House Sales Regression Analysis

Camilla Ren, Yuxuan Wang, Yisa Wu

Introduction

This analysis aims to develop a robust predictive model for housing prices using a comprehensive dataset of over 21,000 housing units in King County, Washington. This dataset captures various structural, location, and neighbourhood characteristics that have an impact on home prices. Predicting house prices provides insights into market dynamics and supports informed decision making for stakeholders such as buyers, sellers, and policymakers. Previous studies have demonstrated the importance of environmental and structural factors in hedonic house price modelling, such as the seminal work of Harrison and Rubinfeld (1978). Building on this, this analysis employs advanced statistical techniques, including polynomial transformations and LASSO regression, to account for data nonlinearity and multicollinearity. The aim is to improve prediction accuracy and to derive meaningful insights into the key drivers of housing prices.

Descriptive Analysis on Dataset

Dataset Description

The dataset includes over 21,000 housing units and provides a comprehensive record of historical house sales in King County, Washington, with detailed information across 21 variables. A random sample of 500 observations was used for efficient and representative analysis. The price of the house serves as the target variable, while the dataset captures a range of property characteristics, including structural features (e.g., number of bedrooms, bathrooms, square footage), location-related attributes (e.g., proximity to the waterfront, view quality, and geographical coordinates), and additional factors like the year the house was built and renovated, as well as neighborhood details. However, some variables, such as date and id, are not relevant for predictive modeling and should be removed.

Distribution of Variables

The distribution of house prices exhibits a strong right skew, with a concentration of lower-priced homes and a few high-value outliers stretching the distribution (Figure 1). This indicates that the price does not follow a normal distribution, which is a key assumption for many regression models. To improve model performance and meet the normality assumption, a transformation of the price variable may be necessary.

Other variables like square footage and number of bedrooms and bathrooms show more typical distributions, with moderate peaks around certain values (Figure 2), suggesting that most homes in King County tend to be of average size. Variables related to the condition, grade, and view of the house appear more evenly distributed, indicating that higher-quality homes or desirable views are less frequent but still present. These insights highlight the need to address the skewness of the target variable to enhance the predictive power of the model.

The boxplot reveals several outliers (Figure 3). The price outliers represent high-value properties, likely luxury or waterfront homes, which are significantly higher than the majority of the data. Similarly, sqft_lot and sqft_lot15 show extreme values, indicating unusually large properties or land sizes. These outliers can skew the regression model, especially for price. To address their impact, techniques like robust regression or data transformations may be needed to improve model accuracy.

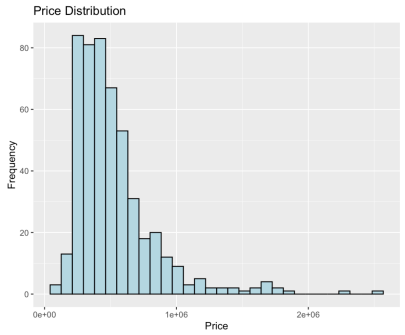


Figure 1

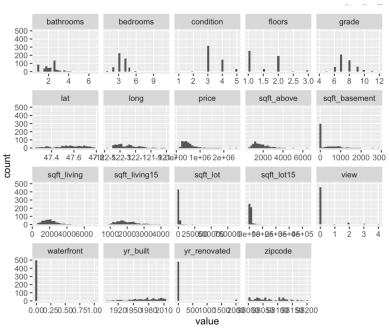


Figure 2

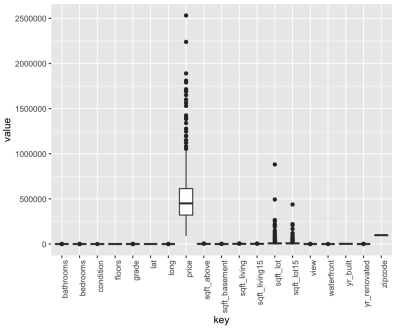


Figure 3

Correlation between Variables

The correlation analysis using heatmap reveals significant relationships between the explanatory variables and the target variable (Figure 4). Variables such as sqft_living, grade, and sqft_above show strong positive correlations with price, indicating that larger, higher-quality homes tend to have higher values. The presence of a waterfront or a greater number of bathrooms also positively influences price, reflecting the added value of these features. However, zip code, yr_built, and condition exhibit weaker correlations with price, suggesting their lesser direct impact on pricing. Additionally, sqft_lot and sqft_lot15 display minimal correlation with price, implying that the lot size may not be as significant a factor in determining the home value in King County.

On the other hand, several explanatory variables show high correlations with each other. For example, sqft_living is strongly correlated with both sqft_above and bathrooms, as larger homes tend to have more space and amenities. These interrelationships among explanatory variables could cause multicollinearity, making it challenging to estimate the individual effects of predictors on the target variable. Techniques like variance inflation factor (VIF) analysis may be needed to reduce the impact of multicollinearity and improve model accuracy.

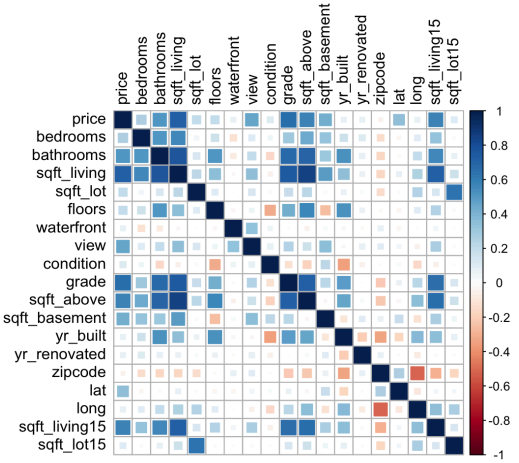


Figure 4

Modeling Analysis

Given the nature of the dataset, multiple linear regression is an appropriate modeling approach for predicting house prices. This technique allows us to quantify the relationships between the target variable (price) and multiple explanatory variables, while accounting for the combined influence of all predictors.

Data Cleaning and Preprocessing

Data preprocessing was an essential step to prepare the dataset for analysis and involved several key tasks. To ensure the quality of the analysis, the following steps were undertaken:

First, irrelevant variables such as id and date were removed, as they do not provide meaningful information for predicting the target variable, price, and could introduce noise into the model. The dataset was then assessed for missing values, but none were found, eliminating the need for imputation or data cleaning, which ensured the dataset was complete and consistent for analysis.

A random sample of 500 observations was selected as the training set for model development, with the remaining data set aside as the testing set to evaluate the model's performance and generalizability. During outlier detection, boxplots revealed significant outliers in variables like price and sqft_living. As these data are reasonable observations reflecting real-world luxury homes or large properties, these outliers were retained in the model.

Initial Modeling (Baseline Model)

The initial multiple linear regression model was built using all explanatory variables in the preprocessed dataset (Table 1). The model is statistically significant in explaining house prices, with a p-value of <2.2e-16. It accounts for approximately 74% of the variance in house prices, as indicated by an R-squared value of 0.7407. Significance of predictors vary and are shown by p-values. The residual standard error of 160,700 reflects moderate variability in the model's predictions, indicating room for further optimization and refinement to improve its predictive accuracy.

Variable Selection & Multicollinearity

To address multicollinearity in the model, diagnostic tests were performed to identify highly correlated predictors. The alias matrix revealed that sqft_basement was perfectly collinear with other predictors, particularly sqft_living and sqft_above, leading to its exclusion. A reduced model was then fitted, and Variance Inflation Factor (VIF) values were calculated to assess multicollinearity among the remaining predictors. Using a VIF cutoff of 5, sqft_living and sqft_above were identified as highly collinear and subsequently removed to improve model stability and interpretability.

For variable selection, stepwise selection was applied using the step() function, combining forward selection and backward elimination based on the Akaike Information Criterion (AIC) to balance model complexity and goodness-of-fit. The final model (Table 1) included 9 features and explains approximately 71% of the variance in house prices (R-squared = 0.7099, Adjusted R-squared = 0.7045) and is statistically significant overall, with an F-statistic of 133.2 and a p-value of <2.2e-16.

Comparison	Baseline Model	After Variable Selection
Number of Features	18	9
R2	0.74	0.71

Summary		

```
Call:
lm(formula = price ~ ., data = data)

Residuals:
     Min       1Q   Median       3Q      Max
-504791  -89882  -11526   71471  928609

Coefficients: (1 not defined because of singularities)
                Estimate Std. Error t value Pr(>|t|)
(Intercept)   -1.193e+06  1.652e+07  -0.072 0.942466
bedrooms      -3.889e+04  9.615e+03  -4.045 6.10e-05 ***
bathrooms      4.628e+04  1.785e+04   2.593 0.009815 **
sqft_living    1.124e+02  2.455e+01   4.579 5.95e-06 ***
sqft_lot       3.999e-01  1.880e-01   2.127 0.033936 *
floors        -1.066e+04  1.990e+04  -0.535 0.592595
waterfront     3.997e+04  1.245e+05   0.321 0.748283
view           8.712e+04  1.219e+04   7.145 3.35e-12 ***
condition      4.818e+04  1.288e+04   3.742 0.000205 ***
grade          8.975e+04  1.116e+04   8.042 6.92e-15 ***
sqft_above     3.084e+01  2.442e+01   1.263 0.207265
sqft_basement         NA         NA      NA       NA
yr_built      -2.204e+03  3.906e+02  -5.642 2.87e-08 ***
yr_renovated  -4.614e+00  2.148e+01  -0.215 0.830005
zipcode       -3.499e+02  1.814e+02  -1.929 0.054261 .
lat            6.132e+05  5.821e+04  10.534  < 2e-16 ***
long          -8.232e+04  6.688e+04  -1.231 0.218941
sqft_living15  3.198e+01  1.767e+01   1.810 0.070858 .
sqft_lot15    -6.837e-01  3.412e-01  -2.004 0.045645 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 160700 on 482 degrees of freedom
Multiple R-squared:  0.7407,    Adjusted R-squared:  0.7315
F-statistic: 80.97 on 17 and 482 DF,  p-value: < 2.2e-16
```

```
Call:
lm(formula = price ~ bedrooms + bathrooms + sqft_lot + view +
    condition + grade + yr_built + lat + sqft_living15, data = data)

Residuals:
     Min       1Q   Median       3Q      Max
-405749  -91653   -9559   70408 1014928

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)   -2.414e+07  2.982e+06  -8.095 4.58e-15 ***
bedrooms      -1.376e+04  9.321e+03  -1.476 0.140565
bathrooms      9.335e+04  1.598e+04   5.840 9.50e-09 ***
sqft_lot       3.031e-01  1.482e-01   2.046 0.041328 *
view           1.002e+05  1.122e+04   8.925  < 2e-16 ***
condition      4.973e+04  1.293e+04   3.846 0.000136 ***
grade          1.245e+05  1.056e+04  11.786  < 2e-16 ***
yr_built      -2.583e+03  3.442e+02  -7.505 2.91e-13 ***
lat            5.950e+05  5.808e+04  10.245  < 2e-16 ***
sqft_living15  8.433e+01  1.617e+01   5.216 2.70e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 168600 on 490 degrees of freedom
Multiple R-squared:  0.7099,    Adjusted R-squared:  0.7045
F-statistic: 133.2 on 9 and 490 DF,  p-value: < 2.2e-16
```

Table 1

Transformation

1. Model Diagnostics

To evaluate the assumptions of the multiple linear regression model, we conducted both visual diagnostics and formal statistical tests. The residuals vs. fitted plot and Q-Q plot (Figure 5) were used as visual tools to assess the assumptions of equal variance and normality of residuals. The residuals vs. fitted plot shows a clear pattern, suggesting non-constant variance, while the Q-Q plot reveals deviations from the diagonal line at the tails, indicating non-normality of the residuals.

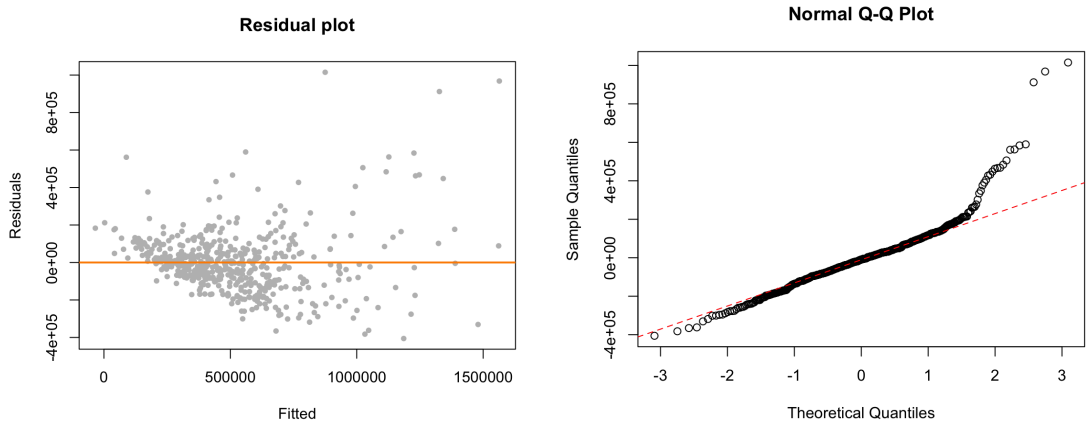


Figure 5

Formal statistical tests were then applied to confirm these observations. To test the assumption of equal variance, the Breusch-Pagan test (bptest) was used, yielding a p-value of 3.430378e-15, which is less than 0.05. This result confirms that the equal variance assumption is violated. For normality, the Shapiro-Wilk test (swtest) was conducted, resulting in a p-value of 3.771701e-18, also less than 0.05, indicating that the residuals do not follow a normal distribution. These findings collectively suggest that both assumptions of normality and equal variance are violated.

2. Transformation of Y

To address the violations of regression assumptions, several steps were undertaken to improve the model's adherence to these requirements. Transformations were applied to the response variable (y) to stabilize variance and improve normality, guided by both diagnostic plots and formal tests.

A square root transformation was initially tested based on the residual plot, which suggested that the variance exhibited a linear relationship with the response variable. Additionally, a Box-Cox transformation was performed to systematically identify the optimal transformation parameter, ensuring the response variable was appropriately scaled to address both heteroscedasticity and non-normality.

To further refine the model and mitigate the influence of extreme data points, diagnostics using Cook's distance and leverage values were conducted(Figure 6). Cook's distance was calculated to identify influential points that could disproportionately impact the regression model, while leverage values were analyzed to detect high-leverage points with a strong potential to influence the overall model fit. Data points exceeding the respective thresholds for Cook's distance (4/(n−k−1)4/(n-k-1)4/(n−k−1)) and leverage (2k/n2k/n2k/n) were flagged as problematic. These influential and high-leverage points were removed to enhance the robustness of the model, and the model was subsequently refitted using the cleaned dataset.

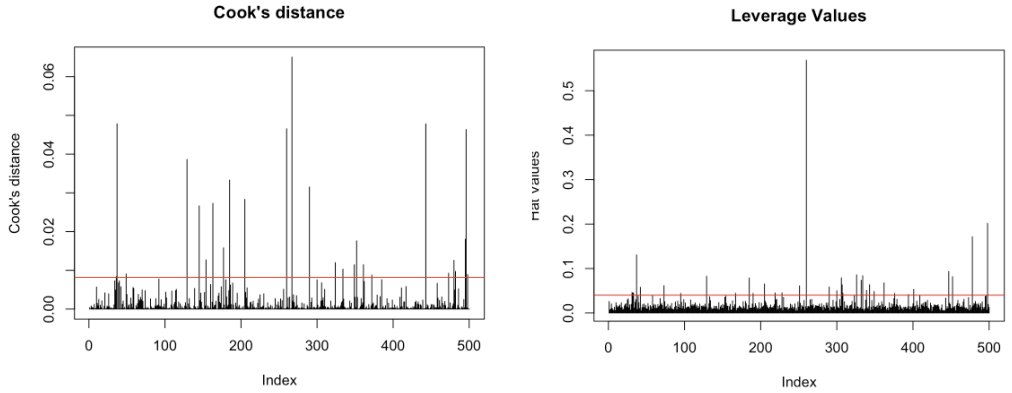


Figure 6

After refitting the model, diagnostic tests were performed to reassess the assumptions. The Shapiro-Wilk normality test yielded a p-value of 0.1682, indicating that the residuals of the cleaned model are now normally distributed, thus satisfying the normality assumption. However, the studentized Breusch-Pagan test produced a p-value of 0.01198, suggesting that the issue of equal variance persists, further optimization opportunities remain to fully resolve the equal variance issue, such as applying transformation on features.

3. Transformation of X

In response to the previous diagnostics, which highlighted the potential for non-linearity and heteroscedasticity in the data, polynomial transformations were applied to several of the predictor variables in order to better fit the assumptions of linear regression. Based on the patterns observed in the scatter plots, quadratic transformations were used for 'bedrooms', 'bathrooms', 'grade' and 'yr_built'. A cubic transformation was deemed appropriate for 'sqft_lot' to account for its more complex non-linear relationship with the response variable 'price_trans'. To capture any subtle curvature in its relationship with the response, the predictor 'sqft_living15' was also transformed quadratically.

Diagnostic plots and statistical tests were used to assess the effectiveness of these transformations. The residuals plot showed an improvement in the model's homoscedasticity, or constant variance across predictions, as the residuals were randomly distributed around the zero line. The normal Q-Q plot, with residuals closely following the theoretical line, confirming the normality of their

distribution, supported this improvement. Statistical validation was provided by the Shapiro-Wilk test. The p-value was 0.1387, indicating normality of the residuals. The Breusch-Pagan test also confirmed the absence of heteroscedasticity, with a p-value of 0.1542. This test confirms that the variance of the residuals is consistent across different levels of the fitted values.

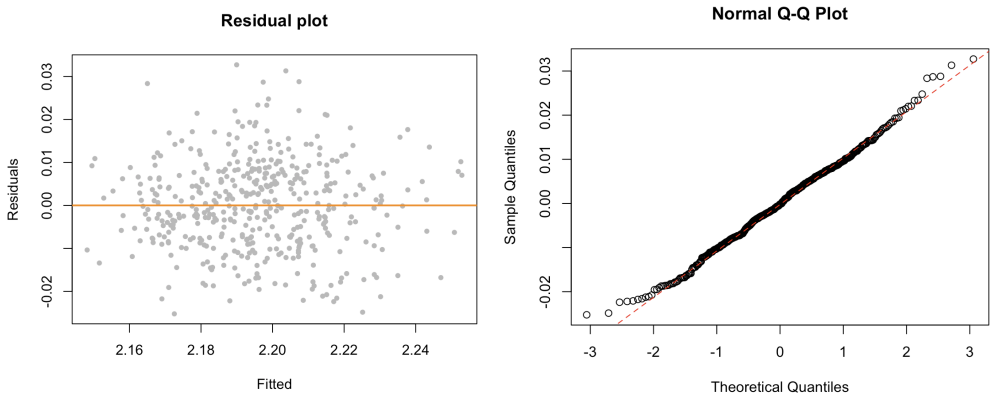


Figure 7

The adjusted R-squared value of 0.774 indicates a robust fit of the model, which explains a significant proportion of the variance in the response. Although the model now satisfies most of the regression assumptions, the persistence of low heteroscedasticity, as indicated by the still significant Breusch-Pagan test, suggests areas for further optimization, such as exploring additional transformations or using weighted regression techniques to fully stabilize the variance across the range of predictions.

4. Regression Shrinkage Method- LASSO

To further improve our regression model for predicting prices, we used regression shrinkage, specifically LASSO, as a crucial step in further refining the model's accuracy and variable selection efficiency. This method was chosen because it has the ability to shrink variables, effectively reducing the impact of less significant predictors by setting the coefficients to zero, thus making the overall model simpler without sacrificing predictive power.

Initially, we compared both LASSO and Ridge regression techniques through a rigorous cross-validation process to determine the most effective approach. LASSO was selected because of its dual variable selection and regularization capabilities, which are essential for dealing with the multicollinearity typically present in datasets with a high number of predictors. The final LASSO model included a diverse set of predictors, including polynomial transformations for variables such as `bedrooms`, `bathrooms`, `sqft_lot` and `yr_built`, as well as categorical variables such as `view` and `condition`. A systematic cross-validation process, focusing on minimising the mean squared error and maximising the R-squared value, was used to determine the optimal regularisation parameter, lambda. The model performed well, explaining approximately 78% of the variance in log transformed prices, indicating a robust fit.

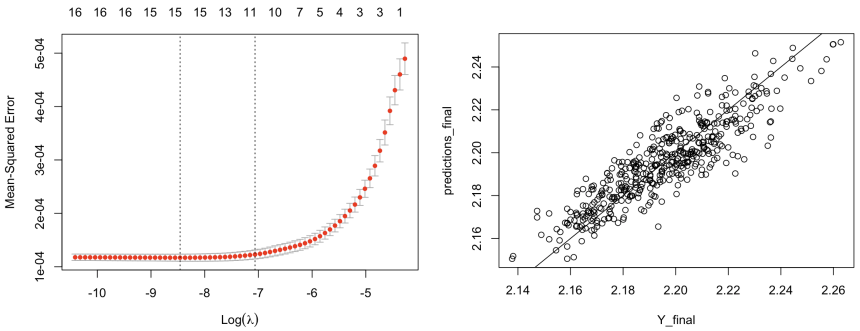


Figure 8

A comprehensive diagnostic of the model was carried out to ensure the reliability of the predictions and the validity of the assumptions of the model. This suggests that the model effectively captures the substantial variation in the data without any obvious signs of heteroskedasticity or unaddressed nonlinearities. The normal Q-Q plot provided further support for the normality of the residuals, with the majority of the data points closely aligned with the theoretical line, although minor deviations were observed at the tails. The Shapiro-Wilk test confirmed the normality of the residuals with a p-value of 0.1696, and the Breusch-Pagan test confirmed homoscedasticity with a p-value of 0.2054.

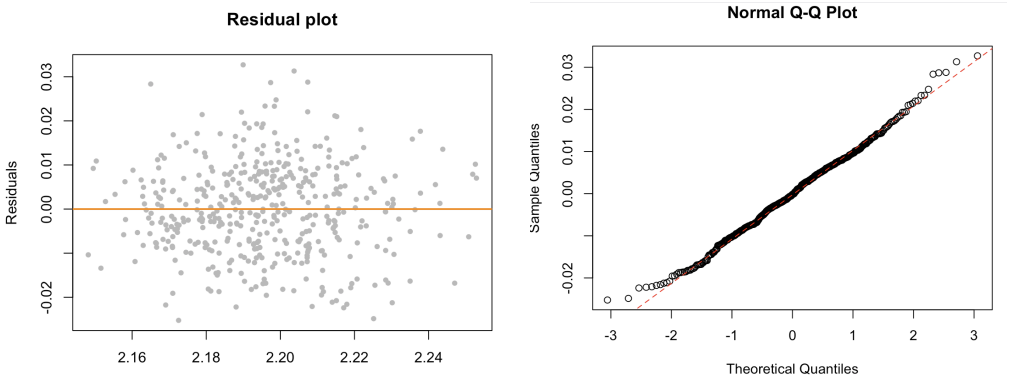


Figure 9

These results confirm that the final LASSO model, refined through cross-validation and equipped with advanced regression techniques, is both statistically robust and reliable in practice, making it an excellent tool for making informed decisions in the pricing of real estate.

