Regression

Kyle Chan, Ryan Banafshay

For regression, we used a dataset (https://www.kaggle.com/datasets/nancyalaswad90/diamonds-prices) that contains information on 54,000 diamonds

Linear regression models relationships between predictors in a dataset. It assumes the relationship is linear (i.e. we can produce a linear function out of a dependent and independent variable). The purpose of linear regression is to find a ‘best fit’ line that maintains minimal distance between actual values and the predicted ones.

Loading in data

We begin by extracting the information of diamonds from our csv file. We should also check to make sure we have appropriate column data types.

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✓ ggplot2 3.4.1     ✓ purrr   1.0.1
## ✓ tibble  3.1.8     ✓ dplyr   1.1.0
## ✓ tidyr   1.3.0     ✓ stringr 1.5.0
## ✓ readr   2.1.4     ✓ forcats 1.0.0
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✕ dplyr::filter() masks stats::filter()
## ✕ dplyr::lag()    masks stats::lag()
```

```
data <- read.csv(file = "Diamonds Prices2022.csv")
str(data)
```

```
## 'data.frame':    53943 obs. of  11 variables:
##  $ X      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ carat  : num  0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
##  $ cut    : chr  "Ideal" "Premium" "Good" "Premium" ...
##  $ color  : chr  "E" "E" "E" "I" ...
##  $ clarity: chr  "SI2" "SI1" "VS1" "VS2" ...
##  $ depth  : num  61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
##  $ table  : num  55 61 65 58 58 57 57 55 61 61 ...
##  $ price  : int  326 326 327 334 335 336 336 337 337 338 ...
##  $ x      : num  3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
##  $ y      : num  3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
##  $ z      : num  2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
```

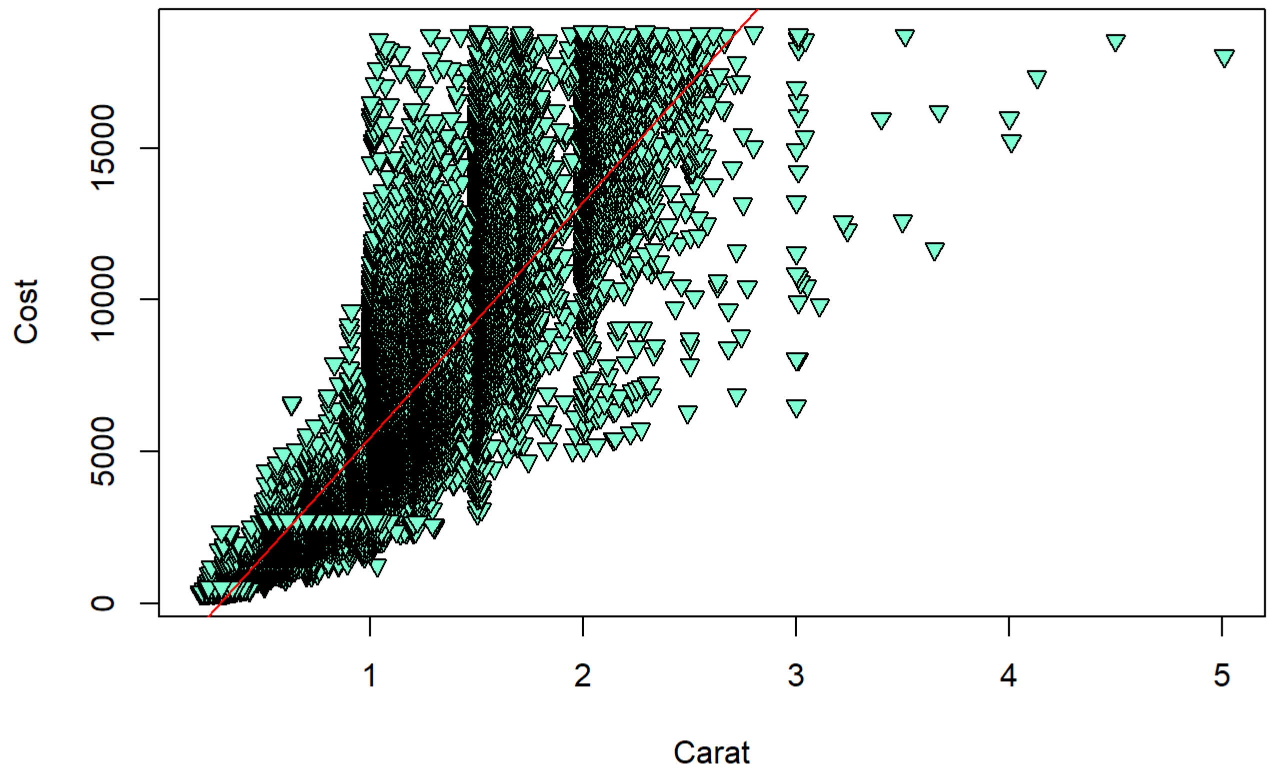
```
summary(data)
```

