Diamonds_EDA_Stat

Ashkaan Moinzadeh, Alejandro Reskala Ruiz, Shrey Singhal

2022-08-21

initialiation

import packages

import diamonds data

```
diamonds <- read_csv("/Users/moinzade/Desktop/UCB_MIDS/W203/lab_2/Diamonds Prices2022.csv")
diamonds <- as_tibble(diamonds)
```

preliminary assessment

examine header

```
head(diamonds)
```

```
## # A tibble: 6 x 11
##    ...1 carat cut       color clarity depth table price     x     y     z
##   <dbl> <dbl> <chr>     <chr> <chr>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
## 2     2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
## 3     3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31
## 4     4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
## 5     5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75
## 6     6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
```

col names and col number

```
colnames(diamonds)
```

```
##  [1] "...1"    "carat"   "cut"     "color"   "clarity" "depth"   "table"
##  [8] "price"   "x"       "y"       "z"
```

```
ncol(diamonds)
```

```
## [1] 11
```

Here, "...1" is the "index" of the dataset.

row number

```
nrow(diamonds)
```

```
## [1] 53943
```

na count

```
sum(is.na(diamonds))
```

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

There are no NA values in the dataset. ## datatypes

```
str(diamonds)
```

```
## tibble [53,943 x 11] (S3: tbl_df/tbl/data.frame)
##  $ ...1   : num [1:53943] 1 2 3 4 5 6 7 8 9 10 ...
##  $ carat  : num [1:53943] 0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
##  $ cut    : chr [1:53943] "Ideal" "Premium" "Good" "Premium" ...
##  $ color  : chr [1:53943] "E" "E" "E" "I" ...
##  $ clarity: chr [1:53943] "SI2" "SI1" "VS1" "VS2" ...
##  $ depth  : num [1:53943] 61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
##  $ table  : num [1:53943] 55 61 65 58 58 57 57 55 61 61 ...
##  $ price  : num [1:53943] 326 326 327 334 335 336 336 337 337 338 ...
##  $ x      : num [1:53943] 3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
##  $ y      : num [1:53943] 3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
##  $ z      : num [1:53943] 2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
```

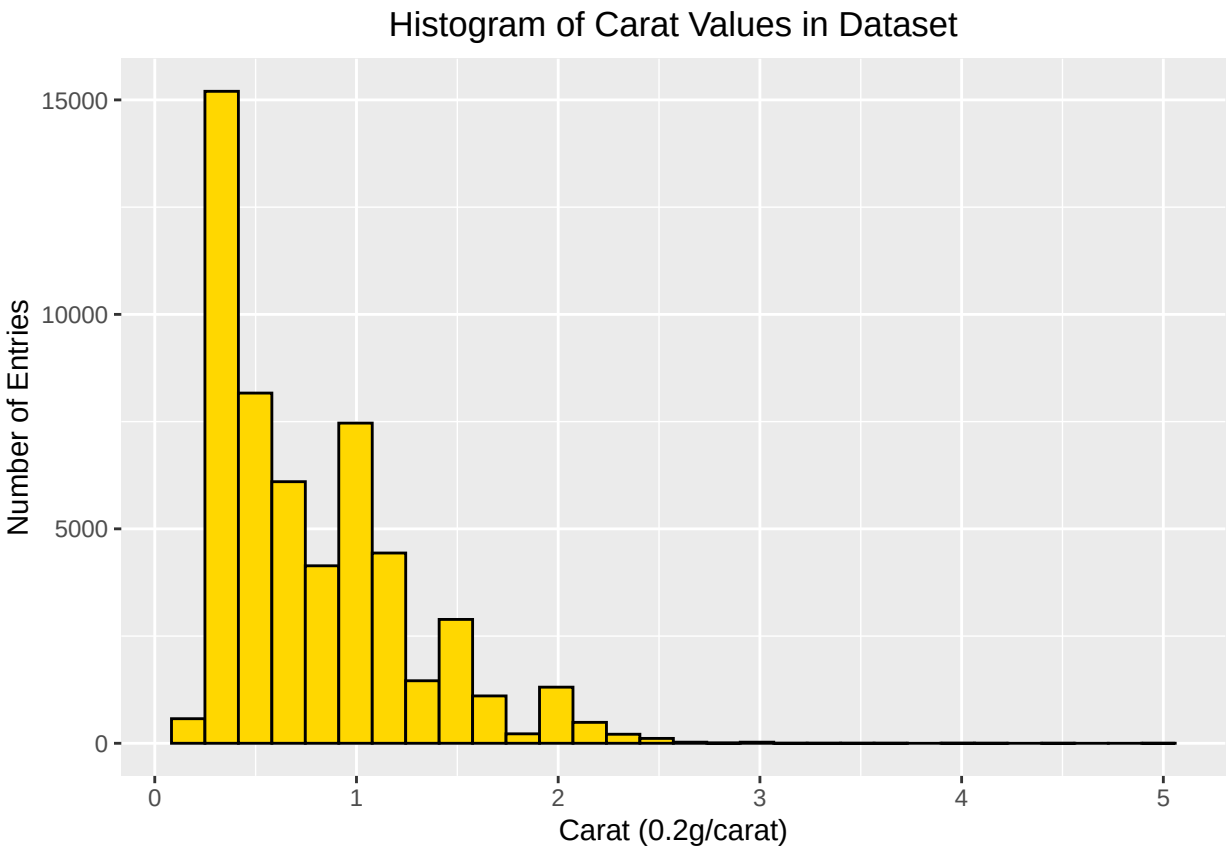
Numeric Data: ...1, carat, depth, table, price, x, y, z Character Data: cut, color, clarity

column-wise assessments:

carat:

Histogram of Carat:

```
carat_hist <- diamonds %>%
  ggplot(mapping = aes(x = carat)) +
  geom_histogram(color = "black", fill = "gold", stat = "bin") +
  xlab("Carat (0.2g/carat)") +
  ylab("Number of Entries") +
  ggtitle("Histogram of Carat Values in Dataset") +
  theme(plot.title = element_text(hjust = 0.5))

carat_hist
```



- Notice how the distribution values is positively skewed. This means that our data will be heavily trained for carat values between (0,1), but will be poorly trained for higher carat values. The skewness of the data is 1.12. - Likewise, there are many values above Carat = 2.54 that are candidates for outliers. It would be useful to remove them from the dataset, since the total number of them is small, but they are shifting the dataset by a lot.

