Exercise 8.2 Housing Data

Janine Par

2022-05-12

```
## Loading required package: carData

##
## Attaching package: 'dplyr'

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

## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

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

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

```r
#A

## Set the working directory to the root of your DSC 520 directory
setwd("C:/Users/janin/OneDrive/Documents/R_repo/dsc520/")

## Load the `data/r4ds/housing.xlsx'
housing_df<- read_excel("data/week-6-housing.xlsx")
#count na
sum(is.na(housing_df$ctyname))
```

```
## [1] 6078
```

```r
# since I'm not using this field as predictor. I'm not filtering na on ctyname
    #housing_df <- housing_df %>% filter(!is.na(ctyname))

# Renaming
housing_df <- rename(housing_df,sale_price=`Sale Price`)
housing_df <-rename(housing_df,sale_date=`Sale Date`)

# split year and month
split_year_month <- function(x)
{
  x_year <- format(x,format = "%Y")
  x_month<- format(x,format = "%m")
  df <- data.frame(x_year,x_month)
}

sale_year_month <- split_year_month(housing_df$sale_date)

housing_df <- cbind(housing_df,sale_year_month)

#convert to number
housing_df$year_built <- as.numeric(housing_df$year_built)
housing_df$x_year <- as.numeric(housing_df$x_year)

#rename
housing_df <-rename(housing_df,sale_year=x_year)
housing_df <-rename(housing_df,sale_month=x_month)

#calculate how old the house on year of sale
housing_df$year_old <- housing_df$sale_year - housing_df$year_built

housing_df$year_old <- ifelse(housing_df$year_old < 1,0,housing_df$year_old)
```

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

Performed the following Transformation and Modification on the Housing Dataset

1. Rename column for easier reference
2. Explore records that are null(na)
3. Created new variable of the Month and Year of sales. This may be used for further analysis
4. Created new variable to get the how old is the house on the year of sale

```r
#B ii.  Create two variables; one that will contain the variables Sale Price and Square Foot of Lot (sa
#and one that will contain Sale Price and several additional predictors of your choice.
#Explain the basis for your additional predictor selections.

houseModelsimpleR <- lm(sale_price ~ sq_ft_lot, data=housing_df)

houseModelmultipleR <- lm(sale_price ~ sq_ft_lot
                          + square_feet_total_living + year_old +bath_full_count ,  data=housing_df)
```

Predictors

I have selected these additional predictors since they are continuous variables and also based on my experience when looking for house, these are key indicators that influence the house price.

1. Square feet of livable space. How big is the house.
2. How old is the house(Years Old)
3. Number of Full Bath

```
#B iii. Execute a summary() function on two variables defined in the previous step to compare the model

summary(houseModelsimpleR)
```

```
##
## Call:
## lm(formula = sale_price ~ sq_ft_lot, data = housing_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2016064  -194842   -63293    91565  3735109
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.418e+05  3.800e+03  168.90   <2e-16 ***
## sq_ft_lot   8.510e-01  6.217e-02   13.69   <2e-16 ***
## ---
## 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: < 2.2e-16
```

```
summary(houseModelmultipleR)
```

```
##
## Call:
## lm(formula = sale_price ~ sq_ft_lot + square_feet_total_living +
##     year_old + bath_full_count, data = housing_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2110329  -120363   -43019    46767  3776089
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)              2.475e+05  1.328e+04  18.633  < 2e-16 ***
## sq_ft_lot                2.752e-01  5.843e-02   4.711 2.49e-06 ***
## square_feet_total_living  1.633e+02  3.875e+00  42.145  < 2e-16 ***
## year_old                -2.340e+03  2.059e+02 -11.362  < 2e-16 ***
## bath_full_count          1.935e+04  6.062e+03   3.192  0.00142 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 357600 on 12860 degrees of freedom
## Multiple R-squared:  0.2182, Adjusted R-squared:  0.218
## F-statistic: 897.3 on 4 and 12860 DF,  p-value: < 2.2e-16
```

#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?

Simple Regression Multiple R-squared: 0.01435, Adjusted R-squared: 0.01428

Multiple Regression Multiple R-squared: 0.2186, Adjusted R-squared: 0.2183

**The inclusion of additional predictors increased the value of R2 which means that the additional predictors added can account for 21.8% of the variation on sales price in comparison with the simple regression where sq_ft_lot can only account for 1.42% of the variation on sales price.

