Analyzing 2022 Diamond Prices With Linear Regression

Cyrus Navasca, Alexander Bloyer, Lucas Webster

December 2024

Introduction

Authored Fall Quarter of 2024 at the University of California, Santa Barbara (UCSB)

Instructor: Dr. Puja Pandey

This project aims to analyze diamond prices in 2022 by utilizing linear regression to identify key predictors. The dataset included attributes such as carat, cut, color, clarity, and physical dimensions, providing a comprehensive view of the elements that contribute to pricing.

Getting Started

First, we will begin by installing the packages needed throughout this project. While `tidyverse` is the most popular of these packages as it includes `ggplot2` and `dplyr` among others, each of the below packages will play a crucial role as we conduct this regression analysis and report the gathered information.

```
# Installing necessary packages
library(tidyverse)
library(skimr)
library(corrplot)
library(knitr)
library(faraway)
library(broom)
library(caret)
library(car)
```

Dataset Information

The dataset used in this project is the "Diamonds Price Dataset" collected by Amirhossein Mirzaei and uploaded to Kaggle. We can observe some basic properties of the dataset below.

```
# Reading in our dataset
diamonds <- read_csv('DiamondsPrices2022.csv') |>
  select(-1)

# Creating a kable of basic summary statistics
basic_stats <- data.frame(
  obs = nrow(diamonds),
  vars = ncol(diamonds),
```

```
  duplicates = sum(duplicated(diamonds)),
  missing = sum(is.na(diamonds))
)

colnames(basic_stats) <- c("Observations", "Variables", "Duplicates", "Missing Values")
kable(basic_stats, caption="Basic Summary of Diamonds Dataset")
```

Table 1: Basic Summary of Diamonds Dataset

Observations	Variables	Duplicates	Missing Values
53943	10	149	0

The response variable in this dataset is `price`, leaving 9 variables as possible predictors. As we can observe, there are no missing values in the dataset, but there are duplicate observations which can be addressed during the Explanatory Data Analysis section.

Variable Information

The **carat** variable is numerical with a minimum of 0.21 and a maximum of 3.5 and is a measurement of size of the diamond.

The **cut** variable is categorical variable with values which can be ordered best to worst by: "Premium", "Very Good", "Good", "Ideal", and "Fair". This measures the quality of how the diamond was cut.

The **color** variable is categorical with values which can be ordered best to worst by: "D, E, F, G, H, I, J" which represents the color of the diamond.

The **clarity** variable is categorical with values which can be ordered best to worst by "IF", "VVS1", "VVS2", "VS1", "VS2", "SI1", "SI2", "I1".

The **depth** variable is numerical with a minimum of 56.9 and a maximum of 66.8 and is a measure of a size of the diamond from the table to the pointed tip of the diamond in millimeters.

The **table** variable is numerical with a minimum of 53 and a maximum of 66 and is a measure of the diamonds table percentage (how much of the width of the diamond is spanned by the table).

The **price** variable is numerical with a minimum of 344 of and a maximum of 18745 of and is a measure of the price of the diamond in dollars.

The **x** variable is numerical with a minimum of 3.88 and a maximum of 9.65 and is a measure of the length of the diamond in millimeters.

The **y** variable is numerical with a minimum of 3.93 and a maximum of 9.59 and is a measure of the width of the diamond in millimeters.

The **z** variable is numerical with a minimum of 2.3 and a maximum of 6.03 and is a measure of the depth of the diamond in millimeters.

Data Preprocessing

To conduct our regression analysis, we will take a random sample of 400 observations from the original dataset to be used for analysis. The purpose of this is to keep the project from becoming computationally expensive as well as prevent jumbled visualizations which may be difficult to interpret. Due to the large sample size taken, we can reasonably expect it to be a valid representation of the population.

```r
# Loading in the dataset and taking a random sample
set.seed(1132024)

diamonds_df <- diamonds |>
  sample_n(size = 400, replace = FALSE)
```

Since our data contains four categorical variables, we will encode them as factors to allow them to be used during analysis throughout this project.

```r
# Encoding variables 'cut', 'clarity' and 'color' as factors
diamonds_df$cut <- factor(diamonds_df$cut,
         levels = c('Fair', 'Good', 'Very Good', 'Premium', 'Ideal'),
         ordered = TRUE)

diamonds_df$clarity <- factor(diamonds_df$clarity,
         levels = c('I1', 'SI2', 'SI1', 'VS2', 'VS1', 'VVS2', 'VVS1', 'IF'),
         ordered = TRUE)

diamonds_df$color <- factor(diamonds_df$color,
         levels = c("D", "E", "F", "G", "H", "I", "J"),
         ordered = TRUE)

diamonds$cut <- factor(diamonds$cut,
         levels = c('Fair', 'Good', 'Very Good', 'Premium', 'Ideal'),
         ordered = TRUE)

diamonds$clarity <- factor(diamonds$clarity,
         levels = c('I1', 'SI2', 'SI1', 'VS2', 'VS1', 'VVS2', 'VVS1', 'IF'),
         ordered = TRUE)

diamonds$color <- factor(diamonds$color,
         levels = c("J", "I", "H", "G", "F", "E", "D"),
         ordered = TRUE)
```

Exploratory Data Analysis (EDA)

We can utilize the same basic summary statistics used on the original data set to examine this random sample:

```r
# Creating a kable of basic summary statistics
sample_stats <- data.frame(
  obs = nrow(diamonds_df),
  vars = ncol(diamonds_df),
  duplicates = sum(duplicated(diamonds_df)),
  missing = sum(is.na(diamonds_df))
)

colnames(sample_stats) <- c("Observations", "Variables", "Duplicates", "Missing Values")
kable(sample_stats, caption="Basic Summary of Diamonds Random Sample")
```

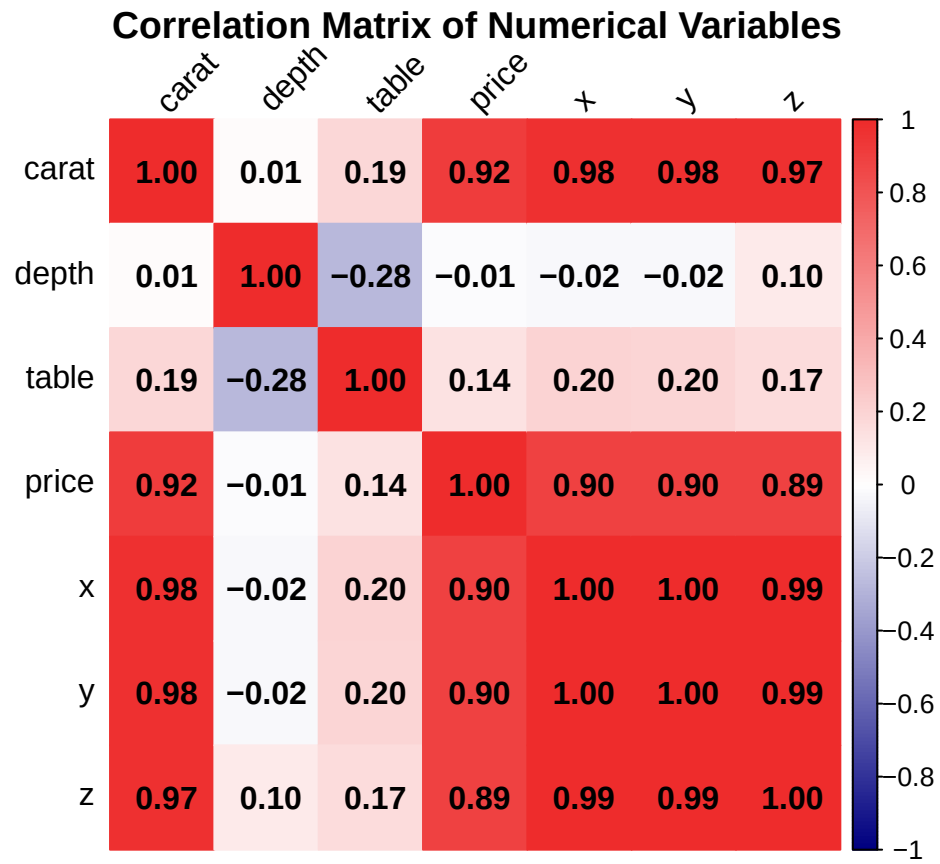
Table 2: Basic Summary of Diamonds Random Sample

Observations	Variables	Duplicates	Missing Values
400	10	0	0

Here, we see that the random sample successfully takes 400 observations and maintains the 10 variables from the original dataset. Luckily, no duplicates are found in this sample which was not the case in the original data meaning no data cleaning is required for this.

Correlation Matrix

```
# Graphic a correlation matrix of numerical variables
num_vars <- diamonds_df |>
  select(carat, depth, table, price, x, y, z)

color_palette <- colorRampPalette(c("firebrick2", "white", "navyblue"))(200)
cor_matrix <- cor(num_vars) |>
  corrplot(title = "Correlation Matrix of Numerical Variables",
           method = "color", mar = c(0, 0, 1, 0), tl.srt = 45,
           col=rev(color_palette), tl.col="black", addCoef.col="black")
```



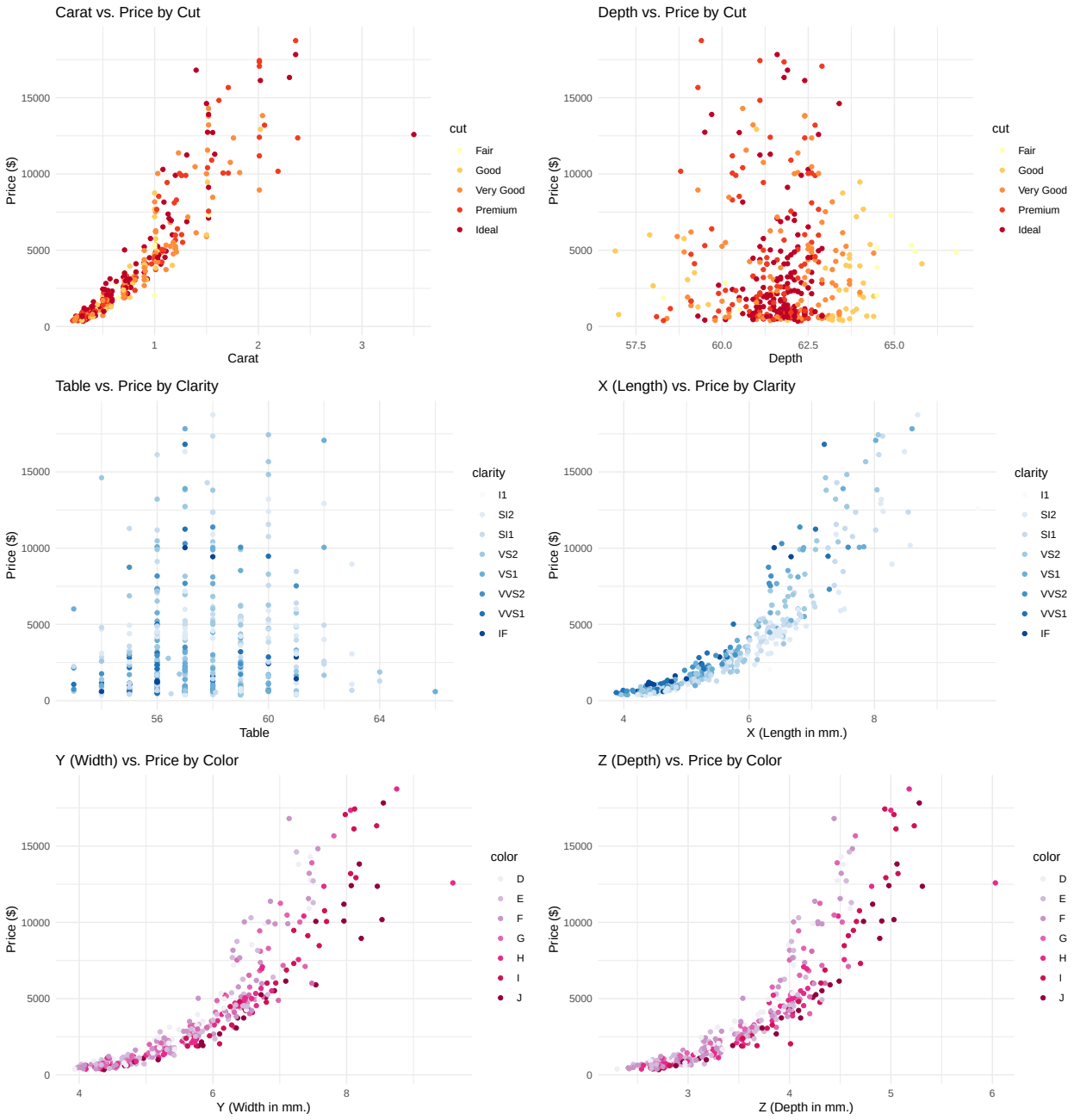
From the above correlation matrix, we observe that several of the dependent variables are highly correlated with each other. The most highly correlated variables are `carat`, `x`, `y`, `z`, and `price`. Each of these variables

have correlations greater than 0.8 with one another, which is a sign of multicolinearity and may increase standard error if included in the linear regression model.

