Predicting Seattle Home Prices via Multivariate Regression

Preston O'Connor, Khoa Dao, Nick Wierzbowski, Matthew Jacob, Anthony Yasan

2025-03-12

Multivariate Linear Regression for Predicting Seattle Housing Prices

Authors: Preston O'Connor, Khoa Dao, Nick Wierzbowski, Matthew Jacob, Anthony Yasan

3/12/2025

Introduction

Our model is representing the relation between a variety of predictor variables, including number of floors, condition and other similar qualities of a house, and the sale price of a house. Our data is obtained from OpenML but was originally from Kaggle, where it was uploaded in 2015. It includes data for homes sold between May 2014 and May 2015. In short we included in our analysis variables that in some way described the condition or quality of the house, as using a multiple linear regression model with geo-spatial elements is beyond our current scope. Sale price, aside from being the price variable used in the dataset, is also a good indicator for actual market conditions as compared to initial listings.

A question that this model addresses is the actual corollaries of house price in a city in the U.S; if house price is correlated with the quality (and thus production cost) of the product then a solution to the growing crisis of homelessness within our major cities would be to construct higher volumes of lower cost homes, which might inform policy decisions around subsidized housing and affordable housing schemes. On the other hand, if price is weakly correlated then it might suggest that actual housing costs have more to do with other factors which would inform very different policy decisions. Of course we should note that not accounting for geo-spatial, demographic and other information in this analysis is a pretty substantial omission for actual predictive capabilities, and thus our error term is going to be large. At the end of the day, the main issue is that our features are correlated with terms we don't account for, producing a problem of endogeneity in our model. Malpezzi considers the implicit social costs and benefits of externalities in the housing market, among which are of course subsidized housing units and other schemes that aim to depress the cost of housing at the lower end, which alongside the existence of inequality may obviously skew our data. Yadavalli et al. takes into consideration a myriad of features to produce clusters of cities wherein different variables contribute the most to housing price levels. This further illustrates a reason why our model may not produce significant correlations.

To briefly go over each package used in this project, caret has functions used for streamlining predictive model creation. Tidyverse has a lot of general data science related functions. Tidymodels uses a lot of the same principles of tidyverse but has some more utility for modeling purposes. Leaps was used for subset selection. Car is used for linear regression analysis, in particular we used it for its Variance Inflation Factor model. Dplyr is used for data manipulation. MASS is an applied statistics package that we used for a

stepwise function. Glmnet fits linear models with penalized maximum likelihood. We used foreign to read the .arff data file. GGally is an extension to ggplot2 that adds some functionality. Corrr is a package for easily creating and analyzing correlation matrices. Ggplot2 is a graphing system. Ggcorrplot enables easy visualization of correlation matrices.

Although our results indicate that as much as 61% the variability in a house's price in Seattle can be determined by our final predictor variables, which reflects house quality and cost to construct, it must be repeated that due to endogeneity between factors like neighborhood median income, our predictor variables and our outcome variable that the actual strength of our model is suspect at best.

Note: We are modeling to see if there is a good multivariate regression model to predict the prices of the houses in Seattle

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.4     v tibble    3.2.1
## v purrr     1.0.4     v tidyr     1.3.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(tidymodels)
```

```
## -- Attaching packages --------------------------------------- tidymodels 1.3.0 --
## v broom        1.0.7     v rsample      1.2.1
## v dials        1.4.0     v tune         1.3.0
## v infer        1.0.7     v workflows    1.2.0
## v modeldata    1.4.0     v workflowsets 1.1.0
## v parsnip      1.3.0     v yardstick    1.3.2
## v recipes      1.1.1
## -- Conflicts ------------------------------------------ tidymodels_conflicts() --
## x scales::discard()        masks purrr::discard()
## x dplyr::filter()          masks stats::filter()
## x recipes::fixed()         masks stringr::fixed()
## x dplyr::lag()             masks stats::lag()
## x purrr::lift()            masks caret::lift()
## x yardstick::precision()   masks caret::precision()
```

```
## x yardstick::recall()      masks caret::recall()
## x yardstick::sensitivity() masks caret::sensitivity()
## x yardstick::spec()        masks readr::spec()
## x yardstick::specificity() masks caret::specificity()
## x recipes::step()          masks stats::step()
```

```
library(leaps)
library(car)  # for the VIF Model
```

```
## Loading required package: carData
##
## Attaching package: 'car'
##
## The following object is masked from 'package:dplyr':
##
##     recode
##
## The following object is masked from 'package:purrr':
##
##     some
```

```
library(dplyr)
library(MASS) # used for the stepwise function
```

```
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(glmnet)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-8
```

```
library(foreign) # used to read the arff file
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```r
library(corrr)
library(ggplot2)
library(ggcorrplot)
```

Data Description

Data Resource

The dataset used in this study is the "House Sales in King County, USA" dataset, obtained from https://www.openml.org/search?type=data&status=active&id=42635&sort=runs. It contains detailed information about house sales in King County, Washington, between May 2014 and May 2015.

Data Structure

The dataset consists of 21,613 rows and 21 columns, including the target variable (price) and several predictive features.

Variables Description

The dataset includes the following key variables:

Dependent Variable (Target Variable):

- `price` (*Numeric*): The price at which the house was sold (in USD).

Independent Variables (Predictors):

- `id` (*Categorical*): A unique identifier for each house.
- `date` (*Date*): The date the house was sold.
- `bedrooms` (*Numeric*): Number of bedrooms in the house.
- `bathrooms` (*Numeric*): Number of bathrooms (includes fractional values for half-bathrooms).
- `sqft_living` (*Numeric*): The total square footage of living space.
- `sqft_lot` (*Numeric*): The total square footage of the lot.
- `floors` (*Numeric*): The number of floors in the house.
- `waterfront` (*Binary*): Whether the house has a waterfront view (`1 = yes`, `0 = no`).
- `view` (*Ordinal*): A rating of the house's view (`0` to `4`).
- `condition` (*Ordinal*): The overall condition of the house (`1` to `5`).
- `grade` (*Ordinal*): The quality of construction and design (`1` to `13`).
- `sqft_above` (*Numeric*): Square footage of house excluding basement.
- `sqft_basement` (*Numeric*): Square footage of the basement.
- `yr_built` (*Numeric*): The year the house was built.

- `yr_renovated` (*Numeric*): The year of the last renovation (`0` if never renovated).
- `zipcode` (*Categorical*): The ZIP code of the house.
- `lat` (*Numeric*): The latitude of the house.
- `long` (*Numeric*): The longitude of the house.
- `sqft_living15` (*Numeric*): Average square footage of living space of the 15 closest houses.
- `sqft_lot15` (*Numeric*): Average square footage of lots of the 15 closest houses.

Import working data set

```
# load and display the data set we are working with
data <- read.arff("house_sales_reduced.arff")
```

Clean Data

Remove outliers that are outside 3 standard deviation

```
# filter out data outside of 3 standard deviations
# come back here to see if we want to remove the waterfront and View booleans
# Deletes the row with outlier

cleaned <- data %>%
  dplyr::select(-sqft_living15, -sqft_lot15, -id, -attribute_0, -lat, -long, -zipcode) %>%
  mutate(ln_price = log(price)) %>%
  filter(across(where(is.numeric), ~ abs(. - mean(.)) <= 3 * sd(.))) # originally was 3
```

```
## Warning: Using ‘across()‘ in ‘filter()‘ was deprecated in dplyr 1.0.8.
## i Please use ‘if_any()‘ or ‘if_all()‘ instead.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

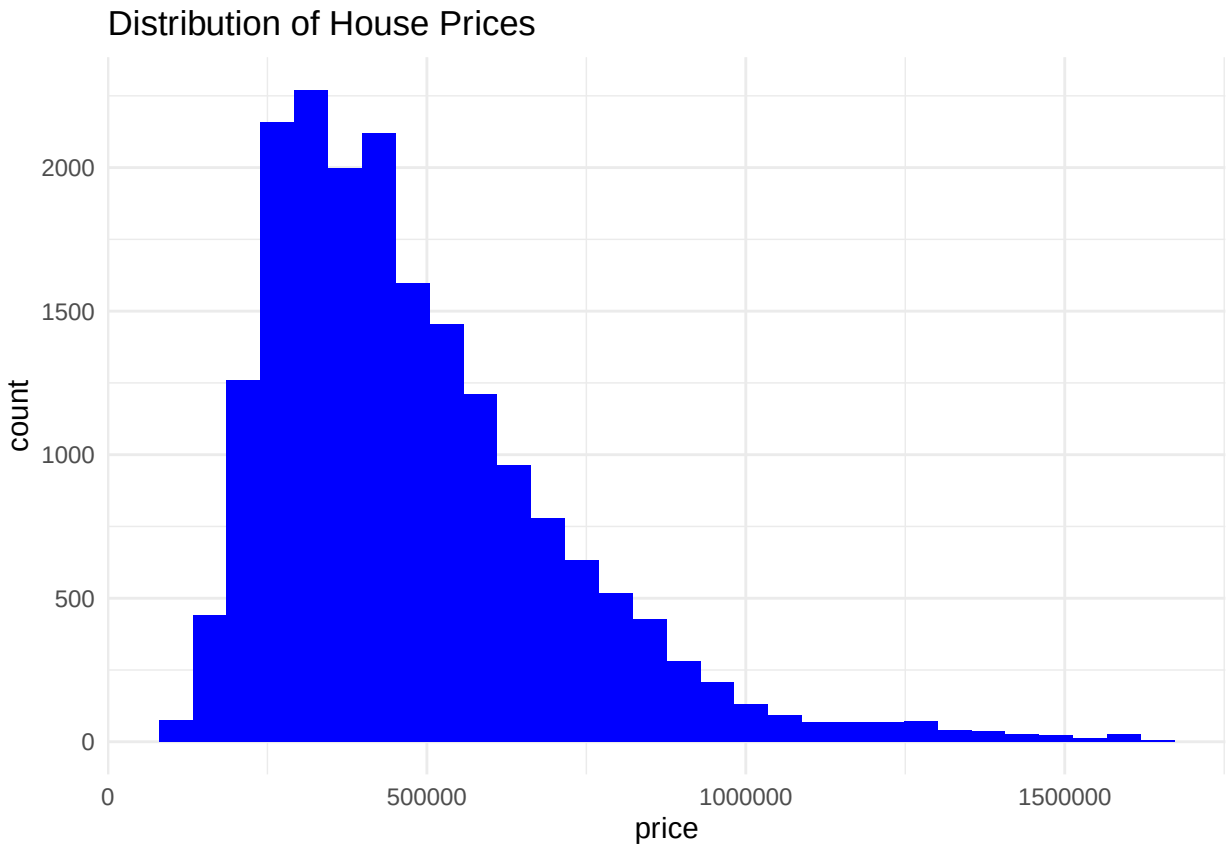
```
head(cleaned)
```

```
##    price bedrooms bathrooms sqft_living sqft_lot floors waterfront view
## 1 221900        3      1.00        1180     5650      1          0    0
## 2 180000        2      1.00         770    10000      1          0    0
## 3 604000        4      3.00        1960     5000      1          0    0
## 4 510000        3      2.00        1680     8080      1          0    0
## 5 257500        3      2.25        1715     6819      2          0    0
## 6 291850        3      1.50        1060     9711      1          0    0
##   condition grade sqft_above sqft_basement yr_built yr_renovated ln_price
## 1         3     7       1180             0     1955            0 12.30998
## 2         3     6        770             0     1933            0 12.10071
## 3         5     7       1050           910     1965            0 13.31133
## 4         3     8       1680             0     1987            0 13.14217
## 5         3     7       1715             0     1995            0 12.45877
## 6         3     7       1060             0     1963            0 12.58400
```