Summary of Output and Results

Our final model includes 18 features, incorporating both linear and non-linear expansions, and demonstrates strong performance in predicting house prices. The model achieves an R2 value of 0.783, meaning approximately 78% of the variability in house prices is explained by the predictors. The Residual Standard Error (RSE) of 0.0105 represents a substantial improvement over baseline models, emphasizing the accuracy of the final predictions. Among the significant predictors positively correlated with house price are property condition (p=6.08 x 10^-8) and grade (p=2×10^−16), with grade showing particularly strong predictive power. Other positive correlations include the square footage of the property lot (2nd polynomial term, p=0.0173) and the living area (p=3.35×10^−15), which are key indicators of property value. Conversely, some predictors show negative correlations, such as the year the property was built (p=3.79×10^−6), suggesting older properties tend to be less valuable. These insights highlight the critical role of property features in influencing house prices and

provide a clear interpretation of the model, which can guide future predictions and practical applications in real estate analytics.

Model Evaluations:

During the model transformation and fitting process, we compared three models: the baseline model, the log-transformed model (before applying LASSO), and the final model. The comparison highlights the progressive impact of variable transformations, outlier handling, and feature selection on model performance. Variable transformations and outlier handling in the log-transformed model helped address deviations from model assumptions, such as non-linearity and heteroscedasticity, which resulted in more stable and interpretable coefficients. This improvement is evident in the performance metrics, as the log-transformed model mitigated some of the limitations of the baseline model, though the R2 value slightly decreased from 0.741 to 0.690. Despite this, the transformation provided strong evidence for its utility in preparing the data for further refinement.

The transition to the final model, achieved through LASSO regularization, brought about a substantial enhancement in performance. LASSO effectively reduced the dimensionality of the model by selecting the most relevant predictors and penalizing less important ones, which improved the overall generalizability and predictive accuracy. This is reflected in the significant increase in R2 to 0.781 and the dramatic reduction in RMSE from 0.556 in the log model to 0.0105 in the final model. The comparison underscores the importance of systematic transformations for meeting modeling assumptions, followed by the use of advanced techniques like LASSO to optimize feature selection and further enhance the model's performance. Collectively, these steps demonstrate the iterative process of refining a predictive model for improved accuracy and robustness.

Model <chr>	**MSE** <dbl>	**RMSE** <dbl>	**R_squared** <dbl>
Model 0	0.2588248968	0.50874836	0.7406564
Model Log	0.3089523031	0.55583478	0.6903550
Model Final	0.0001071711	0.01035235	0.7814330

Table 2

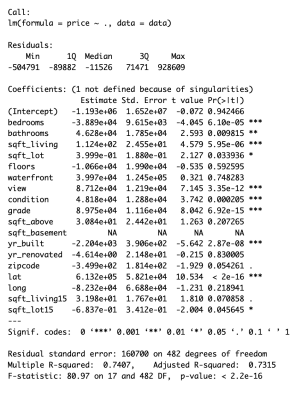
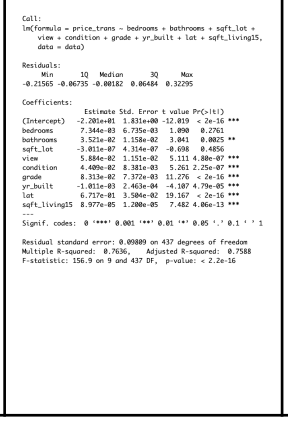
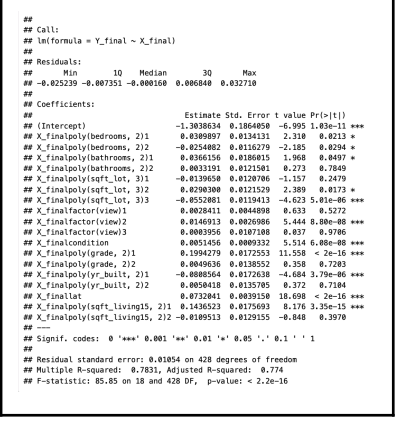
Comparison	Baseline Model	Model Log	Final Model
Number of Features	18	9	9 + 9 (nonlinear expansion)
R2	0.74	0.69	0.78
Summary			

Table 3

Findings and Inferences

The analysis reveals important findings about the drivers of house prices in King County, Washington. Using LASSO regression with polynomial transformations, the final model explained approximately 77% of the variance in log-transformed house prices, an improvement over the baseline model. Key predictors such as square footage, number of bedrooms and bathrooms, property grade and location factors such as latitude and view showed a strong association with prices, consistent with expectations and previous studies such as Harrison and Rubinfeld (1978). Transformations of the response and predictor variables dealt effectively with non-linearity, improving the interpretability and fit of the model. However, despite these refinements, heteroscedasticity persisted as indicated by the Breusch-Pagan test, suggesting that further variance stabilisation techniques could improve the model. The analysis highlights the value of iterative data pre-processing and feature selection in achieving robust predictive performance. While the model provides actionable insights for stakeholders, including real estate professionals and policy makers, future exploration of advanced techniques such as weighted regression or machine learning could address remaining challenges and further improve predictions.

The overall model performed well, with an R^2 of 0.785, indicating that approximately 78.5% of the variation in housing prices is explained by the model. The adjusted R^2 also reached a satisfactory level, further confirming the model's robustness. However, some limitations remain. Residual analysis suggests that the model still struggles with heteroscedasticity, and skewed variables like sqft_lot and sqft_lot15 may introduce bias. These issues could be addressed in future analyses with more complex nonlinear models.

This analysis not only confirms the critical role of factors like living area, quality, and amenities in determining housing prices but also reveals some unexpected findings, such as the limited importance of floors and construction year. These results suggest that housing prices are influenced by more complex interactions and external factors, such as location, market demand, or broader economic conditions.

Limitations and Further Questions

This analysis uncovered some important findings but also has its limitations. The baseline model showed reasonable performance with an R-squared of 0.74 and an MSE of 0.2588, providing a solid basis for comparison. However, its predictive ability was hampered by multicollinearity and limited flexibility in capturing non-linear relationships.

The transformed model aimed at stabilizing the variance and dealing with skewness in the target variable. This transformation resulted in a slight decrease in performance, with an R-squared of 0.69 and a higher MSE of 0.3089. This suggests that the transformation of x and y alone may not have been sufficient to address underlying problems in the data or model structure.

The final LASSO model introduced regularisation to shrink less significant predictors, thereby reducing model complexity. While this approach is effective in many cases, the results did not show a significant improvement over the baseline. The R-squared of 0.7814 and MSE of 0.0001 suggest a good model fit, but this performance may have been influenced by overfitting to the specific data used. In addition, the inherent limitation of LASSO in capturing interactions or non-linear relationships without additional techniques may have limited its performance.

These observations raise questions about the relative utility of LASSO in this context compared to more flexible machine learning models like random forests or gradient boosting. Future work could explore these methods, which can inherently capture non-linearities and interactions. Additionally, cross-validation across multiple datasets or temporal splits would be necessary to ensure the model's robustness and generalizability.

The sample size is another limitation. With only 500 samples, the analysis might not capture the more complex patterns present in larger datasets. A larger sample size could improve the model's robustness and allow it to detect finer trends, such as subtle regional differences in pricing or rare but influential features.

Including more geographic and environmental variables would also enhance the analysis. Factors like proximity to schools, shopping centers, or public transportation, community ratings, and the amount of nearby green space likely have significant impacts on housing prices. These variables were not part of this analysis but could provide a more comprehensive perspective if included. Expanding the dataset and incorporating such features would likely improve both the precision and applicability of the model.

In summary, while this study has its limitations, introducing more advanced models, enriching the dataset with additional variables, and refining feature selection methods could lead to a more thorough understanding of the factors influencing housing prices. These steps would not only enhance future analyses but also provide more reliable insights for decision-making.

Reference

Harlfoxem. (2016, August 25). House sales in King County, USA. *Kaggle*. https://www.kaggle.com/datasets/harlfoxem/housesalesprediction/data

Harrison Jr., D., & Rubinfeld, D. L. (2004, July 28). Hedonic housing prices and the demand for Clean Air. *Journal of Environmental Economics and Management*. https://www.sciencedirect.com/science/article/abs/pii/0095069678900062?via%3Dihub

Appendix

Data Cleaning and Preprocessing

```
#Importing data
library(conflicted)
library(dplyr)
library(ggplot2)
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ forcats   1.0.0     ✔ stringr   1.5.1
## ✔ lubridate 1.9.3     ✔ tibble    3.2.1
## ✔ purrr     1.0.2     ✔ tidyr     1.3.1
## ✔ readr     2.1.5
```

```
library(car)
```

```
## Loading required package: carData
```

```
library(MASS)
library(lmtest)
```

```
## Loading required package: zoo
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(mgcv)
```

```
## Loading required package: nlme
##
## Attaching package: 'nlme'
##
## The following object is masked from 'package:dplyr':
##
##     collapse
##
## This is mgcv 1.9-1. For overview type 'help("mgcv-package")'.
```

```
library(corrplot)
```

```
## corrplot 0.95 loaded
```

```
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
```

```
# Load the data
fulldata <- read.csv("kc_house_data.csv")
```

```
# View Data Structure
head(fulldata)
```

```
##           id            date   price bedrooms bathrooms sqft_living sqft_lot
## 1 7129300520 20141013T000000  221900        3      1.00        1180     5650
## 2 6414100192 20141209T000000  538000        3      2.25        2570     7242
## 3 5631500400 20150225T000000  180000        2      1.00         770    10000
## 4 2487200875 20141209T000000  604000        4      3.00        1960     5000
## 5 1954400510 20150218T000000  510000        3      2.00        1680     8080
## 6 7237550310 20140512T000000 1225000        4      4.50        5420   101930
##   floors waterfront view condition grade sqft_above sqft_basement yr_built
## 1      1          0    0         3     7       1180             0     1955
## 2      2          0    0         3     7       2170           400     1951
## 3      1          0    0         3     6        770             0     1933
## 4      1          0    0         5     7       1050           910     1965
## 5      1          0    0         3     8       1680             0     1987
## 6      1          0    0         3    11       3890          1530     2001
##   yr_renovated zipcode     lat     long sqft_living15 sqft_lot15
## 1            0   98178 47.5112 -122.257          1340       5650
## 2         1991   98125 47.7210 -122.319          1690       7639
## 3            0   98028 47.7379 -122.233          2720       8062
## 4            0   98136 47.5208 -122.393          1360       5000
## 5            0   98074 47.6168 -122.045          1800       7503
## 6            0   98053 47.6561 -122.005          4760     101930
```

```
summary(fulldata)
```

```
##        id                date               price            bedrooms
##  Min.   :1.000e+06   Length:21613       Min.   :  75000   Min.   : 0.000
##  1st Qu.:2.123e+09   Class :character   1st Qu.: 321950   1st Qu.: 3.000
##  Median :3.905e+09   Mode  :character   Median : 450000   Median : 3.000
##  Mean   :4.580e+09                      Mean   : 540088   Mean   : 3.371
##  3rd Qu.:7.309e+09                      3rd Qu.: 645000   3rd Qu.: 4.000
##  Max.   :9.900e+09                      Max.   :7700000   Max.   :33.000
##    bathrooms      sqft_living       sqft_lot           floors
##  Min.   :0.000   Min.   :  290   Min.   :    520   Min.   :1.000
##  1st Qu.:1.750   1st Qu.: 1427   1st Qu.:   5040   1st Qu.:1.000
##  Median :2.250   Median : 1910   Median :   7618   Median :1.500
##  Mean   :2.115   Mean   : 2080   Mean   :  15107   Mean   :1.494
##  3rd Qu.:2.500   3rd Qu.: 2550   3rd Qu.:  10688   3rd Qu.:2.000
##  Max.   :8.000   Max.   :13540   Max.   :1651359   Max.   :3.500
##    waterfront            view          condition         grade
##  Min.   :0.000000   Min.   :0.0000   Min.   :1.000   Min.   : 1.000
##  1st Qu.:0.000000   1st Qu.:0.0000   1st Qu.:3.000   1st Qu.: 7.000
##  Median :0.000000   Median :0.0000   Median :3.000   Median : 7.000
##  Mean   :0.007542   Mean   :0.2343   Mean   :3.409   Mean   : 7.657
##  3rd Qu.:0.000000   3rd Qu.:0.0000   3rd Qu.:4.000   3rd Qu.: 8.000
##  Max.   :1.000000   Max.   :4.0000   Max.   :5.000   Max.   :13.000
##    sqft_above   sqft_basement       yr_built     yr_renovated
##  Min.   : 290   Min.   :   0.0   Min.   :1900   Min.   :   0.0
##  1st Qu.:1190   1st Qu.:   0.0   1st Qu.:1951   1st Qu.:   0.0
##  Median :1560   Median :   0.0   Median :1975   Median :   0.0
##  Mean   :1788   Mean   : 291.5   Mean   :1971   Mean   :  84.4
##  3rd Qu.:2210   3rd Qu.: 560.0   3rd Qu.:1997   3rd Qu.:   0.0
##  Max.   :9410   Max.   :4820.0   Max.   :2015   Max.   :2015.0
##     zipcode           lat             long        sqft_living15
##  Min.   :98001   Min.   :47.16   Min.   :-122.5   Min.   : 399
##  1st Qu.:98033   1st Qu.:47.47   1st Qu.:-122.3   1st Qu.:1490
##  Median :98065   Median :47.57   Median :-122.2   Median :1840
##  Mean   :98078   Mean   :47.56   Mean   :-122.2   Mean   :1987
##  3rd Qu.:98118   3rd Qu.:47.68   3rd Qu.:-122.1   3rd Qu.:2360
##  Max.   :98199   Max.   :47.78   Max.   :-121.3   Max.   :6210
##    sqft_lot15
##  Min.   :   651
##  1st Qu.:  5100
##  Median :  7620
##  Mean   : 12768
##  3rd Qu.: 10083
##  Max.   :871200
```

```
str(fulldata)
```

```
## 'data.frame':    21613 obs. of  21 variables:
##  $ id           : num  7.13e+09 6.41e+09 5.63e+09 2.49e+09 1.95e+09 ...
##  $ date         : chr  "20141013T000000" "20141209T000000" "20150225T000000" "2014120
9T000000" ...
##  $ price        : num  221900 538000 180000 604000 510000 ...
##  $ 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 ...
```

```
# Dropping categorical variable and id
fulldata <- fulldata %>%
  dplyr::select(-id, -date)
```

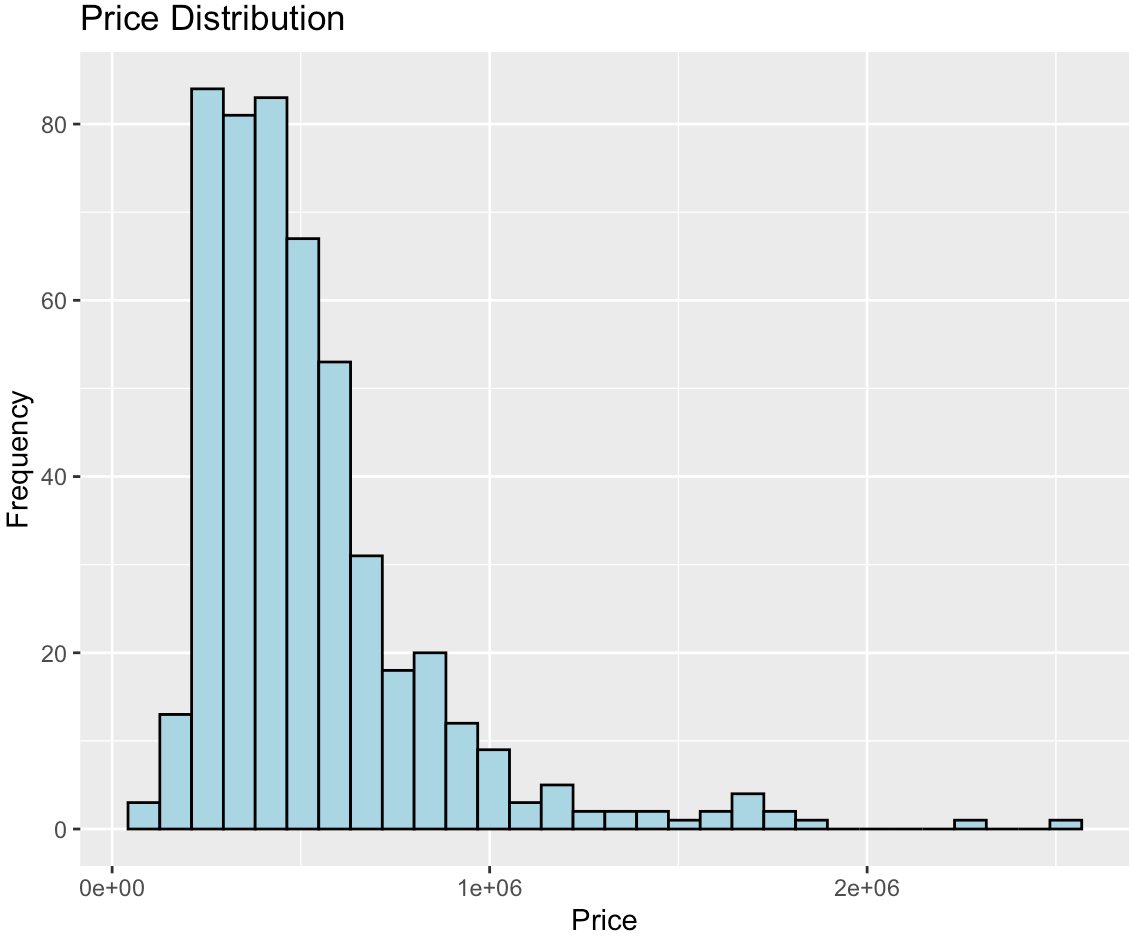
```
# Check for missing values
summarize_missing <- function(fulldata) {
  sapply(fulldata, function(x) sum(is.na(x)))
}
summarize_missing(fulldata)
```

```
##         price      bedrooms     bathrooms   sqft_living      sqft_lot
##             0             0             0             0             0
##        floors    waterfront          view     condition         grade
##             0             0             0             0             0
##    sqft_above sqft_basement      yr_built  yr_renovated       zipcode
##             0             0             0             0             0
##           lat          long sqft_living15    sqft_lot15
##             0             0             0             0
```

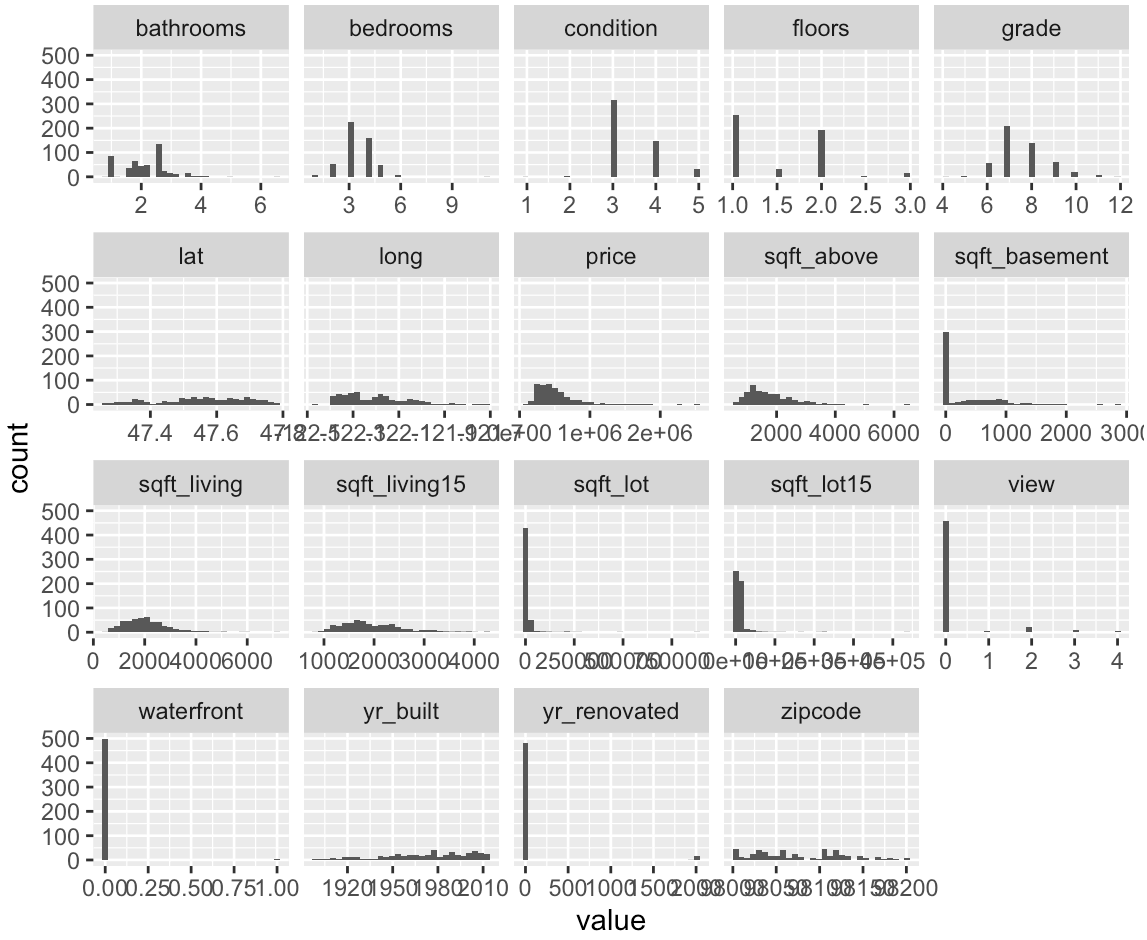
```
# Set the seed for reproducibility
set.seed(42)

# Randomly sample 500 observations for the training set
data <- fulldata %>%
  sample_n(500)
```

Exploratory Data Analysis

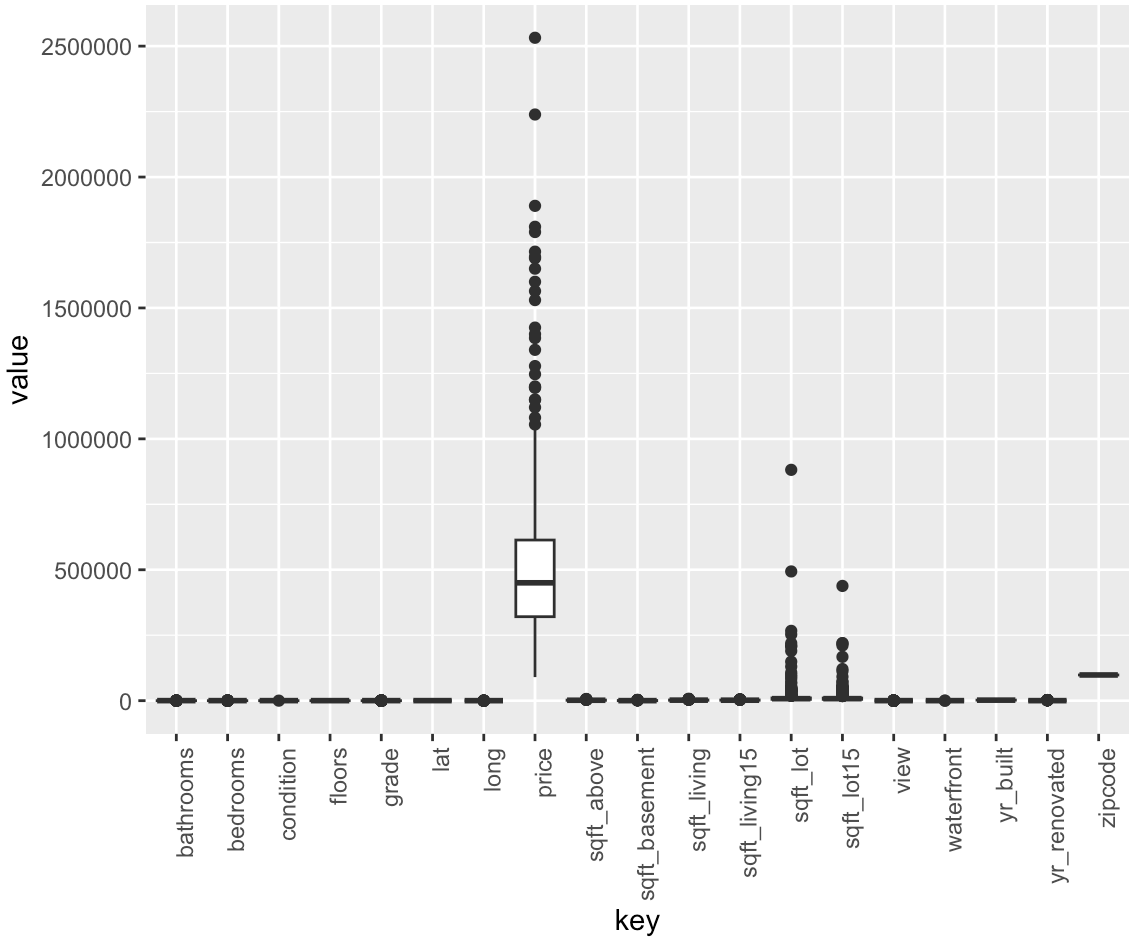
```
# Distribution of prices (target variable)
ggplot(data, aes(x = price)) +
  geom_histogram(bins = 30, color = "black", fill = "lightblue") +
  labs(title = "Price Distribution", x = "Price", y = "Frequency")
```



