Week8_Housing_Data

Akila Selvaraj

2/12/2022

i) Transformations or Modifications made on the dataset.

Some of the sale price in housing dataset are too low or too high. To avoid uncertainity from the model, I excluded these outliers and included only the rows whose sale price is falling between normal price range of 10000 and 10000000

```
Housing_df <- dplyr::filter(Housing_df, Sale.Price<=10000000 &
                              Sale.Price>=10000 & square_feet_total_living > 700)

Housing_df$Saleyear <- as.integer(format(Housing_df$Sale.Date, "%Y"))
```

ii) Creating two variables

square feet as a predictor - simple regression

```
SalePrice_sq_lm <- lm(Sale.Price ~ square_feet_total_living,
                      data=Housing_df,na.action = na.exclude)
```

Multiple R-squared tells us if Sales Price can be explained by square_feet_total_living. In general, the larger the R-squared value of a regression model, the better the explanatory variables are able to predict the value of the response variable. In this case, 20.6% of the variation in salesPrice can be explained square_feet_total_living.

Multiple regression

There are many attributes in the the dataset which are related to sale price of the House. Not all the attributes play a significant role in determining the house price. From the data, we will select only few attributes which may contribute to sales price. House price varies time to time. So,sale year plays a major role in deciding the price. Attributes related to house like square feet of total living, bath count also predicts the sale price. So, I have added those as well. I have included zipcode as house price varies depends on the location.

```
SalePrice_lm <- lm(Sale.Price ~ square_feet_total_living + Saleyear +
                      building_grade + bath_full_count  +  year_built , data=Housing_df)
```

iii) Summary function on two variables

Summary of simple regression

```
summary(SalePrice_sq_lm)
```

```
##
## Call:
## lm(formula = Sale.Price ~ square_feet_total_living, data = Housing_df,
##     na.action = na.exclude)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1816830  -119854   -41490    43431  3813406
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)              1.831e+05  8.756e+03   20.91   <2e-16 ***
## square_feet_total_living 1.883e+02  3.211e+00   58.64   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 358500 on 12819 degrees of freedom
## Multiple R-squared:  0.2115, Adjusted R-squared:  0.2115
## F-statistic:  3439 on 1 and 12819 DF,  p-value: < 2.2e-16
```

Summary of multiple regression

```
summary(SalePrice_lm)
```

```
##
## Call:
## lm(formula = Sale.Price ~ square_feet_total_living + Saleyear +
##     building_grade + bath_full_count + year_built, data = Housing_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1684519  -118540   -39941    41357  3981570
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)              -1.599e+07  1.979e+06  -8.077 7.22e-16 ***
## square_feet_total_living  1.433e+02  4.991e+00  28.712  < 2e-16 ***
## Saleyear                  5.721e+03  9.602e+02   5.958 2.63e-09 ***
## building_grade            3.364e+04  4.436e+03   7.583 3.60e-14 ***
## bath_full_count           1.562e+04  6.049e+03   2.583  0.00981 **
## year_built                2.244e+03  2.088e+02  10.746  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 354400 on 12815 degrees of freedom
## Multiple R-squared:  0.2294, Adjusted R-squared:  0.2291
## F-statistic:   763 on 5 and 12815 DF,  p-value: < 2.2e-16
```

Adjusted R square increased from 0.2115 to 0.2291 after adding multiple variables.

iv)Standardized betas for each parameter

A standardized beta coefficient compares the strength of the effect of each individual independent variable to the dependent variable. The higher the absolute value of the beta coefficient, the stronger the effect. Below

is the beta coefficient of each parameter.

```
lm.beta(SalePrice_lm)
```

```
## square_feet_total_living                 Saleyear           building_grade
##               0.35000234               0.04627018               0.09011235
##          bath_full_count               year_built
##               0.02506534               0.09509817
```

Standard beta values indicate that next to square_feet_total_living, variables year_built and building grade have comparable degree of importance in the model.