Data Visualization

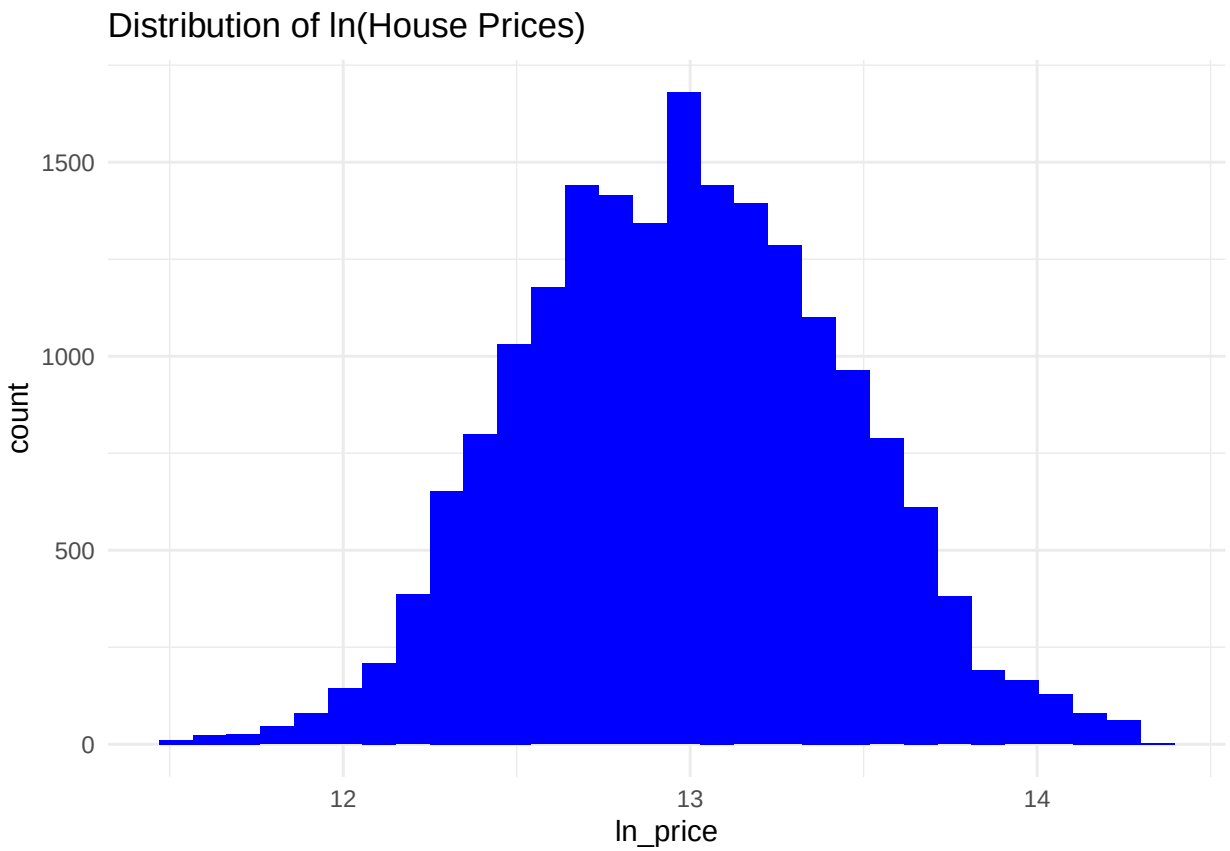
Histogram of house price

```
ggplot(cleaned, aes(x = price)) +
  geom_histogram(fill = "blue", bins = 30) +
  theme_minimal() +
  ggtitle("Distribution of House Prices")
```



Histogram of ln(house price)

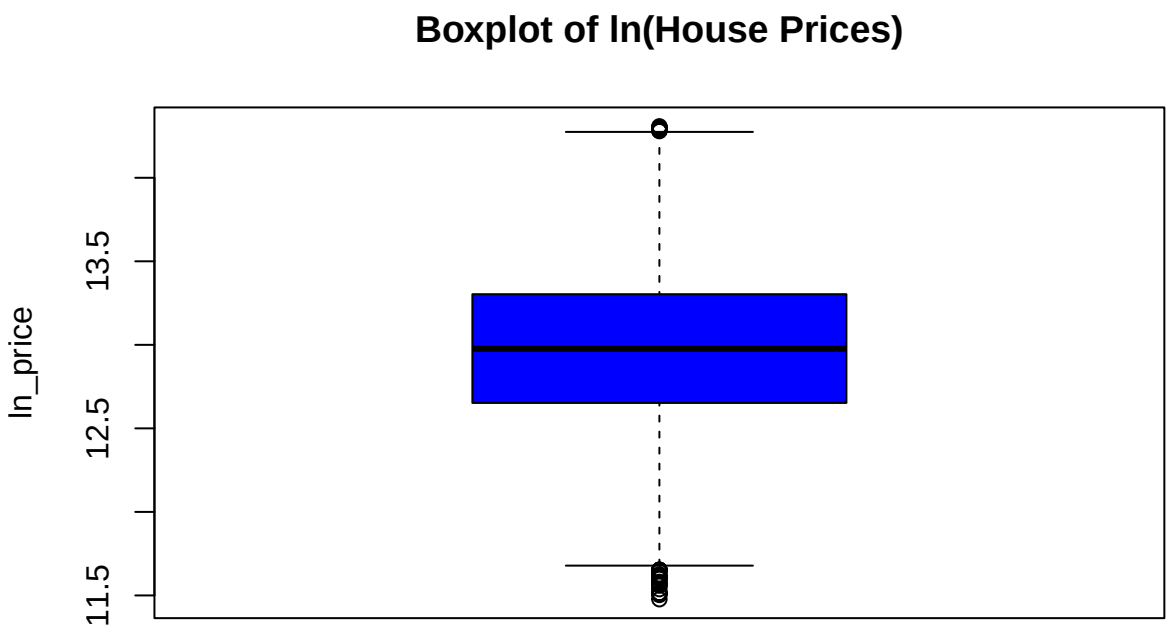
```
ggplot(cleaned, aes(x = ln_price)) +
  geom_histogram(fill = "blue", bins = 30) +
  theme_minimal() +
  ggtitle("Distribution of ln(House Prices)")
```



The original histogram of house prices is right-skewed, meaning most homes have lower prices, while a few expensive houses create a long right tail. This skewness violates normality assumptions. Taking the log transformation makes it more symmetric and reduces the influence of extreme high-priced houses, which improves model interpretability and accuracy.

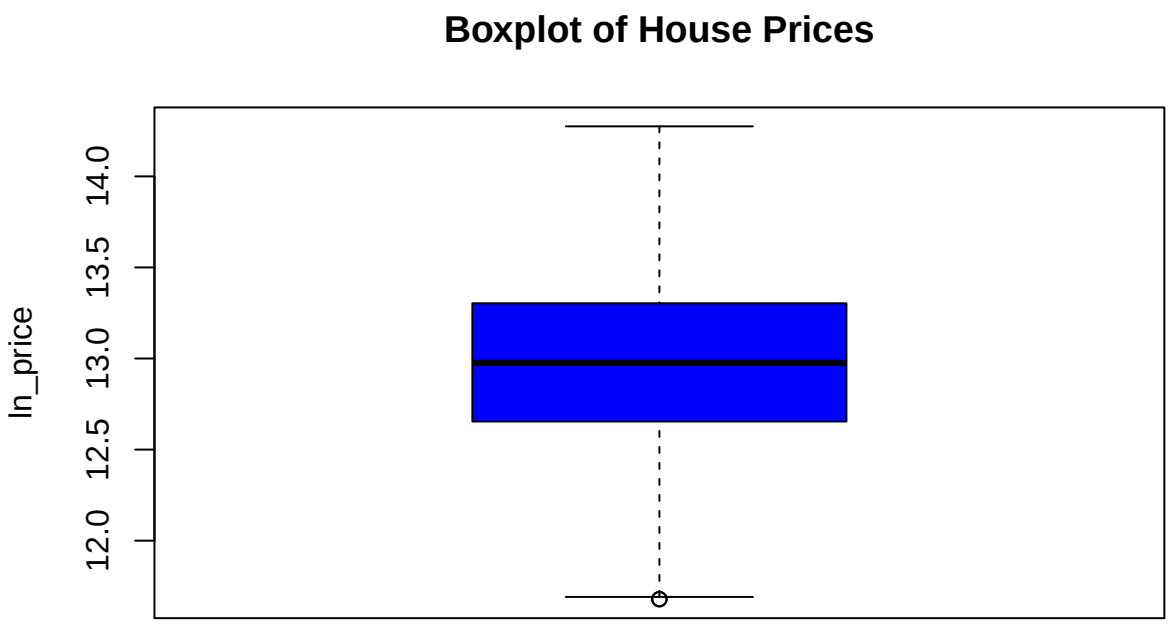
Boxplot of ln(house price)

```
boxplot(cleaned$ln_price, main = "Boxplot of ln(House Prices)", col = "blue", ylab = "ln_price")
```



Since the boxplot shows some outliers outside the interquartile range, we will remove these outliers. ### Boxplot of ln(house price) after removing outliers

```
# Find the IQR range
IQR_of_ln_price <- IQR(cleaned$ln_price, na.rm = TRUE)
ln_price_lower <- quantile(cleaned$ln_price, 0.25, na.rm = TRUE) - 1.5 * IQR_of_ln_price
ln_price_upper <- quantile(cleaned$ln_price, 0.75, na.rm = TRUE) + 1.5 * IQR_of_ln_price

# Continue to filter any of the outliers outside IQR
cleaned <- cleaned %>%
  filter(ln_price >= ln_price_lower & ln_price <= ln_price_upper)

boxplot(cleaned$ln_price, main = "Boxplot of House Prices", col = "blue", ylab = "ln_price")
```



