Weeks 8 & 9 Exercises

Stewart Wilson

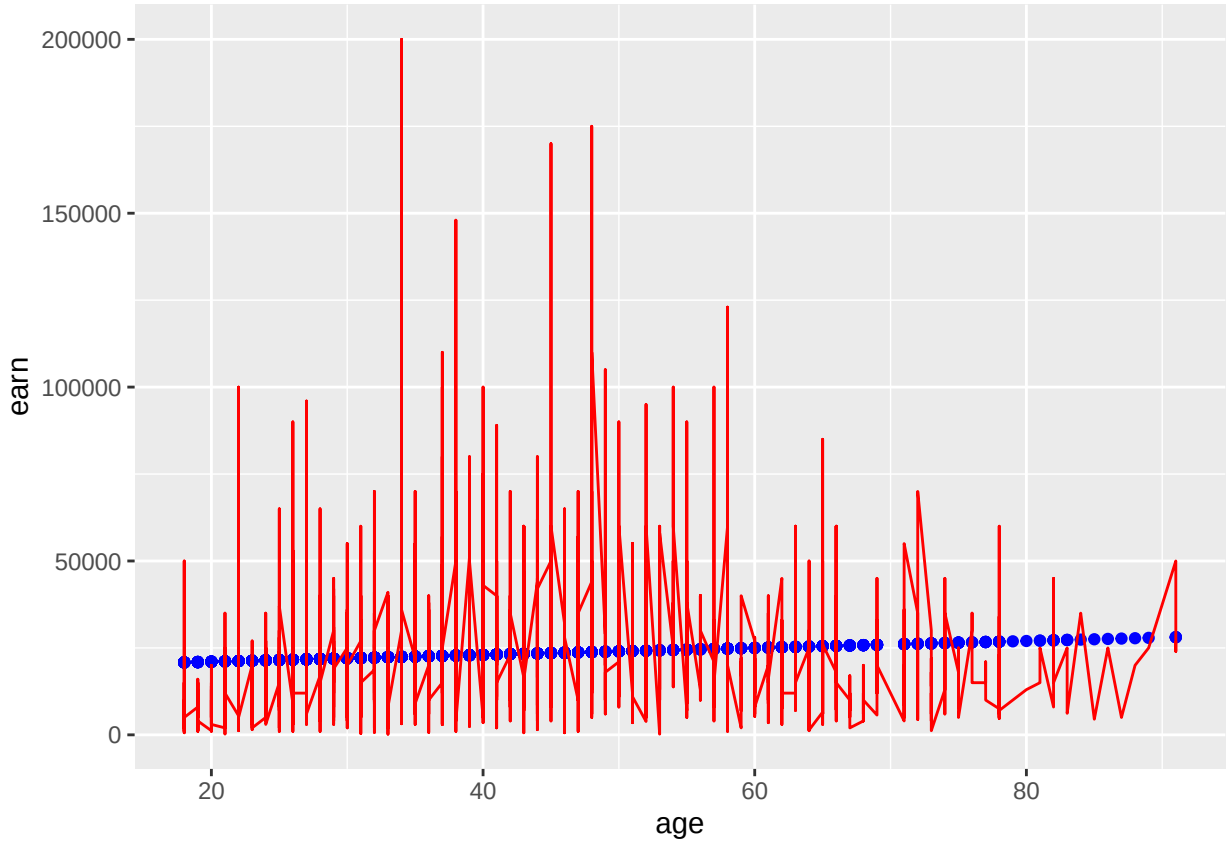
2022-05-14

Assignment 06

```
## Load the `data/r4ds/heights.csv` to
heights_df <- read.csv("C:/Users/Stewart/Documents/GitHUb/dsc520/data/r4ds/heights.csv")

## Load the ggplot2 library
library(ggplot2)

## Fit a linear model using the `age` variable as the predictor and `earn` as the outcome
age_lm <-  lm(earn ~ age, data=heights_df)

## View the summary of your model using `summary()`
summary(age_lm)
```

```
##
## Call:
## lm(formula = earn ~ age, data = heights_df)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -25098 -12622  -3667   6883 177579
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 19041.53    1571.26  12.119  < 2e-16 ***
## age            99.41      35.46   2.804  0.00514 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19420 on 1190 degrees of freedom
## Multiple R-squared:  0.006561,   Adjusted R-squared:  0.005727
## F-statistic:  7.86 on 1 and 1190 DF,  p-value: 0.005137
```

```
## Creating predictions using `predict()`
age_predict_df <- data.frame(earn = predict(age_lm, heights_df), age=heights_df$age)

## Plot the predictions against the original data
ggplot(data = age_predict_df, aes(y = earn, x = age)) +
  geom_point(color='blue') +
  geom_line(color='red',data = heights_df, aes(y=earn, x=age))
```




```
mean_earn <- mean(heights_df$earn)
## Corrected Sum of Squares Total
sst <- sum((mean_earn - heights_df$earn)^2)
## Corrected Sum of Squares for Model
ssm <- sum((mean_earn - age_predict_df$earn)^2)
## Residuals
residuals <- heights_df$earn - age_predict_df$earn
## Sum of Squares for Error
sse <- sum(residuals^2)
## R Squared R^2 = SSM\SST
r_squared <- ssm / sst

## Number of observations
n <- nrow(age_predict_df)
## Number of regression parameters
p <- 2
## Corrected Degrees of Freedom for Model (p-1)
dfm <- p - 1
## Degrees of Freedom for Error (n-p)
dfe <- n - p
## Corrected Degrees of Freedom Total:   DFT = n - 1
dft <- n - 1

## Mean of Squares for Model:   MSM = SSM / DFM
msm <- ssm / dfm
## Mean of Squares for Error:   MSE = SSE / DFE
```

```r
mse <- sse / dfe
## Mean of Squares Total:   MST = SST / DFT
mst <- sst / dft
## F Statistic F = MSM/MSE
f_score <- msm / mse

## Adjusted R Squared R2 = 1 - (1 - R2)(n - 1) / (n - p)
adjusted_r_squared <- 1 - (1 - r_squared) * (dft) / (dfe)

