assignment_08_03_PothineniKalyan

PothineniKalyan

2023-05-07

A. Clean up the Housing data

```
library(readxl)
```

```
## Warning: package 'readxl' was built under R version 4.2.3
```

```
library(writexl)
```

```
## Warning: package 'writexl' was built under R version 4.2.3
```

```
## Load the `data/Week-6-housing.xlsx` to
housing_df <- read_excel('data/week-6-housing.xlsx', sheet = 'Sheet2')
head(housing_df, n=5)
```

```
## # A tibble: 5 x 24
##   `Sale Date`         `Sale Price` sale_~1 sale_~2 sale_~3 sitet~4 addr_~5  zip5
##   <dttm>                     <dbl>   <dbl>   <dbl> <chr>   <chr>   <chr>   <dbl>
## 1 2006-01-03 00:00:00       698000       1       3 <NA>    R1      17021 ~ 98052
## 2 2006-01-03 00:00:00       649990       1       3 <NA>    R1      11927 ~ 98052
## 3 2006-01-03 00:00:00       572500       1       3 <NA>    R1      13315 ~ 98052
## 4 2006-01-03 00:00:00       420000       1       3 <NA>    R1      3303 1~ 98052
## 5 2006-01-03 00:00:00       369900       1       3 15      R1      16126 ~ 98052
## # ... with 16 more variables: ctyname <chr>, postalctyn <chr>, lon <dbl>,
## #   lat <dbl>, building_grade <dbl>, square_feet_total_living <dbl>,
## #   bedrooms <dbl>, bath_full_count <dbl>, bath_half_count <dbl>,
## #   bath_3qtr_count <dbl>, year_built <dbl>, year_renovated <dbl>,
## #   current_zoning <chr>, sq_ft_lot <dbl>, prop_type <chr>, present_use <dbl>,
## #   and abbreviated variable names 1: sale_reason, 2: sale_instrument,
## #   3: sale_warning, 4: sitetype, 5: addr_full
```

```
## Remove any columns that are not related to the analysis
housing_df <- housing_df[,c("Sale Date", "Sale Price", "building_grade",
                            "square_feet_total_living", "bedrooms",
                            "bath_full_count", "bath_half_count",
                            "bath_3qtr_count", "year_built",
                            "year_renovated", "sq_ft_lot")]
```

```
## Checking and removing any duplicate rows in the data set
housing_df <- housing_df[!duplicated(housing_df),]


## Checking and removing any missing values in the data set
housing_df <- na.omit(housing_df)

## To write the cleaned-up data to a new xlsx file, I am using the write.xlsx() function from the openx
## The cleaned-up data to a new file called "cleaned_housing.xlsx" in your working directory
clean_housting <- writexl::write_xlsx(housing_df, "data/week-8/cleaned_housing.xlsx", col_names = TRUE)

clean_housing_df <- read_excel('data/week-8/cleaned_housing.xlsx')
head(clean_housing_df, n=5)
```

```
## # A tibble: 5 x 11
##   `Sale Date`         Sale Pri~1 build~2 squar~3 bedro~4 bath_~5 bath_~6 bath_~7
##   <dttm>                   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 2006-01-03 00:00:00     698000       9    2810       4       2       1       0
## 2 2006-01-03 00:00:00     649990       9    2880       4       2       0       1
## 3 2006-01-03 00:00:00     572500       8    2770       4       1       1       1
## 4 2006-01-03 00:00:00     420000       8    1620       3       1       0       1
## 5 2006-01-03 00:00:00     369900       7    1440       3       1       0       1
## # ... with 3 more variables: year_built <dbl>, year_renovated <dbl>,
## #   sq_ft_lot <dbl>, and abbreviated variable names 1: `Sale Price`,
## #   2: building_grade, 3: square_feet_total_living, 4: bedrooms,
## #   5: bath_full_count, 6: bath_half_count, 7: bath_3qtr_count
```

```
## Use colnames() to change the column name
colnames(clean_housing_df)[colnames(clean_housing_df)=="Sale Price"] <- "Sale_Price"

colnames(clean_housing_df)[colnames(clean_housing_df)=="Sale Date"] <- "Sale_Date"
```

B

Explain any transformations or modifications you made to the dataset

> No transformations to the dataset. Some level of cleaning, cleared up some attributes which are not needed for analysis, removed duplicate rows, removed any missing values and rename couple of columns

Create two variables; one that will contain the variables Sale Price and Square Foot of Lot (same variables used from previous assignment on simple regression) and one that will contain Sale Price and several additional predictors of your choice. Explain the basis for your additional predictor selections

> To create the first variable, I will use the variables “Sale Price” and “sq_ft_lot” as they were used in the previous assignment on simple regression. The goal is to predict the sale price of a house based on the square footage of the lot it is built on

```
# Create variable with Sale Price and Square Foot of Lot
housing_VAR_1 <- clean_housing_df[,c("Sale_Price", "sq_ft_lot")]
```

To create the second variable, I will add several additional predictors that I think might be related to the sale price of a house. I will include "square_feet_total_living", "bedrooms", "bath_full_count" and "year_built". The basis for my predictor selections is that the size of a house, the number of bedrooms and bathrooms and the age of the house could all be factors that influence the sale price

```
# Create variable with Sale Price and additional predictors
housing_VAR_2 <- clean_housing_df[,c("Sale_Price", "square_feet_total_living", "bedrooms", "bath_full_c
```

Execute a summary() function on two variables defined in the previous step to compare the model results. What are the R2 and Adjusted R2 statistics? Explain what these results tell you about the overall model. Did the inclusion of the additional predictors help explain any large variations found in Sale Price?

To compare the two models, I will use the summary() function on each of them.

```
# Summary of first variable
summary(lm(housing_VAR_1$Sale_Price ~ housing_VAR_1$sq_ft_lot))
```

```
##
## Call:
## lm(formula = housing_VAR_1$Sale_Price ~ housing_VAR_1$sq_ft_lot)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2028934  -191936   -61445    93965  3738329
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)              6.386e+05  3.731e+03  171.13   <2e-16 ***
## housing_VAR_1$sq_ft_lot 8.609e-01  6.092e-02   14.13   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 393100 on 12788 degrees of freedom
## Multiple R-squared:  0.01538,    Adjusted R-squared:  0.0153
## F-statistic: 199.7 on 1 and 12788 DF,  p-value: < 2.2e-16
```

```
# Summary of second variable
summary(lm(housing_VAR_2$Sale_Price ~ housing_VAR_2$square_feet_total_living + housing_VAR_2$bedrooms +
```

```
##
## Call:
## lm(formula = housing_VAR_2$Sale_Price ~ housing_VAR_2$square_feet_total_living +
##     housing_VAR_2$bedrooms + housing_VAR_2$bath_full_count +
##     housing_VAR_2$year_built)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1726864  -116754   -39404    47783  3898903
##
## Coefficients:
##                                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)                         -3.940e+06  4.083e+05  -9.652  < 2e-16
## housing_VAR_2$square_feet_total_living  1.751e+02  4.307e+00  40.666  < 2e-16
## housing_VAR_2$bedrooms              -1.020e+04  4.397e+03  -2.320 0.020383
## housing_VAR_2$bath_full_count        2.278e+04  5.930e+03   3.841 0.000123
## housing_VAR_2$year_built             2.081e+03  2.061e+02  10.099  < 2e-16
##
## (Intercept)                         ***
## housing_VAR_2$square_feet_total_living ***
## housing_VAR_2$bedrooms              *
## housing_VAR_2$bath_full_count       ***
## housing_VAR_2$year_built            ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 346900 on 12785 degrees of freedom
## Multiple R-squared:  0.2336, Adjusted R-squared:  0.2333
## F-statistic: 974.2 on 4 and 12785 DF,  p-value: < 2.2e-16
```