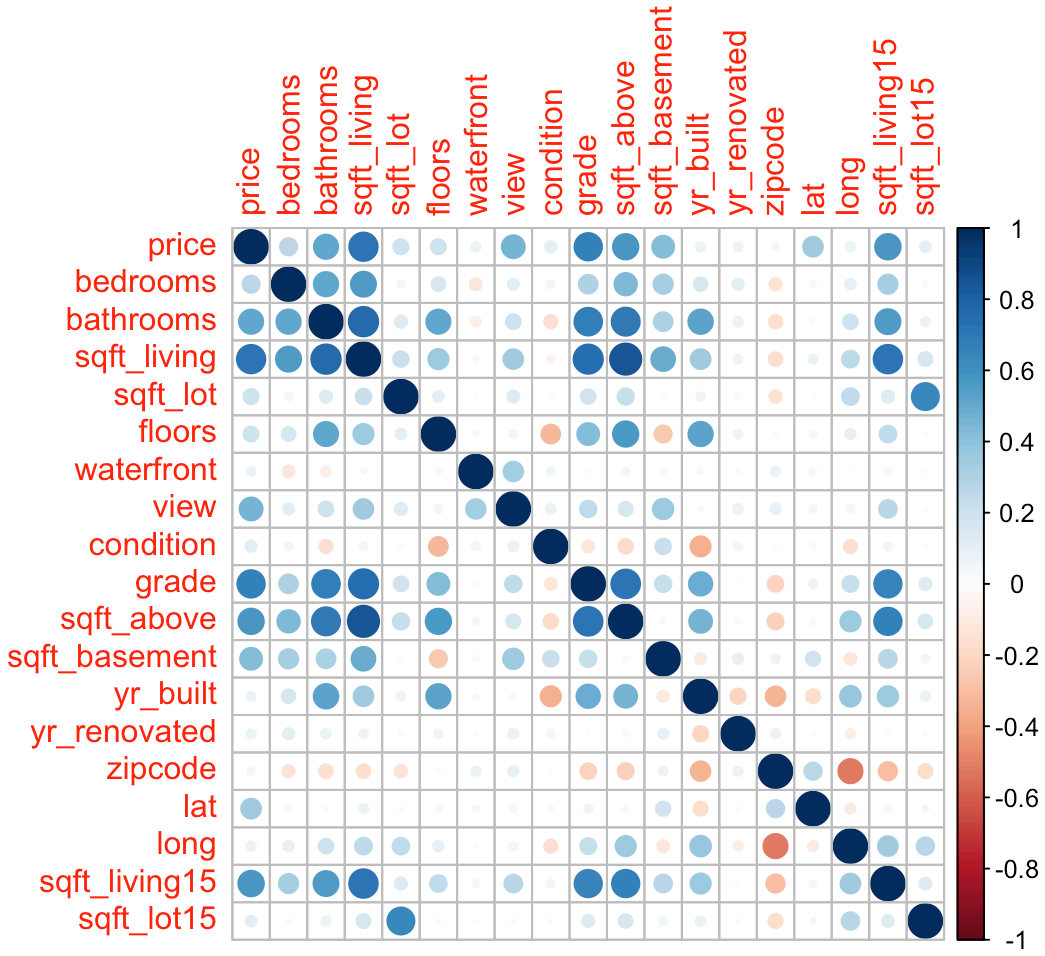

```
# Plotting histograms for each numerical variable
data %>%
  keep(is.numeric) %>%
  gather() %>%
  ggplot(aes(value)) +
    geom_histogram(bins = 30) +
    facet_wrap(~key, scales = 'free_x')
```




```
# Boxplot to check for outliers
data %>%
  keep(is.numeric) %>%
  gather() %>%
  ggplot(aes(x = key, y = value)) +
    geom_boxplot() +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```
# Correlation matrix and visualization
correlation_matrix <- cor(data)
corrplot(correlation_matrix, method = "circle")
```



Model Building

```
# Building a linear regression model (baseline)
model0 <- lm(price ~ ., data = data)
summary(model0)
```

```
##
## Call:
## lm(formula = price ~ ., data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -504791  -89882  -11526   71471  928609
##
## Coefficients: (1 not defined because of singularities)
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -1.193e+06  1.652e+07  -0.072 0.942466
## bedrooms      -3.889e+04  9.615e+03  -4.045 6.10e-05 ***
## bathrooms      4.628e+04  1.785e+04   2.593 0.009815 **
## sqft_living    1.124e+02  2.455e+01   4.579 5.95e-06 ***
## sqft_lot       3.999e-01  1.880e-01   2.127 0.033936 *
## floors        -1.066e+04  1.990e+04  -0.535 0.592595
## waterfront     3.997e+04  1.245e+05   0.321 0.748283
## view           8.712e+04  1.219e+04   7.145 3.35e-12 ***
## condition      4.818e+04  1.288e+04   3.742 0.000205 ***
## grade          8.975e+04  1.116e+04   8.042 6.92e-15 ***
## sqft_above     3.084e+01  2.442e+01   1.263 0.207265
## sqft_basement         NA         NA      NA       NA
## yr_built      -2.204e+03  3.906e+02  -5.642 2.87e-08 ***
## yr_renovated  -4.614e+00  2.148e+01  -0.215 0.830005
## zipcode       -3.499e+02  1.814e+02  -1.929 0.054261 .
## lat            6.132e+05  5.821e+04  10.534  < 2e-16 ***
## long          -8.232e+04  6.688e+04  -1.231 0.218941
## sqft_living15  3.198e+01  1.767e+01   1.810 0.070858 .
## sqft_lot15    -6.837e-01  3.412e-01  -2.004 0.045645 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 160700 on 482 degrees of freedom
## Multiple R-squared:  0.7407, Adjusted R-squared:  0.7315
## F-statistic: 80.97 on 17 and 482 DF,  p-value: < 2.2e-16
```

```
# Building a linear regression model (baseline)
model0 <- lm(scale(data$price) ~ ., data = data)
predictions_model0 <- predict(model0, newdata = data)

# Calculate MSE, RMSE and R-squared
Y_0 <- scale(data$price)
mse_0 <- mean((Y_0 - predictions_model0)^2)
rmse_0 <-sqrt(mse_0)
total_ss_0 <- sum((Y_0 - mean(Y_0))^2)
residual_ss_0 <- sum((Y_0 - predictions_model0)^2)
r_squared_0 <- 1 - (residual_ss_0 / total_ss_0)

print(paste("MSE for Baseline Model: ", mse_0))
```

```
## [1] "MSE for Baseline Model:  0.258824896823843"
```

```
print(paste("RMSE for Baseline Model: ", rmse_0))
```

```
## [1] "RMSE for Baseline Model:  0.508748362969202"
```

```
print(paste("R-squared for Baseline Model: ", r_squared_0))
```

```
## [1] "R-squared for Baseline Model:  0.740656416008172"
```

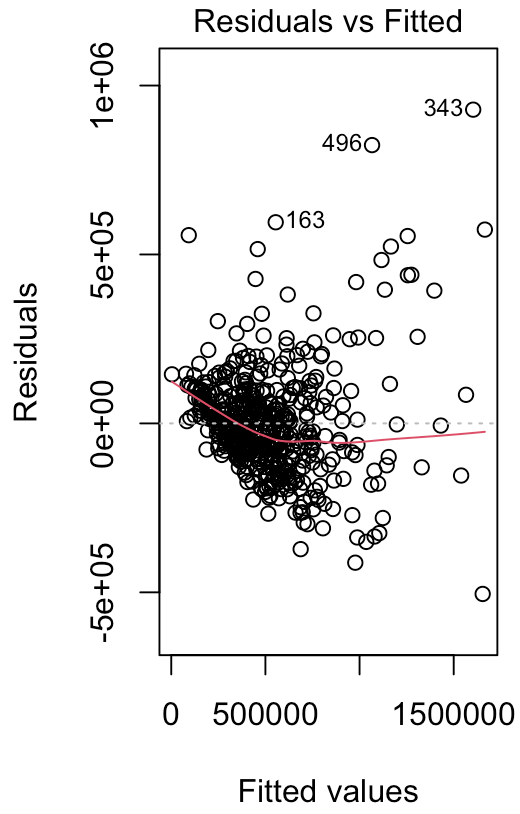
```
# Identify potential issues with multicollinearity or aliasing among predictors
alias_matrix <- alias(model0)
print(alias_matrix)
```

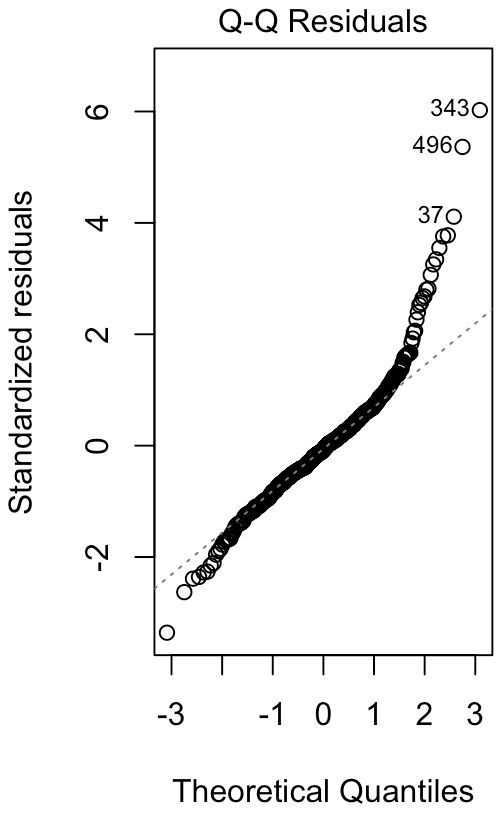
```
## Model :
## scale(data$price) ~ bedrooms + bathrooms + sqft_living + sqft_lot +
##     floors + waterfront + view + condition + grade + sqft_above +
##     sqft_basement + yr_built + yr_renovated + zipcode + lat +
##     long + sqft_living15 + sqft_lot15
##
## Complete :
##               (Intercept) bedrooms bathrooms sqft_living sqft_lot floors
## sqft_basement  0           0        0         1           0        0
##               waterfront view condition grade sqft_above yr_built yr_renovated
## sqft_basement  0          0    0         0    -1          0        0
##               zipcode lat long sqft_living15 sqft_lot15
## sqft_basement  0       0   0    0             0
```

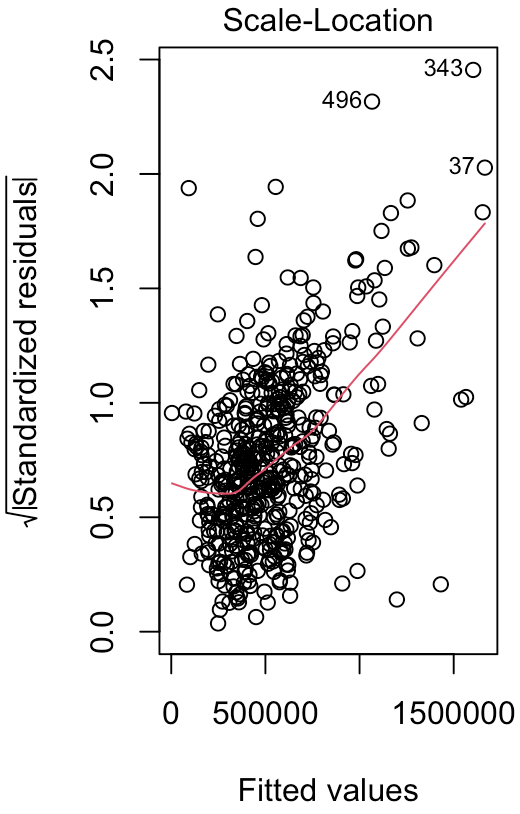
```
# Removing the 'sqft_basement' column to potentially address issues identified in the al
ias matrix
data <- data %>% dplyr::select(-sqft_basement)
```

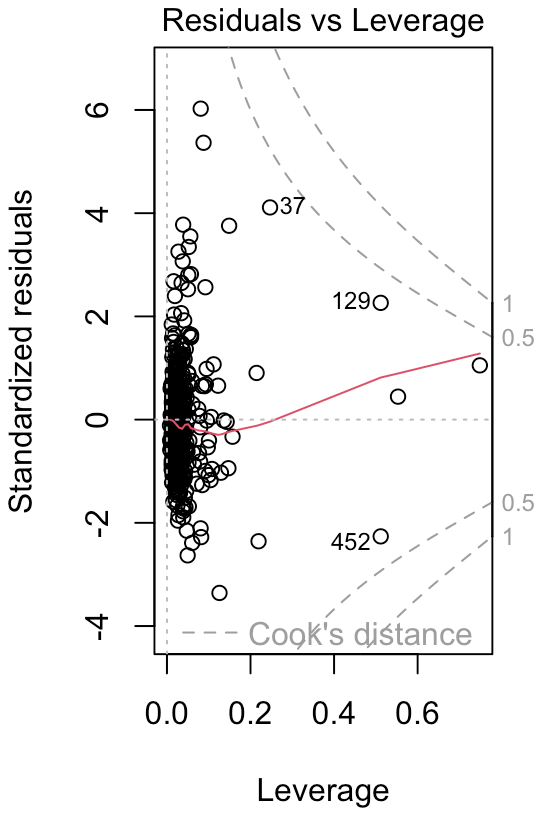
```
# Re-fit the model
model_reduced <- lm(price ~ ., data = data)
```

```
# Model Diagnostic Plots
par(mfrow = c(1, 2))
plot(model_reduced)
```


Residuals vs Fitted
Residuals
Fitted values
343
496
163
1e+06
5e+05
0e+00
-5e+05
0
500000
1500000


Q-Q Residuals
Standardized residuals
Theoretical Quantiles
343
496
37
6
4
2
0
-2
-3
-1
0
1
2
3


Scale-Location
√|Standardized residuals|
Fitted values
343
496
37
2.5
2.0
1.5
1.0
0.5
0.0
0
500000
1500000


Residuals vs Leverage
Standardized residuals
Leverage
37
129
452
1
0.5
0.5
1
Cook's distance
6
4
2
0
-2
-4
0.0
0.2
0.4
0.6

```
# Model Diagnostic Tests
bptest<-bptest(model_reduced)
swtest<-shapiro.test(resid(model_reduced))
cat("To test equal variance, we use bptest, Pvalue =", bptest$p.value, "<0.05, the equal variance assumption violates", "\n",
    "To test normality, we use swtest, Pvalue =", swtest$p.value, "<0.05, the normality assumption violates", "\n")
```

```
## To test equal variance, we use bptest, Pvalue = 2.375839e-18 <0.05, the equal variance assumption violates
##  To test normality, we use swtest, Pvalue = 3.233236e-15 <0.05, the normality assumption violates
```

```
# Identify variables with high multicollinearity
vif_values <- vif(model_reduced)
print(vif_values)
```

```
##      bedrooms     bathrooms   sqft_living      sqft_lot        floors
##      1.689435      3.631730      9.262440      1.857959      2.357172
##    waterfront          view     condition         grade    sqft_above
##      1.194509      1.483284      1.282169      3.091374      6.933765
##      yr_built  yr_renovated       zipcode           lat          long
##      2.517926      1.168554      1.657479      1.180665      1.763118
## sqft_living15    sqft_lot15
##      2.496633      1.822306
```

```
# Apply a cutoff of 5 to VIF values to identify variables to consider for removal
high_vif <- names(vif_values[vif_values > 5])
cat("Variables with VIF > 5, suggesting removal due to high multicollinearity:", high_vif)
```

```
## Variables with VIF > 5, suggesting removal due to high multicollinearity: sqft_living sqft_above
```

```
# Remove sqft_living sqft_above and refit the model
data <- data %>% dplyr::select(-sqft_above, -sqft_living)
model_reduced <- lm(price ~ ., data = data)

vif_values <- vif(model_reduced)
print(vif_values)
```

```
##      bedrooms     bathrooms      sqft_lot        floors    waterfront
##      1.474137      2.939655      1.846508      1.727950      1.192166
##          view     condition         grade      yr_built  yr_renovated
##      1.350286      1.267432      2.561011      2.418154      1.159260
##       zipcode           lat          long sqft_living15    sqft_lot15
##      1.639821      1.153652      1.654328      2.123999      1.795214
```

```
# Perform stepwise selection
model_step <- step(lm(price ~ ., data = data), direction = "both")
```

```
## Start:  AIC=12053.89
## price ~ bedrooms + bathrooms + sqft_lot + floors + waterfront +
##     view + condition + grade + yr_built + yr_renovated + zipcode +
##     lat + long + sqft_living15 + sqft_lot15
##
##                 Df  Sum of Sq        RSS   AIC
## - waterfront     1 3.8525e+07 1.3837e+13 12052
## - long           1 2.2048e+08 1.3837e+13 12052
## - floors         1 2.4195e+09 1.3840e+13 12052
## - yr_renovated   1 1.2633e+10 1.3850e+13 12052
## - zipcode        1 4.1421e+10 1.3879e+13 12053
## - sqft_lot15     1 4.3174e+10 1.3880e+13 12053
## <none>                        1.3837e+13 12054
## - bedrooms       1 6.5278e+10 1.3902e+13 12054
## - sqft_lot       1 1.4012e+11 1.3977e+13 12057
## - condition      1 3.4881e+11 1.4186e+13 12064
## - sqft_living15  1 6.5179e+11 1.4489e+13 12075
## - bathrooms      1 9.1509e+11 1.4752e+13 12084
## - yr_built       1 1.3505e+12 1.5188e+13 12098
## - view           1 1.9626e+12 1.5800e+13 12118
## - lat            1 2.9629e+12 1.6800e+13 12149
## - grade          1 3.8144e+12 1.7652e+13 12174
##
## Step:  AIC=12051.89
## price ~ bedrooms + bathrooms + sqft_lot + floors + view + condition +
##     grade + yr_built + yr_renovated + zipcode + lat + long +
##     sqft_living15 + sqft_lot15
##
##                 Df  Sum of Sq        RSS   AIC
## - long           1 2.0994e+08 1.3837e+13 12050
## - floors         1 2.4291e+09 1.3840e+13 12050
## - yr_renovated   1 1.2637e+10 1.3850e+13 12050
## - zipcode        1 4.1480e+10 1.3879e+13 12051
## - sqft_lot15     1 4.3137e+10 1.3880e+13 12051
## <none>                        1.3837e+13 12052
## - bedrooms       1 6.6308e+10 1.3904e+13 12052
## + waterfront     1 3.8525e+07 1.3837e+13 12054
## - sqft_lot       1 1.4021e+11 1.3977e+13 12055
## - condition      1 3.5016e+11 1.4187e+13 12062
## - sqft_living15  1 6.5319e+11 1.4490e+13 12073
## - bathrooms      1 9.2174e+11 1.4759e+13 12082
## - yr_built       1 1.3525e+12 1.5190e+13 12096
## - view           1 2.2307e+12 1.6068e+13 12125
## - lat            1 2.9756e+12 1.6813e+13 12147
## - grade          1 3.8144e+12 1.7652e+13 12172
##
## Step:  AIC=12049.9
## price ~ bedrooms + bathrooms + sqft_lot + floors + view + condition +
##     grade + yr_built + yr_renovated + zipcode + lat + sqft_living15 +
##     sqft_lot15
##
##                 Df  Sum of Sq        RSS   AIC
```

```
## - floors         1 2.5176e+09 1.3840e+13 12048
## - yr_renovated   1 1.2605e+10 1.3850e+13 12048
## - sqft_lot15     1 4.3997e+10 1.3881e+13 12050
## - zipcode        1 4.5646e+10 1.3883e+13 12050
## <none>                        1.3837e+13 12050
## - bedrooms       1 6.6387e+10 1.3904e+13 12050
## + long           1 2.0994e+08 1.3837e+13 12052
## + waterfront     1 2.7984e+07 1.3837e+13 12052
## - sqft_lot       1 1.4105e+11 1.3979e+13 12053
## - condition      1 3.5702e+11 1.4194e+13 12061
## - sqft_living15  1 6.7842e+11 1.4516e+13 12072
## - bathrooms      1 9.2478e+11 1.4762e+13 12080
## - yr_built       1 1.4082e+12 1.5246e+13 12096
## - view           1 2.2424e+12 1.6080e+13 12123
## - lat            1 2.9804e+12 1.6818e+13 12145
## - grade          1 3.8423e+12 1.7680e+13 12170
##
## Step:  AIC=12047.99
## price ~ bedrooms + bathrooms + sqft_lot + view + condition +
##     grade + yr_built + yr_renovated + zipcode + lat + sqft_living15 +
##     sqft_lot15
##
##                 Df  Sum of Sq        RSS   AIC
## - yr_renovated   1 1.1411e+10 1.3851e+13 12046
## - zipcode        1 4.3312e+10 1.3883e+13 12048
## - sqft_lot15     1 4.6233e+10 1.3886e+13 12048
## <none>                        1.3840e+13 12048
## - bedrooms       1 6.7830e+10 1.3908e+13 12048
## + floors         1 2.5176e+09 1.3837e+13 12050
## + long           1 2.9839e+08 1.3840e+13 12050
## + waterfront     1 3.4296e+07 1.3840e+13 12050
## - sqft_lot       1 1.4682e+11 1.3987e+13 12051
## - condition      1 3.5528e+11 1.4195e+13 12059
## - sqft_living15  1 6.7596e+11 1.4516e+13 12070
## - bathrooms      1 9.9653e+11 1.4837e+13 12081
## - yr_built       1 1.5292e+12 1.5369e+13 12098
## - view           1 2.2400e+12 1.6080e+13 12121
## - lat            1 2.9791e+12 1.6819e+13 12144
## - grade          1 3.9035e+12 1.7743e+13 12170
##
## Step:  AIC=12046.4
## price ~ bedrooms + bathrooms + sqft_lot + view + condition +
##     grade + yr_built + zipcode + lat + sqft_living15 + sqft_lot15
##
##                 Df  Sum of Sq        RSS   AIC
## - zipcode        1 4.2966e+10 1.3894e+13 12046
## - sqft_lot15     1 4.7631e+10 1.3899e+13 12046
## <none>                        1.3851e+13 12046
## - bedrooms       1 7.0806e+10 1.3922e+13 12047
## + yr_renovated   1 1.1411e+10 1.3840e+13 12048
## + floors         1 1.3239e+09 1.3850e+13 12048
## + long           1 2.3721e+08 1.3851e+13 12048
```

