126 Final Project

Luke Maldonado

2025/12/04

Part 1: Data Description and Descriptive Statistics

```
library(car)
library(tidyr)
library(knitr)
library(carData)
library(ggplot2)
library(tidyverse)
library(kableExtra)
```

Loading in the data itself

```
raw_df <- read.csv("Diamonds Prices2022.csv")
raw_df <- raw_df[, -1]
# Will reuse this code to make the tables i display look better
table <- kable(head(raw_df), format = "latex", caption = "Base Data")
#ensures that the table is displayed
kable_styling(table, latex_options = "HOLD_position")
```

Table 1: Base Data

carat	cut	color	clarity	depth	table	price	x	y	z
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
0.29	Premium	I	VS2	62.4	58	334	4.20	4.23	2.63
0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
0.24	Very Good	J	VVS2	62.8	57	336	3.94	3.96	2.48

1.

```
#Setting the seed to 1234 for reproducibility.
set.seed(1234)
#Randomly sampling half of the data set.
df <- raw_df[sample(nrow(raw_df), (nrow(raw_df))),]
```

```
rownames(df) <- 1:nrow(df) #resetting the indexes.
table <- kable(head(df), format = "latex", caption = "Base Data")
kable_styling(table, latex_options = "HOLD_position")
```

Table 2: Base Data

carat	cut	color	clarity	depth	table	price	x	y	z
0.61	Good	E	I1	63.4	57.1	1168	5.37	5.43	3.42
0.53	Premium	G	SI2	60.8	58.0	1173	5.21	5.19	3.16
0.23	Very Good	E	VVS2	62.3	55.0	505	3.90	3.93	2.44
1.33	Ideal	J	VS1	61.3	57.0	6118	7.11	7.08	4.35
0.30	Ideal	E	VVS1	61.6	56.0	838	4.30	4.34	2.66
0.30	Ideal	D	VS2	60.8	57.0	911	4.34	4.31	2.63

2.

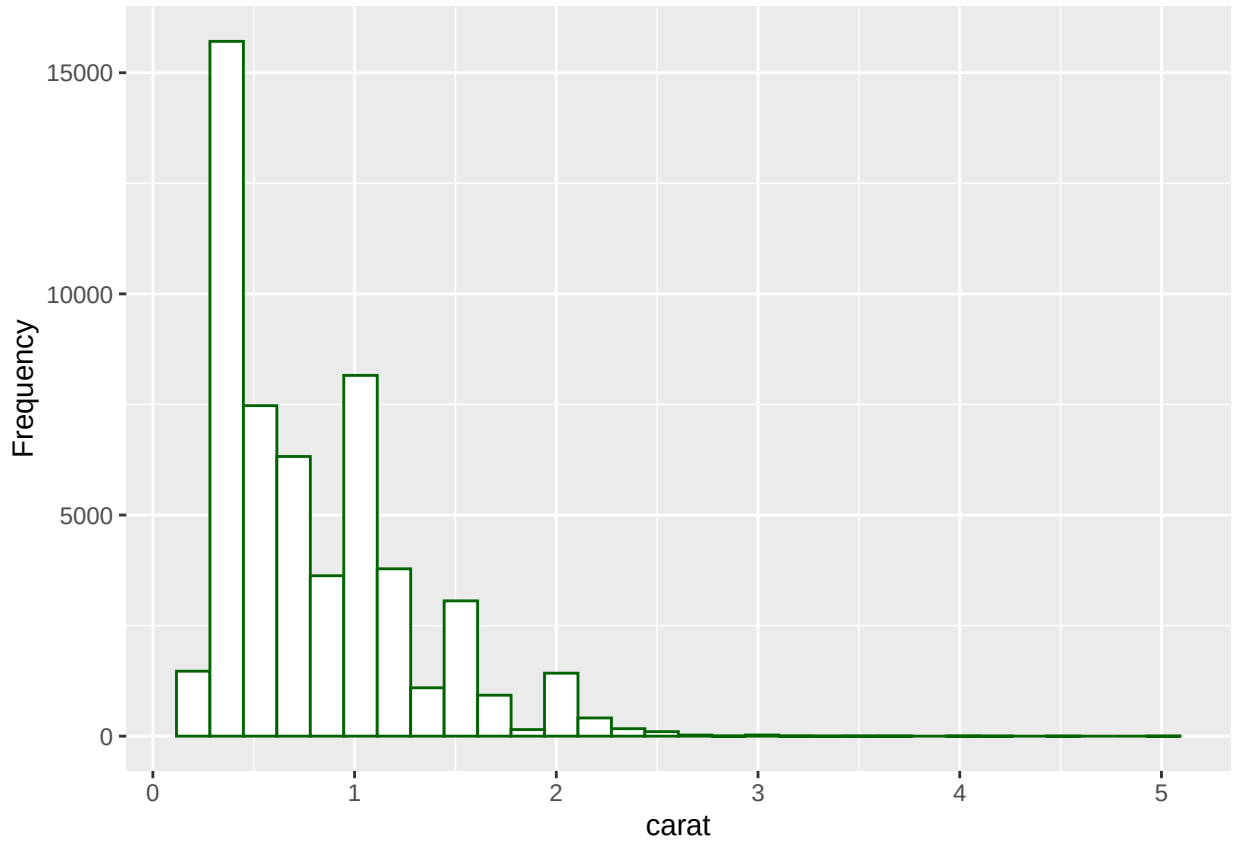
```
summary(df)
```

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

creating histograms with the quantitative continuous variables, all plots separated for knitting.

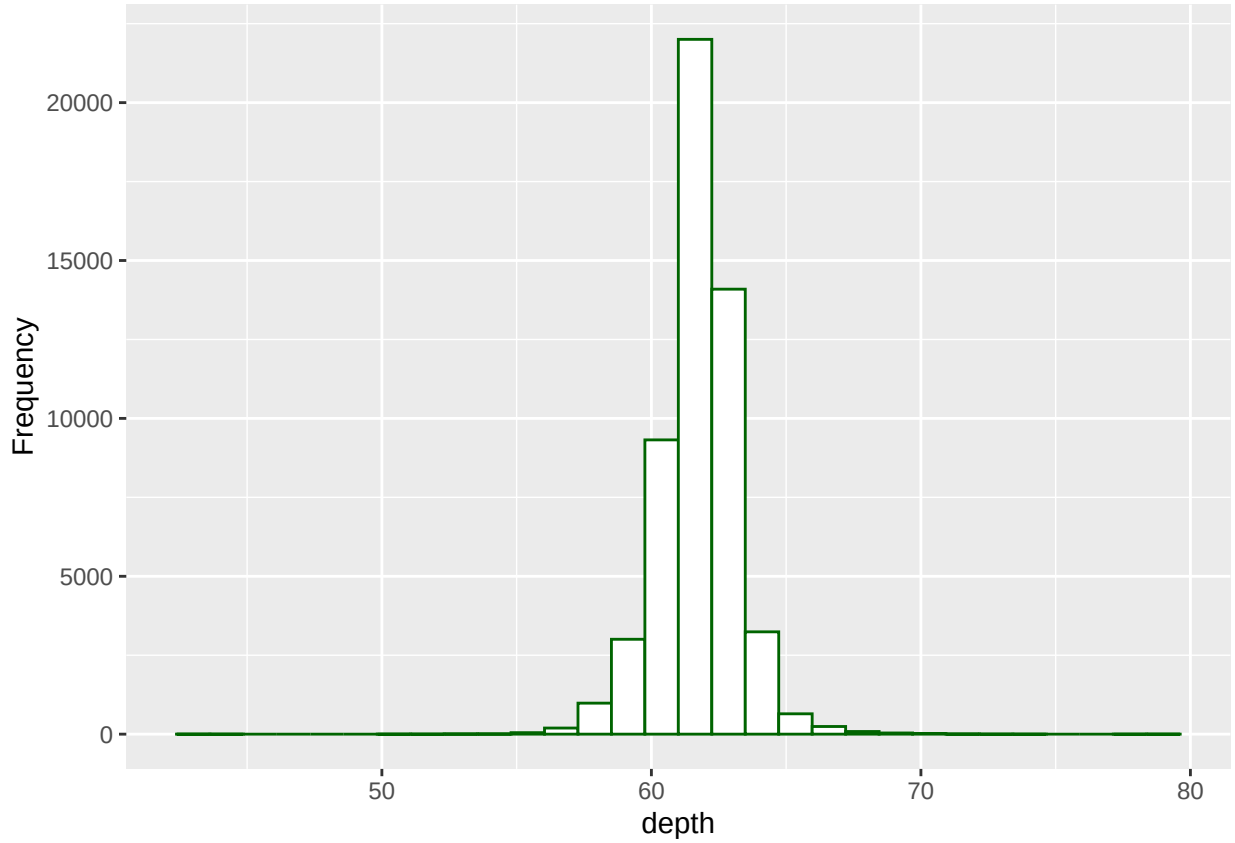
```
continuous_df <- df[, -(2:4)]
ggplot(continuous_df, aes(continuous_df[, 1])) +
  geom_histogram(color='darkgreen', fill='white') +
  labs(x = names(continuous_df)[1], y = "Frequency")
```