```
##       X             carat             cut               color          
##  Min.   :    1   Min.   :0.2000   Length:53943       Length:53943      
##  1st Qu.:13486   1st Qu.:0.4000   Class :character   Class :character  
##  Median :26972   Median :0.7000   Mode  :character   Mode  :character  
##  Mean   :26972   Mean   :0.7979                                        
##  3rd Qu.:40458   3rd Qu.:1.0400                                        
##  Max.   :53943   Max.   :5.0100                                        
##    clarity              depth           table           price      
##  Length:53943       Min.   :43.00   Min.   :43.00   Min.   :  326  
##  Class :character   1st Qu.:61.00   1st Qu.:56.00   1st Qu.:  950  
##  Mode  :character   Median :61.80   Median :57.00   Median : 2401  
##                     Mean   :61.75   Mean   :57.46   Mean   : 3933  
##                     3rd Qu.:62.50   3rd Qu.:59.00   3rd Qu.: 5324  
##                     Max.   :79.00   Max.   :95.00   Max.   :18823  
##       x                y                z         
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.000  
##  1st Qu.: 4.710   1st Qu.: 4.720   1st Qu.: 2.910  
##  Median : 5.700   Median : 5.710   Median : 3.530  
##  Mean   : 5.731   Mean   : 5.735   Mean   : 3.539  
##  3rd Qu.: 6.540   3rd Qu.: 6.540   3rd Qu.: 4.040  
##  Max.   :10.740   Max.   :58.900   Max.   :31.800  
```

First observations and preparations

We plotted the diamonds with respect to carats and the price just to see what we're working with. The abline helps visualize a general line through the datapoints.

```
par(mfrow=c(1,1))
plot(data$price~data$carat, xlab= "Carat", ylab= "Cost", pch=25, bg=c("aquamarine1"))
abline(lm(data$price~data$carat), col = "red")
```



We will divide our dataset into a 80/20 train/test split.

```
set.seed(1234)
split <- sample(1:nrow(data), nrow(data)*0.8, replace = FALSE)
train <- data[split,]
test <- data[-split,]
```

As previously shown in plotting the datapoints, we can see that majority of diamonds are less than 1 carat.

