Assignment 8.2

Aarti Ramani

2023-02-11

```
library(readxl)
## Set the working directory to the root of your DSC 520 directory
#setwd("C:/Masters/GitHub/Winter2022/Ramani-DSC520")

## Load housing data
housing_df <- read_excel(path = "C:/Masters/GitHub/Winter2022/Ramani-DSC520/data/week-6-housing.xlsx",
                         .name_repair = function(col){ gsub(" ", "_", col) })
names(housing_df)
```

```
##  [1] "Sale_Date"                "Sale_Price"
##  [3] "sale_reason"              "sale_instrument"
##  [5] "sale_warning"             "sitetype"
##  [7] "addr_full"                "zip5"
##  [9] "ctyname"                  "postalctyn"
## [11] "lon"                      "lat"
## [13] "building_grade"           "square_feet_total_living"
## [15] "bedrooms"                 "bath_full_count"
## [17] "bath_half_count"          "bath_3qtr_count"
## [19] "year_built"               "year_renovated"
## [21] "current_zoning"           "sq_ft_lot"
## [23] "prop_type"                "present_use"
```

i. Explain any transformations or modifications you made to the dataset

```
library(dplyr)
```

```
##
## 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
```

```r
housing_df <- housing_df %>%
  mutate(ctyname = case_when((is.na(ctyname) & zip5 == 98052) ~ 'REDMOND',
                             (is.na(ctyname) & zip5 == 98053) ~ 'REDMOND',
                             (is.na(ctyname) & zip5 == 98074) ~ 'SAMMAMISH',
                             (is.na(ctyname) & zip5 == 98059) ~ 'RENTON',
                             TRUE~ctyname))
```

Updated missing city names based on zipcode.

ii.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.

```r
price_vs_sqf_lm <- lm(Sale_Price  ~ square_feet_total_living, data = housing_df)
price_vs_sqf_lm
```

```
##
## Call:
## lm(formula = Sale_Price ~ square_feet_total_living, data = housing_df)
##
## Coefficients:
##              (Intercept)  square_feet_total_living
##                 189106.6                     185.7
```

```r
price_and_more <- lm(Sale_Price~square_feet_total_living + bedrooms +
                       bath_half_count + bath_full_count, data=housing_df)
price_and_more
```

```
##
## Call:
## lm(formula = Sale_Price ~ square_feet_total_living + bedrooms +
##     bath_half_count + bath_full_count, data = housing_df)
##
## Coefficients:
##              (Intercept)  square_feet_total_living                  bedrooms
##                 202179.8                     181.8                  -24937.1
##          bath_half_count           bath_full_count
##                  14655.8                   41444.2
```

As additional predictors, I'm using bedrooms and bathrooms as these fields make a significant impact on the sale price.

iii. 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?

```r
summary(price_vs_sqf_lm)
```

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

```
## lm(formula = Sale_Price ~ square_feet_total_living, data = housing_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1800136  -120257   -41547    44028  3811745
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)              1.891e+05  8.745e+03   21.62   <2e-16 ***
## square_feet_total_living 1.857e+02  3.208e+00   57.88   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 360200 on 12863 degrees of freedom
## Multiple R-squared:  0.2066, Adjusted R-squared:  0.2066
## F-statistic:  3351 on 1 and 12863 DF,  p-value: < 2.2e-16
```

```
summary(price_and_more)
```

```
##
## Call:
## lm(formula = Sale_Price ~ square_feet_total_living + bedrooms +
##     bath_half_count + bath_full_count, data = housing_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1766785  -118681   -41745    43659  3823860
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)              202179.839  14052.978  14.387  < 2e-16 ***
## square_feet_total_living    181.839      4.466  40.712  < 2e-16 ***
## bedrooms                 -24937.119   4418.206  -5.644 1.69e-08 ***
## bath_half_count           14655.841   6356.188   2.306   0.0211 *
## bath_full_count           41444.194   5696.926   7.275 3.67e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 359000 on 12860 degrees of freedom
## Multiple R-squared:  0.2123, Adjusted R-squared:  0.2121
## F-statistic: 866.6 on 4 and 12860 DF,  p-value: < 2.2e-16
```

R-squared: 0.2066, Adjusted R-squared: 0.2066 for price and sq feet total size. This tells us there is a sale price accounts to around 21% for sq feet total size.