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



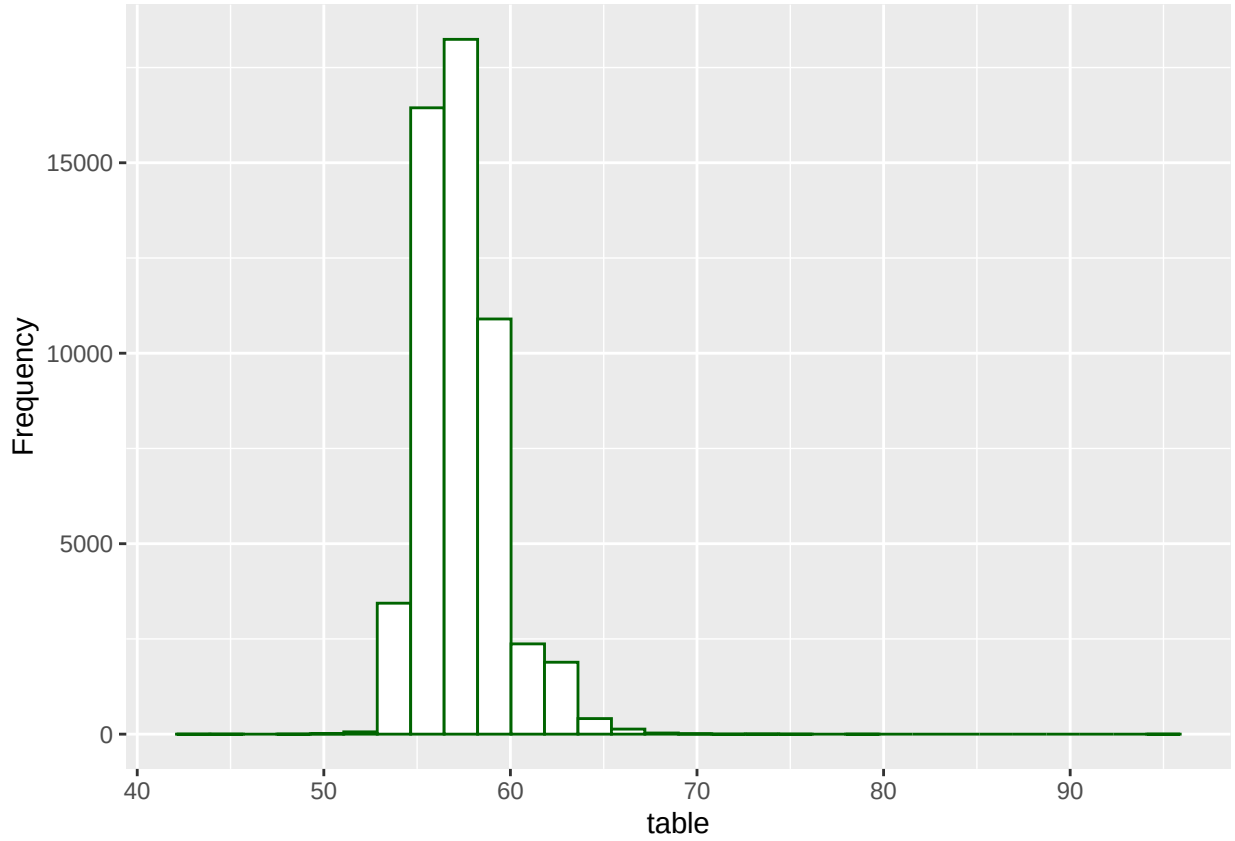

```
ggplot(continuous_df, aes(continuous_df[, 2])) +
  geom_histogram(color='darkgreen', fill='white') +
  labs(x = names(continuous_df)[2], y = "Frequency")
```

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



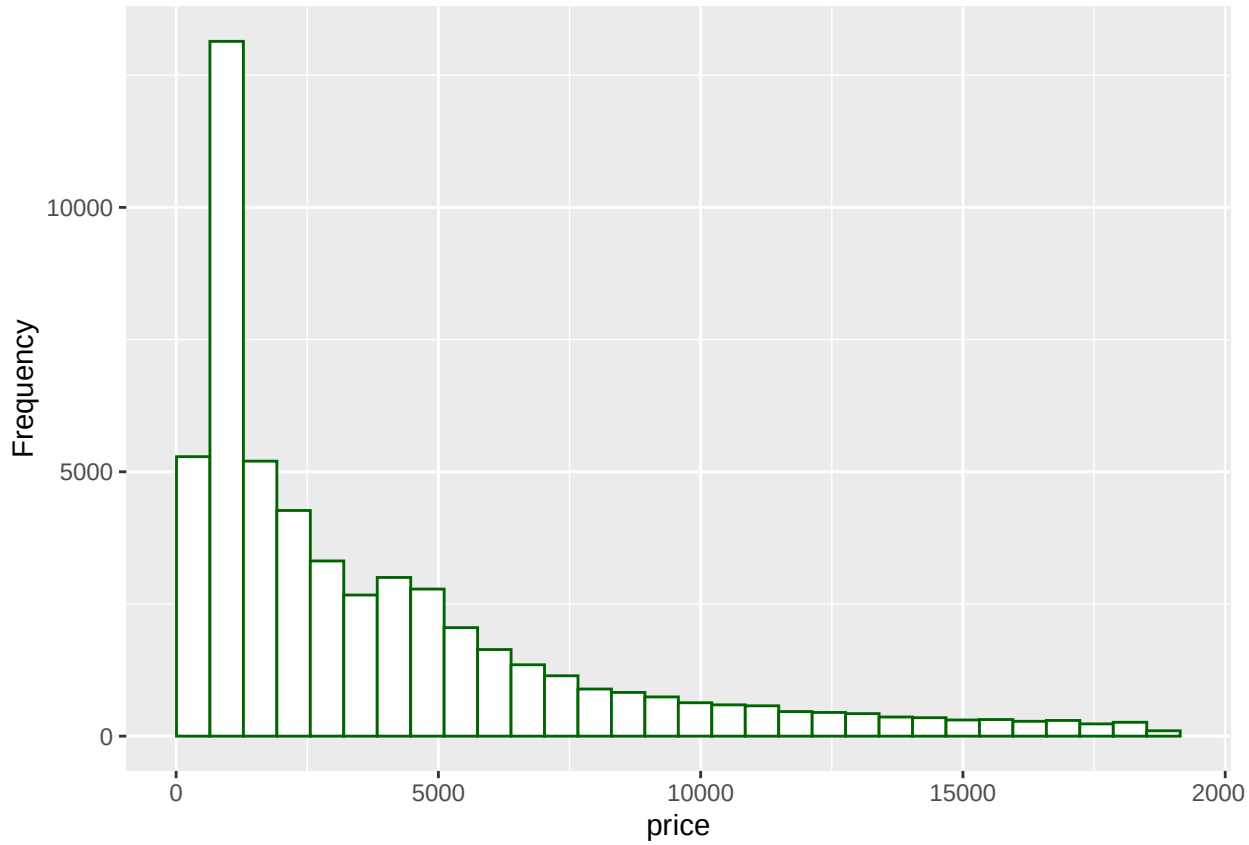

```
ggplot(continuous_df, aes(continuous_df[, 3])) +
  geom_histogram(color='darkgreen', fill='white') +
  labs(x = names(continuous_df)[3], y = "Frequency")
```

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



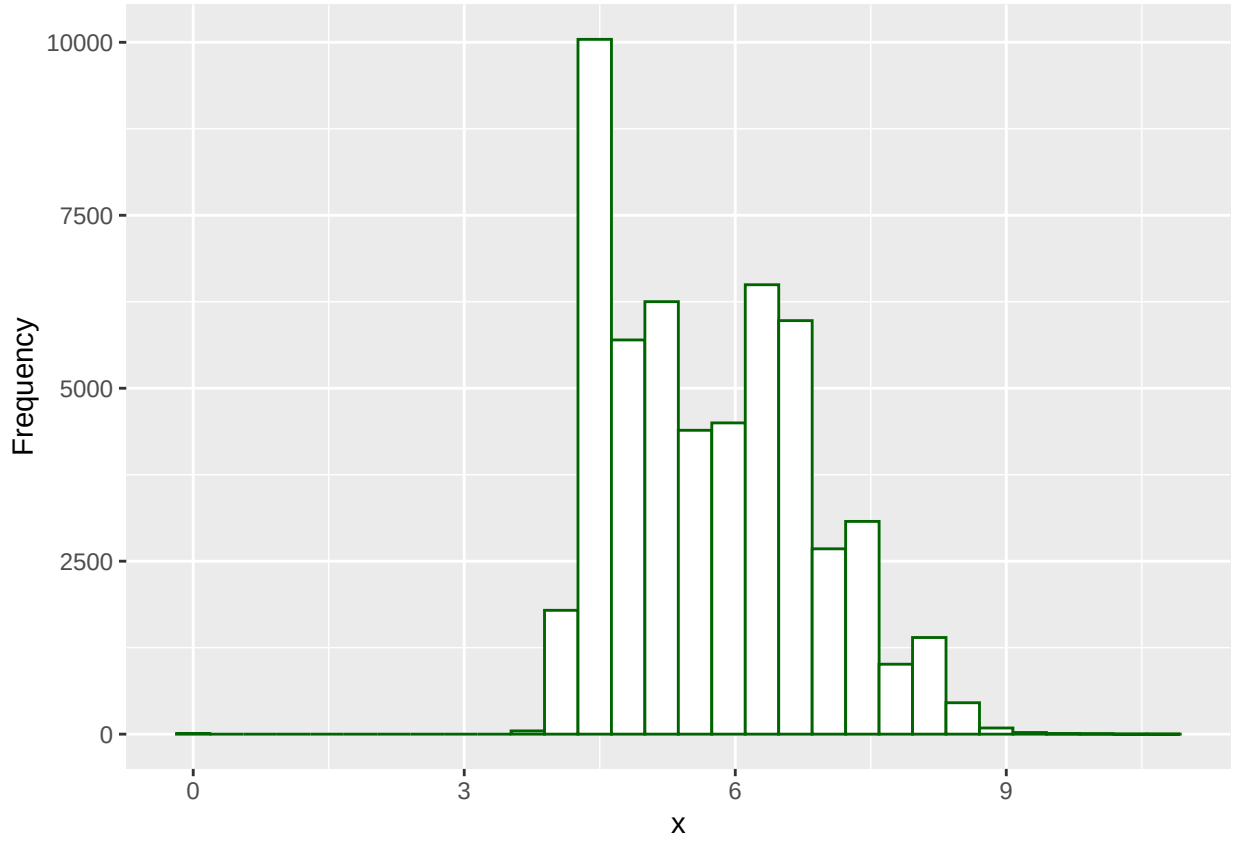

```
ggplot(continuous_df, aes(continuous_df[, 4])) +
  geom_histogram(color='darkgreen', fill='white') +
  labs(x = names(continuous_df)[4], y = "Frequency")
```