```
sum(train$carat<=1)
```

```
## [1] 29145
```

```
sum(train$carat<=2 & train$carat>1)
```

```
## [1] 12508
```

```
sum(train$carat<=3 & train$carat>2)
```

```
## [1] 1478
```

```
sum(train$carat<=4 & train$carat>3)
```

```
## [1] 19
```

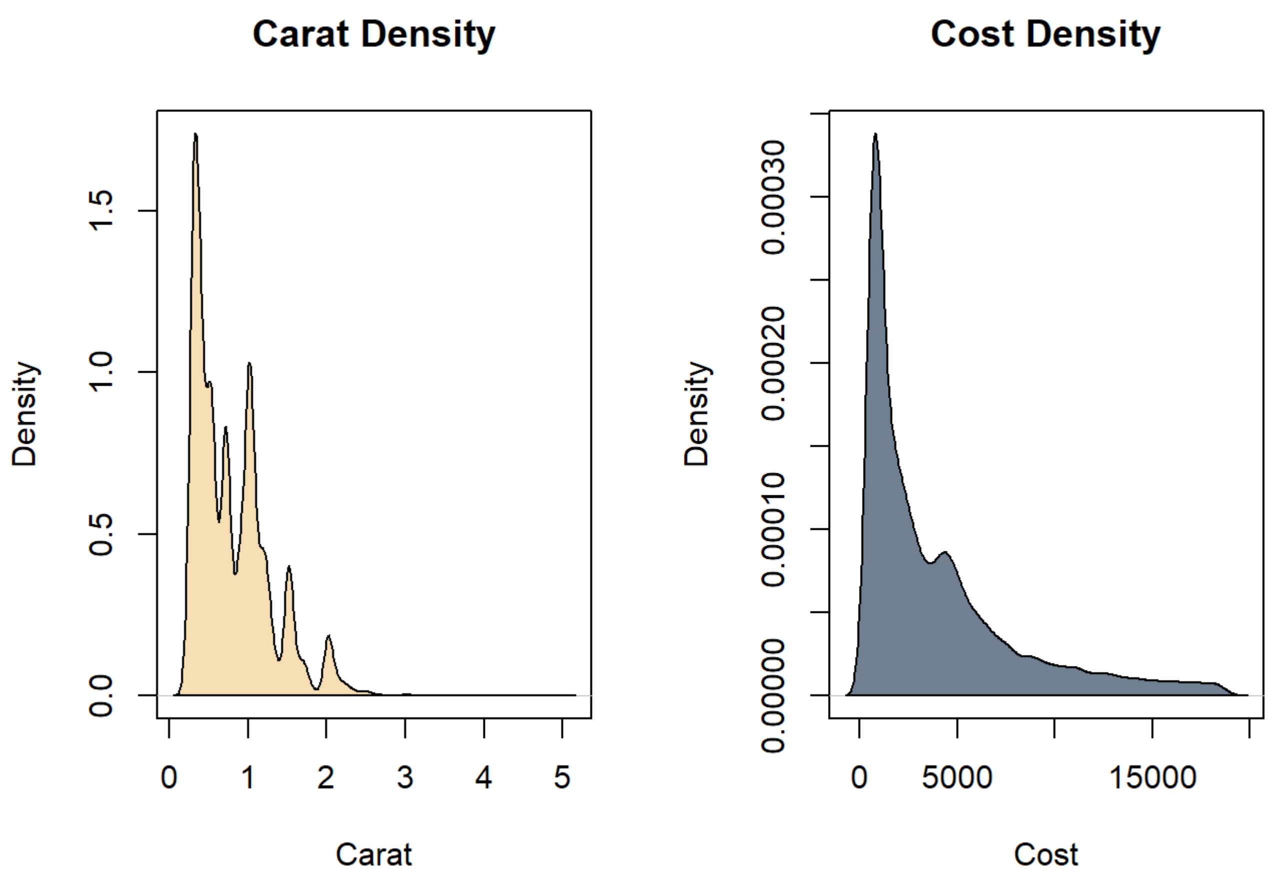
```
sum(train$carat>4)
```

```
## [1] 4
```

We can visualize the densities of the prices and carats in our training data.

```
par(mfrow=c(1,2))
carat_den <- density(train$carat, na.rm = TRUE)
plot(carat_den, main = "Carat Density", xlab = "Carat")
polygon(carat_den, col ="wheat")

cost_den <- density(train$price, na.rm = TRUE)
plot(cost_den, main = "Cost Density", xlab = "Cost")
polygon(cost_den, col ="slategrey")
```



Here are some more functions to further explore our dataset.