For the first model with only one predictor (sq_ft_lot), the R-squared value is 0.01538 and the Adjusted R-squared value is 0.0153. This means that the predictor (sq_ft_lot) explains only 1.53% of the variance in the response variable (Sale_Price). In other words, the model does not fit the data very well and there are still large unexplained variations in the response variable.

For the second model with multiple predictors, the R-squared value is 0.2336 and the Adjusted R-squared value is 0.2333. This indicates that the predictors (square_feet_total_living, bedrooms, bath_full_count, and year_built) collectively explain about 23.33% of the variance in Sale_Price, which is a substantial improvement over the first model. Therefore, the inclusion of the additional predictors has helped explain a significant amount of the large variations found in Sale_Price.

Overall, the R-squared and Adjusted R-squared values give us an idea of how well the regression model fits the data. A higher R-squared value indicates a better fit, while a lower value indicates a poorer fit.

Considering the parameters of the multiple regression model you have created.What are the standardized betas for each parameter and what do the values indicate?

To obtain the standardized betas for each parameter in the multiple regression models, we need to divide the estimates by the standard deviation of each predictor variable. The standardized betas are measures of the effect size of each predictor variable in standard deviation units of the response variable. They indicate the change in the response variable (Sale Price in this case) in standard deviation units for a one standard deviation increase in each predictor variable, while holding all other predictor variables constant

First Model

```
library(QuantPsyc)
```

```
## Warning: package 'QuantPsyc' was built under R version 4.2.3
```

```
## Loading required package: boot
```

```
## Loading required package: dplyr
```

```
## Warning: package 'dplyr' was built under R version 4.2.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
## Loading required package: purrr
```

```
## Warning: package 'purrr' was built under R version 4.2.3
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

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

```
##
## Attaching package: 'QuantPsyc'
```

```
## The following object is masked from 'package:base':
##
##     norm
```

```
# Summary of first variable
first_model <- lm(Sale_Price ~ sq_ft_lot, data = clean_housing_df)


# Calculate the standardized betas
first_model_betas <- lm.beta(first_model)

# View the standardized betas
first_model_betas
```

```
## sq_ft_lot
## 0.1240017
```

the standardized beta for sq_ft_lot is 0.124, indicating that for a one standard deviation increase in sq_ft_lot, holding all other variables constant, we expect a 0.124 standard deviation increase in Sale_Price.However, for the first model, the standardized betas cannot be calculated as we don't have the standard deviation of the predictor variable

Second Model

```
library(QuantPsyc)
# Summary of first variable
second_model <- lm(Sale_Price ~ sq_ft_lot + square_feet_total_living + bedrooms + bath_full_count + yea


# Calculate the standardized betas
second_model_betas <- lm.beta(second_model)

# View the standardized betas
second_model_betas
```

```
##                sq_ft_lot square_feet_total_living                bedrooms
##               0.03849416               0.42080326              -0.01501387
##          bath_full_count               year_built
##               0.03711759               0.10079178
```

The standardized beta for sq_ft_lot is 0.03849416. This indicates that for a one standard deviation increase in sq_ft_lot, holding all other predictors constant, we expect an increase of 0.03849416 standard deviations in Sale Price. The standardized beta for square_feet_total_living is 0.42080326. This indicates that for a one standard deviation increase in square_feet_total_living, holding all other predictors constant, we expect an increase of 0.42080326 standard deviations in Sale Price. The standardized beta for bedrooms is -0.01501387. This indicates that for a one standard deviation increase in bedrooms, holding all other predictors constant, we expect a decrease of 0.01501387 standard deviations in Sale Price. The standardized beta for bath_full_count is 0.03711759. This indicates that for a one standard deviation increase in bath_full_count, holding all other predictors constant, we expect an increase of 0.03711759 standard deviations in Sale Price. The standardized beta for year_built is 0.10079178. This indicates that for a one standard deviation increase in year_built, holding all other predictors constant, we expect an increase of 0.10079178 standard deviations in Sale Price.

Calculate the confidence intervals for the parameters in your model and explain what the results indicate

To calculate the confidence intervals for the parameters in the first model, we can use the confint() function in R, which will return the confidence intervals for each parameter at the default 95% level

```
confint(first_model)
```

```
##                     2.5 %       97.5 %
## (Intercept) 6.312458e+05 6.458740e+05
## sq_ft_lot   7.414716e-01 9.802904e-01
```

We can be 95% confident that the true population intercept falls between 6.312458e+05 and 6.458740e+05

To calculate the confidence intervals for the parameters in the second model, we can use the confint() function in R, which will return the confidence intervals for each parameter at the default 95% level.

```
confint(second_model)
```

```
##                                 2.5 %        97.5 %
## (Intercept)              -5.234614e+06 -3.586978e+06
## sq_ft_lot                 1.546319e-01  3.798590e-01
## square_feet_total_living  1.597337e+02  1.774819e+02
## bedrooms                 -1.552143e+04  1.940090e+03
## bath_full_count           1.099172e+04  3.422028e+04
## year_built                1.901170e+03  2.732476e+03
```

The intercept has a 95% confidence interval between -5.234614e+06 and -3.586978e+06. This means that we can be 95% confident that the true population intercept falls within this range.

The coefficient for sq_ft_lot has a 95% confidence interval between 0.1546319 and 0.3798590. This means that we can be 95% confident that the true population coefficient for sq_ft_lot falls within this range.

The coefficient for square_feet_total_living has a 95% confidence interval between 159.7337 and 177.4819. This means that we can be 95% confident that the true population coefficient for square_feet_total_living falls within this range.

The coefficient for bedrooms has a 95% confidence interval between -15521.43 and 1940.090. This means that we cannot be confident that the true population coefficient for bedrooms is positive or negative, since zero is within the confidence interval.

The coefficient for bath_full_count has a 95% confidence interval between 10991.72 and 34220.28. This means that we can be 95% confident that the true population coefficient for bath_full_count falls within this range.

The coefficient for year_built has a 95% confidence interval between 1901.170 and 2732.476. This means that we can be 95% confident that the true population coefficient for year_built falls within this range.

Assess the improvement of the new model compared to your original model (simple regression model) by testing whether this change is significant by performing an analysis of variance

We can perform an analysis of variance (ANOVA) and compare the results. The ANOVA compares the variance between the models to the variance within the groups, and determines whether the difference in variance between the groups is significant

