DSC520 Week8,9 Housing Data Exercise 8.2

Venkat Jagadeesh Jampani

February 12th 2022

```
library(readxl)
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

library(purrr)
library(ggplot2)
library(lmtest)

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

library(lm.beta)
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:dplyr':
##
##     recode
```

```
## Set workding directory to read source datasets.
setwd("/Users/Jagadeesh/Documents/GitHub/dsc520")
## Read housing dataset
housingdata_df <- read_excel("data/week-6-housing.xlsx")
head(housingdata_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
## # ... with 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>

str(housingdata_df)

## tibble [12,865 x 24] (S3: tbl_df/tbl/data.frame)
##  $ Sale Date               : POSIXct[1:12865], format: "2006-01-03" "2006-
01-03" ...
##  $ Sale Price              : num [1:12865] 698000 649990 572500 420000
369900 ...
##  $ sale_reason             : num [1:12865] 1 1 1 1 1 1 1 1 1 1 ...
##  $ sale_instrument         : num [1:12865] 3 3 3 3 3 15 3 3 3 3 ...
##  $ sale_warning            : chr [1:12865] NA NA NA NA ...
##  $ sitetype                : chr [1:12865] "R1" "R1" "R1" "R1" ...
##  $ addr_full               : chr [1:12865] "17021 NE 113TH CT" "11927
178TH PL NE" "13315 174TH AVE NE" "3303 178TH AVE NE" ...
##  $ zip5                    : num [1:12865] 98052 98052 98052 98052 98052
...
##  $ ctyname                 : chr [1:12865] "REDMOND" "REDMOND" NA
"REDMOND" ...
##  $ postalctyn              : chr [1:12865] "REDMOND" "REDMOND" "REDMOND"
"REDMOND" ...
##  $ lon                     : num [1:12865] -122 -122 -122 -122 -122 ...
##  $ lat                     : num [1:12865] 47.7 47.7 47.7 47.6 47.7 ...
##  $ building_grade          : num [1:12865] 9 9 8 8 7 7 10 10 9 8 ...
##  $ square_feet_total_living: num [1:12865] 2810 2880 2770 1620 1440 4160
3960 3720 4160 2760 ...
##  $ bedrooms                : num [1:12865] 4 4 4 3 3 4 5 4 4 4 ...
##  $ bath_full_count         : num [1:12865] 2 2 1 1 1 2 3 2 2 1 ...
##  $ bath_half_count         : num [1:12865] 1 0 1 0 0 1 0 1 1 0 ...
```