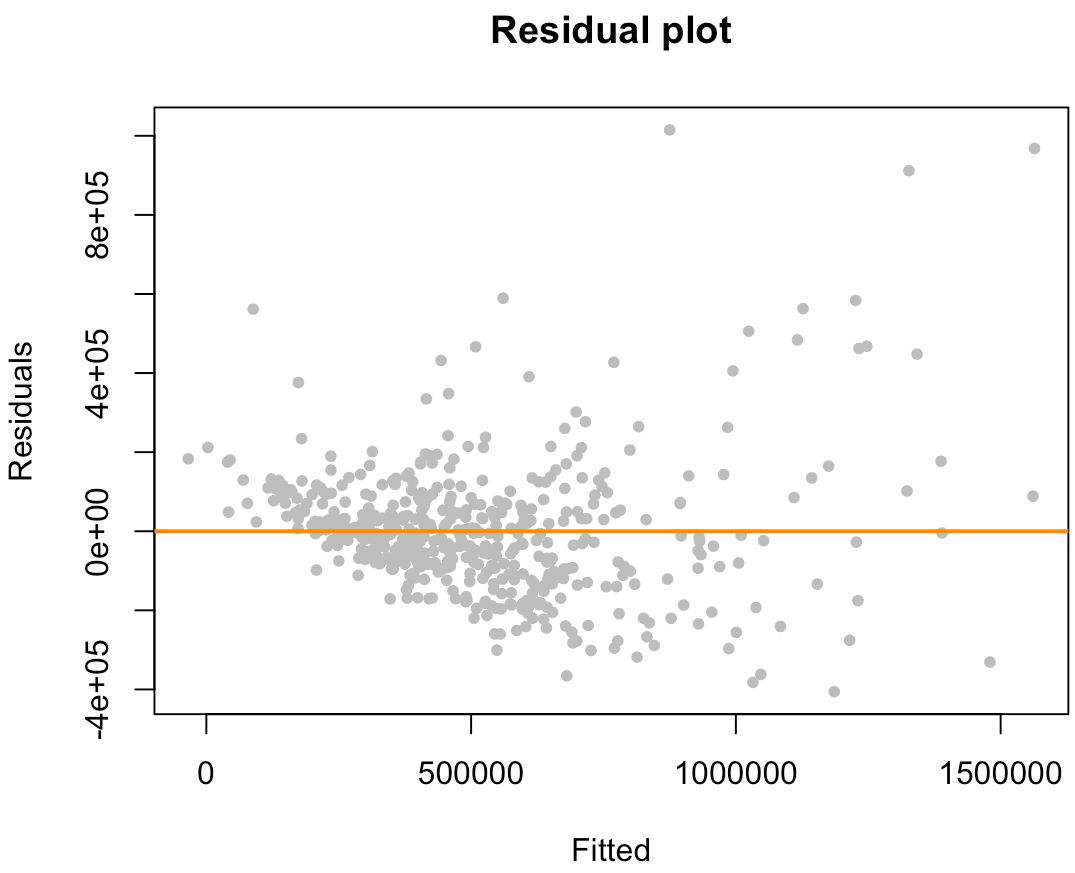
```
## + waterfront     1 3.7543e+07 1.3851e+13 12048
## - sqft_lot       1 1.4766e+11 1.3999e+13 12050
## - condition      1 3.8699e+11 1.4238e+13 12058
## - sqft_living15  1 6.7671e+11 1.4528e+13 12068
## - bathrooms      1 9.8785e+11 1.4839e+13 12079
## - yr_built       1 1.5968e+12 1.5448e+13 12099
## - view           1 2.2301e+12 1.6082e+13 12119
## - lat            1 3.0067e+12 1.6858e+13 12143
## - grade          1 3.9082e+12 1.7760e+13 12169
##
## Step:  AIC=12045.95
## price ~ bedrooms + bathrooms + sqft_lot + view + condition +
##     grade + yr_built + lat + sqft_living15 + sqft_lot15
##
##                 Df  Sum of Sq        RSS   AIC
## - sqft_lot15     1 3.9856e+10 1.3934e+13 12045
## <none>                        1.3894e+13 12046
## - bedrooms       1 6.3201e+10 1.3958e+13 12046
## + zipcode        1 4.2966e+10 1.3851e+13 12046
## + yr_renovated   1 1.1065e+10 1.3883e+13 12048
## + long           1 4.2885e+09 1.3890e+13 12048
## + waterfront     1 4.7318e+07 1.3894e+13 12048
## + floors         1 3.4732e+05 1.3894e+13 12048
## - sqft_lot       1 1.5405e+11 1.4048e+13 12050
## - condition      1 4.2528e+11 1.4320e+13 12059
## - sqft_living15  1 7.9740e+11 1.4692e+13 12072
## - bathrooms      1 9.5177e+11 1.4846e+13 12077
## - yr_built       1 1.5642e+12 1.5459e+13 12097
## - view           1 2.1884e+12 1.6083e+13 12117
## - lat            1 3.0162e+12 1.6911e+13 12142
## - grade          1 3.9329e+12 1.7827e+13 12169
##
## Step:  AIC=12045.38
## price ~ bedrooms + bathrooms + sqft_lot + view + condition +
##     grade + yr_built + lat + sqft_living15
##
##                 Df  Sum of Sq        RSS   AIC
## <none>                        1.3934e+13 12045
## - bedrooms       1 6.1959e+10 1.3996e+13 12046
## + sqft_lot15     1 3.9856e+10 1.3894e+13 12046
## + zipcode        1 3.5191e+10 1.3899e+13 12046
## + yr_renovated   1 1.2359e+10 1.3922e+13 12047
## + long           1 1.6716e+09 1.3933e+13 12047
## + floors         1 4.1365e+08 1.3934e+13 12047
## + waterfront     1 1.2605e+08 1.3934e+13 12047
## - sqft_lot       1 1.1900e+11 1.4053e+13 12048
## - condition      1 4.2072e+11 1.4355e+13 12058
## - sqft_living15  1 7.7373e+11 1.4708e+13 12070
## - bathrooms      1 9.6995e+11 1.4904e+13 12077
## - yr_built       1 1.6019e+12 1.5536e+13 12098
## - view           1 2.2654e+12 1.6200e+13 12119
```

```
## - lat            1 2.9847e+12 1.6919e+13 12140
## - grade          1 3.9503e+12 1.7885e+13 12168
```

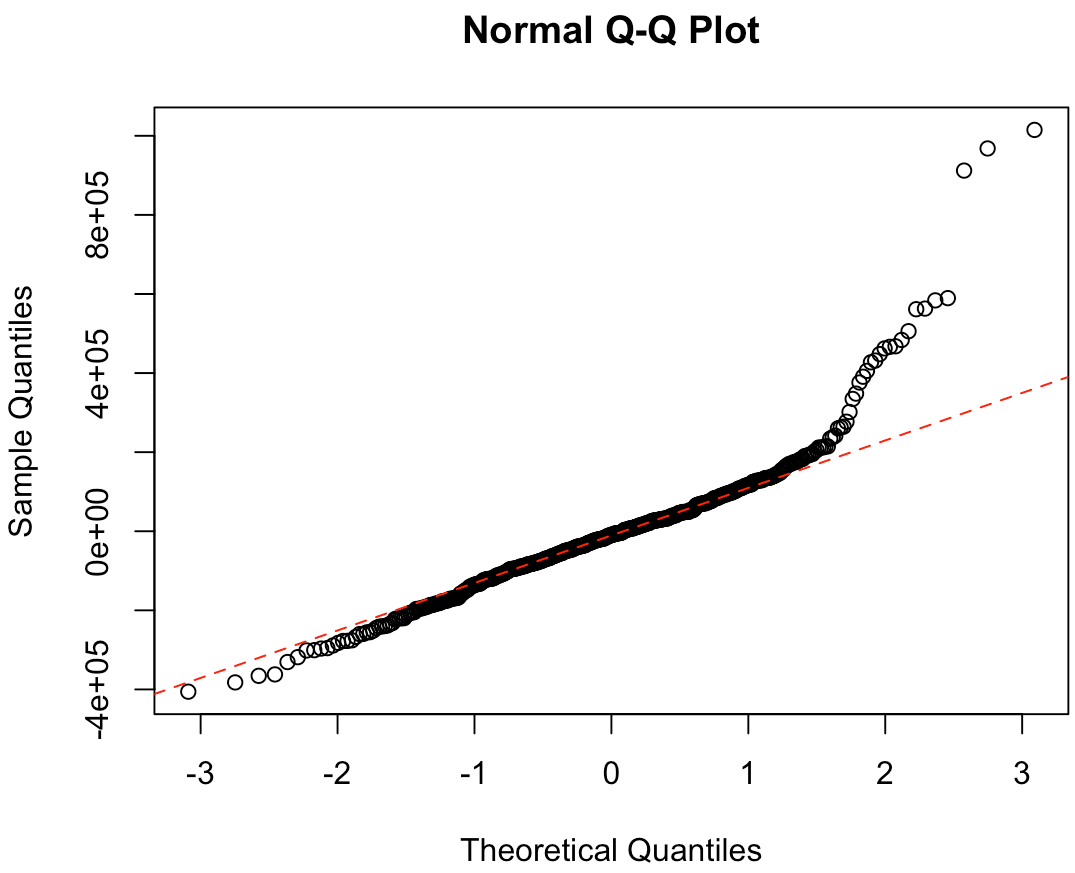
```
summary(model_step)
```

```
##
## Call:
## lm(formula = price ~ bedrooms + bathrooms + sqft_lot + view +
##     condition + grade + yr_built + lat + sqft_living15, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -405749  -91653   -9559   70408 1014928
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -2.414e+07  2.982e+06  -8.095 4.58e-15 ***
## bedrooms      -1.376e+04  9.321e+03  -1.476 0.140565
## bathrooms      9.335e+04  1.598e+04   5.840 9.50e-09 ***
## sqft_lot       3.031e-01  1.482e-01   2.046 0.041328 *
## view           1.002e+05  1.122e+04   8.925  < 2e-16 ***
## condition      4.973e+04  1.293e+04   3.846 0.000136 ***
## grade          1.245e+05  1.056e+04  11.786  < 2e-16 ***
## yr_built      -2.583e+03  3.442e+02  -7.505 2.91e-13 ***
## lat            5.950e+05  5.808e+04  10.245  < 2e-16 ***
## sqft_living15  8.433e+01  1.617e+01   5.216 2.70e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 168600 on 490 degrees of freedom
## Multiple R-squared:  0.7099, Adjusted R-squared:  0.7045
## F-statistic: 133.2 on 9 and 490 DF,  p-value: < 2.2e-16
```

```
#Check model assumptions
plot(fitted(model_step), resid(model_step), col = "grey", pch = 20,
     xlab = "Fitted", ylab = "Residuals", main = "Residual plot")
abline(h = 0, col = "darkorange", lwd = 2)
```



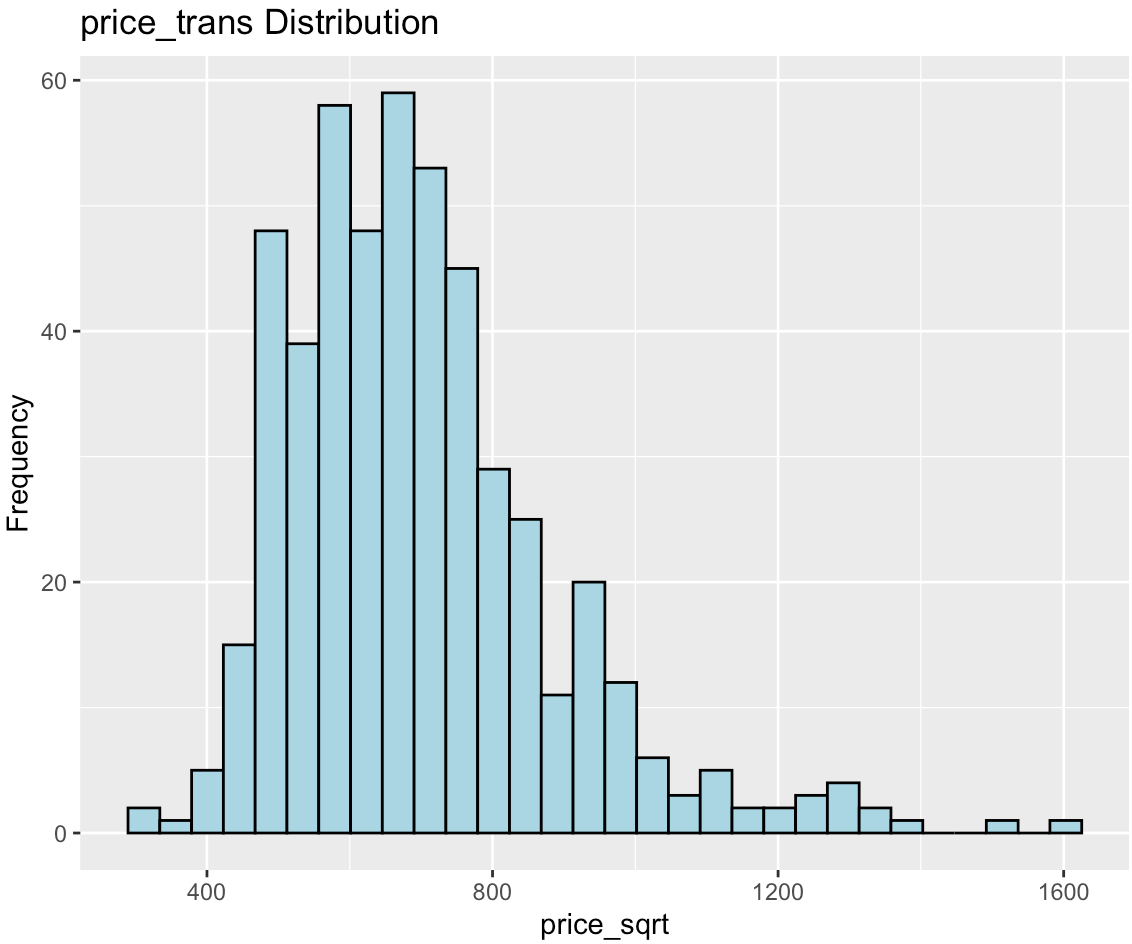

```
#Check model assumptions
qqnorm(resid(model_step),
       main = "Normal Q-Q Plot")
qqline(resid(model_step), col = "red", lty = 2)
```



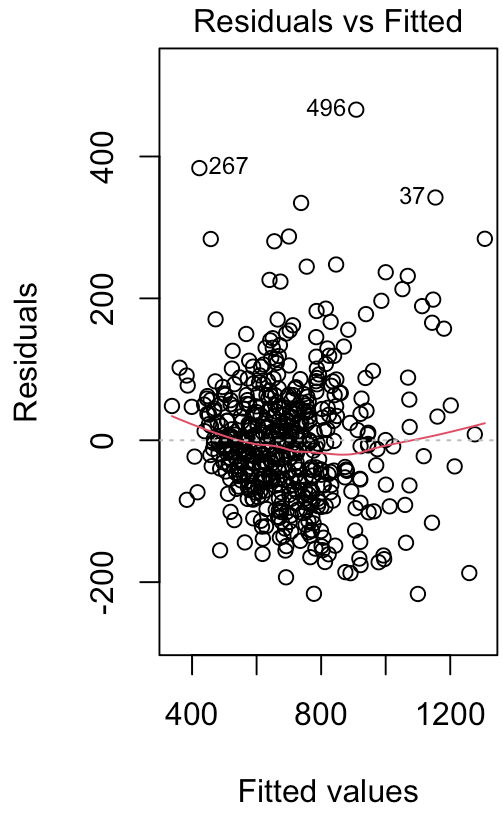

```
#Model Diagnostic Tests
swtest<-shapiro.test(resid(model_step))
bptest<-bptest(model_step)
cat("To test equal variance, we use bptest, Pvalue =",bptest$p.value,"<0.05, the equal v
ariance assumption violates","\n","To test normality , we use swtest, Pvalue =",swtest
$p.value,"<0.05, the normality violates","\n")
```

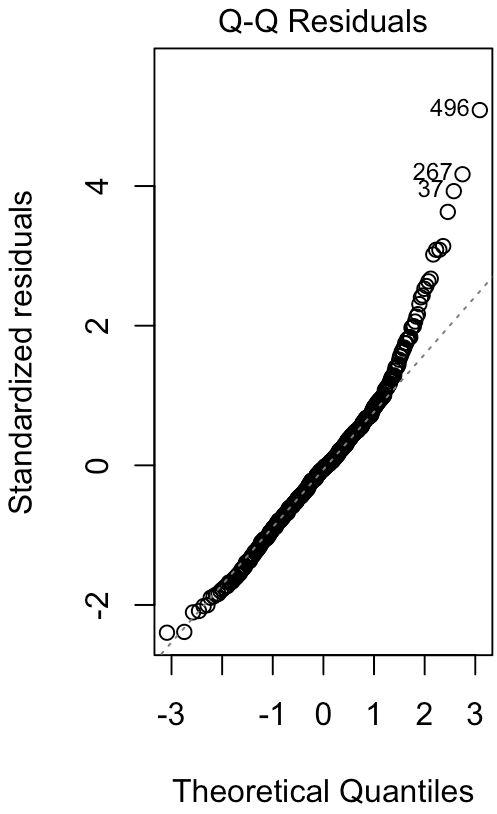
```
## To test equal variance, we use bptest, Pvalue = 3.430378e-15 <0.05, the equal varianc
e assumption violates
##  To test normality , we use swtest, Pvalue = 3.771701e-18 <0.05, the normality violat
es
```

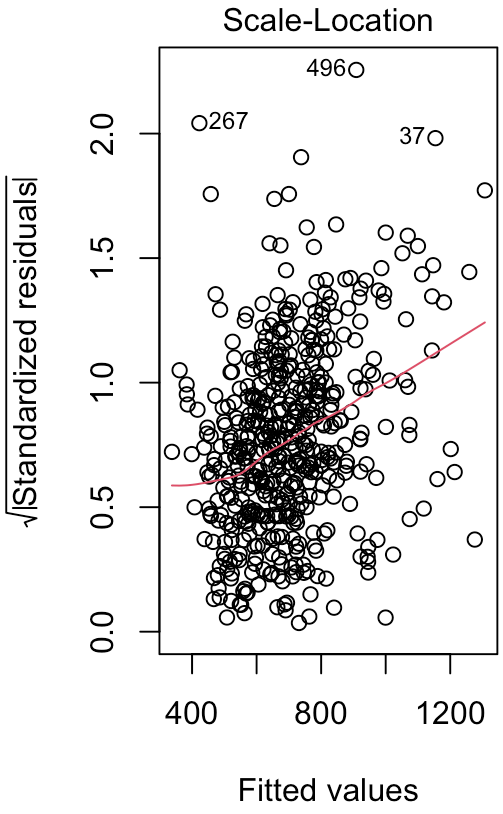
```
#Transform to sqrt
data$price_sqrt <- sqrt(data$price)
ggplot(data, aes(x = price_sqrt)) +
  geom_histogram(bins = 30, color = "black", fill = "lightblue") +
  labs(title = "price_trans Distribution", x = "price_sqrt", y = "Frequency")
```

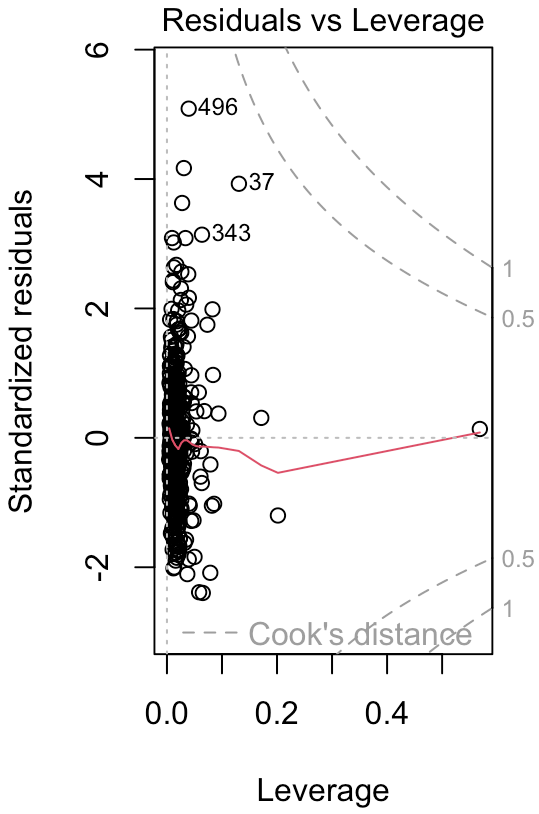



```
model_trans <- lm(price_sqrt ~ bedrooms + bathrooms + sqft_lot + view +
    condition + grade + yr_built + lat + sqft_living15, data = data)
par(mfrow = c(1, 2))
plot(model_trans)
```

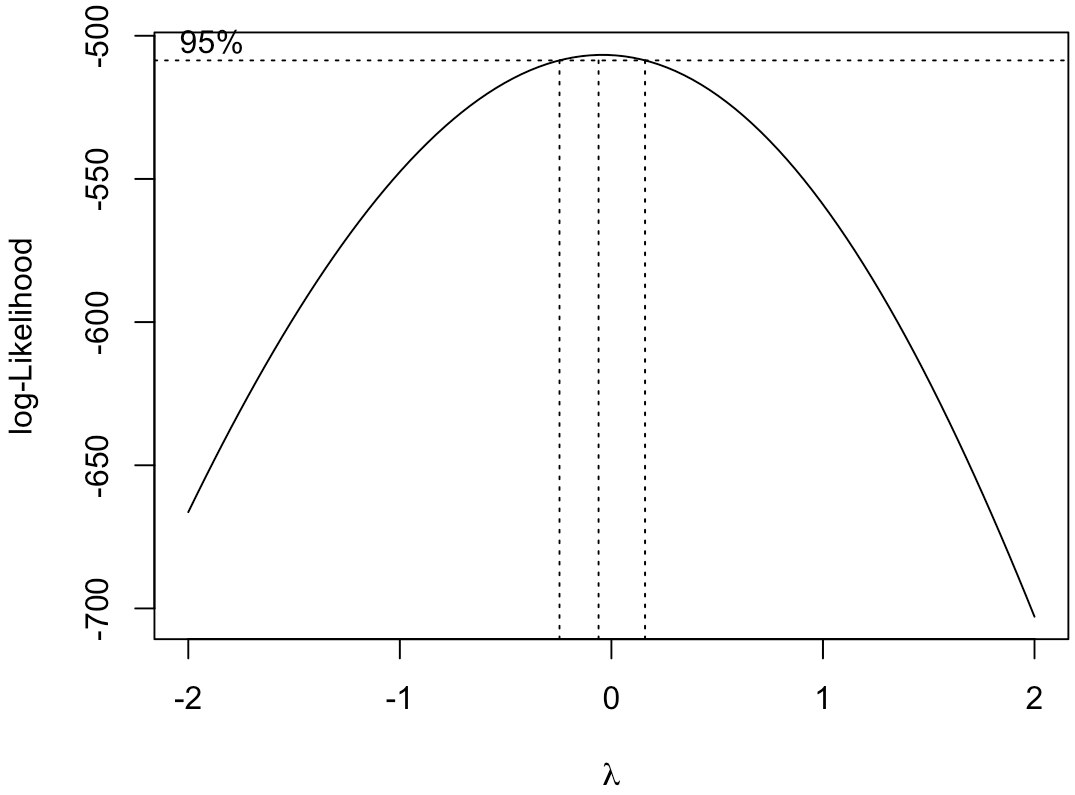

Residuals vs Fitted
Residuals
Fitted values
496
267
37
400
800
1200
-200
0
200
400


