Assignment 3 - Week 8

2022-05-11

Set the working directory to the root of your DSC 520 directory

```
setwd("/Users/marianamacdonald/Documents/DATA SCIENCE/DSC 520/Statistics R/Week 2/dsc520")
library(readxl)
real.estate <- read_excel("data/week-7-housing.xlsx")
```

Complete the following: 1) Explain any transformations or modifications you made to the dataset I have added a . (dot) for variables that were separated by a space (Sale.Price for example)

2) 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.

```
spsqlot1 <- lm(Sale.Price ~ sq_ft_lot, data = real.estate)
spsqlot2 <- lm(Sale.Price ~ sq_ft_lot + bedrooms + bath_full_count, data = real.estate)
```

I chose bedrooms and bath as additional predictor, because I predict that sale price will be higher if the sq_ft_lot and the bed and bath counts are higher.

3) 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?

```
options(scipen=999)
summary(spsqlot1)
```

```
##
## Call:
## lm(formula = Sale.Price ~ sq_ft_lot, data = real.estate)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2016064  -194842   -63293    91565  3735109
##
## Coefficients:
##                 Estimate   Std. Error t value            Pr(>|t|)
## (Intercept) 641821.40609   3799.91526  168.90 <0.0000000000000002 ***
## sq_ft_lot        0.85099      0.06217   13.69 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 401500 on 12863 degrees of freedom
## Multiple R-squared:  0.01435,    Adjusted R-squared:  0.01428
## F-statistic: 187.3 on 1 and 12863 DF,  p-value: < 0.00000000000000022
```

```
summary(spsqlot2)
```

```
##
## Call:
## lm(formula = Sale.Price ~ sq_ft_lot + bedrooms + bath_full_count,
##     data = real.estate)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3566287  -153218   -51375    69545  3736381
##
## Coefficients:
##                    Estimate  Std. Error t value            Pr(>|t|)
## (Intercept)     142928.9011  14812.9592   9.649 <0.0000000000000002 ***
## sq_ft_lot            0.7227      0.0591  12.229 <0.0000000000000002 ***
## bedrooms         68865.1686   4027.4827  17.099 <0.0000000000000002 ***
## bath_full_count 145784.6656   5421.9053  26.888 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 381000 on 12861 degrees of freedom
## Multiple R-squared:  0.1127, Adjusted R-squared:  0.1125
## F-statistic: 544.3 on 3 and 12861 DF,  p-value: < 0.00000000000000022
```

R2 or the coefficient of determination explains how much the predictors explain the output. The adjusted R2 determines if other variables contribute to the model. Also, indicates the loss of predictive power (shrinkage).

When only sq_ft_lot is used as a predictor, the R2 is .014 or 1.4%. This means that sq_ft_lot only accounts for 1.4% of sales price. When bed and bath are included as predictors, this value increases to 0.1127 or 11.27% of variance in price. The adjusted R2 gives some idea of how well the model generalizes. The difference on the second model is 0.1127 - 0.1125 = 0.0002 (about 0.02%). This is good because if the model were derivedfrom the population rather than a sample, it would account for approximately 0.02% less variance in the outcome.

From summary 2, estimate, we can see: st_ft_lot increases by 1 unit, price increases by 0.72 / tvalue = 12.22 add one extra bedroom, price increases by 68K / tvalue = 17.09 extra bath, price increases by $145K / tvalue= 26.88 (this has more influence on sales price than the other 2 predictors)

4) 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?

```
sd(real.estate$Sale.Price)
```

```
## [1] 404381.1
```

```
sd(real.estate$sq_ft_lot)
```

```
## [1] 56933.29
```

```
sd(real.estate$bedrooms)
```

```
## [1] 0.8761273
```

```
sd(real.estate$bath_full_count)
```

```
## [1] 0.6507965
```

```
library(lm.beta)
lm.beta(spsqlot2)
```

```
##
## Call:
## lm(formula = Sale.Price ~ sq_ft_lot + bedrooms + bath_full_count,
##     data = real.estate)
##
## Standardized Coefficients::
##     (Intercept)       sq_ft_lot        bedrooms bath_full_count
##       0.0000000       0.1017484       0.1492025       0.2346207
```

sq_ft_lot (standardized B = .10): This value indicates that as sq_ft_lot increases one standard deviation (56,933), house sales price increases by .10 standard deviations. The standard deviation for sales price is 404,381) and so this constitutes a change of 40,438 in sales price (0.10 x 404381.1). Therefore, for every 56,933 sq_ft_lot, an extra 40,438 is added to sales price.

