Assignment 8.2

Lincoln Brown

7/30/2021

```
library(readxl)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.2     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(stringr)
library(dplyr)
housing <- read_excel("/media/x/disk/School/DSC520/datasets/week-6-housing.xlsx")
```

```
names <- colnames(housing)
print(names)
```

```
##  [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

```
#Remove unnecessary columns
#Removing all columns that contain data that are useless without the codebook.
# I am removing the ctyname column because the zip5 column is better populated
remove_cols <- c(3,4,5,6,9,10,13,21,23)
refined <- housing |> dplyr::select(-all_of(remove_cols))
dim(refined)
```

```
## [1] 12865    15
```

```
library(broom)
library(scales)
```

```
##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:purrr':
##
##     discard
```

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
library(coefplot)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(QuantPsyc)
```

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

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked _by_ '.GlobalEnv':
##
##     housing
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
##
## Attaching package: 'QuantPsyc'
```

```
## The following object is masked from 'package:base':
##
##     norm
```

```
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
```

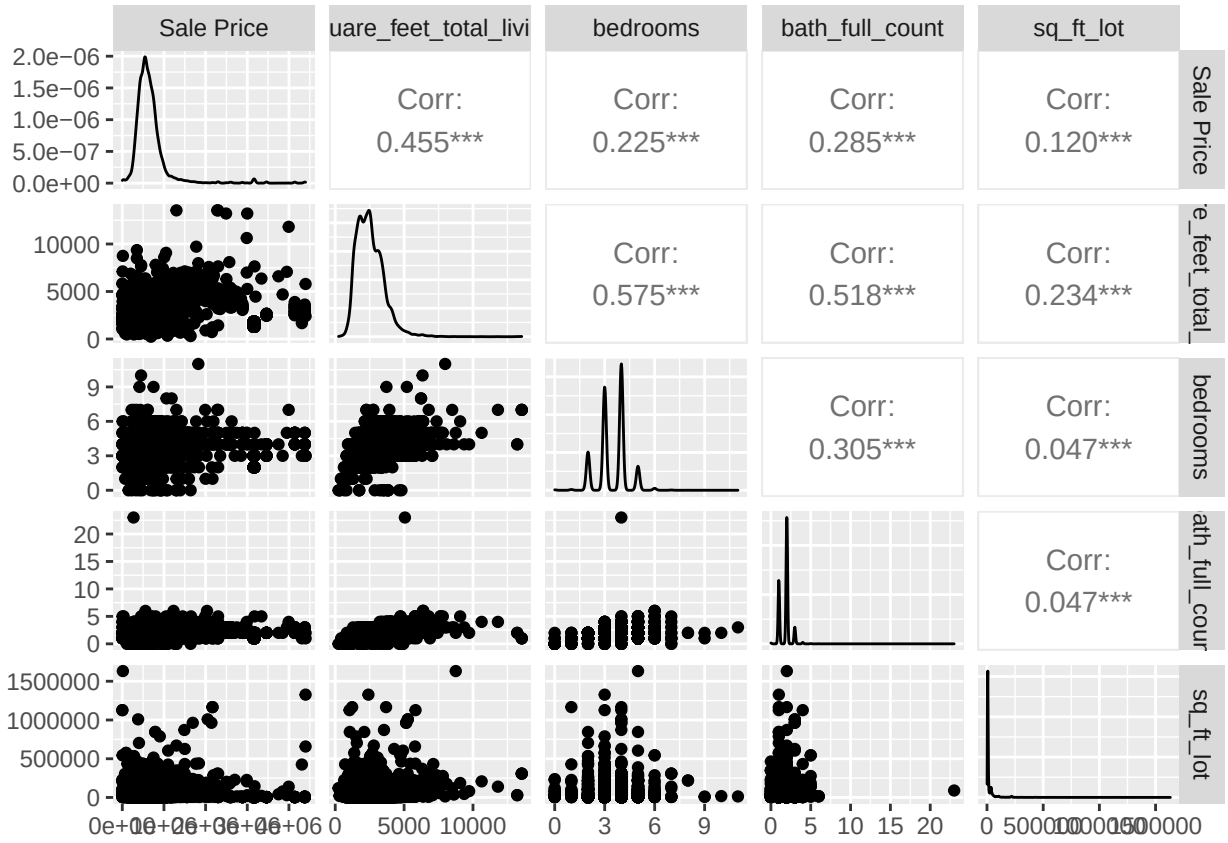
```
avg_zip <- aggregate(`Sale Price` ~ zip5, housing, mean)
cost_size <- housing |> dplyr::select(c(2,14,22))
housing_size <- plyr::ddply(refined, plyr::.(square_feet_total_living), transform,
                      house.size = cut(square_feet_total_living, breaks = c(-Inf, 1000, 2000, 3000, Inf
                                       labels = c("Tiny", "Small", "Medium", "Large"))
)
colnames(refined)
```

```
##  [1] "Sale Date"                "Sale Price"
##  [3] "addr_full"                "zip5"
##  [5] "lon"                      "lat"
##  [7] "square_feet_total_living" "bedrooms"
##  [9] "bath_full_count"          "bath_half_count"
## [11] "bath_3qtr_count"          "year_built"
## [13] "year_renovated"           "sq_ft_lot"
## [15] "present_use"
```

```
colnames(housing_size)
```

```
##  [1] "Sale.Date"                "Sale.Price"
##  [3] "addr_full"                "zip5"
##  [5] "lon"                      "lat"
##  [7] "square_feet_total_living" "bedrooms"
##  [9] "bath_full_count"          "bath_half_count"
## [11] "bath_3qtr_count"          "year_built"
## [13] "year_renovated"           "sq_ft_lot"
## [15] "present_use"              "house.size"
```

```
ggpairs(data=refined, columns=c(2,7,8,9,14))
```



We can see from this that the square_feet_total living has the strongest correlation wtih sale price.

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.

In addition to the square footage of the lot, I also want to compare the square footage of the house.

The number of bedrooms and bathrooms is also a significant factor in the cost of a house, so I would like to test for that.

