Week8Assignment_regression_RiazAhmedTamimAnsari.r

Riaz

2023-10-28

```
# title: "Week_8_Excercise_Riaz"
# author: "Riaz Ahmed Tamim Ansari"
# date: "2023-10-29"
library(readxl)

#Load data using read_excel function from readxl package.
df <-
  read_excel("C:\\Users\\Riaz\\Desktop\\MSDS\\Introduction to Statistics\\Week8&9\\week-6-housing.xlsx")
df
```

```
## # A tibble: 12,865 x 24
##    `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>
## # i 12,855 more rows
## # i 19 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>
```

```
colnames(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"
```

```
#1. Explain any transformations or modifications you made to the dataset.

#I have visually inspected the dataset, did a count of rows and understood the column details.
#Used the function is.na() to check if there are any NA values in the variables that I have chosen.
#Found out 4 duplicated rows and dropped it off by using unique()
#During the course of the excercise, I have also added new columns to the existing dataframe,
#from the outputs of the model and other functions used.

#Quickly inspect the read dataframe and get the total count of rows
head(df)
```

```
## # A tibble: 6 x 24
##   `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
## # i 19 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>
```

```
nrow((df))
```

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

```
#Load tidyverse for dplyr.
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
nrow(unique(df))
```

```
## [1] 12861
```

```r
#There are 4 duplicated rows, which we are dropping off.  So the total number of rows would be 12861
df <- unique(df)
nrow(df)
```

```
## [1] 12861
```

```r
#Check if the Sale Price and sq_ft_lot are having NA values.
is.na(df$`Sale Price`) %>% sum()
```

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

```r
is.na(df$sq_ft_lot) %>% sum()
```

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

```r
#From the above output, there are no NA values.
#Next we start to build the model

#Model1:
#=======

#2. Create a linear regression model where "sq_ft_lot" predicts Sale Price.

model <- lm(`Sale Price` ~ sq_ft_lot, data = df, header = TRUE)
```

```
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'header' will be disregarded
```

```r
#Getting the summary of the first model
summary(model)
```

```
##
## Call:
## lm(formula = `Sale Price` ~ sq_ft_lot, data = df, header = TRUE)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2015913  -194880   -63332    91528  3735071
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.419e+05  3.801e+03  168.87   <2e-16 ***
## sq_ft_lot   8.509e-01  6.218e-02   13.68   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 401500 on 12859 degrees of freedom
## Multiple R-squared:  0.01435,    Adjusted R-squared:  0.01428
## F-statistic: 187.2 on 1 and 12859 DF,  p-value: < 2.2e-16
```

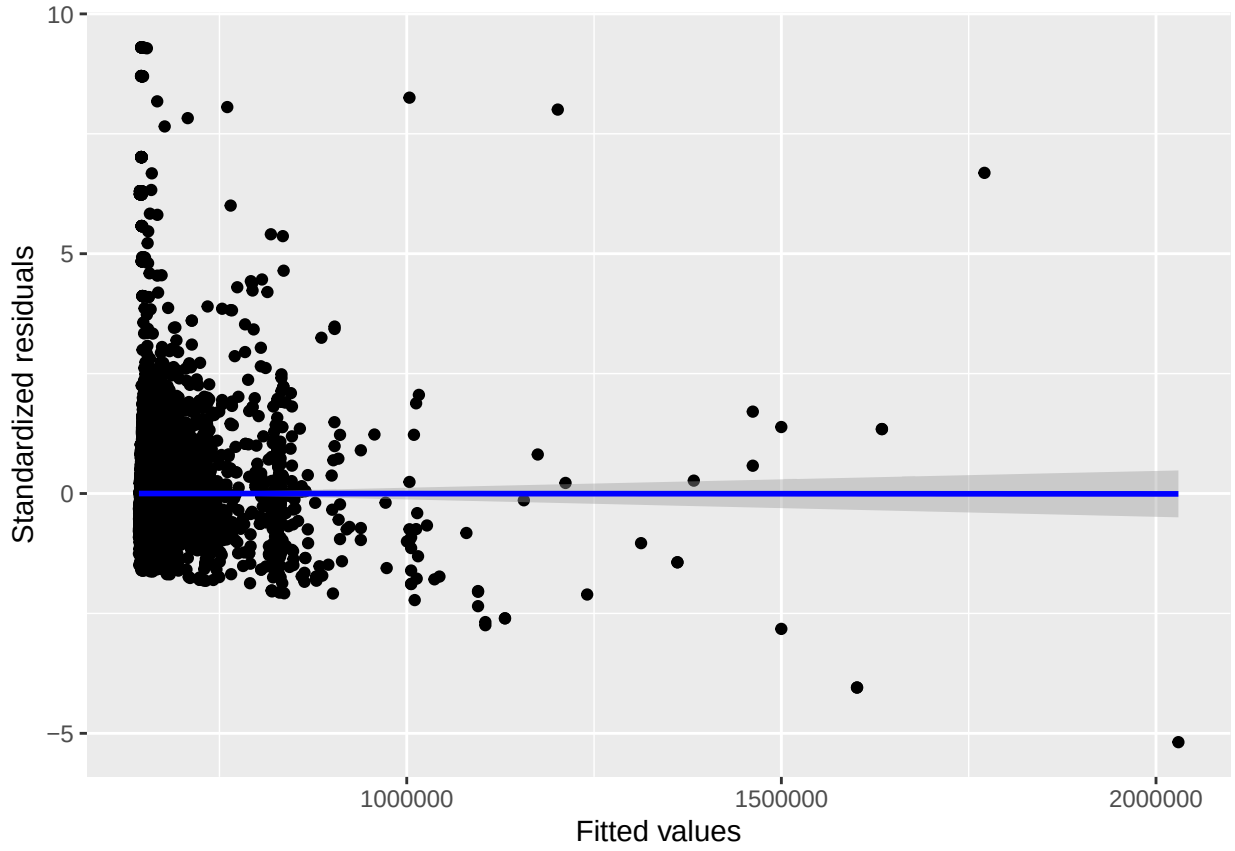
```
#3. Get a summary of your first model and explain your results (i.e., R2, adj. R2, etc.)
#We can see the R2 as 0.01435, which means the sq_ft_lot is accounting to only 1.4% of Sale price
#Which also means the remaining 98.5% of Sale Price is being accounted by other parameters.
#Adjusted R2 (0.01428) means the predictive power or shrinkage and says how well the model generalizes
#for population.It is almost same as R2, which is almost ideal and it means that if this model is
#derived from population rather than sample, there would be no shrinkage or variance when compared to
#model from sample.

#4.  Get the residuals of your model (you can use 'resid' or 'residuals' functions) and plot them.
#What the does the plot tell you about your predictions?
#Storing the residuals/standard residuals and fitted values as columns in the original dataframe
df$residuals <- resid(model)
df$standardized.residuals <- rstandard(model)
df$studentized.residuals <- rstudent(model)
df$fitted <- model$fitted.values
df
```

```
## # A tibble: 12,861 x 28
##    `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>
## # i 12,851 more rows
## # i 23 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>, ...
```

```
#Plotting fitted values vs standardized residuals
library(ggplot2)
scatter <- ggplot(df,aes(fitted,standardized.residuals))
scatter + geom_point() + geom_smooth(method = "lm", color = "Blue") +
  labs(x="Fitted values", y="Standardized residuals")
```