R-squared: 0.2123, Adjusted R-squared: 0.2121 This tells us there is a sale price accounts to around 21% for sq feet total size, bedroom and bathroom. Which means 79% of the sale cannot be explained by these predictors alone. This correlation is however is slightly better than the 20.6%% for price and sq feet total size.

iv. 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?

```r
library(QuantPsyc)
```

```
## Loading required package: boot

## Loading required package: purrr

## 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
```

```r
lm.beta(price_vs_sqf_lm)
```

```
## square_feet_total_living
##                0.4545876
```

```r
lm.beta(price_and_more)
```

```
## square_feet_total_living                 bedrooms           bath_half_count
##               0.44509353              -0.05402847                0.01907365
##          bath_full_count
##               0.06669881
```

```r
##coefficients(price_and_more)
```

square_feet_total_living has a significant relation to the sale price, implying they have a comparable degree of importance.

The standardized betas for the linear model "price_and_more" indicates the sale price increases by 0.44509353 standard deviations when there is an increase in standard deviations for the property's size by total living square feet.

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

```r
confint(price_vs_sqf_lm)
```

```
##                                2.5 %      97.5 %
## (Intercept)              171965.2516 206247.8664
## square_feet_total_living    179.4286    192.0067
```

```r
confint(price_and_more)
```

```
##                                2.5 %      97.5 %
## (Intercept)              174633.9160 229725.7626
## square_feet_total_living    173.0841    190.5938
## bedrooms                 -33597.4597 -16276.7793
## bath_half_count            2196.7687  27114.9132
## bath_full_count           30277.3727  52611.0151
```

The confidence intervals calculated for "price_vs_sqf_lm" have a small range. This indicates that the predictor's *b* value is close to the real *b* value. The confidence intervals calculated for "price_and_more" have a larger range. In addition, these values cross zero and include negative values. This indicates that the sale price can increase or decrease depending on the number of bedrooms. This makes the output for sale price not consistent. However, the other variables do have better consistency and shorter range.

vi. 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.

```r
anova(price_vs_sqf_lm)
```

```
## Analysis of Variance Table
##
## Response: Sale_Price
##                             Df     Sum Sq    Mean Sq F value    Pr(>F)
## square_feet_total_living     1 4.3470e+14 4.3470e+14  3350.5 < 2.2e-16 ***
## Residuals                12863 1.6689e+15 1.2974e+11
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
anova(price_and_more)
```

```
## Analysis of Variance Table
##
## Response: Sale_Price
##                             Df     Sum Sq    Mean Sq   F value    Pr(>F)
## square_feet_total_living     1 4.3470e+14 4.3470e+14 3373.8299 < 2.2e-16 ***
## bedrooms                     1 4.0929e+12 4.0929e+12   31.7663 1.775e-08 ***
## bath_half_count              1 1.0038e+12 1.0038e+12    7.7909  0.005259 **
## bath_full_count              1 6.8189e+12 6.8189e+12   52.9232 3.669e-13 ***
## Residuals                12860 1.6570e+15 1.2885e+11
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
anova(price_vs_sqf_lm, price_and_more)
```

```
## Analysis of Variance Table
##
## Model 1: Sale_Price ~ square_feet_total_living
## Model 2: Sale_Price ~ square_feet_total_living + bedrooms + bath_half_count +
##     bath_full_count
```

```
##   Res.Df        RSS Df  Sum of Sq      F    Pr(>F)
## 1  12863 1.6689e+15
## 2  12860 1.6570e+15  3 1.1916e+13 30.827 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

A non zero F statistic means significant coefficients. P value < 0.05 also implies a significant model. In this case, all predictors have a p value less than 0.05.

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