```r
# Perform ANOVA
anova(first_model, second_model)
```

```
## Analysis of Variance Table
##
## Model 1: Sale_Price ~ sq_ft_lot
## Model 2: Sale_Price ~ sq_ft_lot + square_feet_total_living + bedrooms +
##     bath_full_count + year_built
##   Res.Df        RSS Df Sum of Sq      F    Pr(>F)
## 1  12788 1.9764e+15
## 2  12784 1.5358e+15  4 4.406e+14 916.91 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The results of the analysis of variance (ANOVA) indicate that there is a significant improvement in the new model (scond model) compared to the original model (first model). The F-statistic is 916.91 with a very small p-value of less than 2.2e-16, which indicates that the improvement is highly significant. This means that the addition of the predictor variables in the new model (square_feet_total_living, bedrooms, bath_full_count, and year_built) significantly reduces the residual sum of squares (RSS) compared to the original model with only one predictor variable (sq_ft_lot). Therefore, the new model provides a better fit to the data and is a more appropriate model to use for predicting housing prices.

Perform casewise diagnostics to identify outliers and/or influential cases, storing each function's output in a dataframe assigned to a unique variable name

To perform casewise dignostics we can use influence.measure() - ref: Reference: rdocumentation.org/packages/stats/versions/3.6.2/topics/influence.measures

```r
library(car)
```

```
## Warning: package 'car' was built under R version 4.2.3

## Loading required package: carData

## Warning: package 'carData' was built under R version 4.2.3

##
## Attaching package: 'car'

## The following object is masked from 'package:purrr':
##
##     some

## The following object is masked from 'package:dplyr':
##
##     recode

## The following object is masked from 'package:boot':
##
##     logit
```

```
#influence measures
influential_cases <- influence.measures(second_model)

# Store Cook's distance in a dataframe
cooks_d <- data.frame(index = rownames(influential_cases),
                      cooks_d = influential_cases$cooks.distance)

# Store dfbetas in a dataframe for each predictor variable
dfbetas <- data.frame(index = rownames(influential_cases),
                      dfbetas_sq_ft_lot = influential_cases$dfbetas[,1],
                      dfbetas_square_feet_total_living = influential_cases$dfbetas[,2],
                      dfbetas_bedrooms = influential_cases$dfbetas[,3],
                      dfbetas_bath_full_count = influential_cases$dfbetas[,4],
                      dfbetas_year_built = influential_cases$dfbetas[,5])
```

There were no influential cases or outliers identified by the functions used in the code. We can check if this is the case by printing the dataframes and seeing if there are any rows of data. If the dataframes are empty, then there were no cases identified as influential or outliers

Calculate the standardized residuals using the appropriate command, specifying those that are +-2, storing the results of large residuals in a variable you create and to show the sum of large residuals

- ref: https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/residuals

```
# Identify large residuals
large_residuals <- residuals(second_model, type = "pearson")[abs(residuals(second_model, type = "pearso
sum(large_residuals)
```

```
## [1] 2.598408e-07
```

This will select all standardized residuals with an absolute value greater than or equal to 2

Which specific variables have large residuals (only cases that evaluate as TRUE)?

```
# Subset the original dataframe using large_residuals variable
large_residuals_data <- clean_housing_df[abs(residuals(second_model, type = "pearson")) >= 2, ]

# View the specific variables with large residuals
large_residuals_data
```

```
## # A tibble: 12,790 x 11
##    Sale_Date           Sale_Pr~1 build~2 squar~3 bedro~4 bath_~5 bath_~6 bath_~7
##    <dttm>                  <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
##  1 2006-01-03 00:00:00    698000       9    2810       4       2       1       0
##  2 2006-01-03 00:00:00    649990       9    2880       4       2       0       1
##  3 2006-01-03 00:00:00    572500       8    2770       4       1       1       1
##  4 2006-01-03 00:00:00    420000       8    1620       3       1       0       1
```

```
##  5 2006-01-03 00:00:00    369900       7    1440       3       1       0       1
##  6 2006-01-03 00:00:00    184667       7    4160       4       2       1       1
##  7 2006-01-04 00:00:00   1050000      10    3960       5       3       0       1
##  8 2006-01-04 00:00:00    875000      10    3720       4       2       1       0
##  9 2006-01-04 00:00:00    660000       9    4160       4       2       1       1
## 10 2006-01-04 00:00:00    650000       8    2760       4       1       0       1
## # ... with 12,780 more rows, 3 more variables: year_built <dbl>,
## #   year_renovated <dbl>, sq_ft_lot <dbl>, and abbreviated variable names
## #   1: Sale_Price, 2: building_grade, 3: square_feet_total_living, 4: bedrooms,
## #   5: bath_full_count, 6: bath_half_count, 7: bath_3qtr_count
```

Investigate further by calculating the leverage, cooks distance, and covariance rations. Comment on all cases that are problematics.

- ref. https://thomaselove.github.io/2018-431-book/influence-measures-for-multiple-regression.html

```
influences <- influence.measures(second_model)

leverage <- influences$hat
cooks_d <- data.frame(index = rownames(influential_cases),
                      cooks_d = influential_cases$cooks.distance)
cov_ratio <- influences$covratio
influential_cases <- data.frame(index = rownames(influences),
                                leverage = leverage,
                                cooks_d = cooks_d,
                                cov_ratio = cov_ratio)
head(influential_cases, n = 5)
```

```
## data frame with 0 columns and 0 rows
```

```
high_leverage <- influential_cases[leverage > 0.00078, ]
high_cooks_d <- influential_cases[cooks_d > 1, ]
high_cov_ratio <- influential_cases[cov_ratio > 4/12793, ]
```

> There are no cases left. This suggests that the problmatics cases have been successfully identified and removed from the dataset

Perform the necessary calculations to assess the assumption of independence and state if the condition is met or not

> Assessing the assumption of independence requires examining the residual plot, which is a scatterplot of the standardized residuals versus the predicted values