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

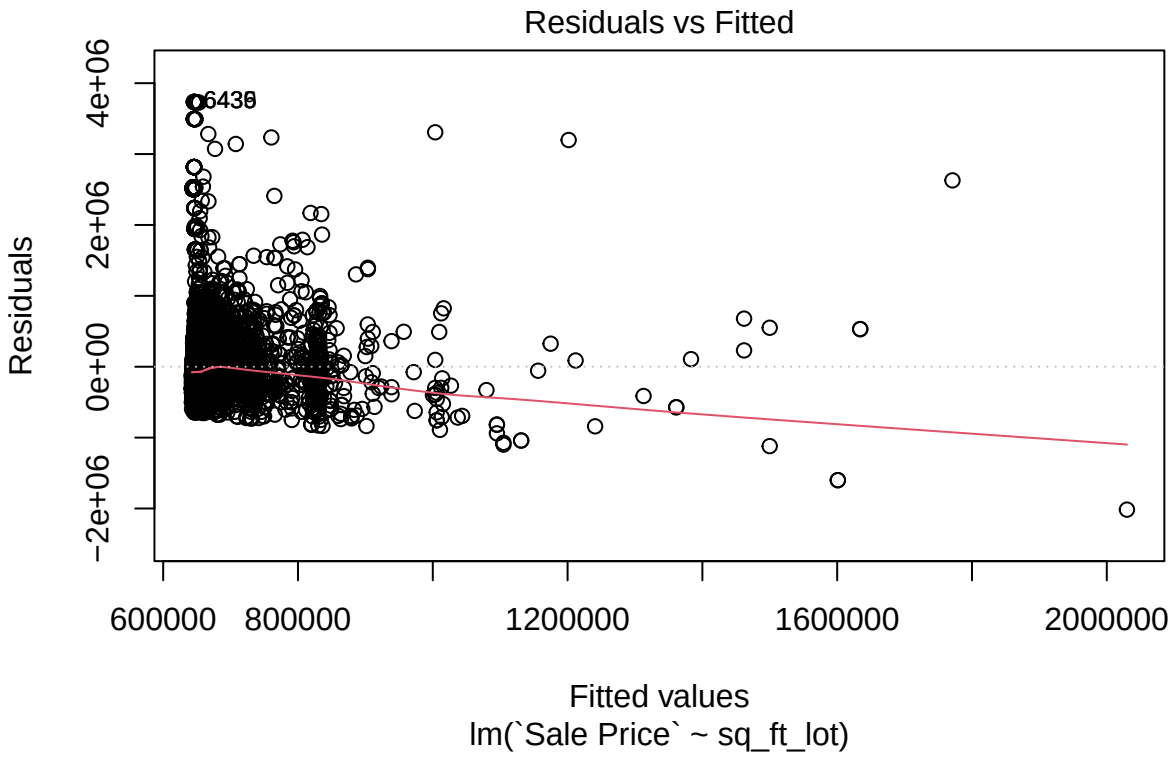


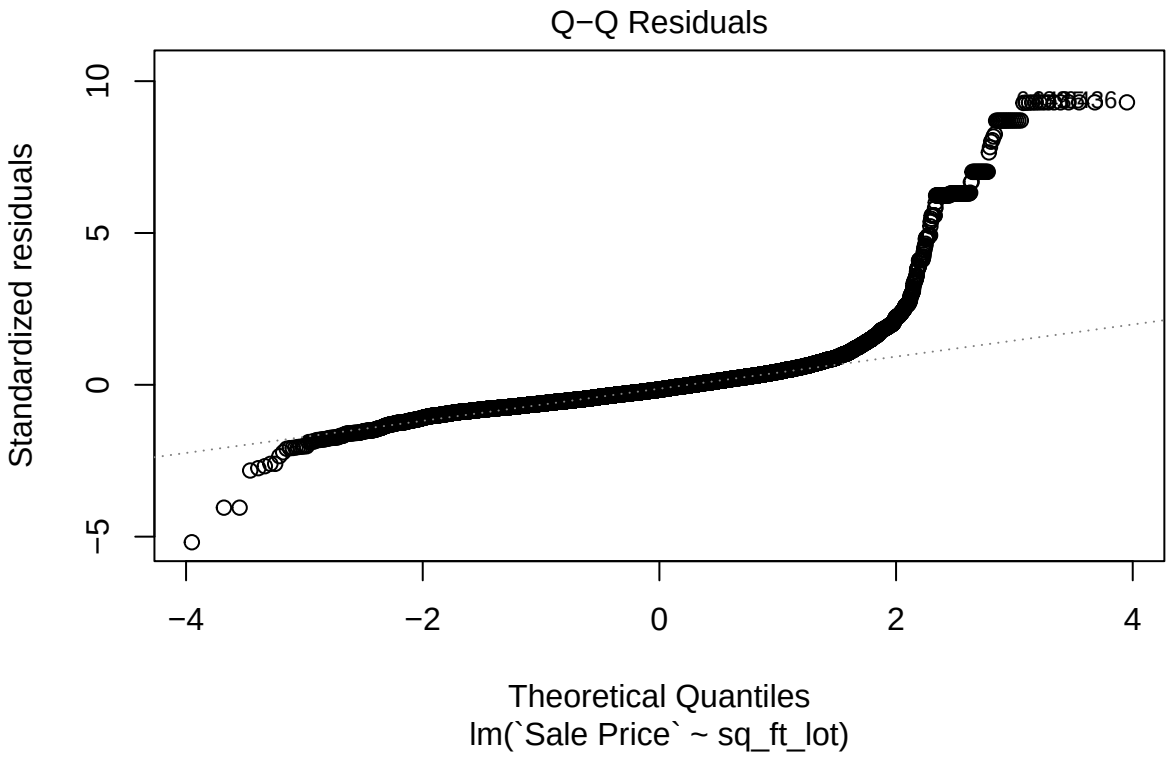

```
cor(df$`Sale Price`,df$sq_ft_lot)
```