Variable Relationships

Since the goal of this project is to observe which independent variables have effects on `price`, we can utilize scatter plots to directly see the relationships between each numerical variable against price.

```
# Plotting each numerical variable vs price
carat_vs_price <- ggplot(diamonds_df, aes(x=carat, y=price, color=cut)) +
                    geom_point() +
                    scale_color_brewer(palette = "YlOrRd") +
                    labs(x="Carat", y="Price ($)",
                         title="Carat vs. Price by Cut") +
                    theme_minimal()


depth_vs_price <- ggplot(diamonds_df, aes(x=depth, y=price, color=cut)) +
                    geom_point() +
                    scale_color_brewer(palette = "YlOrRd") +
                    labs(x="Depth", y="Price ($)",
                         title="Depth vs. Price by Cut") +
                    theme_minimal()

table_vs_price <- ggplot(diamonds_df, aes(x=table, y=price, color=clarity)) +
                    geom_point() +
                    scale_color_brewer(palette = "Blues") +
                    labs(x="Table", y="Price ($)",
                         title="Table vs. Price by Clarity") +
                    theme_minimal()

x_vs_price <- ggplot(diamonds_df, aes(x=x, y=price, color=clarity)) +
                    geom_point() +
                    scale_color_brewer(palette = "Blues") +
                    labs(x="X (Length in mm.)", y="Price ($)",
                         title="X (Length) vs. Price by Clarity") +
                    theme_minimal()

y_vs_price <- ggplot(diamonds_df, aes(x=y, y=price, color=color)) +
                    geom_point() +
                    scale_color_brewer(palette = "PuRd") +
                    labs(x="Y (Width in mm.)", y="Price ($)",
                         title="Y (Width) vs. Price by Color") +
                    theme_minimal()

z_vs_price <- ggplot(diamonds_df, aes(x=z, y=price, color=color)) +
                    geom_point() +
                    scale_color_brewer(palette = "PuRd") +
                    labs(x="Z (Depth in mm.)", y="Price ($)",
                         title="Z (Depth) vs. Price by Color") +
                    theme_minimal()
```



By observing the scatter plots of each numeric variable against our response variable `price`, we are able to make initial observations on the effects on certain predictors. Specifically, we see that `carat`, `x`, `y` and `z` all have some sort of positive relationship with `price`. It is important to recall from the correlation heatmap that `x`, `y` and `z` were extremely correlated with each other, which may explain why they all seem to have similar relationships with the response variable. However, seeing these relationships gives a good first idea on what predictors may be significant in explaining `price`.

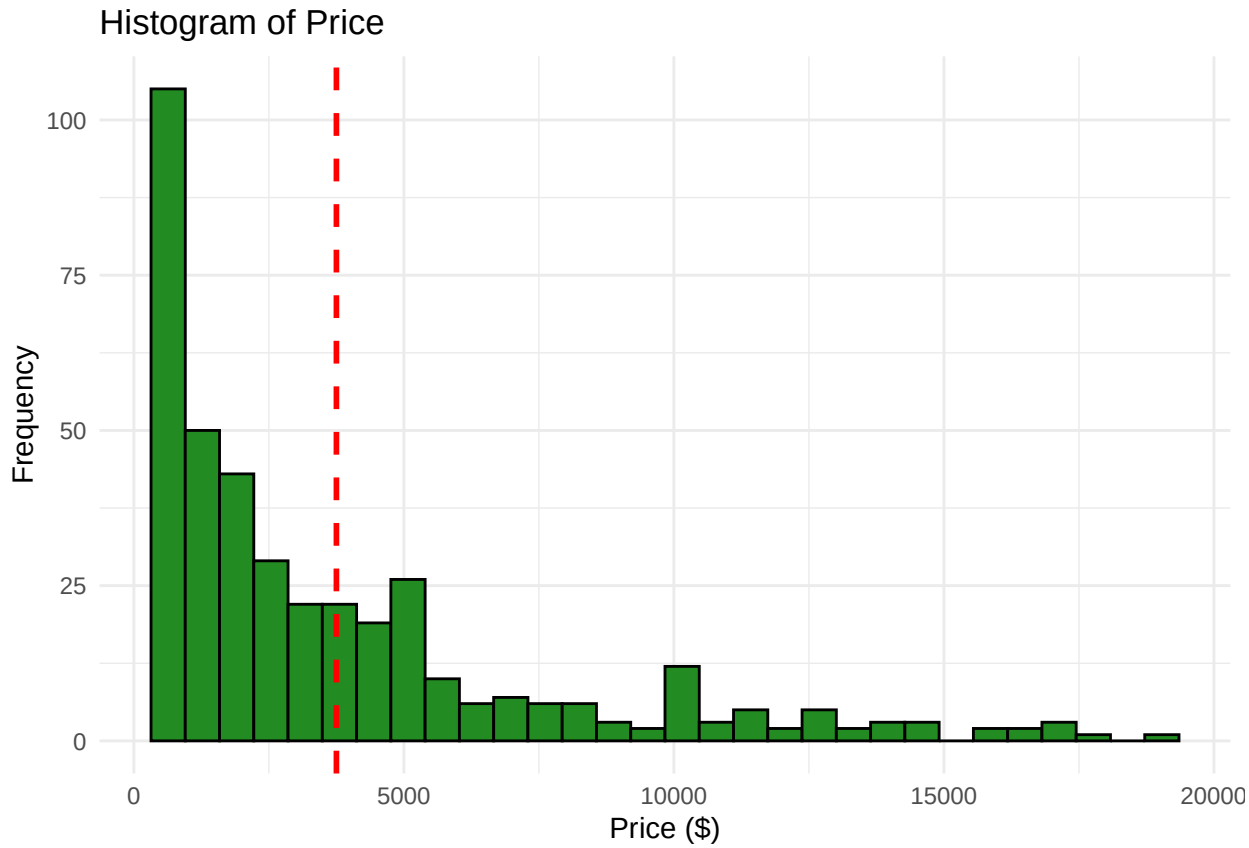
On the other hand, `depth` and `table` do not seem to have any relationship with `price` which is shown by points being randomly scattered. This is as expected, as the correlation heatmap revealed that both of these independent variables had correlations of -0.01 and 0.14 with `price` respectively.

As for the categorical variables which are incorporated via color, we can observe that higher qualities of `cut` seem to match up with higher values of `price`. This does not see, to be the case with `clarity` or `color`, but this is not definitive of anything it can be difficult to observe relationships between numerical and categorical

variables in this way.

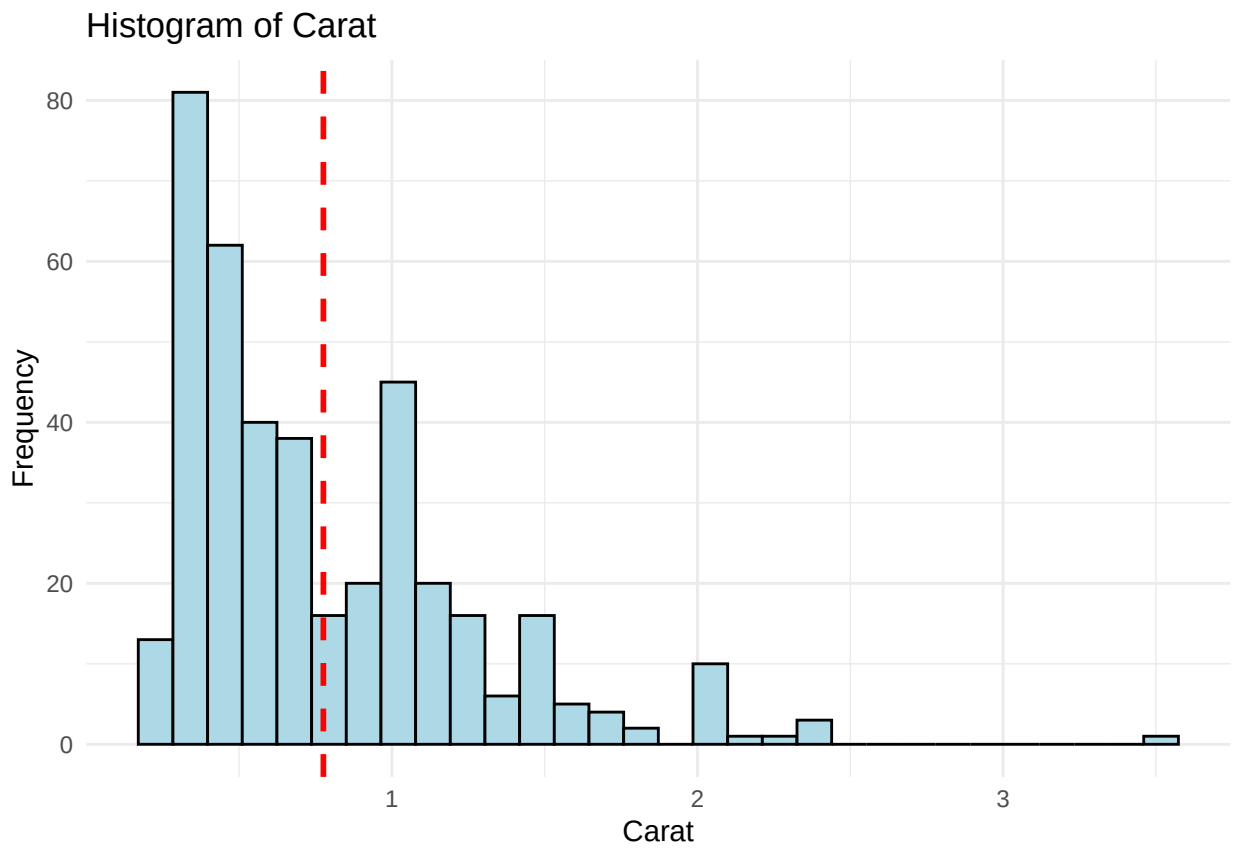
Variable Distributions

```
# Graphic histogram of price
ggplot(diamonds_df) +
  geom_histogram(mapping = aes(x = price), color = "black", fill = "forestgreen") +
  geom_vline(xintercept = mean(diamonds_df$price),
             color = "red", linewidth = 1, linetype = "dashed") +
  ggtitle("Histogram of Price") + xlab("Price ($)") + ylab("Frequency") +
  theme_minimal()
```