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



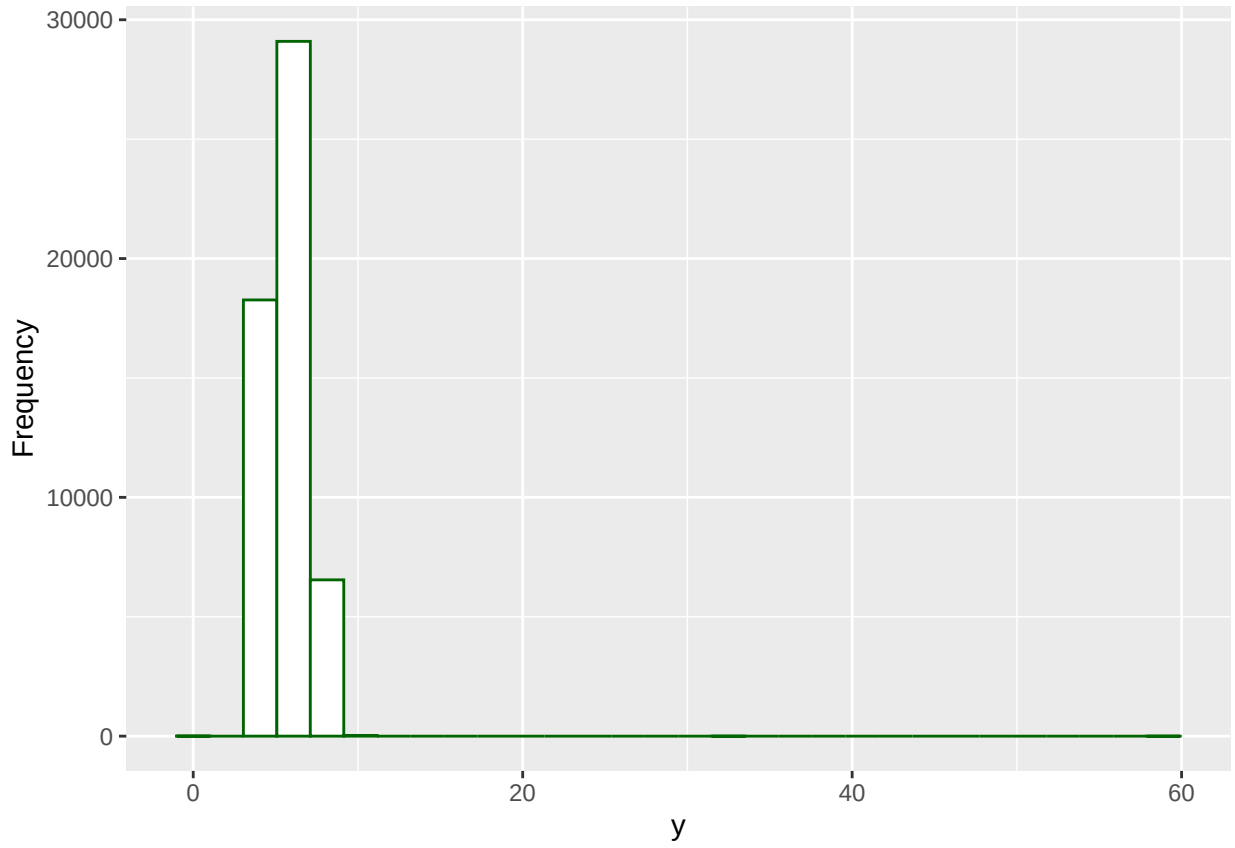

```
ggplot(continuous_df, aes(continuous_df[, 5])) +
  geom_histogram(color='darkgreen', fill='white') +
  labs(x = names(continuous_df)[5], y = "Frequency")
```

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



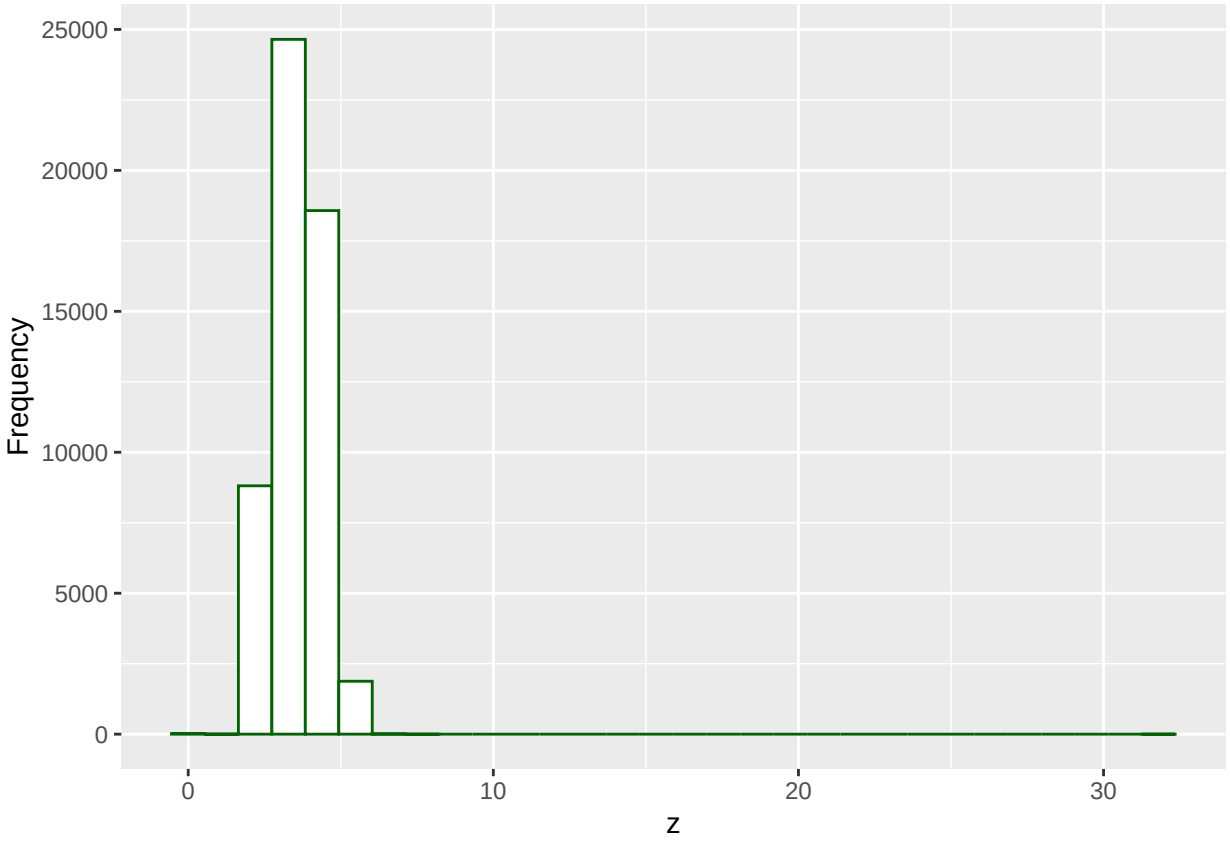

```
ggplot(continuous_df, aes(continuous_df[, 6])) +
  geom_histogram(color='darkgreen', fill='white') +
  labs(x = names(continuous_df)[6], y = "Frequency")
```