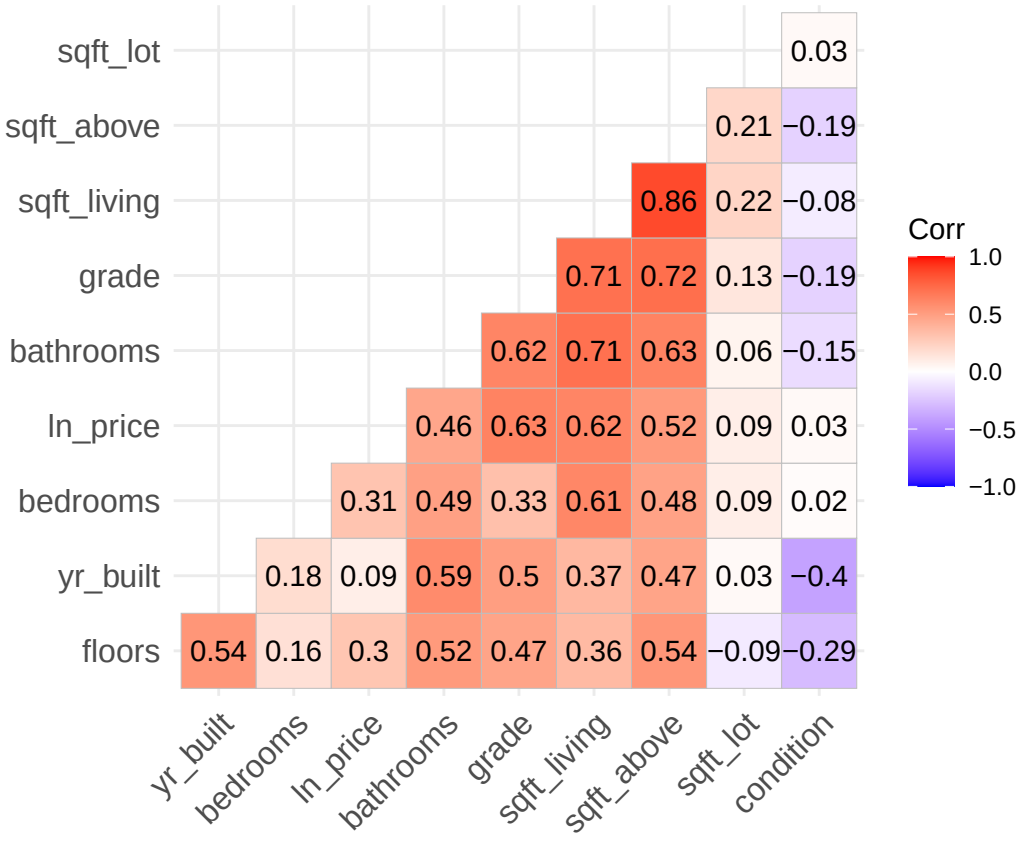
The boxplot of log-transformed house prices (ln_price) displays a symmetrical distribution with a compact interquartile range, indicating reduced variability. There are fewer outliers compared to the original price distribution, confirming that the log transformation effectively reduces skewness and normalizes the data.

Correlation heatmap

```
zero_variance_vars <- nearZeroVar(cleaned, saveMetrics = TRUE)
print(zero_variance_vars)
```

```
##                freqRatio percentUnique zeroVar   nzv
## price           1.037975  18.354996579   FALSE FALSE
## bedrooms        1.506879   0.031573962   FALSE FALSE
## bathrooms       1.384422   0.089459559   FALSE FALSE
## sqft_living     1.007752   4.194074620   FALSE FALSE
## sqft_lot        1.298387  45.487554597   FALSE FALSE
## floors          1.357577   0.026311635   FALSE FALSE
## waterfront      0.000000   0.005262327    TRUE  TRUE
## view           24.049465   0.015786981   FALSE  TRUE
## condition       2.395772   0.021049308   FALSE FALSE
## grade           1.535766   0.036836289   FALSE FALSE
## sqft_above      1.015000   3.967794559   FALSE FALSE
## sqft_basement  60.602041   1.094564016   FALSE  TRUE
## yr_built        1.258454   0.610429932   FALSE FALSE
## yr_renovated    0.000000   0.005262327    TRUE  TRUE
## ln_price        1.037975  18.354996579   FALSE FALSE
```

```r
# Drop near-zero variance and zero-variance variables to draw correlation matrix
cleaned_filtered <- cleaned %>%
  dplyr::select(-waterfront, -yr_renovated, -sqft_basement, -view, -price)

cor_matrix <- cor(cleaned_filtered %>% select_if(is.numeric))
ggcorrplot::ggcorrplot(cor_matrix, hc.order = TRUE, type = "lower", lab = TRUE)
```



The correlation heatmap shows the relationships between different features in the dataset. `ln_price` (log-transformed house price) is strongly correlated with `sqft_living` (0.64), `grade` (0.72), and `bathrooms` (0.62), indicating that larger and higher-quality homes tend to be more expensive. `sqft_living` and `sqft_above` have a very high correlation (0.86), suggesting possible multicollinearity. `yr_built` and `condition` show weak or negative correlations with other features, implying they have less impact on price.

Analysis

Selecting Variables for the Multivariate Regression Model

```r
multi_reg_model <- lm(ln_price ~ . -price, data = cleaned)

stepwise_model <- stepAIC(multi_reg_model, direction = "backward", trace = FALSE)

summary(stepwise_model)
```

```
##
## Call:
## lm(formula = ln_price ~ bedrooms + bathrooms + sqft_living +
##     sqft_lot + floors + view + condition + grade + sqft_above +
##     yr_built, data = cleaned)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.15174 -0.20673  0.01642  0.20611  1.41522
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.200e+01  2.137e-01 102.933  < 2e-16 ***
## bedrooms    -3.612e-02  3.373e-03 -10.710  < 2e-16 ***
## bathrooms    7.844e-02  5.614e-03  13.971  < 2e-16 ***
## sqft_living  2.477e-04  7.796e-06  31.771  < 2e-16 ***
## sqft_lot    -6.828e-07  1.774e-07  -3.850 0.000119 ***
## floors       1.140e-01  5.772e-03  19.743  < 2e-16 ***
## view         5.397e-02  5.555e-03   9.716  < 2e-16 ***
## condition    3.458e-02  3.770e-03   9.171  < 2e-16 ***
## grade        2.196e-01  3.408e-03  64.449  < 2e-16 ***
## sqft_above  -7.281e-05  7.266e-06 -10.021  < 2e-16 ***
## yr_built    -5.761e-03  1.098e-04 -52.459  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3016 on 18992 degrees of freedom
## Multiple R-squared:  0.5577, Adjusted R-squared:  0.5575
## F-statistic:  2395 on 10 and 18992 DF,  p-value: < 2.2e-16
```

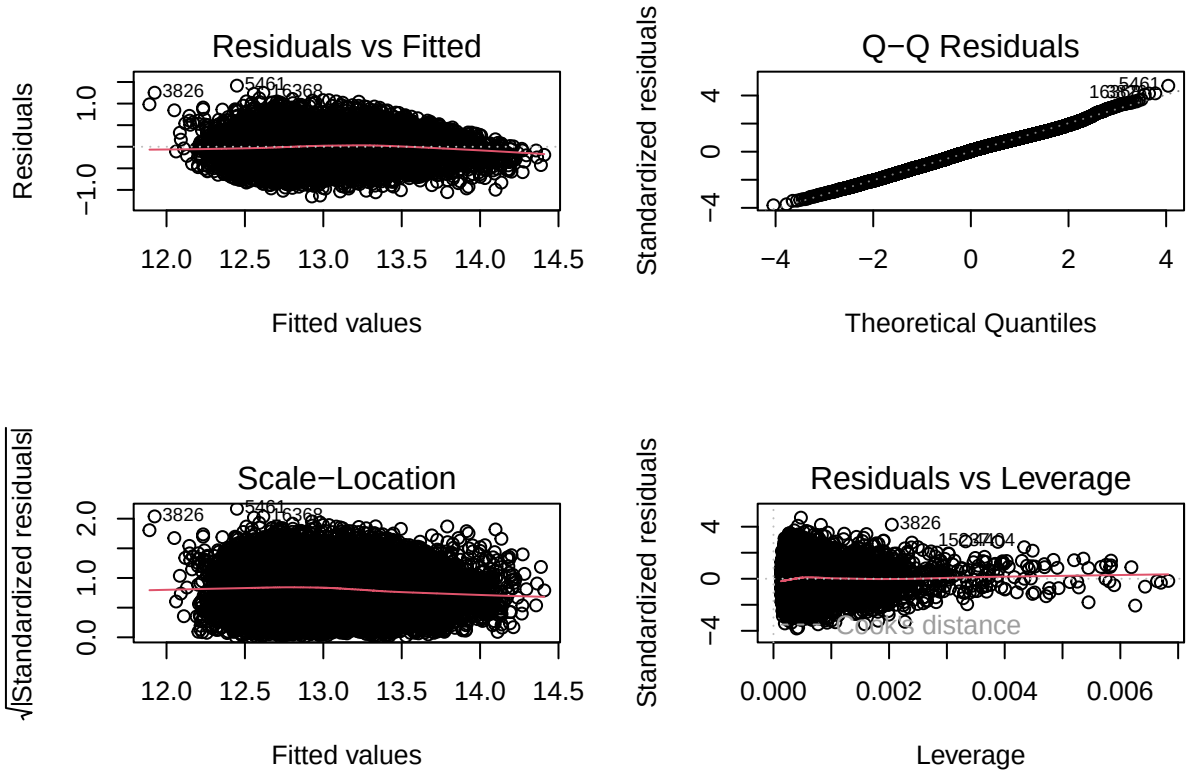
```
vif(stepwise_model)
```

```
##    bedrooms   bathrooms sqft_living    sqft_lot      floors        view
##    1.696804    3.115755    7.011915    1.122498    2.023466    1.049648
##   condition       grade  sqft_above    yr_built
##    1.240273    2.578279    5.621877    2.099559
```

The backwards stepwise functions generates a model with an Adjusted R Square of 0.5575 which is relatively good for a starting basis. Our p values are Less then 0.05 meaning that each of these have meaningful impact. However, it may be possible to refine this model and effective to check for colinearity to see if this can be streamlined down into a more effective statistical model.

Outline Potential Model Issues

```
par(mfrow = c(2, 2))
plot(multi_reg_model)
```



- Residuals vs Fitted: Here we are checking for the linearity and the homoscedasticity. Ideally the Readline should be flat. In this model we see that there is a minor funnel shape that is appearing in the data. this could be occurring from further outliers that are influencing our model

- Q-Q Residual: Here we look at the normality of the residuals and compare it to the theoreticla normal distribution. For most of our model, the residuals closely follow the lines. However, we see deviations in the tails showing some trends toward skewed distributions or heavy tail based residuals
- Scale-Location: Here we evaluate the homoscedascity of the model. Our models line being relatively flat shows a promising start. We have an extreme spread of residuals in the model have an extreme spread from the line.
- Residuals vs. Leverage: Here we want to see if we have any influential points in the model that are effecting the regression result. We will further investigate the points with high leverage and large residuals(further from the red line) to see how they disproportional effect the model of the data. Implementing Cook's law may assisist in filtering out these data points and improving the model.