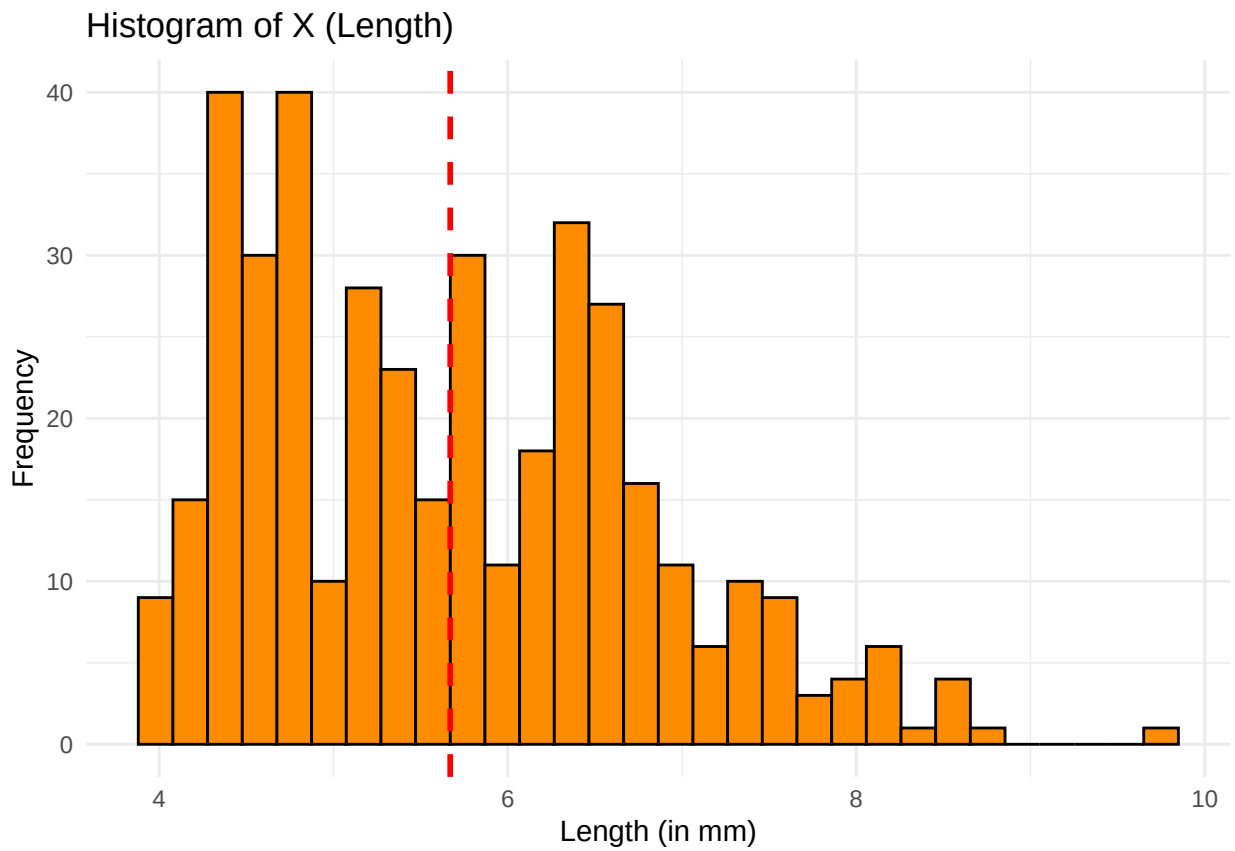
Based on this histogram, we find that in our sample, `price` is highly right skewed with a mean of $3750.60 as denoted by the red dotted line. It is interesting to note that this distribution seems to resemble a Chi-Square distribution with two degrees of freedom.

```
# Graphing histogram of carat
ggplot(diamonds_df) +
  geom_histogram(mapping = aes(x = carat), fill = "lightblue", color = "black") +
  geom_vline(xintercept = mean(diamonds_df$carat),
             color = "red", linewidth = 1, linetype = "dashed") +
  ggtitle("Histogram of Carat") + xlab("Carat") + ylab("Frequency") +
  theme_minimal()
```



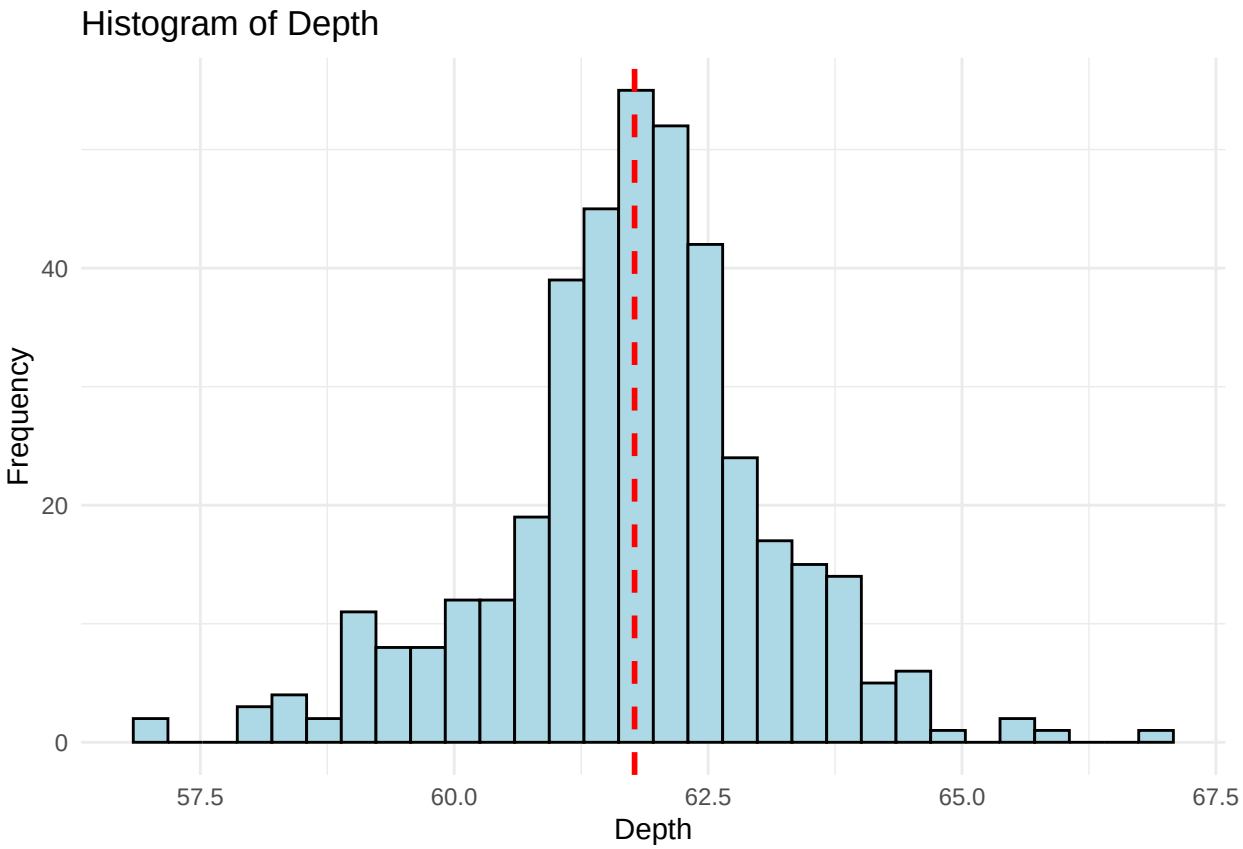
Based on this histogram, we find that `carat` in our sample is skewed to the right, with the average carat being 0.775825 as denoted by the red dotted line. We can attribute the heavy skew to outliers, which seem to fall at about 3.5 carats. We may also note that this implies the carats in our sample not being Normally distributed.

```
# Graphing a histogram of x
ggplot(diamonds_df) +
  geom_histogram(mapping = aes(x = x), color = "black", fill = "darkorange") +
  geom_vline(xintercept = mean(diamonds_df$x),
             color = "red", linewidth = 1, linetype = "dashed") +
  ggtitle("Histogram of X (Length)") + xlab("Length (in mm)") + ylab("Frequency") +
  theme_minimal()
```



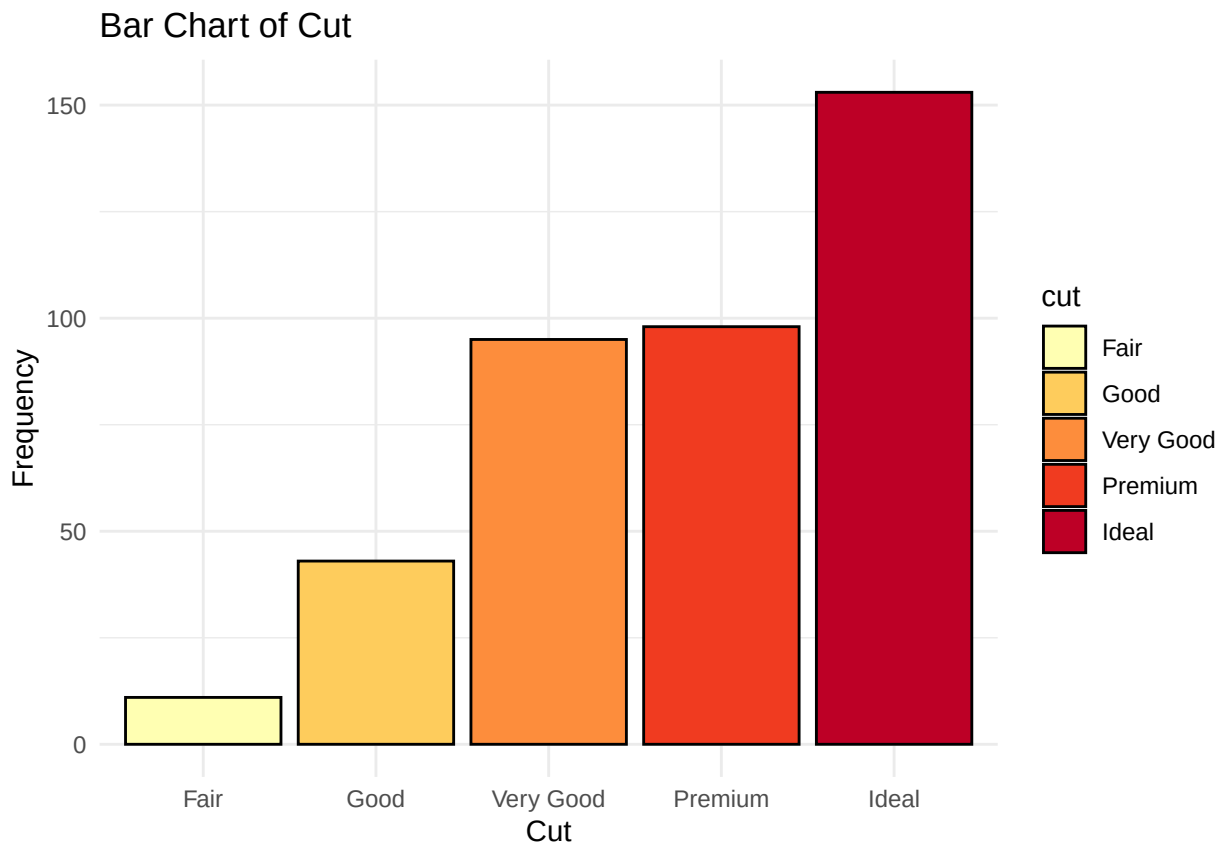
Based on this histogram, we can observe that length, or `x`, in our sample is slightly right skewed and unsymmetrical with a mean of 5.669875 as denoted by the red dotted line. Note that this implies that length in our sample is not Normally distributed.

```
# Graphing histogram of depth
ggplot(diamonds_df) +
  geom_histogram(mapping = aes(x = depth), color = "black", fill = "lightblue") +
  geom_vline(xintercept = mean(diamonds_df$depth),
             color = "red", linewidth = 1, linetype = "dashed") +
  ggtitle("Histogram of Depth") + xlab("Depth") + ylab("Frequency") +
  theme_minimal()
```



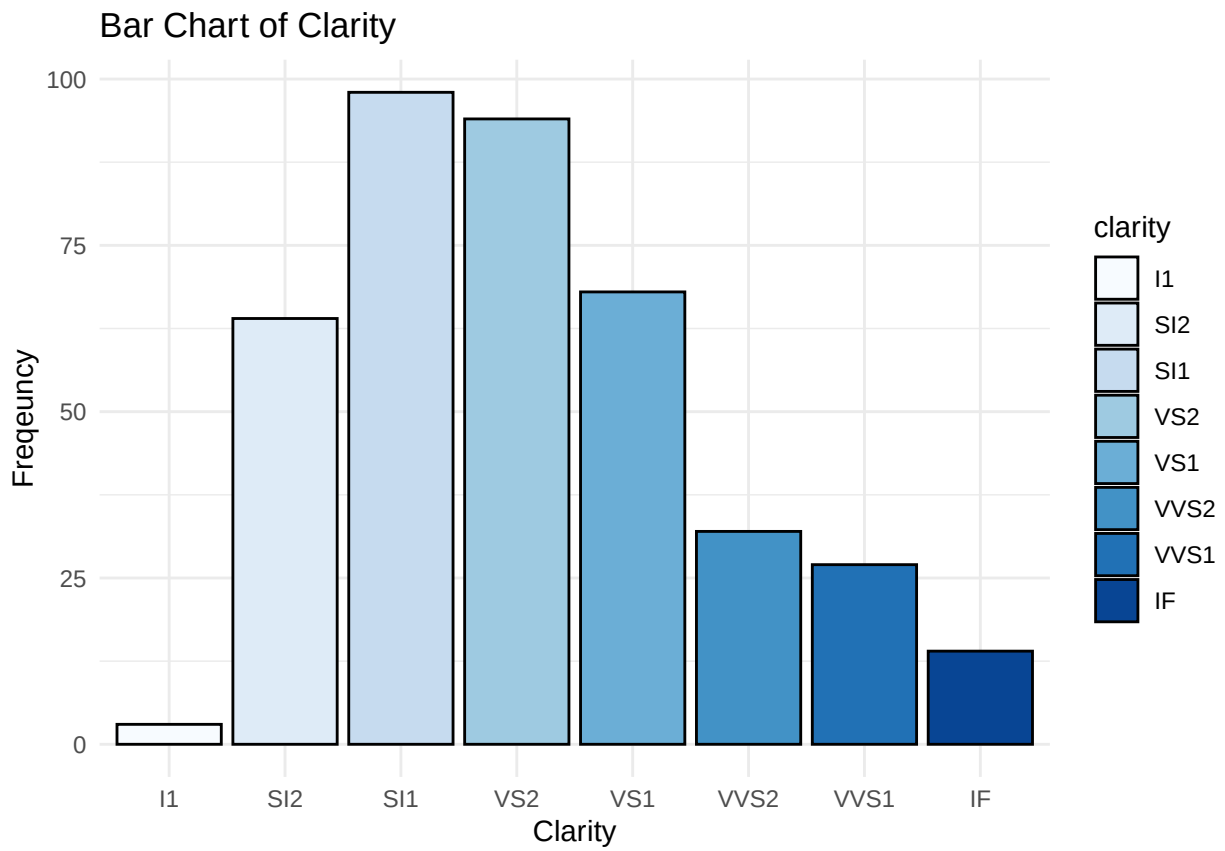
Based on this histogram, we can observe that `depth` seems to be Normally distributed with a mean of 61.77575 as denoted by the red dotted line. We can infer this as the histogram is symmetric, peaks at the mean, and has increasingly less values as we go away from the mean.

```
# Graphic bar chart of cut
ggplot(diamonds_df) +
  geom_bar(mapping = aes(x = cut, fill = cut), color = "black") +
  ggtitle("Bar Chart of Cut") + xlab("Cut") + ylab("Frequency") +
  scale_fill_brewer(palette = 'YlOrRd') +
  theme_minimal()
```