## Calculate the p-value from the F distribution
p_value <- pf(f_score, dfm, dft, lower.tail=F)
```

Assignment 07

```r
## Load the `data/r4ds/heights.csv` to
heights_df <- read.csv("C:/Users/Stewart/Documents/GitHUb/dsc520/data/r4ds/heights.csv")

# Fit a linear model
earn_lm <-  lm(earn ~ ed + age + race + height + sex, data=heights_df)

# View the summary of your model
summary(earn_lm)
```

```
##
## Call:
## lm(formula = earn ~ ed + age + race + height + sex, data = heights_df)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -39423  -9827  -2208   6157 158723
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -41478.4    12409.4  -3.342 0.000856 ***
## ed             2768.4      209.9  13.190  < 2e-16 ***
## age             178.3       32.2   5.537 3.78e-08 ***
## racehispanic  -1414.3     2685.2  -0.527 0.598507
## raceother       371.0     3837.0   0.097 0.922983
## racewhite      2432.5     1723.9   1.411 0.158489
## height          202.5      185.6   1.091 0.275420
## sexmale       10325.6     1424.5   7.249 7.57e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17250 on 1184 degrees of freedom
## Multiple R-squared:  0.2199, Adjusted R-squared:  0.2153
## F-statistic: 47.68 on 7 and 1184 DF,  p-value: < 2.2e-16
```

```r
predicted_df <- data.frame(
  earn = predict(earn_lm, heights_df),
  ed=heights_df$ed, race=heights_df$race, height=heights_df$height,
  age=heights_df$age, sex=heights_df$sex
  )

## Compute deviation (i.e. residuals)
mean_earn <- mean(heights_df$earn)
## Corrected Sum of Squares Total
sst <- sum((mean_earn - heights_df$earn) ^ 2)
## Corrected Sum of Squares for Model
ssm <- sum((mean_earn - predicted_df$earn)^2)
## Residuals
residuals <- heights_df$earn - predicted_df$earn
## Sum of Squares for Error
sse <- sum(residuals ^ 2)
## R Squared
r_squared <- ssm / sst

## Number of observations
n <- nrow(predicted_df)
## Number of regression paramaters
p <- 8
## Corrected Degrees of Freedom for Model
dfm <- p - 1
## Degrees of Freedom for Error
dfe <- n - p
## Corrected Degrees of Freedom Total:   DFT = n - 1
dft <- n - 1

## Mean of Squares for Model:   MSM = SSM / DFM
msm <- ssm / dfm
## Mean of Squares for Error:   MSE = SSE / DFE
mse <- sse / dfe
## Mean of Squares Total:   MST = SST / DFT
mst <- sst / dft
## F Statistic
f_score <- msm / mse

## Adjusted R Squared R2 = 1 - (1 - R2)(n - 1) / (n - p)
adjusted_r_squared <- 1 - (1 - r_squared) * (n - 1) / (n - p)
```