```
colnames(refined)
```

```
##  [1] "Sale Date"                "Sale Price"
##  [3] "addr_full"                "zip5"
##  [5] "lon"                      "lat"
##  [7] "square_feet_total_living" "bedrooms"
##  [9] "bath_full_count"          "bath_half_count"
## [11] "bath_3qtr_count"          "year_built"
## [13] "year_renovated"           "sq_ft_lot"
## [15] "present_use"
```

```
sale_price <- refined$`Sale Price`
sqft <- refined$sq_ft_lot

price_lm <- lm(`Sale Price` ~ sq_ft_lot, refined)
testdf <- refined
testdf$zip5 <- as.character(testdf$zip5)
```

```
sapply(testdf, class)
```

```
## $`Sale Date`
## [1] "POSIXct" "POSIXt"
##
## $`Sale Price`
## [1] "numeric"
##
## $addr_full
## [1] "character"
##
## $zip5
## [1] "character"
##
## $lon
## [1] "numeric"
##
## $lat
## [1] "numeric"
##
## $square_feet_total_living
## [1] "numeric"
##
## $bedrooms
## [1] "numeric"
##
## $bath_full_count
## [1] "numeric"
##
## $bath_half_count
## [1] "numeric"
##
## $bath_3qtr_count
## [1] "numeric"
##
## $year_built
## [1] "numeric"
##
## $year_renovated
## [1] "numeric"
##
## $sq_ft_lot
## [1] "numeric"
##
## $present_use
## [1] "numeric"
```

```
zip_lm <- lm(formula=`Sale Price` ~ zip5 +0, data=testdf)
multi_lm <- lm(formula=`Sale Price` ~ sq_ft_lot + bath_full_count + bedrooms, data=refined)
multisq_lm <- lm(formula=`Sale Price` ~ sq_ft_lot + bath_full_count + square_feet_total_living + bedroo
sqft_lm <- lm(formula=`Sale Price` ~ square_feet_total_living, data=refined)
sqftmulti_lm <- lm(formula=`Sale Price` ~ square_feet_total_living + bath_full_count + bedrooms, data=r

(price_lm)
```

```
##
## Call:
## lm(formula = `Sale Price` ~ sq_ft_lot, data = refined)
##
## Coefficients:
## (Intercept)    sq_ft_lot
##   6.418e+05    8.510e-01
```

```
(zip_lm)
```

```
##
## Call:
## lm(formula = `Sale Price` ~ zip5 + 0, data = testdf)
##
## Coefficients:
## zip598052  zip598053  zip598059  zip598074
##    649375     672624     645000     951544
```

```
(multi_lm)
```

```
##
## Call:
## lm(formula = `Sale Price` ~ sq_ft_lot + bath_full_count + bedrooms,
##     data = refined)
##
## Coefficients:
##     (Intercept)        sq_ft_lot  bath_full_count         bedrooms
##       1.429e+05        7.227e-01        1.458e+05        6.887e+04
```

```
(sqft_lm)
```

```
##
## Call:
## lm(formula = `Sale Price` ~ square_feet_total_living, data = refined)
##
## Coefficients:
##              (Intercept)  square_feet_total_living
##                 189106.6                    185.7
```

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?

```
summary(price_lm)
```

```
##
## Call:
## lm(formula = `Sale Price` ~ sq_ft_lot, data = refined)
##
## 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(multi_lm)
```

```
##
## Call:
## lm(formula = `Sale Price` ~ sq_ft_lot + bath_full_count + bedrooms,
##     data = refined)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3566287  -153218   -51375    69545  3736381
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     1.429e+05  1.481e+04   9.649   <2e-16 ***
## sq_ft_lot       7.227e-01  5.910e-02  12.229   <2e-16 ***
## bath_full_count 1.458e+05  5.422e+03  26.888   <2e-16 ***
## bedrooms        6.887e+04  4.027e+03  17.099   <2e-16 ***
## ---
## 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: < 2.2e-16
```

```
summary(multisq_lm)
```

```
##
## Call:
## lm(formula = `Sale Price` ~ sq_ft_lot + bath_full_count + square_feet_total_living +
##     bedrooms, data = refined)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1917207  -117118   -40819    44256  3789150
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               2.031e+05  1.404e+04  14.471  < 2e-16 ***
## sq_ft_lot                 1.048e-01  5.776e-02   1.814   0.0697 .
## bath_full_count           4.323e+04  5.708e+03   7.574 3.87e-14 ***
## square_feet_total_living  1.820e+02  4.515e+00  40.307  < 2e-16 ***
## bedrooms                 -2.432e+04  4.444e+03  -5.473 4.50e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 359000 on 12860 degrees of freedom
```

```
## Multiple R-squared:  0.2122, Adjusted R-squared:  0.2119
## F-statistic: 865.9 on 4 and 12860 DF,  p-value: < 2.2e-16
```

```
summary(sqft_lm)
```

```
##
## Call:
## lm(formula = `Sale Price` ~ square_feet_total_living, data = refined)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1800136  -120257   -41547    44028  3811745
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               1.891e+05  8.745e+03   21.62   <2e-16 ***
## square_feet_total_living 1.857e+02  3.208e+00   57.88   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 360200 on 12863 degrees of freedom
## Multiple R-squared:  0.2066, Adjusted R-squared:  0.2066
## F-statistic:  3351 on 1 and 12863 DF,  p-value: < 2.2e-16
```