```
#B iv.  Considering the parameters of the multiple regression model you have created. What are the stan

lm.beta(houseModelmultipleR)
```

```
##
## Call:
## lm(formula = sale_price ~ sq_ft_lot + square_feet_total_living +
##     year_old + bath_full_count, data = housing_df)
##
## Standardized Coefficients::
##              (Intercept)                sq_ft_lot square_feet_total_living
##                       NA               0.03875047               0.39972892
##                 year_old          bath_full_count
##              -0.10131076               0.03113786
```

These are estimates that tells us the standard deviations by which outcome may change as a result of one standard change in the predictor.

sq__ft__lot (b=0.03875047) - This tells us that if the sq_ft_lot increases by 1 the housing sale price increases by 0.03875047

square__feet__total__living (b=0.39972892) - This tells us that if the square feet total living increases by 1 the housing sale price increases by 0.39972892

year__old (b= -0.10131076) - This tells us that if the year old increases by 1 the housing sale price decreases by -0.10131076

Bath__full__count(b=0.03113786) - This tells us that if the bath full count increases by 1 the housing sale price increases by 0.03113786

```
#B v. Calculate the confidence intervals for the parameters in your model and explain what the results 
confint(houseModelmultipleR)
```

Confidence Interval no values cross over zero thus indicates that this is not a bad model. The best predictor is sq_ft_lot, square_feet_total_living and year_old because of tight confidence interval while bath_full_count have wider range indicating that this range is less representative

```
#B vi.  Assess the improvement of the new model compared to your original model (simple regression mode

anova(houseModelsimpleR, houseModelmultipleR)
```

```
## Analysis of Variance Table
##
## Model 1: sale_price ~ sq_ft_lot
## Model 2: sale_price ~ sq_ft_lot + square_feet_total_living + year_old +
##     bath_full_count
##   Res.Df        RSS Df  Sum of Sq      F    Pr(>F)
## 1  12863 2.0734e+15
## 2  12860 1.6446e+15  3 4.2881e+14 1117.7 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Anova Result The value of Pr(>F) is < 2.2e-16 *** which is very small, which we can say that the houseModelmultipleR improved the model significantly compare to houseModelsimpleR

```
#B vii. Perform casewise diagnostics to identify outliers and/or influential cases, storing each functi

#storing each function's output in a dataframe assigned to a unique variable name.

housing_df$residuals <-  resid(houseModelmultipleR)
housing_df$standardize_residuals <- rstandard(houseModelmultipleR)
housing_df$studentized_residuals <- rstudent(houseModelmultipleR)
housing_df$cooks_distance <- cooks.distance(houseModelmultipleR)
housing_df$dfbeta <- dfbeta(houseModelmultipleR)
housing_df$dffit <- dffits(houseModelmultipleR)
housing_df$leverage <- hatvalues(houseModelmultipleR)
housing_df$covariance_ratios <- covratio(houseModelmultipleR)
```

```
#B viii.    Calculate the standardized residuals using the appropriate command, specifying those that a

housing_df$residuals <-  resid(houseModelmultipleR)

housing_df$large_residuals <-
  housing_df$standardize_residuals >2 |  housing_df$standardize_residuals < -2
```

```
#B ix.  Use the appropriate function to show the sum of large residuals.
sum(housing_df$large_residuals)
```

```
## [1] 333
```

```
#B x. Which specific variables have large residuals (only cases that evaluate as TRUE)?
housing_df[housing_df$large_residuals=="TRUE",c("standardize_residuals",  "sq_ft_lot","square_feet_tota
```

```
#xi.    Investigate further by calculating the leverage, cooks distance, and covariance rations. Commen
housing_df[housing_df$large_residuals=="TRUE",c("cooks_distance",  "leverage", "covariance_ratios")]
```

```
housing_df$cooks_ratio <-  housing_df$cooks_distance>  1
sum (housing_df$cooks_distance_check) # Cook distance > 1 causes concern
```