Housing Data

i. Initial Transformations to Dataset

I made two initial transformation to the dataset. I first changed the column names so that they were all consistent sylistically (e.g. Sale Price -> sale_price). I also created another column called bath_total which will have the cumulative amount of baths for each house. This will simplify the process when we start to choose predictors.

```
housing <- read_excel('C:/Users/Stewart/Documents/GitHub/dsc520/data/week-7-housing.xlsx')
str(housing)
```

```
## 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 N
##  $ 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 ...
```

```
# check names
names(housing)
```

```
##  [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"
```

```
# make names uniform
housing <- housing %>%
  rename(sale_date = `Sale Date`, sale_price = `Sale Price`, site_type = sitetype,
         city_name = ctyname, postal_city_n = postalctyn, sq_ft_living = square_feet_total_living)
# creates bath_total variable
housing <- housing %>%
  mutate(bath_half_count = bath_half_count * .5, bath_3qtr_count = bath_3qtr_count * .75) %>%
```

```
  select(bath_full_count, bath_half_count, bath_3qtr_count) %>%
  mutate(bath_total = rowSums(.)) %>%
  summarize(housing, cbind(housing, bath_total))
```

ii. Establishing Parameters

```
# simple regression model; predictor = sq_ft_lot, outcome = Sale Price
sales_simple <- lm(sale_price ~ sq_ft_lot, data = housing)
```

Correlation Tests

In order to find additional predictors, I will do correlation tests between Sale Price and other variables. I will use those predictors which have the highest correlation as additional predictors. For the sake of this analysis I will not be considering categorical variables.

```
# correlation tests for beds, sq_ft, sq_ft_living, baths, year_built, and year_ren
beds_cor <- cor(housing$sale_price, housing$bedrooms)
sq_ft_cor <- cor(housing$sale_price, housing$sq_ft_lot)
living_cor <- cor(housing$sale_price, housing$sq_ft_living)
bath_cor <- cor(housing$sale_price, housing$bath_total)
year_built_cor <- cor(housing$sale_price, housing$year_built)
year_ren_cor <- cor(housing$sale_price, housing$year_renovated)
# create list of correlations so we can compare
cor_summary <- list(Bedrooms = beds_cor, Sq_Ft = sq_ft_cor, Living_Area = living_cor,
                    Baths = bath_cor, Year_Built = year_built_cor, Year_Ren = year_ren_cor)
cor_summary
```

```
## $Bedrooms
## [1] 0.2254675
##
## $Sq_Ft
## [1] 0.1198122
##
## $Living_Area
## [1] 0.4545876
##
## $Baths
## [1] 0.3544926
##
## $Year_Built
## [1] 0.2426713
##
## $Year_Ren
## [1] 0.03286429
```