```
##  $ bath_3qtr_count         : num [1:12865] 0 1 1 1 1 1 1 0 1 1 ...
##  $ year_built              : num [1:12865] 2003 2006 1987 1968 1980 ...
##  $ year_renovated          : num [1:12865] 0 0 0 0 0 0 0 0 0 0 ...
##  $ current_zoning          : chr [1:12865] "R4" "R4" "R6" "R4" ...
##  $ sq_ft_lot               : num [1:12865] 6635 5570 8444 9600 7526 ...
##  $ prop_type               : chr [1:12865] "R" "R" "R" "R" ...
##  $ present_use             : num [1:12865] 2 2 2 2 2 2 2 2 2 2 ...

glimpse(housingdata_df)

## Rows: 12,865
## Columns: 24
## $ `Sale Date`             <dttm> 2006-01-03, 2006-01-03, 2006-01-03,
2006-01-~
## $ `Sale Price`            <dbl> 698000, 649990, 572500, 420000, 369900,
18466~
## $ sale_reason             <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, ~
## $ sale_instrument         <dbl> 3, 3, 3, 3, 3, 15, 3, 3, 3, 3, 3, 3, 3,
3, 3,~
## $ sale_warning            <chr> NA, NA, NA, NA, "15", "18 51", NA, NA,
NA, NA~
## $ sitetype                <chr> "R1", "R1", "R1", "R1", "R1", "R1", "R1",
"R1~
## $ addr_full               <chr> "17021 NE 113TH CT", "11927 178TH PL NE",
"13~
## $ zip5                    <dbl> 98052, 98052, 98052, 98052, 98052, 98053,
980~
## $ ctyname                 <chr> "REDMOND", "REDMOND", NA, "REDMOND",
"REDMOND~
## $ postalctyn              <chr> "REDMOND", "REDMOND", "REDMOND",
"REDMOND", "~
## $ lon                     <dbl> -122.1124, -122.1022, -122.1085, -
122.1037, -~
## $ lat                     <dbl> 47.70139, 47.70731, 47.71986, 47.63914,
47.69~
## $ building_grade          <dbl> 9, 9, 8, 8, 7, 7, 10, 10, 9, 8, 9, 8, 8,
9, 1~
## $ square_feet_total_living <dbl> 2810, 2880, 2770, 1620, 1440, 4160, 3960,
372~
## $ bedrooms                <dbl> 4, 4, 4, 3, 3, 4, 5, 4, 4, 4, 3, 3, 4, 3,
3, ~
## $ bath_full_count         <dbl> 2, 2, 1, 1, 1, 2, 3, 2, 2, 1, 2, 2, 1, 2,
2, ~
## $ bath_half_count         <dbl> 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0,
1, ~
## $ bath_3qtr_count         <dbl> 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0,
0, ~
## $ year_built              <dbl> 2003, 2006, 1987, 1968, 1980, 2005, 1993,
198~
```

```
## $ year_renovated           <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, ~
## $ current_zoning           <chr> "R4", "R4", "R6", "R4", "R6", "URPSO",
"RA5",~
## $ sq_ft_lot                <dbl> 6635, 5570, 8444, 9600, 7526, 7280,
97574, 30~
## $ prop_type                <chr> "R", "R", "R", "R", "R", "R", "R", "R",
"R", ~
## $ present_use              <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, ~
```

```
## Check for nulls in all rows
apply(housingdata_df, 2, function(i) any(is.na(i)))
```

```
##                 Sale Date                Sale Price               sale_reason
##                     FALSE                     FALSE                     FALSE
##          sale_instrument             sale_warning                  sitetype
##                     FALSE                      TRUE                     FALSE
##                 addr_full                     zip5                   ctyname
##                     FALSE                     FALSE                      TRUE
##                postalctyn                       lon                       lat
##                     FALSE                     FALSE                     FALSE
##           building_grade square_feet_total_living                  bedrooms
##                     FALSE                     FALSE                     FALSE
##          bath_full_count          bath_half_count          bath_3qtr_count
##                     FALSE                     FALSE                     FALSE
##               year_built           year_renovated           current_zoning
##                     FALSE                     FALSE                     FALSE
##                 sq_ft_lot                 prop_type                present_use
##                     FALSE                     FALSE                     FALSE
```

```
## Looking at the data, there is missing data for sale_warning and ctyname

# I. Explain any transformations or modifications you made to the dataset
colnames(housingdata_df)[1] <- "Sale_Date"
colnames(housingdata_df)[2] <- "Sale_Price"
# I have Changed the column names of "Sale Date" to "Sale_Date" and "Sale
Price" to "Sale_Price" to avoid issues.

# 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.
housingdata_lm1 <- lm(formula = Sale_Price ~ sq_ft_lot, data =
housingdata_df)
housingdata_lm1
```

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

```
## lm(formula = Sale_Price ~ sq_ft_lot, data = housingdata_df)
##
## Coefficients:
## (Intercept)    sq_ft_lot
##   6.418e+05    8.510e-01
```

```
housingdata_lm2 <- lm(formula = Sale_Price ~ zip5 + bedrooms +
bath_full_count + year_built, data = housingdata_df)
housingdata_lm2
```

```
##
## Call:
## lm(formula = Sale_Price ~ zip5 + bedrooms + bath_full_count +
##     year_built, data = housingdata_df)
##
## Coefficients:
##     (Intercept)             zip5         bedrooms  bath_full_count
##      -823057085             8316            82515            93566
##      year_built
##            3963
```

```
# I have included other predictors like zip5, bedrooms, bath_full_count and
year of built as those are important key factors in home price predictions.