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

```
housing_df$large_leverage <- housing_df$leverage>  (5/12865)*3
sum(housing_df$large_leverage)
```

```
## [1] 454
```

```
housing_df$covariance_outside <-
  housing_df$covariance_ratios <(1-((5/12865)*3)) |  housing_df$covariance_ratios > (1+((5/12865)*3))
sum(housing_df$covariance_outside)
```

```
## [1] 766
```

- None of them have cook distance > 1 which mean none of the cases having undue influence on the model
- there are 454 cases that outside the leverage boundaries
- there are 766 cases that deviate from the covariance boundaries

```
#xii.   Perform the necessary calculations to assess the assumption of independence and state if the co

dwt(houseModelmultipleR)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.8816856     0.2309635       0
##  Alternative hypothesis: rho != 0
```

- for DurbinWatsonTest, the value less than 1 or greater than 3 should raise concern and since our result is less than 1 means that the assumption is not met and p value is less than .05

```
#xiii.  Perform the necessary calculations to assess the assumption of no multicollinearity and state i

vif(houseModelmultipleR)
```

```
##                sq_ft_lot square_feet_total_living                 year_old
##                 1.113054                 1.479732                 1.307915
##          bath_full_count
##                 1.565622
```

```
1/vif(houseModelmultipleR)
```

```
##                sq_ft_lot square_feet_total_living                 year_old
##                0.8984293                0.6757981                0.7645759
##          bath_full_count
##                0.6387238
```

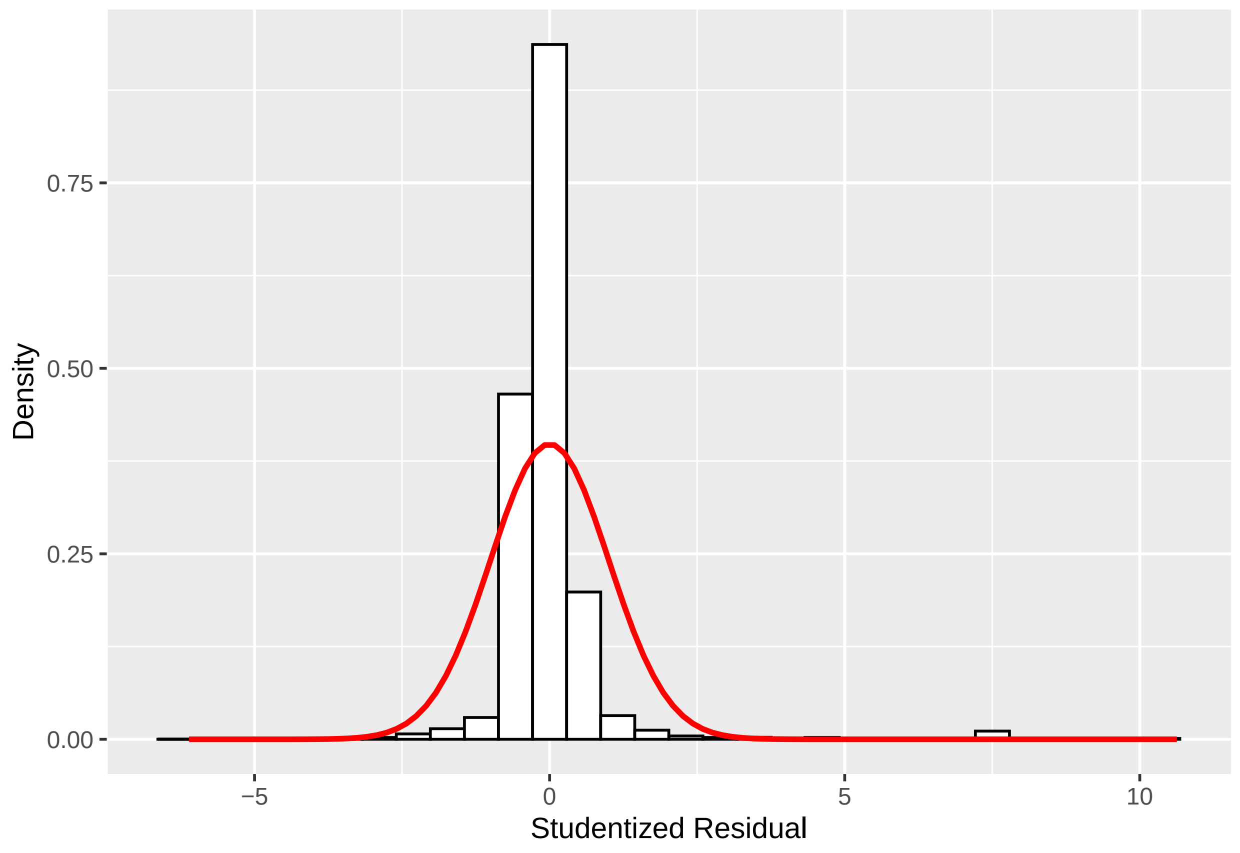
```
mean(vif(houseModelmultipleR))
```

```
## [1] 1.366581
```