Descriptive Statistics of Carat:

```
mean(diamonds$carat)
```

```
## [1] 0.7979347
```

```
median(diamonds$carat)
```

```
## [1] 0.7
```

```
#####
sd(diamonds$carat)
```

```
## [1] 0.4739986
```

```
IQR(diamonds$carat)
```

```
## [1] 0.64
```

```
mad(diamonds$carat)
```

```
## [1] 0.474432
```

```
#####
min(diamonds$carat)
```

```
## [1] 0.2
```

```
max(diamonds$carat)
```

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

```
quantile(diamonds$carat)
```

```
##   0%  25%  50%  75% 100%
## 0.20 0.40 0.70 1.04 5.01
```

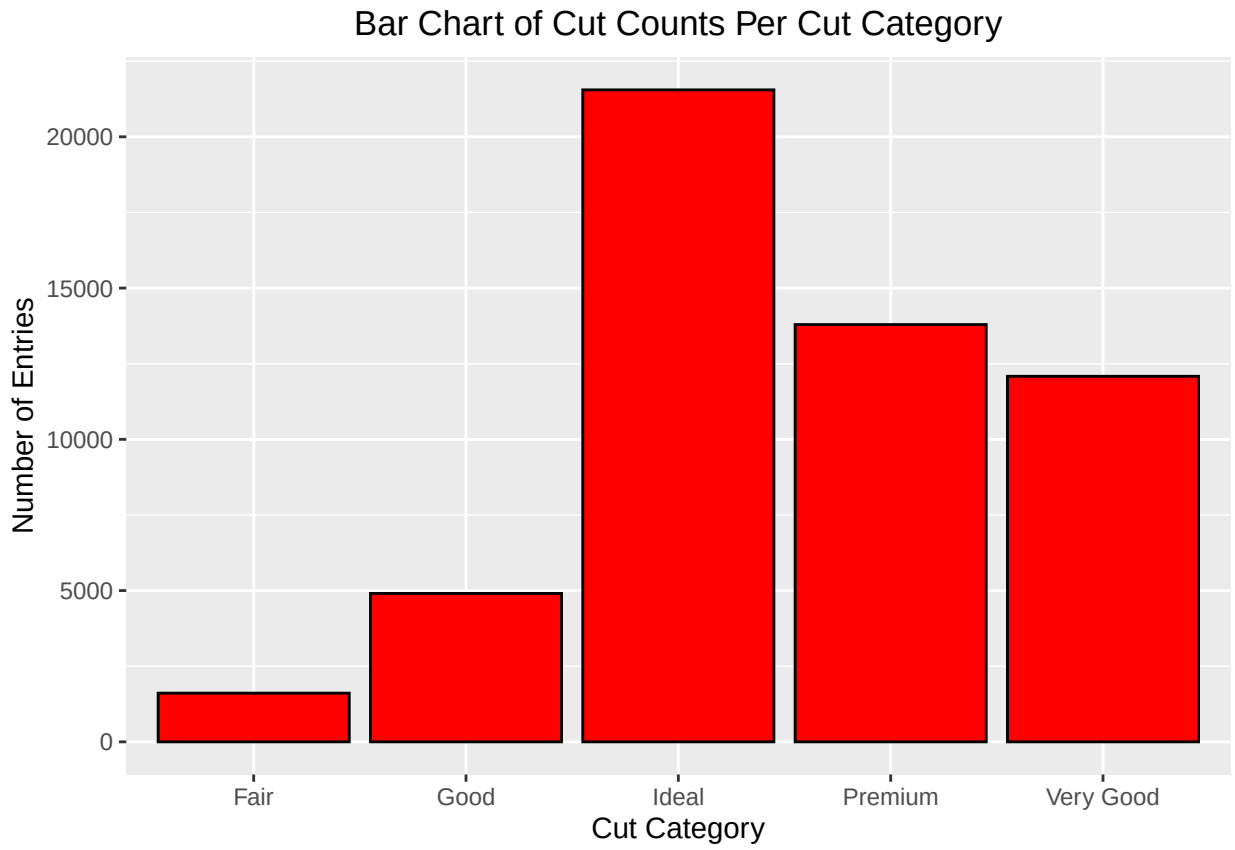
```
#####
skewness(diamonds$carat)
```

```
## [1] 1.116674
```

Cut:

Bar Chart of Cut:

```
cut_bar <- diamonds %>%
  ggplot(mapping = aes(x = cut)) +
  geom_bar(color = "black", fill = "red") +
  xlab("Cut Category") +
  ylab("Number of Entries") +
  ggtitle("Bar Chart of Cut Counts Per Cut Category") +
  theme(plot.title = element_text(hjust = 0.5))

cut_bar
```



- Notice how the majority of cuts are ideal, and decrease based on lower quality. It can be seen how Ideal makes up 40.0% of the dataset, Premium 25.5%, Very Good 22.4%, Good 9.1% and Fair 3.0%.

```
sum(diamonds$cut == "Ideal")/nrow(diamonds)
```

```
## [1] 0.3995143
```

```
sum(diamonds$cut == "Premium")/nrow(diamonds)
```

```
## [1] 0.2556958
```

```
sum(diamonds$cut == "Very Good")/nrow(diamonds)
```

```
## [1] 0.2239957
```

```
sum(diamonds$cut == "Good")/nrow(diamonds)
```