This says that with every increase of one standard deviation in square_feet_total_living, sale price rises by 0.35 standard deviations with the assumption that other variables are held constant. Similarly, Sale price increases by 0.0950 standard deviation with every increase of one standard deviation in year built. Importance of variables 'bath_full_count' are low in comparable to square_feet_of_living.

v)Confidence intervals

```
confint(SalePrice_lm)
```

```
##                                  2.5 %        97.5 %
## (Intercept)              -1.986442e+07 -1.210591e+07
## square_feet_total_living  1.335245e+02  1.530914e+02
## Saleyear                  3.838376e+03  7.602713e+03
## building_grade            2.494424e+04  4.233482e+04
## bath_full_count           3.765773e+03  2.748020e+04
## year_built                1.834808e+03  2.653534e+03
```

The individual contribution of variables to the regression model can be determined using confint function. The sign tells us about the direction of the relationship between the predictor and the outcome. Therefore, we would expect a very bad model to have confidence intervals that cross zero, indicating that in some samples the predictor has a negative relationship to the outcome whereas in others it has a positive relationship. Square feet total living and year built are likely to be the representative of true population.

vi)Assessing the improvement of the new model compared to the original model

Comparing models using R anova() function. Analysis of Variance Table

```
anova(SalePrice_sq_lm, SalePrice_lm)
```

```
## Analysis of Variance Table
##
## Model 1: Sale.Price ~ square_feet_total_living
## Model 2: Sale.Price ~ square_feet_total_living + Saleyear + building_grade +
##     bath_full_count + year_built
##   Res.Df        RSS Df  Sum of Sq      F    Pr(>F)
## 1  12819 1.6471e+15
## 2  12815 1.6097e+15  4 3.7351e+13 74.337 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Here, we can note that the value of F is 74.337, which is the same as the value that we calculated. The value in column labelled Pr(>F) is 2.2e^16 (i.e., 2.2 with the decimal place moved 16 places to the left, or a very

small value indeed); we can say that SalePrice_lm significantly improved the fit of the model to the data compared to SalePrice_sq_lm, F(6, 12857)(F(Df, Res.Df)) = 74.337, p < .001.

vii) Casewise diagnostics to identify outliers and influential cases

Outliers: Residuals can be obtained with the resid() function, standardized residuals with the rstandard() function and studentized residuals with the rstudent() function.

Influential cases: Cook's distances can be obtained with the cooks.distance() function, DFBeta with the dfbeta() function, DFFit with the dffits() function, hat values (leverage) with the hatvalues() function, and the covariance ratio with the covratio() function.

```
Housing_df$residuals<-resid(SalePrice_lm)
Housing_df$standardized.residuals<- rstandard(SalePrice_lm)
Housing_df$studentized.residuals<-rstudent(SalePrice_lm)
Housing_df$cooks.distance<-cooks.distance(SalePrice_lm)
Housing_df$dfbeta<-dfbeta(SalePrice_lm)
Housing_df$dffit<-dffits(SalePrice_lm)
Housing_df$leverage<-hatvalues(SalePrice_lm)
Housing_df$covariance.ratios<-covratio(SalePrice_lm)
```

viii)Standardized residuals calculation

```
Housing_df$large.residual <-
  (Housing_df$standardized.residuals > 2 |
     Housing_df$standardized.residuals < -2 )
```

ix)Sum of large residuals.

```
  sum(Housing_df$large.residual)
```

```
## [1] 317
```

This will tell us that only 317 cases had a large residual.

x)Variables with large residuals

It might be better to know not just how many cases there are, but also which cases they are. We can do this by selecting only those cases for which the variable large.residual is TRUE.

We will select parts of the data set which has large residuals

```
Housing_df[Housing_df$large.residual,c("Saleyear", "Sale.Price",
```

```
##     Saleyear Sale.Price square_feet_total_living year_built
## 25      2006     265000                     4920       2007
## 177     2006     390000                     5800       2008
## 238     2006    1588359                     3360       2005
## 244     2006    1450000                     3480       1972
## 245     2006    1450000                      900       1918
## 286     2006     163000                     4710       2014
##     standardized.residuals
## 25               -2.352573
```

```
## 177              -2.457276
## 238               2.209731
## 244               2.075006
## 245               3.696884
## 286              -2.416387
```

In above output, we will see the columns only we selected but only for cases for which large.residual is TRUE.

From this output, we can see that we have 317(1.02%) cases that are outside of the limits. Therefore, our sample is within 1% of what we would expect. In addition, 99% of cases lie within ±2.5 and so we would expect only 1% of cases to lie outside of these limits.

xi) Calculating the leverage, cooks distance, and covariance rations

We have saved a range of other casewise diagnostics from our model. One useful strategy is to use the casewise diagnostics to identify cases that we want to investigate further. Let's continue to look at the diagnostics for the cases of interest. Let's look now at the leverage (hat value), Cook's distance and covariance ratio for these 312 cases that have large residuals. We can do this by using the same command as before, but listing different variables (columns) in the data set:

```
Housing_df[Housing_df$large.residual,
           c("cooks.distance","leverage","covariance.ratios")] %>% head()
```

```
##     cooks.distance     leverage covariance.ratios
## 25    0.0011940575 0.0012927911         0.9991703
## 177   0.0015113033 0.0014994887         0.9991414
## 238   0.0002838021 0.0003486074         0.9985313
## 244   0.0004807681 0.0006695102         0.9991221
## 245   0.0043500350 0.0019060913         0.9959819
## 286   0.0009712081 0.0009970048         0.9987321
```

Above are the cases for which large.residual is TRUE. These cases don't have a Cook's distance greater than 1 , so these cases are having an undue influence on the model.

k=5, N = 12787 (in which N is the number of cases or participants, and k is the number of predictors in the model)

The average leverage can be calculated as 0.0005 (k + 1/n = 4/200) and so we are looking for values either twice as large as this (0.0010) or three times as large (0.0015) depending on which statistician you trust most. In our results, almost all cases are within the boundary of three times the average and only few cases are close to two times the average.

There is also a column for the covariance ratio.

we need to use the following criteria:

CVRi > 1 + [3(k + 1)/n] = 1 + [3(6 + 1)/12787] = 1.001; CVRi < 1 - [3(k + 1)/n] = 1 - [3(6 + 1)/12787] = 0.9983.

Therefore, we are looking for any cases that deviate substantially from these boundaries. Most of our 312 potential outliers have CVR values within or just little outside these boundaries. However, from this minimal set of diagnostics we appear to have a fairly reliable model that has not been unduly influenced by any subset of cases.

xii) Assess the assumption of independence - Durbin-Watson test

One of the key assumptions in linear regression is that there is no correlation between the residuals, e.g. the residuals are independent. we can test the assumption of independent errors using the Durbin-Watson

test. We can obtain this statistic along with a measure of autocorrelation and a p-value in R using the durbinWatsonTest(). To see the Durbin-Watson test for our SalePrice_sq_lm model, we would execute:

```
dwtest(SalePrice_lm)
```

```
##
##  Durbin-Watson test
##
## data:  SalePrice_lm
## DW = 0.53406, p-value < 2.2e-16
## alternative hypothesis: true autocorrelation is greater than 0
```

```
durbinWatsonTest(SalePrice_lm)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.7329679     0.5340587       0
##  Alternative hypothesis: rho != 0
```

As a conservative rule we can say that values less than 1 or greater than 3 should definitely raise alarm bells. The closer to 2 that the value is, the better, and for these data the value is 0.53348, which is not close to 2 that the assumption has almost certainly not been met. The p-value of .7 confirms this conclusion (it is very much bigger than .05 and, therefore, not remotely significant).

From the output we can see that the test statistic is 0.5334828 which is not closer to 2 and it is lesser than 1 which says that it is positively correlated The corresponding p-value is 0. Since this p-value is less than 0.05, we can reject the null hypothesis and conclude that the residuals in this regression model are autocorrelated. This would give us enough evidence to state that our independence assumption is not met.

xiii)Assumption of no multicollinearity and state if the condition is met or not.

The VIF and tolerance statistics (with tolerance being 1 divided by the VIF) are useful statistics to assess collinearity. We can obtain the VIF using the vif() function. To get the VIF statistics for the SalePrice_sq_lm model, we execute:

The tolerance doesn't have its own function, but we can calculate it very easily, if we remember that tolerance = 1/VIF. Therefore, we can get the values by executing:

Tolerance is 0.770601

It can be useful to look at the average VIF too. To calculate the average VIF,we can add the VIF values for each predictor and divide by the number of predictors (k).

Average VIF is 1.2976885

VIF of all variables is:

```
car::vif(SalePrice_lm)
```

```
## square_feet_total_living                 Saleyear           building_grade
##                 2.471130                 1.003126                 2.348268
##          bath_full_count               year_built
##                 1.566372                 1.302450
```

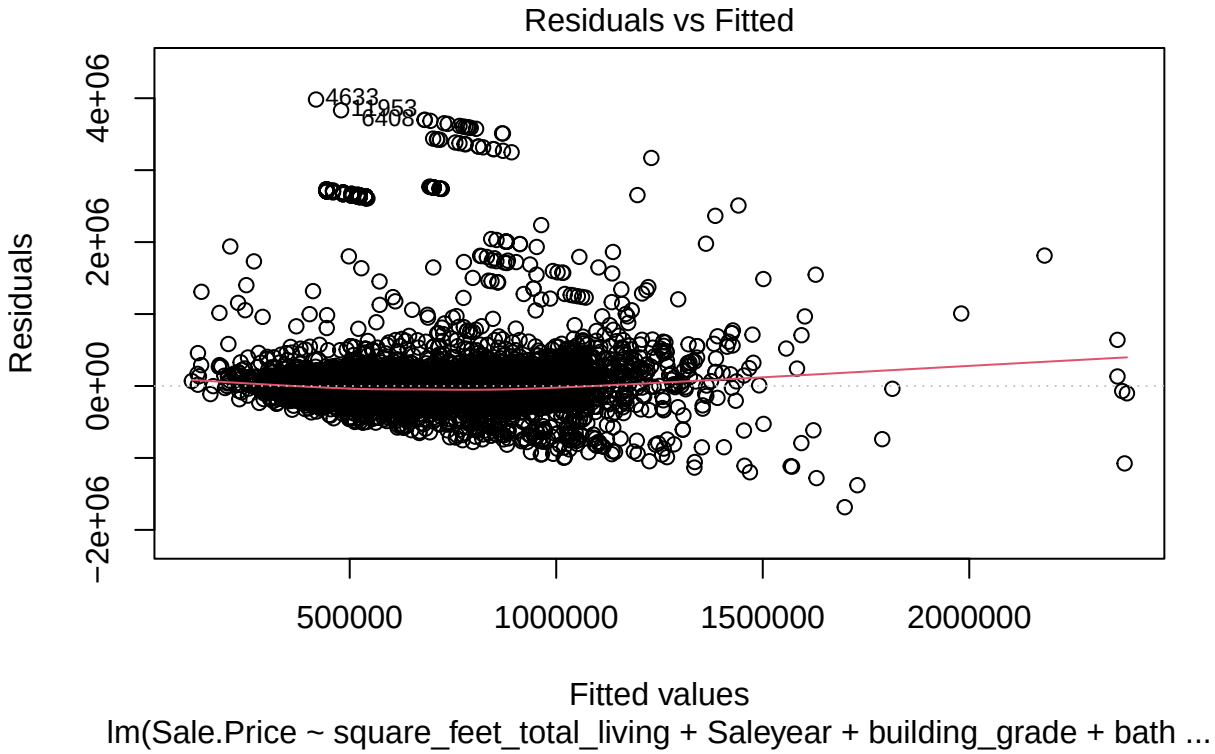
