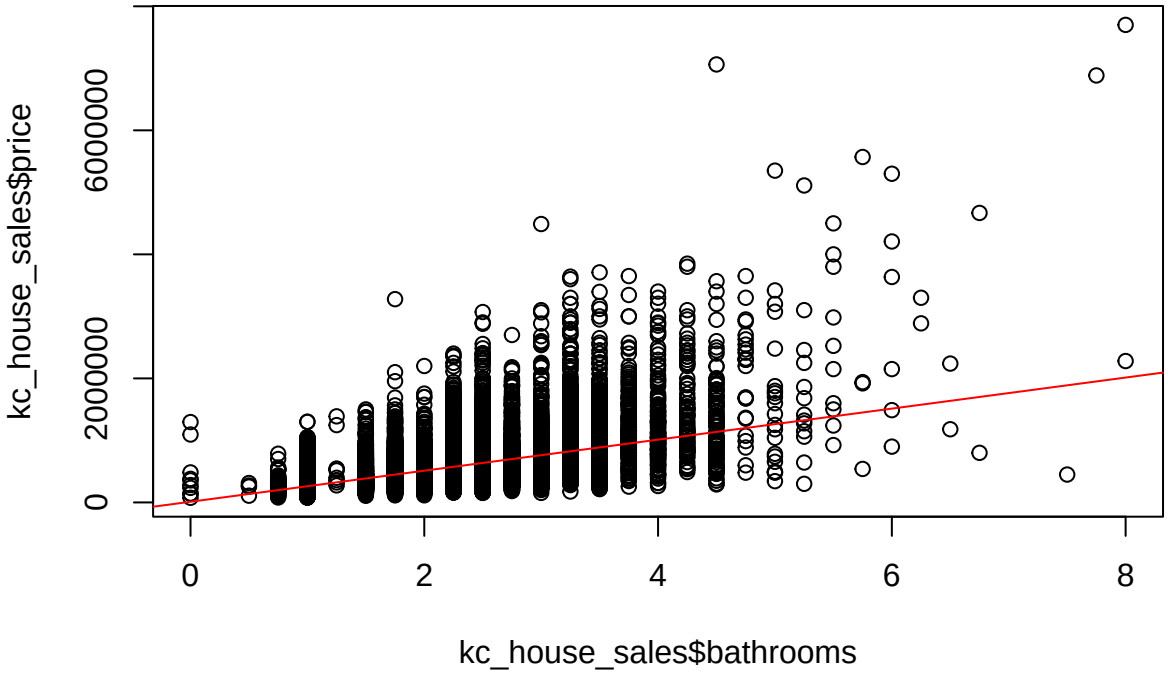


Figure C−5. bathrooms Distributi




```r
par(mfrow=c(1,1))
plot(kc_house_sales$bathrooms, kc_house_sales$price,  main = "Figure C-6. bathrooms Scatter Plot");
abline(lm(price ~ bathrooms, data = kc_house_sales), col = "red")
```

Figure C−6. bathrooms Scatter Plot



The bathroom predictor variable presented confusion upon first glance. Assuming the data is for residential real estate in the King County district, it seems odd that it would be possible for a house to have less than one full bathroom, yet there were multiple observations with less than 1.0 bathroom (and some with 0 bathrooms or fractions of a bathroom ending in .25). Upon researching the definitions used for bathroom terminology in the real estate industry [4], it appears as though each major component of a bathroom–toilet, sink, shower, and tub–counts as 0.25. It might have been more interesting for the bathroom data to be split into two columns: full_bathrooms and partial_bathrooms since, for example, if a house has 2 bathrooms according to the data set, it is unclear whether that means two full bathrooms, one full bathroom and two half bathrooms (toilet and sink), or another configuration. As for the houses with less than 1.0 bathroom, it would be interesting to know the reason–is the house unfinished? Is it a tiny house with just a composting toilet in the bathroom?

Overall, per Figure C-4 and Figure C-5, most houses have around 2 bathrooms, with outliers at the lower and upper end of the ranges. Per Figure C-6, it appears as though there is a linear relationship between number of bathrooms and price of the house.

```r
# sqft_lot box plot, histogram, scatter plot vs. price
par(mfrow=c(1,2))
boxplot(kc_house_sales$sqft_lot, main = "Figure C-7. sqft_lot Box Plot")
hist(kc_house_sales$sqft_lot, main = "Figure C-8. sqft_lot Distribution")
```