# 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?
summary(housingdata_lm1)
```

```
##
## Call:
## lm(formula = Sale_Price ~ sq_ft_lot, data = housingdata_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(housingdata_lm2)
```

```
##
## Call:
## lm(formula = Sale_Price ~ zip5 + bedrooms + bath_full_count +
##     year_built, data = housingdata_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2512477  -157243   -53551    63114  4087842
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -8.231e+08  1.945e+08  -4.232 2.33e-05 ***
## zip5             8.316e+03  1.984e+03   4.192 2.78e-05 ***
## bedrooms         8.252e+04  4.046e+03  20.393  < 2e-16 ***
## bath_full_count  9.357e+04  6.114e+03  15.302  < 2e-16 ***
## year_built       3.963e+03  2.198e+02  18.032  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 378100 on 12860 degrees of freedom
## Multiple R-squared:  0.1263, Adjusted R-squared:  0.126
## F-statistic: 464.6 on 4 and 12860 DF,  p-value: < 2.2e-16

# R2 for housingdata_lm1: 0.01435  adjusted: 0.01428
# R2 for housingdata_lm2: 0.11263 adjusted: 0.1126
# R-Squared is a statistical measure of fit for the model.
# These low R-Squared values mean that the model is not a good fit for
analysis.
# The multiple regression seems better, but not ideal for the given data set.
# 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?
coef_lmbeta <- lm.beta(housingdata_lm2)
coef_lmbeta

##
## Call:
## lm(formula = Sale_Price ~ zip5 + bedrooms + bath_full_count +
##     year_built, data = housingdata_df)
##
## Standardized Coefficients::
##     (Intercept)            zip5        bedrooms bath_full_count
year_built
##      0.00000000      0.03485771      0.17877693      0.15058250
0.16877309

# zip5 (standardized β = 0.03485771) - This value indicates that as zip code
increase by 1 standard deviation,
# the sales price increase by 0.03485771 standard deviation.
```

```
# bedrooms (standardized β = 0.17877693) -This value indicates that as
bedrooms increase by 1 standard deviation,
# the sales price of the house increase by 0.17877693 standard deviation.
# bath_full_count (standardized β =  0.15058250) -This value indicates that
as full bath room  increase by 1 standard deviation,
# the sales price of the house increase by  0.15058250 standard deviation.
# year_built(standardized β = 0.16877309 ) - This value indicates that as
year_# built increase by 1 standard deviation,
# the sales price increase by 0.16877309 standard deviation.

# V. Calculate the confidence intervals for the parameters in your model and
#     explain what the results indicate.
confint(housingdata_lm2)
```

```
##                         2.5 %        97.5 %
## (Intercept)     -1.204277e+09 -4.418367e+08
## zip5             4.427196e+03  1.220396e+04
## bedrooms         7.458418e+04  9.044667e+04
## bath_full_count  8.158105e+04  1.055518e+05
## year_built       3.532462e+03  4.394133e+03
```

```
# In the selected model, the predictor (year_built) have very tight
confidence intervals,
# which describes that the estimates for the current model are likely
# to be representative of the true population.
# The confidence interval for (zip5, bedrooms, bath_full_count) is wider but
still does not cross zero,
# indicating that the parameter for this variable is less representative of
the population,
# but are still significant for the selected model.

# 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.
anova(housingdata_lm1,housingdata_lm2)
```

```
## Analysis of Variance Table
##
## Model 1: Sale_Price ~ sq_ft_lot
## Model 2: Sale_Price ~ zip5 + bedrooms + bath_full_count + year_built
##   Res.Df        RSS Df  Sum of Sq      F    Pr(>F)
## 1  12863 2.0734e+15
## 2  12860 1.8380e+15  3 2.3539e+14 548.99 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
## The p value is very small value indeed,
## we can say that housingdata_lm2 significantly improved
## the fit of the model to the data compared to housingdata_lm1
# 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.
housingdata_df$residuals<-resid(housingdata_lm2)
housingdata_df$standardized.residuals<- rstandard(housingdata_lm2)
housingdata_df$studentized.residuals<-rstudent(housingdata_lm2)
housingdata_df$cooks.distance<-cooks.distance(housingdata_lm2)
housingdata_df$dfbeta<-dfbeta(housingdata_lm2)
housingdata_df$dffit<-dffits(housingdata_lm2)
housingdata_df$leverage<-hatvalues(housingdata_lm2)
housingdata_df$covariance.ratios<-covratio(housingdata_lm2)

head(housingdata_df)
```

```
## # A tibble: 6 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
## # ... with 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>, studentized.residuals <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.