```
summary(sqftmulti_lm)
```

```
##
## Call:
## lm(formula = `Sale Price` ~ square_feet_total_living + bath_full_count +
##     bedrooms, data = refined)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1760583  -117559   -41529    43918  3832099
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)               204679.013  14013.468  14.606  < 2e-16 ***
## square_feet_total_living    184.150      4.353  42.302  < 2e-16 ***
## bath_full_count           42309.808   5685.497   7.442 1.06e-13 ***
## bedrooms                 -25206.328   4417.404  -5.706 1.18e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 359000 on 12861 degrees of freedom
## Multiple R-squared:  0.212,  Adjusted R-squared:  0.2118
## F-statistic:  1153 on 3 and 12861 DF,  p-value: < 2.2e-16
```

```
models <- list(price_lm, multi_lm, multisq_lm, sqft_lm, sqftmulti_lm)
model_names <- c("price_lm", "multi_lm", "multisq_lm", "sqft_lm", "sqftmulti_lm")
glanced <- sapply(models, glance)
colnames(glanced) <- model_names
(glanced["r.squared",])
```

```
## $price_lm
## [1] 0.01435497
```

```
##
## $multi_lm
## [1] 0.1126625
##
## $multisq_lm
## [1] 0.2121903
##
## $sqft_lm
## [1] 0.2066499
##
## $sqftmulti_lm
## [1] 0.2119887
```

```
(glanced["adj.r.squared",])
```

```
## $price_lm
## [1] 0.01427835
##
## $multi_lm
## [1] 0.1124555
##
## $multisq_lm
## [1] 0.2119453
##
## $sqft_lm
## [1] 0.2065882
##
## $sqftmulti_lm
## [1] 0.2118049
```

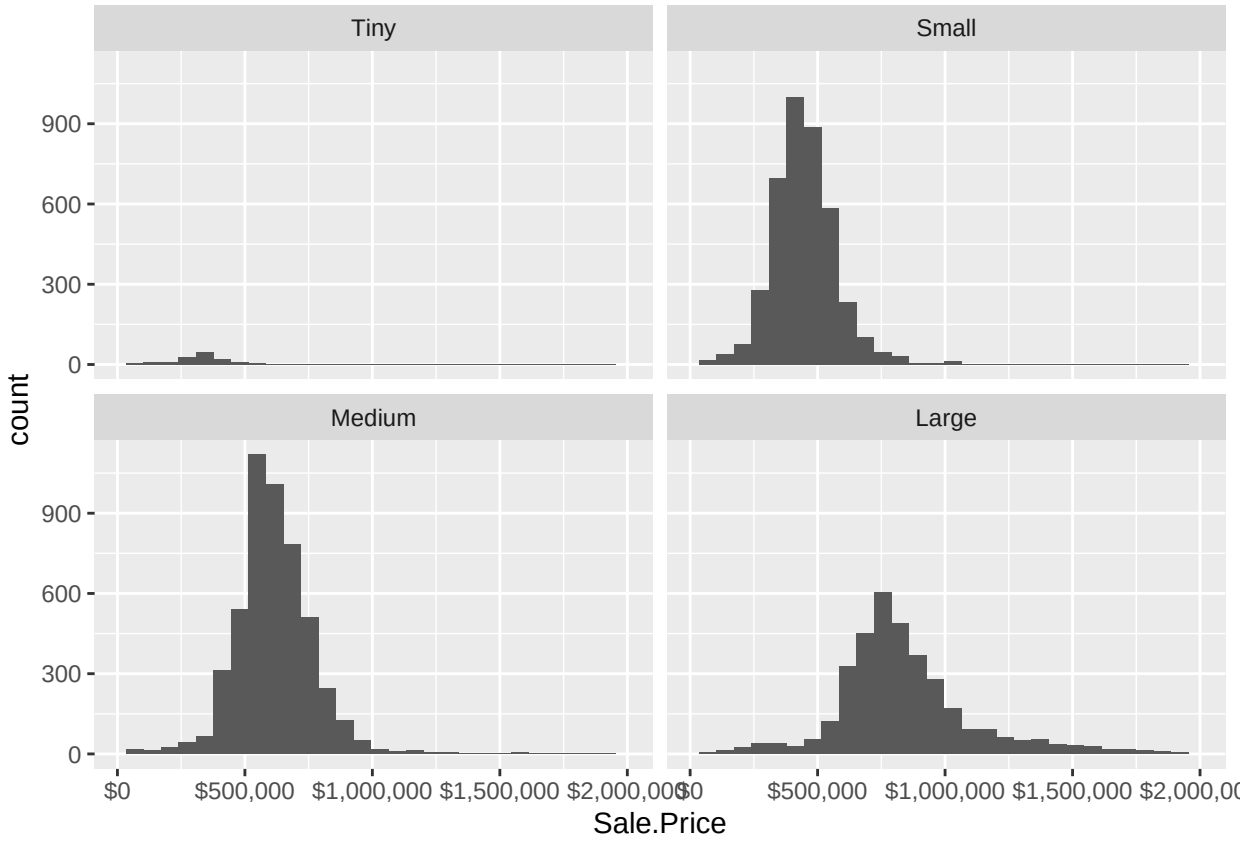
```
price_summary <- housing_size |> group_by(house.size) |> summarize(mean_by_size = mean(Sale.Price))
price_summary
```