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




```
ggplot(continuous_df, aes(continuous_df[, 7])) +
  geom_histogram(color='darkgreen', fill='white') +
  labs(x = names(continuous_df)[7], y = "Frequency")
```

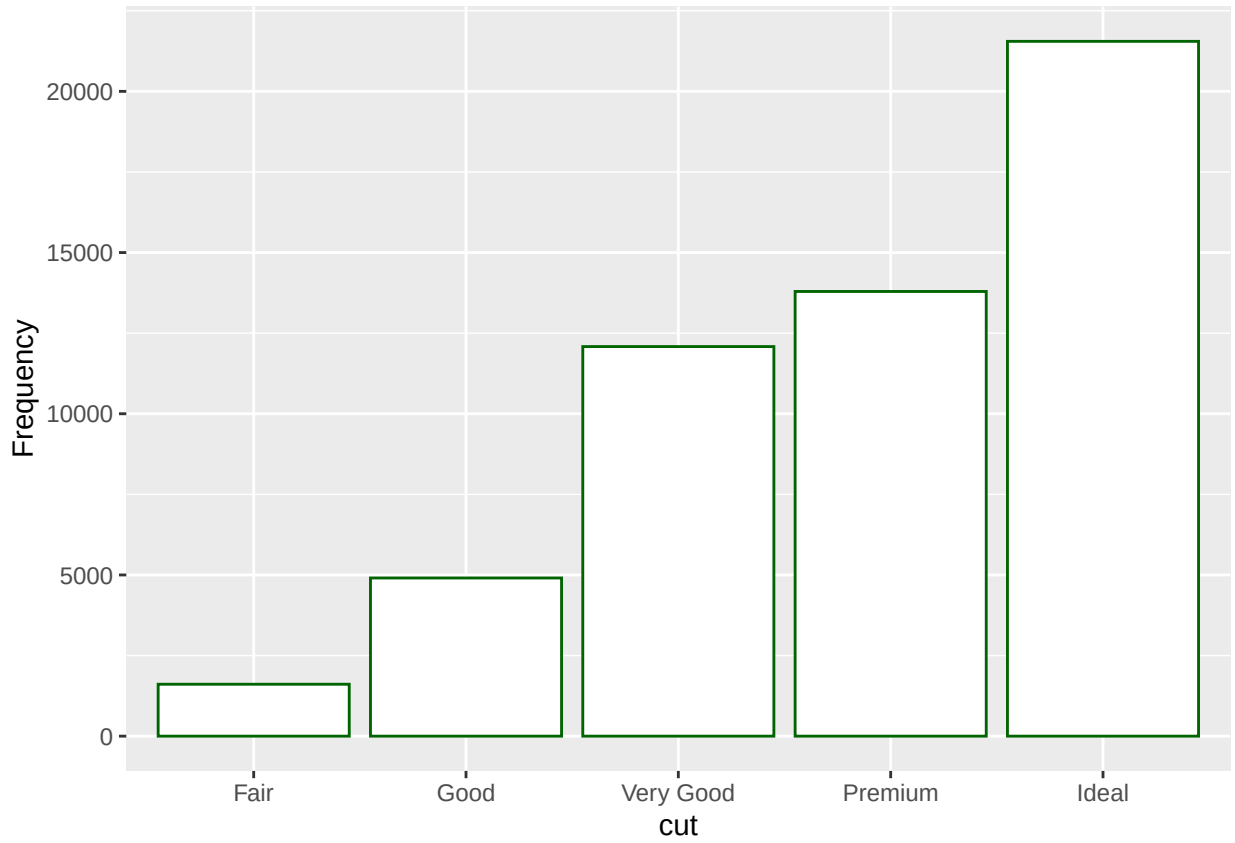
```
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```



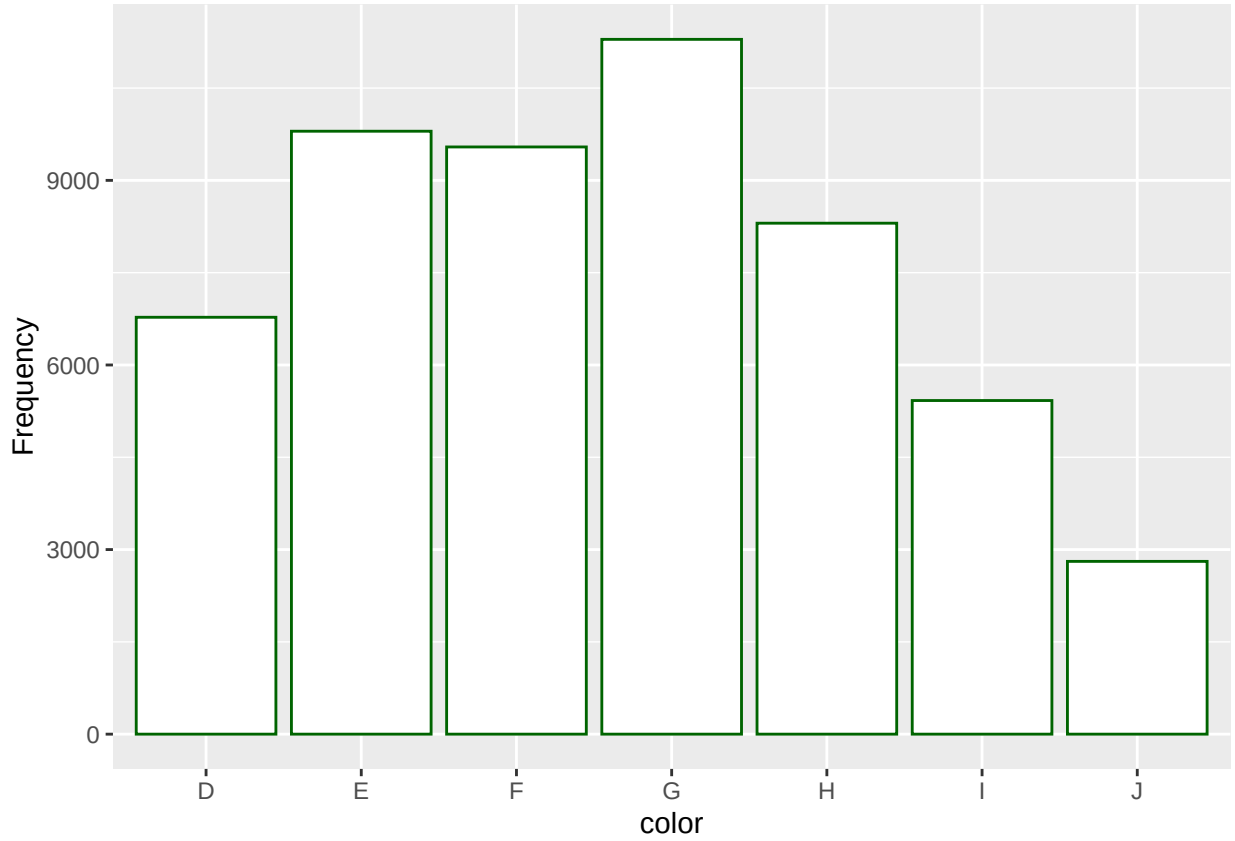
Generally, it seems like almost all of the variables seem relatively clustered around a mean (somewhat normal) with the exception of the carat and the price variable which seem very skewed left.

Creating bar plots for the categorial variables

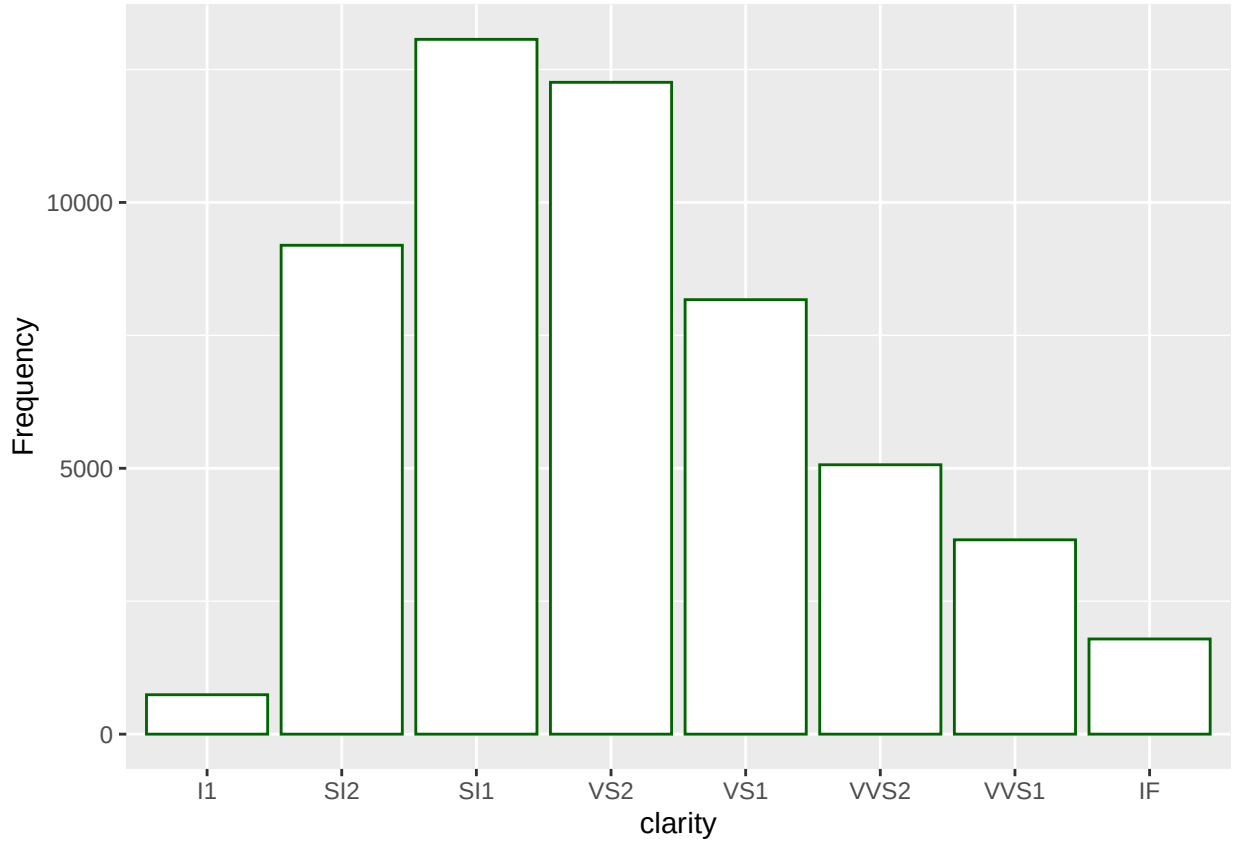
```
categorical_df <- df[, (2:4)]