Q-Q Residuals
Standardized residuals
Theoretical Quantiles
496
267
37
-3
-1
0
1
2
3
-2
0
2
4


Scale-Location
√|Standardized residuals|
Fitted values
496
267
37
400
800
1200
0.0
0.5
1.0
1.5
2.0


Residuals vs Leverage
Standardized residuals
Leverage
496
37
343
1
0.5
0.5
1
Cook's distance
0.0
0.2
0.4
-2
0
2
4
6

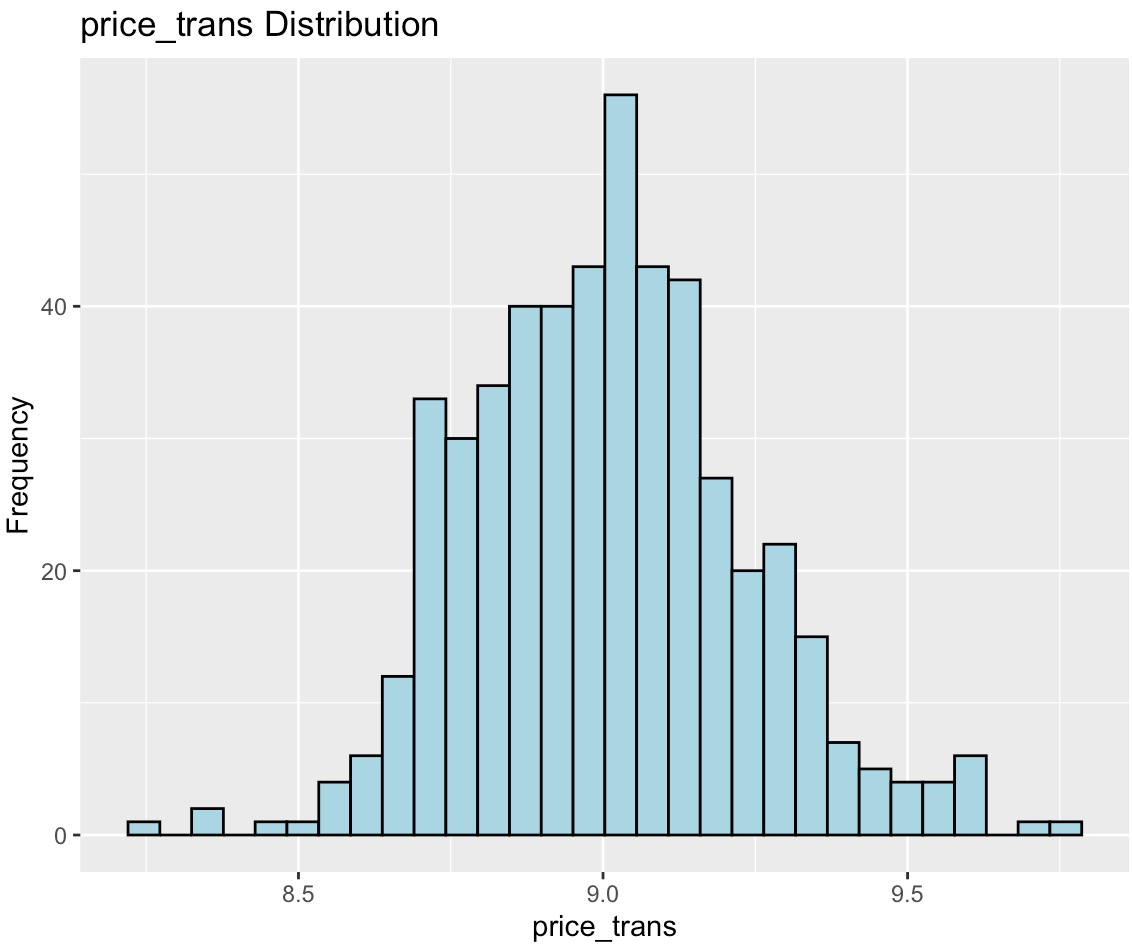
```
# Box-Cox Transform
boxcox_result <- boxcox(model_trans, lambda = seq(-2, 2, by = 0.1))
```



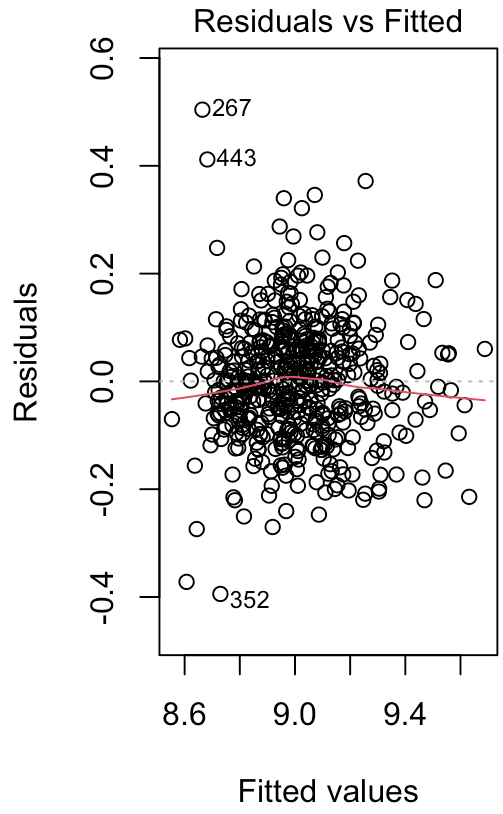

```
lambda <- boxcox_result$x[which.max(boxcox_result$y)]
cat("Optimal lambda:", lambda, "\n")
```

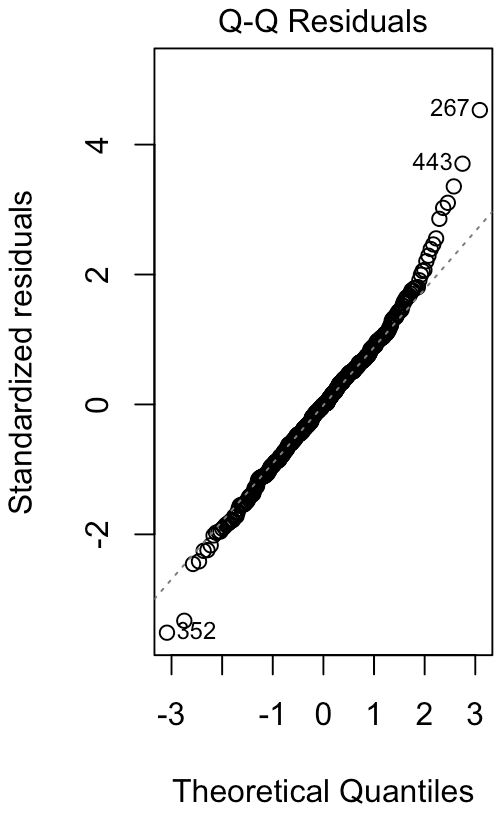
```
## Optimal lambda: -0.06060606
```

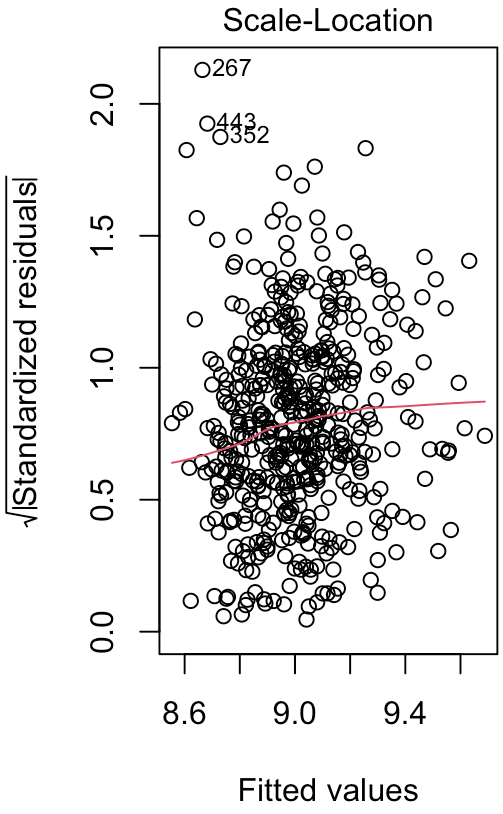
```
# Further transformation
data$price_trans <- (data$price^lambda - 1) / lambda
ggplot(data, aes(x = price_trans)) +
  geom_histogram(bins = 30, color = "black", fill = "lightblue") +
  labs(title = "price_trans Distribution", x = "price_trans", y = "Frequency")
```

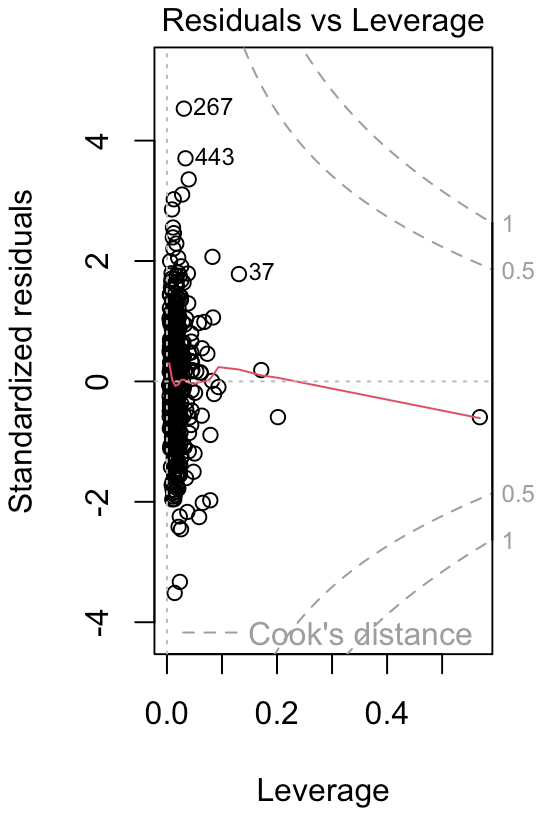



```
model_trans <- lm(price_trans ~ bedrooms + bathrooms + sqft_lot + view +
    condition + grade + yr_built + lat + sqft_living15, data = data)
par(mfrow = c(1, 2))
plot(model_trans)
```

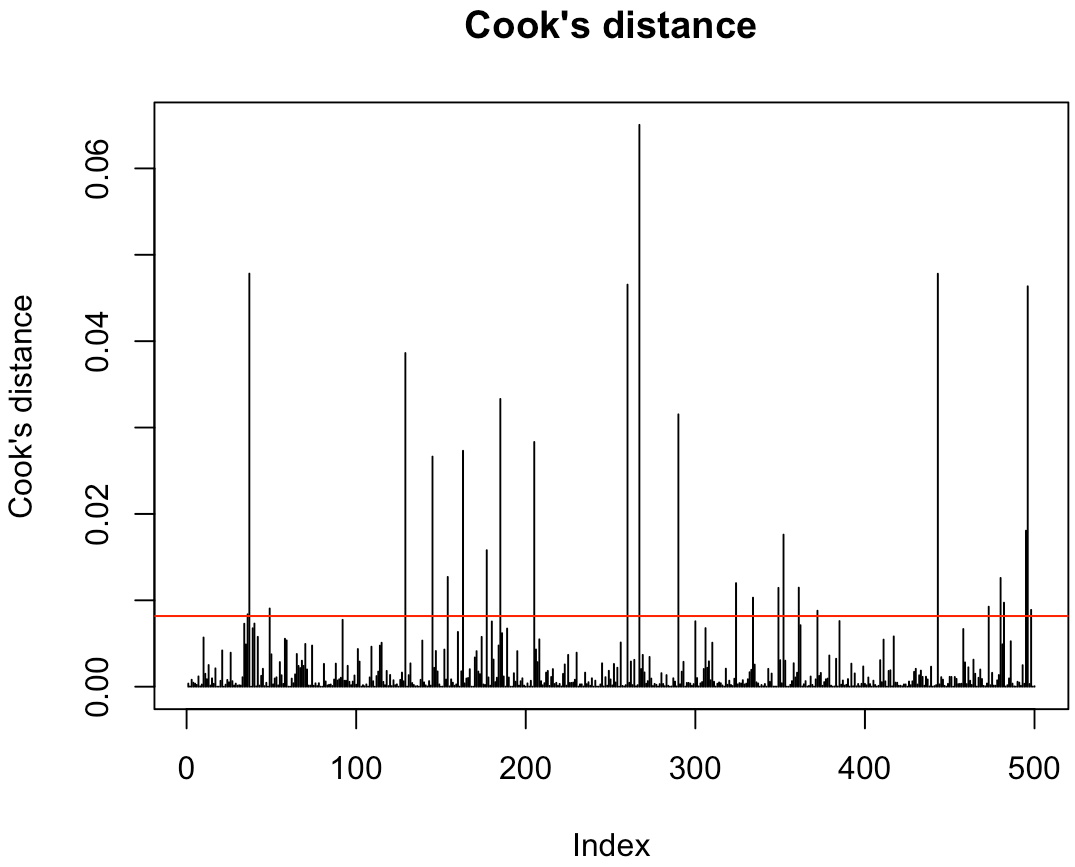

Residuals vs Fitted
Residuals
Fitted values
267
443
352
8.6
9.0
9.4
-0.4
-0.2
0.0
0.2
0.4
0.6


Q-Q Residuals
Standardized residuals
Theoretical Quantiles
267
443
352
-3
-1
0
1
2
3
-2
0
2
4

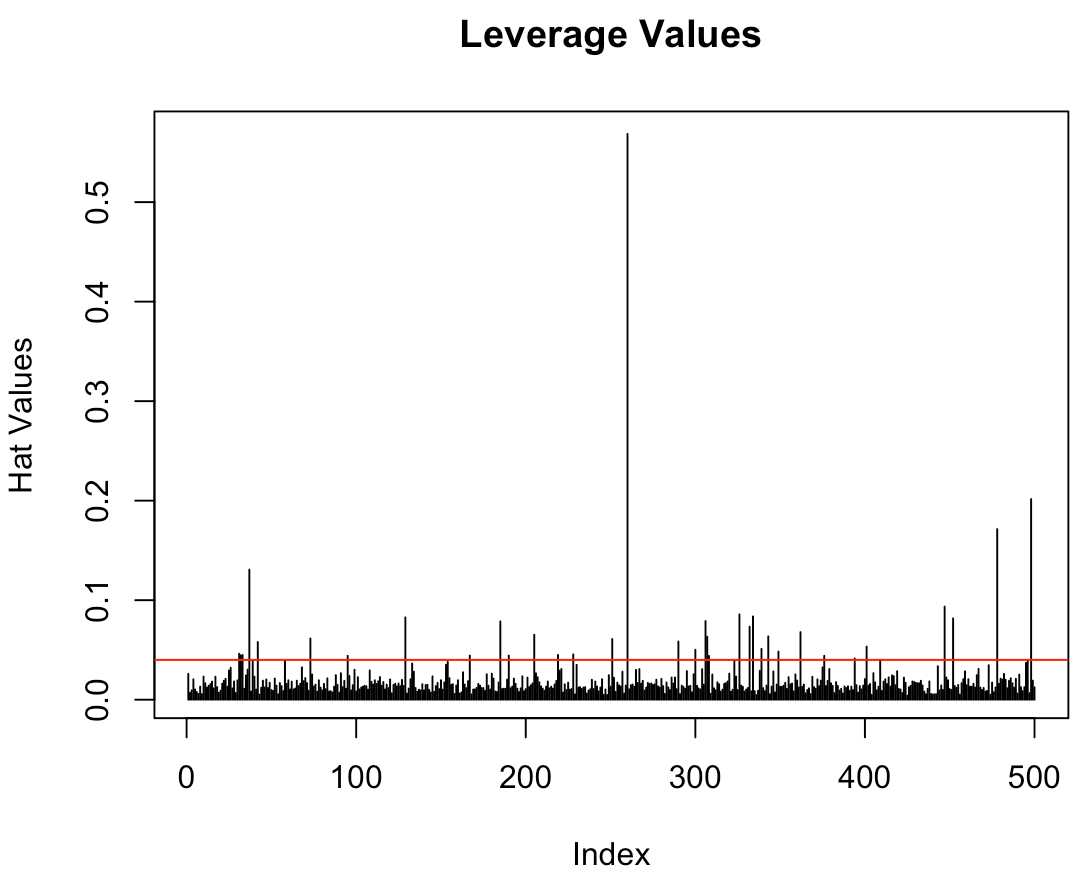

Scale-Location
√|Standardized residuals|
Fitted values
267
443
352
8.6
9.0
9.4
0.0
0.5
1.0
1.5
2.0


Residuals vs Leverage
Standardized residuals
Leverage
267
443
37
1
0.5
0.5
1
Cook's distance
0.0
0.2
0.4
-4
-2
0
2
4

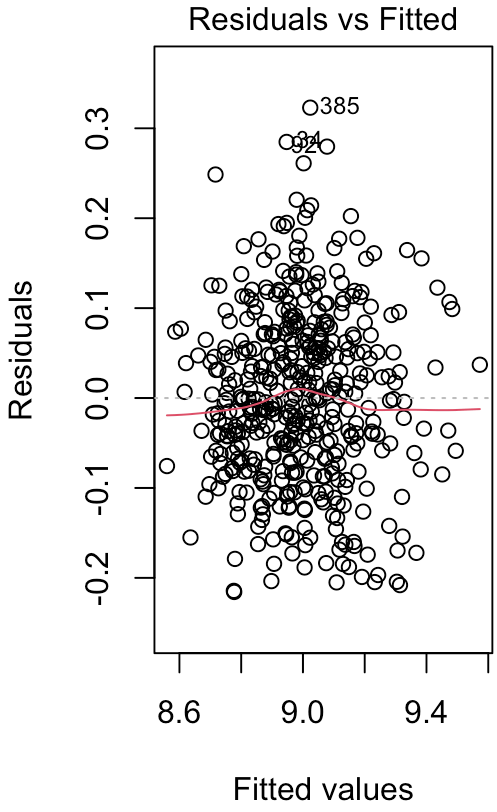
```
# Calculate Cook's distance to identify influential points
cooksD <- cooks.distance(model_trans)
# Cook's distance larger than 4/(n-k-1) to be considered as influential points
threshold <- 4 / (length(cooksD) - length(coef(model_trans)) - 1)
influential_points <- which(cooksD > threshold)

# Plot Cook's distance
plot(cooksD, type="h", main="Cook's distance", ylab="Cook's distance")
abline(h=threshold, col="red")
```

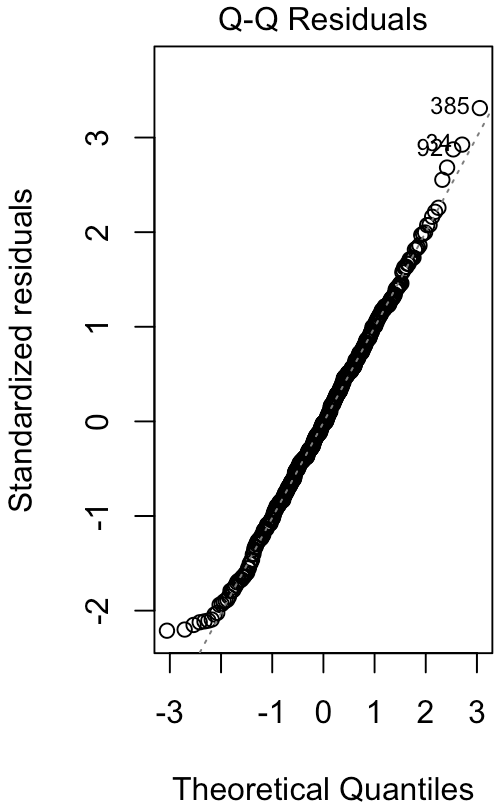


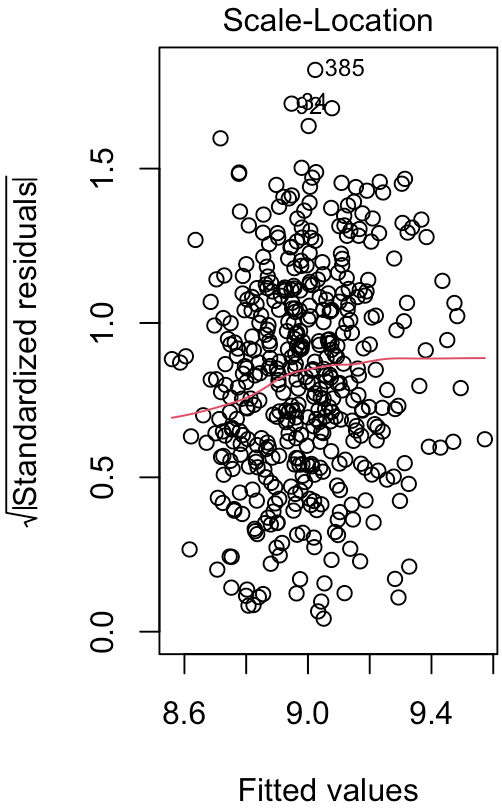

```
# Identify high leverage points
leverage <- hatvalues(model_trans)
# Leverage threshold of 2*k/n
lev_threshold <- 2 * length(coef(model_trans)) / length(leverage)
high_leverage_points <- which(leverage > lev_threshold)

# Plot leverage
plot(leverage, type="h", main="Leverage Values", ylab="Hat Values")
abline(h=lev_threshold, col="red")
```

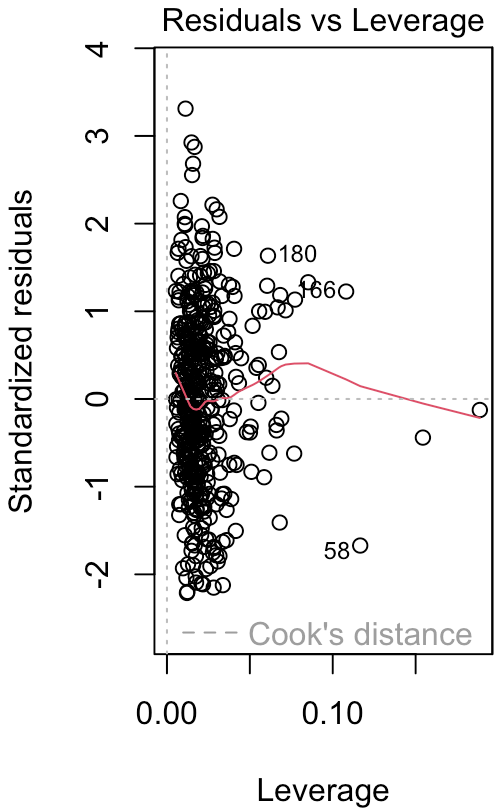



```
# Remove influential points based on Cook's distance or leverage
data <- data[-c(influential_points, high_leverage_points), ]

# Refit the model without the influential points
model_clean <- lm(price_trans ~ bedrooms + bathrooms + sqft_lot + view +
    condition + grade + yr_built + lat + sqft_living15, data=data)
par(mfrow=c(1,2))
plot(model_clean)
```


Residuals vs Fitted
Residuals
Fitted values
385
34
92
-0.2
-0.1
0.0
0.1
0.2
0.3
8.6
9.0
9.4


Q-Q Residuals
Standardized residuals
Theoretical Quantiles
385
34
92
-2
-1
0
1
2
3
-3
-1
0
1
2
3


Scale-Location
√|Standardized residuals|
Fitted values
385
34
92
0.0
0.5
1.0
1.5
8.6
9.0
9.4