housingdata_df$large.residual <-housingdata_df$standardized.residuals >2 |
housingdata_df$standardized.residuals < -2
head(housingdata_df$large.residual)
```

```
##     1     2     3     4     5     6
## FALSE FALSE FALSE FALSE FALSE FALSE
```

```
# IX. Use the appropriate function to show the sum of large residuals.
sum(housingdata_df$large.residual)
```

```
## [1] 342
```

```
# X. Which specific variables have large residuals (only cases that evaluate
as TRUE)?
housingdata_df[housingdata_df$large.residual,c("Sale_Price", "zip5",
"bedrooms", "bath_full_count", "year_built", "standardized.residuals")]
```

```
## # A tibble: 342 x 6
##    Sale_Price  zip5 bedrooms bath_full_count year_built
standardized.residuals
##         <dbl> <dbl>    <dbl>           <dbl>      <dbl>
<dbl>
##  1    1900000 98053        4               3       1990
2.89
##  2    1520000 98052        5               2       1952
2.33
##  3    1390000 98053        0               1       1955
3.28
##  4    1588359 98053        2               2       2005
2.59
##  5    1450000 98052        3               2       1972
2.37
##  6    1450000 98052        2               1       1918
3.41
##  7     270000 98053        4              23       2016                    -
7.05
##  8    2500000 98053        4               2       2005
4.57
##  9    2169000 98053        4               3       2005
3.44
## 10    1534000 98052        4               1       1963
2.72
## # ... with 332 more rows

# XI. Investigate further by calculating the
#    leverage,
#    cooks distance,
#    and covariance ratios.
# Comment on all cases that are problematics.
housingdata_df[housingdata_df$large.residual , c("cooks.distance",
"leverage", "covariance.ratios")]

## # A tibble: 342 x 3
##    cooks.distance leverage covariance.ratios
##             <dbl>    <dbl>             <dbl>
##  1       0.000744 0.000446             0.998
##  2       0.000919 0.000843             0.999
##  3       0.00381  0.00177              0.998
##  4       0.000494 0.000368             0.998
##  5       0.000381 0.000338             0.999
##  6       0.00449  0.00193              0.998
##  7       1.26     0.112                1.11
##  8       0.000627 0.000150             0.992
##  9       0.000819 0.000345             0.996
## 10       0.000572 0.000386             0.998
## # ... with 332 more rows
```

```r
# Except one (1.26)remaining of the values has a Cook's distance less than 1
,
# The leverage values is very low.
# XII. Perform the necessary calculations to assess the assumption of
independence
#    and state if the condition is met or not.
durbinWatsonTest(housingdata_lm2)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.6360363     0.7279246       0
##  Alternative hypothesis: rho != 0
```