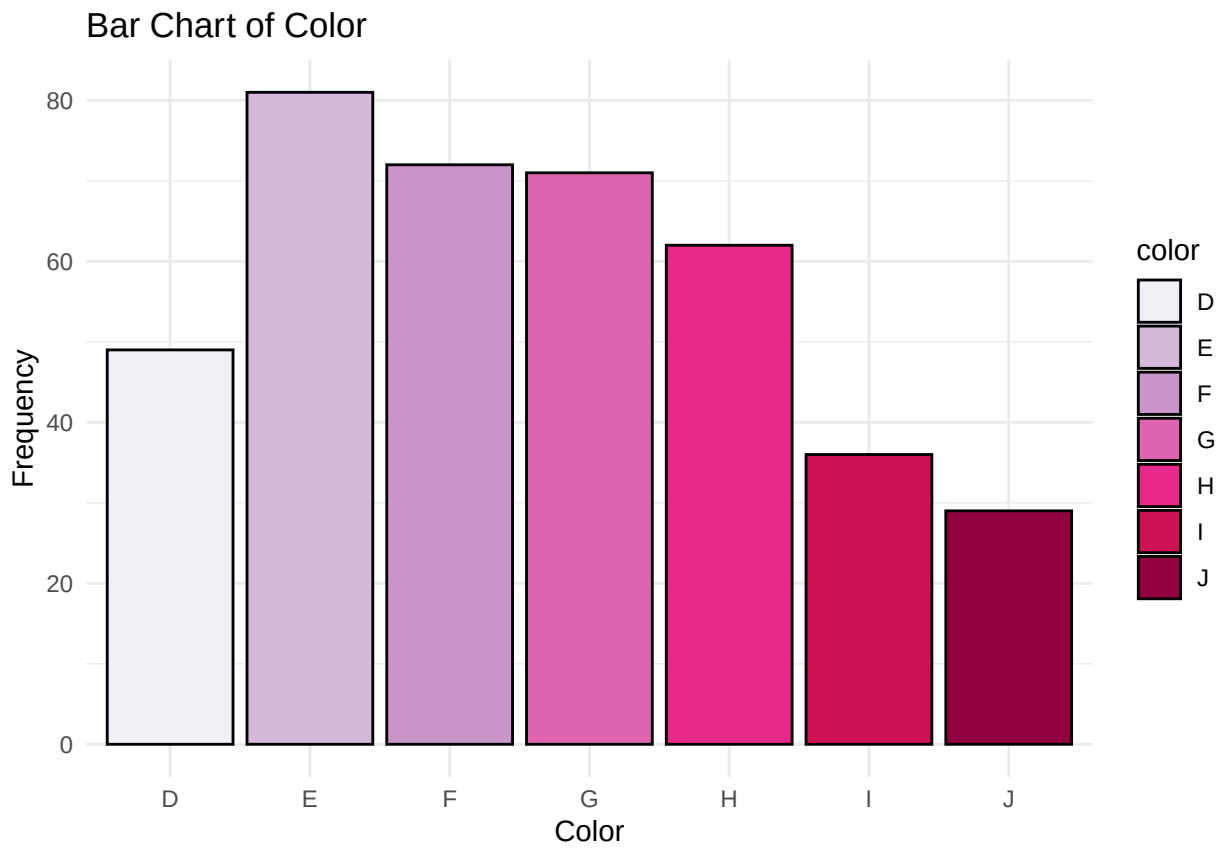
Based on this bar chart, we find that in our sample, higher qualities of cut are more common. Since `cut` is ordinal, we could say that the distribution of cut in our sample is left skewed and not Normally distributed.

```
# Graphic bar chart of clarity
ggplot(diamonds_df) +
  geom_bar(mapping = aes(x = clarity, fill = clarity), color = "black") +
  ggtitle("Bar Chart of Clarity") + xlab("Clarity") + ylab("Freqeuncy") +
  scale_fill_brewer(palette = 'Blues') +
  theme_minimal()
```



Based on this bar chart, we find clarity in our sample to be nearly bimodal with peaks at SI1 and VS2 as well as nearly symmetrical. I1 and IF are the most uncommon clarity levels in this sample with SI1 being the most common. Note that the bimodal nature of this distribution implies that clarity is not Normally distributed.

```
# Graphic bar chart of color
ggplot(diamonds_df) +
  geom_bar(mapping = aes(x = color, fill = color), color = "black") +
  ggtitle("Bar Chart of Color") + xlab("Color") + ylab("Frequency") +
  scale_fill_brewer(palette = 'PuRd') +
  theme_minimal()
```



Based on this bar chart, we can observe that the most common color type in our sample is "E" with the least common being "J". We can also observe that "F", "G" and "H" have around the same amount of occurrences and that "I" has just slightly more occurrences than "J".

Linear Regression

Defining the Full Model

The "full model" is a linear regression model that utilizes all available predictors in our dataset. We can define it for our data as shown below:

```
# Defining the full model
full_model <- lm(price ~ carat + cut + color + clarity + depth + table + x + y + z,
                 data = diamonds_df)

# Creating a kable of the coefficient summary
kable(summary(full_model)[[4]], caption="Coefficient Summary of Full Model")
```

Table 3: Coefficient Summary of Full Model

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-36718.162836	16984.23212	-2.1618971	0.0312564
carat	9447.711742	642.53010	14.7039209	0.0000000
cut.L	627.125695	296.65549	2.1139865	0.0351743

	Estimate	Std. Error	t value	Pr(>|t|)
cut.Q	-250.717480	237.82063	-1.0542293	0.2924549
cut.C	97.177728	207.34262	0.4686819	0.6395686
cut^4	-19.045803	157.50001	-0.1209257	0.9038145
color.L	-1739.231487	208.55636	-8.3393837	0.0000000
color.Q	-533.461998	189.17406	-2.8199532	0.0050575
color.C	-307.336693	180.17307	-1.7057859	0.0888738
color^4	-212.662601	174.25030	-1.2204433	0.2230617
color^5	-389.860217	165.75031	-2.3520934	0.0191831
color^6	190.462144	154.90702	1.2295256	0.2196436
clarity.L	4935.168594	484.16296	10.1931974	0.0000000
clarity.Q	-3080.754300	465.79333	-6.6139940	0.0000000
clarity.C	1963.867956	395.29430	4.9681161	0.0000010
clarity^4	-988.019426	287.45851	-3.4370854	0.0006537
clarity^5	518.032545	222.26300	2.3307188	0.0202963
clarity^6	-210.580837	184.61188	-1.1406678	0.2547342
clarity^7	292.324541	155.50706	1.8798152	0.0609063
depth	564.294141	264.96899	2.1296610	0.0338486
table	3.448973	40.11107	0.0859855	0.9315237
x	2790.309662	1956.48177	1.4261874	0.1546440
y	2387.411566	1951.23300	1.2235400	0.2218920
z	-9164.157647	4358.03736	-2.1028176	0.0361459

By examining the summary of the full model, we can observe that of the numeric features in our dataset, `carat` is the only one with a p-value less than the Bonferroni corrected significance level of $\alpha_{Bonferroni} = \frac{0.05}{5} = 0.01$. This gives us evidence to suggest that `carat` has a statistically significant effect on `price` but do not have the same evidence for `depth`, `table`, `x`, `y` or `z`.

It becomes more tricky to interpret the p-values of `cut`, `color` and `clarity` as they are categorical variables and we are given p-values for each factor (minus the reference) rather than for the predictor itself. Thus, the best we can do for these variables is to examine how many of the factors are statistically significant and make a rough conclusion on the significance of the predictor itself.

When analyzing the p-values for the factors of `cut`, we find that none have a p-value less than $\alpha_{Bonferroni} = 0.01$, which means that we do not have sufficient evidence to suggest that `cut`has a significant effect of `price`.

Furthermore, four out of the six factors for `clarity` have p-values less than $\alpha_{Bonferroni} = 0.01$, which may give slight evidence that `clarity` has a statistically significant effect on `price`, but is not enough to reach a more definite conclusion.

For clarity, four of the seven factors have p-values less than $\alpha_{Bonferroni} = 0.01$ while three of the factors were greater than $\alpha_{Bonferroni} = 0.01$. Because of this nearly even split, it is difficult to make a conclusion on whether or not `clarity` has a statistically significant effect on `price`.

Moving away from p-values, we can calculate the R^2_{adj} of the model to examine its goodness of fit. Calculating this shows that the full model has an R^2_{adj} of `r`round(summary(full_model)$adj.r.squared, 2). `This reveals that the model is able to explain 90% of variation in the dependent variable,`price‘. When $R^2_{adj} > 0.7$, it typically indicates that the model is a good fit which is the case here. However, R^2_{adj} does not tell the full story, and we will continue to improve upon this model in the following sections.

Defining the Null Model

On the contrary, the null model utilizes no predictors and just represents the target variable with a constant intercept. We may initialize it here as:

```r
# Defining the null model
null_model <- lm(price ~ 1,data = diamonds_df)

# Creating a kable of the coefficient summary
kable(summary(null_model)[[4]], caption="Coefficient Summary of Null Model")
```

Table 4: Coefficient Summary of Null Model

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	3750.602	194.0462	19.3284	0

Additionally, we find the adjusted R-squared of the null model to be 0 which is as expected. While this model may seem useless in itself, it will be utilized in our next section in what is called "forward selection" to find the most significant predictors in our data.

Feature Selection

In order to find the features which lead to the best fitting model, we will use both forward and backward selection to minimize both Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC), for a total of four models. Afterwards, we will make a selection from these four models to obtain the best fitting model.

```r
# Using backwards and forwards selection to minimize AIC
aic_backward <- step(full_model, direction = 'backward', trace=0)
aic_forward <- step(null_model, direction = 'forward', trace=0)

# Using backwards and forwards selection to minimize BIC
n <- nrow(diamonds_df)
bic_backward <- step(full_model, k = log(n), direction = 'backward', trace = 0)
bic_forward <- step(null_model, k = log(n), direction = 'forward', trace = 0)

# Creating a data frame for use with knitr
feature_df <- data.frame(
  "AIC" = c(AIC(aic_backward), AIC(aic_forward), AIC(bic_backward), AIC(bic_forward)),
  "BIC" = c(BIC(aic_backward), BIC(aic_forward), BIC(bic_backward), BIC(bic_forward)),
  "R2_adj" = c(round(summary(aic_backward)$adj.r.squared, 2),
               round(summary(aic_forward)$adj.r.squared, 2),
               round(summary(bic_backward)$adj.r.squared, 2),
               round(summary(bic_forward)$adj.r.squared, 2))
  )

rownames(feature_df) <- c("Backward (AIC)", "Forward (AIC)",
                          "Backward (BIC)", "Forward (BIC)")
colnames(feature_df) <- c("AIC", "BIC", "Adjusted R^2")

# Creating a kable of BIC values
kable(feature_df, caption="BIC Feature Selection Statistics")
```

Table 5: BIC Feature Selection Statistics

	AIC	BIC	Adjusted R^2
Backward (AIC)	6841.593	6921.422	0.9
Forward (AIC)	7749.212	7757.195	0.0
Backward (BIC)	6842.629	6906.492	0.9
Forward (BIC)	7749.212	7757.195	0.0

By observing the table of statistics above, we first find that the models found using backward selection R^2_{adj}. Next, we find that the third model in the table minimizes BIC by a great margin and while it does not minimize AIC, it is only greater than the minimum by a small margin. This gives us evidence to conclude that the third model in this table is the best fitting.