```
## # A tibble: 4 x 2
##   house.size mean_by_size
##   <fct>             <dbl>
## 1 Tiny            432586.
## 2 Small           495161.
## 3 Medium          636017.
## 4 Large           890808.
```

```
ggplot(housing_size, aes(`Sale.Price`)) +
  geom_histogram(bins=30) +
  facet_wrap(vars(house.size)) +
  scale_x_continuous(labels=dollar, limits=c(10000, 2000000))
```

```
## Warning: Removed 228 rows containing non-finite values (stat_bin).
```

```
## Warning: Removed 8 rows containing missing values (geom_bar).
```



From r2 and adjusted r2 we can see that none of the models perform particularly well, but there are some that perform better than others. The highest performing model was multisq_lm which uses multiple predictor variables, including both the square footage of the lot and the square footage of the living space. Out of curiosity, I decided to compare the performance of sq_ft_lot and square_feet_total_living to Sale Price individually to see which was a better predictor.

I discovered that square_feet_total_living (sqft_lm) performs significantly better than sq_ft_lot (price_lm). With r^2 of .014 for price_lm and r^2 of .206 for sqft_lm

From this, I deduced that the square_feet_total_living is actually the best predictor, but the model performs best when both variables are included.

With multisq_lm performing the best with an r^2 of .2121 and adj r^2 of .2119.

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?

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

```
##
## Attaching package: 'lm.beta'
```

```
## The following object is masked from 'package:QuantPsyc':
##
##     lm.beta
```

```
sapply(models, lm.beta)
```

```
##                           [,1]
```

```
## coefficients              numeric,2
## residuals                 numeric,12865
## effects                   numeric,12865
## rank                      2
## fitted.values             numeric,12865
## assign                    integer,2
## qr                        qr,5
## df.residual               12863
## xlevels                   list,0
## call                      expression
## terms                     `Sale Price` ~ sq_ft_lot
## model                     data.frame,2
## standardized.coefficients numeric,2
##                           [,2]
## coefficients              numeric,4
## residuals                 numeric,12865
## effects                   numeric,12865
## rank                      4
## fitted.values             numeric,12865
## assign                    integer,4
## qr                        qr,5
## df.residual               12861
## xlevels                   list,0
## call                      expression
## terms                     `Sale Price` ~ sq_ft_lot + bath_full_count + bedrooms
## model                     data.frame,4
## standardized.coefficients numeric,4
##                           [,3]          [,4]
## coefficients              numeric,5     numeric,2
## residuals                 numeric,12865 numeric,12865
## effects                   numeric,12865 numeric,12865
## rank                      5             2
## fitted.values             numeric,12865 numeric,12865
## assign                    integer,5     integer,2
## qr                        qr,5          qr,5
## df.residual               12860         12863
## xlevels                   list,0        list,0
## call                      expression    expression
## terms                     terms,3       `Sale Price` ~ square_feet_total_living
## model                     data.frame,5  data.frame,2
## standardized.coefficients numeric,5     numeric,2
##                           [,5]
## coefficients              numeric,4
## residuals                 numeric,12865
## effects                   numeric,12865
## rank                      4
## fitted.values             numeric,12865
## assign                    integer,4
## qr                        qr,5
## df.residual               12861
## xlevels                   list,0
## call                      expression
## terms                     `Sale Price` ~ square_feet_total_living + bath_full_count + bedrooms
## model                     data.frame,4
```

```
## standardized.coefficients numeric,4
```

```
print("Multisq_lm betas")
```

```
## [1] "Multisq_lm betas"
```

```
print(lm.beta(multisq_lm))
```

```
##
## Call:
## lm(formula = `Sale Price` ~ sq_ft_lot + bath_full_count + square_feet_total_living +
##     bedrooms, data = refined)
##
## Standardized Coefficients::
##              (Intercept)                sq_ft_lot          bath_full_count
##               0.00000000               0.01475150               0.06957036
## square_feet_total_living                 bedrooms
##               0.44543067              -0.05269668
```

The standardized betas are the coefficients printed above. The values indicate that for ever unit increase in sale price, there is an expected .014 increase in sq_ft_lot, .07 in bath_full_count, .45 in square_feet_total_living, and bedrooms reacts negatively. Bedrooms acts negatively because the number of bedrooms in a house plateaus eventually, but the price of housing does not.

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

```
confint(multisq_lm)
```

```
##                                 2.5 %        97.5 %
## (Intercept)               1.756271e+05  2.306594e+05
## sq_ft_lot                -8.439713e-03  2.179912e-01
## bath_full_count           3.204091e+04  5.441602e+04
## square_feet_total_living  1.731271e+02  1.908263e+02
## bedrooms                 -3.303294e+04 -1.561191e+04
```

```
min(refined$`Sale Price`)
```

```
## [1] 698
```

```
max(refined$`Sale Price`)
```

```
## [1] 4400000
```

With sq_ft_lot crossing zero, this appears to be a bad model. We would expect the true value of b to lie within these intervals. Square_feet_total_living has the best margin of error, indicating that it is the best predictor in this model.

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(price_lm, multisq_lm)
```

```
## Analysis of Variance Table
##
## Model 1: `Sale Price` ~ sq_ft_lot
## Model 2: `Sale Price` ~ sq_ft_lot + bath_full_count + square_feet_total_living +
```

```
##     bedrooms
##   Res.Df        RSS Df  Sum of Sq      F    Pr(>F)
## 1  12863 2.0734e+15
## 2  12860 1.6572e+15  3 4.1616e+14 1076.5 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

With an F score of 1076.5 and a p-value of 2.2e-16 (much less than 0), the changes made by this model were significant.

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.