Based on the preceding analysis, I will be using bedrooms, sq_ft_living, bath_total, and year_built as additional predictors.

```
# multiple regression model using sq_ft_living, bedrooms, baths, year_built as predictors
sales_mult <- lm(sale_price ~ sq_ft_living + bedrooms + bath_total + year_built, data = housing)
```

iii. Summary of Models

```
# create a summary for both models
summary(sales_simple)
```

```
##
## Call:
## lm(formula = sale_price ~ sq_ft_lot, data = housing)
##
## 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(sales_mult)
```

```
##
## Call:
## lm(formula = sale_price ~ sq_ft_living + bedrooms + bath_total +
##     year_built, data = housing)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1743595  -120873   -42763    45142  3905187
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -4.837e+06  4.091e+05 -11.822   <2e-16 ***
## sq_ft_living  1.779e+02  5.081e+00  35.015   <2e-16 ***
## bedrooms     -1.275e+04  4.629e+03  -2.754   0.0059 **
## bath_total    1.308e+03  7.225e+03   0.181   0.8564
## year_built    2.552e+03  2.065e+02  12.362   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 357400 on 12860 degrees of freedom
## Multiple R-squared:  0.2189, Adjusted R-squared:  0.2187
## F-statistic: 901.1 on 4 and 12860 DF,  p-value: < 2.2e-16
```

Interpretation

- R Squared for Simple Model: .01435
- Adjusted R Squared for Simple Model: .01428
- R Squared for Multiple Model: .2189
- Adjusted R Squared for Multiple Model: .2187

Comparing the r squared values to one another, we can see that adding additional parameters greatly increased the amount of variance explained by our model. While this always occurs as more parameters are added, the adjusted r squared value accounts for additional parameters. Given this, it is safe to say that including these additional predictors helped explain more of the variations found in Sale Price

iv. Standardized Betas

```
# standardized betas for predictors
lm.beta(sales_mult)
```

```
##
## Call:
## lm(formula = sale_price ~ sq_ft_living + bedrooms + bath_total +
##     year_built, data = housing)
##
## Standardized Coefficients::
##  (Intercept) sq_ft_living     bedrooms   bath_total   year_built
##           NA  0.435501453 -0.027618273  0.002247921  0.108684218
```

- Total Living Area Standardized Beta: .436
- Bedrooms Standardized Beta: -.028
- Baths Standardized Beta: .002
- Year Built Standardized Beta: .109

The standardized beta tells us how many standard deviations the outcome changes with each standard deviation change in each predictor when all other predictors are held constant. Each beta is in standard deviations so they are directly comparable. From the above we can see that total living area had the highest effect on the Sale Price.

v. Confidence Intervals

```
# confidence intervals for predictors
confint(sales_mult)
```

```
##                     2.5 %        97.5 %
## (Intercept)  -5638494.4794 -4034680.4359
## sq_ft_living      167.9602      187.8802
## bedrooms       -21821.0260    -3673.6882
## bath_total     -12855.1497    15470.3098
## year_built       2147.5372     2956.9241
```

Confidence intervals tell us the odds that the value of b in our sample is the true value of b in the population. The smaller the confidence interval, the more representative it is of the population. In the parameters we are using, sq_ft_living has the tightest confidence interval, indicating it likely representative of the population. On the other hand, bedrooms has a huge confidence interval, from -21821 to -3673 meaning that the parameter is not nearly as representative.

The bath_total result reveals something interesting. Its confidence interval crosses zero, meaning that some samples have a negative relationship and some have a positive relationship. As such, bath_total is not a good parameter for this model.

vi. Comparing Models

```
# compare simple and multiple model using anova
anova(sales_simple, sales_mult)
```

```
## Analysis of Variance Table
##
## Model 1: sale_price ~ sq_ft_lot
## Model 2: sale_price ~ sq_ft_living + bedrooms + bath_total + year_built
##   Res.Df        RSS Df  Sum of Sq      F    Pr(>F)
## 1  12863 2.0734e+15
## 2  12860 1.6431e+15  3 4.3031e+14 1122.7 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The F Ratio is 1122.7, which, given a p-value $< .001$, means that the multiple model is a significant improvement on the simple model.

vii. Casewise Diagnostics

```
# calculates and stores casewise diagnostics
housing$residuals <- resid(sales_mult)
housing$studnetized_residuals <- rstandard(sales_mult)
housing$standardized_residuals <- rstudent(sales_mult)
housing$cooks_distance <- cooks.distance(sales_mult)
housing$leverage <- hatvalues(sales_mult)
housing$covariance_ratios <- covratio(sales_mult)
```

viii. Large Standardized Residuals