The summary for this model is found below as:

```
# Creating a kable of the coefficient summary for the minimum BIC Model
min_model <- bic_backward
kable(summary(min_model)[[4]], caption="Summary of Selected Model")
```

Table 6: Summary of Selected Model

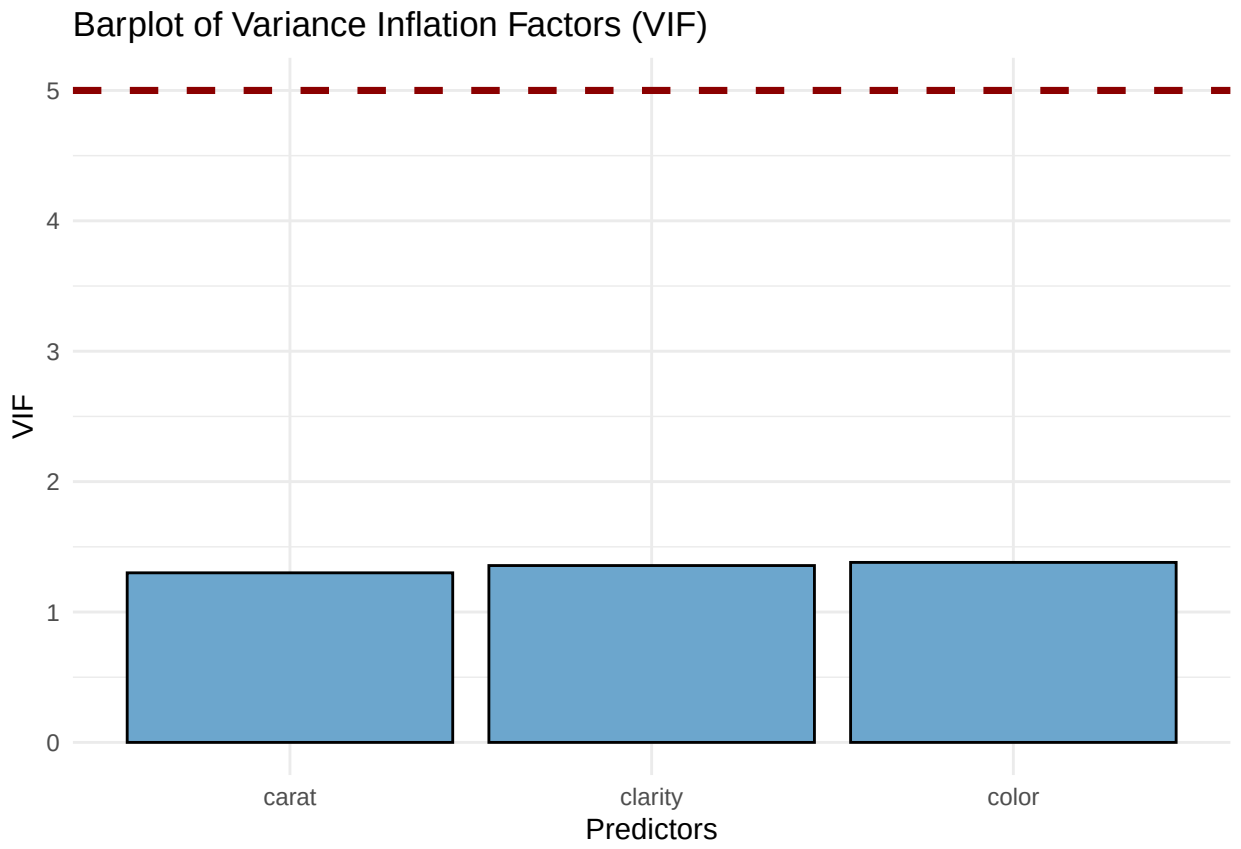
	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-3356.8464	171.6087	-19.5610508	0.0000000
carat	8361.6554	145.6690	57.4017609	0.0000000
color.L	-1659.4309	205.7116	-8.0667840	0.0000000
color.Q	-513.2369	189.0992	-2.7141150	0.0069441
color.C	-314.2829	179.9466	-1.7465339	0.0815157
color^4	-159.4218	173.2442	-0.9202142	0.3580371
color^5	-300.4593	163.3143	-1.8397608	0.0665730
color^6	222.8576	154.4609	1.4428093	0.1498873
clarity.L	4925.7528	462.1877	10.6574726	0.0000000
clarity.Q	-3003.7790	438.2194	-6.8545087	0.0000000
clarity.C	1862.8675	370.3009	5.0306864	0.0000008
clarity^4	-1039.5778	277.7258	-3.7431806	0.0002094
clarity^5	469.6846	215.8040	2.1764410	0.0301295
clarity^6	-239.1974	181.7802	-1.3158604	0.1890039
clarity^7	267.6589	154.2983	1.7346850	0.0835969

The formula for this resulting model is `price ~ carat + color + clarity`. It may be important to note that AIC prioritizes prediction accuracy, while BIC seeks simpler models for the sake of finding the "true" model. Here, we have found a model the adequately balances the two, but if prediction accuracy was an extreme priority, is it likely that this might not be the model that best fits our needs.

Variance Inflation Factor (VIF) is a measure of how much the confidence interval of a regression coefficient will grow due to the presence of multicolinearity. In general, if the VIF of each dependent variable is less than 5, we can conclude that multicolinearity is not present in our data. Here, we will find the VIFs of both dependent variables in the model found above.

```
# Finding VIF of each variable in the above model
vif_vals <- vif(min_model)[,1]

# Creating a data frame for usage with ggplot2
```

```r
vif_df <- data.frame(
  vars = names(vif(min_model)[,1]),
  vif = vif_vals
)

# Creating bar chart of each VIF in relation to a threshold of 5
ggplot(vif_df, aes(x = vars, y = vif)) +
  geom_bar(stat = "identity", fill = "skyblue3", color="black") +
  geom_hline(yintercept = 5, linetype = "dashed", color="red4", linewidth = 1.25) +
  xlab("Predictors") + ylab("VIF") + ggtitle("Barplot of Variance Inflation Factors (VIF)") +
  theme_minimal()
```



Since we observe no VIF values that are greater than 5, there is no cause for concern in terms of multicollinearity among the remaining predictors.

Model Diagnostics

Now that we have found the model that AIC and BIC, we must perform diagnostic tests to further improve our model. This includes validating the assumptions of linear regression, identifying outliers and handling multicolinearity.

Recall that the assumptions of linear regression are:

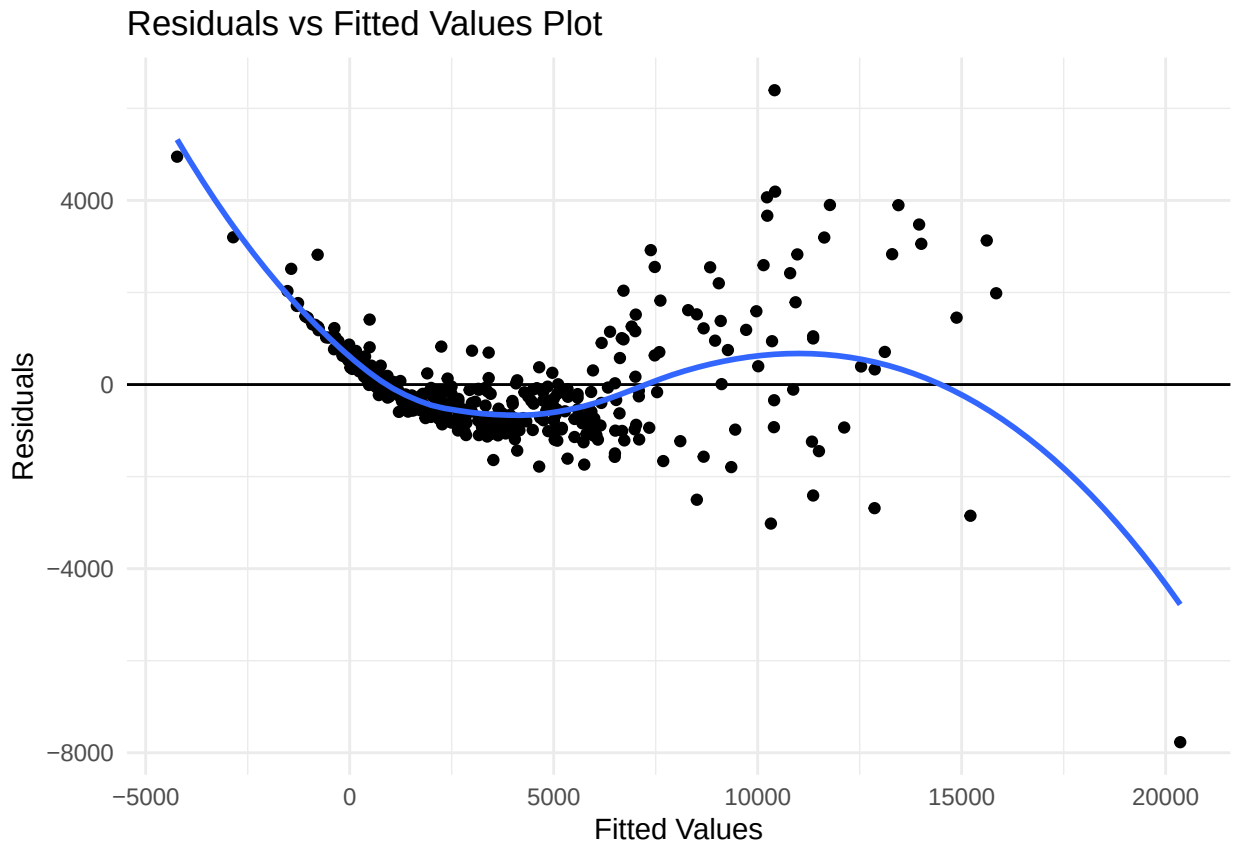
- Equality of variances (Homoscedasticity)

- Normality of residuals
- Independence of errors

Homoscedasticity

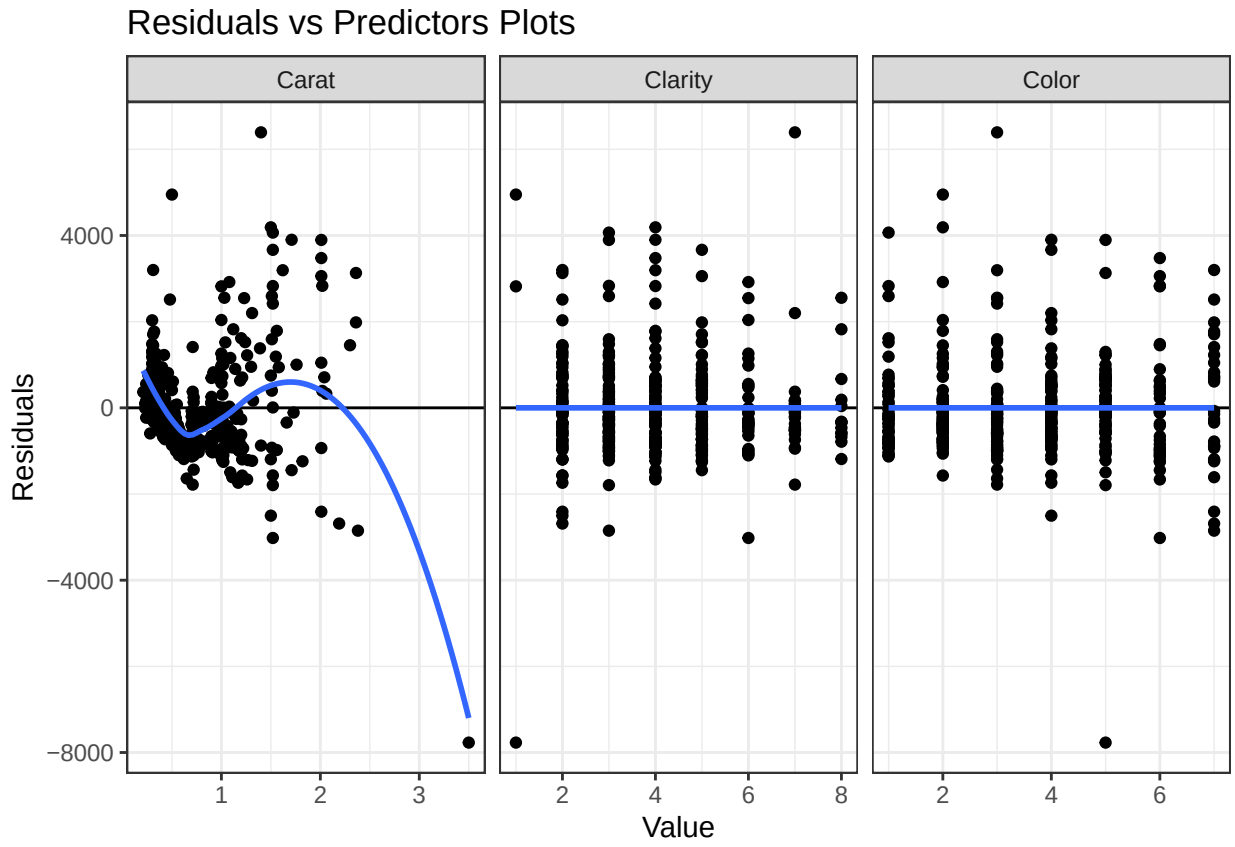
First, we will check if our model meets the homoscedasticity assumption. We will do this by using a Residual vs Fitted Values Plot. Ideally, the points for plot should be randomly scattered across zero. Let us conduct the following tests below:

```
# Augmenting data based on above model to include fitted values and residuals
augment_df <- augment(min_model, diamonds_df)
augment_df$clarity <- as.numeric(augment_df$clarity)
augment_df$color <- as.numeric(as.factor(augment_df$color))

# Graphing Residuals vs Fitted Values Plot
ggplot(augment_df, aes(x = .fitted, y = .resid)) +
  geom_point() + geom_hline(aes(yintercept = 0)) +
  geom_smooth(method = "loess", formula = "y~x", se = FALSE, span = 0.7) +
  xlab("Fitted Values") + ylab("Residuals") + ggtitle("Residuals vs Fitted Values Plot") +
  theme_minimal()
```