If the largest VIF is greater than 10 then there is cause for concern. If the average VIF is substantially greater than 1 then the regression may be biased Tolerance below 0.1 indicates a serious problem. Tolerance below 0.2 indicates a potential problem

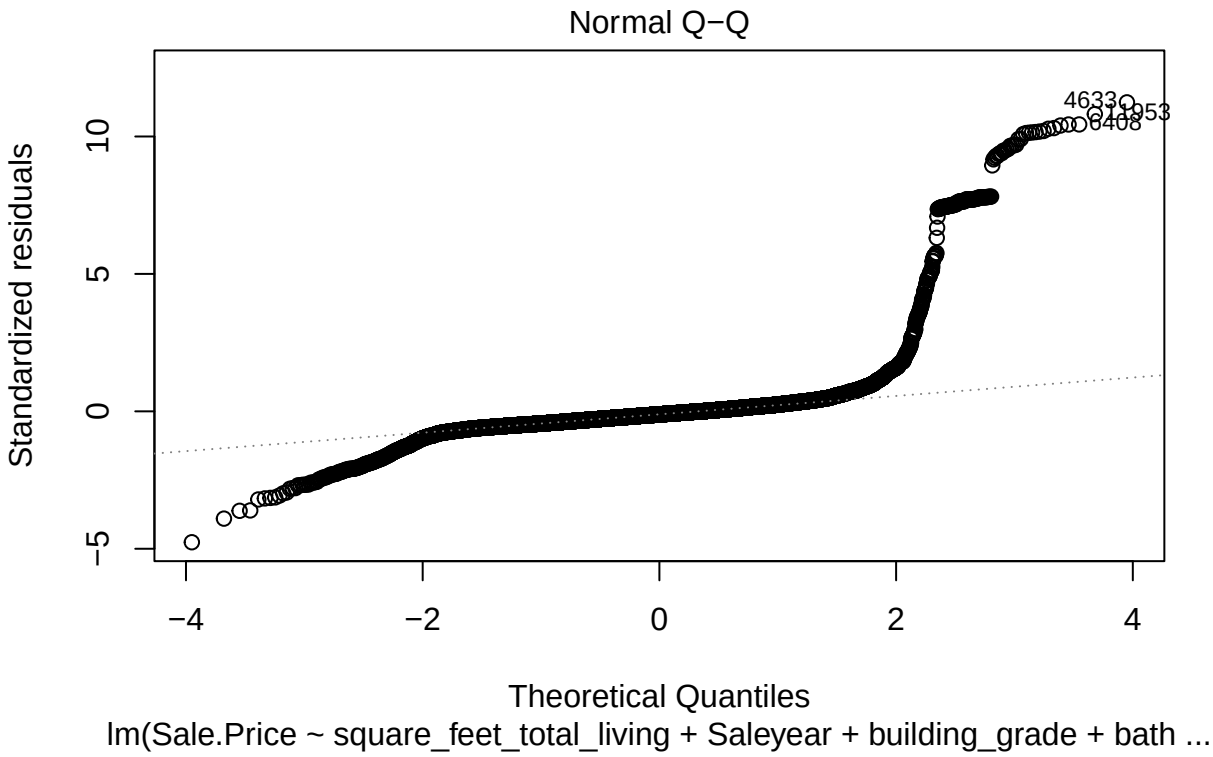