bedrooms (standardized B = .14): This value indicates that as bedroom number increases one standard deviation (0.87), house sales price increases by .14 standard deviations. The standard deviation for sales price is 404,381 and so this constitutes a change of 56,613 at sales price (0.14 x 404,381). Therefore, for every 0.87 extra bedroom, an extra 56,613 is added to sales price.

full bath count (standardized B = .23): This value indicates that as full bath count number increases one standard deviation (0.65), house sales price increases by .23 standard deviations. The standard deviation for sales price is 404,381 and so this constitutes a change of 93,007 at sales price (0.23 x 404,381). Therefore, for every 0.65 additional bath, an extra 93,007 is added to sales price.

adding a bathroom to a house, will increase the sales price more than having extra sq_ft_lot or bedrooms.

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

```
confint(spsqlot2)
```

```
##                        2.5 %         97.5 %
## (Intercept)     113893.30200 171964.5001988
## sq_ft_lot            0.60685      0.8385307
## bedrooms         60970.70465  76759.6325970
## bath_full_count 135156.92628 156412.4048972
```

only sq_ft_lot has tight confidence intervals (0.60 and 0.83), indicating that the estimates for the current model (sqsqlot1) are likely representative of the true population values.

6) 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.

```
anova(spsqlot1, spsqlot2)
```

```
## Analysis of Variance Table
##
## Model 1: Sale.Price ~ sq_ft_lot
## Model 2: Sale.Price ~ sq_ft_lot + bedrooms + bath_full_count
##   Res.Df              RSS Df       Sum of Sq      F               Pr(>F)
## 1  12863 2073376756946868
## 2  12861 1866579716716768  2 206797040230100 712.43 < 0.00000000000000022 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can say that model 2 (spsqlot2) significantly improved the fit of the model to the data compared to model 1 (spsqlot1), F = 712.43,p<.001

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

```
real.estate$residuals <- resid(spsqlot2)

real.estate$standardized.residuals<- rstandard(spsqlot2)
real.estate$studentized.residuals<-rstudent(spsqlot2)
real.estate$cooks.distance<- cooks.distance(spsqlot2)
real.estate$dfbeta<- dfbeta(spsqlot2)
real.estate$dffit<- dffits(spsqlot2)
real.estate$leverage<- hatvalues(spsqlot2)
real.estate$covariance.ratios<- covratio(spsqlot2)
real.estate
```

```
## # A tibble: 12,865 x 32
##    Sale.Date           Sale.Price sale_reason sale_instrument sale_warning
##    <dttm>                   <dbl>       <dbl>           <dbl> <chr>
##  1 2006-01-03 00:00:00     698000           1               3 <NA>
##  2 2006-01-03 00:00:00     649990           1               3 <NA>
##  3 2006-01-03 00:00:00     572500           1               3 <NA>
##  4 2006-01-03 00:00:00     420000           1               3 <NA>
##  5 2006-01-03 00:00:00     369900           1               3 15
##  6 2006-01-03 00:00:00     184667           1              15 18 51
##  7 2006-01-04 00:00:00    1050000           1               3 <NA>
##  8 2006-01-04 00:00:00     875000           1               3 <NA>
##  9 2006-01-04 00:00:00     660000           1               3 <NA>
## 10 2006-01-04 00:00:00     650000           1               3 <NA>
## # ... with 12,855 more rows, and 27 more variables: sitetype <chr>,
## #   addr_full <chr>, zip5 <dbl>, 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>, standardized.residuals <dbl>, ...
```

```
real.estate[, c('Sale.Price', 'sq_ft_lot', 'bedrooms', "bath_full_count", "residuals",
                "standardized.residuals", "studentized.residuals", "cooks.distance",
                "dfbeta")]
```

```
## # A tibble: 12,865 x 9
##    Sale.Price sq_ft_lot bedrooms bath_full_count residuals standardized.residua~
##         <dbl>     <dbl>    <dbl>           <dbl>     <dbl>                 <dbl>
##  1     698000      6635        4               2   -16754.              -0.0440
##  2     649990      5570        4               2   -63994.              -0.168
##  3     572500      8444        4               1     2223.               0.00584
##  4     420000      9600        3               1   -82247.              -0.216
##  5     369900      7526        3               1  -130848.              -0.343
##  6     184667      7280        4               2  -530553.              -1.39
##  7    1050000     97574        5               3    54875.               0.144
##  8     875000     30649        4               2   142891.               0.375
##  9     660000     42688        4               2   -80809.              -0.212
## 10     650000     94889        4               1    17250.               0.0453
## # ... with 12,855 more rows, and 3 more variables: studentized.residuals <dbl>,
## #   cooks.distance <dbl>, dfbeta <dbl[,4]>
```

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

```
real.estate$large.residual <- real.estate$standardized.residuals>2 |
  real.estate$standardized.residuals < -2
```

9) Use the appropriate function to show the sum of large residuals.