Square Footage - Lot

Figure C−7. sqft_lot Box Plot

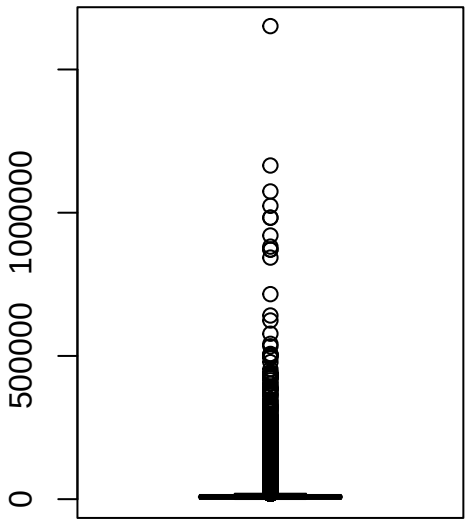
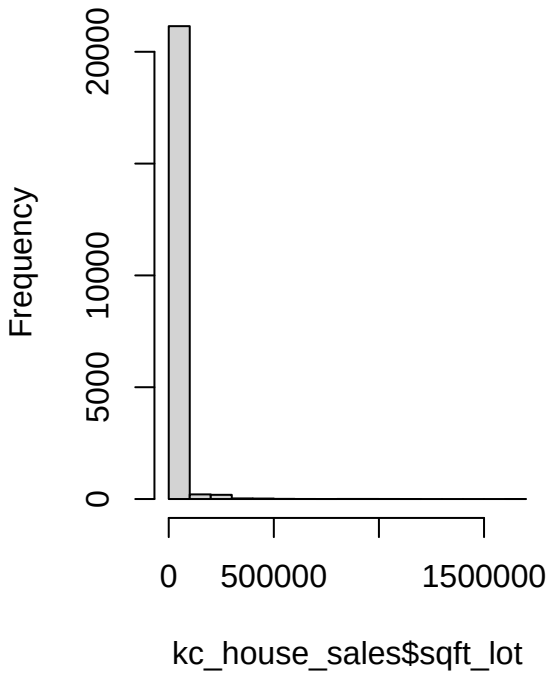
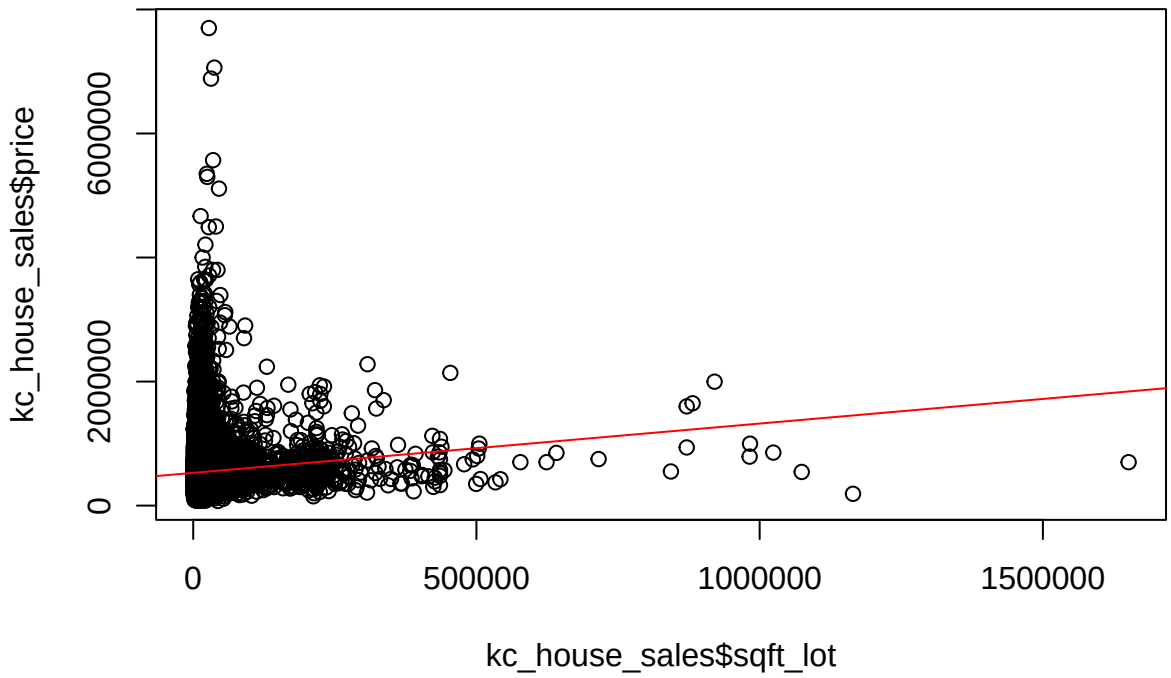


Figure C−8. sqft_lot Distributior




```r
par(mfrow=c(1,1))
plot(kc_house_sales$sqft_lot, kc_house_sales$price,  main = "Figure C-9. sqft_lot Scatter Plot");
abline(lm(price ~ sqft_lot, data = kc_house_sales), col = "red")
```

Figure C−9. sqft_lot Scatter Plot



Per Figure C-7 and Figure C-8 showing the box plot and distribution of the sqft_lot it appears as though the lot size is very concentrated around the lower end of the spectrum, with a few extreme outliers. According to the scatter plot in C-9, it looks like there is somewhat of a negative linear relationship between lot size and house price.