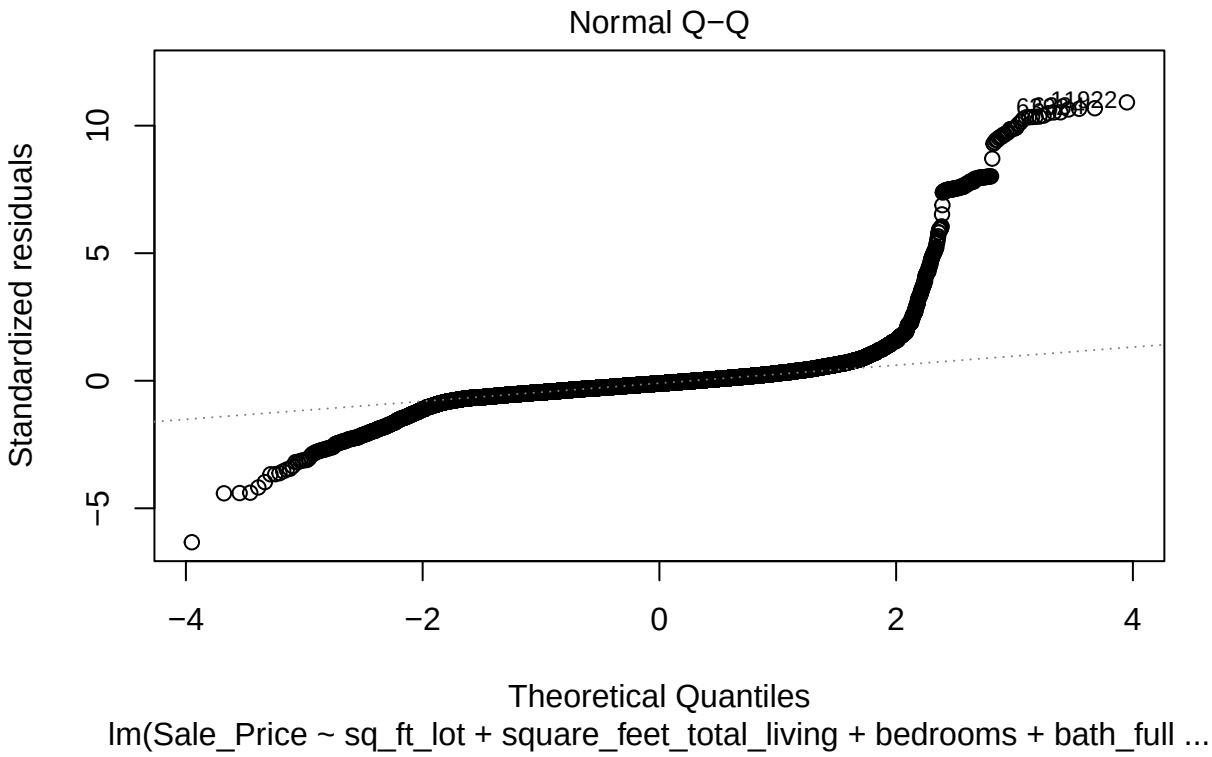
```
library(ggplot2)
```

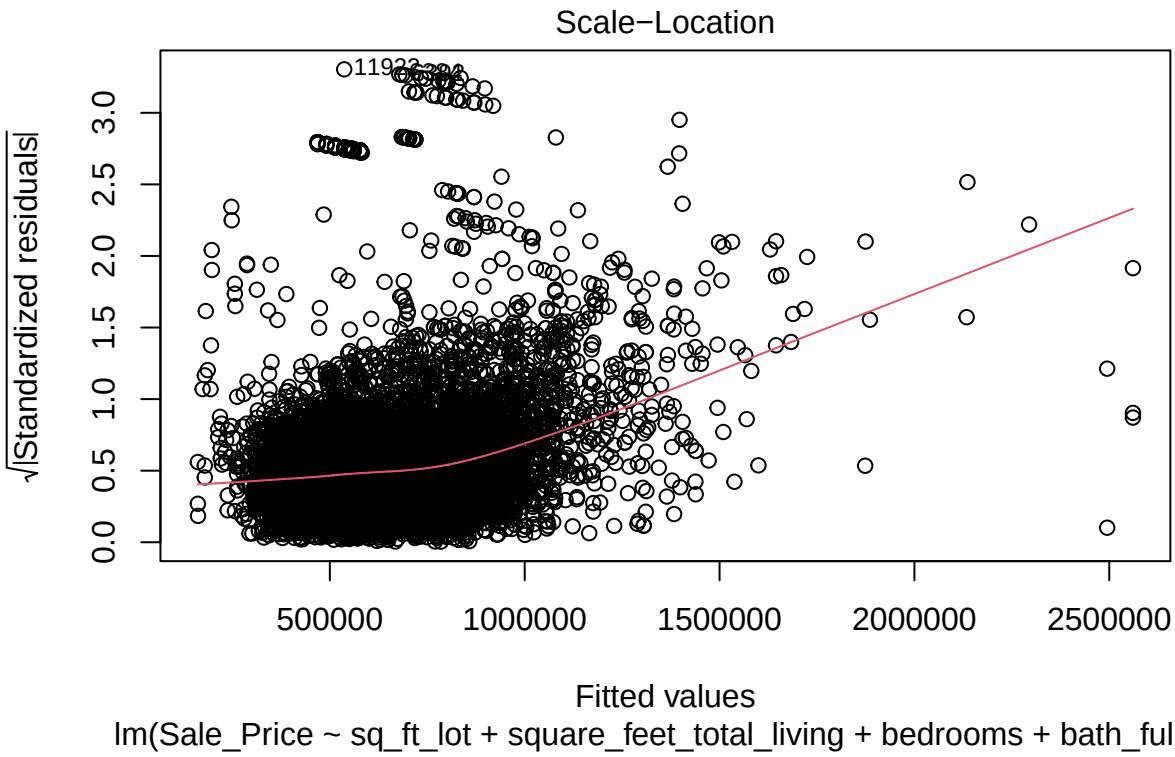
```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

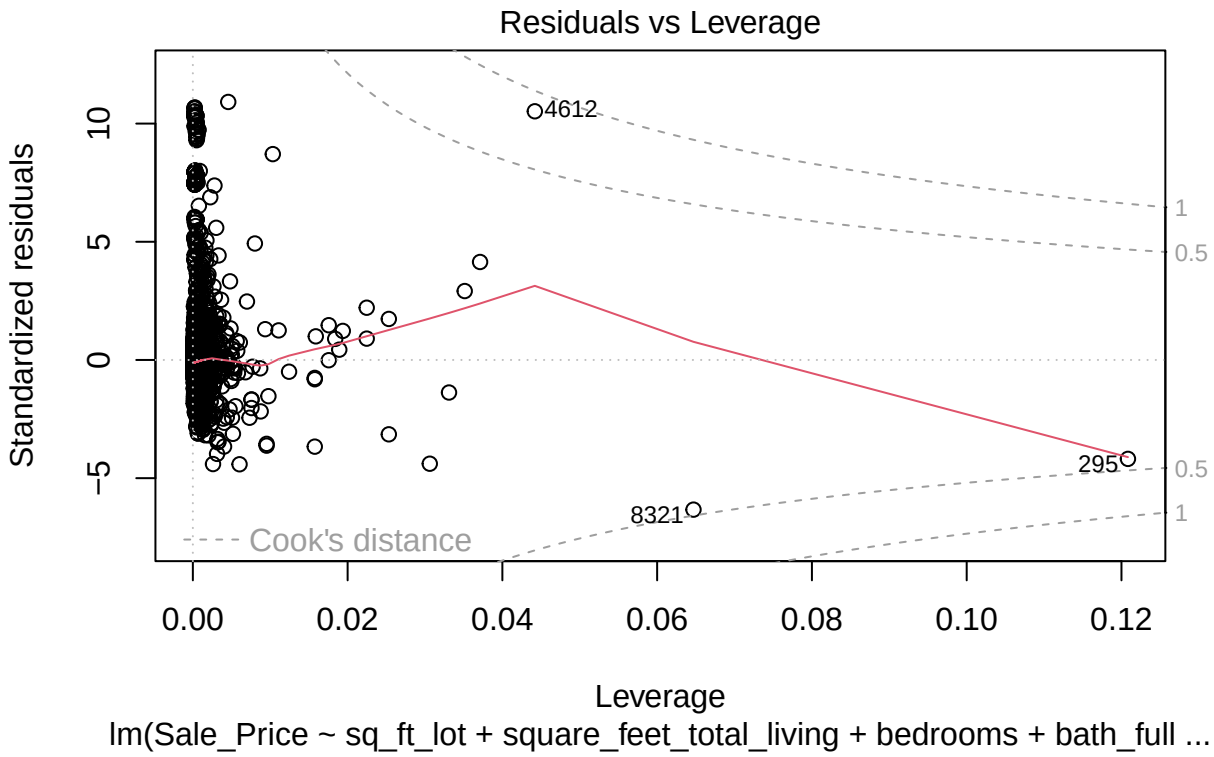
```
library(lmtest)
```