```
sum(real.estate$large.residual)
```

```
## [1] 329
```

The sample has 12,865 rows, and 329 cases are TRUE, meaning that they have a residual more than 2 or less than -2.

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

```
real.estate[real.estate$large.residual, c("Sale.Price", "sq_ft_lot", "bedrooms",
                                          "bath_full_count","standardized.residuals")]
```

```
## # A tibble: 329 x 5
##    Sale.Price sq_ft_lot bedrooms bath_full_count standardized.residuals
##         <dbl>     <dbl>    <dbl>           <dbl>                  <dbl>
##  1     265000    112650        4               4                  -2.15
##  2    1900000     37017        4               3                   2.67
##  3    1390000    225640        0               1                   2.47
##  4    1588359      8752        2               2                   2.65
##  5    1450000     14043        3               2                   2.10
##  6    1450000     14043        2               1                   2.66
##  7     270000     89734        4              23                  -9.81
```

```
##  8      90000    574992        3               1                  -2.16
##  9      90000    574992        3               1                  -2.16
## 10    2500000     36362        4               2                   4.63
## # ... with 319 more rows
```

We have 329 out of 12,865 or 2.55%

11) Investigate further by calculating the leverage, cooks distance, and covariance ratios. Comment on all cases that are problematic.

```
real.estate[real.estate$large.residual, c("cooks.distance", "leverage", "covariance.ratios")]
```

```
## # A tibble: 329 x 3
##    cooks.distance leverage covariance.ratios
##             <dbl>    <dbl>             <dbl>
##  1       0.00132  0.00115              1.00
##  2       0.000616 0.000345             0.998
##  3       0.00367  0.00241              1.00
##  4       0.000633 0.000360             0.998
##  5       0.000134 0.000122             0.999
##  6       0.000607 0.000343             0.998
##  7       2.38     0.0900               1.07
##  8       0.00901  0.00765              1.01
##  9       0.00901  0.00765              1.01
## 10       0.000591 0.000110             0.994
## # ... with 319 more rows
```

cooks distance less than 1. If higher than 1, they have an undue influence on the model. leverage : between zero and 1. covariance ratio: between 0.94 and 1.06

from the first 10 cases, I can see that like 10 is problematic because it's out of the range for all 3 measures.

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

```
library(car)
```

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

```
durbinWatsonTest(spsqlot2)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.6487202     0.7025582       0
##  Alternative hypothesis: rho != 0
```

The condition has not been met because DW is 0.70 (less than 1) and p-value = 0. There is no autocorrelation.

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

```
library(car)
library(carData)
vif(spsqlot2)
```

```
##       sq_ft_lot        bedrooms bath_full_count
##        1.003410        1.103585        1.103567
```

```
1/vif(spsqlot2)
```

```
##       sq_ft_lot        bedrooms bath_full_count
##       0.9966015       0.9061373       0.9061528
```