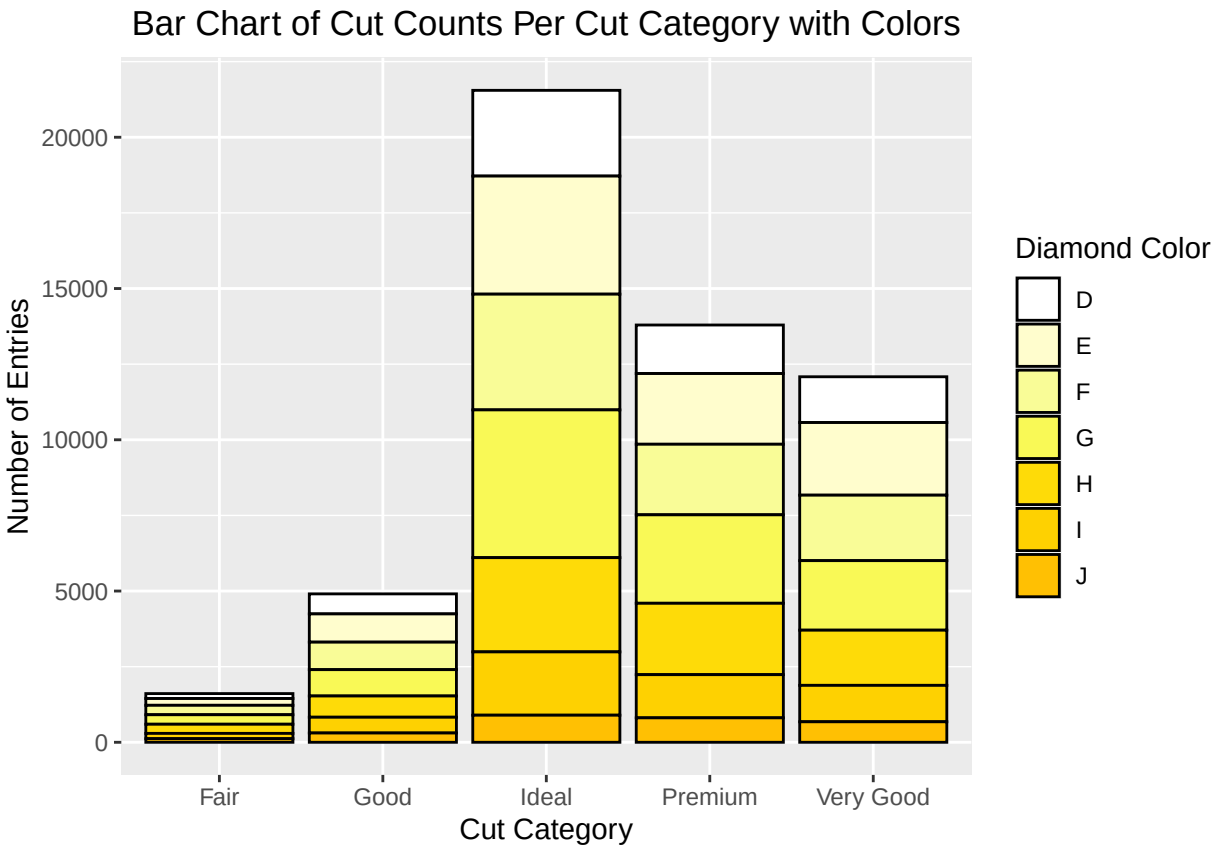
```
## [1] 0.09094785
```

```
sum(diamonds$cut == "Fair")/nrow(diamonds)
```

```
## [1] 0.02984632
```

Bar Chart of Cut with Colors:

```
cut_bar_color <- diamonds %>%
  ggplot(mapping = aes(x = cut, fill = color)) +
  geom_bar(color = "black") +
  xlab("Cut Category") +
  ylab("Number of Entries") +
  ggtitle("Bar Chart of Cut Counts Per Cut Category with Colors") +
  theme(plot.title = element_text(hjust = 0.5)) +
  guides(fill = guide_legend(title = "Diamond Color")) +
  scale_fill_manual(values = c("#FFFFFF",
                       "#FFFDCD",
                       "#F9FC97",
                       "#F9F956",
                       "#FEDB08",
                       "#FED101",
                       "#FFC003"))

cut_bar_color
```