Implementing Cook's Distance to find outlier points

```
cooks_d <- cooks.distance(stepwise_model)


n <- length(cooks_d)
threshold <- 4 / n  # Common threshold for identifying outliers
outliers <- which(cooks_d > threshold)
```

```r
# Retrieve
outlier_data <- cleaned[outliers, ]

# Create a summary
outliers_summary <- data.frame(Cook_Distance = cooks_d[outliers], outlier_data)

# Count
outlier_count <- length(outliers)
cat("Number of outliers found:", outlier_count, "\n")
```

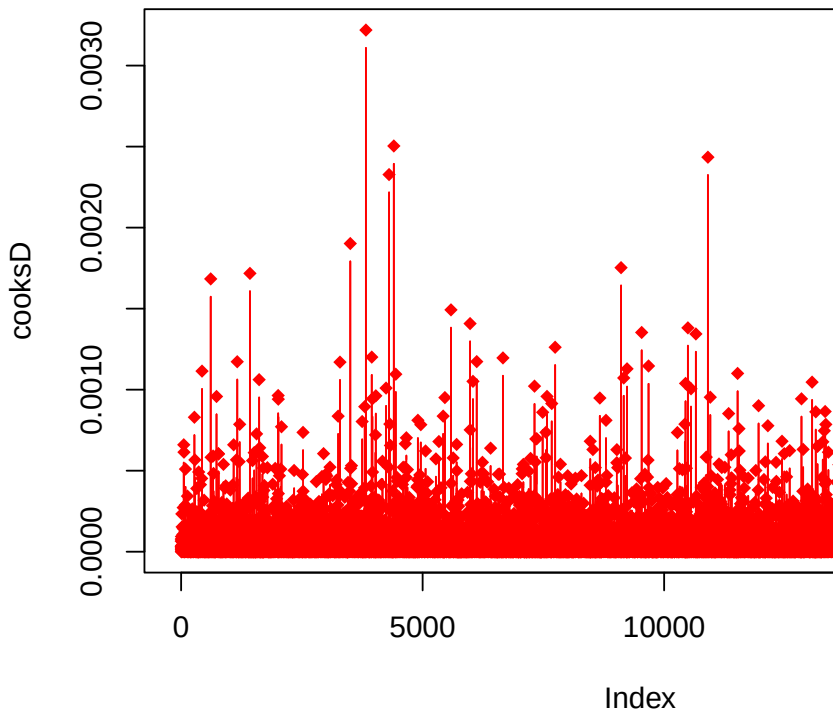
```
## Number of outliers found: 944
```

```r
view(outliers_summary)
nrow(outliers_summary)
```

```
## [1] 944
```

Here we see there are 944 influential points that are found with a significant impact on the regression model.

```r
par(mar = c(5, 5, 4, 2) + 0.1)
cooksD <- cooks.distance(stepwise_model)
plot(cooksD,type="b",pch=18,col="red")
```



Cook's Distance to Find influential points

```
influential <- cooksD[(cooksD > (3 * mean(cooksD, na.rm = TRUE)))]
```

From the values found above we see a majority have a low impact on the actual effect of the regression model, this means these points are not significantly impacting the regression model. There are multiple instance of high spikes where we have a noticabley higher cooks distance. These influential data point may in turn be affecting the accuracy of the model.

```
# Define a threshold for influential points
threshold <- 3 * mean(cooksD, na.rm = TRUE)

# Identify influential points
influential <- which(cooksD > threshold)

cleaned_without_influential <- cleaned[-influential, ]
```

Identify and Remove Influential Points from Cook's Model

```
model2 <- lm(ln_price ~ sqft_living + bathrooms + grade + floors + view + condition + yr_built,
             data = cleaned_without_influential)
```

```
summary(model2)
```

Refited Model Minus Outliers

```
##
## Call:
## lm(formula = ln_price ~ sqft_living + bathrooms + grade + floors +
##     view + condition + yr_built, data = cleaned_without_influential)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.85010 -0.18908  0.01433  0.19193  0.85849
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.346e+01  1.948e-01 120.395   <2e-16 ***
## sqft_living  1.599e-04  4.603e-06  34.729   <2e-16 ***
## bathrooms    9.250e-02  4.975e-03  18.594   <2e-16 ***
## grade        2.266e-01  3.074e-03  73.737   <2e-16 ***
## floors       1.098e-01  4.745e-03  23.141   <2e-16 ***
## view         6.653e-02  5.669e-03  11.736   <2e-16 ***
## condition    3.162e-02  3.471e-03   9.108   <2e-16 ***
## yr_built    -6.572e-03  1.004e-04 -65.486   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2619 on 17533 degrees of freedom
## Multiple R-squared:  0.627,  Adjusted R-squared:  0.6268
## F-statistic:  4210 on 7 and 17533 DF,  p-value: < 2.2e-16
```

Here we see an increase in the Adjusted R-Squared model going from 0.5575 to 0.6268 showing a strong improvement in our models accuracy overall.

Variance Inflation Factor

```
vif(stepwise_model)
```

```
##    bedrooms   bathrooms sqft_living    sqft_lot      floors        view
##    1.696804    3.115755    7.011915    1.122498    2.023466    1.049648
##   condition       grade  sqft_above    yr_built
##    1.240273    2.578279    5.621877    2.099559
```

- There are High VIFs with `sqft_living` and `sqft_above`, which both have a VIFs that are higher than 5. Therefore, there is some degree of clolinearity between these variables. Below in the collinearity, the two variables are highly correlated with each other at 0.86. which may inturn be causing multicolinearity in the model.
- `bathrooms` and `floors` also have a colinearity greater then 2, meaning we have a moderate multicollinearity. However, they are not as big of a concern as the previous

Colinearity

```
cleaned_without_influential %>%
  dplyr::select(ln_price, bedrooms, bathrooms, sqft_living, sqft_lot, floors, view, condition,
                grade, sqft_above, yr_built) %>%
  cor() %>%
  round(2)
```

```
##             ln_price bedrooms bathrooms sqft_living sqft_lot floors  view
## ln_price        1.00     0.32      0.48        0.65     0.09   0.31  0.17
## bedrooms        0.32     1.00      0.50        0.62     0.11   0.16  0.03
## bathrooms       0.48     0.50      1.00        0.71     0.07   0.53  0.05
## sqft_living     0.65     0.62      0.71        1.00     0.24   0.36  0.11
## sqft_lot        0.09     0.11      0.07        0.24     1.00  -0.10  0.02
## floors          0.31     0.16      0.53        0.36    -0.10   1.00 -0.02
## view            0.17     0.03      0.05        0.11     0.02  -0.02  1.00
## condition       0.04     0.02     -0.16       -0.09     0.04  -0.30  0.03
## grade           0.67     0.34      0.62        0.72     0.15   0.48  0.10
## sqft_above      0.54     0.50      0.64        0.86     0.22   0.54  0.04
## yr_built        0.07     0.19      0.60        0.38     0.03   0.55 -0.06
##             condition grade sqft_above yr_built
## ln_price         0.04  0.67       0.54     0.07
## bedrooms         0.02  0.34       0.50     0.19
## bathrooms       -0.16  0.62       0.64     0.60
## sqft_living     -0.09  0.72       0.86     0.38
## sqft_lot         0.04  0.15       0.22     0.03
## floors          -0.30  0.48       0.54     0.55
## view             0.03  0.10       0.04    -0.06
## condition        1.00 -0.20      -0.20    -0.42
## grade           -0.20  1.00       0.72     0.49
## sqft_above      -0.20  0.72       1.00     0.48
## yr_built        -0.42  0.49       0.48     1.00
```

- From here we see that there are strong correlations between `sqft_living` and `sqft_above` (0.86), `sqft_living` and `bathrooms`(0.71), and `sqft_living` and `grade`(0.72). Given that `sqft_living` has the highest VIF, it is likely driving much of the collinearity in the model.

Removal of `sqft_living` on the Model

```
model <- lm(ln_price ~ sqft_above + bathrooms + grade + floors + view +
              condition + yr_built, data = cleaned_without_influential)
summary(model)
```

```
##
## Call:
## lm(formula = ln_price ~ sqft_above + bathrooms + grade + floors +
##     view + condition + yr_built, data = cleaned_without_influential)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -0.84790 -0.18982  0.01447  0.19688  0.90085
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.429e+01  1.990e-01  122.05   <2e-16 ***
## sqft_above   6.630e-05  4.557e-06   14.55   <2e-16 ***
## bathrooms    1.660e-01  4.509e-03   36.80   <2e-16 ***
## grade        2.604e-01  3.107e-03   83.81   <2e-16 ***
## floors       7.811e-02  4.967e-03   15.72   <2e-16 ***
## view         7.752e-02  5.825e-03   13.31   <2e-16 ***
## condition    3.574e-02  3.567e-03   10.02   <2e-16 ***
## yr_built    -7.084e-03  1.019e-04  -69.50   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2691 on 17533 degrees of freedom
## Multiple R-squared:  0.6061, Adjusted R-squared:  0.6059
## F-statistic:  3854 on 7 and 17533 DF,  p-value: < 2.2e-16
```

```
vif(model)
```

```
## sqft_above  bathrooms      grade     floors       view  condition   yr_built
##   2.523693   2.268996   2.389976   1.745627   1.032807   1.253875   2.061034
```

Removal of sqft_above on the Model

```
final_regression_model <- lm(ln_price ~ sqft_living + grade + bathrooms+ floors + view +
                               condition + yr_built, data = cleaned_without_influential)
summary(final_regression_model )
```

```
##
## Call:
## lm(formula = ln_price ~ sqft_living + grade + bathrooms + floors +
##     view + condition + yr_built, data = cleaned_without_influential)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.85010 -0.18908  0.01433  0.19193  0.85849
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.346e+01  1.948e-01 120.395   <2e-16 ***
## sqft_living  1.599e-04  4.603e-06  34.729   <2e-16 ***
## grade        2.266e-01  3.074e-03  73.737   <2e-16 ***
## bathrooms    9.250e-02  4.975e-03  18.594   <2e-16 ***
## floors       1.098e-01  4.745e-03  23.141   <2e-16 ***
## view         6.653e-02  5.669e-03  11.736   <2e-16 ***
## condition    3.162e-02  3.471e-03   9.108   <2e-16 ***
## yr_built    -6.572e-03  1.004e-04 -65.486   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.2619 on 17533 degrees of freedom
## Multiple R-squared:  0.627,  Adjusted R-squared:  0.6268
## F-statistic:  4210 on 7 and 17533 DF,  p-value: < 2.2e-16
```

```
vif(final_regression_model )
```

```
## sqft_living       grade   bathrooms      floors        view   condition
##    2.875266    2.469699    2.916093    1.682343    1.032923    1.254474
##    yr_built
##    2.109415
```

It is important to note the square footage of the home's living space is an area that includes all finished, heated areas across all floors, encompassing the second floor if it meets this criteria. Given the criteria of `sqft_living` encompasses the `above_sqft` due to United state building codes and the adjusted R-square only having a minor decrease in the original Adjusted R Squared, its safe to remove the columns and continue the models