```r
housing_df_outliers <-  housing_df
names(housing_df_outliers)
```

```
##  [1] "Sale_Date"                "Sale_Price"
##  [3] "sale_reason"              "sale_instrument"
##  [5] "sale_warning"             "sitetype"
##  [7] "addr_full"                "zip5"
##  [9] "ctyname"                  "postalctyn"
## [11] "lon"                      "lat"
## [13] "building_grade"           "square_feet_total_living"
## [15] "bedrooms"                 "bath_full_count"
## [17] "bath_half_count"          "bath_3qtr_count"
## [19] "year_built"               "year_renovated"
## [21] "current_zoning"           "sq_ft_lot"
## [23] "prop_type"                "present_use"
```

```r
#housing_df_outliers <- na.omit(housing_df_outliers)
nrow(housing_df_outliers)
```

```
## [1] 12865
```

```r
nrow(housing_df)
```

```
## [1] 12865
```

```r
#rsid
housing_df_outliers$residuals <- resid(price_and_more)

#standardized residuals
housing_df_outliers$stand_residuals <-  rstandard(price_and_more)
#studentized residuals
housing_df_outliers$studentized_residuals <- rstudent(price_and_more)
#cooks distance
housing_df_outliers$cooks_distance <- cooks.distance(price_and_more)
#DFBeta
housing_df_outliers$dfbeta <- dfbeta(price_and_more)
#DFFit
housing_df_outliers$dffits <- dffits(price_and_more)
#hat values (leverage)
```

```r
housing_df_outliers$leverage <- hatvalues(price_and_more)
#covariance ratio
housing_df_outliers$covariance_ratio <- covratio(price_and_more)

names(housing_df_outliers)
```

```
##  [1] "Sale_Date"                "Sale_Price"
##  [3] "sale_reason"              "sale_instrument"
##  [5] "sale_warning"             "sitetype"
##  [7] "addr_full"                "zip5"
##  [9] "ctyname"                  "postalctyn"
## [11] "lon"                      "lat"
## [13] "building_grade"           "square_feet_total_living"
## [15] "bedrooms"                 "bath_full_count"
## [17] "bath_half_count"          "bath_3qtr_count"
## [19] "year_built"               "year_renovated"
## [21] "current_zoning"           "sq_ft_lot"
## [23] "prop_type"                "present_use"
## [25] "residuals"                "stand_residuals"
## [27] "studentized_residuals"    "cooks_distance"
## [29] "dfbeta"                   "dffits"
## [31] "leverage"                 "covariance_ratio"
```

```r
nrow(housing_df_outliers)
```

```
## [1] 12865
```

```r
head(housing_df_outliers)
```

```
## # A tibble: 6 x 32
##   Sale_Date           Sale_Price sale_re~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
## 6 2006-01-03 00:00:00     184667         1      15 18 51   R1      8101 2~ 98053
## # ... with 24 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>,
## #   residuals <dbl>, stand_residuals <dbl>, studentized_residuals <dbl>,
## #   cooks_distance <dbl>, dfbeta <dbl[,5]>, dffits <dbl>, leverage <dbl>, ...
```

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

```r
housing_df_outliers$residual_flag <-
  housing_df_outliers$stand_residuals > 2|housing_df_outliers$stand_residuals < -2
```

ix. Use the appropriate function to show the sum of large residuals.

```r
sum(housing_df_outliers$residual_flag)
```

```
## [1] 320
```

320 cases have a large residual (>2 and <-2)

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

```r
housing_df_outliers[housing_df_outliers$residual_flag==TRUE,
                      c("Sale_Price","square_feet_total_living","bedrooms",
                        "bath_full_count" ,"bath_half_count","stand_residuals")]
```

```
## # A tibble: 320 x 6
##    Sale_Price square_feet_total_living bedrooms bath_full_count bath_h~1 stand~2
##         <dbl>                    <dbl>    <dbl>           <dbl>    <dbl>   <dbl>
##  1     184667                     4160        4               2        1   -2.15
##  2     265000                     4920        4               4        1   -2.54
##  3    1390000                      660        0               1        0    2.86
##  4     390000                     5800        5               4        1   -2.57
##  5    1588359                     3360        2               2        1    2.03
##  6    1450000                      900        2               1        0    3.04
##  7     163000                     4710        4               2        1   -2.49
##  8     270000                     5060        4              23        1   -5.07
##  9     200000                     6880        5               1        1   -3.31
## 10     187000                     5140        4               2        1   -2.64
## # ... with 310 more rows, and abbreviated variable names 1: bath_half_count,
## #   2: stand_residuals
```