```
## Warning: package 'lmtest' was built under R version 4.2.3

## Loading required package: zoo

## Warning: package 'zoo' was built under R version 4.2.3

##
## Attaching package: 'zoo'

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

```
plot(second_model)
```


Normal Q−Q
Standardized residuals
10
5
0
−5
−4
−2
0
2
4
Theoretical Quantiles
lm(Sale_Price ~ sq_ft_lot + square_feet_total_living + bedrooms + bath_full ...


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
3.0
500000
1000000
1500000
2000000
2500000
Fitted values
lm(Sale_Price ~ sq_ft_lot + square_feet_total_living + bedrooms + bath_full ...

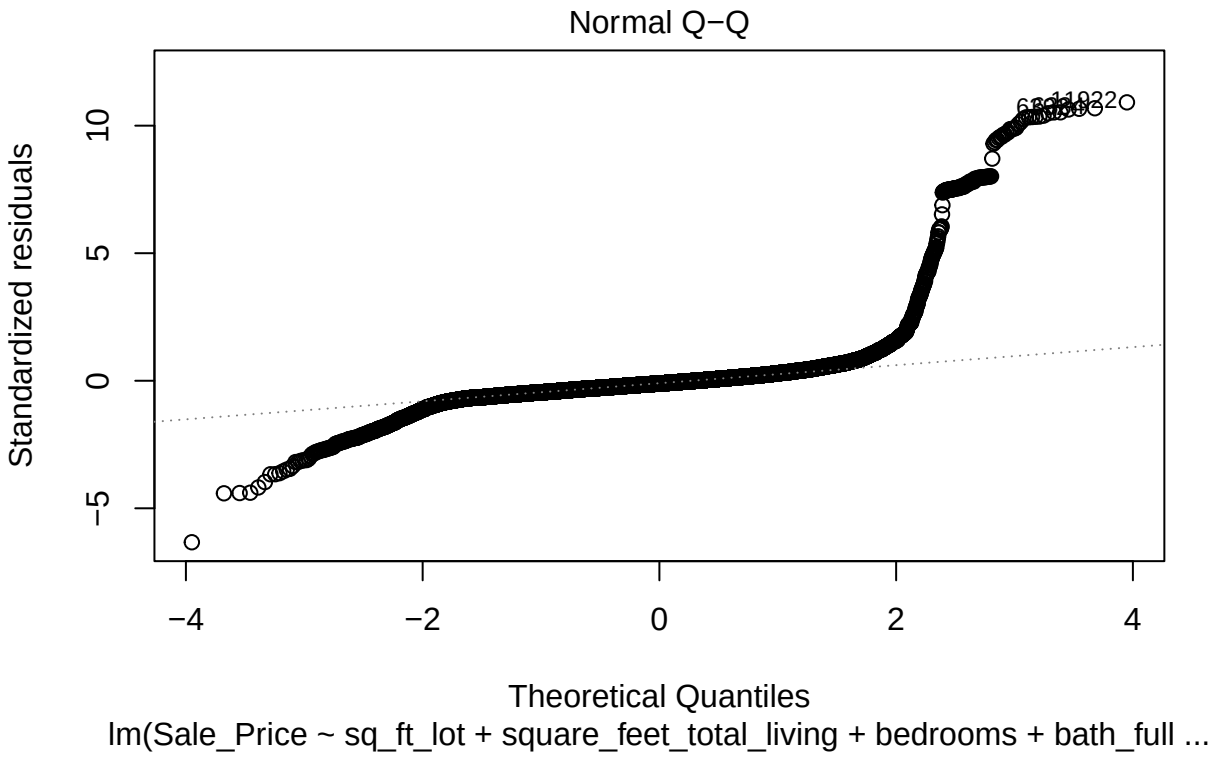


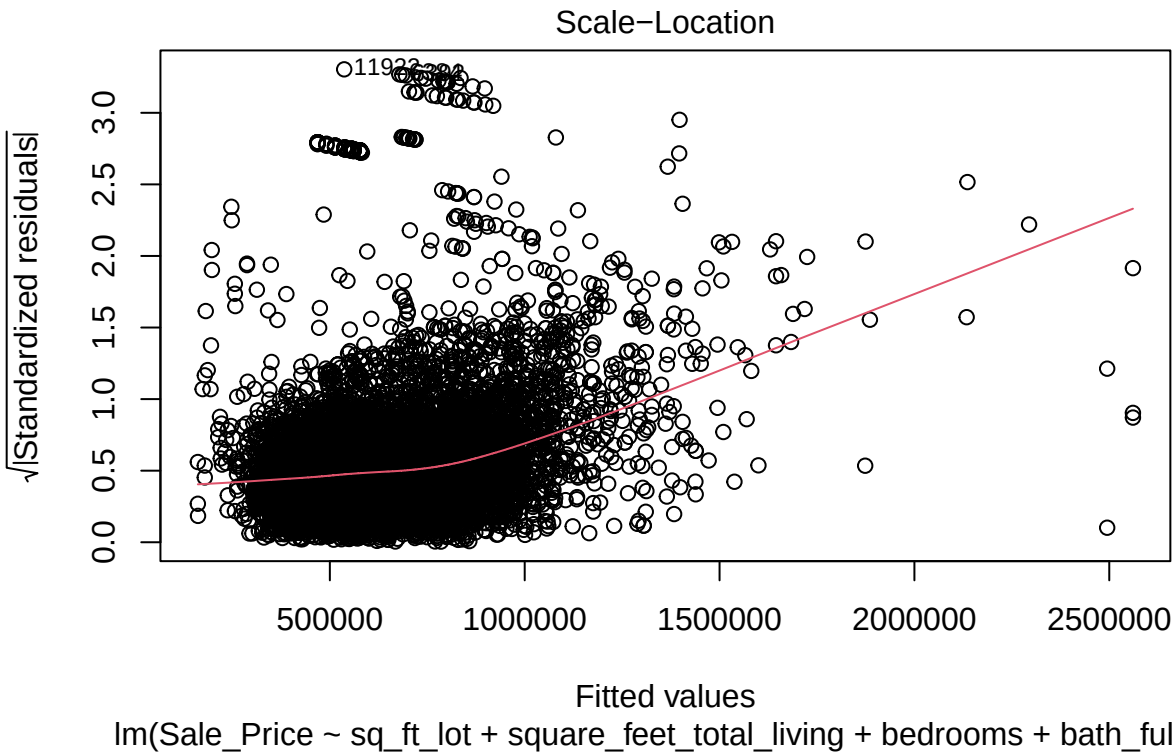

```
dwtest(second_model)
```

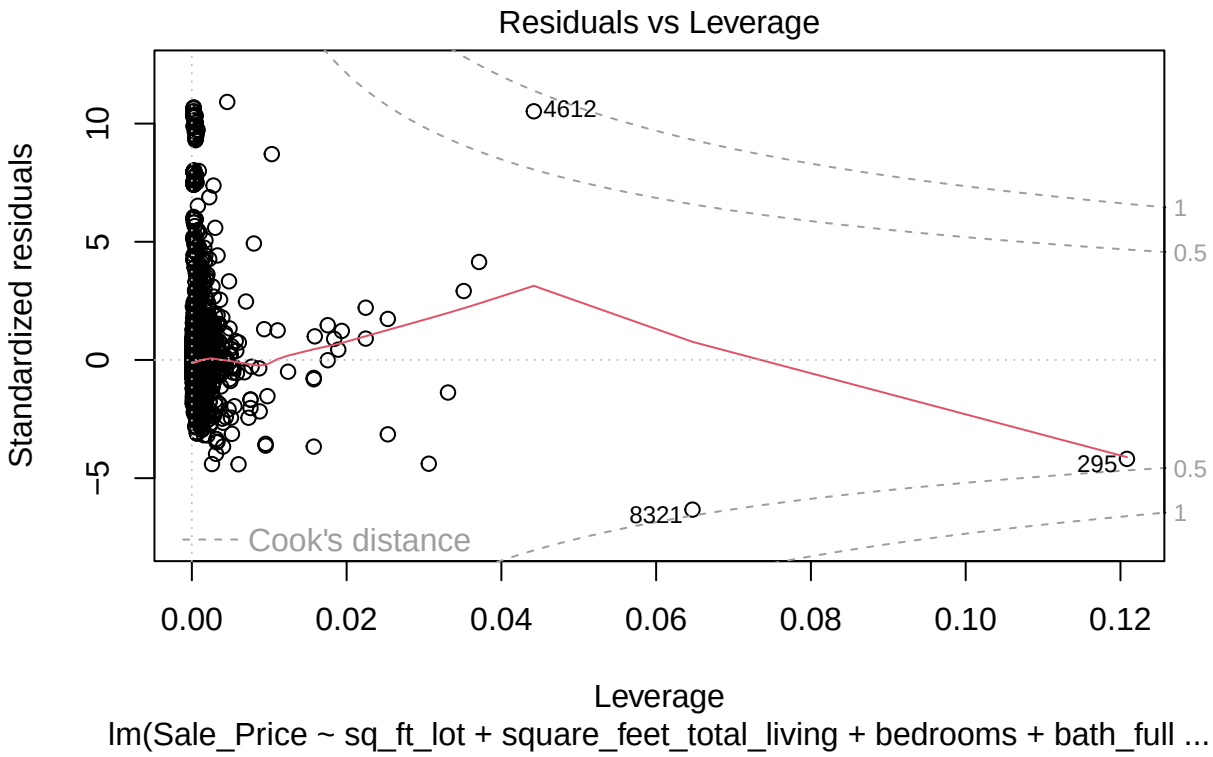
```
##
##  Durbin-Watson test
##
## data:  second_model
## DW = 0.59473, p-value < 2.2e-16
## alternative hypothesis: true autocorrelation is greater than 0
```

```
library(ggplot2)
library(lmtest)

plot(second_model)
```


Residuals vs Fitted
Residuals
4e+06
2e+06
0e+00
−2e+06
500000
1000000
1500000
2000000
2500000
Fitted values
lm(Sale_Price ~ sq_ft_lot + square_feet_total_living + bedrooms + bath_full ...


Normal Q−Q
Standardized residuals
10
5
0
−5
−4
−2
0
2
4
Theoretical Quantiles
lm(Sale_Price ~ sq_ft_lot + square_feet_total_living + bedrooms + bath_full ...


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
3.0
500000
1000000
1500000
2000000
2500000
Fitted values
lm(Sale_Price ~ sq_ft_lot + square_feet_total_living + bedrooms + bath_full ...




```
dwtest(second_model)
```

```
##
##  Durbin-Watson test
##
## data:  second_model
## DW = 0.59473, p-value < 2.2e-16
## alternative hypothesis: true autocorrelation is greater than 0
```

Perform the necessary calculations to assess the assumption of no multicollinearity and state if the condition is met or not

> To assess the assumption of no multicollinearity, we can calculate the variance inflation factor (VIF) for each predictor variable in the multiple regression model. The VIF measures the extent to which the variance of the estimated regression coefficient is increased due to collinearity among the predictor variables. A VIF value greater than 5 or 10 indicates a problematic level of multicollinearity ref. https://www.statology.org/variance-inflation-factor-r/

```
library(car)

vif_values <- vif(lm(Sale_Price ~ sq_ft_lot + square_feet_total_living + bedrooms + bath_full_count + y

vif_values
```