#putting the categories in order from least "good" to most good.
#color already in order as it is alphabetical.
order_cut <- c("Fair", "Good", "Very Good", "Premium", "Ideal")
order_clarity <- c("I1", "SI2", "SI1", "VS2", "VS1", "VVS2", "VVS1", "IF")
categorical_df$cut <- factor(categorical_df$cut, levels = order_cut,
                             ordered = TRUE)
categorical_df$clarity <- factor(categorical_df$clarity,
                                 levels = order_clarity, ordered = TRUE)

ggplot(categorical_df, aes(categorical_df[, 1])) +
  geom_bar(data = categorical_df, color='darkgreen', fill='white') +
  labs(x = names(categorical_df)[1], y = "Frequency")
```




```
ggplot(categorical_df, aes(categorical_df[, 2])) +
  geom_bar(data = categorical_df, color='darkgreen', fill='white') +
  labs(x = names(categorical_df)[2], y = "Frequency")
```




```
ggplot(categorical_df, aes(categorical_df[, 3])) +
  geom_bar(data = categorical_df, color='darkgreen', fill='white') +
  labs(x = names(categorical_df)[3], y = "Frequency")
```



The cut variable seems to be skewed right while both color and clarity seem to be somewhat normal as they cluster around a mean near the center.

3.

```
model <- lm(price ~ depth + table + carat + color + clarity, data = df)
vif(model)
```

```
##             GVIF Df GVIF^(1/(2*Df))
## depth   1.125178  1        1.060744
## table   1.163072  1        1.078458
## carat   1.319929  1        1.148882
## color   1.165631  6        1.012854
## clarity 1.254320  7        1.016317
```

The VIF value for each variable heavily points to low or non-existent multicolinearity (A predictor depending on another one).

4.

```
summary(model)
```

```
##
```

```
## Call:
## lm(formula = price ~ depth + table + carat + color + clarity,
##     data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -17232.1   -679.3   -195.3    469.2  10258.3
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   498.886    309.791    1.61    0.107
## depth         -65.860      3.710  -17.75   <2e-16 ***
## table         -53.469      2.418  -22.11   <2e-16 ***
## carat        8890.888     12.144  732.10   <2e-16 ***
## colorE       -211.495     18.422  -11.48   <2e-16 ***
## colorF       -311.697     18.615  -16.74   <2e-16 ***
## colorG       -506.790     18.228  -27.80   <2e-16 ***
## colorH       -976.653     19.388  -50.38   <2e-16 ***
## colorI      -1433.205     21.783  -65.80   <2e-16 ***
## colorJ      -2323.593     26.889  -86.42   <2e-16 ***
## clarityIF    5569.812     52.088  106.93   <2e-16 ***
## claritySI1   3718.752     44.392   83.77   <2e-16 ***
## claritySI2   2757.077     44.676   61.71   <2e-16 ***
## clarityVS1   4682.519     45.379  103.19   <2e-16 ***
## clarityVS2   4370.979     44.648   97.90   <2e-16 ***
## clarityVVS1  5231.069     48.047  108.88   <2e-16 ***
## clarityVVS2  5120.628     46.720  109.60   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1164 on 53926 degrees of freedom
## Multiple R-squared:  0.9149, Adjusted R-squared:  0.9149
## F-statistic: 3.625e+04 on 16 and 53926 DF,  p-value: < 2.2e-16
```

5.

The data seemed to behave and follow a distribution I expected. However, the fact that none of the variables have much colinearity at all is somewhat surprising to me as there are definitely some variables I expected to have some relationship.

Part 2: Simple Linear Regression

1 & 2

```r
carat_mean <- 7756.44
simple_model <- lm(price ~ carat, data = df)
print(summary(simple_model))
```

```
##
## Call:
## lm(formula = price ~ carat, data = df)
##
## 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
```

```
print(confint(simple_model))
```

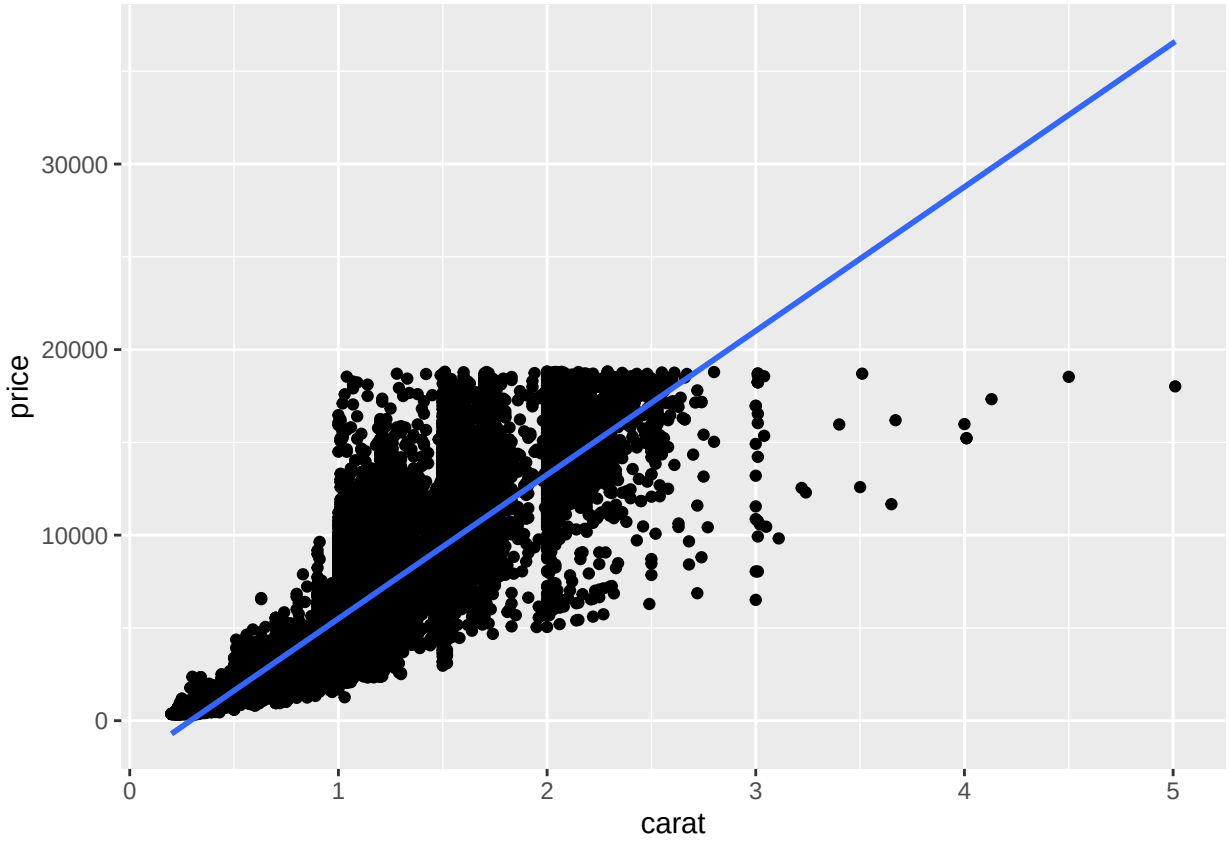
```
##                 2.5 %    97.5 %
## (Intercept) -2281.983 -2230.807
## carat        7728.866  7784.006
```

```
ggplot(data = df, aes(x = carat, y = price)) + geom_point() +
  geom_smooth(method = lm)
```

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