```r
nrow(housing_df_outliers[housing_df_outliers$residual_flag==TRUE,
                      c("Sale_Price","square_feet_total_living","bedrooms",
                        "bath_full_count" ,"bath_half_count","stand_residuals")])
```

```
## [1] 320
```

```r
names(housing_df_outliers[housing_df_outliers$residual_flag==TRUE,
                      c("Sale_Price","square_feet_total_living","bedrooms",
                        "bath_full_count" ,"bath_half_count","stand_residuals")])
```

```
## [1] "Sale_Price"               "square_feet_total_living"
## [3] "bedrooms"                 "bath_full_count"
## [5] "bath_half_count"          "stand_residuals"
```

320 rows out of 12865

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

```r
#Cooks Distance
nrow(housing_df_outliers[ (housing_df_outliers$residual_flag==TRUE &
                           housing_df_outliers$cooks_distance > 1),])
```

```
## [1] 0
```

```r
#Average leverage -  (3(k + 1)/n) a
threshold <- (3*(4 + 1)/12865)
#threshold - 0.001165954
nrow(housing_df_outliers[housing_df_outliers$residual_flag==TRUE &
                           housing_df_outliers$leverage > threshold,])
```

```
## [1] 56
```

```r
#Covariance ratio
positive_boundary <- 1 + (3*(4 + 1)/12865)
negative_boundary <- 1 - (3*(4 + 1)/12865)

nrow(housing_df_outliers[housing_df_outliers$residual_flag==TRUE &
                           (housing_df_outliers$covariance_ratio > positive_boundary&
                              housing_df_outliers$covariance_ratio > negative_boundary),])
```

```
## [1] 10
```

COOKS DISTANCE: No rows have cooks distance > 1, so none of the cases is having an undue influence on the model.

LEVERAGE: 264 rows have leverage greater than the threshold

COVARIATIONS RATIO: 366 rows deviate from substantially from the covariance_ratio boundaries.

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

```r
library(car)
```

```
## Loading required package: carData

##
## Attaching package: 'car'

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

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

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

```r
durbinWatsonTest(price_vs_sqf_lm)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.7283092     0.5433798       0
##  Alternative hypothesis: rho != 0
```

```r
durbinWatsonTest(price_and_more)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.7281333     0.5437297       0
##  Alternative hypothesis: rho != 0
```

price_vs_sqf_lm - The value is0.5433798 which is less than 1. This means assumption of independence is not met.

price_and_more - The value is 0.5437297 which is less than 1. This means assumption of independence is not met.

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

```r
library(regclass)
```

```
## Loading required package: bestglm

## Loading required package: leaps

## Loading required package: VGAM

## Loading required package: stats4

## Loading required package: splines

##
## Attaching package: 'VGAM'

## The following object is masked from 'package:car':
##
##     logit

## The following objects are masked from 'package:boot':
##
##     logit, simplex

## Loading required package: rpart

## Loading required package: randomForest

## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

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

## Important regclass change from 1.3:
## All functions that had a . in the name now have an _
## all.correlations -> all_correlations, cor.demo -> cor_demo, etc.
```

```
vif(price_and_more)
```

```
## square_feet_total_living                 bedrooms          bath_half_count
##                 1.951365                 1.496004                 1.117191
##          bath_full_count
##                 1.372392
```

```
1/vif(price_and_more)
```

```
## square_feet_total_living                 bedrooms          bath_half_count
##                0.5124618                0.6684474                0.8951019
##          bath_full_count
##                0.7286550
```

```
mean(vif(price_and_more))
```

```
## [1] 1.484238
```