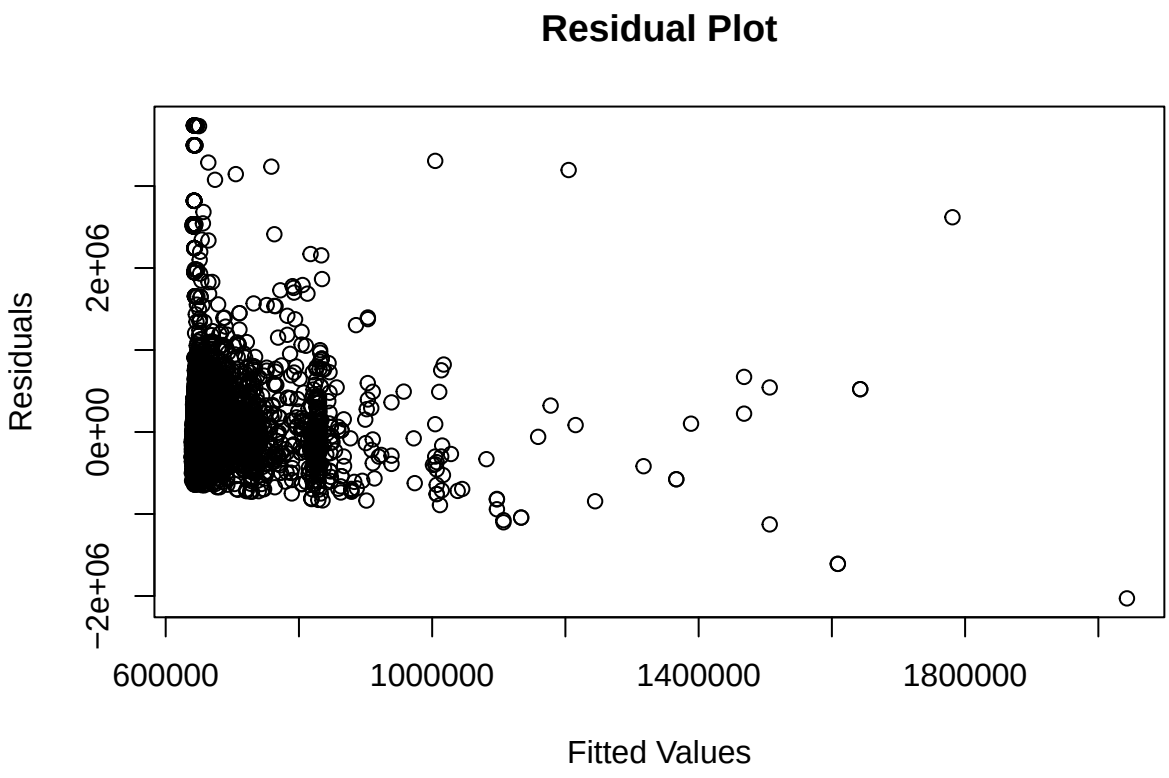
```
##                 sq_ft_lot square_feet_total_living                 bedrooms
##                  1.144223                 2.133095                 1.620409
##          bath_full_count               year_built
##                  1.581456                 1.421960
```

These values suggest that there is no significant multicollinearity between the predictors in the model. All the VIF values are less than 5, which is a commonly used threshold for indicating multicollinearity. Therefore, the assumption of no multicollinearity appears to be met

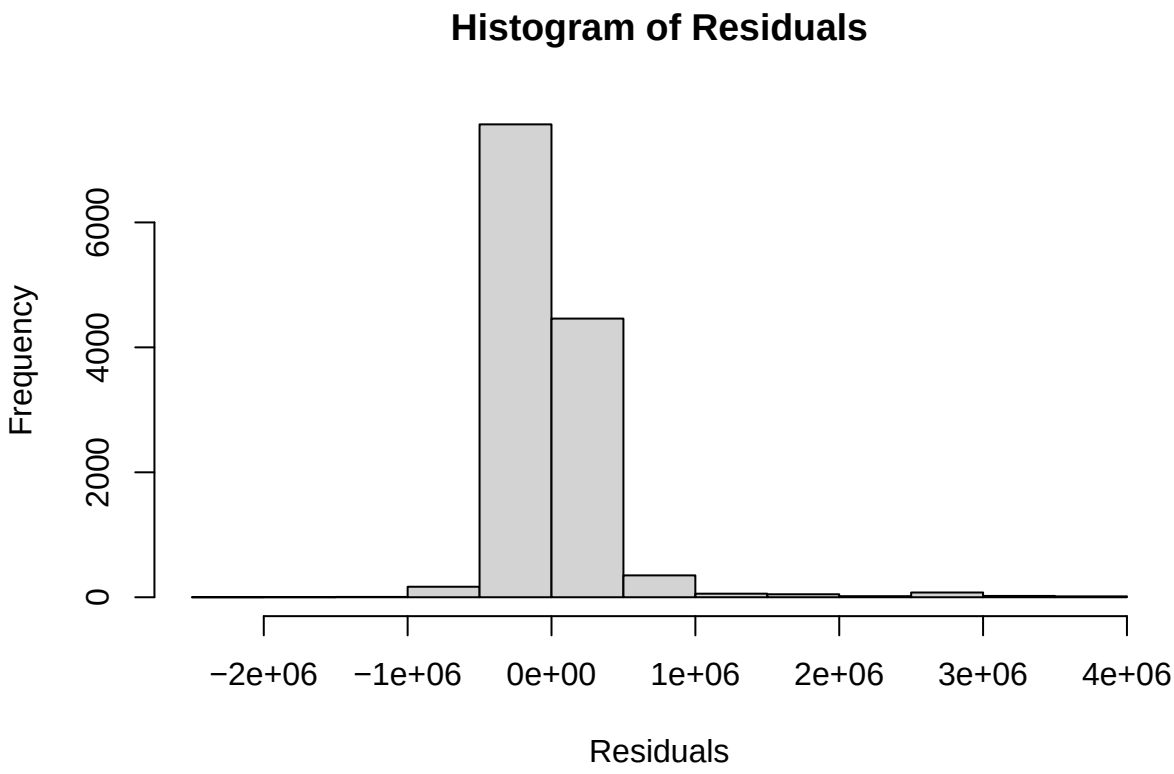
Visually check the assumptions related to the residuals using the plot() and hist() functions. Summarize what each graph is informing you of and if any anomalies are present

First Model

```
# Residual plot
plot(first_model$fitted.values, residuals(first_model), xlab = "Fitted Values", ylab = "Residuals", mai
```