```
residuals <- resid(multisq_lm)
stand.res <- rstandard(multisq_lm)
student.res <- rstudent(multisq_lm)
cooks.distance <- cooks.distance(multisq_lm)
dfbeta <- dfbeta(multisq_lm)
dffit <- dffits(multisq_lm)
leverage <- hatvalues(multisq_lm)
covariance.ratios <- covratio(multi_lm)

df <- data.frame(residuals, stand.res, student.res, cooks.distance, dfbeta, dffit, leverage, covariance
```

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.

```
df$large.residual <- df$stand.res > 2 | df$stand.res < -2
```

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

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

```
## [1] 322
```

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

```
colnames(df)
```

```
##  [1] "residuals"                "stand.res"
##  [3] "student.res"              "cooks.distance"
##  [5] "X.Intercept."             "sq_ft_lot"
##  [7] "bath_full_count"          "square_feet_total_living"
##  [9] "bedrooms"                 "dffit"
## [11] "leverage"                 "covariance.ratios"
## [13] "large.residual"
```

```
lrg_residuals <- df[df$large.residual, ]
```

I had 322 rows that returned as TRUE.

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

```
#Make sure that no more than five percent of cases have absolute values above 2.
nrow(lrg_residuals) / nrow(refined)
```

```
## [1] 0.02502915
```

```
#Only 2.5% of cases have absolute values above 2.
problematics <- lrg_residuals[lrg_residuals$large.residual , c("cooks.distance", "leverage", "covariance
print(nrow(problematics))
```

```
## [1] 322
```

```
any(problematics$cooks.distance>1)
```

```
## [1] FALSE
```

The cooks distance looks to be fine. Now I will check the leverage.

```
k = 5
n = nrow(refined)
avg_leverage = (k+1)/n

threshold_1 <- avg_leverage * 2
threshold_2 <- avg_leverage * 3


results_thresh_1 <- problematics[problematics$leverage > threshold_1,]
results_thresh_2 <- problematics[problematics$leverage > threshold_2,]

print(nrow(results_thresh_1))
```

```
## [1] 83
```

```
print(nrow(results_thresh_2))
```

```
## [1] 58
```

Of the 322 rows, 83 are greater than the double the average leverage and 58 are greater than the 3 times the average leverage.

Now I will check the covariance ratios.

```
upper_cvr <- 1 + threshold_2
lower_cvr <- 1- threshold_2

results_upper_cvr <- problematics[problematics$covariance.ratios > upper_cvr,]
results_lower_cvr <- problematics[problematics$covariance.ratios < lower_cvr,]

print(nrow(results_upper_cvr))
```

```
## [1] 21
```

```
print(nrow(results_lower_cvr))
```

```
## [1] 213
```

There are 21 cases that violate the upper limits There are 213 cases that violate the lower limits

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

```
dwt(multisq_lm)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.7284596     0.5430776       0
##  Alternative hypothesis: rho != 0
```

The condition is not met. The Statistic is less than one, which means it is not close to 2

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

```
print("VIF")
```

```
## [1] "VIF"
```

```
vif(multisq_lm)
```

```
##                sq_ft_lot          bath_full_count square_feet_total_living
##                 1.079454                 1.377275                 1.993497
##                 bedrooms
##                 1.513152
```

```
print("Tolerance")
```

```
## [1] "Tolerance"
```

```
1/vif(multisq_lm)
```

```
##                sq_ft_lot          bath_full_count square_feet_total_living
##                0.9263945                0.7260714                0.5016311
##                 bedrooms
##                0.6608722
```