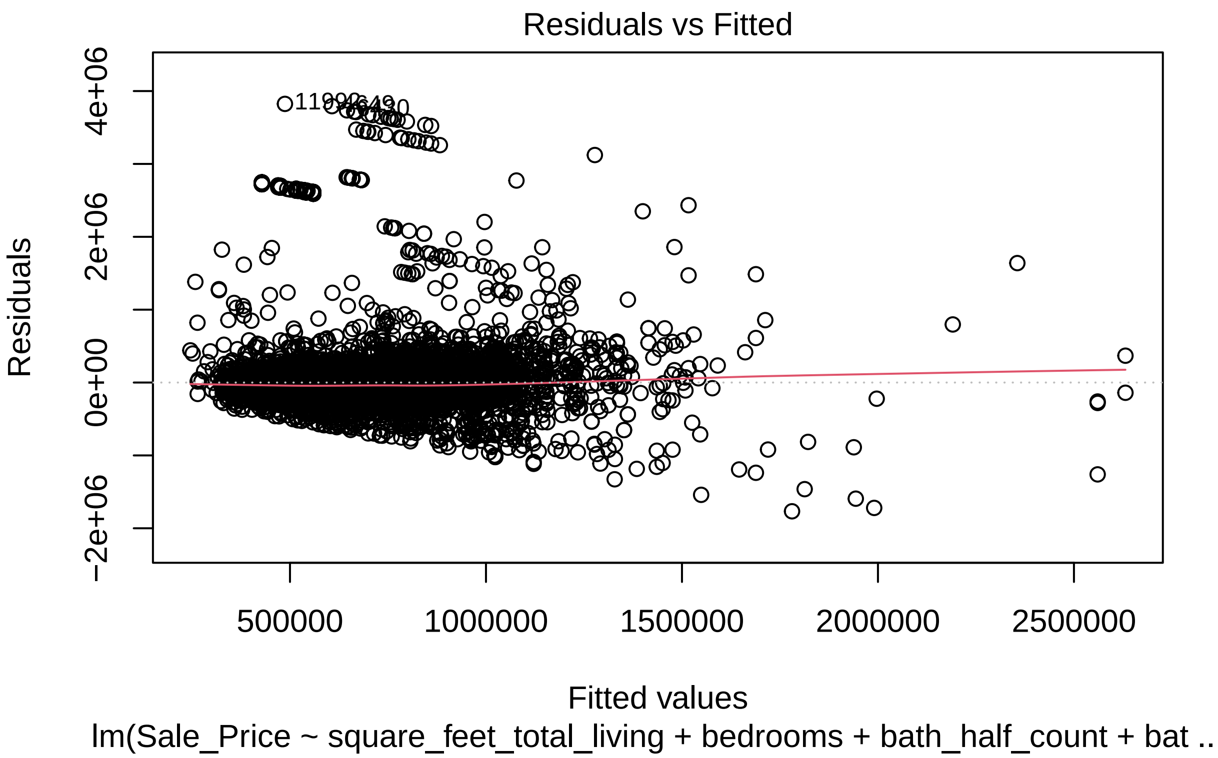
Largest VIF is not greater than 10. Tolerance is not below 0.1 or 0.2. There is no collinearity within the data. Average Vif (1.48) is greater than 1, the regression may be biased.