```
summary(train$carat)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  0.2000  0.4000  0.7000  0.7976  1.0400  5.0100
```

```
summary(train$price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##     326     949    2396    3929    5322   18823
```

```
range(train$carat)
```

```
## [1] 0.20 5.01
```

```
range(train$price)
```

```
## [1]   326 18823
```

```
cor(train$carat, train$price, use = "complete.obs")
```

```
## [1] 0.9216323
```

First model

Let’s build a simple linear regression model using one predictor.

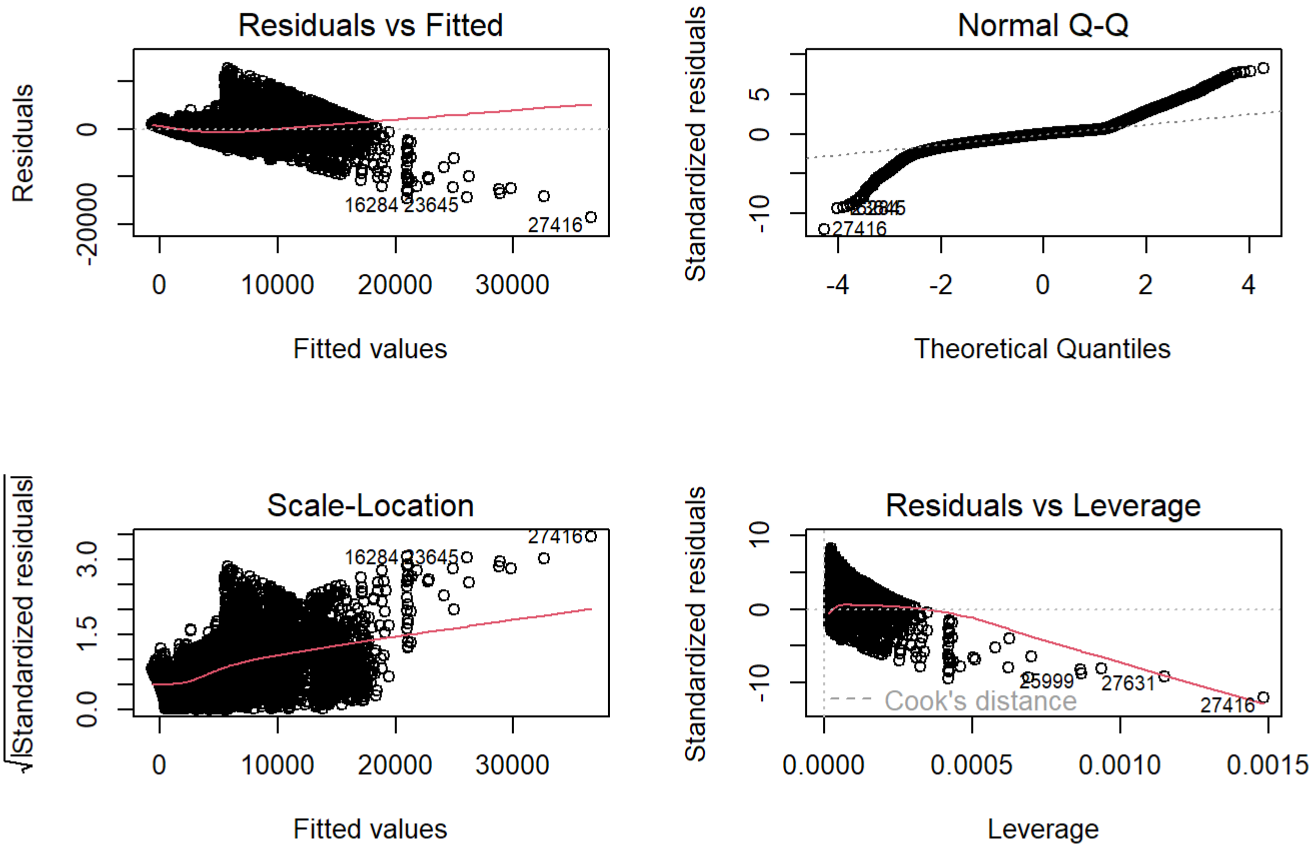
```
lm1 <- lm(price~carat, data=data)
summary(lm1)
```