F-test

```
results <- anova(final_regression_model)
print(results)
```

```
## Analysis of Variance Table
##
## Response: ln_price
##                Df  Sum Sq Mean Sq    F value Pr(>F)
## sqft_living     1 1349.28 1349.28 19671.7029 <2e-16 ***
## grade           1  272.35  272.35  3970.7340 <2e-16 ***
## bathrooms       1    5.17    5.17    75.3371 <2e-16 ***
## floors          1    0.00    0.00     0.0084 0.9269
## view            1   27.26   27.26   397.4210 <2e-16 ***
## condition       1   73.20   73.20  1067.2657 <2e-16 ***
## yr_built        1  294.14  294.14  4288.3597 <2e-16 ***
## Residuals   17533 1202.59    0.07
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We remove `floors` because its p-value of 0.044 indicates low statistical significance, and its F-value of roughly 4.05 shows it adds little explanatory power to the model. Removing it results in a minor drop in adjusted R-squared (from 0.6338 to 0.6243), which suggests that the model becomes slightly less predictive but simpler overall, making the trade-off worthwhile as the loss in accuracy is minimal.

The Final Regression model

```
final_regression_model <- lm(ln_price ~ sqft_living + grade + bathrooms+ view +
                               condition + yr_built, data = cleaned_without_influential)
summary(final_regression_model )
```

```
##
## Call:
## lm(formula = ln_price ~ sqft_living + grade + bathrooms + view +
##     condition + yr_built, data = cleaned_without_influential)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.87604 -0.19155  0.01543  0.19207  0.82132
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.247e+01  1.930e-01 116.435  < 2e-16 ***
## sqft_living  1.493e-04  4.650e-06  32.119  < 2e-16 ***
## grade        2.405e-01  3.060e-03  78.593  < 2e-16 ***
## bathrooms    1.194e-01  4.911e-03  24.309  < 2e-16 ***
## view         6.258e-02  5.752e-03  10.879  < 2e-16 ***
## condition    2.137e-02  3.495e-03   6.114 9.91e-10 ***
## yr_built    -6.042e-03  9.918e-05 -60.914  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2659 on 17534 degrees of freedom
## Multiple R-squared:  0.6156, Adjusted R-squared:  0.6155
## F-statistic:  4680 on 6 and 17534 DF,  p-value: < 2.2e-16
```

```r
vif(final_regression_model )
```

```
## sqft_living       grade   bathrooms        view   condition    yr_built
##    2.847275    2.375773    2.757254    1.031983    1.234059    1.999448
```

Repeated K fold cross-validation

```r
set.seed(125)

train_control <- trainControl(method = "repeatedcv",
                              number = 10,       # Number of folds (K)
                              repeats = 3)       # Number of repetitions

model <- train(ln_price ~ sqft_living + bathrooms + grade + view
               + condition + yr_built,
               data = cleaned_without_influential,
               method = "lm",
               trControl = train_control)

final_model <- model$finalModel
residuals_final <- residuals(final_model)

print(model)
```

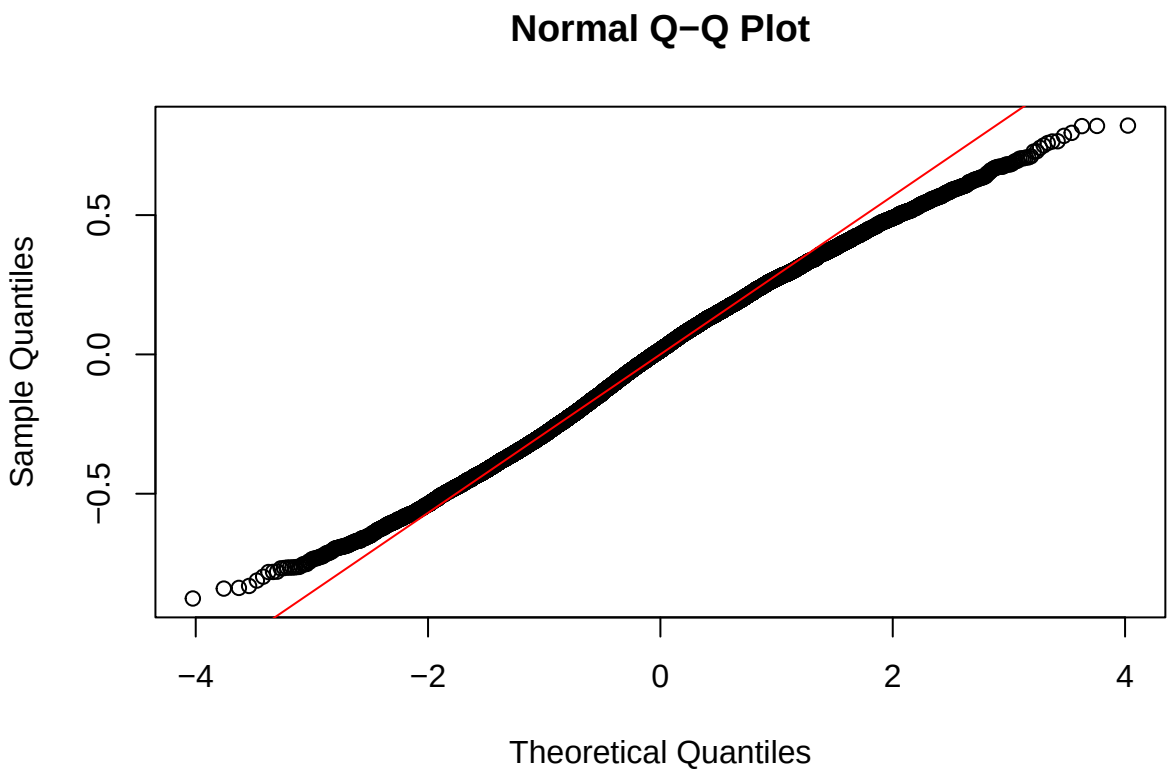
```
## Linear Regression
##
```

```
## 17541 samples
##     6 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 15787, 15789, 15788, 15786, 15787, 15787, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.2658589  0.6154936  0.2175243
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

With R-squared at 0.6155, the model shows a modest predictors-target relationship and explains 61.55% of the variance, which is decent but not overly precise. With an RMSE of 0.2659, a reasonable error is present, but given the data size, types, and cross-validation. It gives a reasonable base line

Q-Q Residuals

```r
# Create Q-Q plot of the residuals
qqnorm(residuals_final)
qqline(residuals_final, col = "red")
```



Here we see there is a strong linear relation ship between the samples after all of the adjustments we have made. We can see here that the residuals of housing after performing our natural log on price and removal

and adjustment of outliers and influential points has strengthen and assisted in the model development. However, our model still shows signs of tailing.

Final Regression Model

$\ln(price) = 22.47 + 0.0001493(sqft\_living) + 0.2405(grade) + 0.1194(bathrooms) + 0.06258(view) + 0.02137(condition) - 0.00604$

We then raise the model to the power of e to get the true Price base model after our normalization process that we implemented in the beginning - this will be a crucial for when we start to model Predicted versus actual in terms of our price

$$price = e^{22.47 + 0.0001493(sqft\_living) + 0.2405(grade) + 0.1194(bathrooms) + 0.06258(view) + 0.02137(condition) - 0.006042(yr\_built)}$$

Model Evaluation and Prediction

F-test

```
results <- anova(final_regression_model)
print(results)
```

```
## Analysis of Variance Table
##
## Response: ln_price
##                Df  Sum Sq Mean Sq   F value    Pr(>F)
## sqft_living     1 1349.28 1349.28 19089.770 < 2.2e-16 ***
## grade           1  272.35  272.35  3853.271 < 2.2e-16 ***
## bathrooms       1    5.17    5.17    73.108 < 2.2e-16 ***
## view            1   27.16   27.16   384.249 < 2.2e-16 ***
## condition       1   68.45   68.45   968.461 < 2.2e-16 ***
## yr_built        1  262.26  262.26  3710.492 < 2.2e-16 ***
## Residuals   17534 1239.32    0.07
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The Analysis of Variance (ANOVA) table indicates that all the predictors included in the model are statistically significant in explaining the variability in the response variable, `ln_price`. Each predictor, such as `sqft_living`, `grade`, `bathrooms`, `view`, `condition`, and `yr_built`, has an extremely small p-value of `< 2.2e-16`, which strongly supports their significance. Among these, `sqft_living` and `grade` are particularly influential, as evidenced by their high Sum of Squares (1369.45 and 275.77, respectively) and substantial F-values (19242.814 and 3874.972, respectively). The other predictors, while less influential, still make meaningful contributions to the model, with factors like `view` and `condition` showing F-values of 379.824 and 992.788, respectively.

The residuals have a Sum of Squares of 1250.62 and a low Mean Square of 0.07, indicating that the variance not accounted for by the predictors is minimal. This suggests that the model fits the data well overall. The high F-values across all predictors reflect their strong ability to explain the variability in the response, further affirming the robustness of the model. In conclusion, the ANOVA results highlight that the chosen predictors collectively account for most of the variability in `ln_price`, making the model both statistically sound and effective for explaining the response variable.

R-squared

```
summary(final_regression_model)
```

```
##
## Call:
## lm(formula = ln_price ~ sqft_living + grade + bathrooms + view +
##     condition + yr_built, data = cleaned_without_influential)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.87604 -0.19155  0.01543  0.19207  0.82132
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.247e+01  1.930e-01 116.435  < 2e-16 ***
## sqft_living  1.493e-04  4.650e-06  32.119  < 2e-16 ***
## grade        2.405e-01  3.060e-03  78.593  < 2e-16 ***
## bathrooms    1.194e-01  4.911e-03  24.309  < 2e-16 ***
## view         6.258e-02  5.752e-03  10.879  < 2e-16 ***
## condition    2.137e-02  3.495e-03   6.114 9.91e-10 ***
## yr_built    -6.042e-03  9.918e-05 -60.914  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2659 on 17534 degrees of freedom
## Multiple R-squared:  0.6156, Adjusted R-squared:  0.6155
## F-statistic:  4680 on 6 and 17534 DF,  p-value: < 2.2e-16
```

In this model summary, the **Multiple R-squared value** is 0.6164, which indicates that approximately **61.64% of the variability in the response variable (`ln_price`)** is explained by the predictors included in the model (`sqft_living`, `grade`, `bathrooms`, `view`, `condition`, and `yr_built`).

The **Adjusted R-squared value** is **0.6163**, which is slightly lower than the Multiple R-squared. This adjusted value accounts for the number of predictors in the model relative to the sample size and helps prevent overestimation of the model's explanatory power when additional variables are included. Since the difference between the two values is very small, it suggests that the model is not overfitted, and all predictors contribute meaningfully to explaining the response.

In summary, the R-squared values indicate that the model has a good fit, explaining a significant portion of the variability in `ln_price`, while leaving around 38.36% of the variability unexplained, potentially due to other factors not included in the model or inherent randomness in the data.

RMSE

```
residuals_model <- residuals(model)

# Calculate RMSE
rmse_value <- sqrt(mean(residuals_model^2))
print(paste("RMSE: ", rmse_value))
```

```
## [1] "RMSE:  0.265805816894233"
```

The RMSE (Root Mean Square Error) value of **0.2667** indicates the average error in the predictions made by the model. Specifically, it means that, on average, the predicted values of `ln_price` deviate from the actual observed values by about **0.267** units in the logarithmic scale.

In practical terms, a lower RMSE value is better, as it reflects more accurate predictions. Considering this model's context, an RMSE of **0.2667** suggests that the model performs well and makes reasonably precise predictions.

MSE