Looking at some of the individual data points, it seems as though some of the lot sizes might be impossible considering the square footage of the living area. For example, the row with ID 9828702895 has a sqft_lot size of 520, a sqft_living of 2420, no basement (sqft_above is also 2420), and only 1.5 floors. With only 1.5 floors and no basement, it does not seem possible to have a square footage of 2420 sqft on a lot size of 520 sqft, considering that the lot size typically includes the house's footprint [5]. Unfortunately, without further clarification, it is not possible to exclude data points based on this rationale. The "square footage of the land space" definition provided with the data set could mean the land excluding the house's footprint.

```r
# sqft_above box plot, histogram, scatter plot vs. price
par(mfrow=c(1,2))
boxplot(kc_house_sales$sqft_above, main = "Figure C-10. sqft_above Box Plot")
hist(kc_house_sales$sqft_above, main = "Figure C-11. sqft_above Distribution")
```

Square Footage - Above Ground

Figure C−10. sqft_above Box Plc

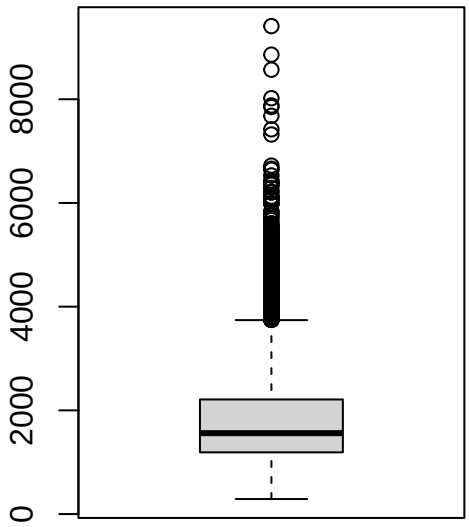
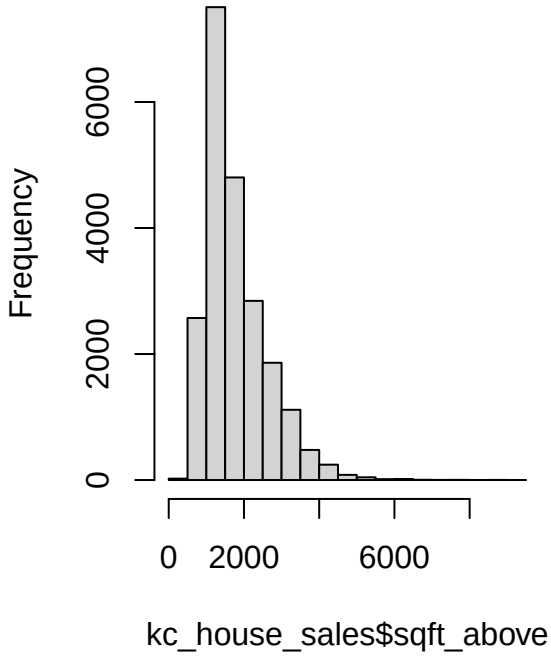
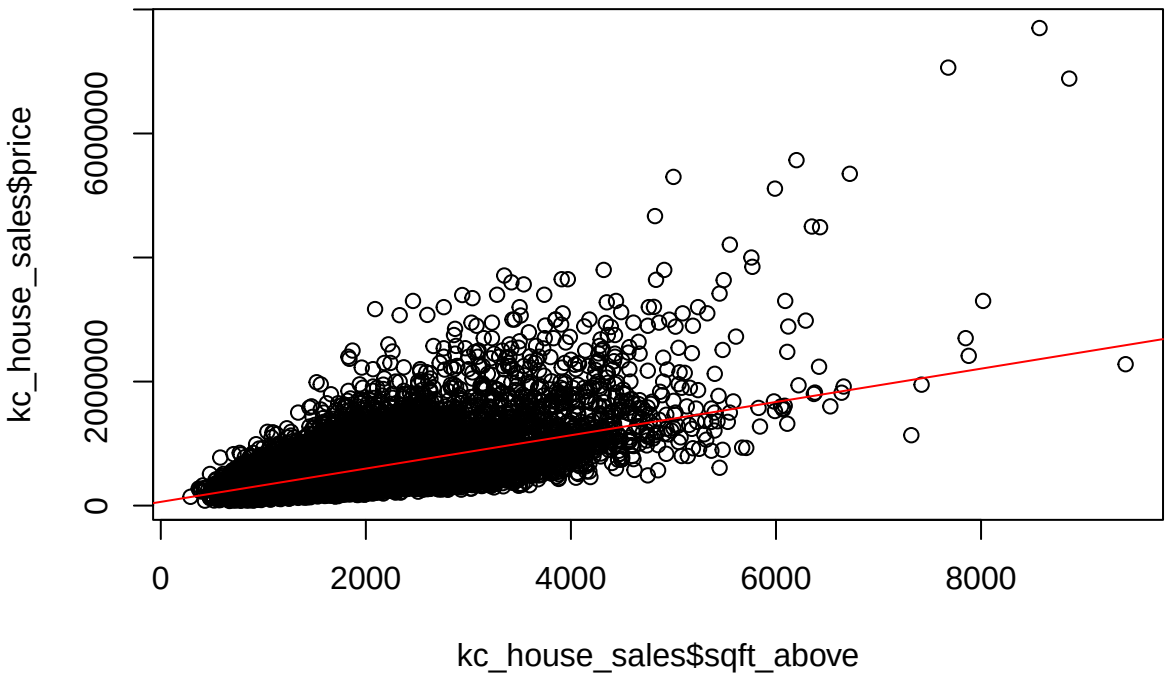