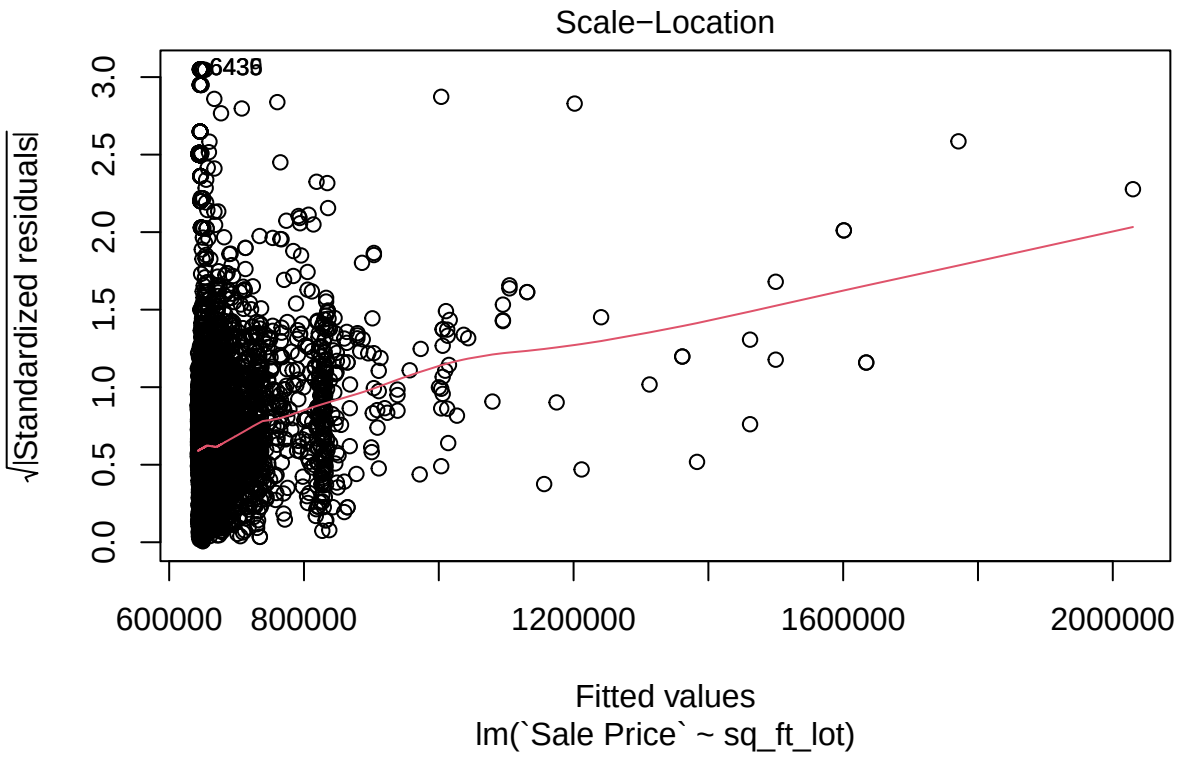
```
## [1] 0.1197999
```

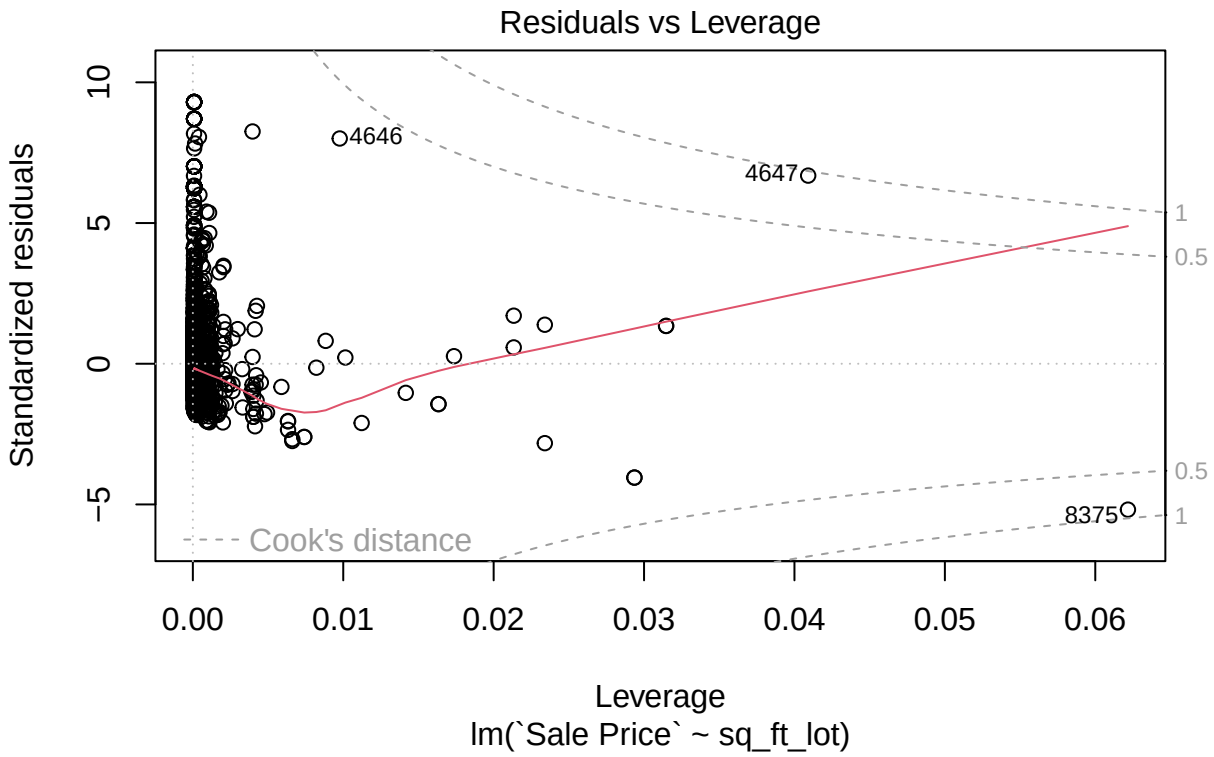
```
#From the plot I can say there is a clear non linear relationship between predicted and predictor
#variables. It means that the linearity assumption has been violated and it has also been checked by
#running correlational test. Also there is no even distribution of dots around zero.

#5.  Use a qq plot to observe your residuals. Do your residuals meet the normality assumption?
#I also used the plot on the model generated
plot(model)
```


Residuals vs Fitted
Residuals
4e+06
2e+06
0e+00
−2e+06
600000
800000
1200000
1600000
2000000
Fitted values
lm(`Sale Price` ~ sq_ft_lot)


Q−Q Residuals
Standardized residuals
10
5
0
−5
−4
−2
0
2
4
Theoretical Quantiles
lm(`Sale Price` ~ sq_ft_lot)


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
3.0
600000
800000
1200000
1600000
2000000
Fitted values
lm(`Sale Price` ~ sq_ft_lot)