```
##
## Call:
## lm(formula = price ~ carat, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18585.4   -804.7    -19.1    537.5  12731.7
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2256.40      13.05  -172.8   <2e-16 ***
## carat        7756.44      14.07   551.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1549 on 53941 degrees of freedom
## Multiple R-squared:  0.8493, Adjusted R-squared:  0.8493
## F-statistic: 3.041e+05 on 1 and 53941 DF,  p-value: < 2.2e-16
```

According to our first linear regression model's summary, every whole carat increase would increase the price of the diamond by $7756.44, give-or-take roughly $14. The R-squared value is also good, at 0.8493, since being closer to 1 is ideal. The F-statistic is way greater than 1, and our p-value is very low. This all indicates we have a relatively good model to predict future values.

Let's also plot and analyze our residuals.

```
par(mfrow=c(2,2))
plot(lm1)
```



Residuals vs Fitted (1) We can see that most datapoints are scattered realtively equally and randomly around the horizontal line 0, meaning that there is a high chance the relationship between carats and price is linear.

Normal Q-Q (2) The data largely follows the line, indicating a normal distribution. However, the deviations on the ends of either side may indicate some skew and heavy tails.

Scale-Location (3) The residuals are mostly randomly scattered along the prediction. This shows that we generally maintain equal variance across our data.

Residuals vs Leverage (4) Our residual vs leverage plot demonstrate we have a few strongly influential datapoints. We can attribute these as outliers in our data.

Second model

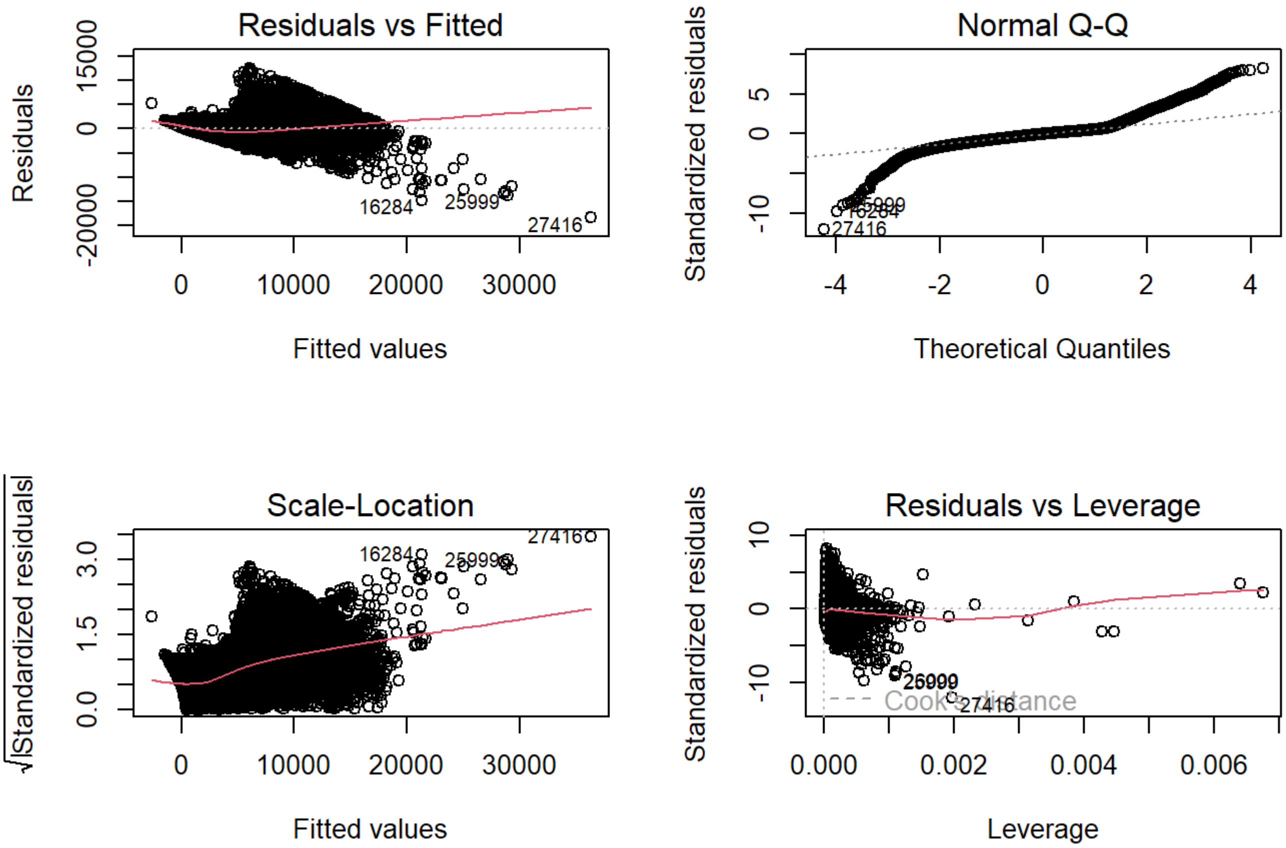
Let's attempt to improve on our first model by adding in multiple predictors.