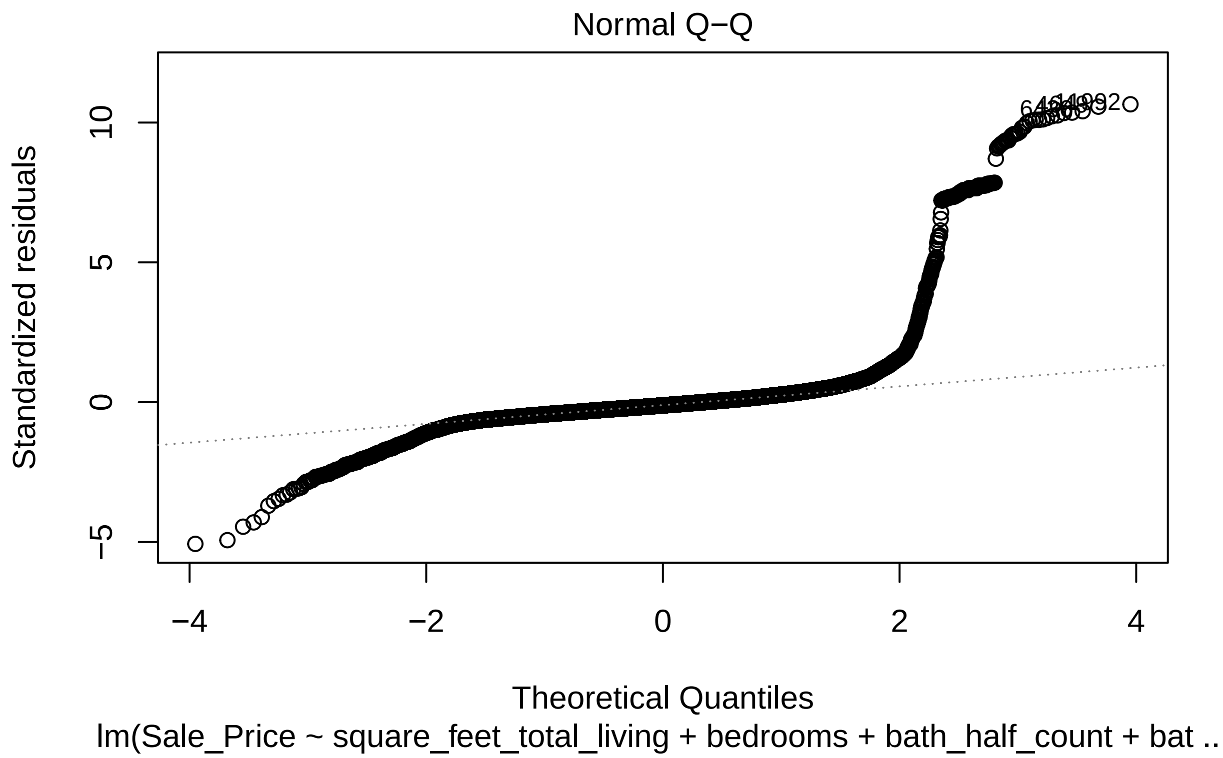
xiv. Visually check the assumptions related to the residuals using the plot() and hist() functions. Summarize what each graph is informing you of and if any anomalies are present.