Residuals vs Leverage
Standardized residuals
Leverage
180
166
58
Cook's distance
-2
-1
0
1
2
3
4
0.00
0.10

```
#Model Diagnostic Tests
shapiro.test(resid(model_clean))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(model_clean)
## W = 0.99509, p-value = 0.1682
```

```
bptest(model_clean)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  model_clean
## BP = 21.154, df = 9, p-value = 0.01198
```

```
# List of predictors
predictors <- c("bedrooms", "bathrooms", "sqft_lot", "view",
                "condition", "grade", "yr_built", "lat", "sqft_living15")

# Loop through each predictor and create a plot
for (pred in predictors) {
    p <- ggplot(data, aes_string(x = pred, y = "price_trans")) +
        geom_point(alpha = 0.5) +
        geom_smooth(method = "loess", color = "blue", se = FALSE) +
        labs(title = paste("Scatterplot of Price vs", pred),
             x = pred,
             y = "price_trans")
    print(p)
}
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## ℹ Please use tidy evaluation idioms with `aes()`.
## ℹ See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

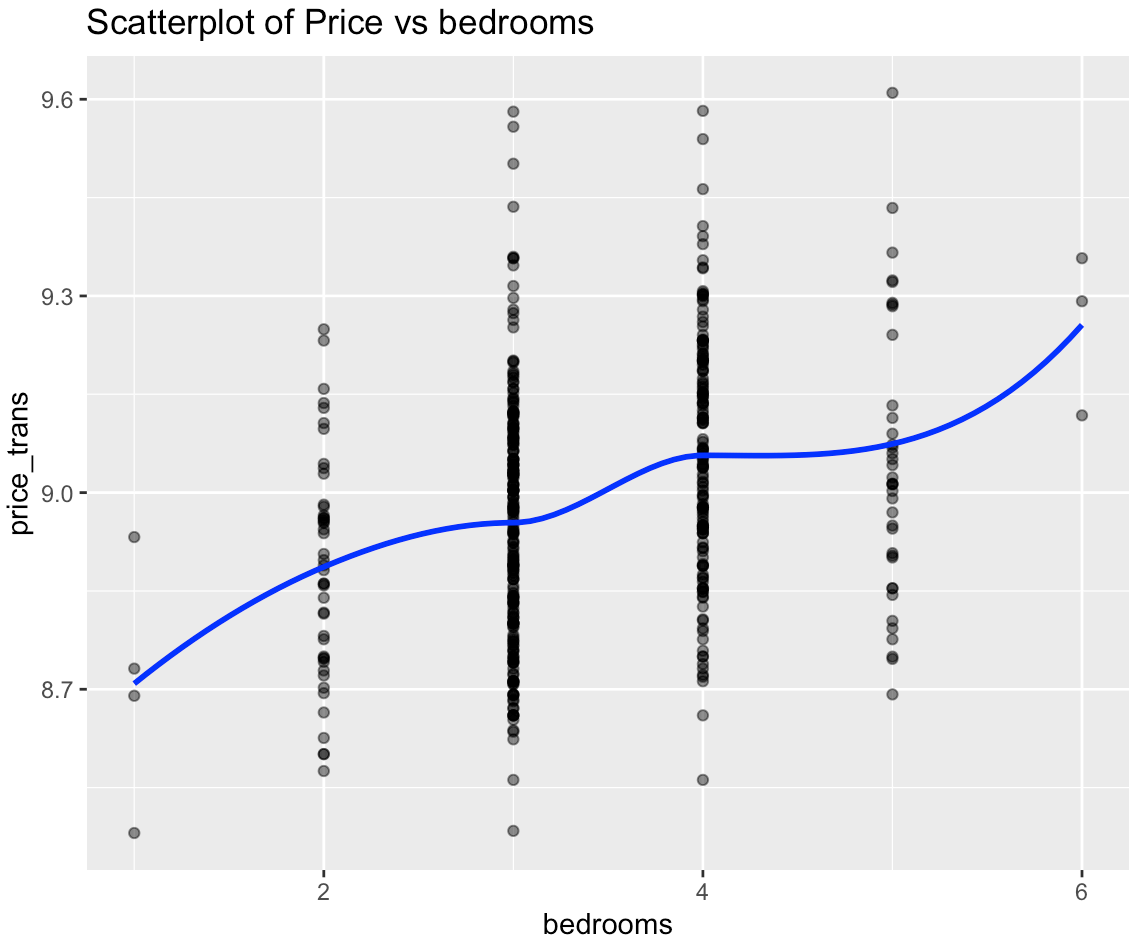
```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : pseudoinverse used at 3
```

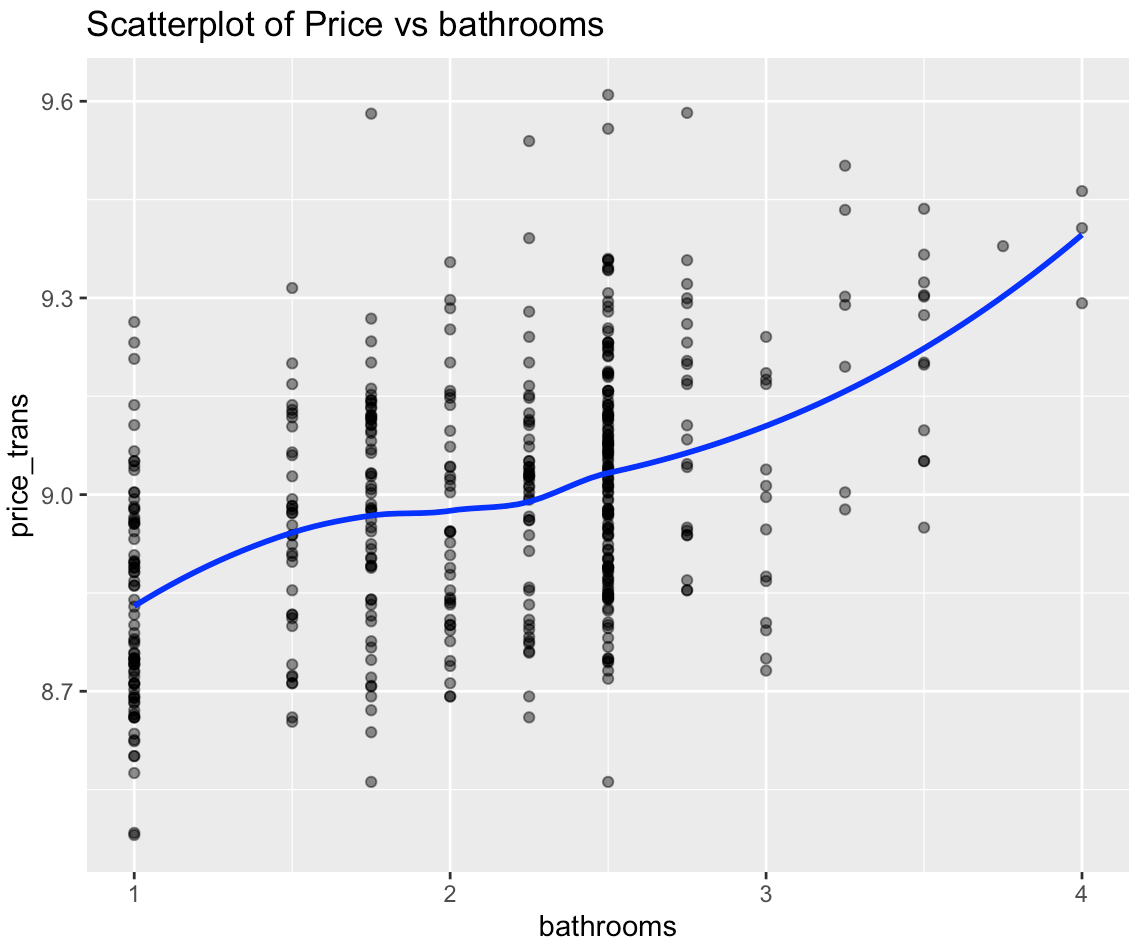
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : neighborhood radius 1
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : reciprocal condition number 0
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : There are other near singularities as well. 1
```




```
## `geom_smooth()` using formula = 'y ~ x'
```




```
## `geom_smooth()` using formula = 'y ~ x'
```


```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : at -0.015
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : radius 0.000225
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : all data on boundary of neighborhood. make span bigger
```

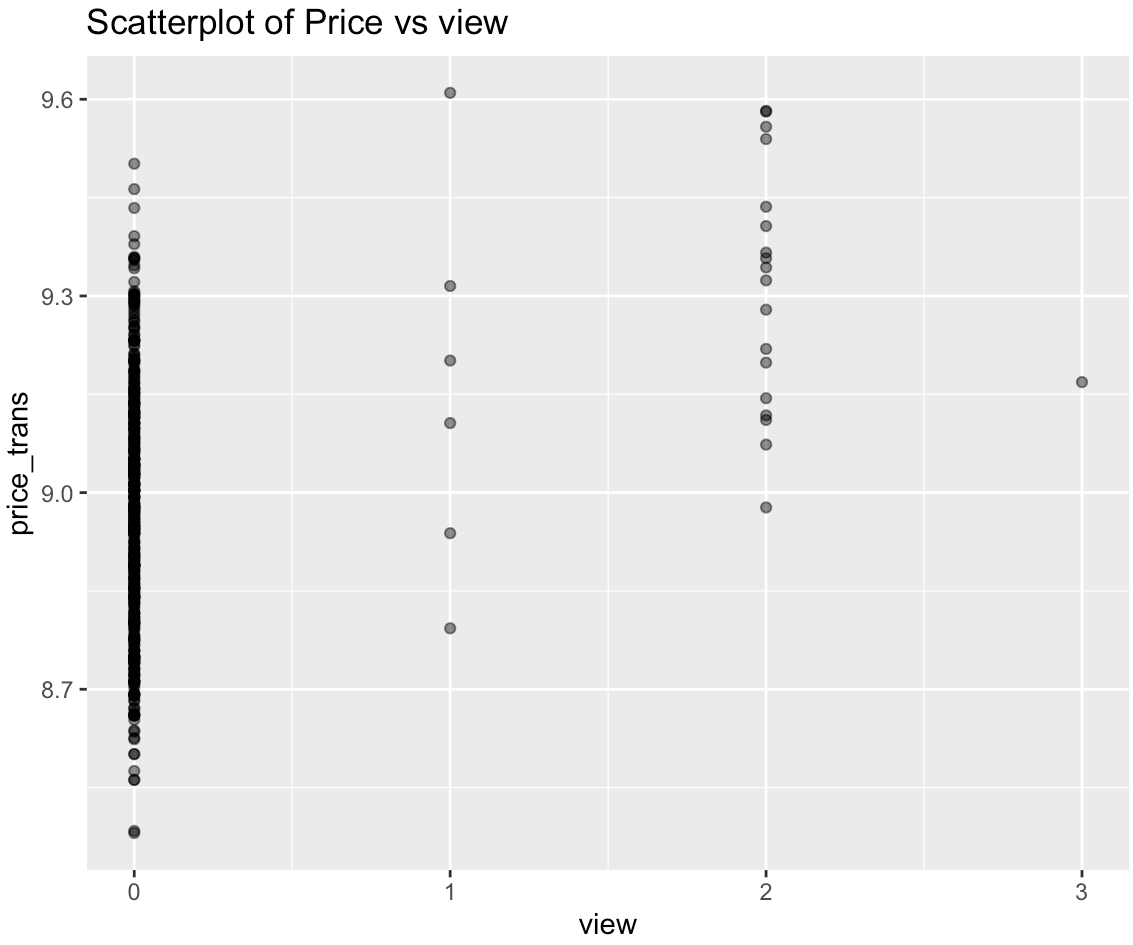
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : pseudoinverse used at -0.015
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : neighborhood radius 0.015
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : reciprocal condition number 1
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : zero-width neighborhood. make span bigger
```

```
## Warning: Failed to fit group -1.
## Caused by error in `predLoess()`:
## ! NA/NaN/Inf in foreign function call (arg 5)
```



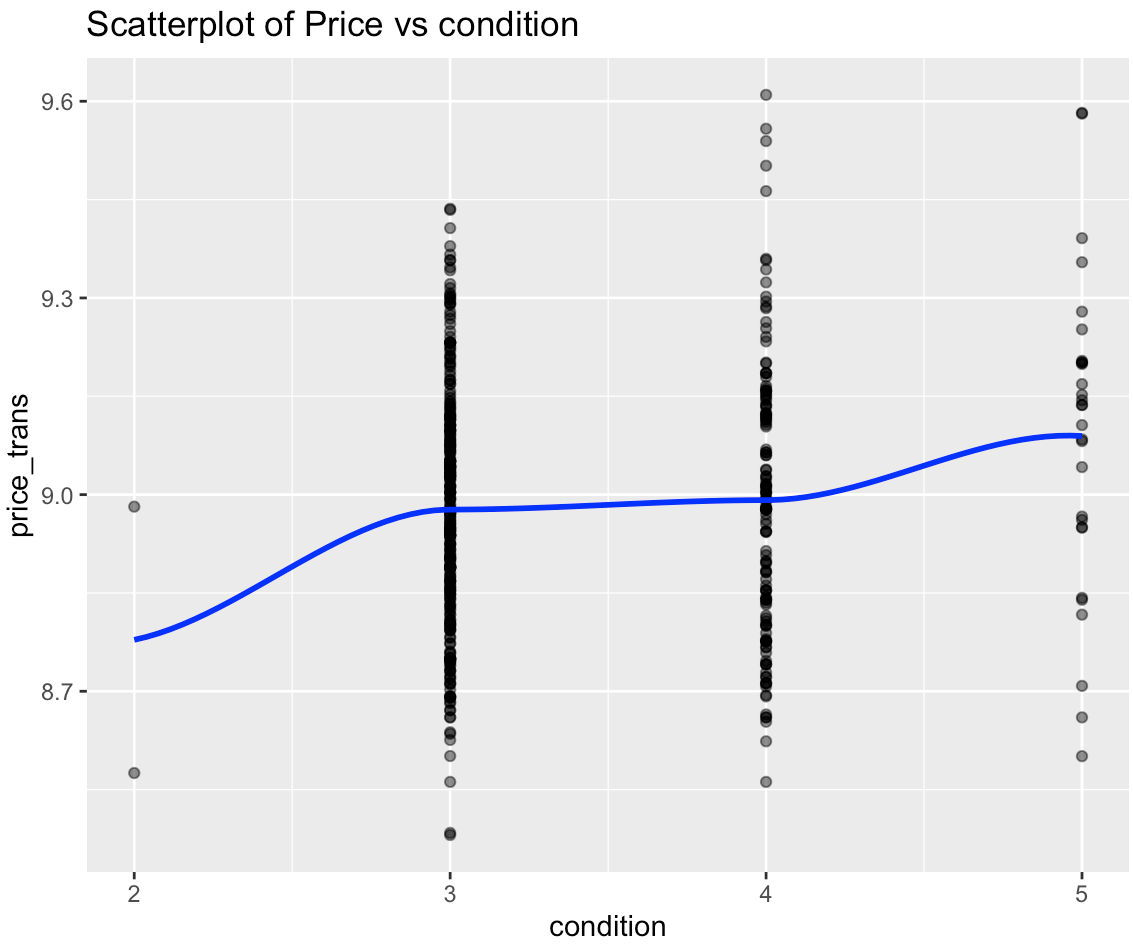

```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : pseudoinverse used at 1.985
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : neighborhood radius 2.015
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : reciprocal condition number 7.6803e-16
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : There are other near singularities as well. 4.0602
```



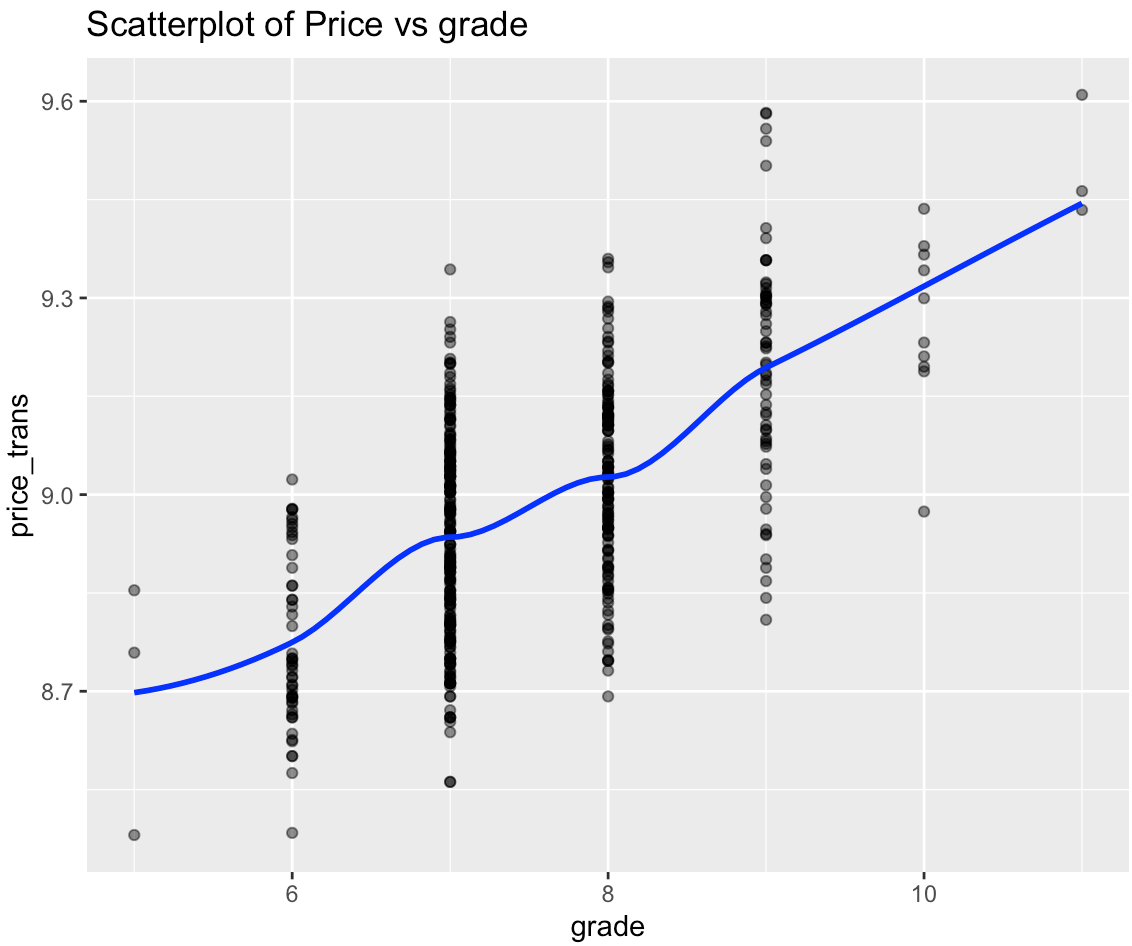

```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : pseudoinverse used at 7
```

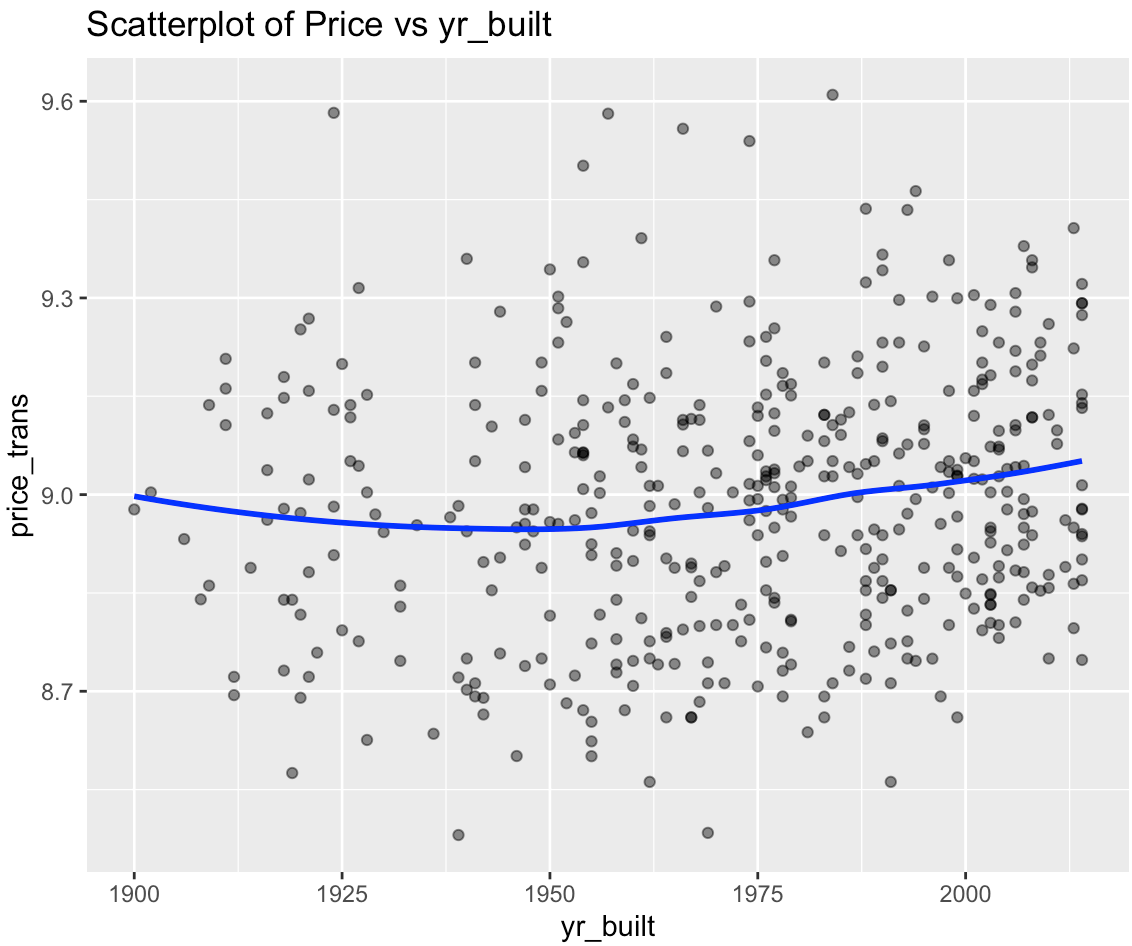
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : neighborhood radius 1
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : reciprocal condition number 0
```