```
lm2 <- lm(price~carat+depth+table, data=train)
summary(lm2)
```

```
##
## Call:
## lm(formula = price ~ carat + depth + table, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18302.7   -784.0    -30.4    527.7  12484.8
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13041.077    436.295   29.89   <2e-16 ***
## carat        7862.406     15.822  496.92   <2e-16 ***
## depth        -150.764      5.372  -28.07   <2e-16 ***
## table        -105.695      3.510  -30.11   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1526 on 43150 degrees of freedom
## Multiple R-squared:  0.8538, Adjusted R-squared:  0.8538
## F-statistic: 8.4e+04 on 3 and 43150 DF,  p-value: < 2.2e-16
```

We can start to see some minor improvements to our model as a result. Our new model has an R-squared value of 0.8538, which is an improvement of 0.0045 from our previous model.

```
par(mfrow=c(2,2))
plot(lm2)
```

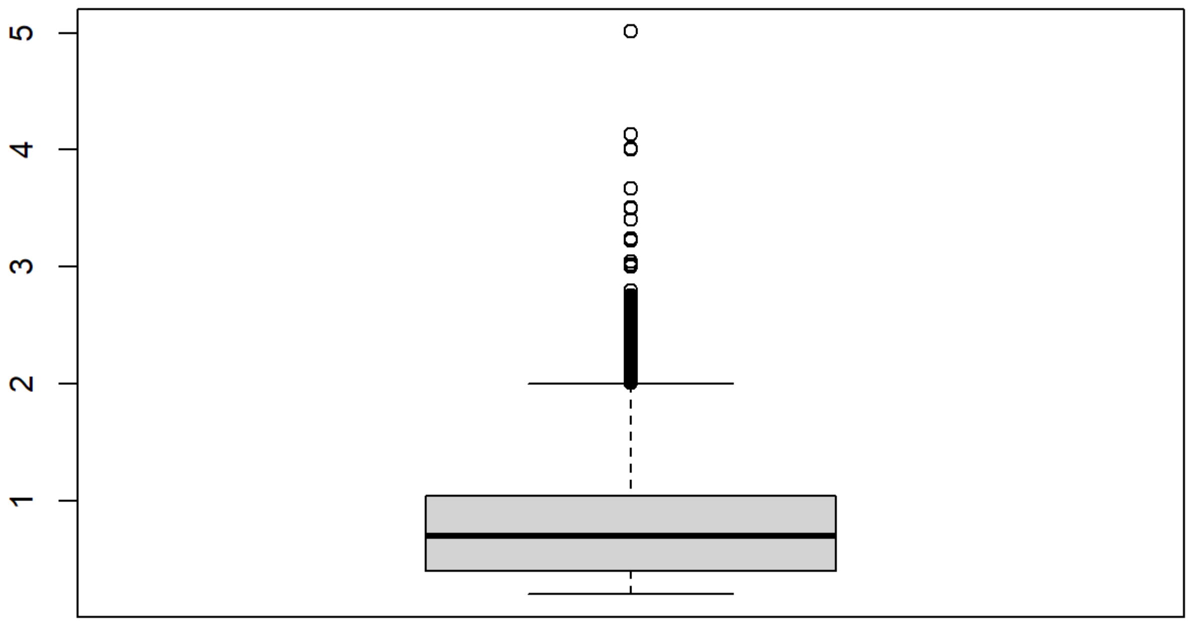


Third model

We will now attempt to increase our model's accuracy by removing outliers.

First, let's find the outliers in our training data.