```
mean(vif(spsqlot2))
```

```
## [1] 1.070187
```

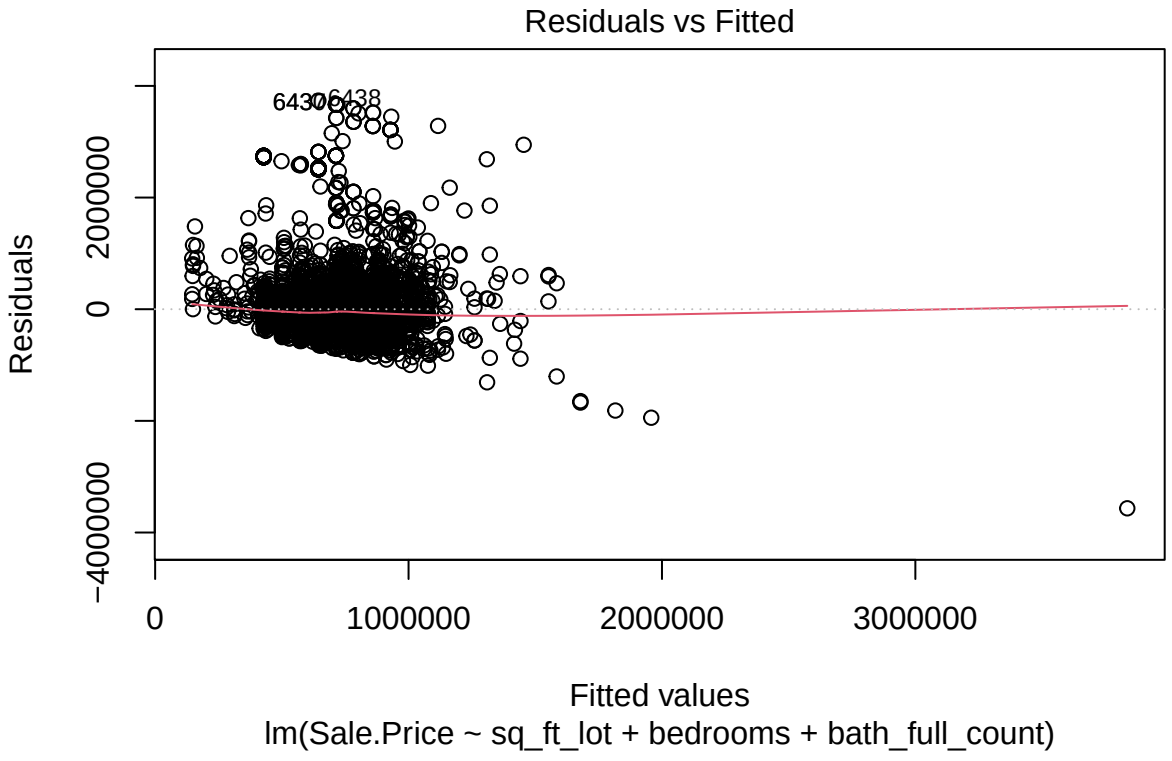
```
vif(spsqlot2)
```

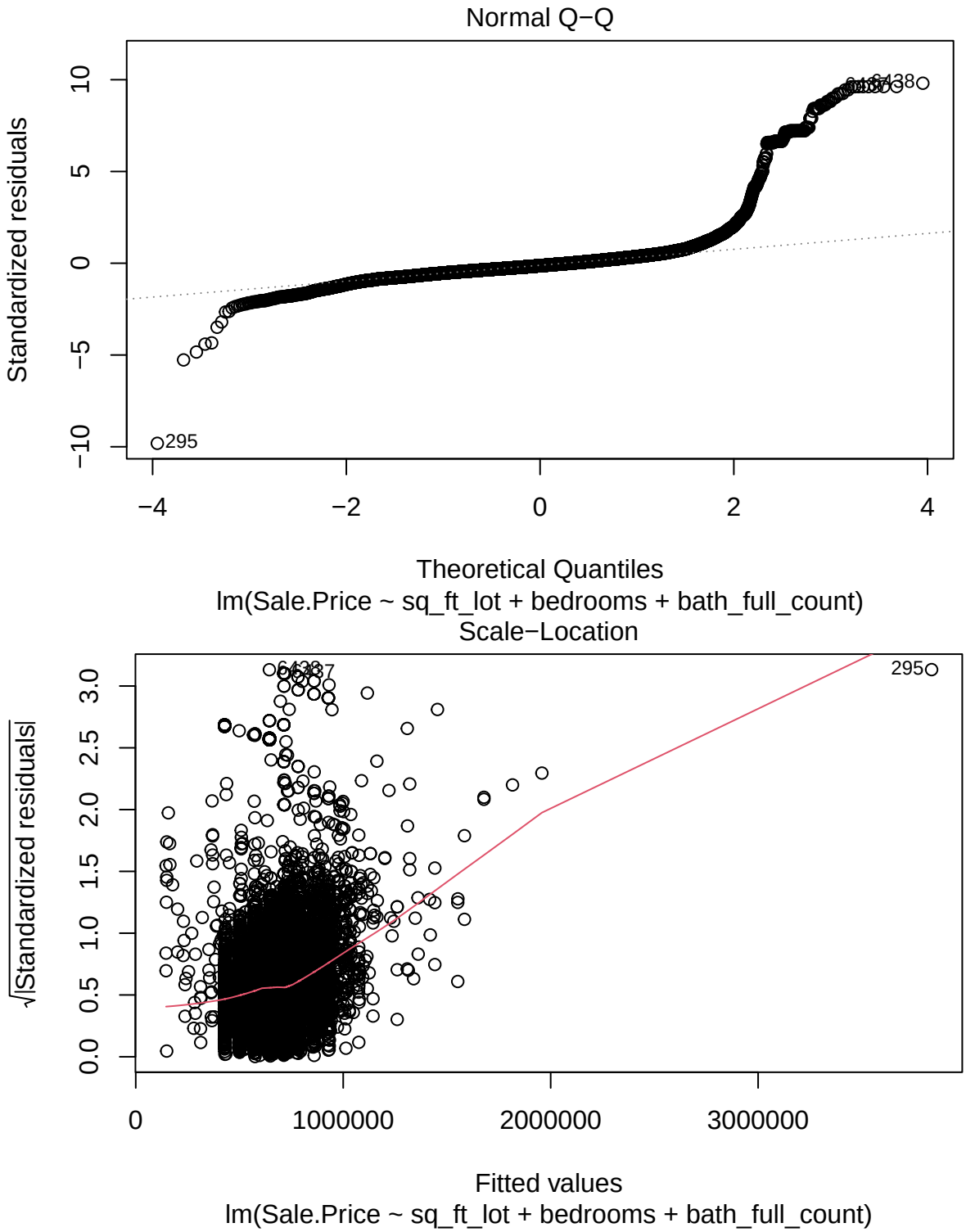
```
##       sq_ft_lot        bedrooms bath_full_count
##        1.003410        1.103585        1.103567
```

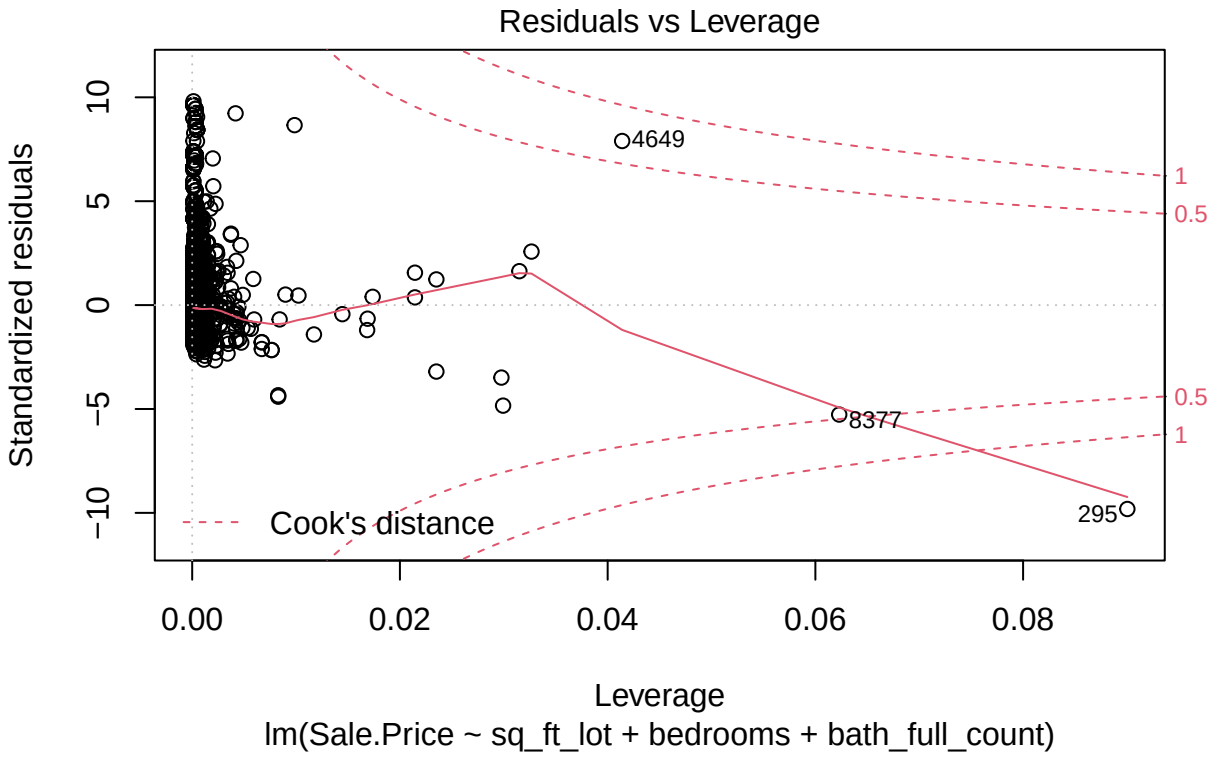
There is no collinearity within the data since VIF values are below 10 and tolerance statistics are above 0.2).