```r
## The test statistic is 0.7442029 and 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.
## Value less than 1 suggests that the assumption might not been met.
# XIII. Perform the necessary calculations to assess the assumption of no
multicollinearity
#    and state if the condition is met or not.
vif(housingdata_lm2)
```

```
##            zip5        bedrooms bath_full_count      year_built
##        1.017721        1.131133        1.425250        1.289420
```

```r
## tolerance statistics
1/vif(housingdata_lm2)
```

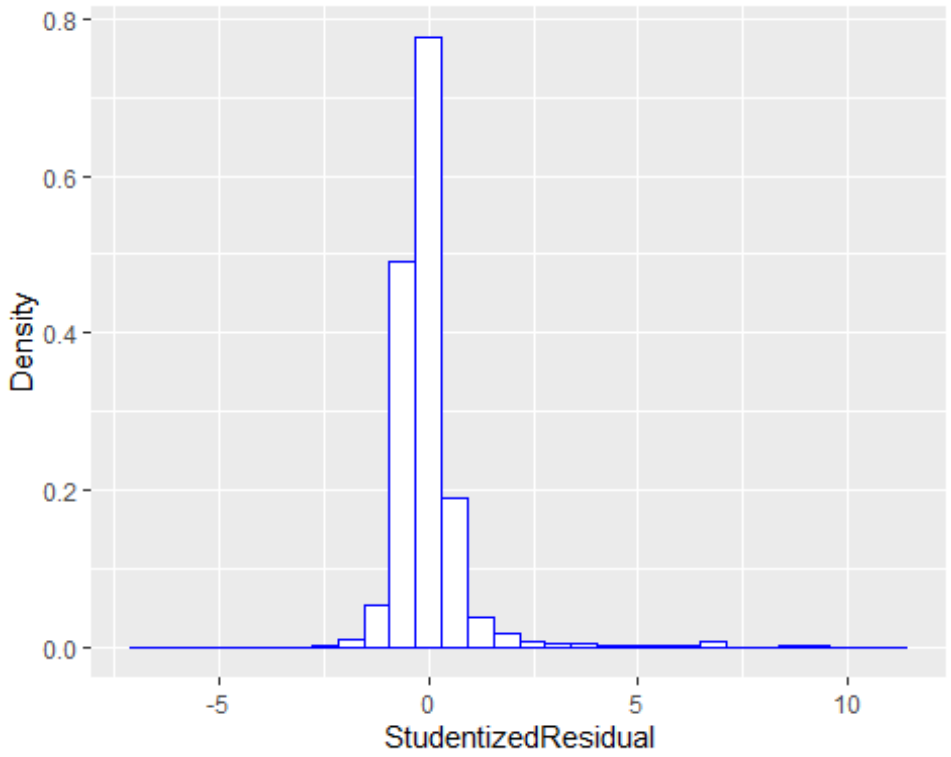
```
##            zip5        bedrooms bath_full_count      year_built
##       0.9825871       0.8840696       0.7016315       0.7755422
```

```r
mean(vif(housingdata_lm2))
```

```
## [1] 1.215881
```

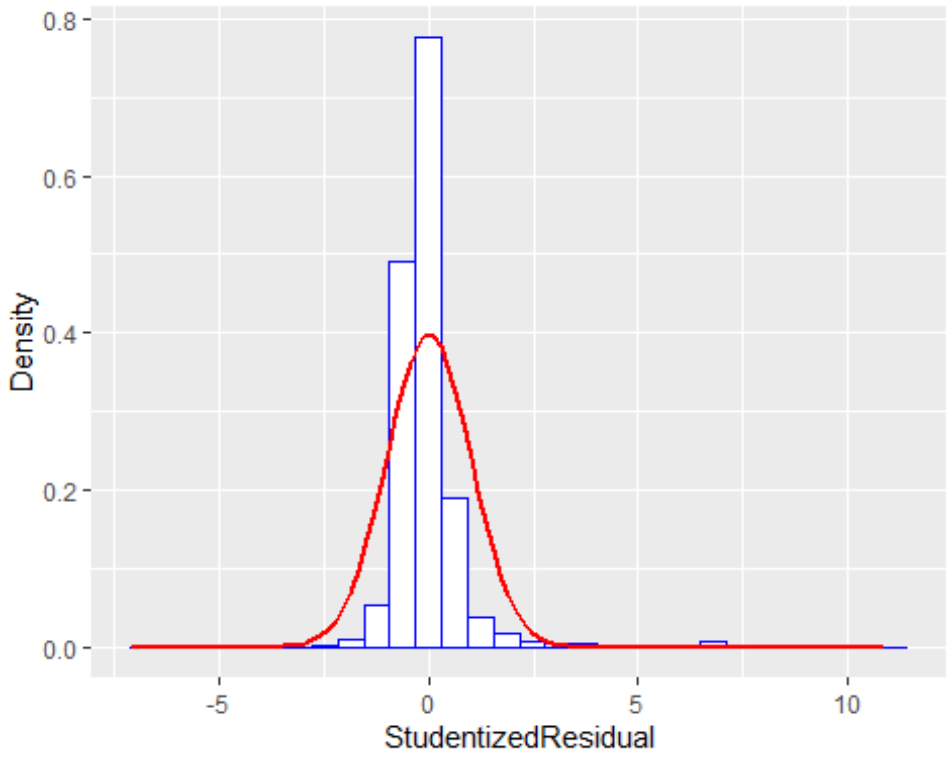
```r
## VIF values are all below 10 and the tolerance statistics above 0.2.
## Also, the mean VIF is ~ 1.
## Based on these results we can conclude that there is no collinearity in
data.