```
par(mfrow=c(1,1))
outliers <- boxplot(train$carat, plot=TRUE)$out
```




```
min(outliers)
```

```
## [1] 2.01
```

```
length(outliers)
```

```
## [1] 1501
```

As we can see, all carats at and above 2.01 are deemed as outliers. We also observe that there are 1501 or these outliers affecting our model.

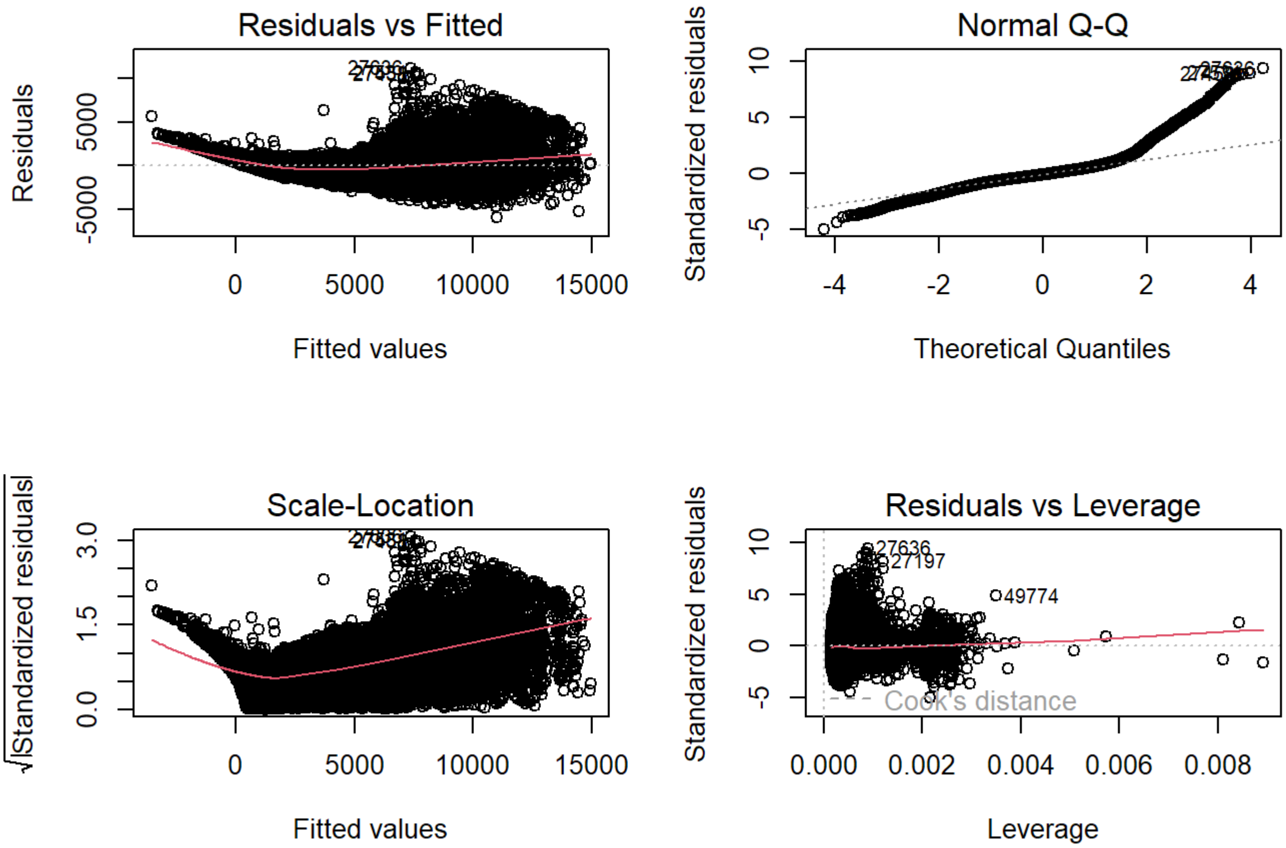
Now, we can remove all these diamonds from our training data, and create a new model with more predictors in an attempt to improve our accuracy.