14) 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.

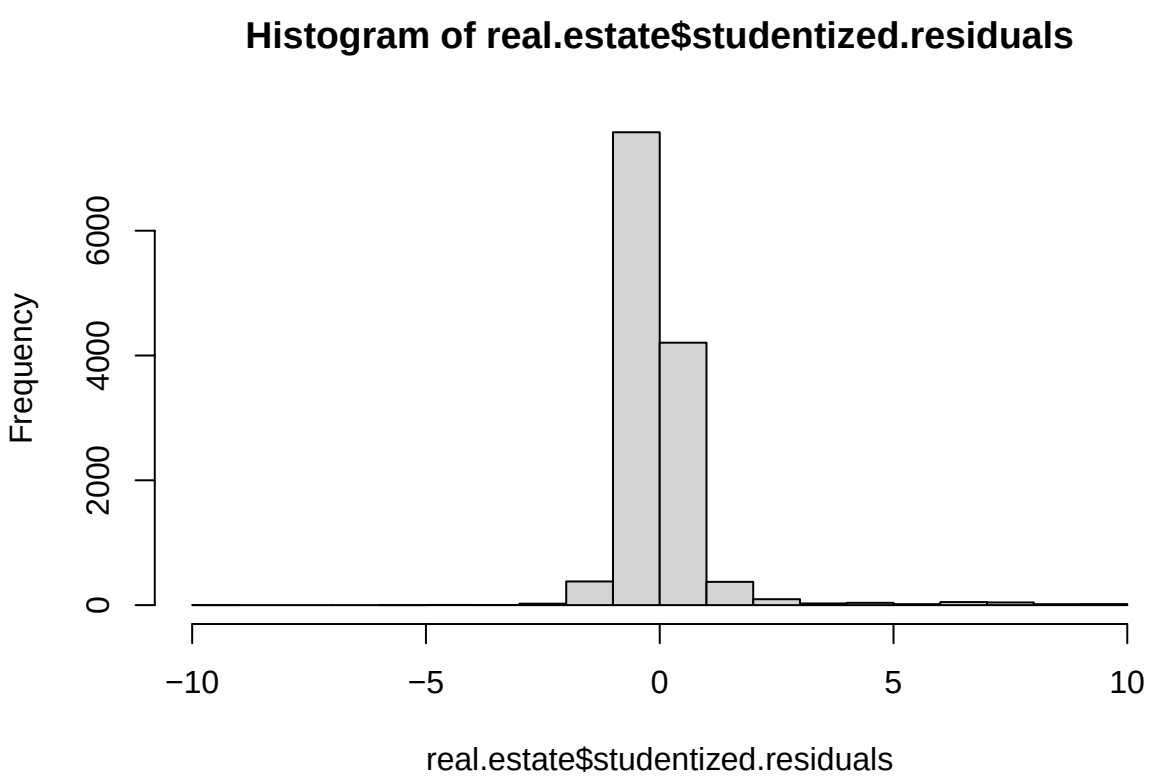
```
plot(spsqlot2)
```




Normal Q−Q
Standardized residuals
10
5
0
−5
−10
295
−4
−2
0
2
4
Theoretical Quantiles
lm(Sale.Price ~ sq_ft_lot + bedrooms + bath_full_count)
Scale−Location
√|Standardized residuals|
3.0
2.5
2.0
1.5
1.0
0.5
0.0
295
0
1000000
2000000
3000000
Fitted values
lm(Sale.Price ~ sq_ft_lot + bedrooms + bath_full_count)




```
hist(real.estate$studentized.residuals)
```



The first plot (residuals vs Fitted) shows a random pattern, which indicates that the assumptions of linearity, randomness and homoscedasticity have been met.The second plot (Normal Q-Q), shows a deviation from normality, skewed.The hist looks like a normal distribution.

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

Yes, it is unbiased. It generalizes well. The model can be applied to the entire population.