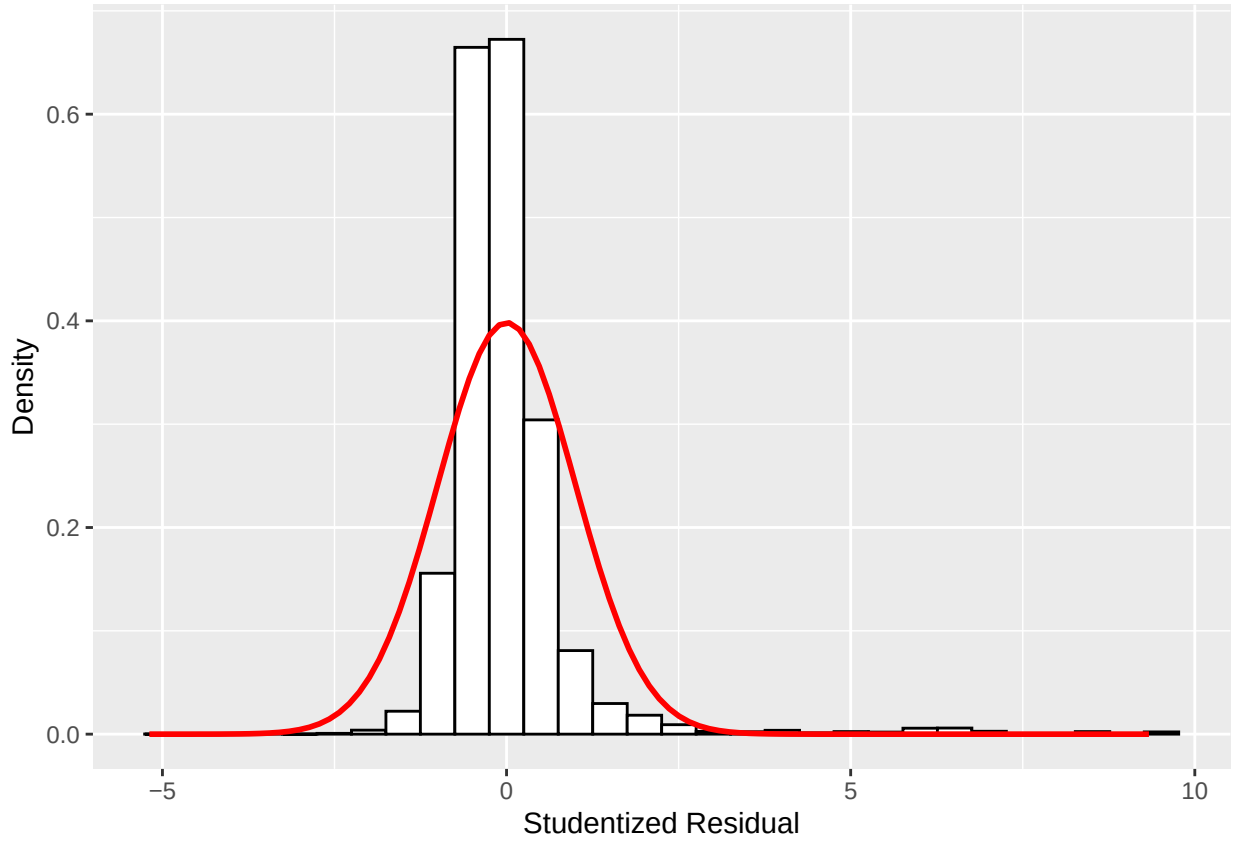

```
#Looking at the QQ plot, I can see there is a very clear indication of Non-normally distributed
#residuals, as all the observed residuals are not lying on the straight line.
#Same observation has been confirmed by running histogram of standardized residuals.

histogram1<-ggplot(df, aes(studentized.residuals)) +
  geom_histogram(aes(y = ..density..), colour = "black", fill = "white") +
  labs(x = "Studentized Residual", y = "Density")
histogram1 +
  stat_function(fun = dnorm, args = list(mean = mean(df$studentized.residuals, na.rm = TRUE),
                                         sd = sd(df$studentized.residuals, na.rm = TRUE)),
                            colour = "red", size = 1)
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.

## Warning: The dot-dot notation (‘..density..‘) was deprecated in ggplot2 3.4.0.
## i Please use ‘after_stat(density)‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.

## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```




```
#MODEL2:
#======


#6.  Now, create a linear regression model that uses multiple predictor variables to predict
#Sale Price (feel free to derive new predictors from existing ones). Explain why you think each of
#these variables may add explanatory value to the model.

#Check if any of these variables are having NA values,

sum(is.na(df$square_feet_total_living))
```

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

```
sum(is.na(df$year_built))
```

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

```
sum(is.na(df$building_grade))
```

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

```r
#I am going to create the next linear multiple regression model by using square_feet_total_living and
#year_built and building grade.
model2 <- lm(`Sale Price` ~ square_feet_total_living + year_built + building_grade, data = df,
             header = TRUE)
```

```
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'header' will be disregarded
```

```r
# I have picked up these  predictors as I strongly feel that the sale price is based on the total sqft,
#the grade of the building and also how new is the house. Historically, this is the way housing
#industry works.

#7. Get a summary of your next model and explain your results.
# Following is the summary,
summary(model2)
```

```
##
## Call:
## lm(formula = `Sale Price` ~ square_feet_total_living + year_built +
##     building_grade, data = df, header = TRUE)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1684753  -120025   -43929    41336  3968769
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)              -4.715e+06  3.840e+05 -12.279  < 2e-16 ***
## square_feet_total_living  1.464e+02  4.775e+00  30.650  < 2e-16 ***
## year_built                2.377e+03  1.964e+02  12.104  < 2e-16 ***
## building_grade            3.244e+04  4.417e+03   7.344  2.2e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 356800 on 12857 degrees of freedom
## Multiple R-squared:  0.2217, Adjusted R-squared:  0.2215
## F-statistic:  1221 on 3 and 12857 DF,  p-value: < 2.2e-16
```

```r
df$m2residuals <- resid(model2)
df$m2standardized.residuals <- rstandard(model2)
df$m2studentized.residuals <- rstudent(model2)
df$m2fitted <- model2$fitted.values


#We can see the R2 as 0.22, which means the combination of Sqft_total living, year_built and
#building_grade are accounting to 22% of Sale price
#Adjusted R2 (0.22) means the predictive power or shrinkage and says how well the model generalizes
#for population.It is same as R2, which is almost ideal and it means that if this model is
#derived from population rather than sample, there would be no shrinkage or variance when compared to
#model from sample.
#Looking at the coefficients there are no -ve values, which means that as these predictor variables
#increase predicted values also goes up.
```

```
#Every one unit increase of sqft_totalliving will increase the sales price by 146$
#Every one unit increase of year_built will increase the sales price by 2377$
#Every one unit increase of building_grade will increase the sales price by 32440$
#As all the Pr(>|t|) values are less than 0.05, these predictors are making significant contribution
#From the value of t, we can see which of these predictors are more significant. I can say that the
#sq_ft_totalliving has a t value of 30 and year_built has 12 and these contribute more when compared to
#building_grade which is only 7. Using the standardized beta values, sq_ft_totalliving value is
#significantly higher and year_built comes next.These two give a higher importance of predictor
#variables in the model.