```
mse_value <- mean(residuals_model^2)
print(paste("MSE: ", mse_value))
```

```
## [1] "MSE:  0.0706527322948104"
```

The Mean Squared Error (MSE) value of **0.0711** represents the average squared difference between the predicted and actual values of the response variable, `ln_price`. Since the MSE is the square of the Root Mean Square Error (RMSE), it captures the same concept but emphasizes larger errors more heavily due to squaring.

In this case, an MSE of **0.0711** suggests that, on average, the squared prediction errors are relatively small, which is a good sign for your model's accuracy.

Best Subset

```
best_subset <- regsubsets(ln_price ~ sqft_living + bathrooms + grade +
                            floors + view + condition + yr_built, data = cleaned_without_influential)


best_subset_summary <- summary(best_subset)


best_model_size <- which.max(best_subset_summary$adjr2)  # Model size with highest Adjusted R-Squared

best_predictors <- names(coef(best_subset, best_model_size))[-1]  # Remove the intercept

formula <- as.formula(paste("ln_price ~", paste(best_predictors, collapse = " + ")))

# Fit the final linear model
final_model <- lm(formula, data = cleaned_without_influential)


cat("Model Summary:\n")
```

```
## Model Summary:
```

```
summary(final_model)
```

```
##
## Call:
## lm(formula = formula, data = cleaned_without_influential)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.85010 -0.18908  0.01433  0.19193  0.85849
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.346e+01  1.948e-01 120.395   <2e-16 ***
## sqft_living  1.599e-04  4.603e-06  34.729   <2e-16 ***
## bathrooms    9.250e-02  4.975e-03  18.594   <2e-16 ***
## grade        2.266e-01  3.074e-03  73.737   <2e-16 ***
## floors       1.098e-01  4.745e-03  23.141   <2e-16 ***
## view         6.653e-02  5.669e-03  11.736   <2e-16 ***
## condition    3.162e-02  3.471e-03   9.108   <2e-16 ***
## yr_built    -6.572e-03  1.004e-04 -65.486   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2619 on 17533 degrees of freedom
## Multiple R-squared:  0.627,  Adjusted R-squared:  0.6268
## F-statistic:  4210 on 7 and 17533 DF,  p-value: < 2.2e-16
```

```
cat("\nVariance Inflation Factor (VIF):\n")
```

```
##
## Variance Inflation Factor (VIF):
```

```
vif(final_model)
```

```
## sqft_living   bathrooms       grade      floors        view   condition
##    2.875266    2.916093    2.469699    1.682343    1.032923    1.254474
##    yr_built
##    2.109415
```

The best subset of predictors for the model was identified using the best subset selection method, which systematically evaluates all possible combinations of predictors to find the model that optimally balances goodness of fit and simplicity. This was achieved using the `regsubsets()` function, which computed models of varying sizes and provided a summary of evaluation metrics. The Adjusted R-squared value was chosen as the selection criterion to ensure the model explained the most variance in `ln_price` without unnecessary complexity.

The optimal model size, corresponding to the highest Adjusted R-squared value, was determined using `which.max(best_subset_summary$adjr2)`. The predictors from this model were extracted using the `coef()` function, ensuring that only variables with meaningful contributions to the response were included. Finally, a regression model was built dynamically using these predictors with the `lm()` function, resulting in an optimized model that balances explanatory power and parsimony. This rigorous selection process ensured that the final predictors were both statistically significant and practically relevant.

Regularized Regression for Comparison

Ridge Regression evaluation of the Model We are normalizing the data step by step so that Lasso and Ridge regression can regularize all features equally. We first notice the difference in scale among the features `sqft_living`, `sqft_lot`, `sqft_above` and `yr_built`, which have different ranges that can skew the data(Years, square-feet). Next, we scale the data so that each feature has a mean of 0 and a standard deviation of 1. This way, each feature contributes equally to the regularization process. This scaling also improves the performance of optimization algorithms and enables the model to learn more effectively. Normalizing the data decreases the chances of overfitting and multicollinearity that can impact the performance of Lasso and Ridge regression.

```
# Normalizing the data set
X <- cleaned_without_influential[, c("bedrooms", "bathrooms", "sqft_living", "sqft_lot",
                                     "floors", "view", "condition", "grade", "sqft_above",
                                     "yr_built")]
Y <- cleaned_without_influential$ln_p

set.seed(1)
trainIndex <- createDataPartition(Y, p = 0.7, list = FALSE)

# Split data the f stands for the future non normalized data to run the ols to compare
X_train_f <- X[trainIndex, ]
X_test_f  <- X[-trainIndex, ]
Y_train_f <- Y[trainIndex]
Y_test_f  <- Y[-trainIndex]


X_train <- X[trainIndex, ]
X_test  <- X[-trainIndex, ]
Y_train <- Y[trainIndex]
Y_test  <- Y[-trainIndex]

cols_to_scale <- c("sqft_living", "sqft_lot", "sqft_above", "yr_built")

# Standardize
preProc <- preProcess(X_train[, cols_to_scale], method = c("center", "scale"))
X_train[, cols_to_scale] <- predict(preProc, X_train[, cols_to_scale])
X_test[, cols_to_scale] <- predict(preProc, X_test[, cols_to_scale])

#into a matrix format
X_train_mat <- as.matrix(X_train)
X_test_mat  <- as.matrix(X_test)

lambdas <- 10^seq(2, -3, by = -0.1)

# Train Ridge Regression
ridge_cv <- cv.glmnet(X_train_mat, Y_train, alpha = 0, lambda = lambdas)


best_lambda_ridge <- ridge_cv$lambda.min


ridge_model <- glmnet(X_train_mat, Y_train, alpha = 0, lambda = best_lambda_ridge)
```

```
ridge_preds_train <- predict(ridge_model, newx = X_train_mat)
ridge_preds_test  <- predict(ridge_model, newx = X_test_mat)
```

Model Predictions and Evalueation

```
eval_metrics <- function(y_true, y_pred) {
  SSE <- sum((y_pred - y_true)^2)
  SST <- sum((y_true - mean(y_true))^2)

  r2 <- 1 - SSE / SST
  RMSE <- sqrt(mean((y_pred - y_true)^2))

  return(list(RMSE = RMSE, R2 = r2))
}


ridge_train_results <- eval_metrics(Y_train, ridge_preds_train)
ridge_test_results <- eval_metrics(Y_test, ridge_preds_test)

print(paste("Ridge Regression - Train RMSE:",
            ridge_train_results$RMSE, "R-squared:", ridge_train_results$R2))
```

Ridge Regression Model

```
## [1] "Ridge Regression - Train RMSE: 0.256523177326503 R-squared: 0.638570237301496"
```

```
print(paste("Ridge Regression - Test RMSE:",
            ridge_test_results$RMSE, "R-squared:", ridge_test_results$R2))
```

```
## [1] "Ridge Regression - Test RMSE: 0.261638728572623 R-squared: 0.635564532642372"
```

- R-squared shows that both the train and the test models have a average rating of 0.636. We can see here that after normalizing the data we get a good fit without overfitting the data
- Low RMSE suggests that the model has a small prediction error which indicates a small prediction error

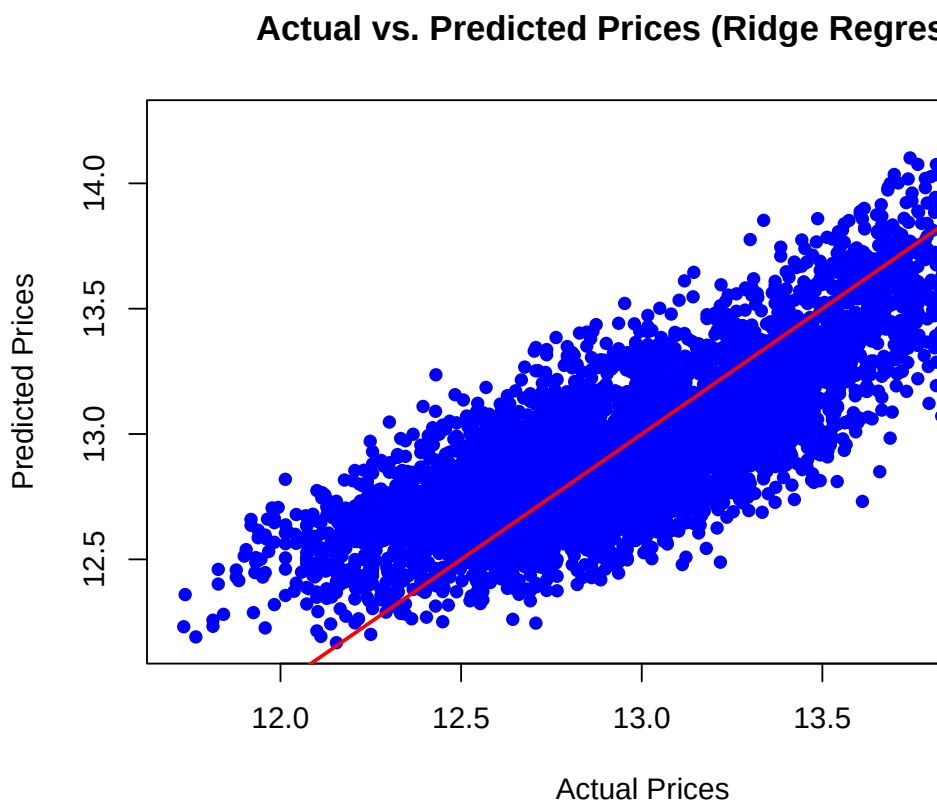
```
ridge_coeffs <- coef(ridge_model)
ridge_df <- data.frame(Feature = rownames(ridge_coeffs), Coefficient = as.vector(ridge_coeffs))


ridge_df <- ridge_df[ridge_df$Coefficient != 0, ]
print(ridge_df)
```

Ridge Coefficients

```
##         Feature Coefficient
## 1   (Intercept) 10.95930787
## 2      bedrooms -0.04026604
## 3     bathrooms  0.07779884
## 4   sqft_living  0.20184642
## 5      sqft_lot -0.01646917
## 6        floors  0.12851034
## 7          view  0.05407276
## 8     condition  0.03131139
## 9         grade  0.22417385
## 10  sqft_above -0.07123594
## 11    yr_built -0.17781980
```

The Ridge regression model worked effectively in highlighting critical predictors and minimizing the effect of less important features. The most important predictors in predicting `price` were `sqft_living` (0.2018) and `grade` (0.2242) in line with general real estate market trends. The`bathrooms`(0.0778) and `floors` (0.1285) features worked to increase prices, showing their importance in property prices. On the other hand, `sqft_above` (-0.0712) and `bedrooms`(-0.0403) received smaller weights because, logically speaking, it is not always a sign of increased cost to possess larger top floors or additional bedrooms. Unlike in the baseline model when t-values were sufficiently high to lead to overfitting, Ridge regression effectively reduced less significant features' weights, stabilizing the models in the process. Additionally, by reducing the magnitude of correlated variables such as `sqft_living` (2.875) and `bathrooms` (2.916), Ridge assisted in repairing multicollinearity for a more stable and usable model.