```
print("Average VIF")
```

```
## [1] "Average VIF"
```

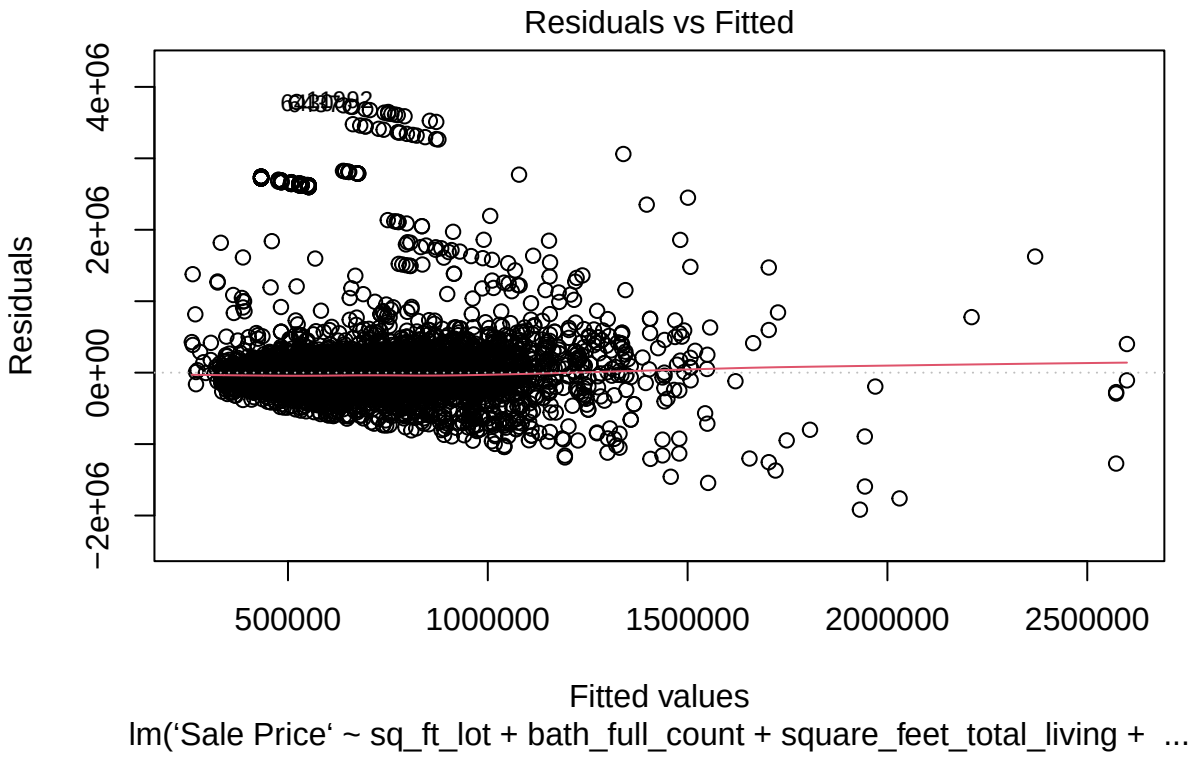
```
mean(vif(multisq_lm))
```

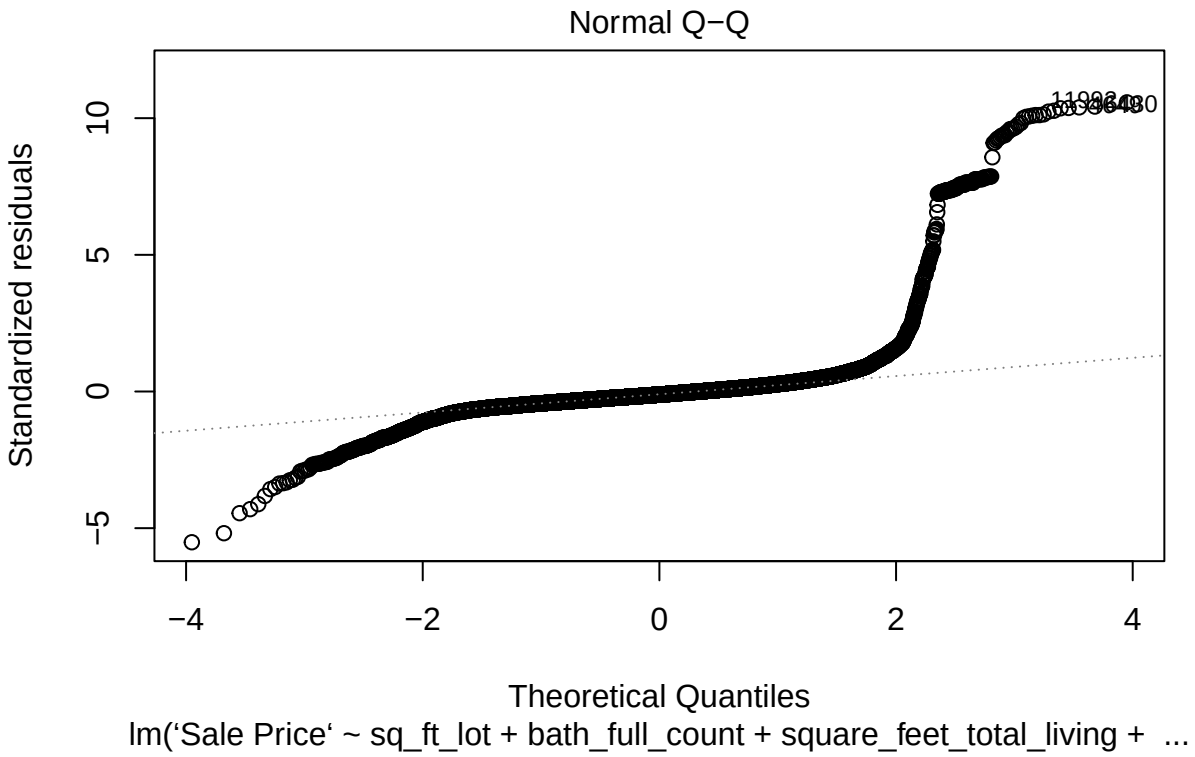
```
## [1] 1.490844
```

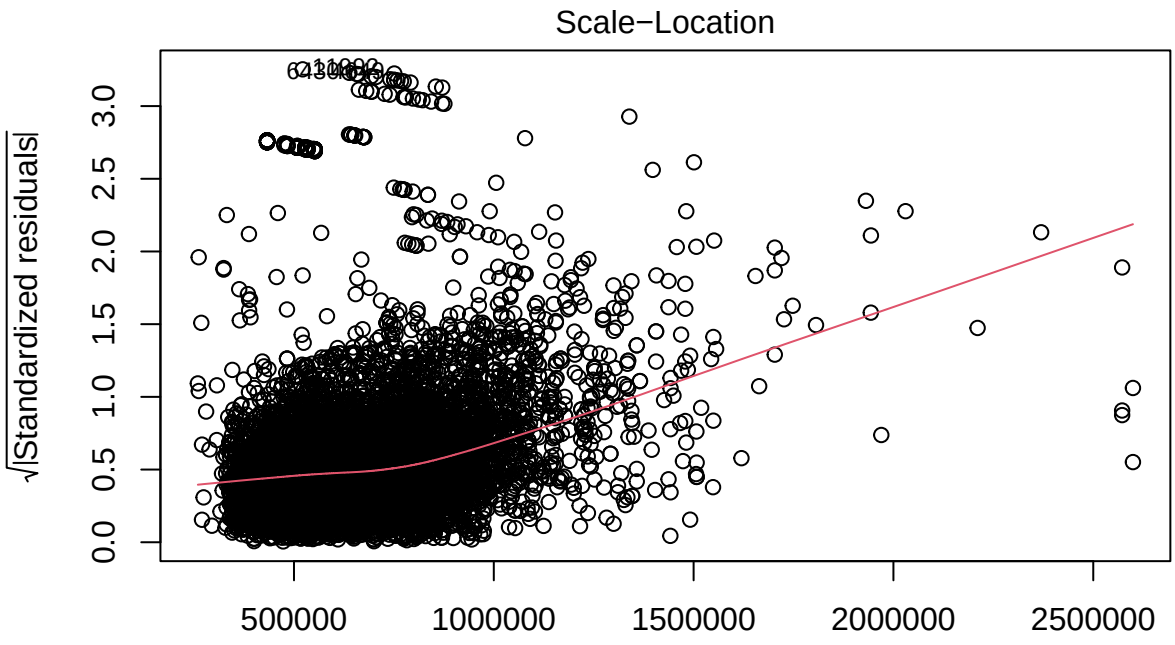
The VIF values are all below 10, the tolerance statistics are above .2 and