library(QuantPsyc)
```

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

```
lm.beta(model2)
```

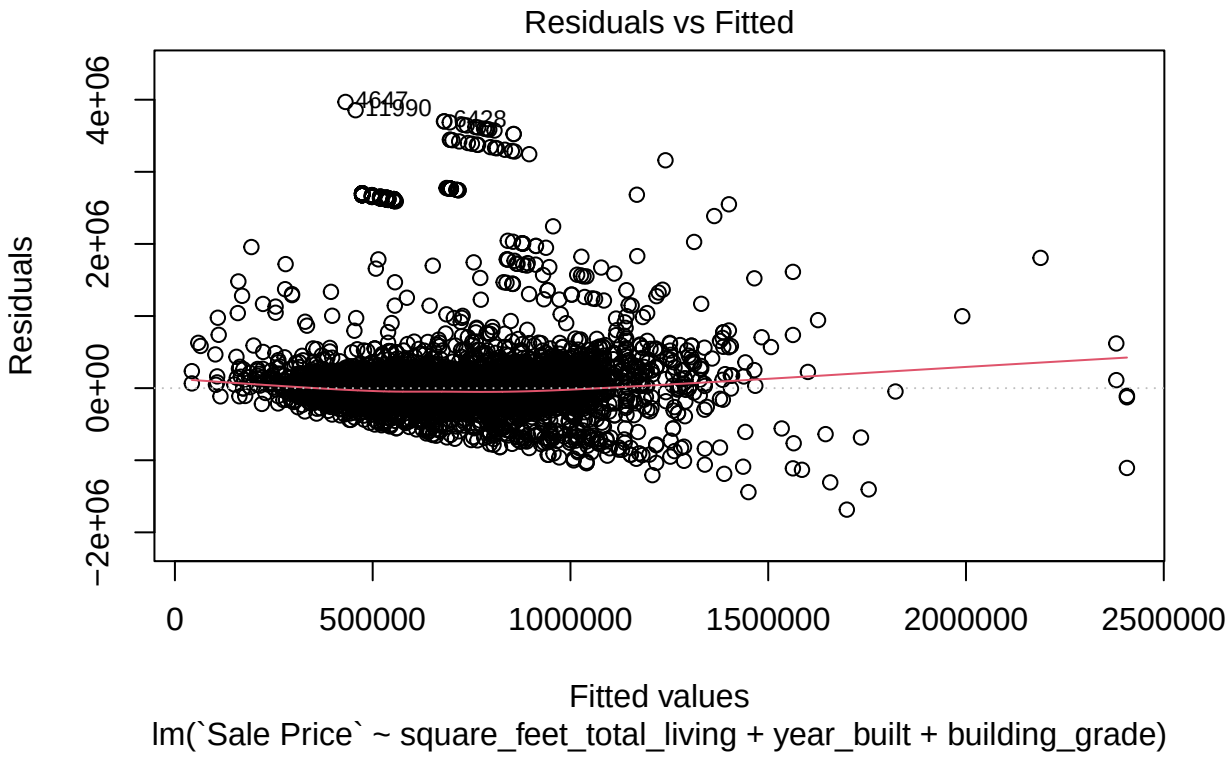
```
## square_feet_total_living               year_built          building_grade
##                0.3582307                0.1011971                0.0876422
```

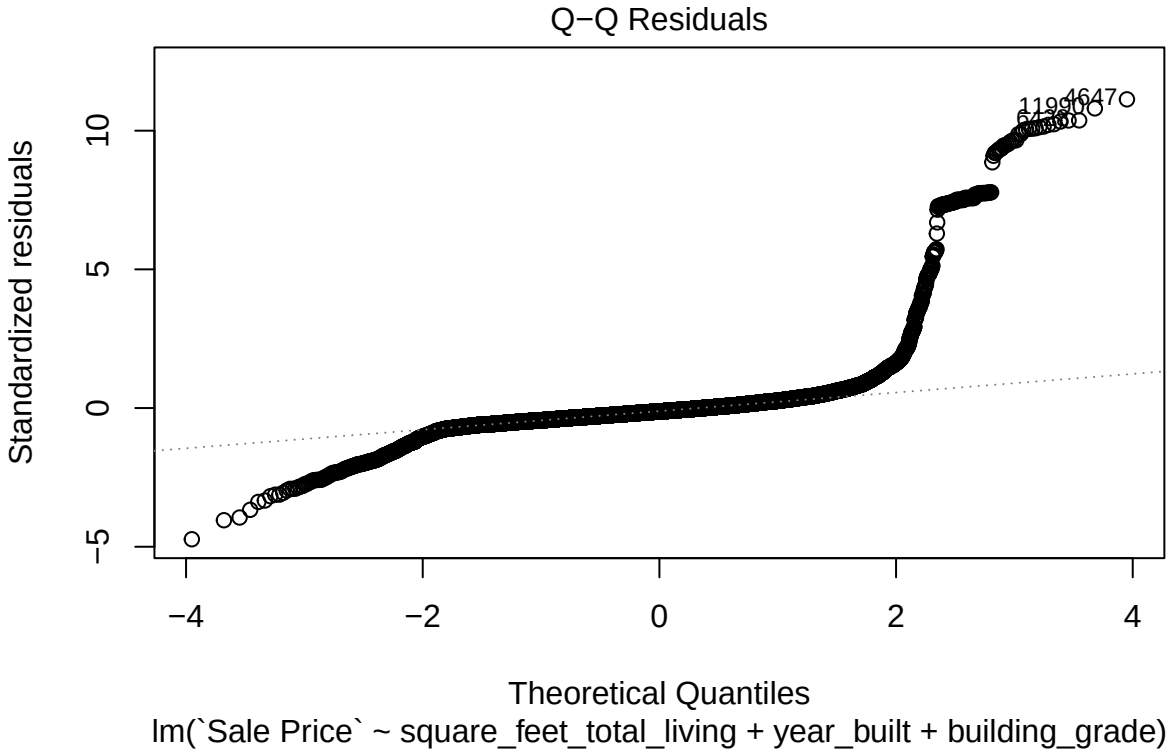
```
#I have also calculated the CI of the b values,
confint(model2)
```

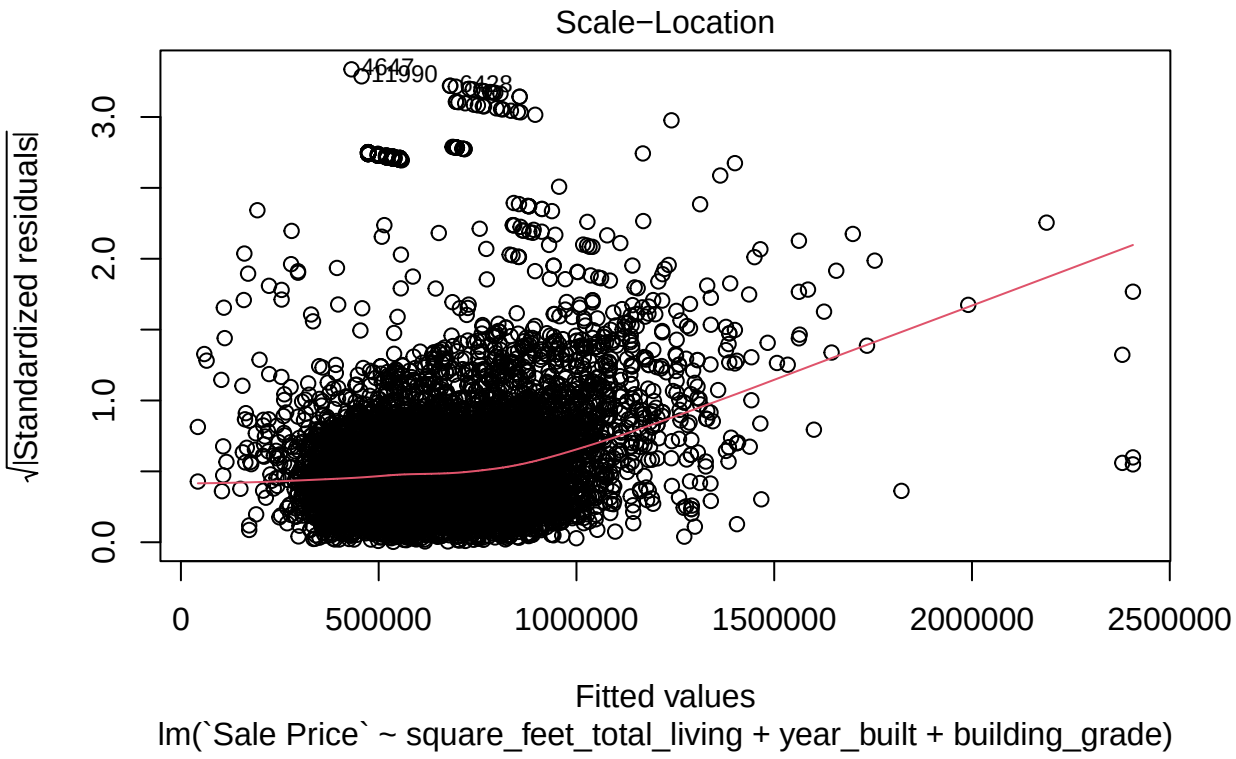
```
##                                  2.5 %        97.5 %
## (Intercept)              -5468148.0334 -3962703.9259
## square_feet_total_living      136.9964      155.7164
## year_built                   1991.9892     2761.8158
## building_grade              23780.1954    41096.9047
```

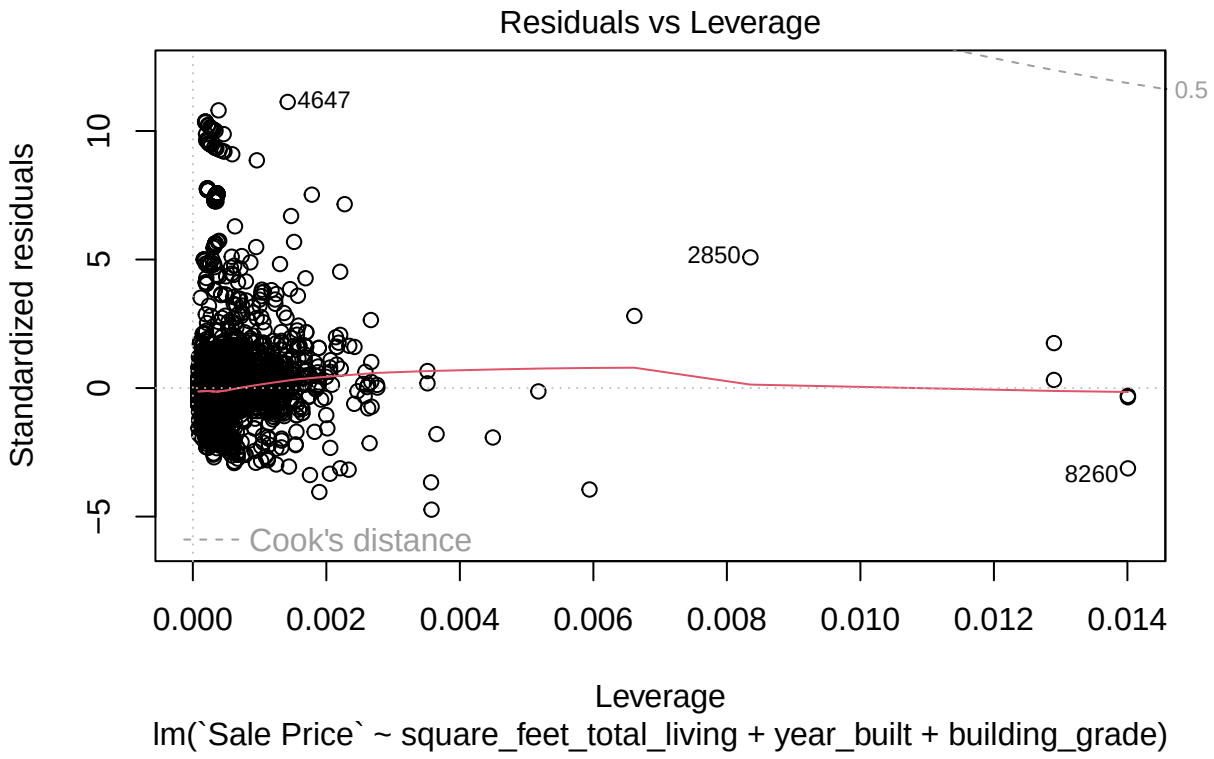
```
#All are showing positive which means as predictor increase predicted value also goes up.
#The CI of sq_ft_totalliving is small meaning that the value of b in this sample is close to the value
#of b in population.