For our current model the VIF values are all well below 10 and the tolerance statistics all well above 0.2. Also, the average VIF is very close to 1. Based on these measures we can safely conclude that there is no

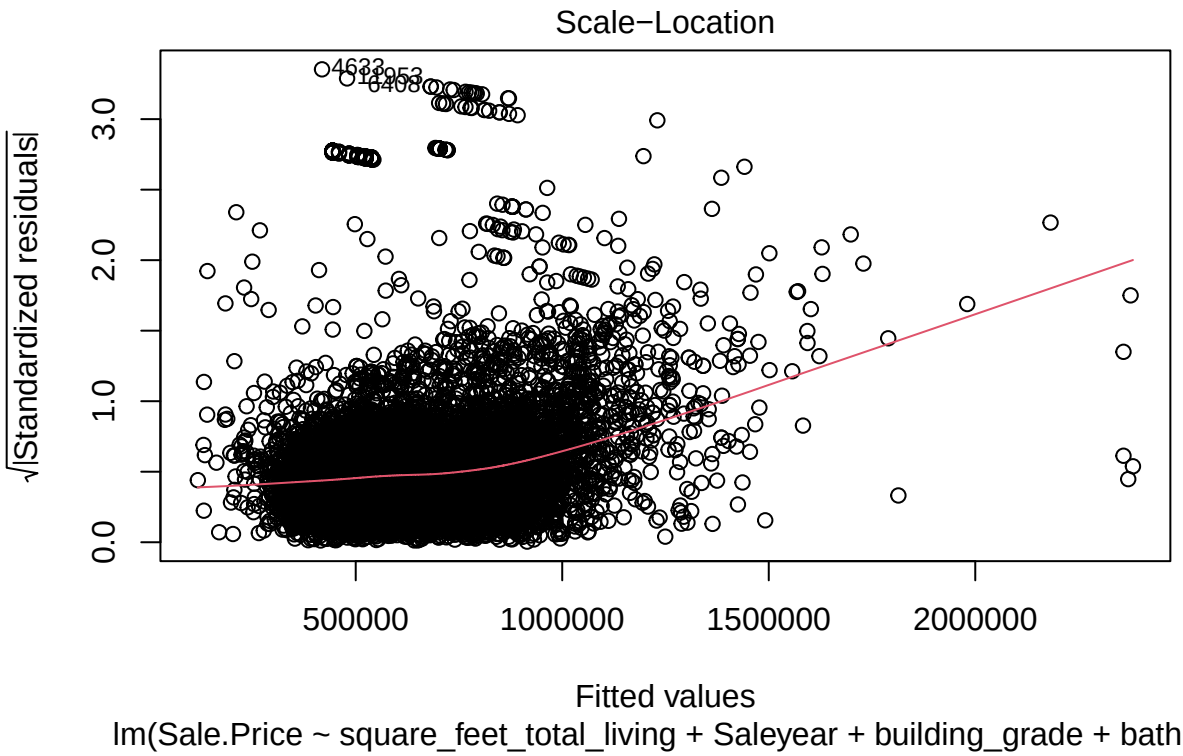
collinearity within our data.

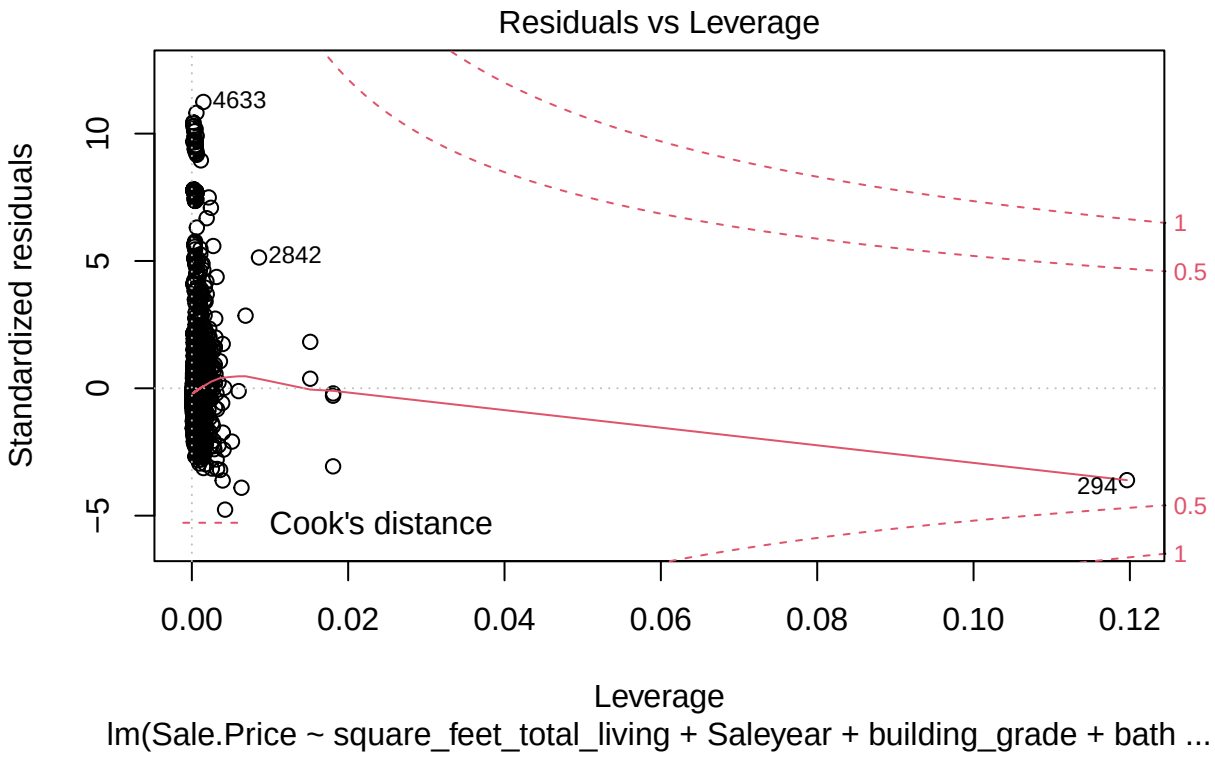
xiv) Plot() and hist() functions