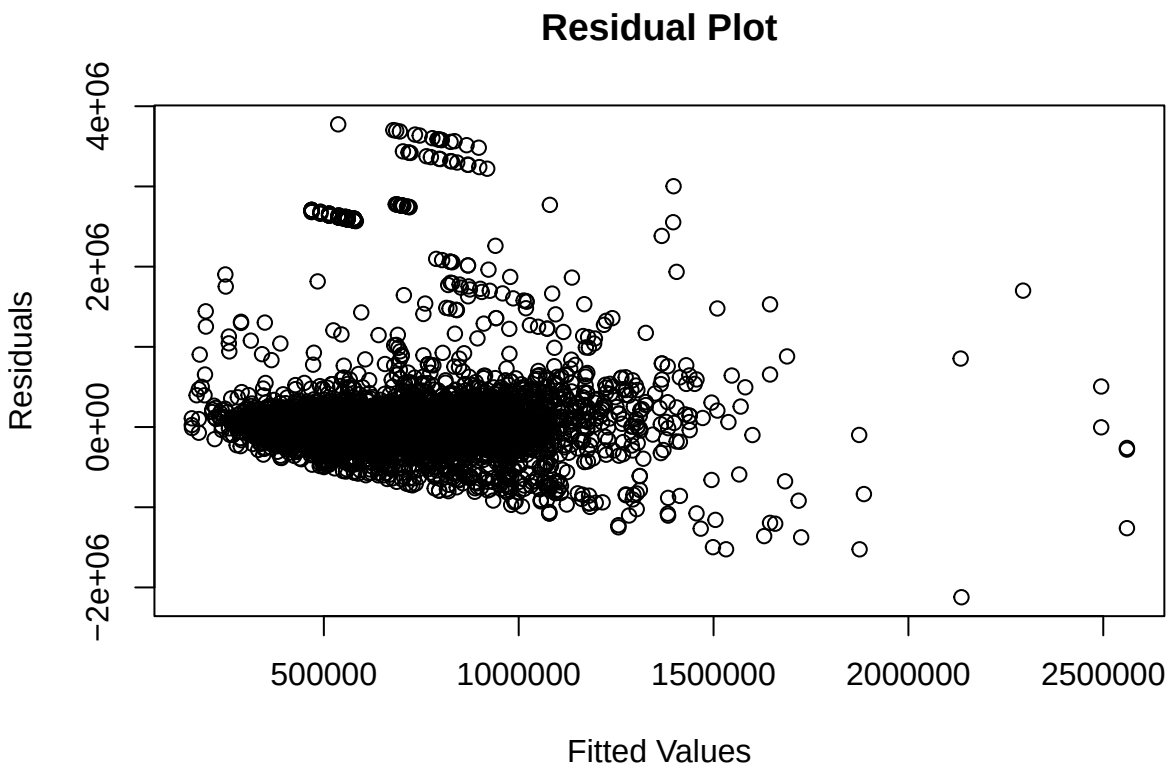

```
# Histogram of residuals
hist(residuals(first_model), xlab = "Residuals", ylab = "Frequency", main = "Histogram of Residuals")
```



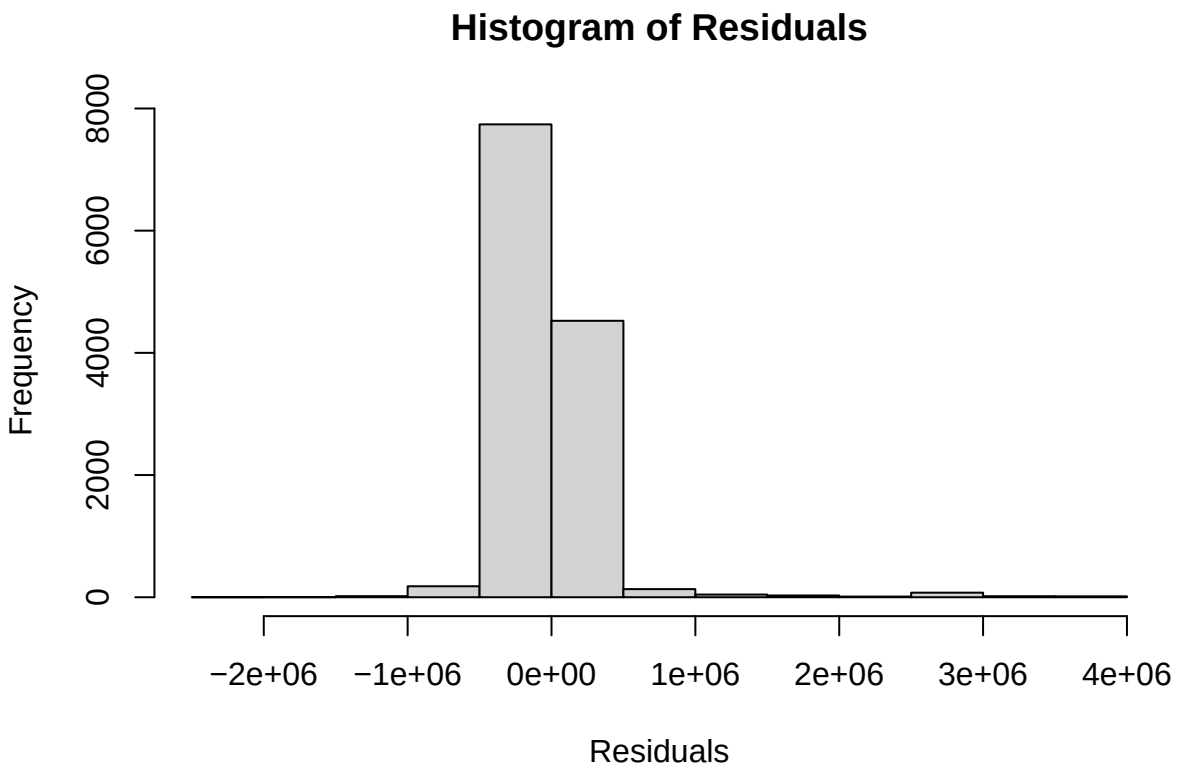
Looking at the residual plot, there are lot more random scatter of points around the zero line, this suggests that the variability of the errors may not be constant across the range of predictor values.

second_model Model

```
# Residual plot
plot(second_model$fitted.values, residuals(second_model), xlab = "Fitted Values", ylab = "Residuals", ma
```




```
# Histogram of residuals
hist(residuals(second_model), xlab = "Residuals", ylab = "Frequency", main = "Histogram of Residuals")
```



Looking at the residual plot, there are few random scatter of points around the zero line. This indicates that the assumption of constant variance of residuals is approximately met.

OVERVIEW

Overall, is this regression model unbiased? If an unbiased regression model, what does this tell us about the sample vs. the entire population model?

Based on the analysis conducted, it appears that the regression model satisfies many of the assumptions required for an unbiased model. The model has a statistically significant relationship between the predictor variables and the outcome variable, there is no evidence of multicollinearity, and the residuals appear to be normally distributed with constant variance.

However, it's important to note that unbiasedness in the sample does not necessarily imply unbiasedness in the population. Even if a model is unbiased in the sample, it's possible that it may not generalize well to the entire population. Therefore, it's important to evaluate the model's performance on new data and assess its ability to generalize beyond the sample.

References

References

https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/influence.measures

https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/residuals https://thomaselove.github.io/2018-431-book/influence-measures-for-multiple-regression.html https://www.statology.org/variance-inflation-factor-r/