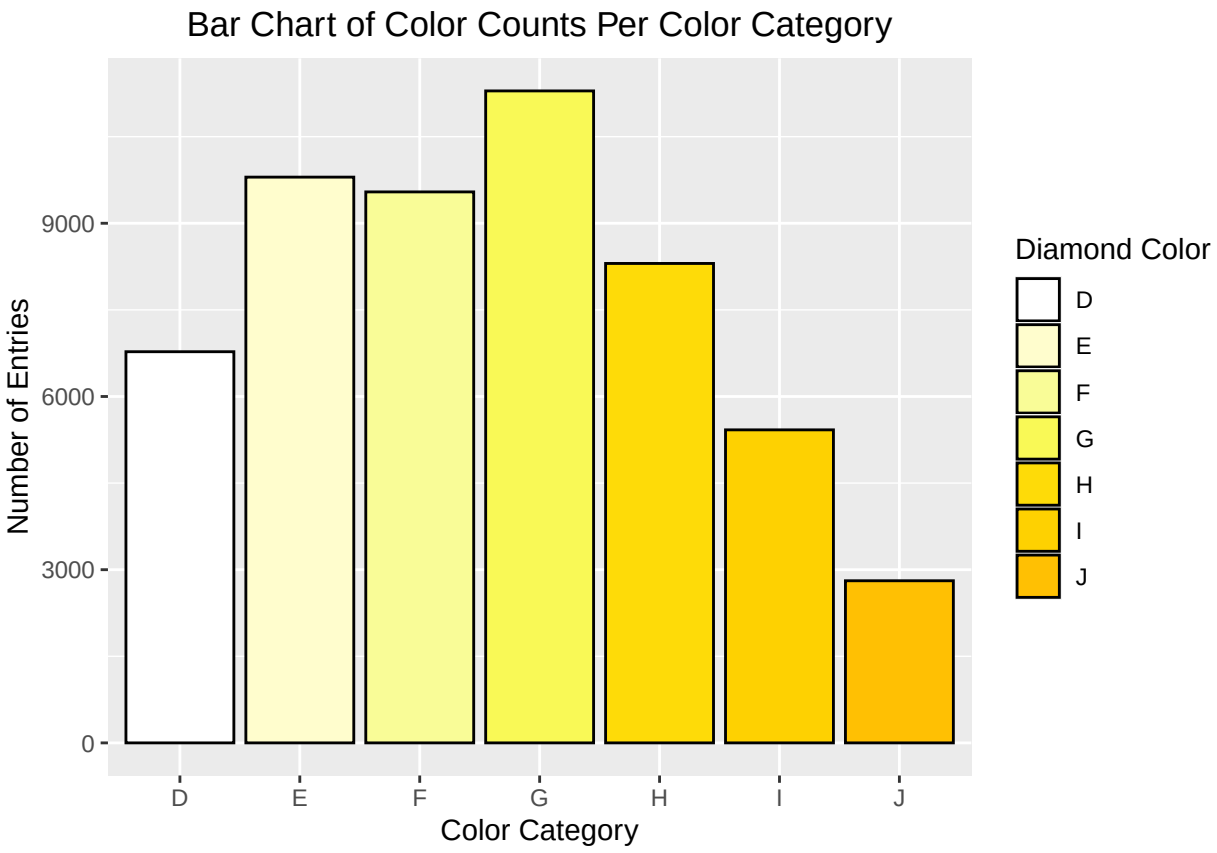


Distribution of Colors per Cut is fairly even distributed.

Color:

Bar Chart of Color:

```
color_bar <- diamonds %>%
  ggplot(mapping = aes(x = color, fill = color)) +
  geom_bar(color = "black") +
  xlab("Color Category") +
  ylab("Number of Entries") +
  ggtitle("Bar Chart of Color Counts Per Color Category") +
  theme(plot.title = element_text(hjust = 0.5)) +
  guides(fill = guide_legend(title = "Diamond Color")) +
  scale_fill_manual(values = c("#FFFFFF",
                        "#FFFDCD",
                        "#F9FC97",
                        "#F9F956",
                        "#FEDB08",
                        "#FED101",
                        "#FFC003"))

color_bar
```



- Note how the color scale in this dataset only goes from D-J (and excludes K-Z, which are murkier colors).
- Secondly, note how D makes up 12.6% of the data, E 18.2%, F 17.7%, G 20.9%, H 15.4%, I 10.1% and J 5.2%.

```r
sum(diamonds$color == "D")/nrow(diamonds)
```

```
## [1] 0.1255955
```

```r
sum(diamonds$color == "E")/nrow(diamonds)
```

```
## [1] 0.1816547
```

```r
sum(diamonds$color == "F")/nrow(diamonds)
```

```
## [1] 0.176909
```

```r
sum(diamonds$color == "G")/nrow(diamonds)
```

```
## [1] 0.2093321
```

```r
sum(diamonds$color == "H")/nrow(diamonds)
```

```
## [1] 0.1539403
```

```r
sum(diamonds$color == "I")/nrow(diamonds)
```

```
## [1] 0.1005135
```

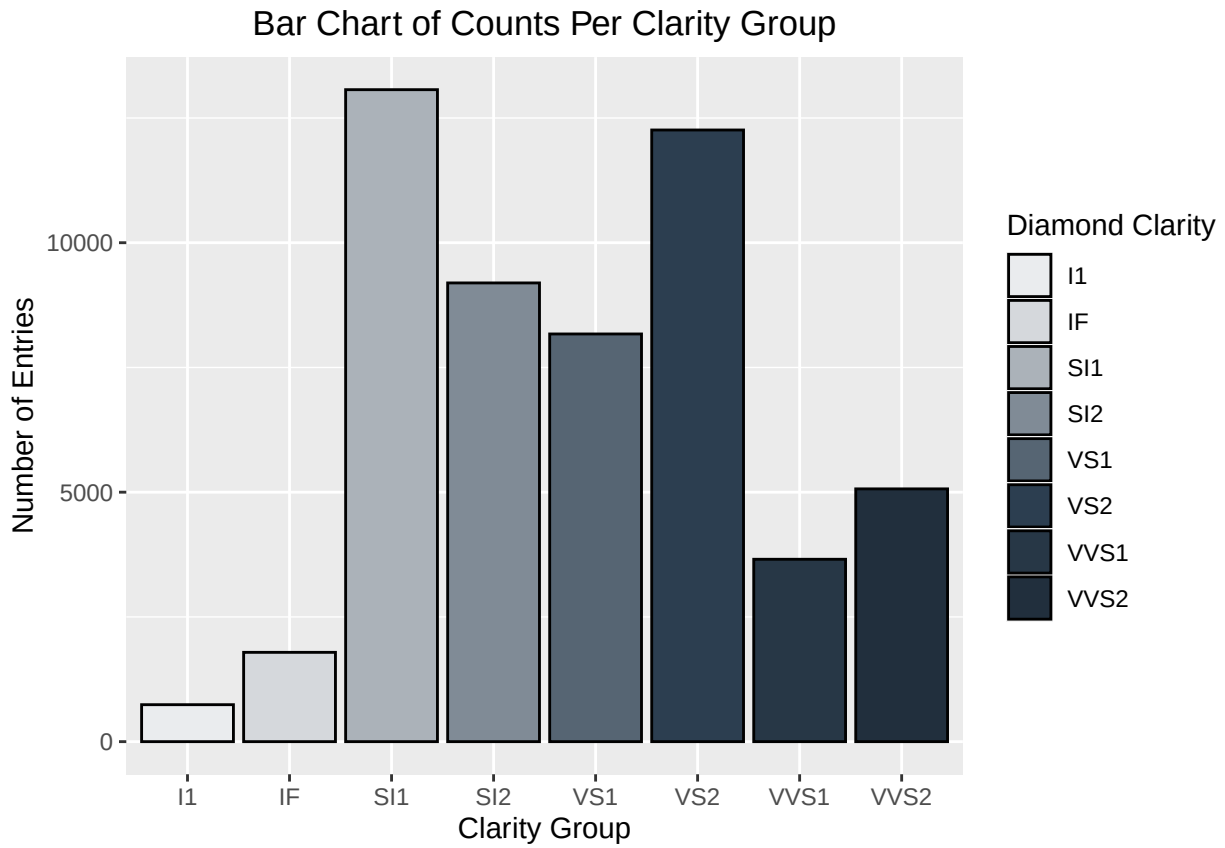
```r
sum(diamonds$color == "J")/nrow(diamonds)
```

```
## [1] 0.05205495
```