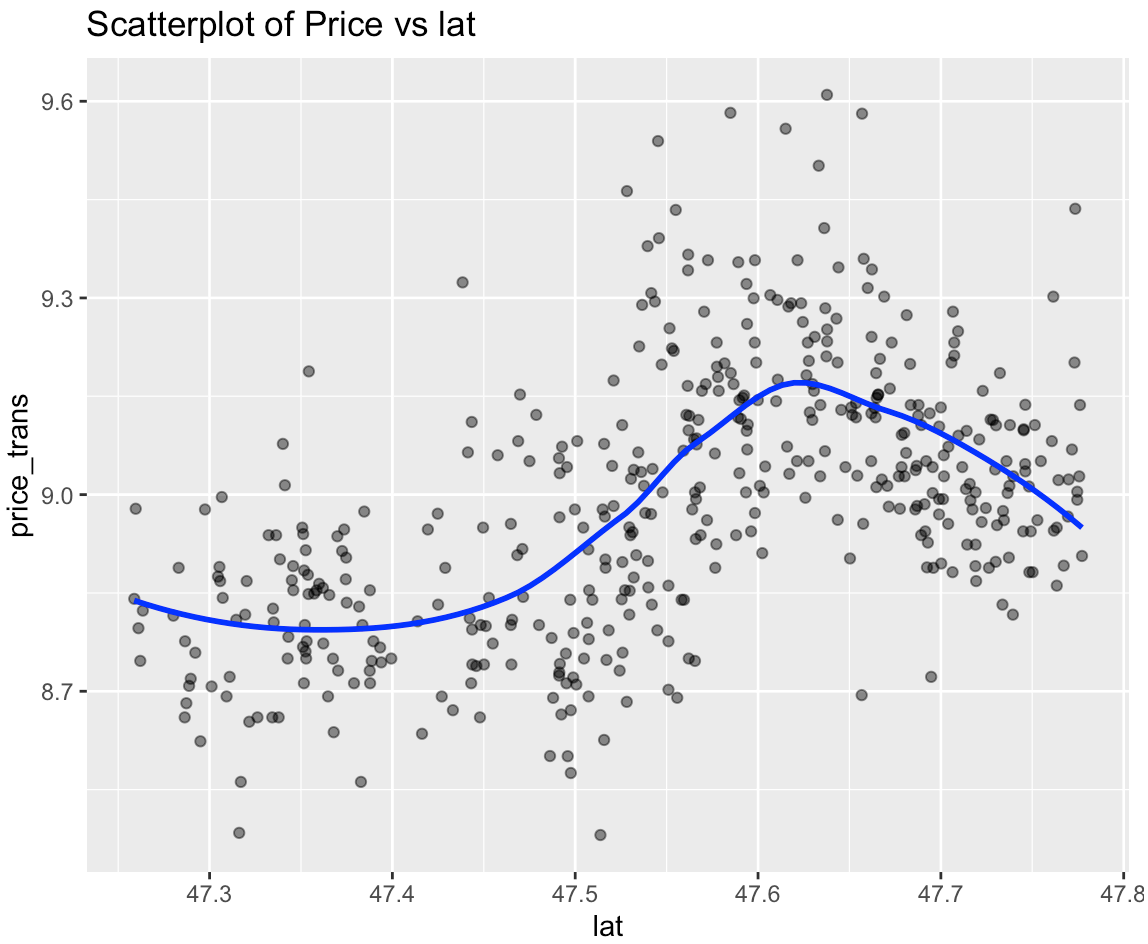
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : There are other near singularities as well. 1
```



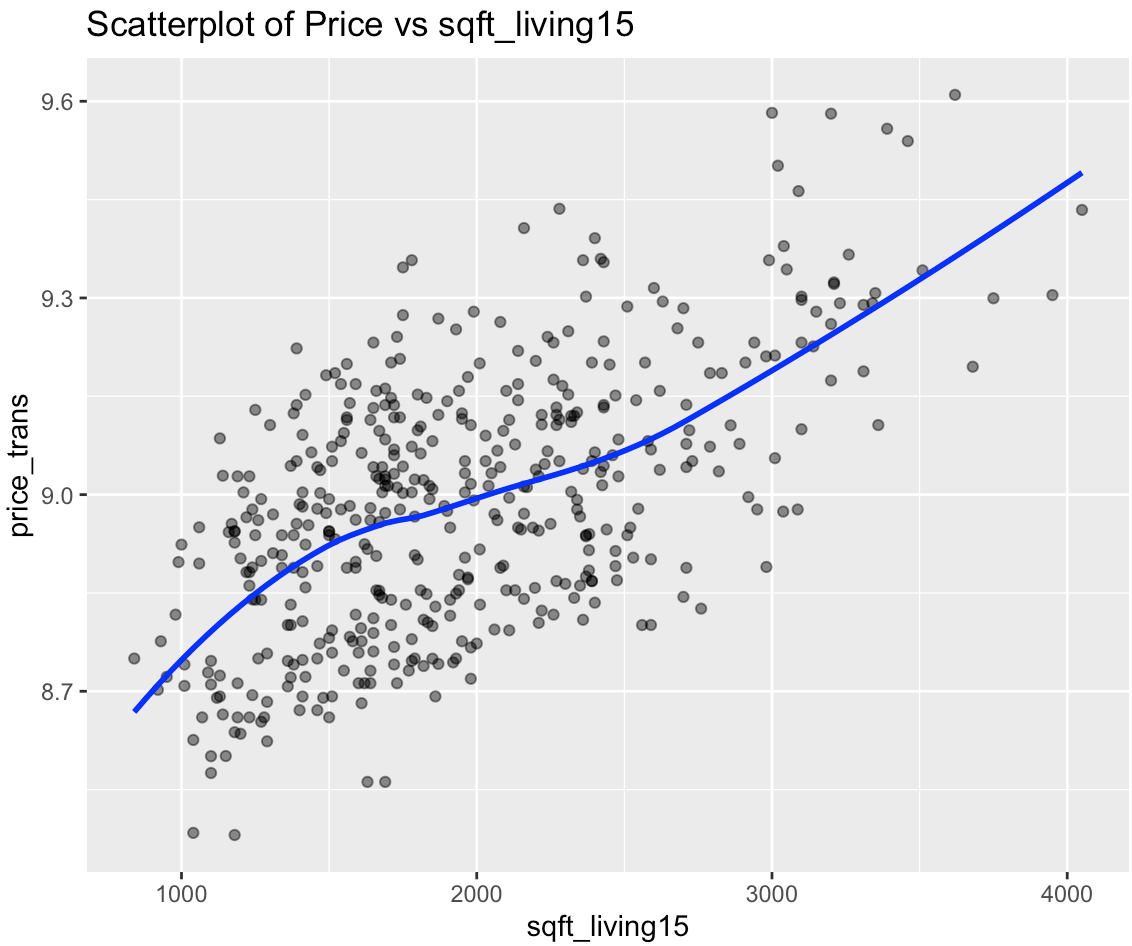

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
## `geom_smooth()` using formula = 'y ~ x'
```




```
## `geom_smooth()` using formula = 'y ~ x'
```




```
# Fit a model with polynomial terms for selected predictors
model_poly <- lm(price_trans ~ poly(bedrooms, 2) + poly(bathrooms, 2) + poly(sqft_lot,
3) +
                   factor(view) + condition + poly(grade, 2) + poly(yr_built, 2) +
                   lat + poly(sqft_living15, 2), data = data)

summary(model_poly)
```

```
##
## Call:
## lm(formula = price_trans ~ poly(bedrooms, 2) + poly(bathrooms,
##     2) + poly(sqft_lot, 3) + factor(view) + condition + poly(grade,
##     2) + poly(yr_built, 2) + lat + poly(sqft_living15, 2), data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.22352 -0.06647 -0.00205  0.06122  0.29548
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)             -22.267434   1.679562 -13.258  < 2e-16 ***
## poly(bedrooms, 2)1        0.273710   0.120856   2.265   0.0240 *
## poly(bedrooms, 2)2       -0.226170   0.104771  -2.159   0.0314 *
## poly(bathrooms, 2)1       0.329554   0.167605   1.966   0.0499 *
## poly(bathrooms, 2)2       0.034508   0.109476   0.315   0.7528
## poly(sqft_lot, 3)1       -0.127889   0.108760  -1.176   0.2403
## poly(sqft_lot, 3)2        0.263413   0.109501   2.406   0.0166 *
## poly(sqft_lot, 3)3       -0.493792   0.107594  -4.589 5.85e-06 ***
## factor(view)1             0.027630   0.040454   0.683   0.4950
## factor(view)2             0.136282   0.024316   5.605 3.74e-08 ***
## factor(view)3             0.002962   0.096507   0.031   0.9755
## condition                 0.046756   0.008408   5.561 4.74e-08 ***
## poly(grade, 2)1           1.801164   0.155475  11.585  < 2e-16 ***
## poly(grade, 2)2           0.061209   0.124839   0.490   0.6242
## poly(yr_built, 2)1       -0.730502   0.155552  -4.696 3.58e-06 ***
## poly(yr_built, 2)2        0.036313   0.122274   0.297   0.7666
## lat                       0.653752   0.035276  18.533  < 2e-16 ***
## poly(sqft_living15, 2)1   1.296791   0.158305   8.192 3.00e-15 ***
## poly(sqft_living15, 2)2  -0.083802   0.116372  -0.720   0.4718
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09496 on 428 degrees of freedom
## Multiple R-squared:  0.783,  Adjusted R-squared:  0.7739
## F-statistic: 85.81 on 18 and 428 DF,  p-value: < 2.2e-16
```

```
residuals <- residuals(model_poly)
shapiro.test(residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals
## W = 0.99482, p-value = 0.1387
```

```
bptest(model_poly)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  model_poly
## BP = 24.025, df = 18, p-value = 0.1542
```

```
# Log Transformation
model_log <- model_poly <- lm(log(price_trans) ~ poly(bedrooms, 2) + poly(bathrooms, 2)
+ poly(sqft_lot, 3) + factor(view) + condition + poly(grade, 2) + poly(yr_built, 2) + la
t + poly(sqft_living15, 2), data = data)

summary(model_log)
```

```
##
## Call:
## lm(formula = log(price_trans) ~ poly(bedrooms, 2) + poly(bathrooms,
##     2) + poly(sqft_lot, 3) + factor(view) + condition + poly(grade,
##     2) + poly(yr_built, 2) + lat + poly(sqft_living15, 2), data = data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.025239 -0.007351 -0.000160  0.006840  0.032710
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)              -1.3038634  0.1864050  -6.995 1.03e-11 ***
## poly(bedrooms, 2)1        0.0309897  0.0134131   2.310   0.0213 *
## poly(bedrooms, 2)2       -0.0254082  0.0116279  -2.185   0.0294 *
## poly(bathrooms, 2)1       0.0366156  0.0186015   1.968   0.0497 *
## poly(bathrooms, 2)2       0.0033191  0.0121501   0.273   0.7849
## poly(sqft_lot, 3)1       -0.0139650  0.0120706  -1.157   0.2479
## poly(sqft_lot, 3)2        0.0290300  0.0121529   2.389   0.0173 *
## poly(sqft_lot, 3)3       -0.0552081  0.0119413  -4.623 5.01e-06 ***
## factor(view)1             0.0028411  0.0044898   0.633   0.5272
## factor(view)2             0.0146913  0.0026986   5.444 8.80e-08 ***
## factor(view)3             0.0003956  0.0107108   0.037   0.9706
## condition                 0.0051456  0.0009332   5.514 6.08e-08 ***
## poly(grade, 2)1           0.1994279  0.0172553  11.558  < 2e-16 ***
## poly(grade, 2)2           0.0049636  0.0138552   0.358   0.7203
## poly(yr_built, 2)1       -0.0808564  0.0172638  -4.684 3.79e-06 ***
## poly(yr_built, 2)2        0.0050418  0.0135705   0.372   0.7104
## lat                       0.0732041  0.0039150  18.698  < 2e-16 ***
## poly(sqft_living15, 2)1   0.1436523  0.0175693   8.176 3.35e-15 ***
## poly(sqft_living15, 2)2 -0.0109513  0.0129155  -0.848   0.3970
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01054 on 428 degrees of freedom
## Multiple R-squared:  0.7831, Adjusted R-squared:  0.774
## F-statistic: 85.85 on 18 and 428 DF,  p-value: < 2.2e-16
```

```
residuals <- residuals(model_log)
shapiro.test(residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals
## W = 0.99518, p-value = 0.1806
```

```
bptest(model_log)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  model_log
## BP = 22.625, df = 18, p-value = 0.2054
```

```
model_log1 <- model_poly <- lm(scale(log(price_trans)) ~ poly(bedrooms, 2) + poly(bathro
oms, 2) + poly(sqft_lot, 3) + factor(view) + condition + poly(grade, 2) + poly(yr_built,
2) + lat + poly(sqft_living15, 2), data = data)

predictions_model_log <- predict(model_log1, newdata = data)

# Calculate MSE, RMSE and R-squared
Y_log <- scale(data$price)
mse_log <- mean((Y_log - predictions_model_log)^2)
rmse_log <-sqrt(mse_log)
total_ss_log <- sum((Y_log - mean(Y_log))^2)
residual_ss_log <- sum((Y_log - predictions_model_log)^2)
r_squared_log <- 1 - (residual_ss_log / total_ss_log)

print(paste("MSE for Transformed Model: ", mse_log))
```

```
## [1] "MSE for Transformed Model:  0.308952303122669"
```

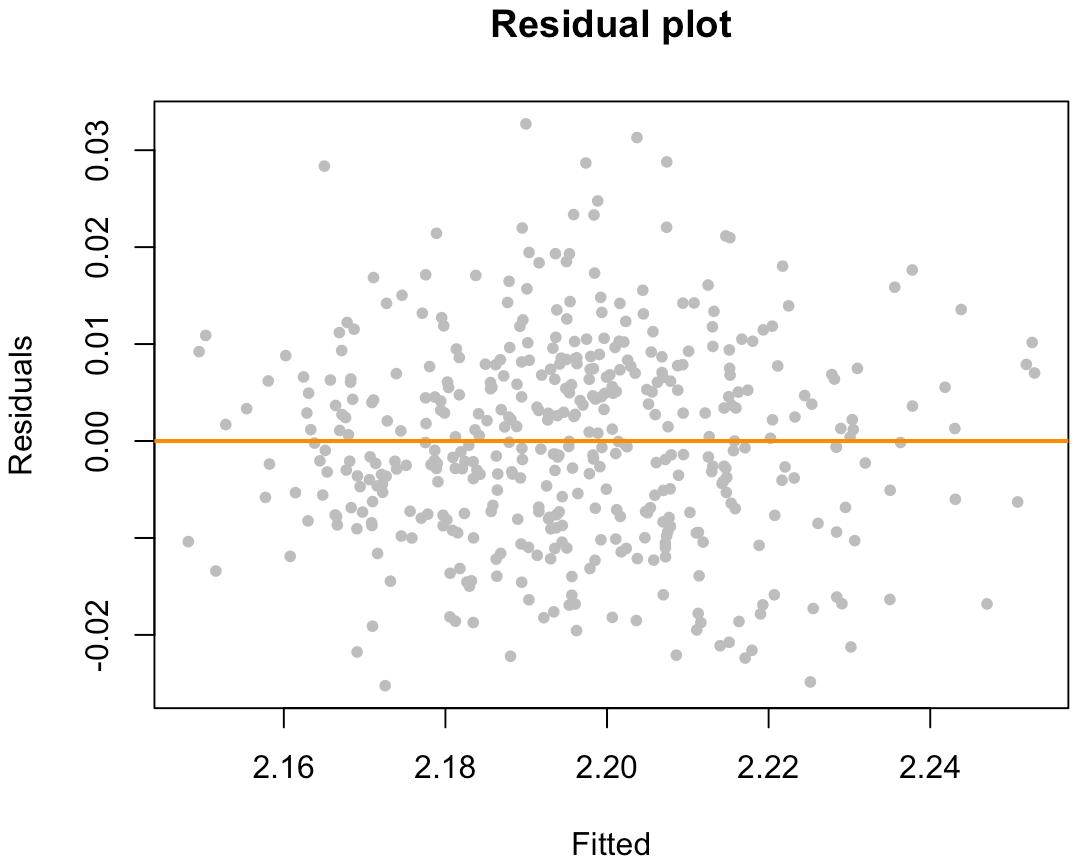
```
print(paste("RMSE for Transformed Model: ", rmse_log))
```

```
## [1] "RMSE for Transformed Model:  0.555834780418308"
```

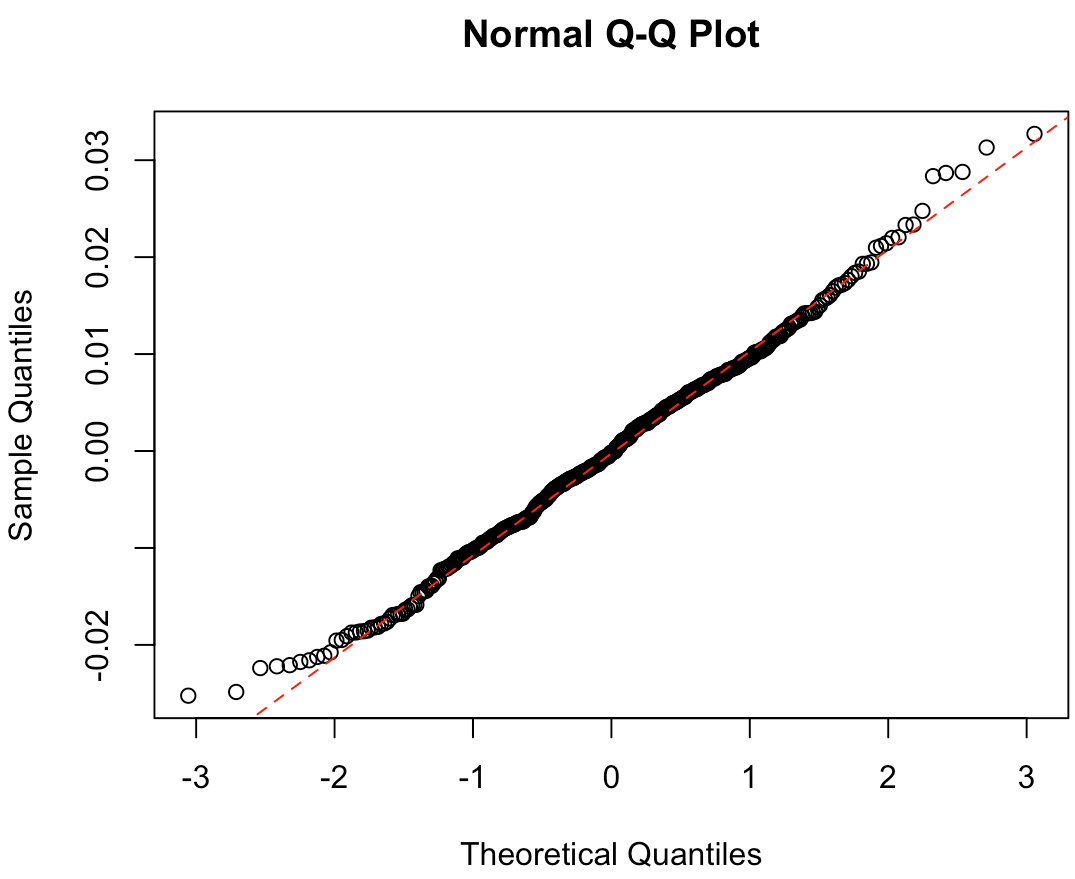
```
print(paste("R-squared for Transformed Model: ", r_squared_log))
```

```
## [1] "R-squared for Transformed Model:  0.690354978708895"
```

```
#Check model assumptions
plot(fitted(model_log), resid(model_log), col = "grey", pch = 20,
     xlab = "Fitted", ylab = "Residuals", main = "Residual plot")
abline(h = 0, col = "darkorange", lwd = 2)
```




```
qqnorm(resid(model_log),
       main = "Normal Q-Q Plot")
qqline(resid(model_log), col = "red", lty = 2)
```




```
Y <- log(data$price_trans)

# Create a matrix of predictors
X <- model.matrix(Y ~ poly(bedrooms, 2) + poly(bathrooms, 2) + poly(sqft_lot, 3) + facto
r(view) + condition + poly(grade, 2) + poly(yr_built, 2) + lat + poly(sqft_living15, 2),
data = data)[,-1]
```

```
# Set up cross-validation for Lasso and Ridge
set.seed(123)
cv_lasso <- cv.glmnet(X, Y, alpha = 1, nfolds = 10)  # Lasso
cv_ridge <- cv.glmnet(X, Y, alpha = 0, nfolds = 10)  # Ridge

# Extract the best lambda values
lambda_best_lasso <- cv_lasso$lambda.min
lambda_best_ridge <- cv_ridge$lambda.min
```

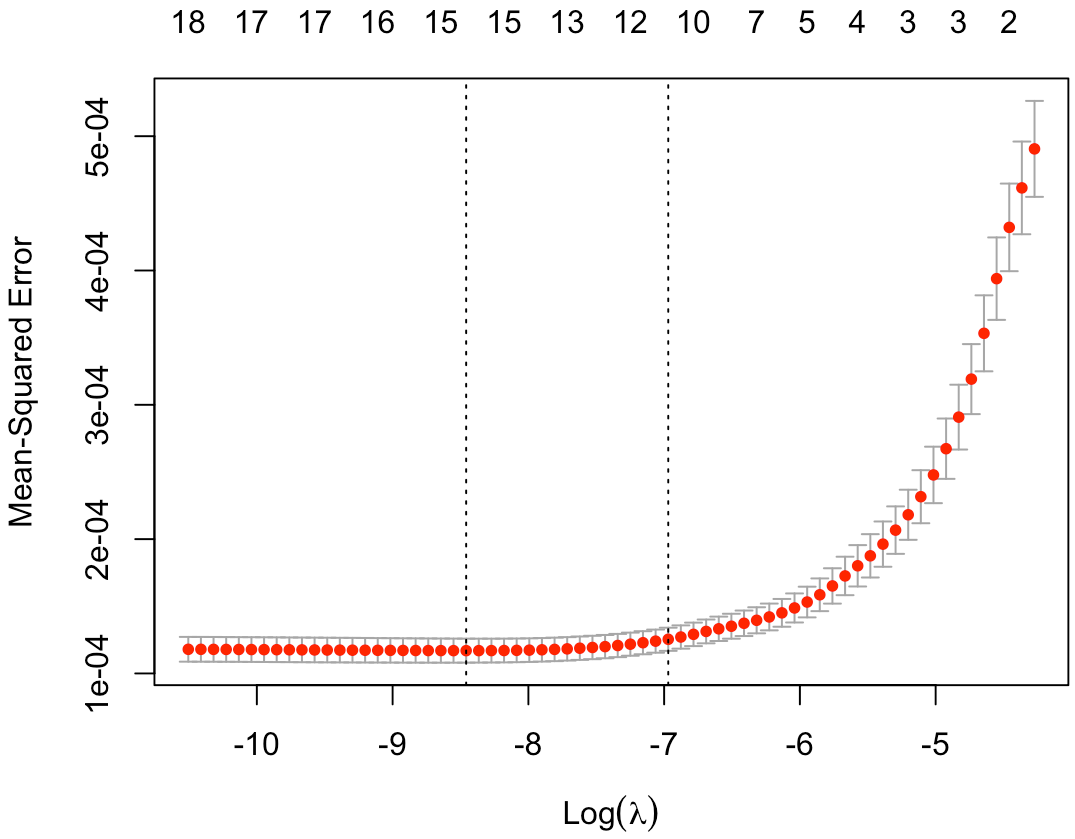
```
# Coefficients at the best lambda for Lasso
coefficients_lasso <- coef(cv_lasso, s = "lambda.min")
print(coefficients_lasso)
```

```
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                                  s1
## (Intercept)             -1.282850495
## poly(bedrooms, 2)1       0.026753718
## poly(bedrooms, 2)2      -0.017784942
## poly(bathrooms, 2)1      0.034227517
## poly(bathrooms, 2)2      .
## poly(sqft_lot, 3)1      -0.005767811
## poly(sqft_lot, 3)2       0.020523079
## poly(sqft_lot, 3)3      -0.046419052
## factor(view)1            0.001963401
## factor(view)2            0.014390697
## factor(view)3            .
## condition                0.004895916
## poly(grade, 2)1          0.193765041
## poly(grade, 2)2          .
## poly(yr_built, 2)1      -0.067234876
## poly(yr_built, 2)2       0.006882148
## lat                      0.072780761
## poly(sqft_living15, 2)1  0.137697663
## poly(sqft_living15, 2)2 -0.003244168
```