#8. Get the residuals of your second model (you can use 'resid' or 'residuals' functions) and plot them
#What the does the plot tell you about your predictions?
plot(model2)
```


Residuals vs Fitted
Residuals
Fitted values
lm(`Sale Price` ~ square_feet_total_living + year_built + building_grade)
4647
11990
6428
4e+06
2e+06
0e+00
−2e+06
0
500000
1000000
1500000
2000000
2500000

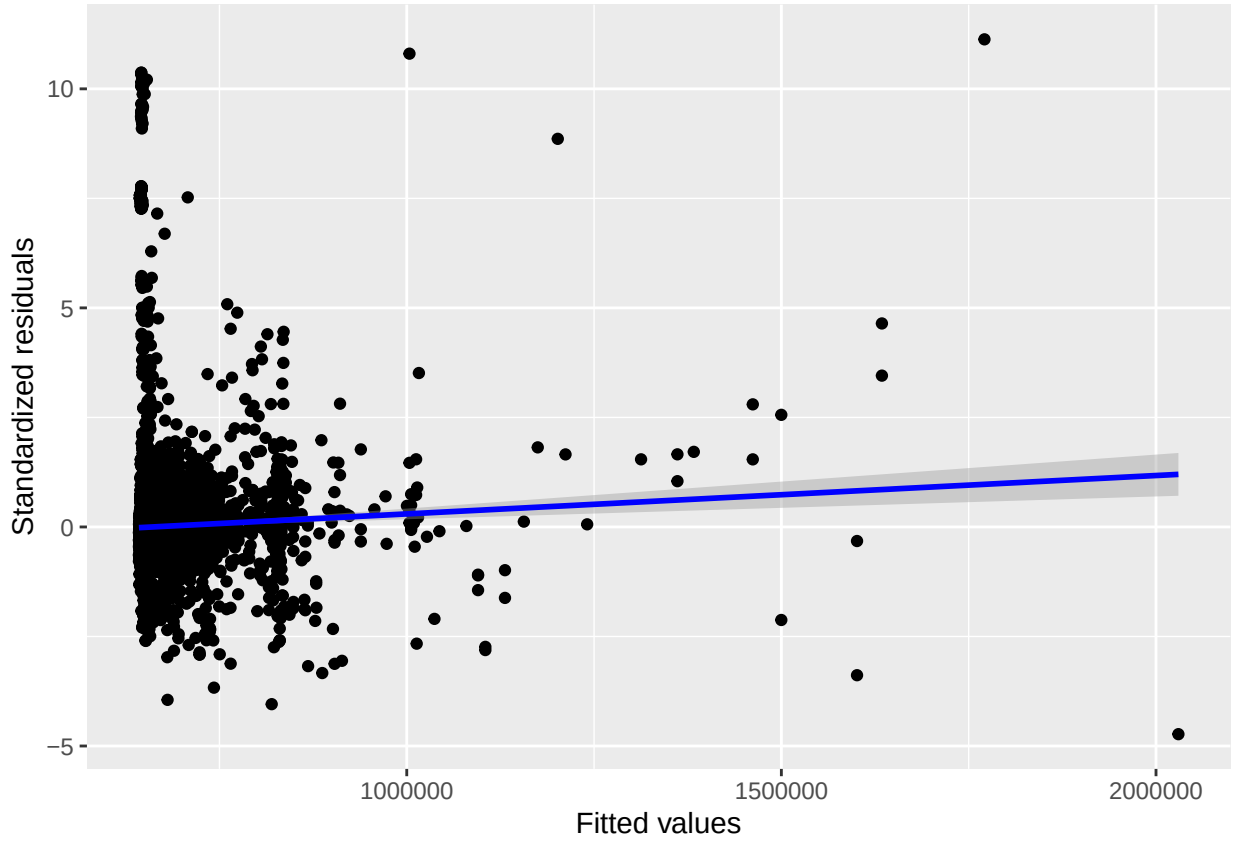

Q−Q Residuals
Standardized residuals
10
5
0
−5
−4
−2
0
2
4
4647
Theoretical Quantiles
lm(`Sale Price` ~ square_feet_total_living + year_built + building_grade)


Scale−Location
√|Standardized residuals|
0.0
1.0
2.0
3.0
4647
11990
6428
0
500000
1000000
1500000
2000000
2500000
Fitted values
lm(`Sale Price` ~ square_feet_total_living + year_built + building_grade)