Figure C−11. sqft_above Distributi




```r
par(mfrow=c(1,1))
plot(kc_house_sales$sqft_above, kc_house_sales$price,  main = "Figure C-12. sqft_above Scatter Plot");
abline(lm(price ~ sqft_above, data = kc_house_sales), col = "red")
```

Figure C−12. sqft_above Scatter Plot



Per Figure C-10 and C-11, the median square footage above ground appears to be slightly less than 2000 sqft, with outliers present at the higher end of the range.

Per Figure C-12, there appears to be a linear relationship between square feet of living space and price, with price increasing as the square footage increases.

```r
# floors box plot, histogram, scatter plot vs. price
par(mfrow=c(1,2))
boxplot(kc_house_sales$floors, main = "Figure C-13. floors Box Plot")
hist(kc_house_sales$floors, main = "Figure C-14. floors Distribution")
```

Floors

Figure C−13. floors Box Plot

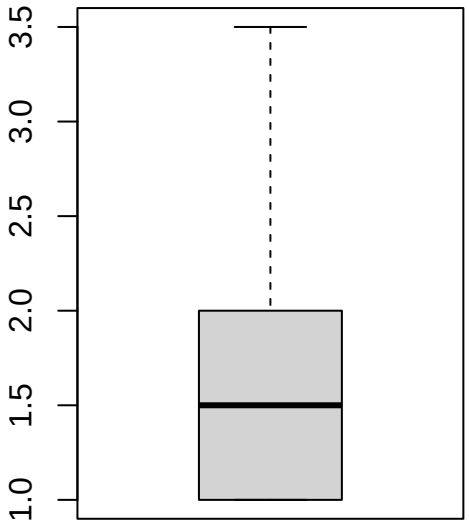
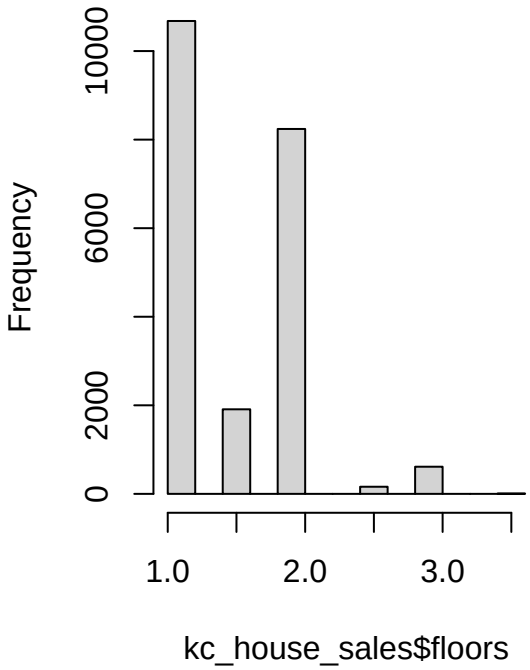
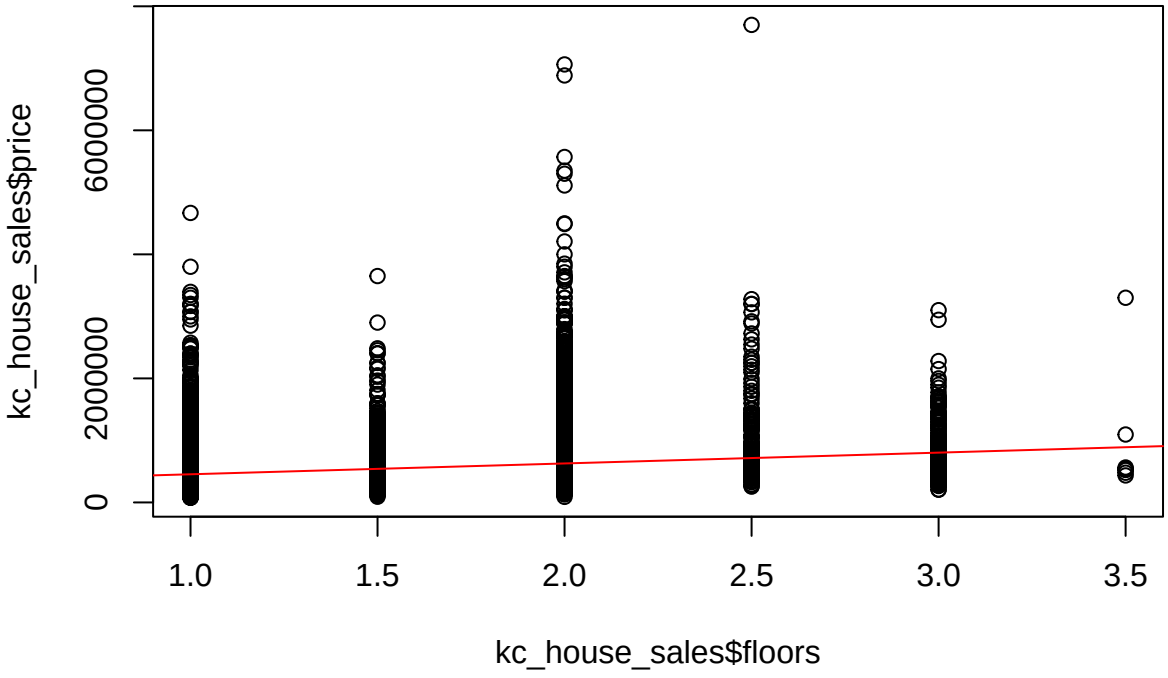


Figure C−14. floors Distribution




```r
par(mfrow=c(1,1))
plot(kc_house_sales$floors, kc_house_sales$price,  main = "Figure C-15. floors Scatter Plot");
abline(lm(price ~ floors, data = kc_house_sales), col = "red")
```

Figure C−15. floors Scatter Plot



Per Figures C-13, the median house has 1.5 floors, and there are no outliers in the data set. Although the median is 1.5 floors, looking at the distribution in Figure C-14, most houses have either 1 floor or 2 floors.

Looking at Figure C-15, there appears to be a nonlinear relationship between nnumber of floors and price. Houses with 1 floor or 2 floors appear to have a wider price range with higher maximum prices than houses with a fractional floor or those with 3 foors.

```r
# condition box plot, histogram, scatter plot vs. price
par(mfrow=c(1,2))
boxplot(kc_house_sales$condition, main = "Figure C-16. condition Box Plot")
hist(kc_house_sales$condition, main = "Figure C-17. condition Distribution")
```