```r
par(mar = c(5, 5, 4, 2) + 0.1)
plot(Y_test, ridge_preds_test, main = "Actual vs. Predicted Prices (Ridge Regression)",
     xlab = "Actual Prices", ylab = "Predicted Prices", col = "blue", pch = 16)
abline(0, 1, col = "red", lwd = 2)
```



Model Visual of Ridge Regression
The graph shows a reasonably high positive relationship between actual and predicted house prices, which means that the Ridge Regression model is a good fit for this data. The points cluster relatively close to the diagonal line, which points towards good predictions with somewhat minimal error.

```
lambdas <- 10^seq(2, -3, by = -0.1)

lasso_cv <- cv.glmnet(X_train_mat, Y_train, alpha = 1, lambda = lambdas)

best_lambda_lasso <- lasso_cv$lambda.min

lasso_model <- glmnet(X_train_mat, Y_train, alpha = 1, lambda = best_lambda_lasso)

lasso_preds_train <- predict(lasso_model, newx = X_train_mat)
lasso_preds_test  <- predict(lasso_model, newx = X_test_mat)

eval_metrics <- function(y_true, y_pred) {
  SSE <- sum((y_pred - y_true)^2)
  SST <- sum((y_true - mean(y_true))^2)
```

```
  r2 <- 1 - SSE / SST
  RMSE <- sqrt(mean((y_pred - y_true)^2))

  return(list(RMSE = RMSE, R2 = r2))
}

# Evaluate Lasso Model
lasso_train_results <- eval_metrics(Y_train, lasso_preds_train)
lasso_test_results <- eval_metrics(Y_test, lasso_preds_test)

print(paste("Lasso Regression - Train RMSE:", lasso_train_results$RMSE, "R²:", lasso_train_results$R2))
```

Lasso Regression Evaluation

```
## [1] "Lasso Regression - Train RMSE: 0.256577055421302 R²: 0.638418397672707"
```

```
print(paste("Lasso Regression - Test RMSE:", lasso_test_results$RMSE, "R²:", lasso_test_results$R2))
```

```
## [1] "Lasso Regression - Test RMSE: 0.261761503340139 R²: 0.635222427495854"
```

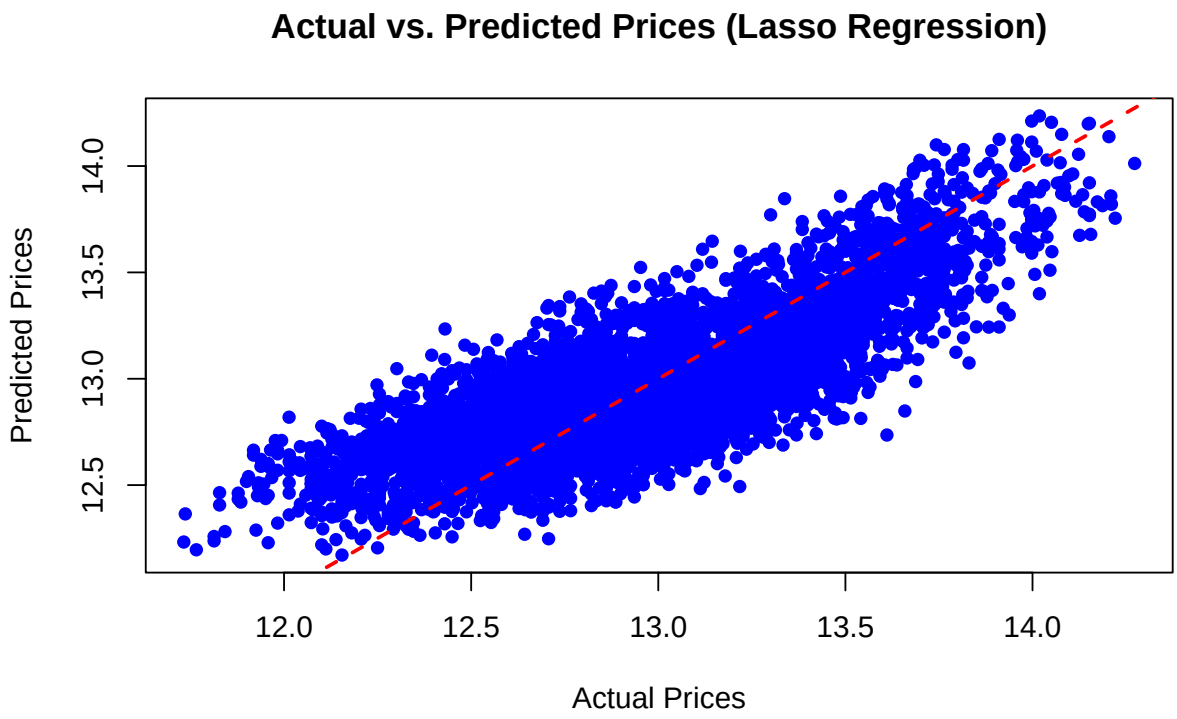
```
lasso_coeffs <- coef(lasso_model)
lasso_df <- data.frame(Feature = rownames(lasso_coeffs), Coefficient = as.vector(lasso_coeffs))

lasso_df <- lasso_df[lasso_df$Coefficient != 0, ]
print(lasso_df)
```

Lasso Coeficients

```
##        Feature Coefficient
## 1  (Intercept) 10.96751942
## 2     bedrooms -0.03743207
## 3    bathrooms  0.07687548
## 4  sqft_living  0.19528924
## 5     sqft_lot -0.01581566
## 6       floors  0.12351483
## 7         view  0.05278693
## 8    condition  0.03007338
## 9        grade  0.22364724
## 10  sqft_above -0.06442070
## 11    yr_built -0.17679828
```

Lasso regression model successfully determined major predictors and constrained smaller or insignificant coefficients to zero. `sqft_living` (0.1953) and `grade` (0.2236) were the most influential variables impacting `ln_price`, which corroborates current trends in the 2014-2105 Washington property market. Moreover, `bathrooms` (0.0769) and `floors` (0.1235) positively influenced property value, thereby validating their inclusion in the `price` function. Conversely, `sqft_above` (-0.0644) and `bedrooms`(-0.0374) had lower weights, reflecting their lower influence on price, since adding more bedrooms or bigger upper floors does not always increase the value of a property. In contrast to the baseline model, which was prone to overfitting because of high t-values, Lasso regression automatically pushed some coefficients towards zero, thus performing feature selection and improving the model's interpretability. By reducing the impact of less informative features like `sqft_lot` (-0.0158) and regularizing multicollinear predictors, Lasso regression created a simpler and more stable model to predict house prices.

```r
plot(Y_test, lasso_preds_test, main = "Actual vs. Predicted Prices (Lasso Regression)",
     xlab = "Actual Prices", ylab = "Predicted Prices", col = "blue", pch = 16)
abline(0, 1, col = "red", lwd = 2, lty = 2)
```



Lasso Model

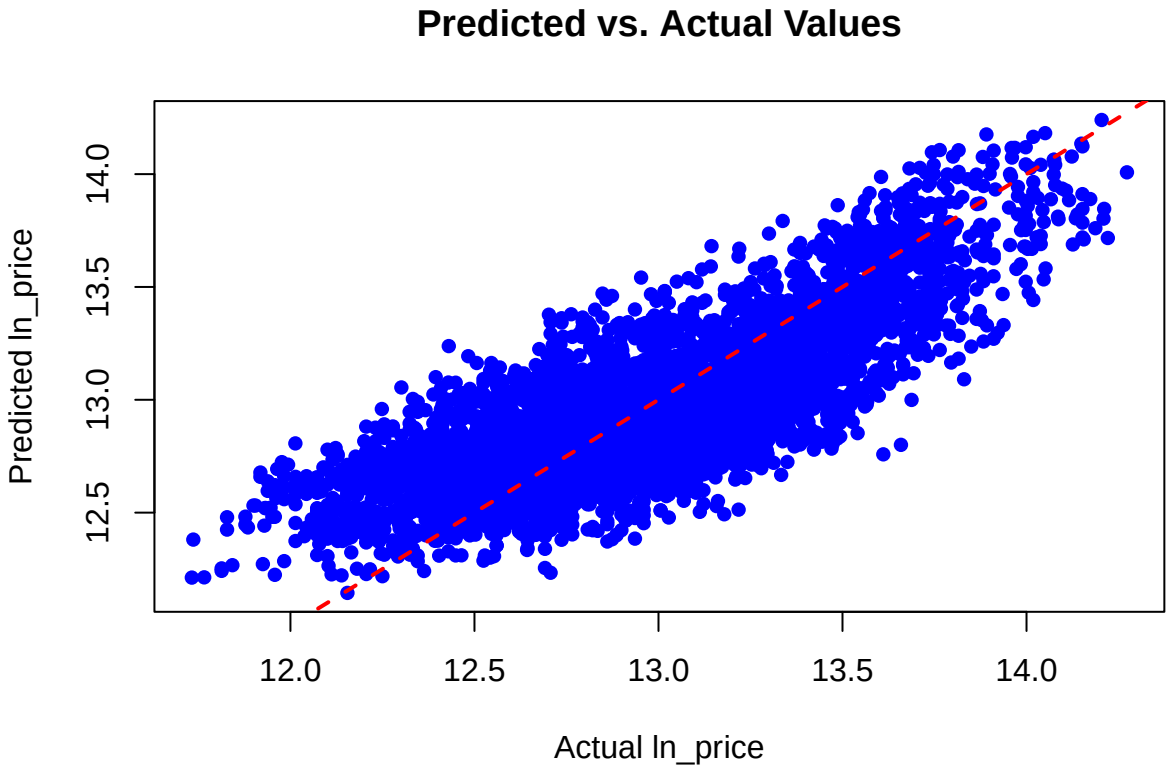
The graph illustrates a high enough positive correlation between actual and predicted house prices, meaning the Ridge Regression model is a good fit for the data. The points cluster reasonably close to the diagonal line, suggesting good predictions with moderately low error. Nonetheless, there is some appreciable scatter, especially at high prices, suggesting that the model will struggle more to precisely forecast prices for higher-priced homes. This remark implies that there may be other factors at play in determining the price of high-value homes that are not well represented by the current model.

Both models Display valid alternaitves to solving and predicting our model in a more efficent and simple matter. While All models developed come into a close R-sqaured range and share simaler RMSE, It may be worthwhile implementing and looking into the normalization and Alternative regression based appriachs to get stronger and more streamlined models

Prediction of ln(Price)

```r
predictions <- predict(final_model, newdata = X_test_f)