```
scatter1 <- ggplot(df,aes(fitted,m2standardized.residuals))
scatter1 + geom_point() + geom_smooth(method = "lm", color = "Blue") +
  labs(x="Fitted values", y="Standardized residuals")
```

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

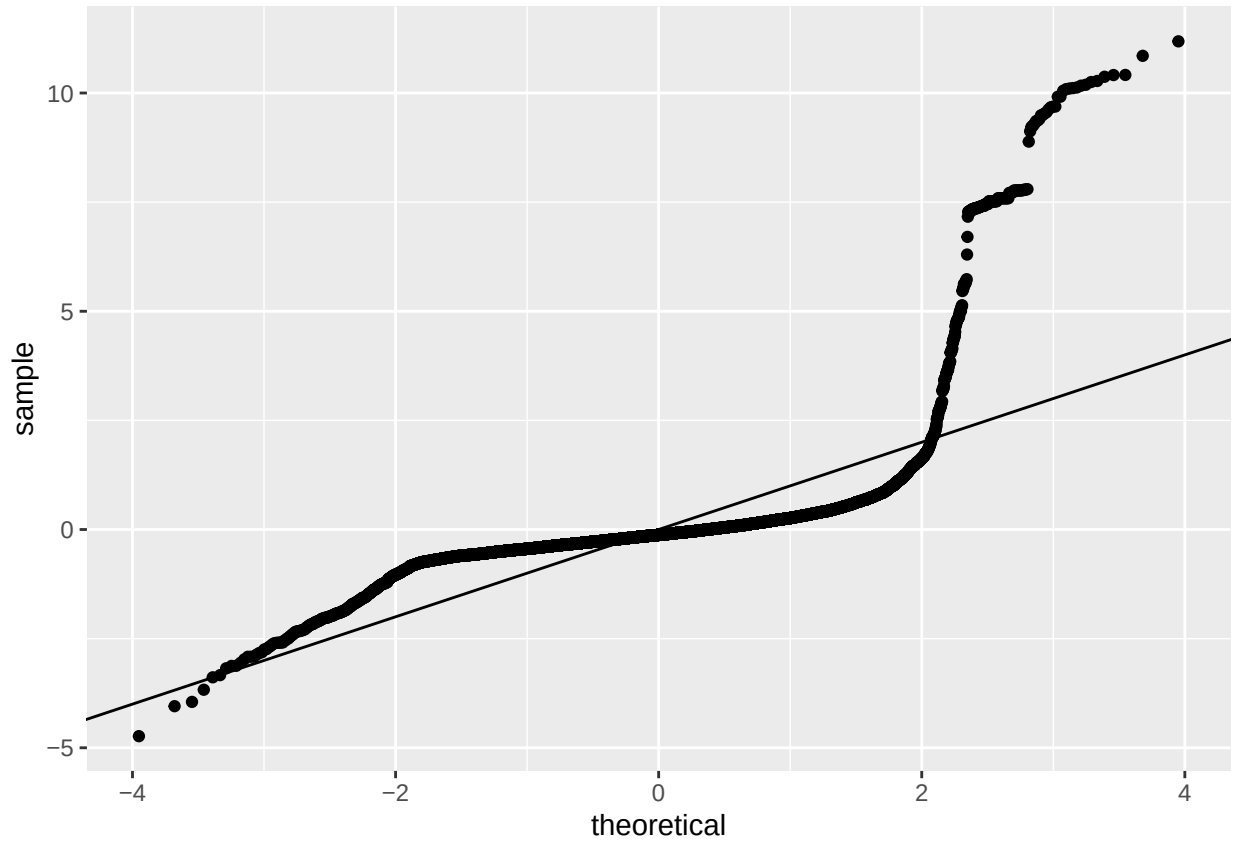



```
#The plot of Residuals vs Fitted is evenly distributed around 0 and above and below regression line
#and it satisfies the conditions of linearity and homoscedasticity.


#9. Use a qq plot to observe your residuals. Do your residuals meet the normality assumption?

library(ggplot2)

qqplot.m2resid <- ggplot(df, aes(sample=m2studentized.residuals)) + stat_qq() + geom_abline()
qqplot.m2resid
```

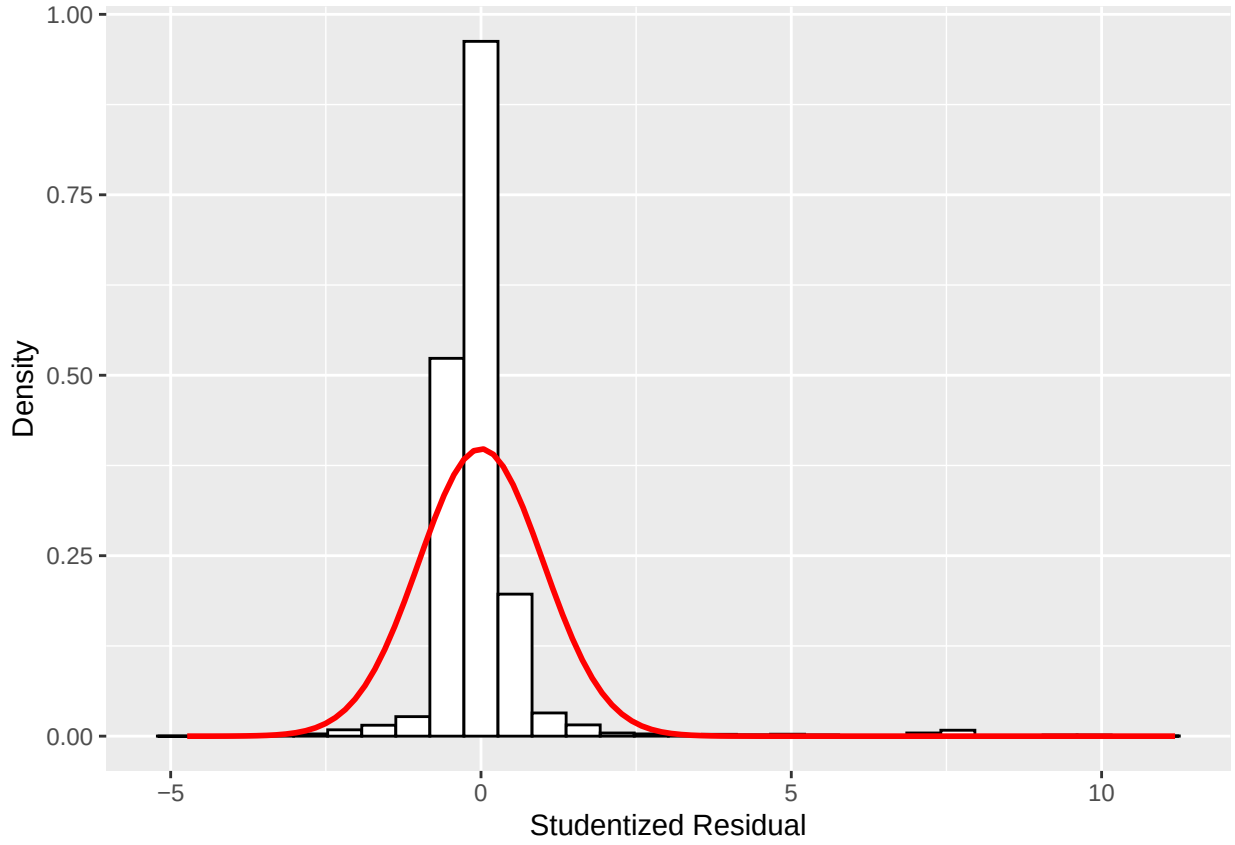