By examining the Residuals vs Fitted Values Plot above, we observe that the points are not randomly scattered around zero and seem to curve, implying that our data does not meet the assumption of homoscedasticity. In order to further examine this issue, we can utilize Residuals vs Predictors Plots to see if any specific predictor is responsible for this issue.

```r
# Lengthening augmented data for facet plot
augment_long <- pivot_longer(augment_df, cols = c(carat, clarity, color))

# Graphing Residuals vs Predictors Plots
ggplot(augment_long, aes(x = value, y = .resid)) +
  facet_wrap(~ name,  scales = "free_x",
             labeller = labeller(name = c(".fitted" = "Fitted Values",
                                          "carat" = "Carat",
                                          "clarity" = "Clarity",
                                          "color" = "Color"))) +
  geom_point() + geom_hline(aes(yintercept = 0)) +
  geom_smooth(method = "loess", formula = "y~x", se = FALSE, span = 0.7) +
  xlab("Value") + ylab("Residuals") + ggtitle("Residuals vs Predictors Plots") +
  theme_bw()
```

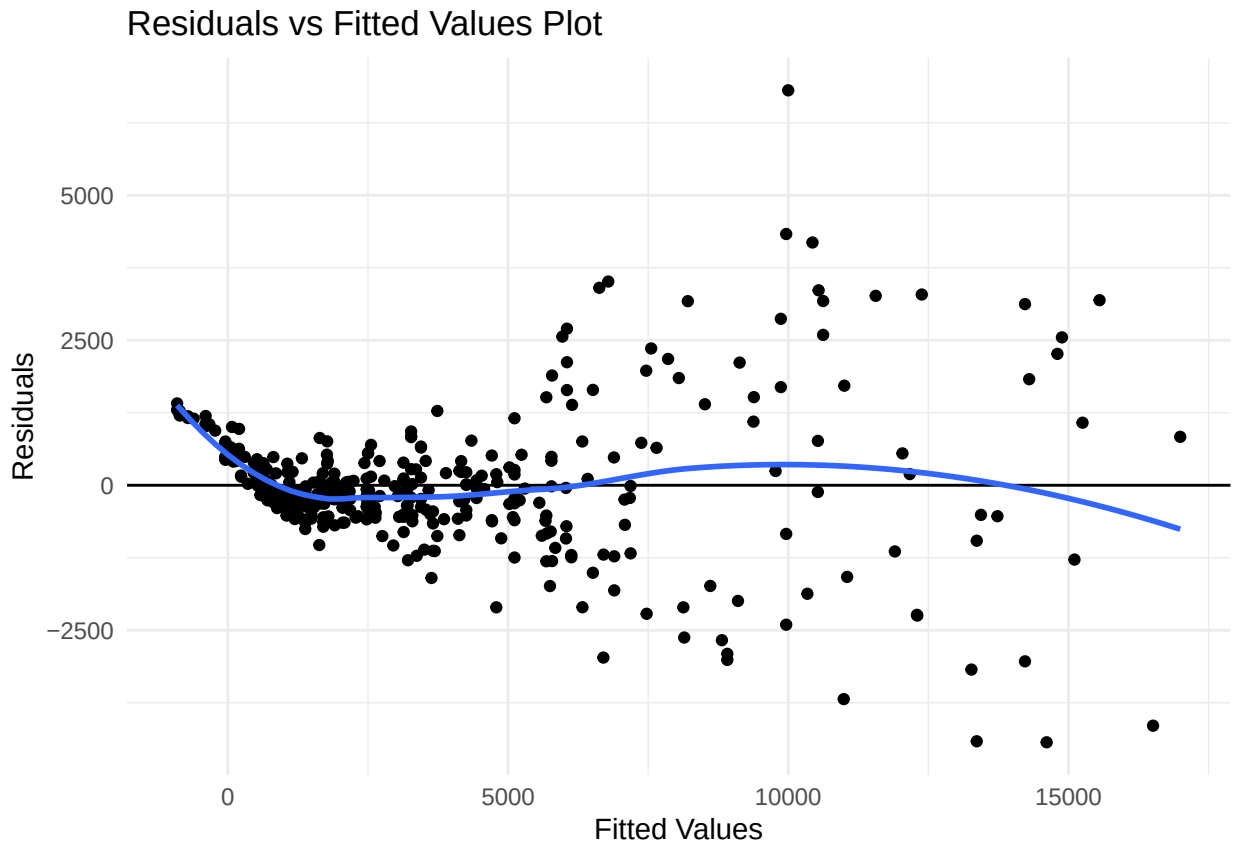


By analyzing the Fitted vs Predictors Plots, we observe that the issue lies with the `carat` variable. It is here that we see the same cubic pattern as in the Fitted vs Residuals Plot. This is validated by the fact that the Fitted vs Predictor Plots for `clarity` and `color` appears to have each factor's residuals randomly scattered along zero, with a linear trend line. Thus, we find that implies that the model fails to capture a cubic trend in the `carat` variable which impacts homoscedasticity and prediction.

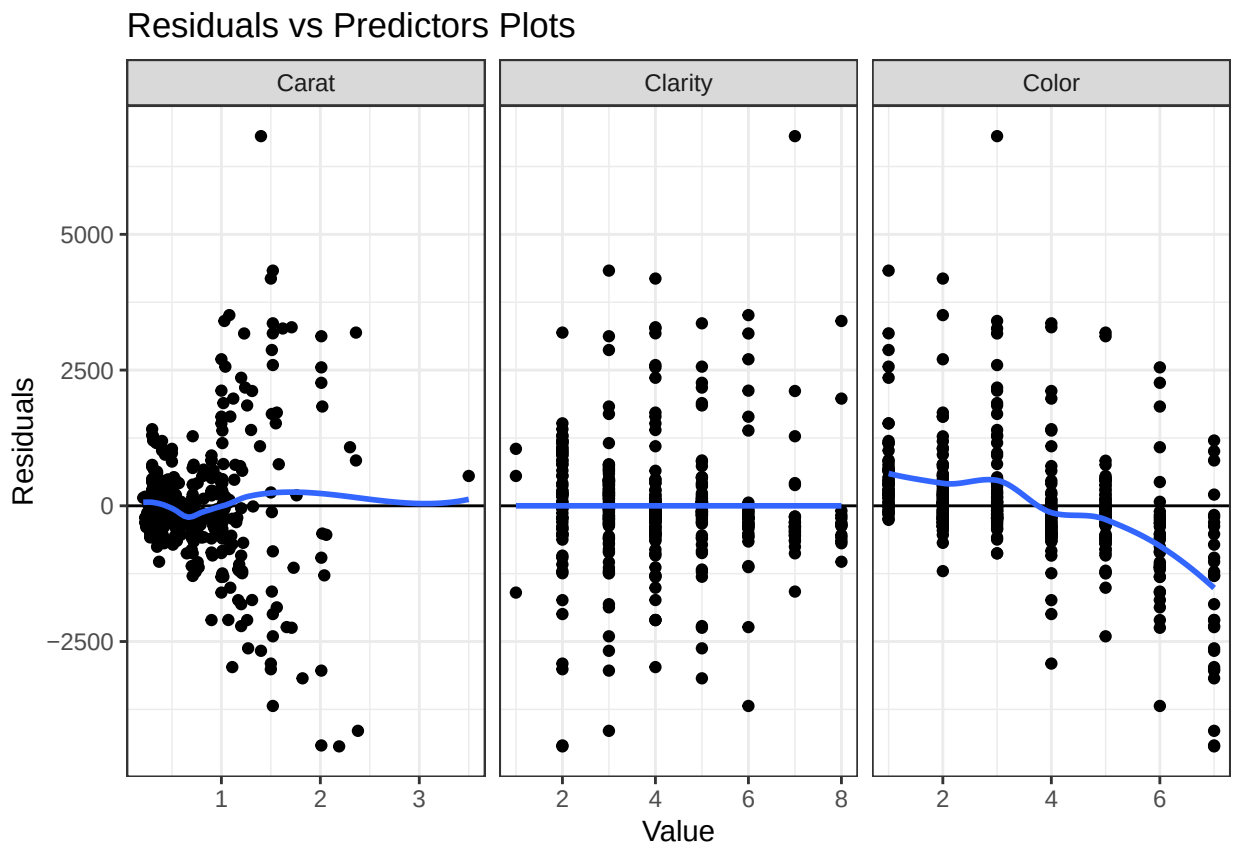
We can apply a cubic transformation to `carat` handle this issue. Let us plot the Residuals vs Fitted Values and Residuals vs Predictors Plots of the consquent model as such:

```r
# Applying cubic transformation to carat variable
trans_model <- lm(price ~ poly(carat, 3, raw=T) + clarity, data=diamonds_df)
```

```r
# Augmenting data based on above model to include fitted values and residuals
augment_df2 <- augment(trans_model, diamonds_df)
augment_df2$clarity <- as.numeric(augment_df2$clarity)
augment_df2$color <- as.numeric(as.factor(augment_df2$color))

# Graphing Residuals vs Fitted Values Plot
ggplot(augment_df2, aes(x = .fitted, y = .resid)) +
  geom_point() + geom_hline(aes(yintercept = 0)) +
  geom_smooth(method = "loess", formula = "y~x", se = FALSE, span = 0.7) +
  xlab("Fitted Values") + ylab("Residuals") + ggtitle("Residuals vs Fitted Values Plot") +
  theme_minimal()
```