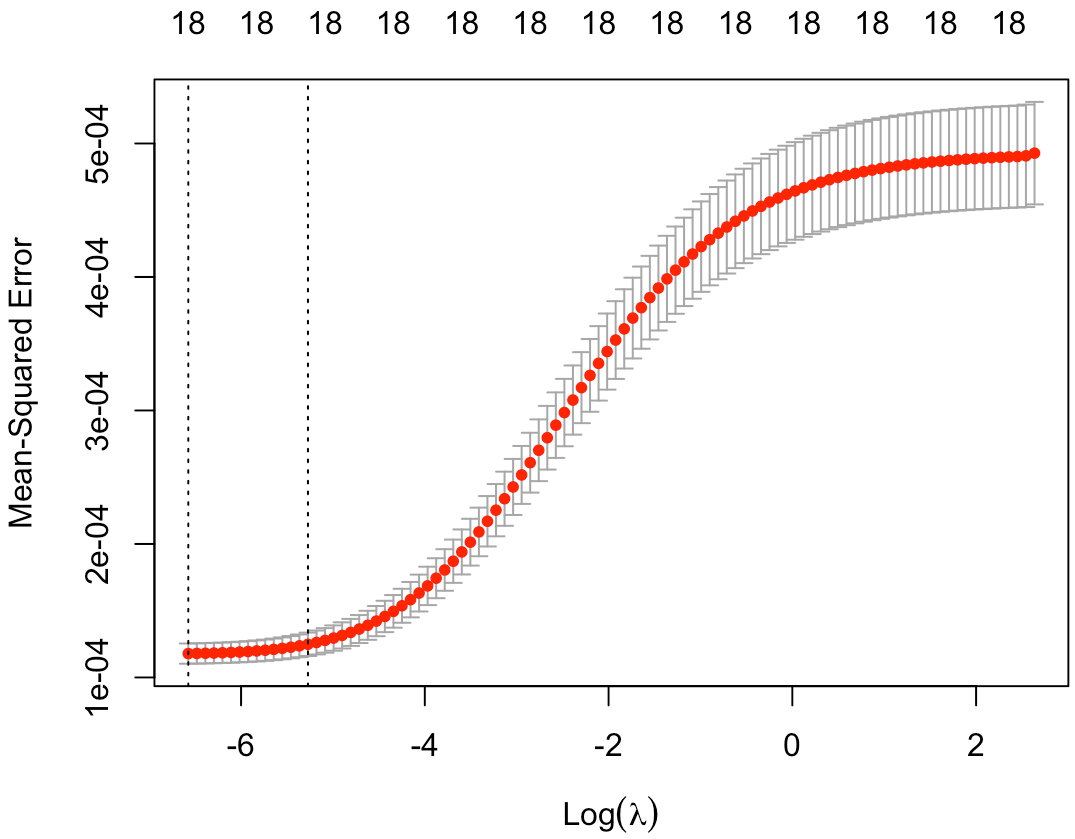
```
# Coefficients at the best lambda for Ridge
coefficients_ridge <- coef(cv_ridge, s = "lambda.min")
print(coefficients_ridge)
```

```
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                                  s1
## (Intercept)             -1.158907002
## poly(bedrooms, 2)1       0.030670862
## poly(bedrooms, 2)2      -0.023270483
## poly(bathrooms, 2)1      0.041193389
## poly(bathrooms, 2)2      0.003617920
## poly(sqft_lot, 3)1      -0.008606424
## poly(sqft_lot, 3)2       0.023879439
## poly(sqft_lot, 3)3      -0.046935338
## factor(view)1            0.004113282
## factor(view)2            0.014845952
## factor(view)3            0.002014084
## condition                0.004996282
## poly(grade, 2)1          0.181123429
## poly(grade, 2)2          0.003099383
## poly(yr_built, 2)1      -0.066477272
## poly(yr_built, 2)2       0.010141443
## lat                      0.070166161
## poly(sqft_living15, 2)1  0.135583265
## poly(sqft_living15, 2)2 -0.006999234
```

```
# Plot the CV results for Lasso
plot(cv_lasso)
```




```
# Plot the CV results for Ridge
plot(cv_ridge)
```




```
# Predictions using the best lambda
predictions_lasso <- predict(cv_lasso, newx = X, s = "lambda.min")
predictions_ridge <- predict(cv_ridge, newx = X, s = "lambda.min")

# Calculate MSE and R-squared
mse_lasso <- mean((Y - predictions_lasso)^2)
mse_ridge <- mean((Y - predictions_ridge)^2)

print(paste("MSE for Lasso: ", mse_lasso))
```

```
## [1] "MSE for Lasso:  0.000107171130672979"
```

```
print(paste("MSE for Ridge: ", mse_ridge))
```

```
## [1] "MSE for Ridge:  0.000107427842958468"
```

```
ss_res_lasso <- sum((Y - predictions_lasso)^2)
ss_tot_lasso <- sum((Y - mean(Y))^2)
r_squared_lasso <- 1 - ss_res_lasso / ss_tot_lasso
cat("R-squared for Lasso:", r_squared_lasso, "\n")
```

```
## R-squared for Lasso: 0.781433
```

```
ss_res_ridge <- sum((Y - predictions_ridge)^2)
ss_tot_ridge <- sum((Y - mean(Y))^2)
r_squared_ridge <- 1 - ss_res_ridge / ss_tot_ridge
cat("R-squared for Ridge:", r_squared_ridge, "\n")
```

```
## R-squared for Ridge: 0.7809094
```

```
# Prepare the matrix of predictors for glmnet, excluding non-significant predictors iden
tified from Lasso
X_filtered <- model.matrix(~ poly(bedrooms, 2) + poly(bathrooms, 2) + poly(sqft_lot, 3)
+factor(view) + condition + poly(grade, 2) + poly(yr_built, 2) +lat + poly(sqft_living1
5, 2), data = data)[,-1]

# Exclude columns identified as non-significant if needed
X_final <- X_filtered[, !colnames(X_filtered) %in% c("poly(bathrooms, 2)2", "factor(vie
w)3")]

# Response variable already log-transformed
Y_final <- log(data$price_trans)
```

```
# Refitting Lasso model with final predictor set
cv_lasso_final <- cv.glmnet(X_final, Y_final, alpha = 1, nfolds = 10)
best_lambda_final <- cv_lasso_final$lambda.min
```

```
# Extracting coefficients at the best lambda
coefficients_final <- coef(cv_lasso_final, s = "lambda.min")
print(coefficients_final)
```

```
## 17 x 1 sparse Matrix of class "dgCMatrix"
##                                  s1
## (Intercept)             -1.282850495
## poly(bedrooms, 2)1       0.026753718
## poly(bedrooms, 2)2      -0.017784942
## poly(bathrooms, 2)1      0.034227517
## poly(sqft_lot, 3)1      -0.005767811
## poly(sqft_lot, 3)2       0.020523079
## poly(sqft_lot, 3)3      -0.046419052
## factor(view)1            0.001963401
## factor(view)2            0.014390697
## condition                0.004895916
## poly(grade, 2)1          0.193765041
## poly(grade, 2)2          .
## poly(yr_built, 2)1      -0.067234876
## poly(yr_built, 2)2       0.006882148
## lat                      0.072780761
## poly(sqft_living15, 2)1  0.137697663
## poly(sqft_living15, 2)2 -0.003244168
```

```
# Predictions using the best lambda
predictions_final <- predict(cv_lasso_final, newx = X_final, s = "lambda.min")

# Calculate new MSE, RMSE and R-squared
mse_final <- mean((Y_final - predictions_final)^2)
rmse_final <-sqrt(mse_final)
total_ss_final <- sum((Y_final - mean(Y_final))^2)
residual_ss_final <- sum((Y_final - predictions_final)^2)
r_squared_final <- 1 - (residual_ss_final / total_ss_final)

print(paste("New MSE for Final Lasso Model: ", mse_final))
```

```
## [1] "New MSE for Final Lasso Model:  0.000107171130672979"
```

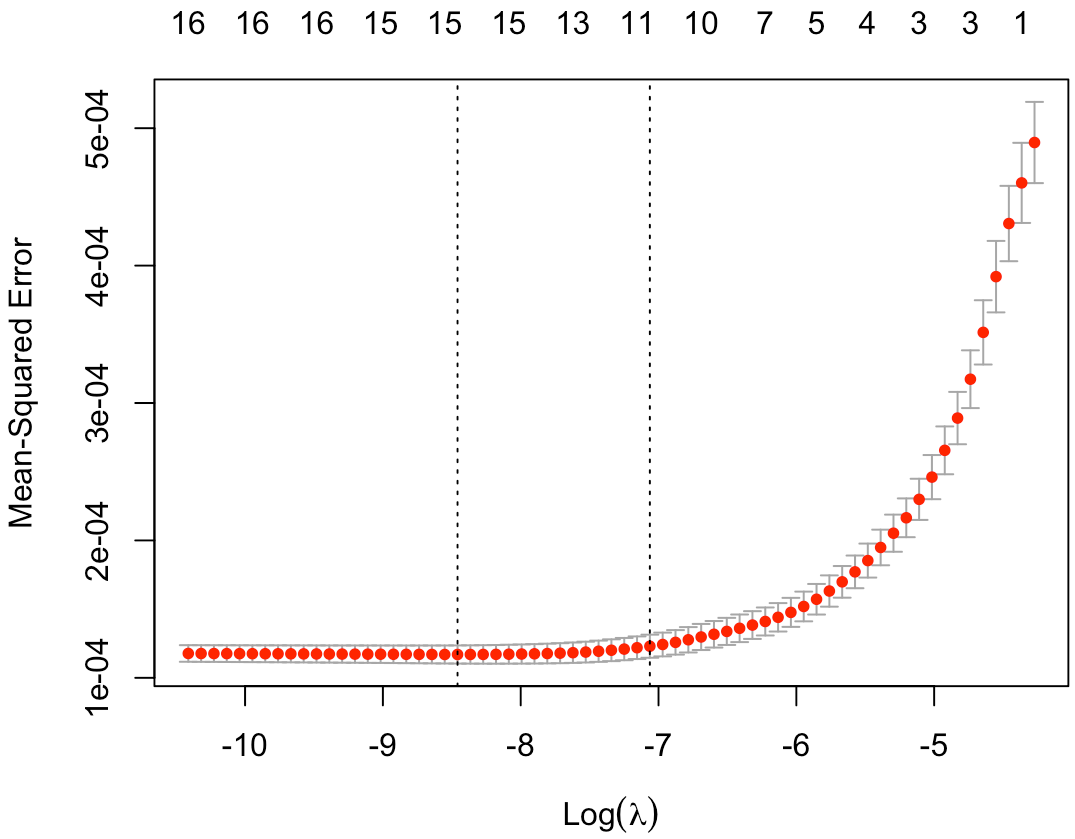
```
print(paste("New RMSE for Final Lasso Model: ", rmse_final))
```

```
## [1] "New RMSE for Final Lasso Model:  0.0103523490413036"
```

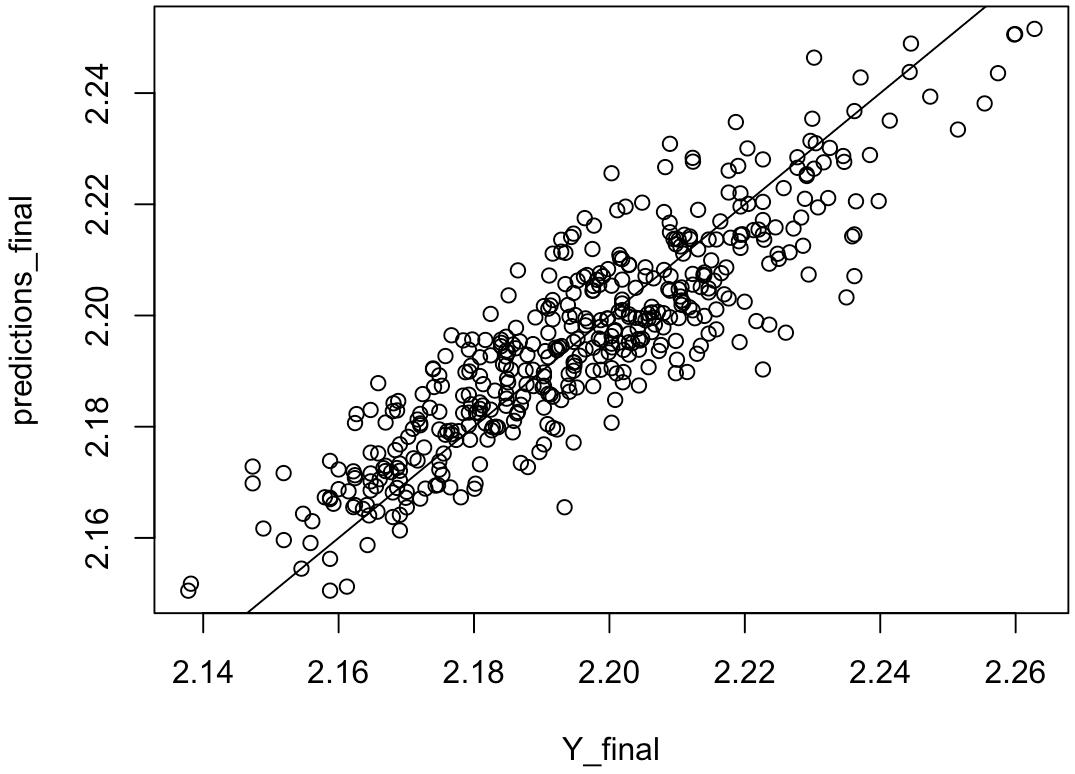
```
print(paste("New R-squared for Final Lasso Model: ", r_squared_final))
```

```
## [1] "New R-squared for Final Lasso Model:  0.781432992919511"
```

```
# Plotting MSE path for the final Lasso model
plot(cv_lasso_final)
```




```
# Optionally, plot diagnostics for residuals
plot(Y_final, predictions_final)
abline(0, 1)
```




```
# Model Diagnosis for final model
final_model_formula <- log(data$price_trans) ~ poly(bedrooms, 2) + poly(bathrooms, 2) +
poly(sqft_lot, 3) + factor(view) + condition +
poly(grade, 2) + poly(yr_built, 2) + lat + poly(sqft_living15, 2)


Y_final <- log(data$price_trans)
X_final <- model.matrix(final_model_formula, data = data)[, -1]
residuals_final <- Y_final - predictions_final

model_final <- lm(Y_final ~ X_final)

bp_results <- bptest(model_final)
shapiro_results <- shapiro.test(residuals_final)

print(shapiro_results)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals_final
## W = 0.9951, p-value = 0.1696
```

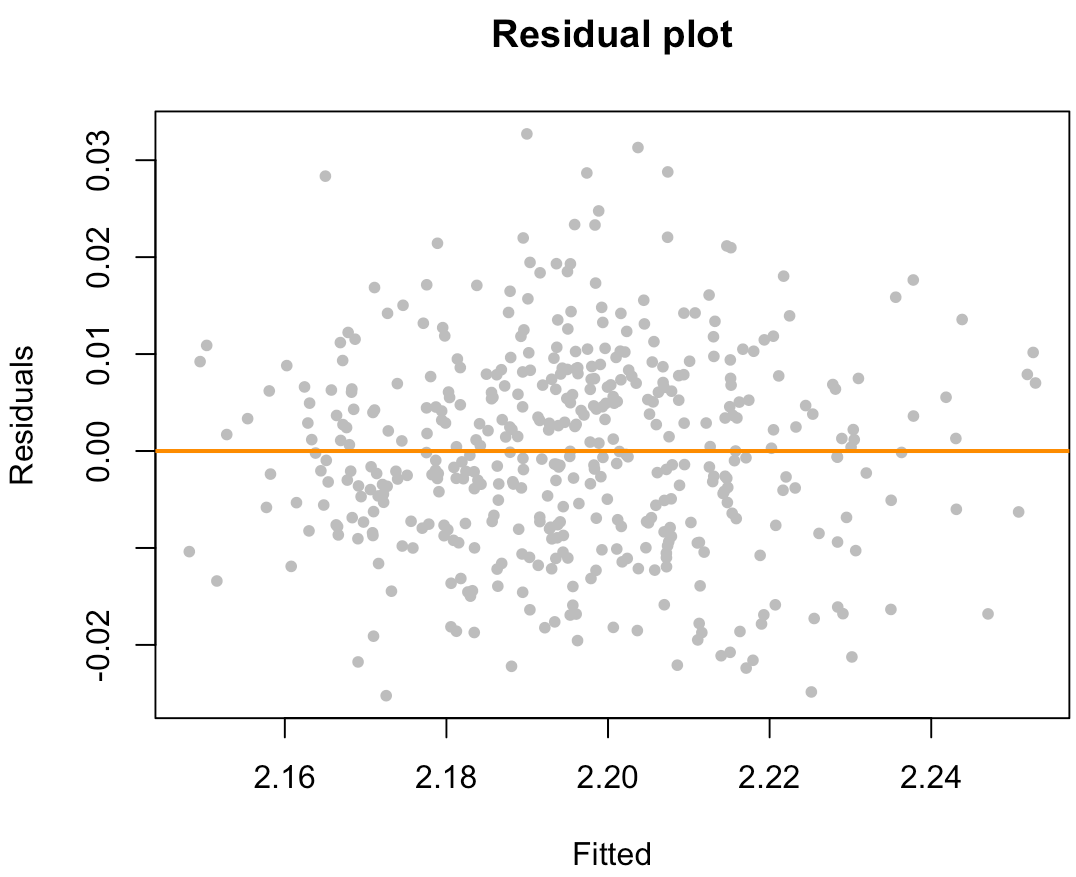
```
print(bp_results)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  model_final
## BP = 22.625, df = 18, p-value = 0.2054
```

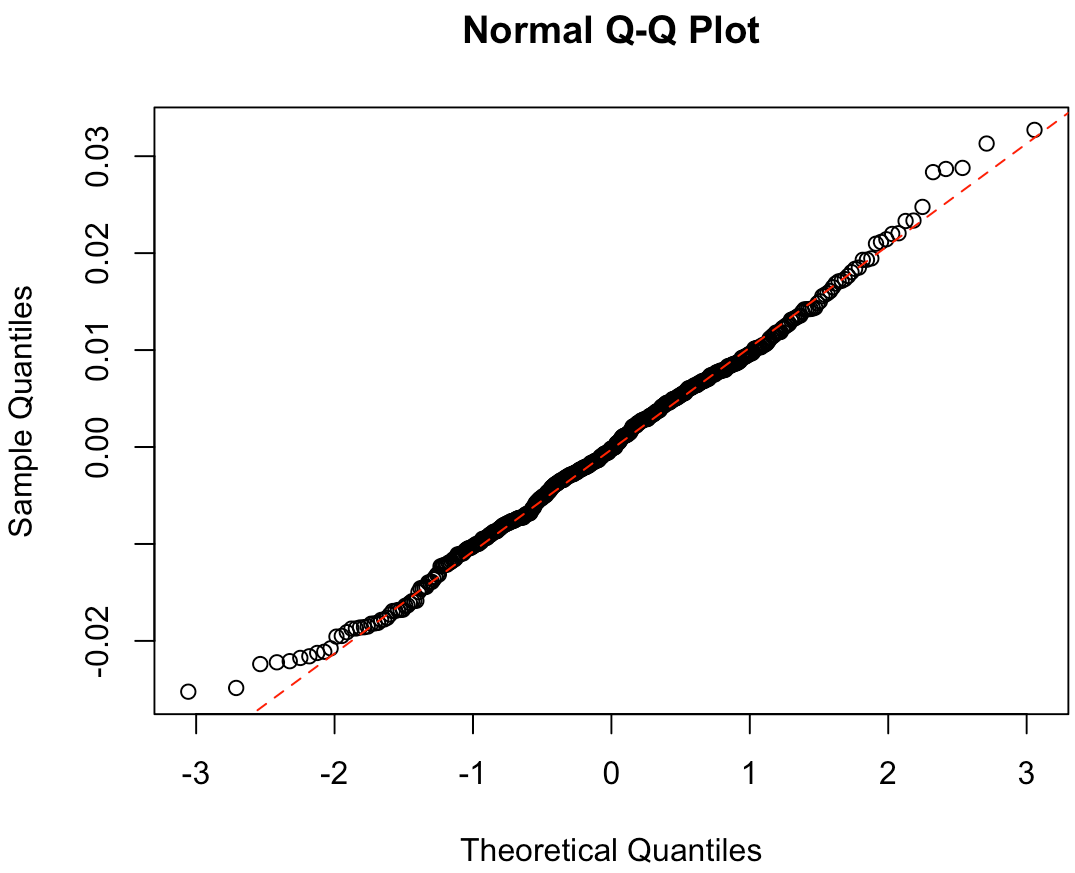
```
summary(model_final)
```

```
##
## Call:
## lm(formula = Y_final ~ X_final)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.025239 -0.007351 -0.000160  0.006840  0.032710
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   -1.3038634  0.1864050  -6.995 1.03e-11 ***
## X_finalpoly(bedrooms, 2)1      0.0309897  0.0134131   2.310   0.0213 *
## X_finalpoly(bedrooms, 2)2     -0.0254082  0.0116279  -2.185   0.0294 *
## X_finalpoly(bathrooms, 2)1     0.0366156  0.0186015   1.968   0.0497 *
## X_finalpoly(bathrooms, 2)2     0.0033191  0.0121501   0.273   0.7849
## X_finalpoly(sqft_lot, 3)1     -0.0139650  0.0120706  -1.157   0.2479
## X_finalpoly(sqft_lot, 3)2      0.0290300  0.0121529   2.389   0.0173 *
## X_finalpoly(sqft_lot, 3)3     -0.0552081  0.0119413  -4.623 5.01e-06 ***
## X_finalfactor(view)1           0.0028411  0.0044898   0.633   0.5272
## X_finalfactor(view)2           0.0146913  0.0026986   5.444 8.80e-08 ***
## X_finalfactor(view)3           0.0003956  0.0107108   0.037   0.9706
## X_finalcondition               0.0051456  0.0009332   5.514 6.08e-08 ***
## X_finalpoly(grade, 2)1         0.1994279  0.0172553  11.558  < 2e-16 ***
## X_finalpoly(grade, 2)2         0.0049636  0.0138552   0.358   0.7203
## X_finalpoly(yr_built, 2)1     -0.0808564  0.0172638  -4.684 3.79e-06 ***
## X_finalpoly(yr_built, 2)2      0.0050418  0.0135705   0.372   0.7104
## X_finallat                     0.0732041  0.0039150  18.698  < 2e-16 ***
## X_finalpoly(sqft_living15, 2)1  0.1436523  0.0175693   8.176 3.35e-15 ***
## X_finalpoly(sqft_living15, 2)2 -0.0109513  0.0129155  -0.848   0.3970
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01054 on 428 degrees of freedom
## Multiple R-squared:  0.7831, Adjusted R-squared:  0.774
## F-statistic: 85.85 on 18 and 428 DF,  p-value: < 2.2e-16
```

```
#Check model assumptions
plot(fitted(model_final), resid(model_final), col = "grey", pch = 20,
     xlab = "Fitted", ylab = "Residuals", main = "Residual plot")
abline(h = 0, col = "darkorange", lwd = 2)
```




```
qqnorm(resid(model_final),
       main = "Normal Q-Q Plot")
qqline(resid(model_final), col = "red", lty = 2)
```



Model Evaluation Comparison

```
# Combine the metrics into a data frame for comparison
comparison_table <- data.frame(
  Model = c("Model 0", "Model Log", "Model Final"),
  MSE = c(mse_0, mse_log, mse_final),
  RMSE = c(rmse_0, rmse_log, rmse_final),
  R_squared = c(r_squared_0, r_squared_log, r_squared_final)
)

# Print the comparison table
print(comparison_table)
```

```
##         Model          MSE       RMSE R_squared
## 1     Model 0 0.2588248968 0.50874836 0.7406564
## 2   Model Log 0.3089523031 0.55583478 0.6903550
## 3 Model Final 0.0001071711 0.01035235 0.7814330
```