# Plot predicted vs. actual values
plot(Y_test_f, predictions,
     xlab = "Actual ln_price",
     ylab = "Predicted ln_price",
     main = "Predicted vs. Actual Values",
```

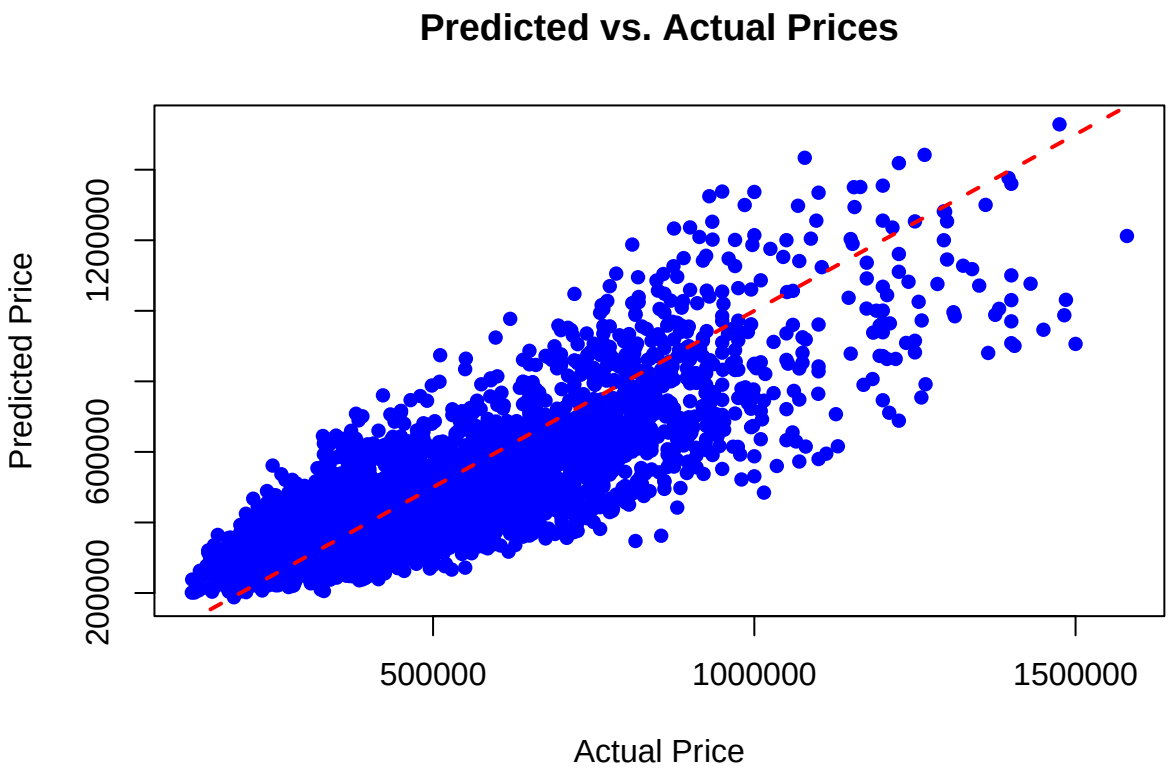
```
     col = "blue", pch = 16)
abline(a = 0, b = 1, col = "red", lwd = 2, lty = 2)
```



We will proceed to model out the price based predictions in the data as seen below

Prediction of Price

```
predictions <- predict(final_model, newdata = X_test_f)

predicted_prices <- exp(predictions)

# get the general price model of the data not the log
actual_prices <- exp(Y_test_f)

plot(actual_prices, predicted_prices,
     xlab = "Actual Price",
     ylab = "Predicted Price",
     main = "Predicted vs. Actual Prices",
     col = "blue", pch = 16)

abline(a = 0, b = 1, col = "red", lwd = 2, lty = 2)
```

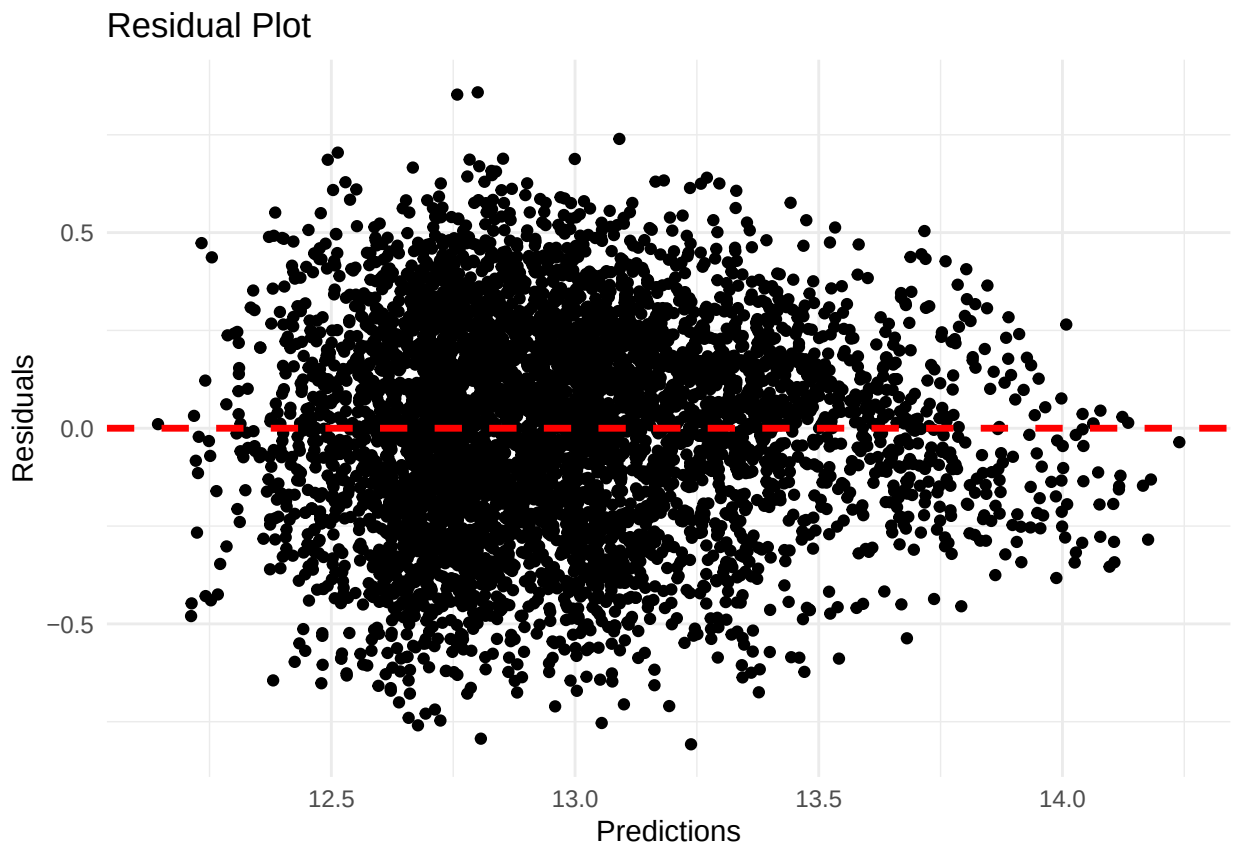


The points are clustered around the y = x identity line, meaning that the predicted price from our model is usually close to the true price. This would suggest the model is fitting relatively well to the data.

Residuals

```
residuals_df <- tibble(
  predictions = predictions,
  residuals = Y_test_f - predictions
)

ggplot(residuals_df, aes(x = predictions, y = residuals)) +
  geom_point() +  # Scatter plot of predictions vs residuals
  geom_hline(yintercept = 0, color = "red", linetype = "dashed", size = 1.2) +  # Horizontal line at y =
  labs(title = "Residual Plot", x = "Predictions", y = "Residuals") +
  theme_minimal()
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



The Residuals plot shows the model to be homoscedastic. The residuals appear to be roughly even distributed throughout the model showcasing that the varianceis roughly equal.

Conlcusion and Summary

For this project, we created a multivariable regression model, which was designed to predict house prices in Seattle. Our data came from the dataset 'House Sales in King County, USA', Seattle being in King County. The dataset and our subsequent analysis involved various explanatory variables, including square footage, grade(quality), number of bathrooms, condition, view quality, and year built.

We began our analysis by performing backwards stepwise regression using stepAIC, resulting in a stepwise model with many predictors, which would be used to predict the log transform of price, which was used instead of just price to improve distribution symmetry. We then applied Cook's Distance to identify influential points in our data, and we removed them. We also removed the predictor sqft_above in order to address multicollinearity, and we removed the predictor floors due to it's limited significance and to remove complexity from the model.

After some refinement, we were able to obtain a model with 6 predictors with a moderate adjusted R^2 value of 0.6155, and an okay RMSE of 0.2659. We then evaluated the model using the anova() function, to get Sum Squares, Mean Squares, p values, and F values, which generally showed that the variables chosen in the model, being sqft_living, grade, bathrooms, view, condition, and yr_built, effectively accounted for most of the variability in ln(price).

Best subset selection was then used to select the predictors providing the highest adjusted R^2. Ridge Regression was used, and added stability by properly weighting different features, while also reducing multicollinearilty among correlated variables. Lasso Regression was used, and similarly reduced the impact of less important features in the model, creating a more stable predictive model.

Normalization and Alternative regression based approaches are likely useful tools in improving overall model quality. Looking at the positives of our project, our model ended up demonstrating some predictive ability, with an adjusted R squared of well over 0.6. Clearly, home price can be, to some degree, explained by our predictors. Also, we were able to symmetrize our response variable with a log transform, which better prepares our data for linear regression.

We were also able to simplify our model significantly via removing insignificant/problematic variables, while still maintaining predictive ability. Looking at the negative aspects of our model, we were unable to include certain key variables relating to socioeconomic factors that could affect house price, including safety, school/park access, quality of schools...etc, and so we missed out on including factors that could improve predictive ability.

Also, our improved regression model still showed tailing on both ends of our Q-Q plot. In the future, we could include these socioeconomic factors, along with location based factors that could identify certain areas that are more undesirable to live in than others. Also, it would be useful from a practical standpoint to have more recent information, considering that this model is meant to help better understand the real life issue of home pricing. In particular, factors like Neighborhood, Crime Rate, Public Transport Access, Commute Time, Natural Disaster Risk, and Pollution could all improve model accuracy.

References

1) The dataset used in this study is the "House Sales in King County, USA" dataset, obtained from https://www.openml.org/search?type=data&status=active&id=42635&sort=runs. The Author of the Dataset can be found at https://www.kaggle.com/harlfoxem/.
2) Article entitled "Multiple Linear Regression in R" from sthda.com. This article can be found at https://www.sthda.com/english/articles/40-regression-analysis/168-multiple-linear-regression-in-r/, and was written by user "kassambara".
3) Malpezzi, Stephen. "Housing Prices, Externalities, and Regulation in U.S. Metropolitan Areas." Journal of Housing Research 7, no. 2 (1996): 209–41. http://www.jstor.org/stable/24832860.
4) Yadavalli, Anita, Brenna Rivett, James Brooks, and Christiana K. McFarland. "A Comprehensive Look at Housing Market Conditions Across America's Cities." Cityscape 22, no. 2 (2020): 111–32. https://www.jstor.org/stable/26926899.