```
no_outliers_data <- subset(train, carat < 2.01)
lm3 <- lm(price~carat+depth+table+cut+clarity, data=no_outliers_data)
summary(lm3)
```

```
##
## Call:
## lm(formula = price ~ carat + depth + table + cut + clarity, data = no_outliers_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5937.9  -606.8  -109.3   450.7 11180.2
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2635.785    439.399  -5.999 2.01e-09 ***
## carat         8417.905     15.534 541.889  < 2e-16 ***
## depth          -33.398      4.795  -6.966 3.32e-12 ***
## table          -31.083      3.510  -8.856  < 2e-16 ***
## cutGood        493.683     40.562  12.171  < 2e-16 ***
## cutIdeal       740.454     40.447  18.307  < 2e-16 ***
## cutPremium     676.735     39.047  17.331  < 2e-16 ***
## cutVery Good   650.797     38.998  16.688  < 2e-16 ***
## clarityIF     4380.099     62.489  70.093  < 2e-16 ***
## claritySI1    2643.827     54.210  48.770  < 2e-16 ***
## claritySI2    1782.307     54.664  32.605  < 2e-16 ***
## clarityVS1    3511.541     55.153  63.669  < 2e-16 ***
## clarityVS2    3294.359     54.459  60.493  < 2e-16 ***
## clarityVVS1   4054.245     58.045  69.846  < 2e-16 ***
## clarityVVS2   4030.118     56.632  71.164  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1191 on 41638 degrees of freedom
## Multiple R-squared:  0.8787, Adjusted R-squared:  0.8787
## F-statistic: 2.155e+04 on 14 and 41638 DF,  p-value: < 2.2e-16
```

Our R-squared value increased to 0.8787, which is a 0.0249 increase from our second linear regression model.

```
par(mfrow=c(2,2))
plot(lm3)
```



Model comparisons

Our first model was a simple linear regression model with respect to the price and carat of a diamond, with an R^2 value of 0.8493. The second model was a multiple linear regression model plotting multiple predictors such as carat, depth, and table with respect to the price of a diamond. The multiple linear regression model was slightly better with an R^2 value of 0.8538.

In the third and final model, we attempted to improve the previous ones by removing outliers that we thought skewed our data. We also introduced a couple more predictors (cut and clarity) to aid in this. We observed a minimum of 2.5% increase in accuracy, with R^2 being 0.8787. Residual standard error also decreased by roughly 300, meaning our model fits the dataset better.

```
prediction <- predict(lm3, newdata=test)
correlation <- cor(prediction, test$price)
mse <- mean((prediction - test$price)^2)
rmse <- sqrt(mse)

print(correlation)
```

```
## [1] 0.9467995
```

```
print(mse)
```

```
## [1] 1643148
```

```
print(rmse)
```

```
## [1] 1281.853
```

Conclusions

We can conclude that there is a strong positive correlation with the price of a diamond and variables such as carat, depth, table, clarity, and cut.

The dataset does not fit the fitted line quite well, however. This indicates that although the variables can be strong predictors of the price, there is still a lot of variance that may be derived from elsewhere, such as a diamond's sentimental value, reputation, and history.