```
#Also most of the observations lie on QQ plot.  However some are also lying outside of the straight lin
#This confirms the normality assumption of residuals.  Same thing is confirmed by the histogram as show

histogram2<-ggplot(df, aes(m2studentized.residuals)) +
  geom_histogram(aes(y = ..density..),
                 colour = "black", fill = "white") +
  labs(x = "Studentized Residual", y = "Density")
histogram2 +
  stat_function(fun = dnorm,
                args = list(mean = mean(df$m2studentized.residuals, na.rm = TRUE), sd =
                                                    sd(df$m2studentized.residuals, na.rm = TRUE)),
                colour = "red", size = 1, bins = 10)
```

```
## Warning in stat_function(fun = dnorm, args = list(mean =
## mean(df$m2studentized.residuals, : Ignoring unknown parameters: `bins`

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
#I am also confirming if any of the residuals which are outside -2 and +2.
#It is 327, which is less than 5% of the total 12861
df$m2large.residuals <- df$m2standardized.residuals > 2 | df$m2standardized.residuals < -2
sum(df$m2large.residuals)
```

```
## [1] 327
```

```
#The cooks distance has been calculated, all of them lie under 1, which is not a cause of concern.
#This says it is a reliable model.

df$m2cooks.distance <- cooks.distance(model2)
nrow(df)
```

```
## [1] 12861
```

```
sum(df$m2cooks.distance < 1)
```

```
## [1] 12861
```

```
#10. Compare the results (i.e., R2, adj R2, etc) between your first and second model.
#Does your new model show an improvement over the first? To confirm a 'significant'
#improvement between the second and first model, use ANOVA to compare them. What are the results?

#Yes comparing the R2 and adj R2 from first model and second model, there is a significant improvement
```

```r
#R2 and adj R2 from 0.014 to 0.22
#Using anova test to compare the two models, model and model2.
anova(model,model2)
```

```
## Analysis of Variance Table
##
## Model 1: `Sale Price` ~ sq_ft_lot
## Model 2: `Sale Price` ~ square_feet_total_living + year_built + building_grade
##   Res.Df        RSS Df  Sum of Sq      F    Pr(>F)
## 1  12859 2.0732e+15
## 2  12857 1.6371e+15  2 4.3608e+14 1712.3 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
#The RSS which is residual sum of squares is lower for model2 saying it is better.
#For the second model we are having F values as 1712.3 with probability less than .05.
#Hence model2 is a better model and we can use that one over model1.

#Checked by using AIC and BIC, lower the better, which is for model2
AIC(model,model2)
```

```
##        df      AIC
## model   3 368393.9
## model2  5 365360.8
```

```r
BIC(model,model2)
```

```
##        df      BIC
## model   3 368416.3
## model2  5 365398.1
```

```r
#11. After observing both models (specifically, residual normality),
#provide your thoughts concerning whether the model is biased or not.

#By checking for the collinearity using vif, they are all within 1, so the model is not biased
library(car)
```

```
## Loading required package: carData
##
## Attaching package: 'car'
##
## The following object is masked from 'package:boot':
##
##     logit
##
## The following object is masked from 'package:dplyr':
##
##     recode
##
## The following object is masked from 'package:purrr':
##
##     some
```

```r
vif(model2)
```

```
## square_feet_total_living               year_built          building_grade
##                 2.256594                 1.154617                 2.352735
```

```r
#12. Another important aspect of regression tasks is determining the accuracy of your predictions.
#For this section, we will look at root mean square error (RMSE), a common accuracy metric for
#regression models.

#Predicting using predict() function by creating a new data frame,
newdf <- data.frame(square_feet_total_living=c(1500,900,2810),year_built=c(1963,1980,2003),
                    building_grade=c(1,2,9))
newdf
```

```
##   square_feet_total_living year_built building_grade
## 1                     1500       1963              1
## 2                      900       1980              2
## 3                     2810       2003              9
```

```r
predict(model2,newdf)
```

```
##        1        2        3
## 202406.8 187438.8 748718.2
```

```r
#Predicting using predict() function on the existing data frame using model1,

summary(model)
```

```
##
## Call:
## lm(formula = `Sale Price` ~ sq_ft_lot, data = df, header = TRUE)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2015913  -194880   -63332    91528  3735071
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.419e+05  3.801e+03  168.87   <2e-16 ***
## sq_ft_lot   8.509e-01  6.218e-02   13.68   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 401500 on 12859 degrees of freedom
## Multiple R-squared:  0.01435,    Adjusted R-squared:  0.01428
## F-statistic: 187.2 on 1 and 12859 DF,  p-value: < 2.2e-16
```

```r
summary(model2)
```

```
##
```

```
## Call:
## lm(formula = `Sale Price` ~ square_feet_total_living + year_built +
##     building_grade, data = df, header = TRUE)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1684753  -120025   -43929    41336  3968769
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)               -4.715e+06  3.840e+05 -12.279  < 2e-16 ***
## square_feet_total_living   1.464e+02  4.775e+00  30.650  < 2e-16 ***
## year_built                 2.377e+03  1.964e+02  12.104  < 2e-16 ***
## building_grade             3.244e+04  4.417e+03   7.344  2.2e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 356800 on 12857 degrees of freedom
## Multiple R-squared:  0.2217, Adjusted R-squared:  0.2215
## F-statistic:  1221 on 3 and 12857 DF,  p-value: < 2.2e-16
```

```
preds1 <- predict(model,df)
head(preds1)
```

```
##        1        2        3        4        5        6
## 647505.5 646599.4 649044.8 650028.4 648263.7 648054.4
```

```
df$m1predicted <- preds1
head(df$m1predicted)
```

```
##        1        2        3        4        5        6
## 647505.5 646599.4 649044.8 650028.4 648263.7 648054.4
```

```
#Predicting using predict() function on the existing data frame using model2,

preds <- predict(model2,df)
head(preds)
```

```
##        1        2        3        4        5        6
## 748718.2 766093.8 672394.9 458923.9 428664.0 886176.0
```

```
df$m2predicted <- preds
head(df$m2predicted)
```

```
##        1        2        3        4        5        6
## 748718.2 766093.8 672394.9 458923.9 428664.0 886176.0
```

```
#Calculating the RMSE for both the models,

library(Metrics)
rmse(df$`Sale Price`,df$m1predicted)
```

```
## [1] 401499.6
```

```
rmse(df$`Sale Price`,df$m2predicted)
```

```
## [1] 356784.4
```

```
#The lower the RMSE, better is the model.  The second model has lower value of 356784.4 when compared
#to the 401499.6. There is a difference of 44,715$, hence the second model is a better model.
```