Condition

Figure C−16. condition Box Plot

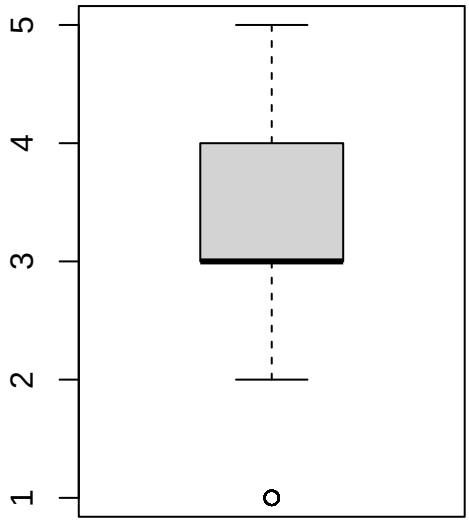
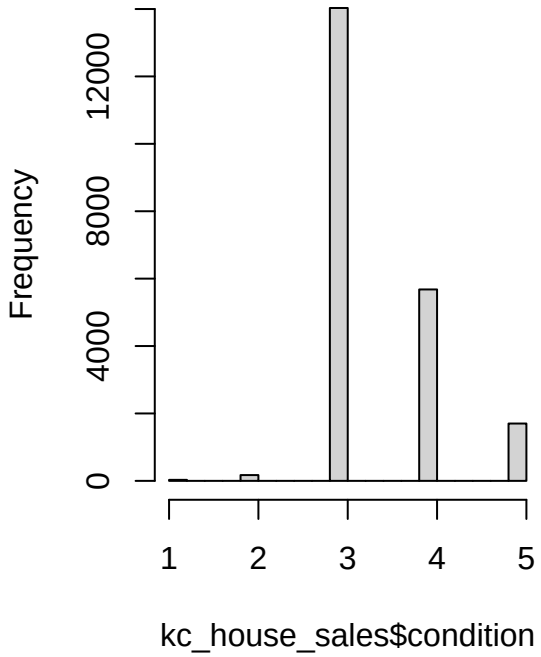
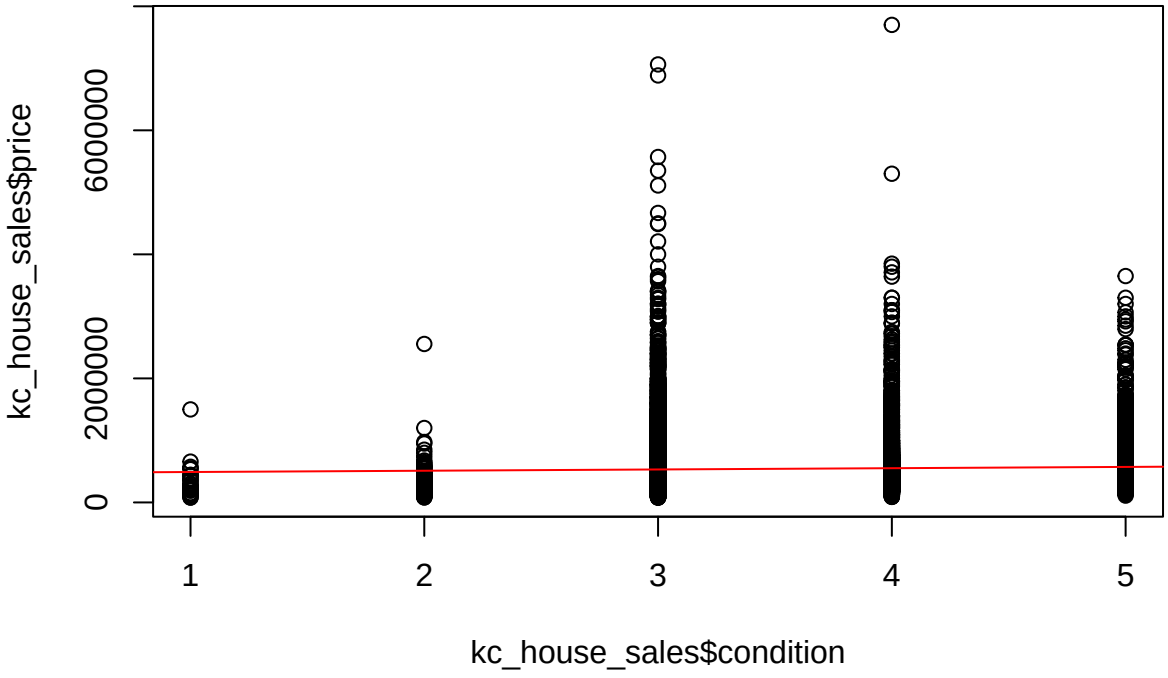


Figure C−17. condition Distributi




```r
par(mfrow=c(1,1))
plot(kc_house_sales$condition, kc_house_sales$price,  main = "Figure C-18. condition Scatter Plot");
abline(lm(price ~ condition, data = kc_house_sales), col = "red")
```

Figure C−18. condition Scatter Plot



The box plot in Figure C-16 indicates that the median house has a condition of 3, which is average, and houses with a condition of 1 are outliers. Figure C-17 with the distribution again shows that the vast majority of houses have a condition of 3. Interestingly, in the scatter plot in Figure C-18, it appears as though if the condition is at least 3, there is not much of an impact on housing price, but below 3 and the price decreases.

```r
# sqft_lot15 box plot, histogram, scatter plot vs. price
par(mfrow=c(1,2))
boxplot(kc_house_sales$sqft_lot15, main = "Figure C-19. sqft_lot15 Box Plot")
hist(kc_house_sales$sqft_lot15, main = "Figure C-20. sqft_lot15 Distribution")
```