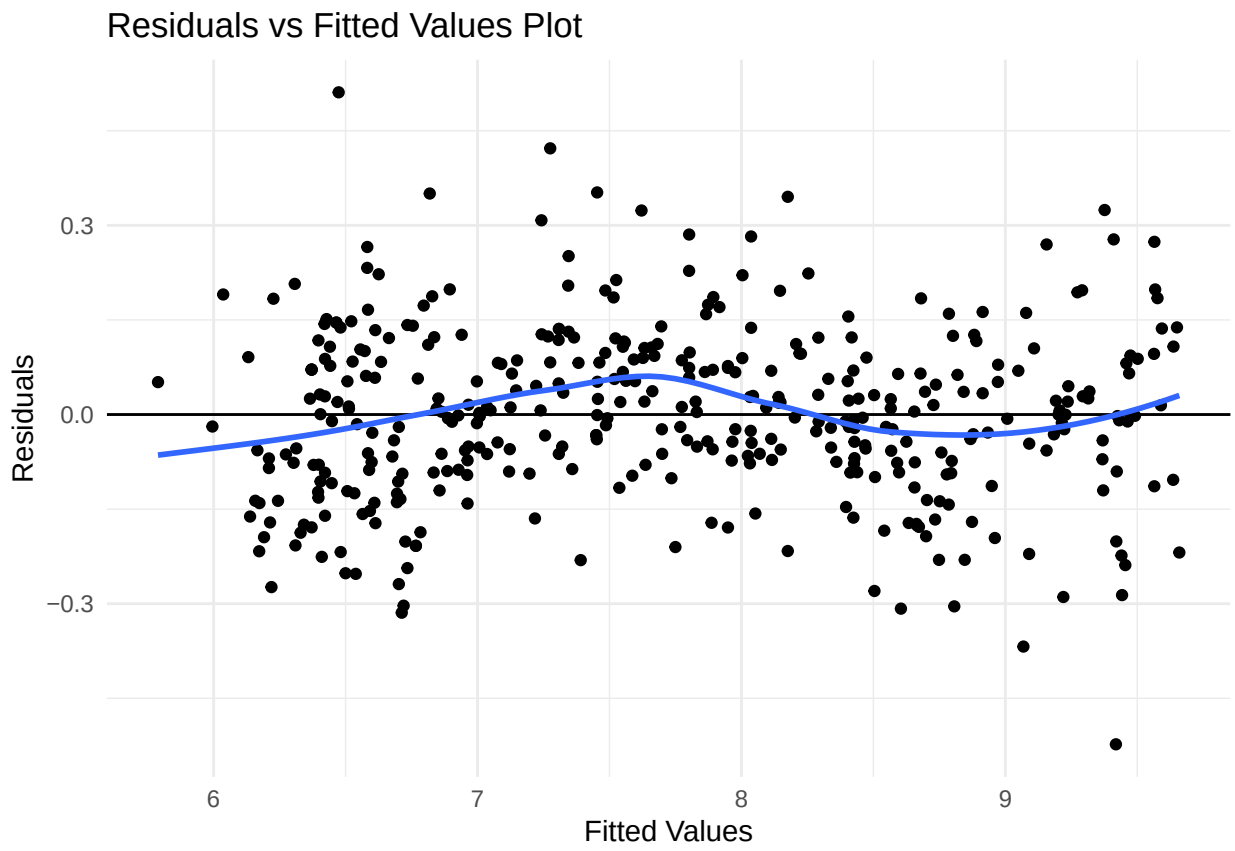

```r
# Lengthening augmented data for facet plot
augment_long2 <- pivot_longer(augment_df2, cols = c(carat, clarity, color))

# Graphing Residuals vs Predictors Plots
ggplot(augment_long2, aes(y = .resid, x = value)) +
  facet_wrap(~ name, scales = "free_x",
             labeller = labeller(name = c("carat" = "Carat",
                                          "clarity" = "Clarity",
                                          "color" = "Color"))) +
  geom_point() + geom_hline(aes(yintercept = 0)) +
  geom_smooth(method = "loess", formula = "y~x", se = FALSE, span = 0.7) +
  xlab("Value") + ylab("Residuals") + ggtitle("Residuals vs Predictors Plots") +
  theme_bw()
```



Upon examination of the Fitted vs Residuals Plot and Fitted vs Predictors Plots for the transformed model, we find that the cubic pattern in `carat` is now appropriately captured. However, there is now a funneling shape in the Fitted vs Residuals Plot which means that the issue of homoscedasticity still has not been fixed. Thus, another transformation is needed to handle this issue.

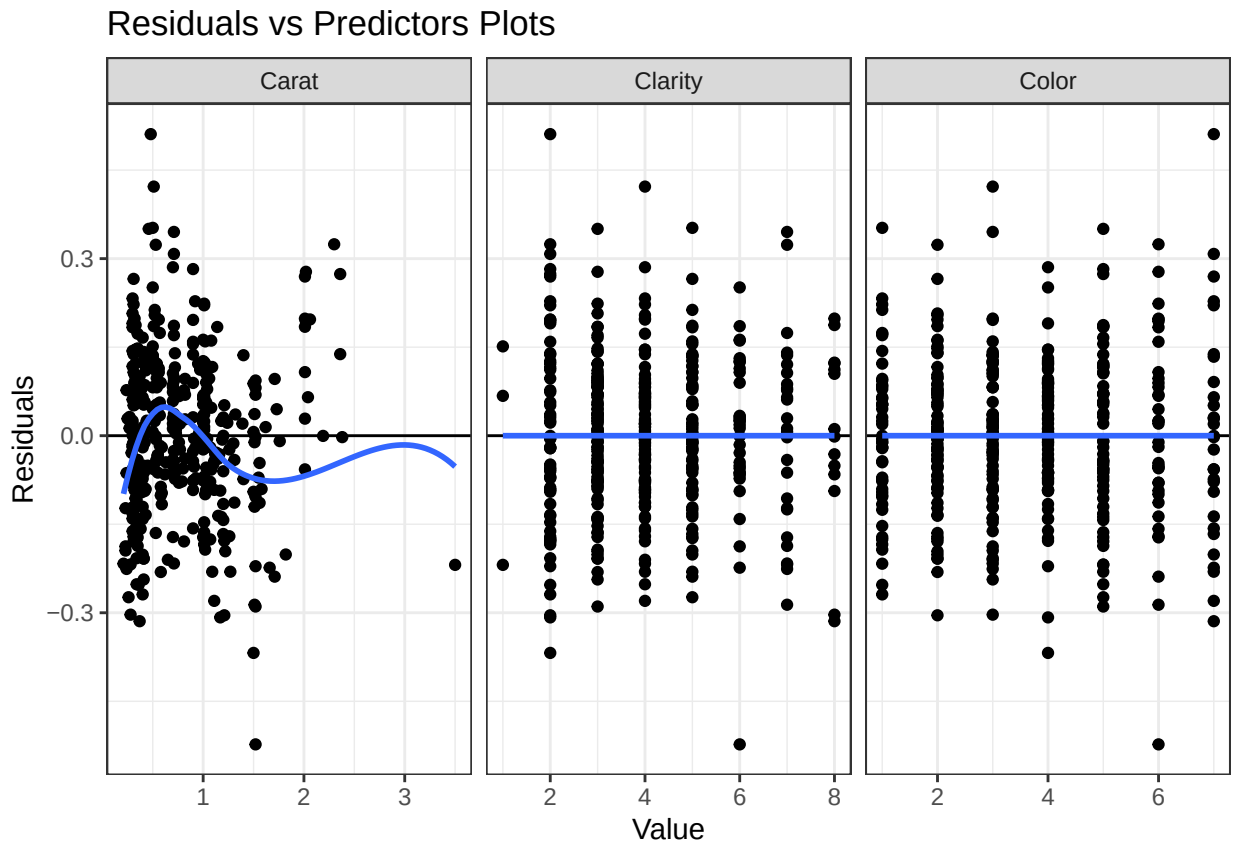
One potential solution is to apply a logarithmic transformation to our dependent variable, `price`. Graphing the Fitted vs Residuals and Fitted vs Predictors Plots of this model shows the following:

```
# Applying logarithmic transformation to the response variable
trans_model2 <- lm(log(price) ~ poly(carat, 3, raw=T) + clarity + color, data=diamonds_df)
augment_df3 <- augment(trans_model2, diamonds_df)
augment_df3$clarity <- as.numeric(augment_df3$clarity)
augment_df3$color <- as.numeric(as.factor(augment_df3$color))

# Graphing Residuals vs Fitted Values Plot
ggplot(augment_df3, aes(x = .fitted, y = .resid)) +
  geom_point() + geom_hline(aes(yintercept = 0)) +
  geom_smooth(method = "loess", formula = "y~x", se = FALSE, span = 0.7) +
  xlab("Fitted Values") + ylab("Residuals") +  ggtitle("Residuals vs Fitted Values Plot") +
  theme_minimal()
```



After applying this logarithmic transformation, we find that the resulting Residuals vs Fitted Values Plot shows points to be randomly scattered along zero with no systematic trend. This shows that this final model meets the homoscedasticity assumption of linear regression. To validate this notion and to see if the cubic trend found in the original model has been appropriately captured, we can again inspect the Residuals vs Predictors Plots.

```
# Lengthening augmented data for facet plot
augment_long3 <- pivot_longer(augment_df3, cols = c(carat, clarity, color))

# Graphing Residuals vs Predictors Plots
ggplot(augment_long3, aes(y = .resid, x = value)) +
  facet_wrap(~ name, scales = "free_x",
             labeller = labeller(name = c("carat" = "Carat",
                                          "clarity" = "Clarity",
                                          "color" = "Color"))) +
  geom_point() + geom_hline(aes(yintercept = 0)) +
  geom_smooth(method = "loess", formula = "y~x", se = FALSE, span = 0.7) +
  xlab("Value") + ylab("Residuals") + ggtitle("Residuals vs Predictors Plots") +
  theme_bw()
```

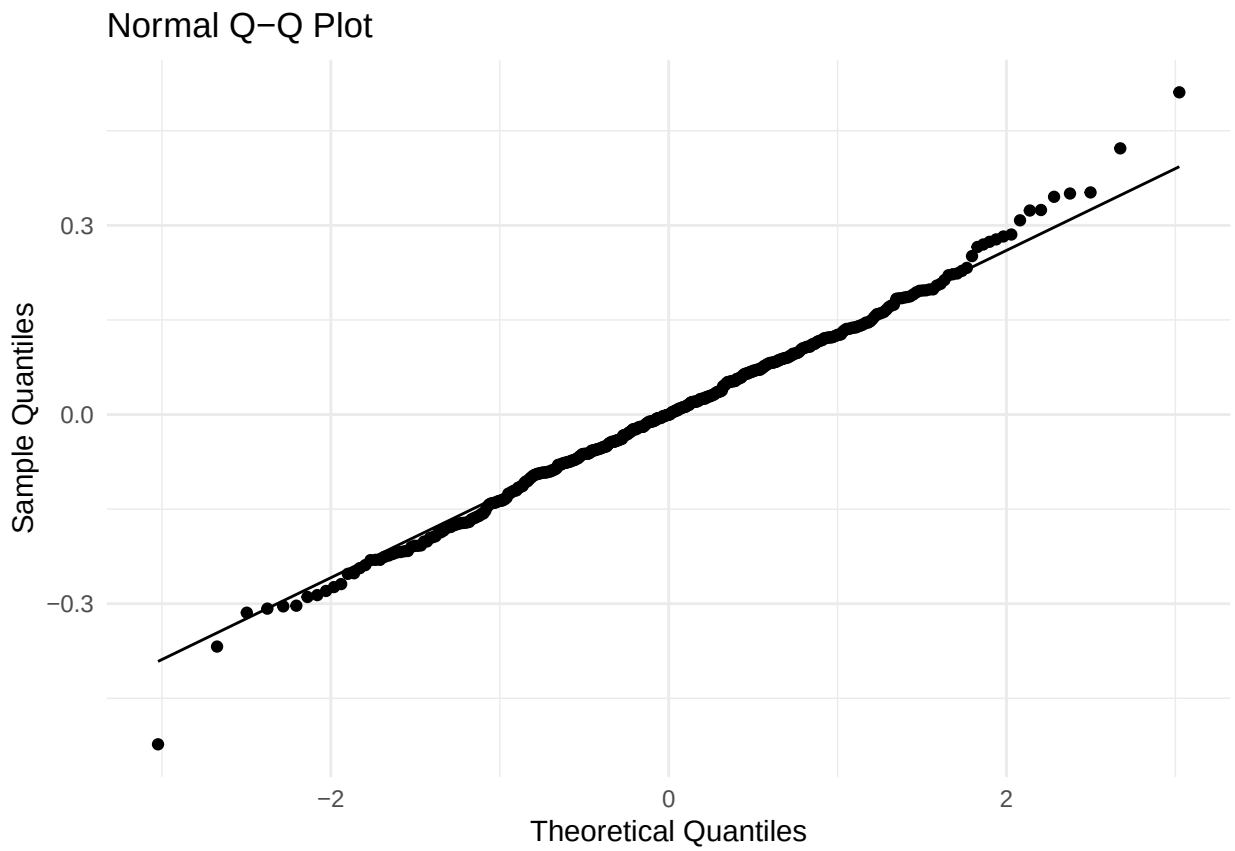


Finally, we find that while there is still a slight trend in `carat`, it is no longer as extreme as we originally saw and seems to stay around `y=0`. In addition to the fact that `clarity` and `color` still have each factor's residuals

Normality

In order to ensure that our model meets the normality of residuals assumption, we utilize a Normal Q-Q Plot. If our model does in fact meet this condition, then the points on the plot will generally follow the sloped Q-Q Line.

```
# Graphing Normal Q-Q Plot
ggplot(augment_df3, aes(sample = .resid)) +
  geom_qq() + geom_qq_line() +
  xlab("Theoretical Quantiles") + ylab("Sample Quantiles") +
  ggtitle("Normal Q-Q Plot") +
  theme_minimal()
```



By examining the Normal Q-Q Plot above, we observe that the points tend to follow the Q-Q Line. This implies that our transformed model adequately meets the Normality assumption. We may note that as the theoretical quantiles becomes less than -2, the points start to deviate from the Q-Q Line. However, this does not impact our conclusion as the plot resembles a Normal distribution, meaning points here are interpreted as more than 2 standard deviations from the mean. Thus, we may put less weight on those points in our decision and still conclude that our model satisfies the Normality assumption.