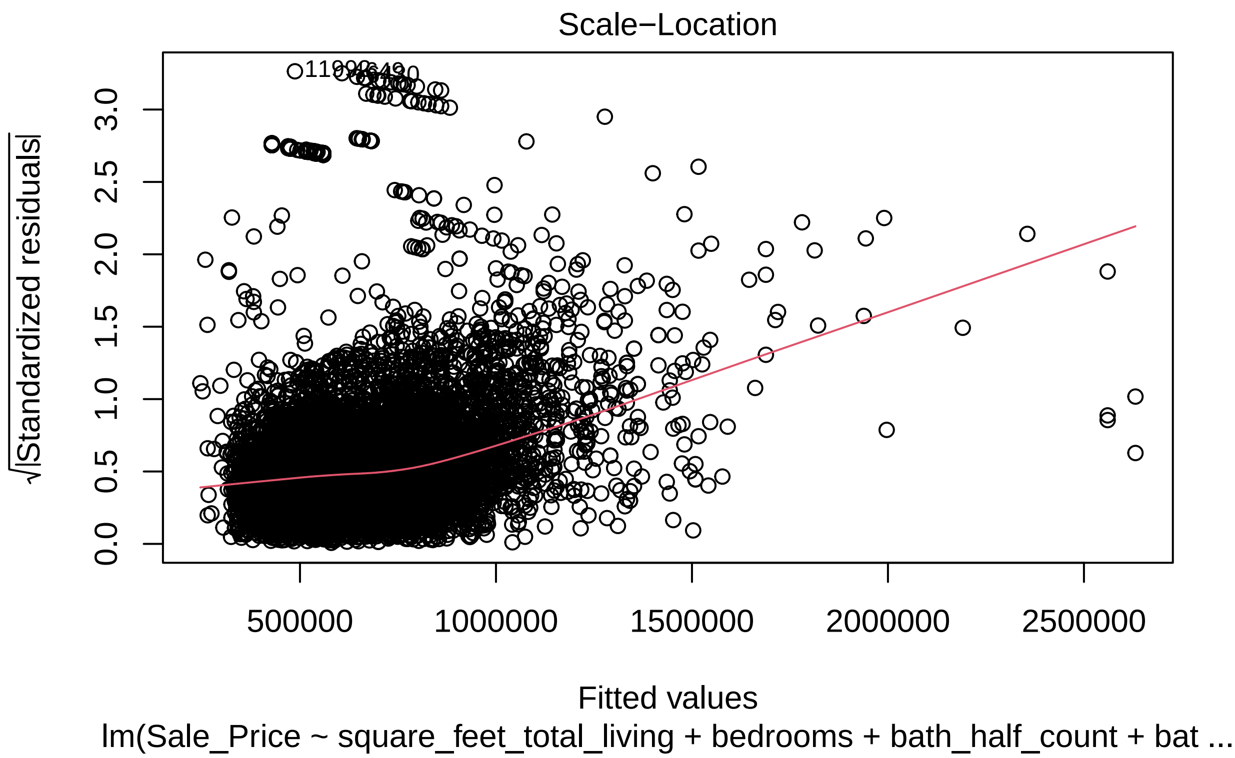
```
library(ggplot2)
```

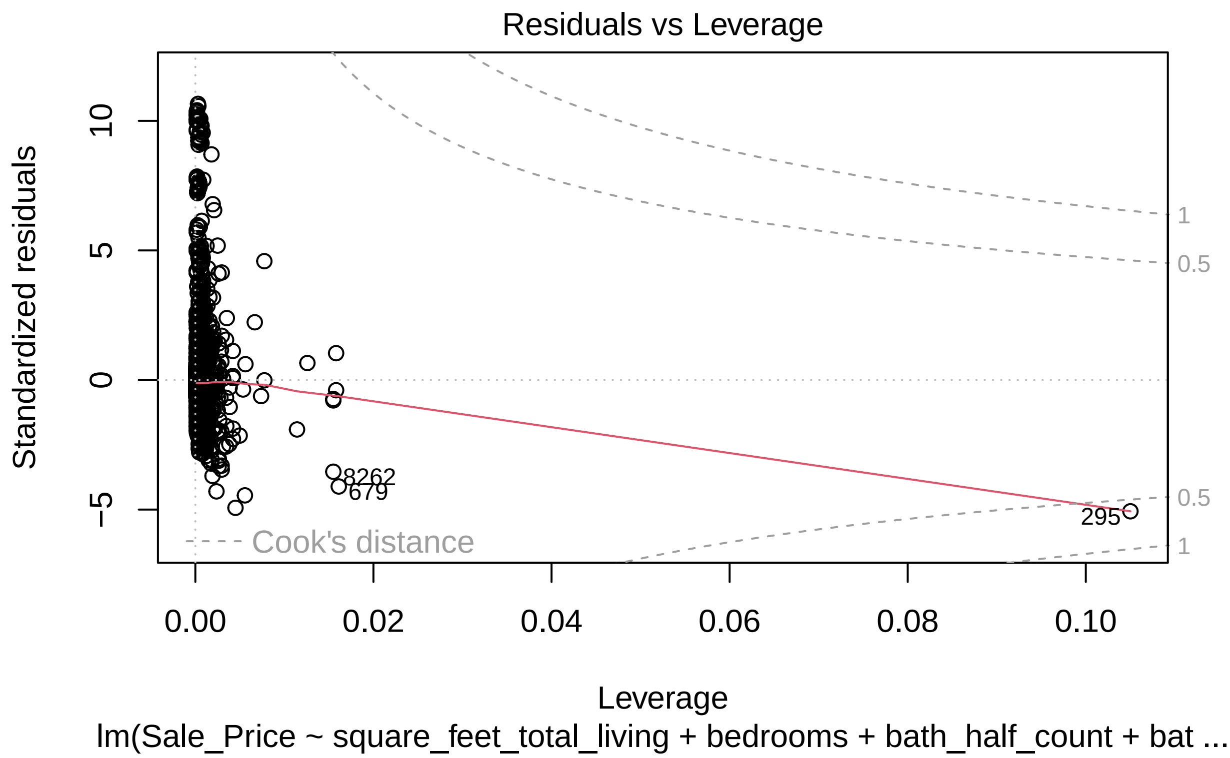
```
##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':
##
##     margin
```

```
plot(price_and_more)
```