Neighbor Square Footage - Lot

Figure C−19. sqft_lot15 Box Plo

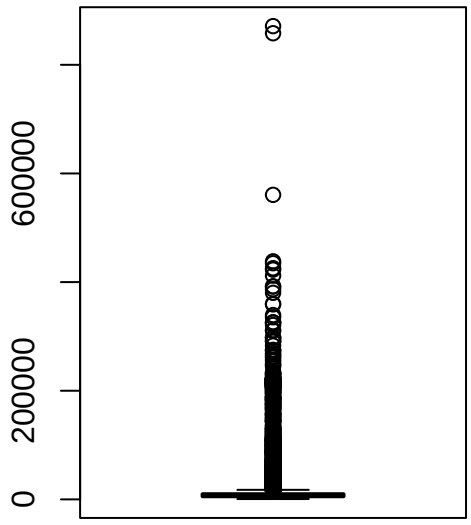
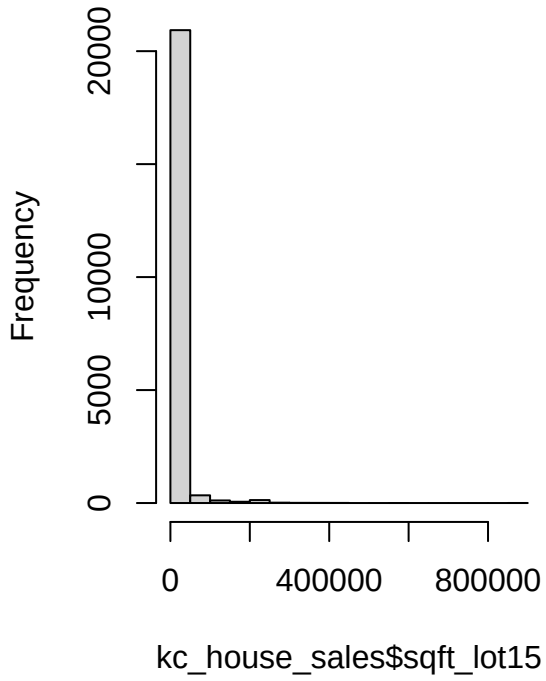
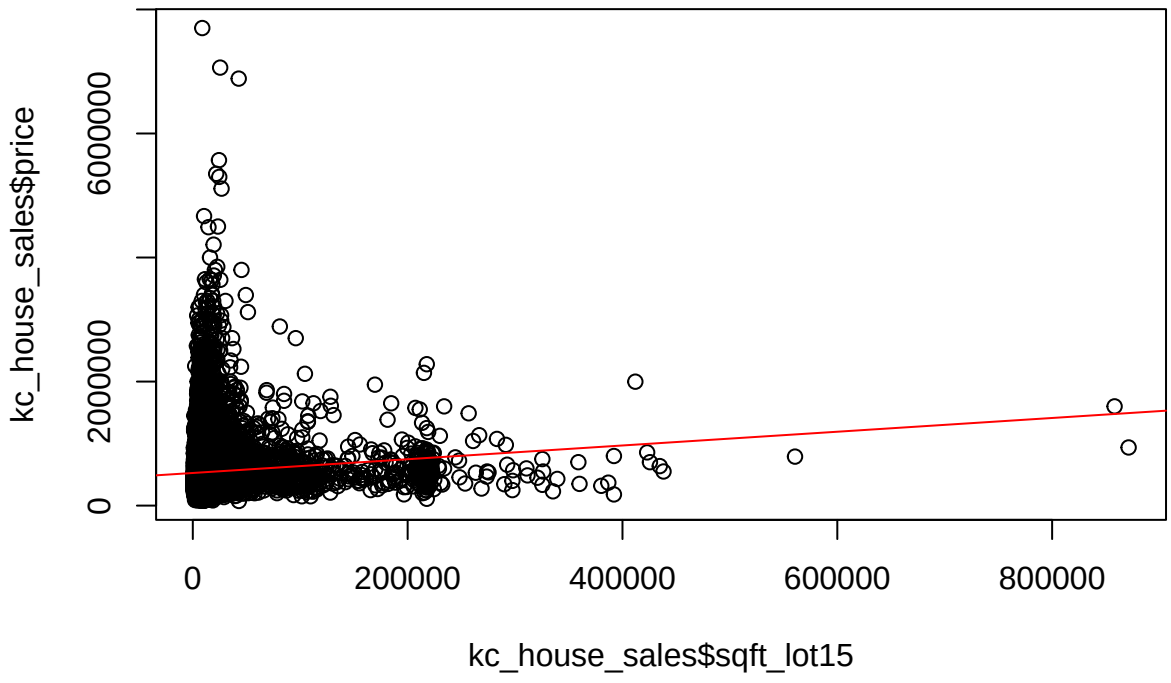