```
plot(SalePrice_lm)
```




Normal Q−Q
Standardized residuals
10
5
0
−5
−4
−2
0
2
4
Theoretical Quantiles
lm(Sale.Price ~ square_feet_total_living + Saleyear + building_grade + bath ...


Scale−Location
4633
4953
4403
√|Standardized residuals|
0.0
1.0
2.0
3.0
500000
1000000
1500000
2000000
Fitted values
lm(Sale.Price ~ square_feet_total_living + Saleyear + building_grade + bath ...



Residual vs Fitted

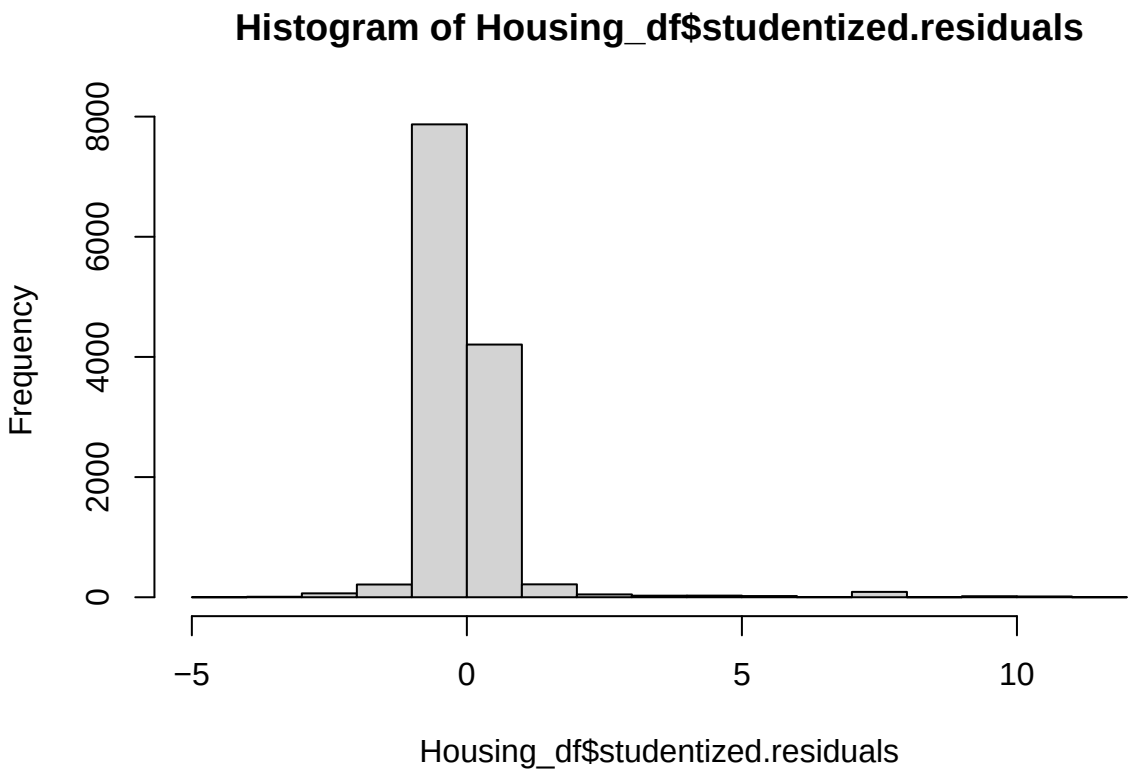
Plot of fitted values against residuals look like a random array of dots dispersed around zero. Hopefully, it's clear to us that the graph for the residuals in our sales price model shows a fairly random pattern, which is indicative of a situation in which the assumptions of linearity, randomness and homoscedasticity have been met.

Q-Q Plot

The second plot that is produced by the plot() function is a Q-Q plot, which shows up deviations from normality. The straight line in this plot represents a normal distribution, and the points represent the observed residuals. Therefore, in a perfectly normally distributed data set, all points will lie on the line. This is not pretty much what we see for the sales price data. In this plot, few dots are very distant from the line (at the extremes), which indicates a deviation from normality (in this particular case skew).

hist plot

Another useful way to check whether the residuals deviate from a normal distribution is to inspect the histogram of the residuals (or the standardized or studentized residuals). We can obtain this plot easily using the hist() function. We saved the studentized residuals in our dataframe earlier so we could enter this variable into the function and execute it:



We can generate the same plot by entering the rstudent() function that into hist(): hist(rstudent(SalePrice_lm)) Figure shows the histogram of the data. The histogram should look like a normal distribution (a bell-shaped curve). For the housing data, the distribution looks normal but at the same time, it is slightly deviated. it should be clear that the distribution is slightly skewed and not roughly normal. The greater the deviation, the more non-normally distributed the residuals. For both the histogram and normal Q-Q plots, the non-normal examples are extreme cases and you should be aware that the deviations from normality are likely to be subtler. We could summarize by saying that the model appears, in most senses, to be both accurate for the sample and generalization to the population. Therefore, we could conclude that in our sample, building_grade and zip are fairly equally important in predicting sales price.

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?

Some of the assumptions seem to have met. So, we can probably assume that this model would generalize Housing price.At the same time, for some of the assumptions doesn't meet as expected. We can't say clearly if the regression model is completely unbiased. There are chances that sample we took may not give the same result for the entire population.