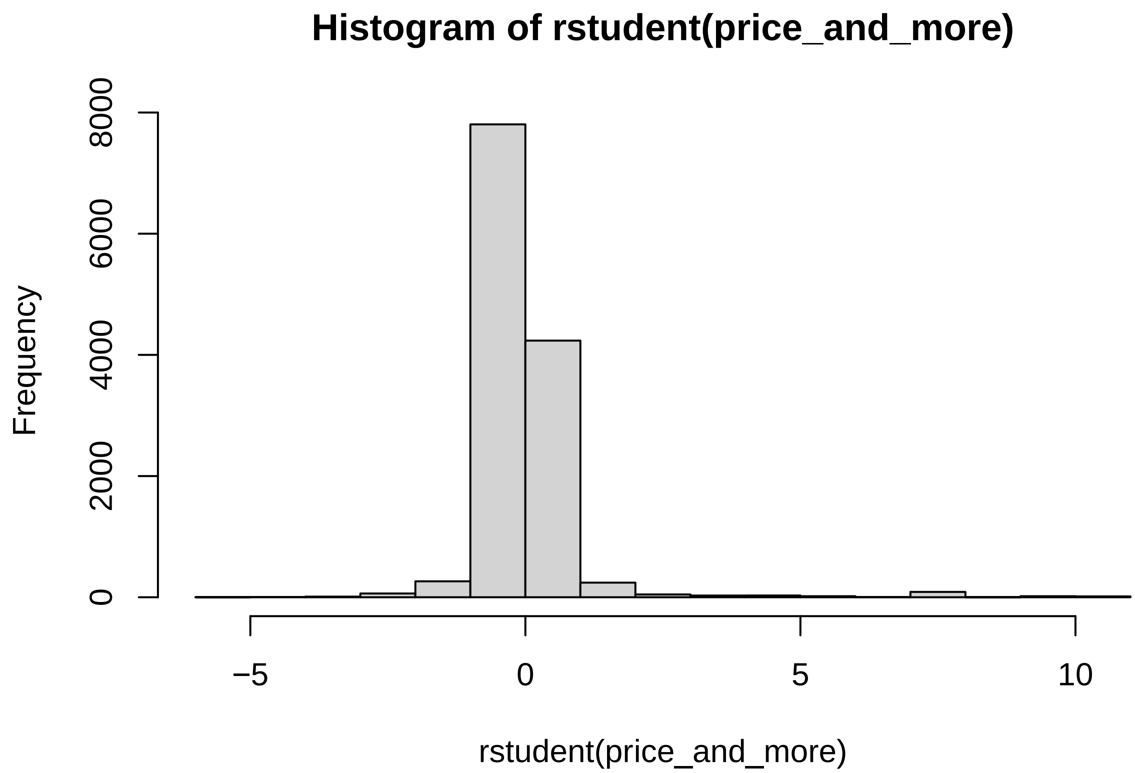

Residuals vs Fitted
Residuals
4e+06
2e+06
0e+00
−2e+06
500000
1000000
1500000
2000000
2500000
Fitted values
lm(Sale_Price ~ square_feet_total_living + bedrooms + bath_half_count + bat ...


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


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 ~ square_feet_total_living + bedrooms + bath_half_count + bat ...




```
hist(rstudent(price_and_more))
```



Histogram distributions is not normal. Q-Q plot does not looks like a diagonal line. The plot standardized residuals plotted against the fitted (predicted) values is not scattered, which means this is probably a violation of the assumption of homogeneity of variance

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

This regression is not entirely unbiased. Based on the vif avaerage, it is possible the model may be biased.