Figure C−20. sqft_lot15 Distributi




```r
par(mfrow=c(1,1))
plot(kc_house_sales$sqft_lot15, kc_house_sales$price,  main = "Figure C-21. sqft_lot15 Scatter Plot");
abline(lm(price ~ sqft_lot15, data = kc_house_sales), col = "red")
```

Figure C−21. sqft_lot15 Scatter Plot

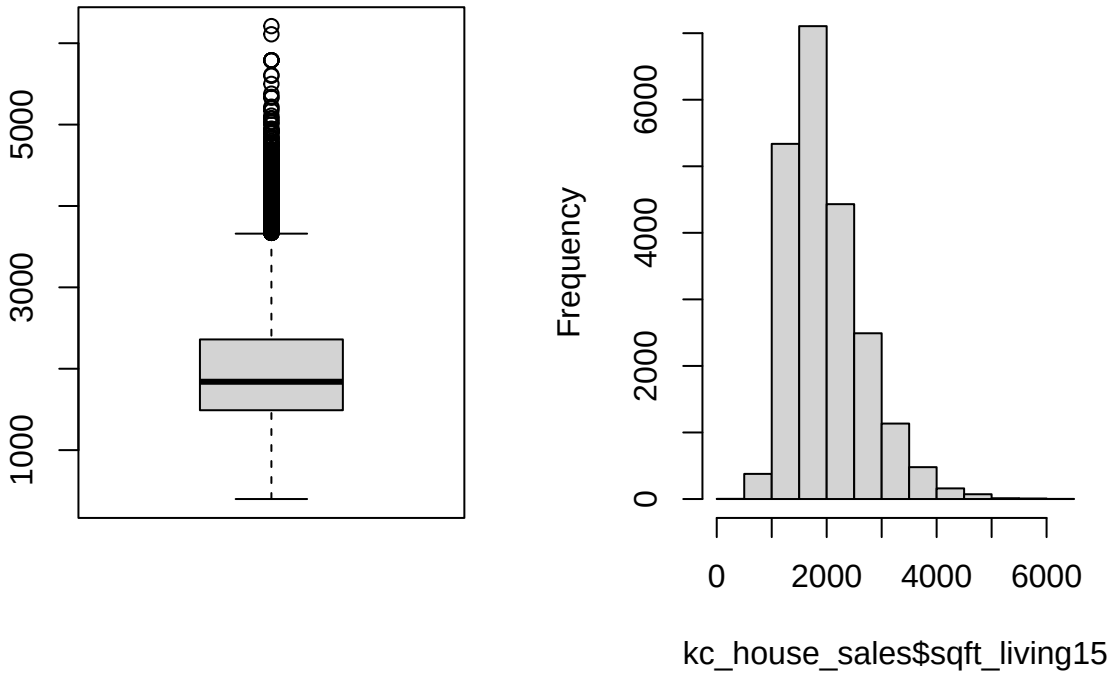


According to the box plot and histogram in Figure C-19 and C-20, the lot size of the 15 closest neighbors is generally grouped around 0, with outliers at the higher end of the range. It would be interesting to learn whether the outliers are indeed the correct lot sizes or whether there are typos or other errors in the data set. The scatter plot in Figure C-21 appears to show somewhat of a negative linear relationship between price and the square footage of lot size of the nearest 15 neighbors. Since King County encompasses Seattle, it is likely that most houses near the city do not have that much land surrounding them. Perhaps houses closer to the city center, which likely have smaller lot sizes, are worth more.

```r
# sqft_living15 box plot, histogram, scatter plot vs. price
par(mfrow=c(1,2))
boxplot(kc_house_sales$sqft_living15, main = "Figure C-22. sqft_living15 Box Plot")
hist(kc_house_sales$sqft_living15, main = "Figure C-23. sqft_living15 Distribution")
```

Neighbor Square Footage - Living

Figure C−22. sqft_living15 Box Pl **Figure C−23. sqft_living15 Distribu**




```r
par(mfrow=c(1,1))
plot(kc_house_sales$sqft_living15, kc_house_sales$price,  main = "Figure C-24. sqft_living15 Scatter Plot");
abline(lm(price ~ sqft_living15, data = kc_house_sales), col = "red")
```

Figure C−24. sqft_living15 Scatter Plot

According to the box plot and histogram in Figure C-22 and C-23, the square footage of living space among the 15 closest neighbors is grouped around slightly under 2000 square feet, with outliers at the higher end of the range. The scatter plot in Figure C-24 shows a linear relationship between the square footage of living space of the 15 nearest neighbors and price.