the average is just slightly above 1, indicating there is no collinearity within our data.

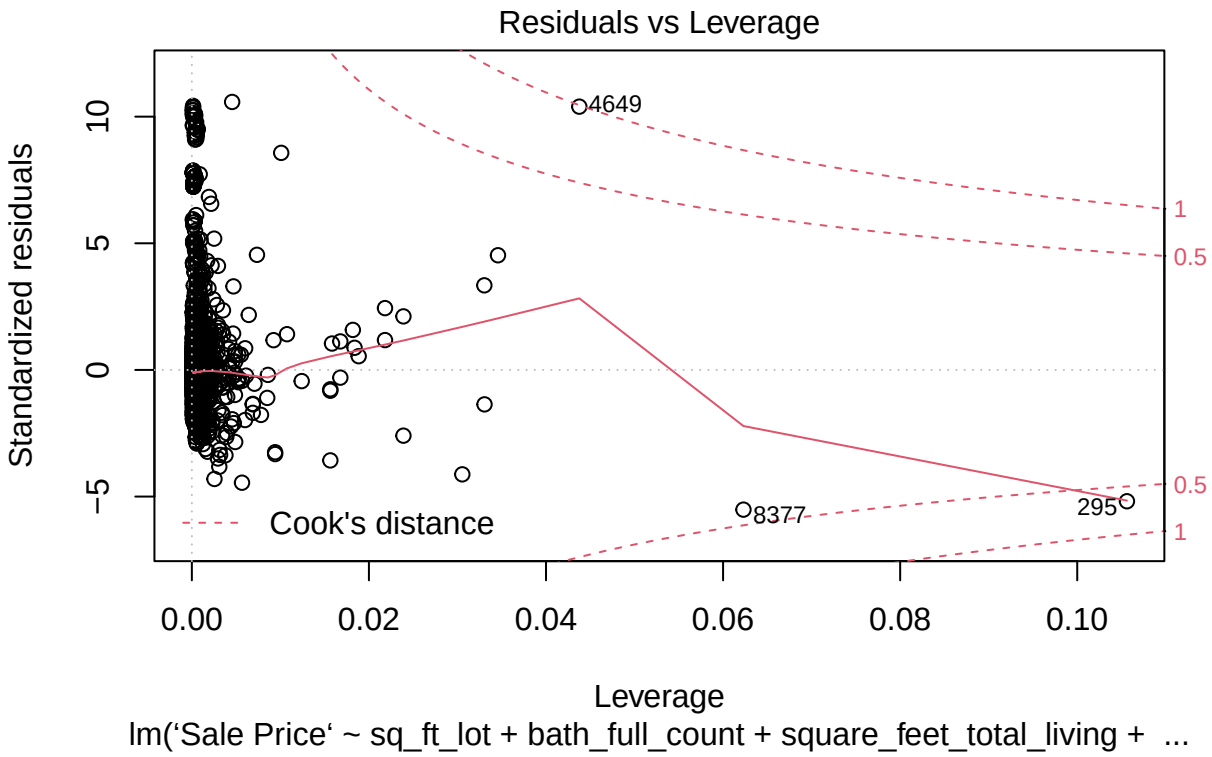
14. Visually check the assumptions related to the residuals using the plot() and hist() functions. Summarize what each graph is informing you of and if any anomalies are present.