```
# stores residuals that are +/- 2 in own variable
housing$large_residuals <- housing$standardized_residuals > 2 | housing$standardized_residuals < -2
```

```
# adds together all large residuals
lg_res_ct <- sum(housing$large_residuals)
lg_res_ct
```

ix. Sum of Large Residuals

```
## [1] 330
```

```
# prints all problematic cases for parameters used in model
housing[housing$large_residuals, c("sale_price", "sq_ft_living", "bedrooms", "bath_total", "year_built"
```

x. Variables with Large Residuals

```
## # A tibble: 330 x 6
##    sale_price sq_ft_living bedrooms bath_total year_built standardized_residuals
##         <dbl>        <dbl>    <dbl>      <dbl>      <dbl>                  <dbl>
##  1     184667         4160        4       3.25       2005                  -2.21
##  2     265000         4920        4       4.5        2007                  -2.38
##  3    1390000          660        0       1          1955                   3.13
##  4     229000         3840        0       0          2008                  -2.08
##  5     390000         5800        5       4.5        2008                  -2.44
##  6    1588359         3360        2       2.5        2005                   2.05
##  7    1450000          900        2       1          1918                   3.52
##  8     163000         4710        4       4          2014                  -2.61
##  9     270000         5060        4      23.5        2016                  -2.80
## 10     200000         6880        5       4.5        2008                  -3.51
## # ... with 320 more rows
```

xi. Leverage, Cook's Distance, Covariance Ratios for Problematic Cases

```
larg_resid <- housing[housing$large_residuals, c("cooks_distance", "leverage", "covariance_ratios")]
# calculate how many times cook distance is greater than one
cook_threshold <- 1
cook_violations_count <- sum(larg_resid$cooks_distance > cook_threshold)
cook_violations_count
```

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

```
# calculates how many times leverage is greater than threshold ((k + 1) / n) * 3
leverage_threshold <- (5 / nrow(housing) * 3)
leverage_violations_count <- sum(larg_resid$leverage > leverage_threshold)
leverage_violations_count
```

```
## [1] 63
```

```
# calculates how many times covariance ratio is greater or lower than upper/lower bounds
cov_threshold_upper <- 1 + (3 * (5)/nrow(housing))
cov_threshold_lower <- 1 - (3 * 5 / nrow(housing))
cov_violations_count <- sum(larg_resid$covariance_ratios > cov_threshold_upper | larg_resid$covariance_
cov_violations_count
```

```
## [1] 261
```

xii. Assumption of Independence

The D-W Statistic is .558, which means the assumption of independence is not met. ## xiii. Assumption of No Multicollinearity

```
vif(sales_mult)
```

```
## sq_ft_living     bedrooms   bath_total   year_built
##     2.546897     1.656094     2.540306     1.272659
```