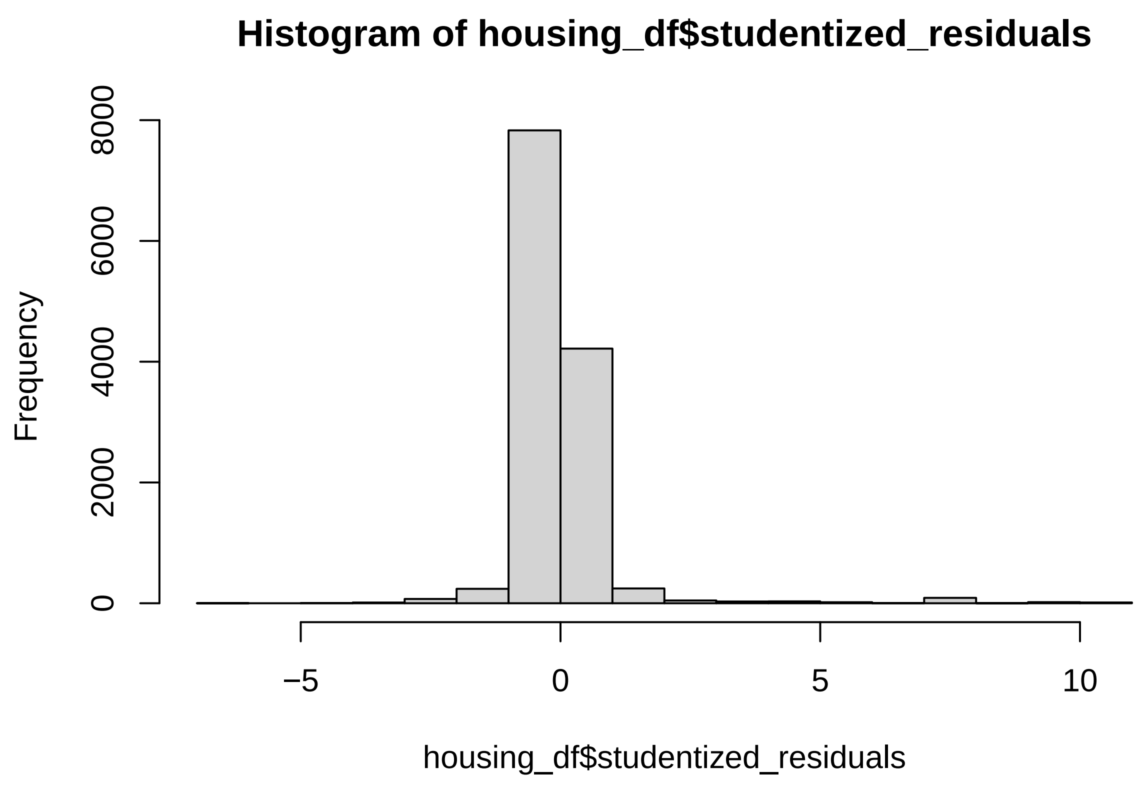
- Our current Model, the VIF values are below 10 and tolerance statistics are above 0.2 and and Average VIF is not substantially greater than 1. These are indicators that there is no collinearity in our data.

```
#xiii.  Visually check the assumptions related to the residuals using the plot() and hist() functions. 
housing_df$fitted <- houseModelmultipleR$fitted.values

histogram<-ggplot(housing_df, aes(studentized_residuals)) + geom_histogram(aes(y = ..density..), colour
histogram + stat_function(fun = dnorm, args = list(mean = mean(housing_df$studentized_residuals, na.rm 
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
hist(housing_df$studentized_residuals)
```