```
plot(multisq_lm)
```

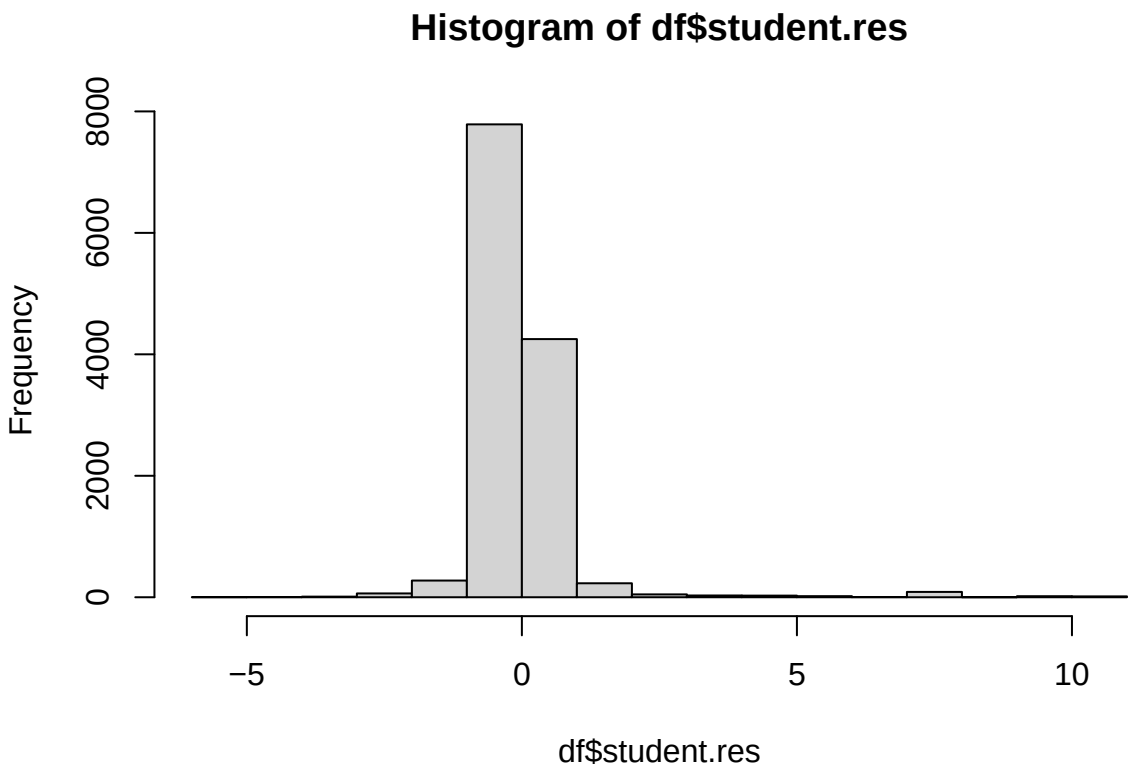

Residuals vs Fitted
Residuals
4e+06
2e+06
0e+00
−2e+06
500000
1000000
1500000
2000000
2500000
Fitted values
lm(‘Sale Price‘ ~ sq_ft_lot + bath_full_count + square_feet_total_living + ...


Normal Q−Q
Standardized residuals
10
5
0
−5
−4
−2
0
2
4
Theoretical Quantiles
lm(‘Sale Price‘ ~ sq_ft_lot + bath_full_count + square_feet_total_living + ...

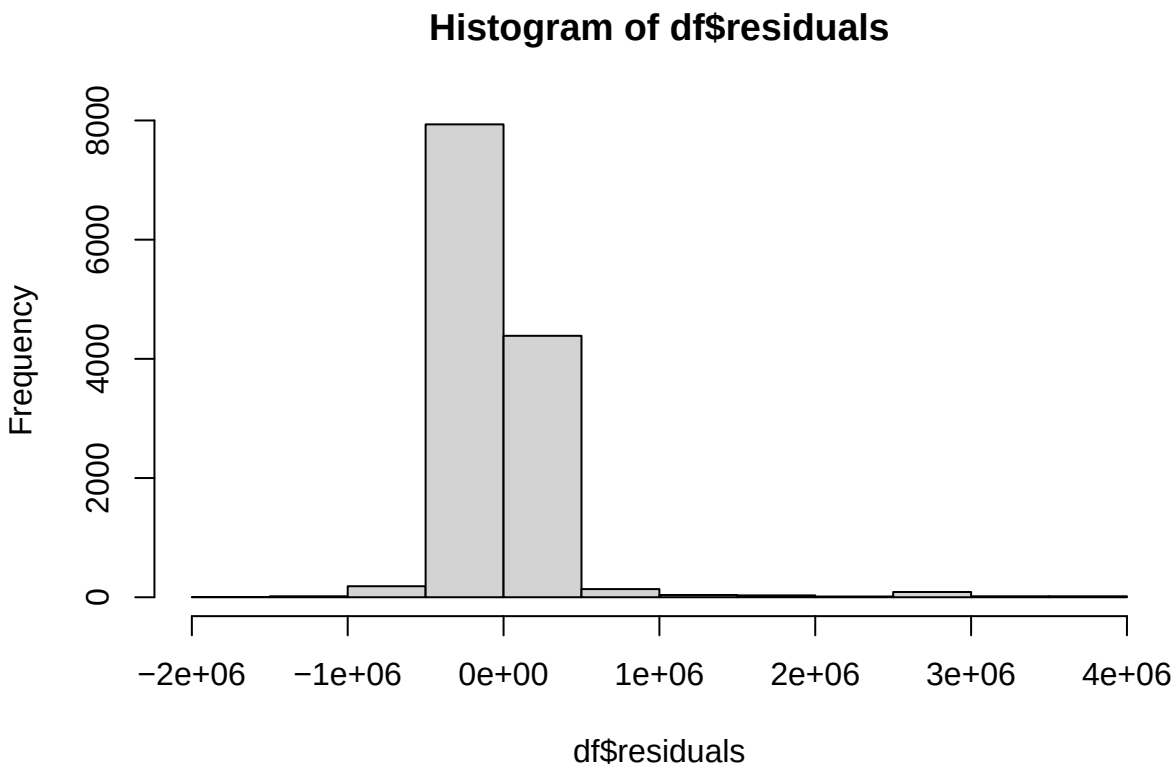

Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
3.0
500000
1000000
1500000
2000000
2500000
Fitted values
lm(‘Sale Price‘ ~ sq_ft_lot + bath_full_count + square_feet_total_living + ...




```
hist(df$student.res)
```




```
hist(df$residuals)
```



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?

Overall, I would say this model is biased. There are 322 outliers and 141 influential cases.