Clarity:

Barchart of Clarity WITHOUT Colors

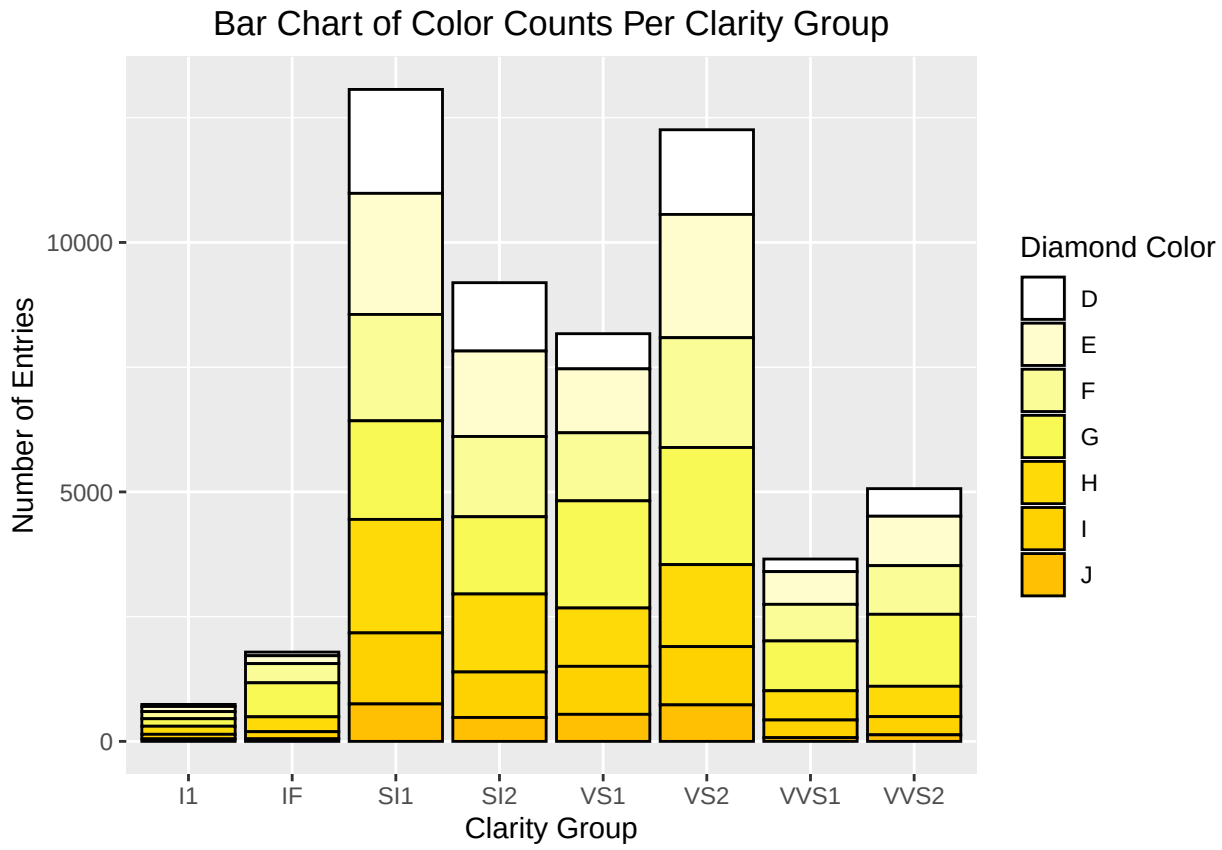
```
clarity_bar <- diamonds %>%
  ggplot(mapping = aes(x = clarity, fill = clarity)) +
  geom_bar(color = "black") +
  xlab("Clarity Group") +
  ylab("Number of Entries") +
  ggtitle("Bar Chart of Counts Per Clarity Group") +
  theme(plot.title = element_text(hjust = 0.5)) +
  guides(fill = guide_legend(title = "Diamond Clarity")) +
  scale_fill_manual(values = c("#EAECEE",
                       "#D5D8DC",
                       "#ABB2B9",
                       "#808B96",
                       "#566573",
                       "#2C3E50",
                       "#273746",
                       "#212F3D"))

clarity_bar
```



Limitation: While an even distribution of Clarity Groups would be useful to predict clarity across groups evenly, we also are working with a population that resembles what may be realistically found on the market. For that reason, we are choosing to stick with the data and work with this limitation.