```
print(predict(simple_model, data.frame(carat = carat_mean), interval = "prediction", level = 0.95))
```

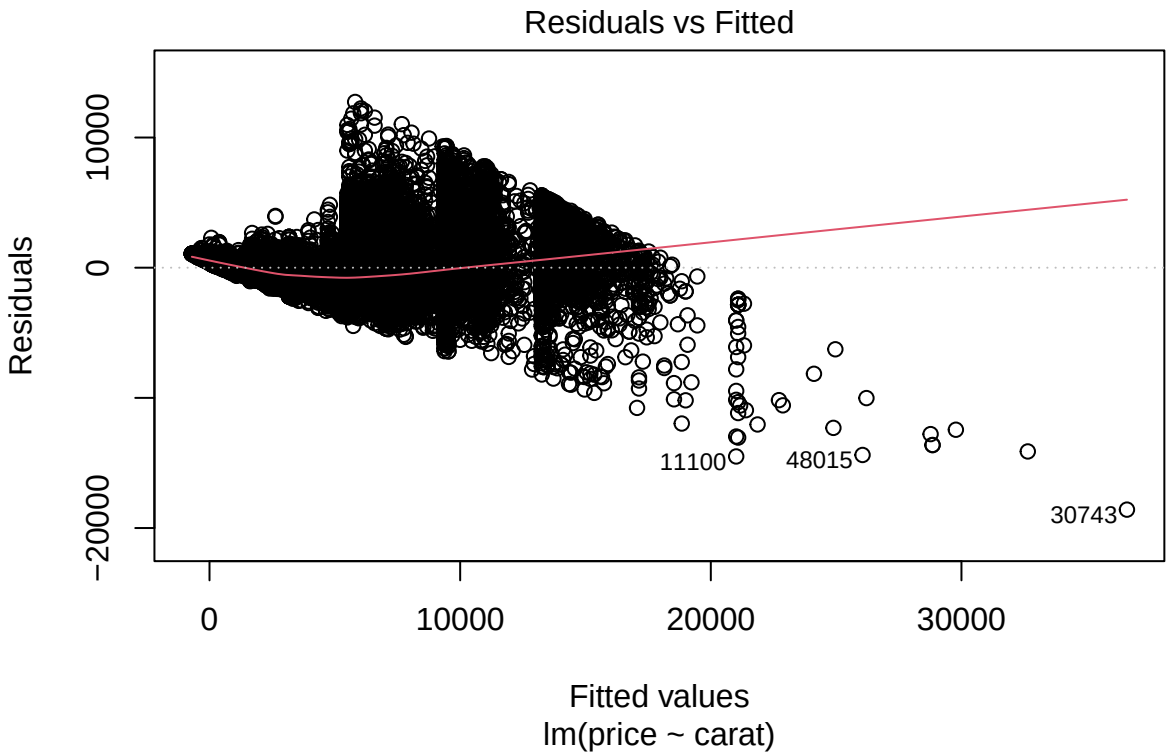
```
##        fit      lwr      upr
## 1 60160075 59946232 60373919
```



For the residuals, it seems that most of the actual values are below the estimator. The estimate value for the intercept (β_0) (the price) is extremely low while the estimate for carat (β_1) is significantly higher. The model itself and the predictor have a very low P value which suggests that the predictor has a high statistical significance to the model. Given the R^2_{adj} (and the regular R^2) the large majority of the total variation is explained with the model. The confidence interval for both the intercept and the carat is very narrow, covering only a difference in upper and lower quantities of 51.176 and 55.140 respectively. The prediction interval is a much wider interval with a difference between the high and low of 427687.

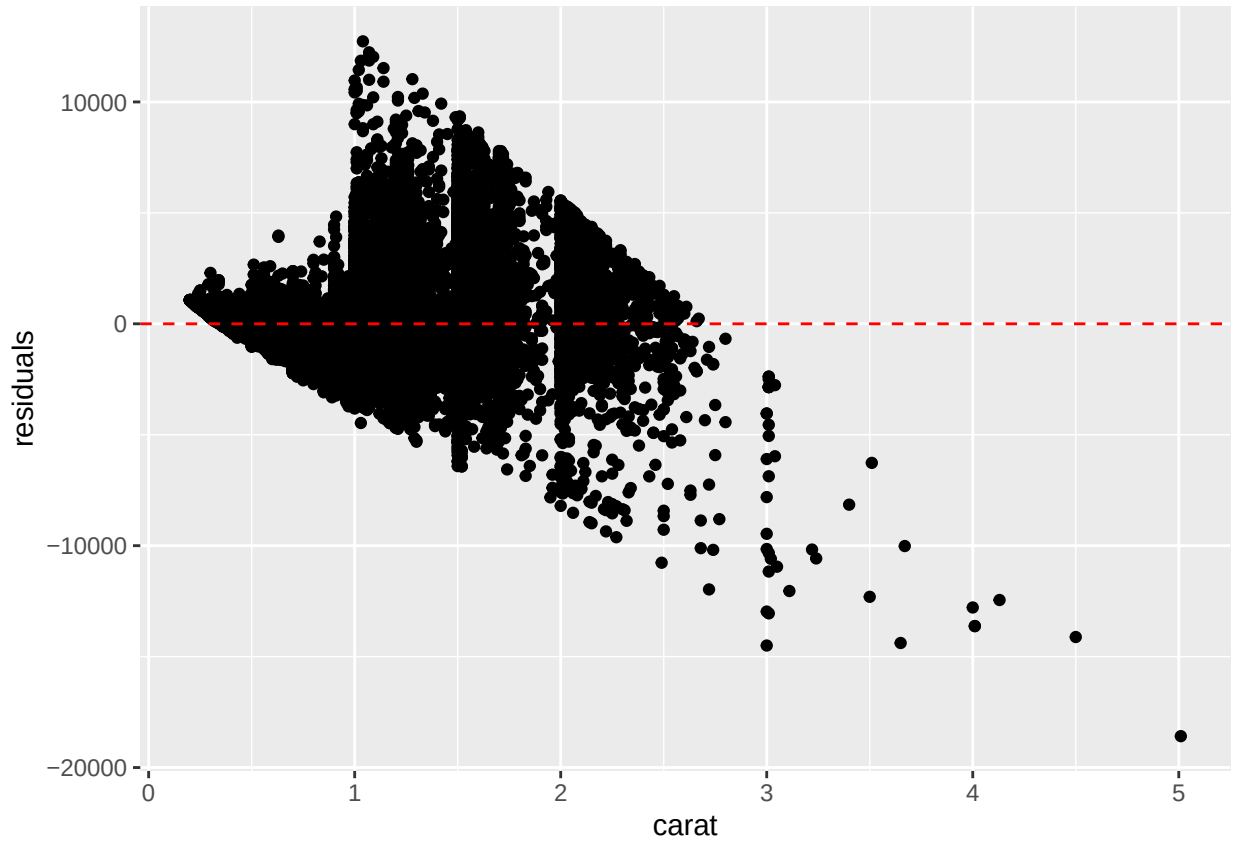
3.

```
plot(simple_model, which = 1)
```



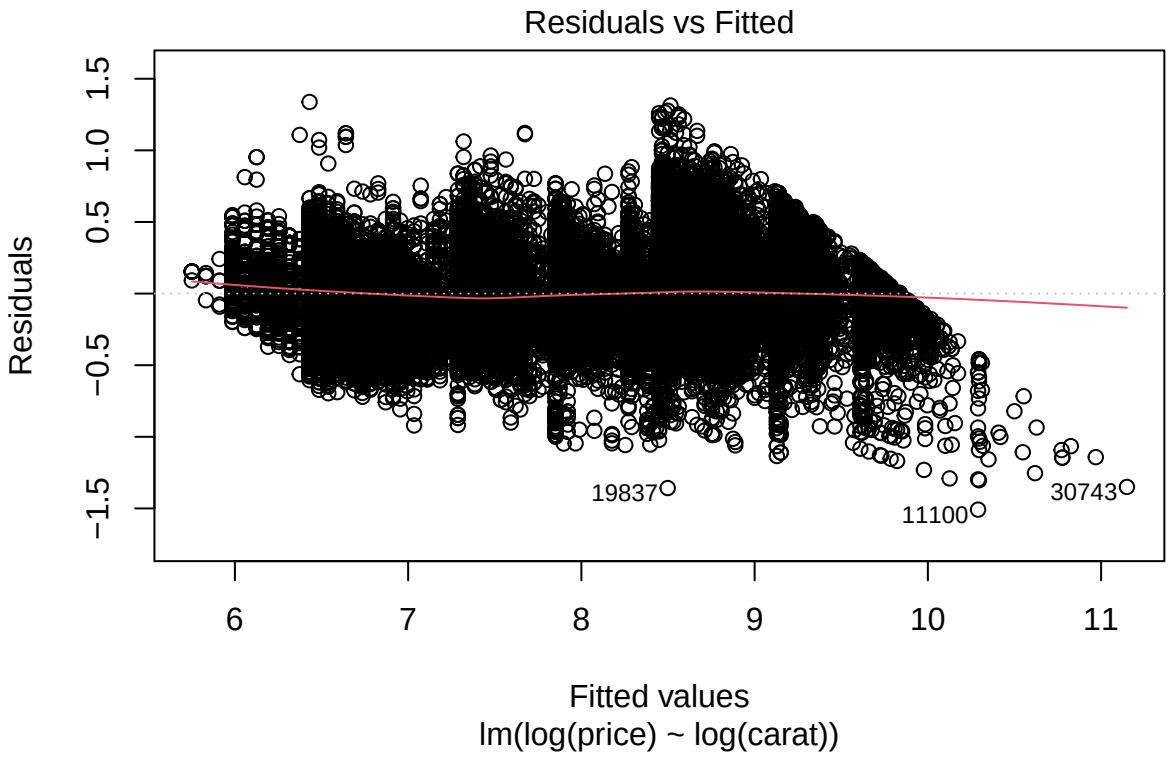
from the residuals vs fitted plot, it is clear that the points do not follow a linear distribution.
To fix this, we first check the residuals vs predictors plot:

```
df$residuals <- residuals(simple_model)

ggplot(df, aes(x = carat, y = residuals)) + geom_point() +
  geom_hline(yintercept = 0, col = "red", lty = 2)
```



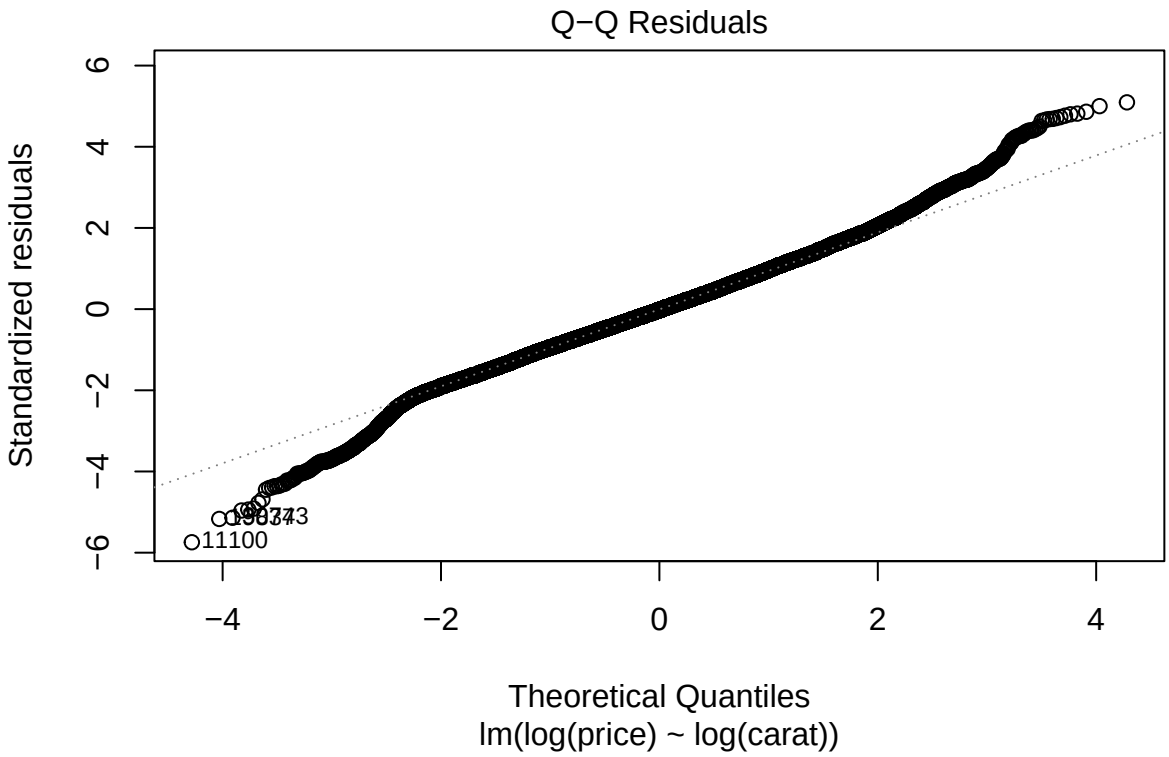
Clearly, the resiudals plotted against the carat show a non linear relationship
To fix, we try different ways to make the data cluster around the mean more.

```
fix_simp_model <- lm(log(price) ~ log(carat), data = df)
plot(fix_simp_model, which = 1)
```



After the transformation, the data appears to be much more linear in nature.
Now, we plot the quantile-quantile plot of residuals:

```
plot(fix_simp_model, which = 2)
```



The Q-Q plot falls within an acceptable visual range of deviation from the mean which suggests normality.

4.

```
summary(fix_simp_model)
```

```
##
## Call:
## lm(formula = log(price) ~ log(carat), data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5083 -0.1695 -0.0059  0.1664  1.3379
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 8.448664   0.001365  6191.2   <2e-16 ***
## log(carat)  1.675817   0.001934   866.6   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2627 on 53941 degrees of freedom
## Multiple R-squared:  0.933,  Adjusted R-squared:  0.933
## F-statistic: 7.51e+05 on 1 and 53941 DF,  p-value: < 2.2e-16
```

Since P value stays very small and therefore statistically significant, the most notable difference now is the R^2_{adj} and normal R^2 which now suggests that the model explains a huge 93.3% of the overall variation in the data.

5.

From the code used to conduct the best possible regression model, the best combination is a model predicting the price based on the the log of carat and the levels of color and clarity which boosts the R^2_{adj} and regular R^2 to over 98%.

6.

The most interesting thing I found in this part was just how good of a model that was able to be made. With an R^2_{adj} vlaue of over 98%, the error in the model accounts for under 2% of the total variation. A model of this accuracy is a significant achievement.

Part 3: Multiple Linear Regression

1.

```
last_model1 <- lm(log(price) ~ log(carat) + color + clarity + depth, data = df)
last_model2 <- lm(log(price) ~ log(carat) + color + clarity + table, data = df)
last_model3 <- lm(log(price) ~ log(carat) + color + clarity + cut, data = df)
last_model4 <- lm(log(price) ~ log(carat) + color + clarity + x, data = df)
last_model5 <- lm(log(price) ~ log(carat) + color + clarity + y, data = df)
last_model6 <- lm(log(price) ~ log(carat) + color + clarity + z, data = df)


AIC(last_model1, last_model2, last_model3, last_model4, last_model5, last_model6)
```

```
##             df       AIC
## last_model1 17 -60943.56
## last_model2 17 -61075.51
## last_model3 20 -63899.32
## last_model4 17 -61233.22
## last_model5 17 -60821.90
## last_model6 17 -60706.49
```

```
BIC(last_model1, last_model2, last_model3, last_model4, last_model5, last_model6)
```

```
##             df       BIC
## last_model1 17 -60792.34
## last_model2 17 -60924.28
## last_model3 20 -63721.41
## last_model4 17 -61081.99
## last_model5 17 -60670.67
```

```
## last_model6 17 -60555.26
```

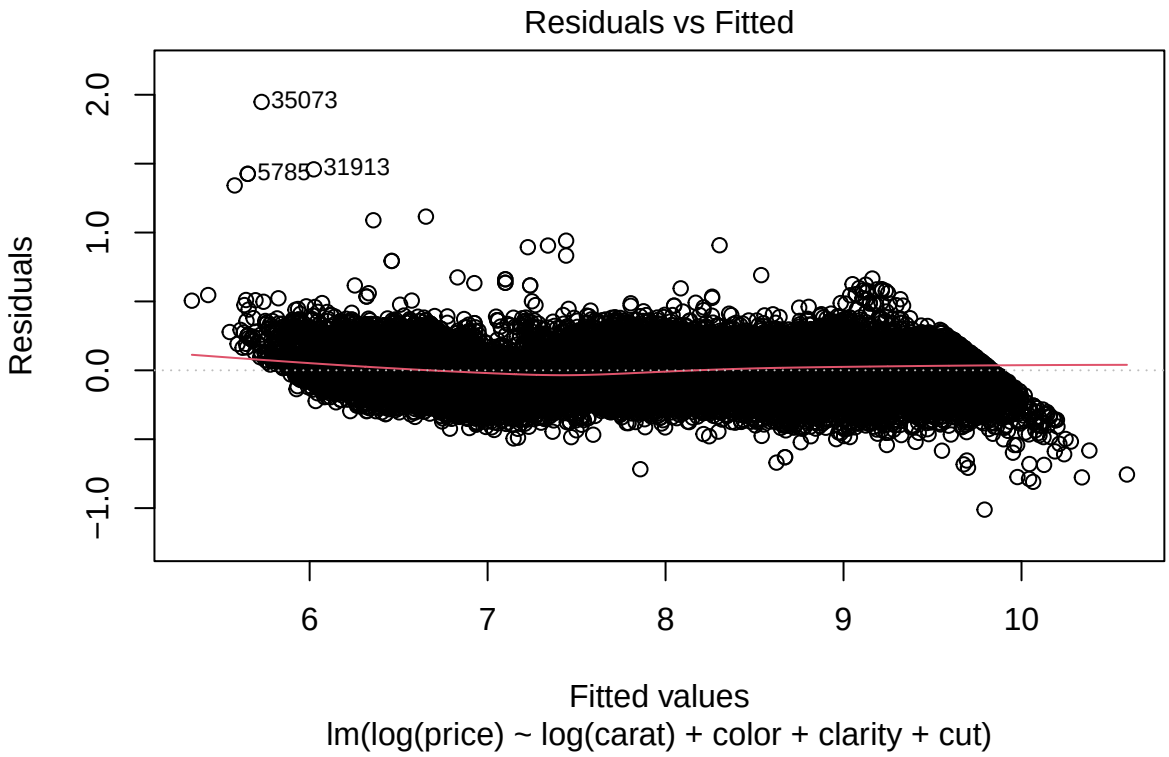
```
fin_model <- last_model3
summary(fin_model)
```