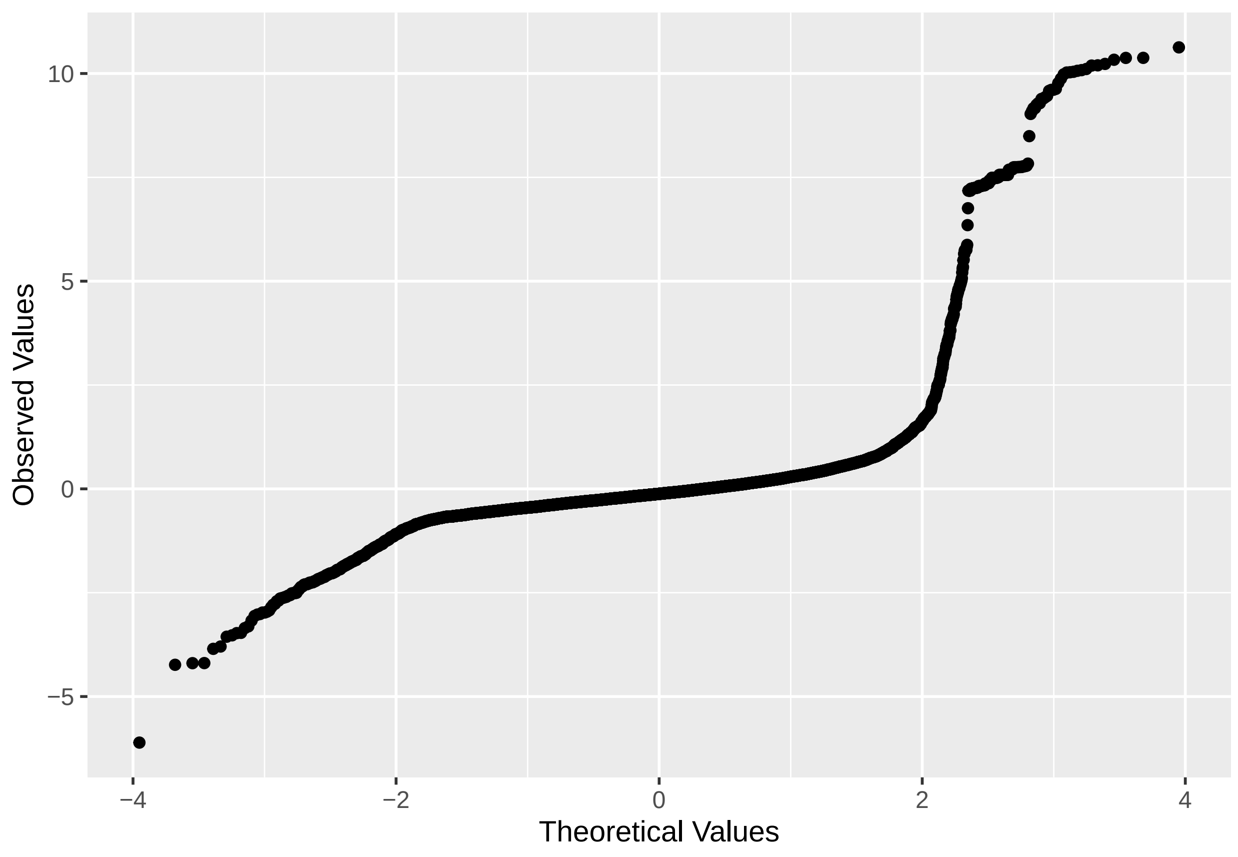


```
qqplot.resid <- qplot(sample = housing_df$studentized_residuals, stat="qq", ylab = "Observed Values", xl
```

```
## Warning: ‘stat‘ is deprecated
```