Independence of Errors

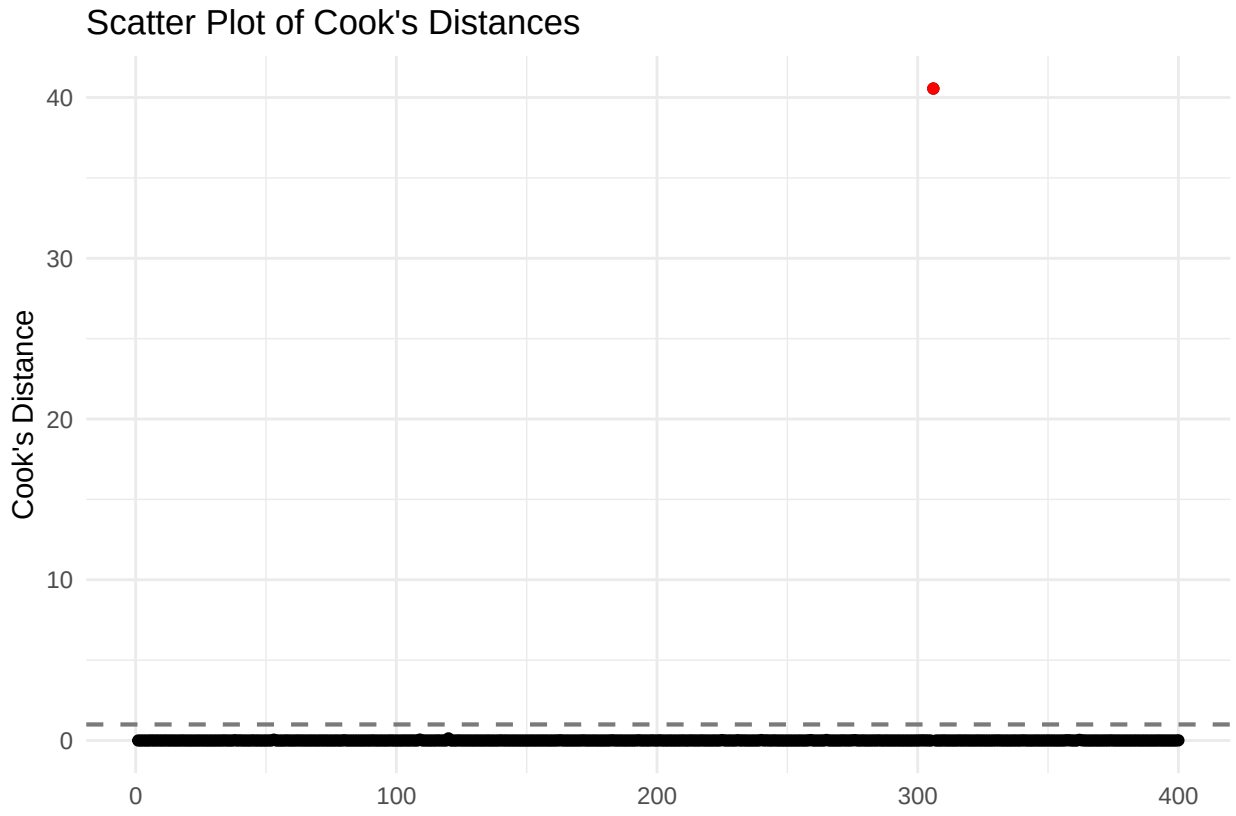
The independence of errors is a critical assumption in regression analysis, but it is not typically something we test for directly. Instead, it is ensured (or violated) based on how the data is collected. Here, we will assume that the assumption is met and will proceed.

Unusual points

After perfecting our model, we must identify observations in our random sample that may skew the performance of our model or may be a result of a clerical error. First, we will identify influential points, which have disproportionate effects on the regression line. In order to do this we will plot Cook's distance, and observations who have a Cook's distance greater than 1 will be considered an influential point.

```
# Mutating a new column with row indices
indexed_df <- mutate(augment_df3, index = row_number())

# Identifying influential points
```

```r
influence_points <- filter(indexed_df, `.cooksd` > 1)

# Graphing Cook's distance and highlighting influential points
ggplot(indexed_df, aes(x = index, y = `.cooksd`)) +
  geom_point() +
  geom_hline(aes(yintercept = 1), linetype="dashed", color="gray48", linewidth=0.75) +
  geom_point(data = influence_points, color = 'red') +
  xlab("") + ylab("Cook's Distance") +
  ggtitle("Scatter Plot of Cook's Distances") +
  theme_minimal()
```




```r
# Creating a kable of influential points
influence_df <- data.frame(
  "Index" = influence_points["index"],
  "Cook's Distance" =  round(influence_points[".cooksd"], 2)
)

colnames(influence_df) <- c("Index", "Cook's Distance")
kable(influence_df, caption="Table of Influential Points")
```

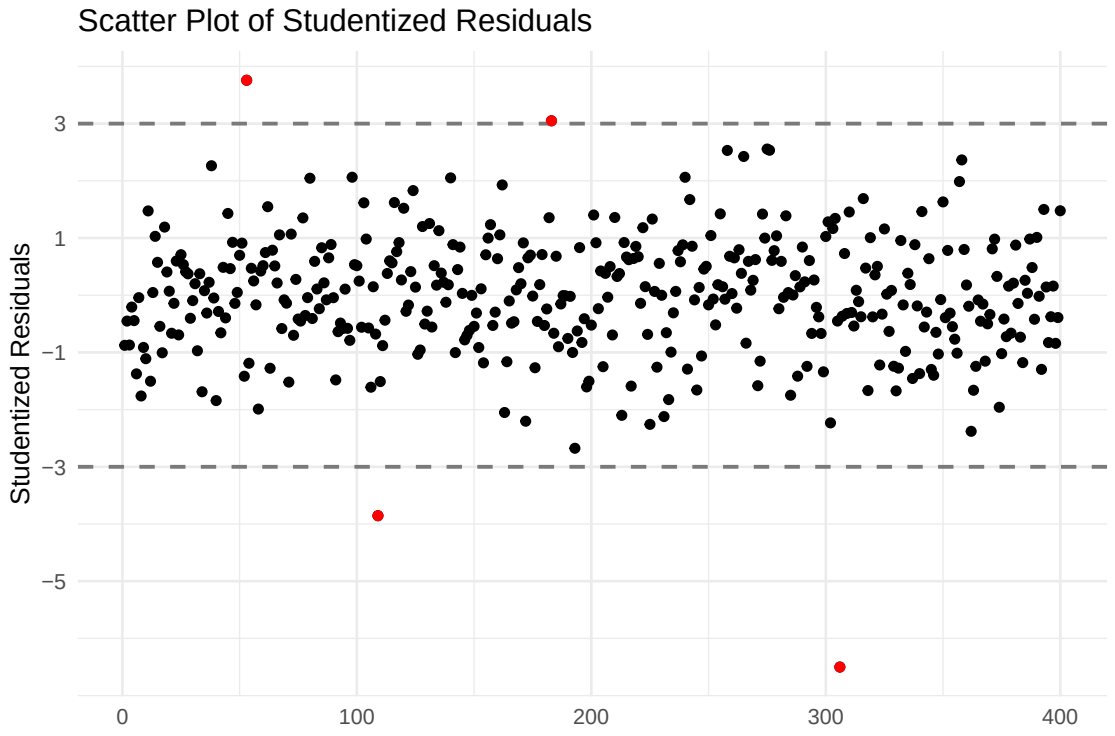
Table 7: Table of Influential Points

Index	Cook's Distance
306	40.56

The scatter plot and table above show that each observation has a Cook's distance below 1 with the exception of one. This makes the point is influential with a Cook's distance of 40.56, which is extremely high especially in comparison to the rest of the observations.

Before making a decision on how to handle this influential point, we can first point out outliers in our sample to see if this point falls in that category which would make this an easy decision to remove the point. To do this, we will calculate the Studentized residual of each observation, and classify it as an outlier if its value is less than -3 or greater than 3.

```
# Identifying outliers
outliers <- filter(indexed_df, abs(`.std.resid`) > 3)

# Graphing studentized residuals and highlighting outliers
ggplot(indexed_df, aes(x = index, y = `.std.resid`)) +
  geom_point() +
  geom_hline(aes(yintercept = 3), linetype="dashed", color="gray48", linewidth=0.75) +
  geom_hline(aes(yintercept = -3), linetype="dashed", color="gray48", linewidth=0.75) +
  geom_point(data = outliers, color = 'red') +
  xlab("") + ylab("Studentized Residuals") +
  scale_y_continuous(breaks=seq(-5, 5, 2)) +
  ggtitle("Scatter Plot of Studentized Residuals") +
  theme_minimal()
```



```r
# Creating a kable of outliers
outliers_df <- data.frame(
  "Index" = outliers["index"],
  "Cook's Distance" =  round(outliers[".std.resid"], 2)
)

colnames(outliers_df) <- c("Index", "Studentized Residual")
kable(outliers_df, caption="Table of Outliers")
```

Table 8: Table of Outliers

Index	Studentized Residual
53	3.76
109	-3.86
183	3.05
306	-6.50

By finding the Studentized residuals of each observation, we find that there are 4 outliers in our random sample. Additionally, the observation at index 306, which we previously classified as an influential point, also appears as an outlier. Furthermore, it's Studentized residual is more extreme than any other in the dataset. This gives us confidence in choosing to remove it from our random sample. However, for the other three outliers, we can examine them further to see if they are the result of some clerical error.

Since it appears that there are no clerical errors in these outliers, we can assume that they are still accurate samples from the population and leave them to be used in our final model.

To finish, let us remove our singular influential outlier and proceed to re-fit our model.

```r
# Removing the influential outlier
final_df <- filter(diamonds_df, price != 12587)
```

Implementing and Evaluating Final Model

Let us take our new sample which does not include any influential outliers, and run our transformed model on it.

```r
final_model <- lm(log(price) ~ poly(carat, 3, raw=T) + clarity + color, data=final_df)
kable(summary(final_model)[[4]], caption="Coefficient Summary of Final Model")
```

Table 9: Coefficient Summary of Final Model

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	4.5837660	0.0480337	95.4280835	0.0000000
poly(carat, 3, raw = T)1	6.4917181	0.1667084	38.9405671	0.0000000
poly(carat, 3, raw = T)2	-3.1199628	0.1639703	-19.0276048	0.0000000
poly(carat, 3, raw = T)3	0.5796515	0.0461195	12.5684760	0.0000000
clarity.L	0.9102812	0.0569620	15.9804897	0.0000000
clarity.Q	-0.2434491	0.0553658	-4.3970979	0.0000142
clarity.C	0.1367210	0.0464022	2.9464355	0.0034117

	Estimate	Std. Error	t value	Pr(>|t|)
clarity^4	-0.0567493	0.0336691	-1.6855032	0.0927081
clarity^5	0.0515187	0.0246330	2.0914499	0.0371478
clarity^6	-0.0018344	0.0198671	-0.0923352	0.9264801
clarity^7	0.0514864	0.0166633	3.0898141	0.0021495
color.L	-0.4525244	0.0227789	-19.8659376	0.0000000
color.Q	-0.0708077	0.0206597	-3.4273369	0.0006758
color.C	-0.0584423	0.0194508	-3.0046193	0.0028345
color^4	0.0218404	0.0187395	1.1654722	0.2445551
color^5	0.0142511	0.0177370	0.8034673	0.4222043
color^6	0.0550154	0.0167253	3.2893521	0.0010974

This final model has an adjusted R-squared of $R^2_{adj} = 0.98$ which is a slight improvement from when the model was run with the influential outlier present. This means that this model is able to explain 98% of the variation in diamond price.

It uses transformations to reveal that carat, clarity, and color are statistically significant predictors of diamond prices.

Conclusion

In this project, we conducted a series of model evaluations and visual diagnostics to understand the factors influencing diamond prices. We first randomly sampled 400 observations from the dataset for visualization and model building. Afterwards, we started by using histograms, scatter plots and violin plots to understand variable relationships and distributions before building any models. By visualizing a correlation heatmap, we were able to get a general idea of what variables would cause multicolinearity, which we formalized by later examining Variance Inflation Factor (VIF). Using different techniques to conduct feature selection, it was ultimately found that using backward selection to minimize BIC was the best fitting model. Next, we verified the the assumptions of linear regression where variable transformation was also conducted to properly capture non-linear trends. Finally, after removing influential outliers, our final model achieved an adjusted R-squared of 0.98, which is a 0.08 increase from the full model. The regression analysis ultimately found that carat, clarity and color had statistically significant effects on diamonds prices in 2022.