```
##
## Call:
## lm(formula = log(price) ~ log(carat) + color + clarity + cut,
##     data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.01106 -0.08635 -0.00022  0.08340  1.94777
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.856853   0.005758 1364.46   <2e-16 ***
## log(carat)    1.883716   0.001129 1668.79   <2e-16 ***
## colorE       -0.054280   0.002118  -25.62   <2e-16 ***
## colorF       -0.094587   0.002142  -44.16   <2e-16 ***
## colorG       -0.160377   0.002097  -76.49   <2e-16 ***
## colorH       -0.251071   0.002225 -112.85   <2e-16 ***
## colorI       -0.372573   0.002492 -149.51   <2e-16 ***
## colorJ       -0.510982   0.003074 -166.24   <2e-16 ***
## clarityIF     1.113731   0.006030  184.70   <2e-16 ***
## claritySI1    0.592963   0.005149  115.17   <2e-16 ***
## claritySI2    0.427879   0.005178   82.64   <2e-16 ***
## clarityVS1    0.812277   0.005257  154.53   <2e-16 ***
## clarityVS2    0.742158   0.005177  143.34   <2e-16 ***
## clarityVVS1   1.018743   0.005575  182.74   <2e-16 ***
## clarityVVS2   0.947272   0.005418  174.83   <2e-16 ***
## cutGood       0.080048   0.003890   20.58   <2e-16 ***
## cutIdeal      0.161218   0.003548   45.44   <2e-16 ***
## cutPremium    0.139353   0.003579   38.94   <2e-16 ***
## cutVery Good  0.117209   0.003619   32.39   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1338 on 53924 degrees of freedom
## Multiple R-squared:  0.9826, Adjusted R-squared:  0.9826
## F-statistic: 1.693e+05 on 18 and 53924 DF,  p-value: < 2.2e-16
```

From this, we are able to see that the best model for predicting the price is a model that includes the log of carat, the color, the clarity, and the cut. Based on the VIF of the model, there appears to be little multicolinearity thus the variables do not depend on each other at all.

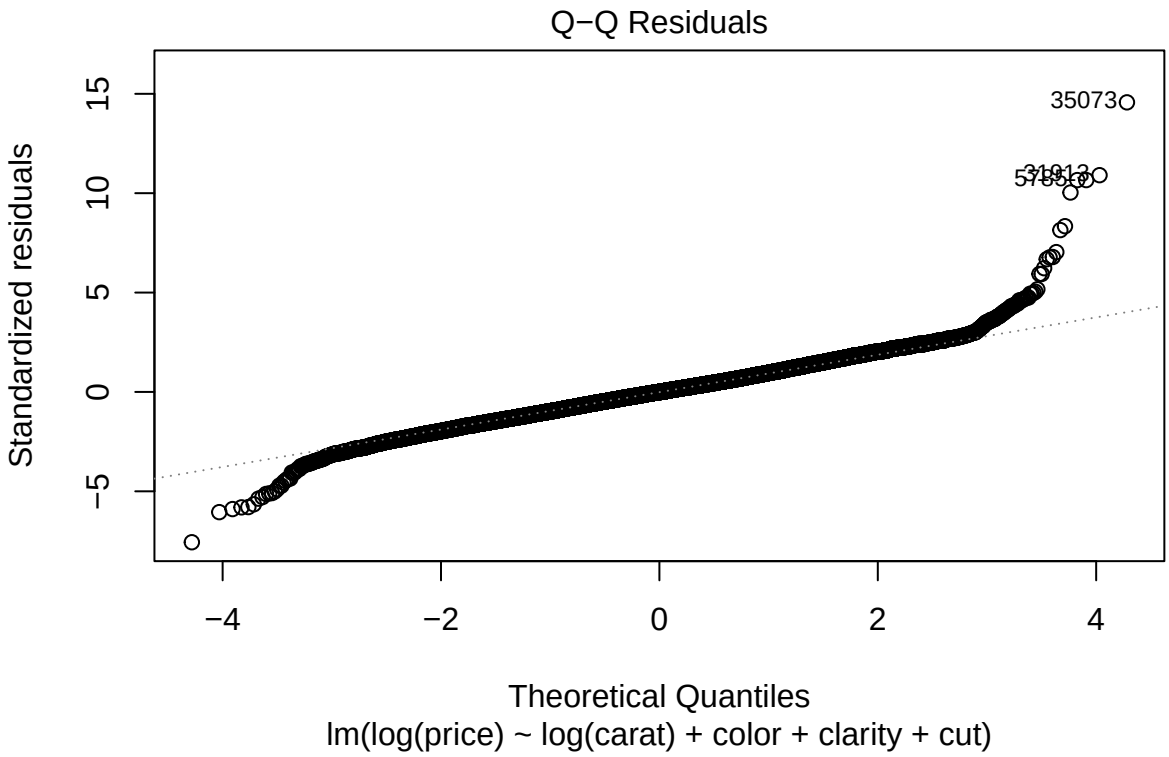
We also must ensure that the assumptions for a linear regression model hold for this particular one. We will check this visually with plots. First, we do residuals vs fitted

```
plot(fin_model, which = 1)
```



As expected (since cut is a categorical variable), the model still displays linearity and passes this check. Next, we check the Q-Q plot of the residuals which still should still be a fairly straight line with most points clustered around a given slope.

```
plot(fin_model, which = 2)
```



despite the uneven look of the tails of the plot, the overall shape of it is still highly normal and suggests that the error is normally distributed.

2.

```
vif(fin_model)
```

```
##                GVIF Df GVIF^(1/(2*Df))
## log(carat) 1.313054  1        1.145885
## color      1.139971  6        1.010977
## clarity    1.323330  7        1.020212
## cut        1.105627  4        1.012631
```

The VIF values being very close to 1 indicate that the individual coefficients have very low multicolinearity which suggests that none of them are dependent on each other (with makes intuitive sense as the color, cut, clarity, and carat have no affect on each other).

3.

```
print(confint(fin_model))
```

```
##                   2.5 %      97.5 %
```

```
## (Intercept)   7.84556670  7.86813903
## log(carat)    1.88150394  1.88592883
## colorE       -0.05843226 -0.05012870
## colorF       -0.09878570 -0.09038900
## colorG       -0.16448664 -0.15626783
## colorH       -0.25543157 -0.24671042
## colorI       -0.37745774 -0.36768901
## colorJ       -0.51700678 -0.50495752
## clarityIF     1.10191233  1.12555000
## claritySI1    0.58287163  0.60305416
## claritySI2    0.41773085  0.43802746
## clarityVS1    0.80197454  0.82258033
## clarityVS2    0.73201022  0.75230587
## clarityVVS1   1.00781576  1.02966971
## clarityVVS2   0.93665195  0.95789128
## cutGood       0.07242258  0.08767305
## cutIdeal      0.15426425  0.16817212
## cutPremium    0.13233853  0.14636664
## cutVery Good  0.11011682  0.12430168
```

```
print(predict(fin_model, data.frame(carat = 2, color = "E", clarity = "IF", cut = "Fair"), interval = "
```

```
##      fit      lwr      upr
## 1 10.222 9.959552 10.48444
```

All of the confidence intervals for the means are extremely small which indicates that the predictors work very well at giving a statistically sound estimate of the data. Since we must keep in mind that the response has the log taken, the real values of the prediction interval for a 2 carat diamond of the highest clarity, most pristine color, and worst cut that the data set offers is between $21153.32 and $35754.81 with an exact value put at $27501.61.

4.

Throughout this report, we have looked at a data set representing different attributes of a diamond including carat, cut, color, clarity, depth, table, price, x, y, and z. From these predictors, we examined their distribution by checking their summary statistics and generating plots in order to visually analyze their distributions. After this, we chose the price, depth, table, carat, color, clarity variables to make a model from. First, we tested a simple regression model that modeled the price based only off of the carat of the diamond which for being simple had a rather high accuracy. The model was not linear at first but through a logistic transformation, we were able to better display the data as linear and was able to improve upon the model from there. Next, I tested the statistical significance of each of the predictors paired with the log of the carat predictor and found that the color and clarity were the only predictors that increased the accuracy by a relatively significant amount. After making this model, we then tested it along with the other predictors (depth, table, cut, x, y, and z) and used both AIC and BIC to see the most significant addition we could make. From the results of this, it was determined that the most accurate model with the smallest chance of overfitting was a model that predicted the price of a diamond based on the carat, color, clarity, and cut. After determining this, we then checked the multicolinearity of these predictors and also ensured that the final model was still satisfying the assumptions of a linear regression model. Lastly, we checked the confidence interval of the means of the predictors along with the prediction interval where both further reinforced the accuracy of the model.