```
1/vif(sales_mult)
```

```
## sq_ft_living     bedrooms   bath_total   year_built
##    0.3926347    0.6038305    0.3936534    0.7857566
```

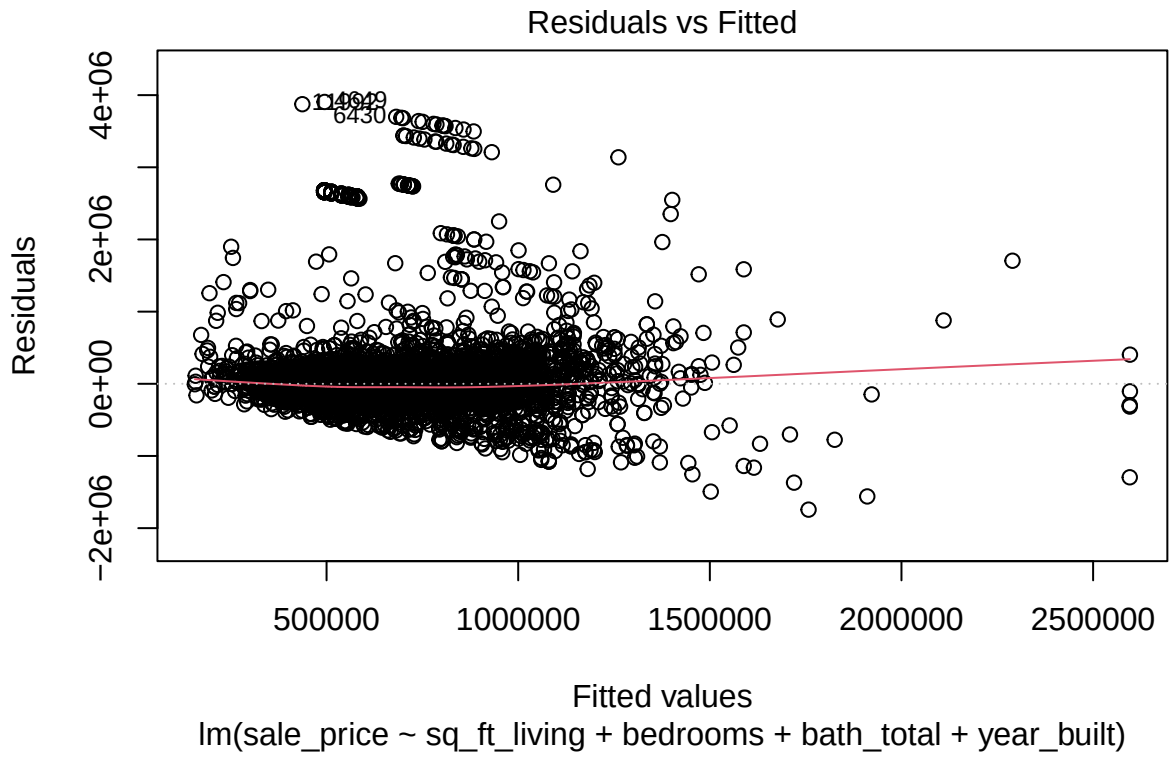
```
mean(vif(sales_mult))
```

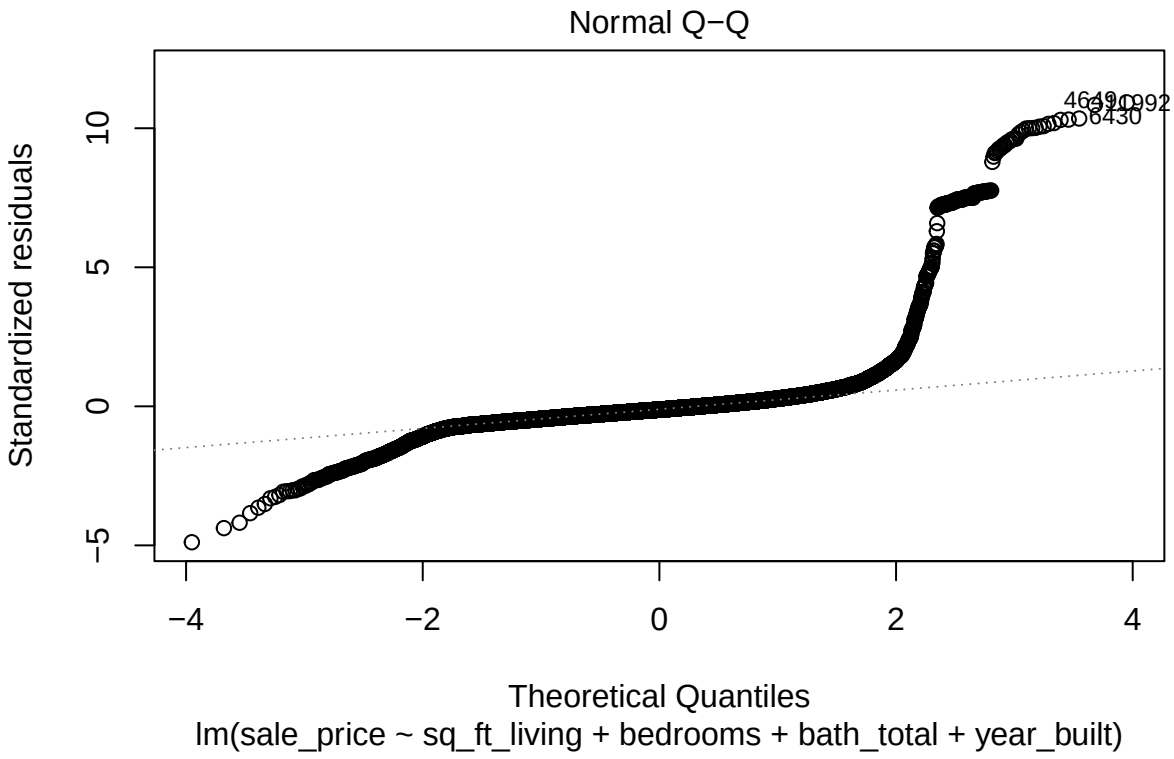
```
## [1] 2.003989
```

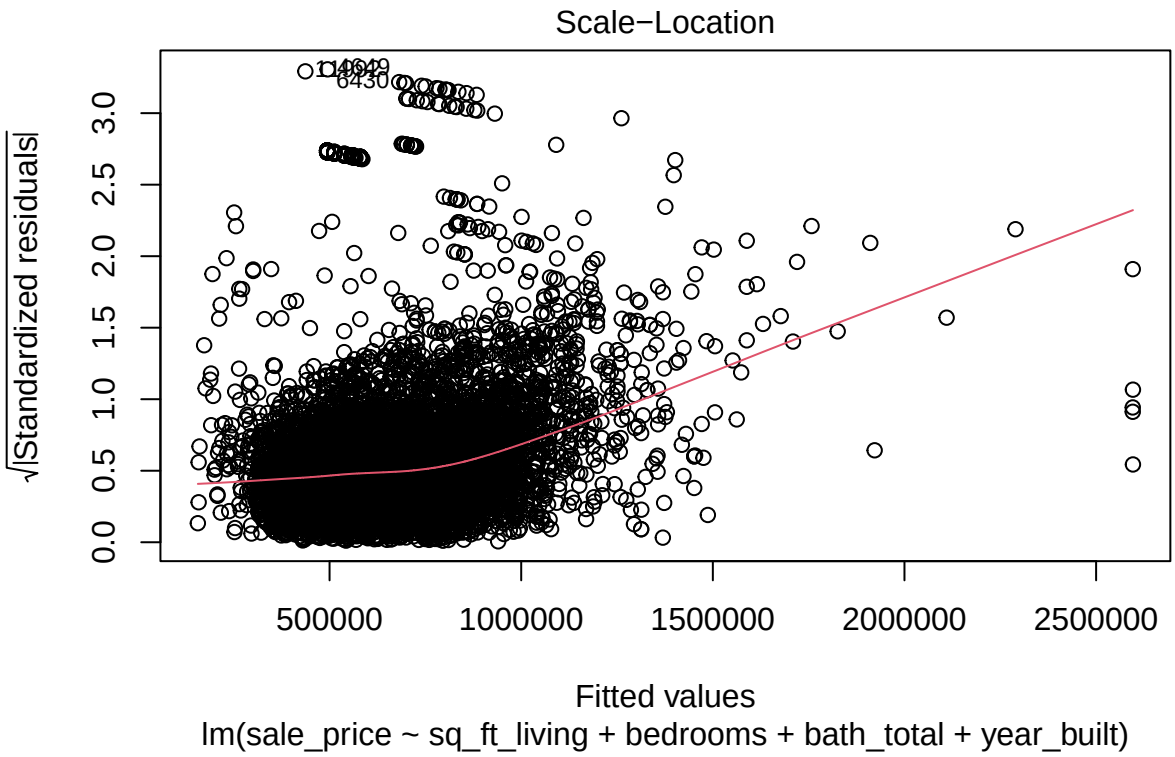
There is no VIF larger than 10 nor tolerance lower than .2 which suggests that there is no collinearity within the data. However, the average VIF is greater than 1, which could indicate possible bias.

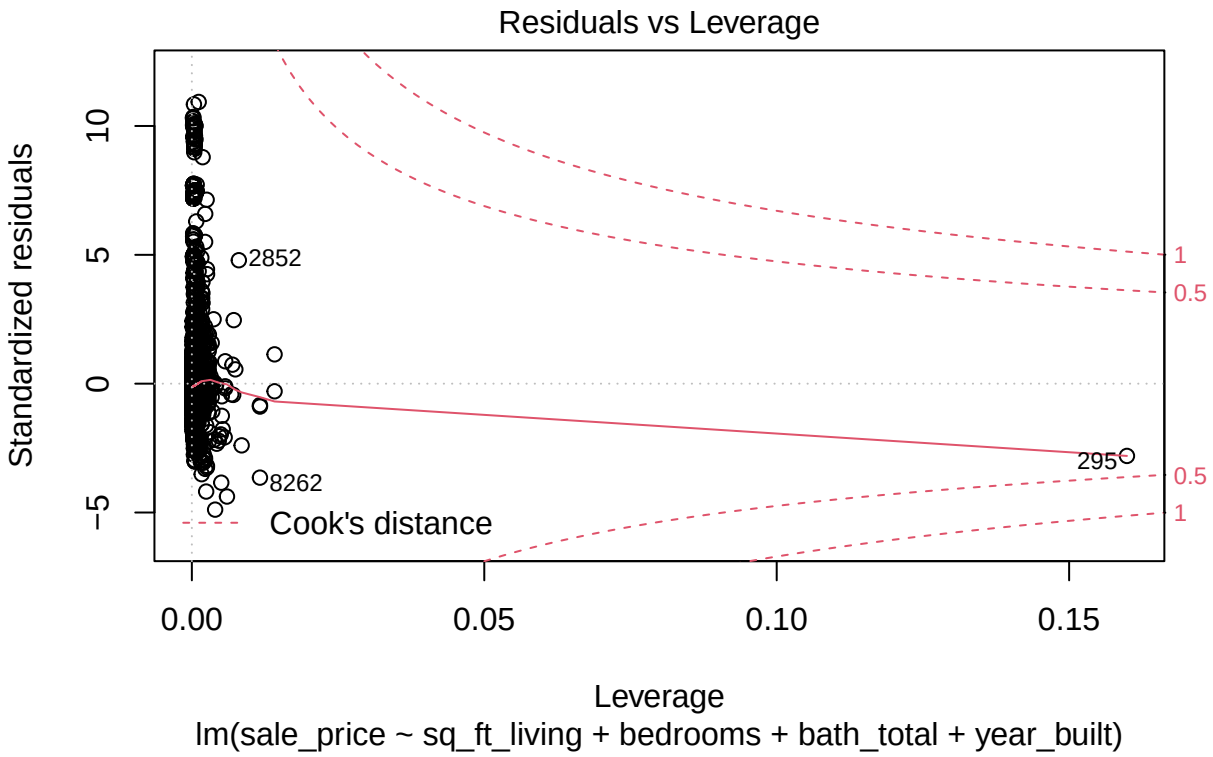
xiv. Assumptions About Residuals

```
plot(sales_mult)
```




Normal Q−Q
Standardized residuals
10
5
0
−5
−4
−2
0
2
4
Theoretical Quantiles
lm(sale_price ~ sq_ft_living + bedrooms + bath_total + year_built)


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_living + bedrooms + bath_total + year_built)




```
hist(housing$standardized_residuals)
```

xv. Overall Conclusions

Overall, the regression model I created is biased. In the first place it violates the assumption of independence. The confidence intervals of the bath_total cross zero, indicating that it is not a good parameter. When we observe scatterplots and histograms of the residuals it is likewise seen to deviate from the norm.

As such, while we may be able to draw some conclusions and insights in this particular housing sample, we cannot say that this sample can generalize to the entire population.

In future analyses there are some steps we can take to try and meet these assumptions. We can do individual simple regression between sale_price and variables in order to see how they stack up on their own. We could then do a more rigid hierarchical regression where we add parameters in order of importance.