Bar Chart of Clarity WITH Color Included

```
clarity_color_bar <- diamonds %>%
  ggplot(mapping = aes(x = clarity, fill = color)) +
  geom_bar(color = "black") +
  xlab("Clarity Group") +
  ylab("Number of Entries") +
  ggtitle("Bar Chart of Color Counts Per Clarity Group") +
  theme(plot.title = element_text(hjust = 0.5)) +
  guides(fill = guide_legend(title = "Diamond Color")) +
  scale_fill_manual(values = c("#FFFFFF",
                        "#FFFDCD",
                        "#F9FC97",
                        "#F9F956",
                        "#FEDB08",
                        "#FED101",
                        "#FFC003"))

clarity_color_bar
```



- Note how the colors are evenly distributed through the clarity groups. This suggests a lack of colinerarity between clarity and color. - Our dataset primarily contains Clarity Groups SI1, SI2, VS1 and VS2, whereas I1 and IF only make up 1.4% and 3.3% of the data, respectively.

```r
sum(diamonds$clarity == "I1")/nrow(diamonds)
```

```
## [1] 0.01373672
```

```r
sum(diamonds$clarity == "IF")/nrow(diamonds)
```

```
## [1] 0.03318317
```

```r
sum(diamonds$clarity == "SI1")/nrow(diamonds)
```

```
## [1] 0.2422372
```

```r
sum(diamonds$clarity == "SI2")/nrow(diamonds)
```

```
## [1] 0.1704392
```

```r
sum(diamonds$clarity == "VS1")/nrow(diamonds)
```

```
## [1] 0.1514747
```

```r
sum(diamonds$clarity == "VS2")/nrow(diamonds)
```

```
## [1] 0.2272584
```

```r
sum(diamonds$clarity == "VVS1")/nrow(diamonds)
```

```
## [1] 0.06775671
```

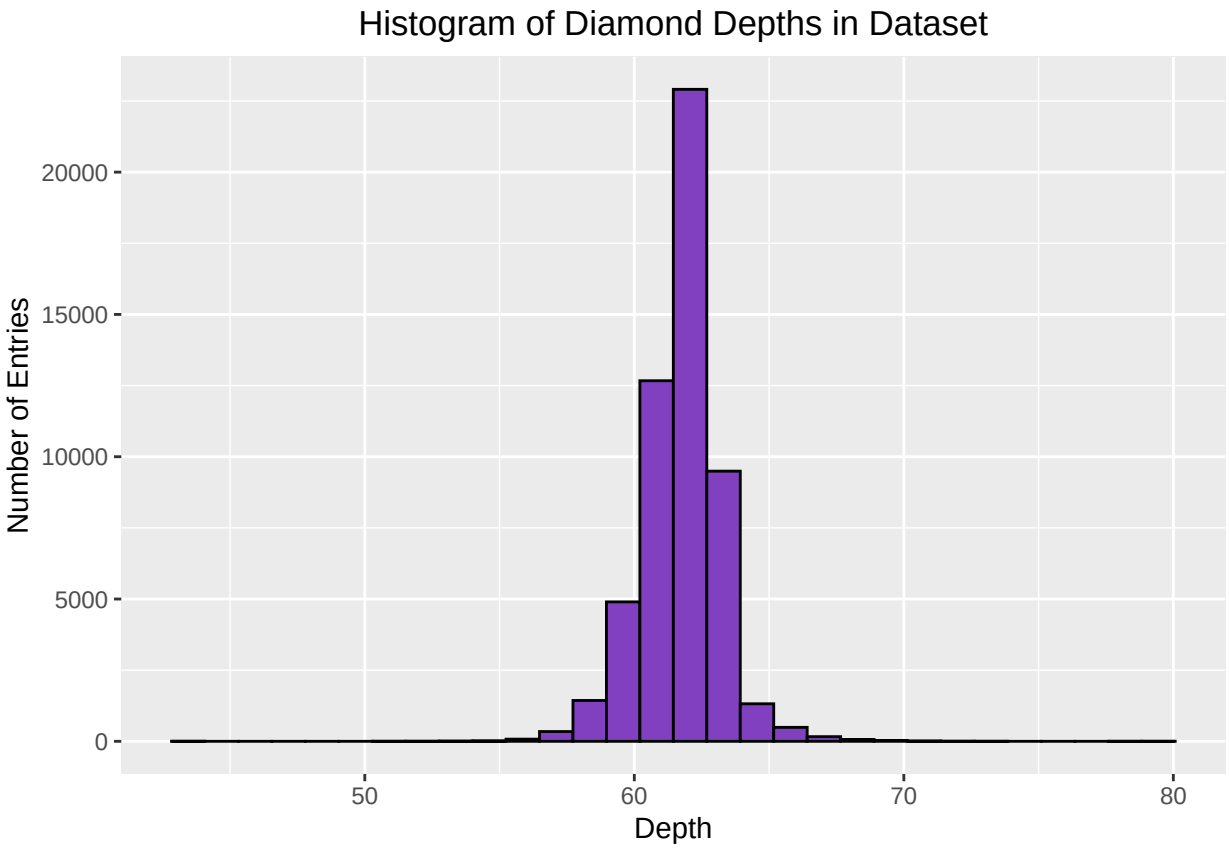
```r
sum(diamonds$clarity == "VVS2")/nrow(diamonds)
```

```
## [1] 0.09391395
```

Depth:

Histogram of Diamond Depth:

```
depth_hist <- diamonds %>%
  ggplot(mapping = aes(x = depth)) +
  geom_histogram(color = "black", fill = "#8040c0", stat = "bin") +
  xlab("Depth") +
  ylab("Number of Entries") +
  ggtitle("Histogram of Diamond Depths in Dataset") +
  theme(plot.title = element_text(hjust = 0.5))

depth_hist
```



- Variables of Depth appear to be normal in distribution. The skew is only -0.08. - *Between 50% of the data lays between depth values of 61.0 and 62.5. There are clear outliers after these depth values. It would be beneficial to remove depth values < 55 and > 70, since there are barely any visible entries in these areas. Removing diamonds in these regions would correspond to removing 22 and 23 diamonds, respectively.*

```
mean(diamonds$depth)
```

```
## [1] 61.74932
```

```r
median(diamonds$depth)
```

```
## [1] 61.8
```

```r
#####
sd(diamonds$depth)
```

```
## [1] 1.432626
```

```r
IQR(diamonds$depth)
```

```
## [1] 1.5
```

```r
mad(diamonds$depth)
```

```
## [1] 1.03782
```

```r
#####
min(diamonds$depth)
```

```
## [1] 43
```

```r
max(diamonds$depth)
```

```
## [1] 79
```

```r
quantile(diamonds$depth)
```

```
##   0%  25%  50%  75% 100%
## 43.0 61.0 61.8 62.5 79.0
```

```r
#####
skewness(diamonds$depth)
```

```
## [1] -0.08218493
```

```r
#####
sum(diamonds$depth < 55)
```

```
## [1] 22
```

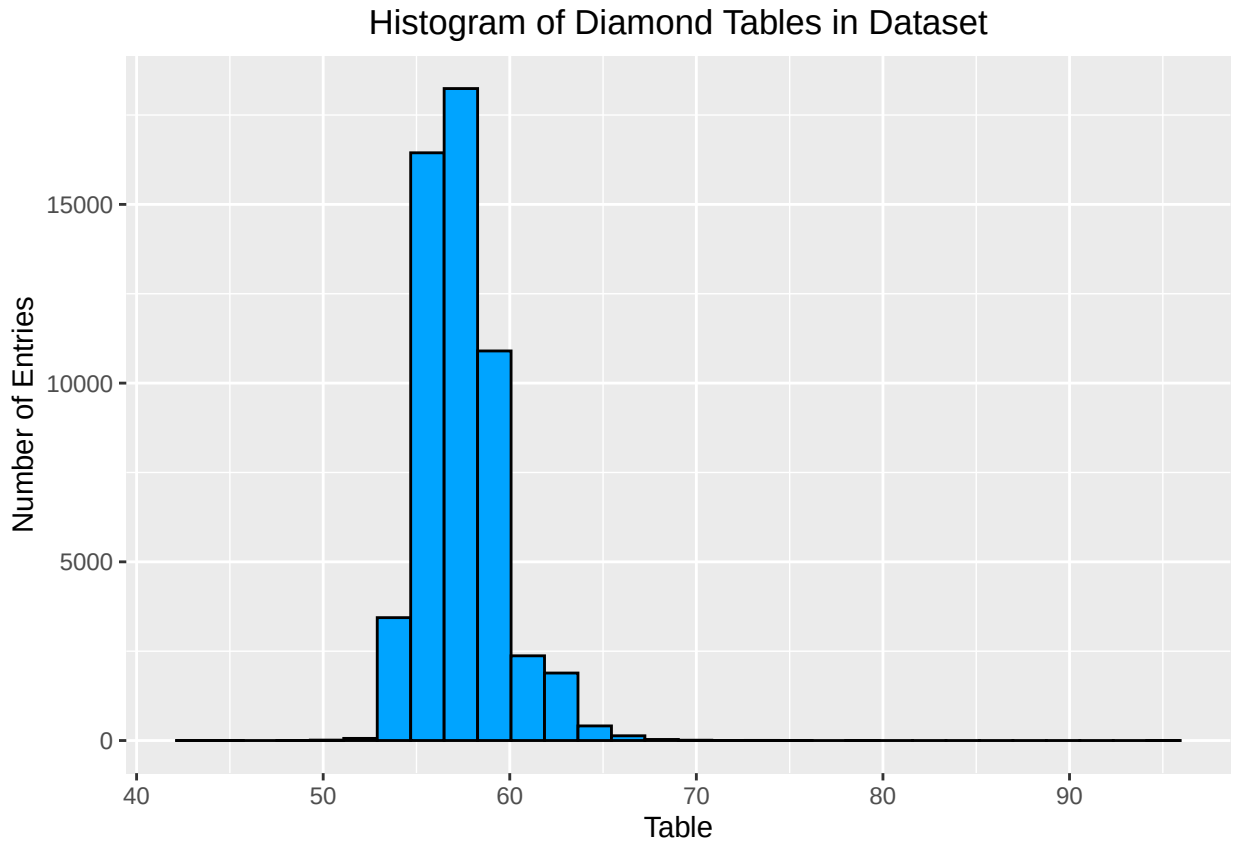
```r
sum(diamonds$depth > 70)
```

```
## [1] 23
```

Table:

Histogram of Diamond Table

```
table_hist <- diamonds %>%
  ggplot(mapping = aes(x = table)) +
  geom_histogram(color = "black", fill = "#00a4ff", stat = "bin") +
  xlab("Table") +
  ylab("Number of Entries") +
  ggtitle("Histogram of Diamond Tables in Dataset") +
  theme(plot.title = element_text(hjust = 0.5))

table_hist
```