# XIV. Visually check the assumptions related to the residuals using the
plot() and hist() functions.
#     Summarize what each graph is informing you of and if any anomalies are
present.
housingdata_df$fitted <- housingdata_lm2$fitted.values
histogram<-ggplot(housingdata_df, aes(studentized.residuals)) +
  geom_histogram(aes(y = ..density..), colour = "blue", fill = "white") +
  labs(x = "StudentizedResidual", y = "Density")
histogram
```

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



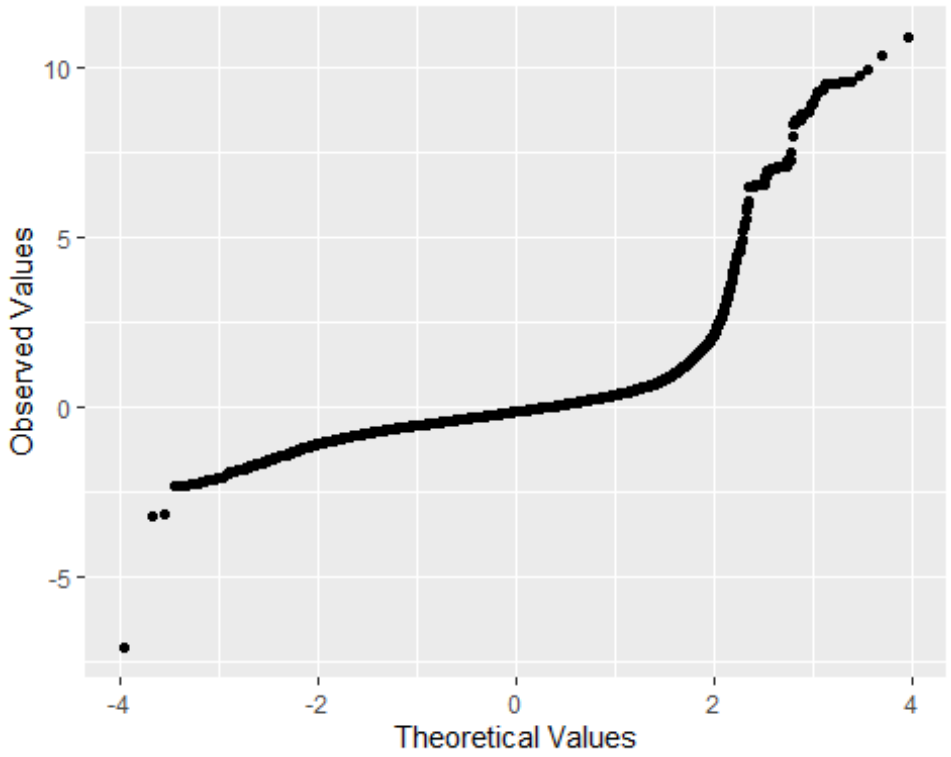

```
histogram + stat_function(fun = dnorm, args = list(mean =
mean(housingdata_df$studentized.residuals, na.rm = TRUE),
  sd = sd(housingdata_df$studentized.residuals,na.rm = TRUE)), colour= "red",
size = 1)

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




```
qplot(sample = housingdata_df$studentized.residuals, stat="qq") + labs(x
="Theoretical Values", y = "Observed Values")

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




```
# The distribution is briefly normal.
# To summarize, the model appears to be accurate for the sample and can be
generalized to the population.
# 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?
# Based on vif score/values calculated above, since the values are not close
to 5, the predictors doesn't have
# any significant multi-collinearity.
# Mean vif is also just above 1 but no where near 5.
# so, Model does not appear to be biased.
```