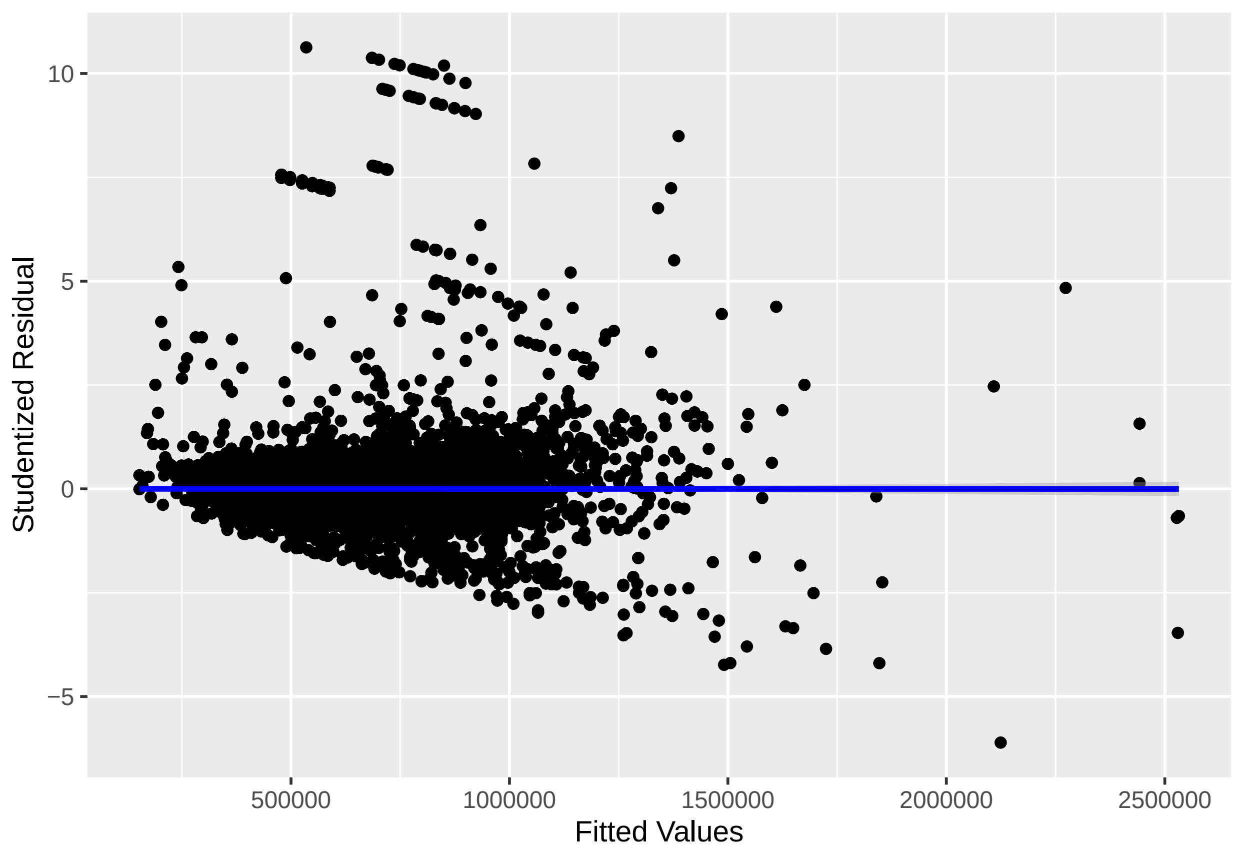
```
qplot(sample = housing_df$studentized_residuals, stat="qq") + labs(x ="Theoretical Values", y = "Observe
```

```
## Warning: ‘stat‘ is deprecated
```



```
scatter <- ggplot(housing_df, aes(fitted,studentized_residuals))
scatter + geom_point() + geom_smooth(method = "lm", colour = "Blue") + labs (x= "Fitted Values", y="Stu
```

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



* Histogram shows bell-shaped curve and shows distribution is normal * QQ plot looks like a diagonal line which also shows normality

#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?

Overall, the model shows unbiased regression model because of the following:

- Confidence Interval shows no values cross over zero thus indicates that this is not a bad model.
- None of them have cook distance > 1 which mean none of the cases having undue influence on the model
- There are 454 cases that outside the leverage boundaries
- There are 766 cases that deviate from the covariance boundaries
- No indication of multicollinearity since the VIF values are below 10 and tolerance statistics are above 0.2 and and Average VIF is not substantially greater than 1.
- Histogram shows normal distribution which mean that our model is accurate for the sample and generalized to the population.