- The skew in Table likewise appears to be minimal, with a skew value of only 0.08 as well. - *It would be beneficial to remove table values < 50 and > 70, based on the minuscule number of diamonds outside the range of [50,70]. There are only 4 and 8 diamonds with a table < 50 and table > 70, respectively.*

```
mean(diamonds$table)
```

```
## [1] 57.45725
```

```
median(diamonds$table)
```

```
## [1] 57
```

```
#####
sd(diamonds$table)
```

```
## [1] 2.234549
```

```
IQR(diamonds$table)
```

```
## [1] 3
```

```
mad(diamonds$table)
```

```
## [1] 1.4826
```

```
#####
min(diamonds$table)
```

```
## [1] 43
```

```
max(diamonds$table)
```

```
## [1] 95
```

```
quantile(diamonds$table)
```

```
##   0%  25%  50%  75% 100%
##   43   56   57   59   95
```

```
#####
skewness(diamonds$table)
```

```
## [1] 0.7968138
```

```
#####
sum(diamonds$table < 50)
```

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

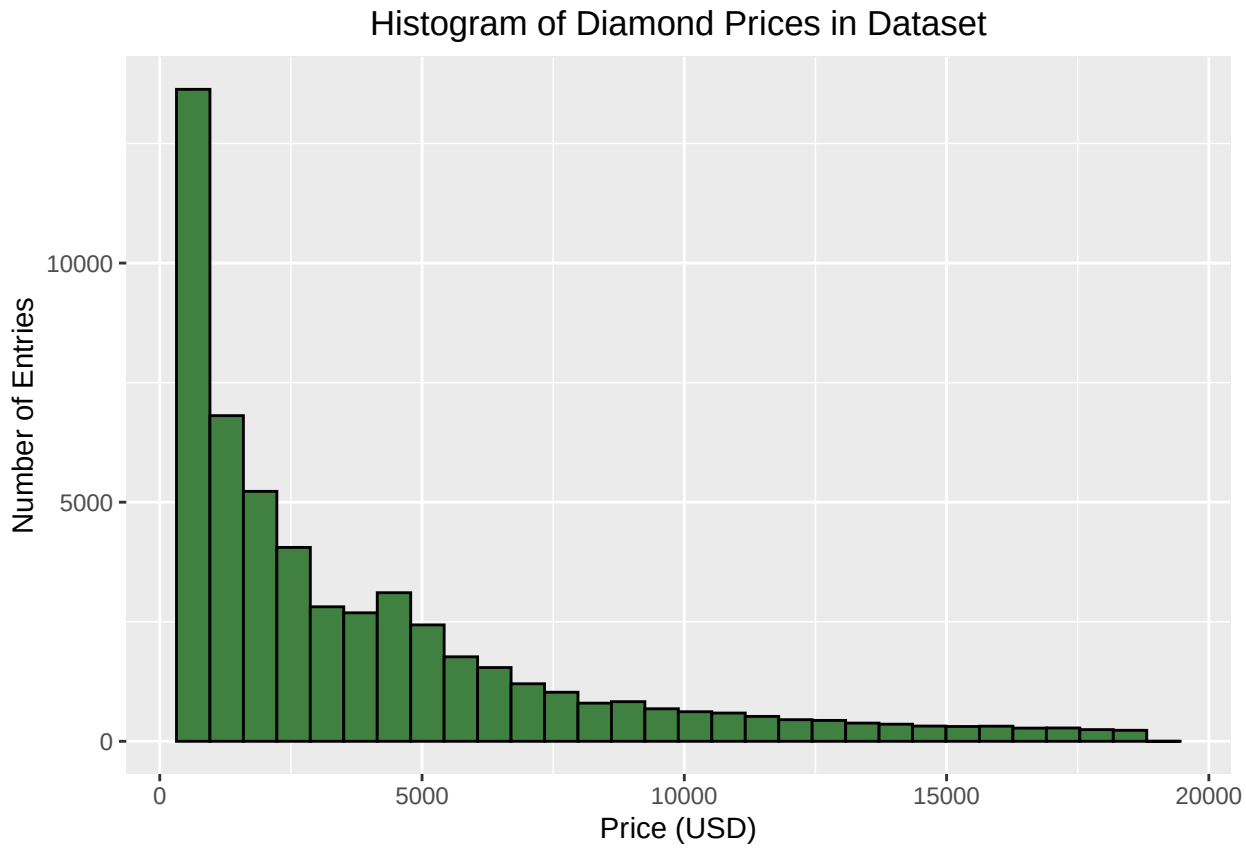
```
sum(diamonds$table > 70)
```

```
## [1] 8
```

Price:

Histogram of Diamond Price

```
price_hist <- diamonds %>%
  ggplot(mapping = aes(x = price)) +
  geom_histogram(color = "black", fill = "#408040", stat = "bin") +
  xlab("Price (USD)") +
  ylab("Number of Entries") +
  ggtitle("Histogram of Diamond Prices in Dataset") +
  theme(plot.title = element_text(hjust = 0.5))

price_hist
```



- There is a clear positive skew in the Price evaluations of our dataset, and there is likewise a thick tail as price -> max. The skew value is 1.618431. - This distribution is also skewed similarly to the carat histogram. - In this case, our model would be mostly effective at explaining prices on the lower end (ex, 0 dollars to 5000 dollars). Our model would possess less data to work with above the 5000 dollar range, although approximately 14714 data entries would be above the 5000 range.

```
mean(diamonds$price)
```

```
## [1] 3932.734
```

```r
median(diamonds$price)
```

```
## [1] 2401
```

```r
#####
sd(diamonds$price)
```

```
## [1] 3989.338
```

```r
IQR(diamonds$price)
```

```
## [1] 4374
```

```r
mad(diamonds$price)
```

```
## [1] 2475.942
```

```r
#####
min(diamonds$price)
```

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

```r
max(diamonds$price)
```

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

```r
quantile(diamonds$price)
```

```
##    0%   25%   50%   75%  100%
##   326   950  2401  5324 18823
```

```r
#####
skewness(diamonds$price)
```

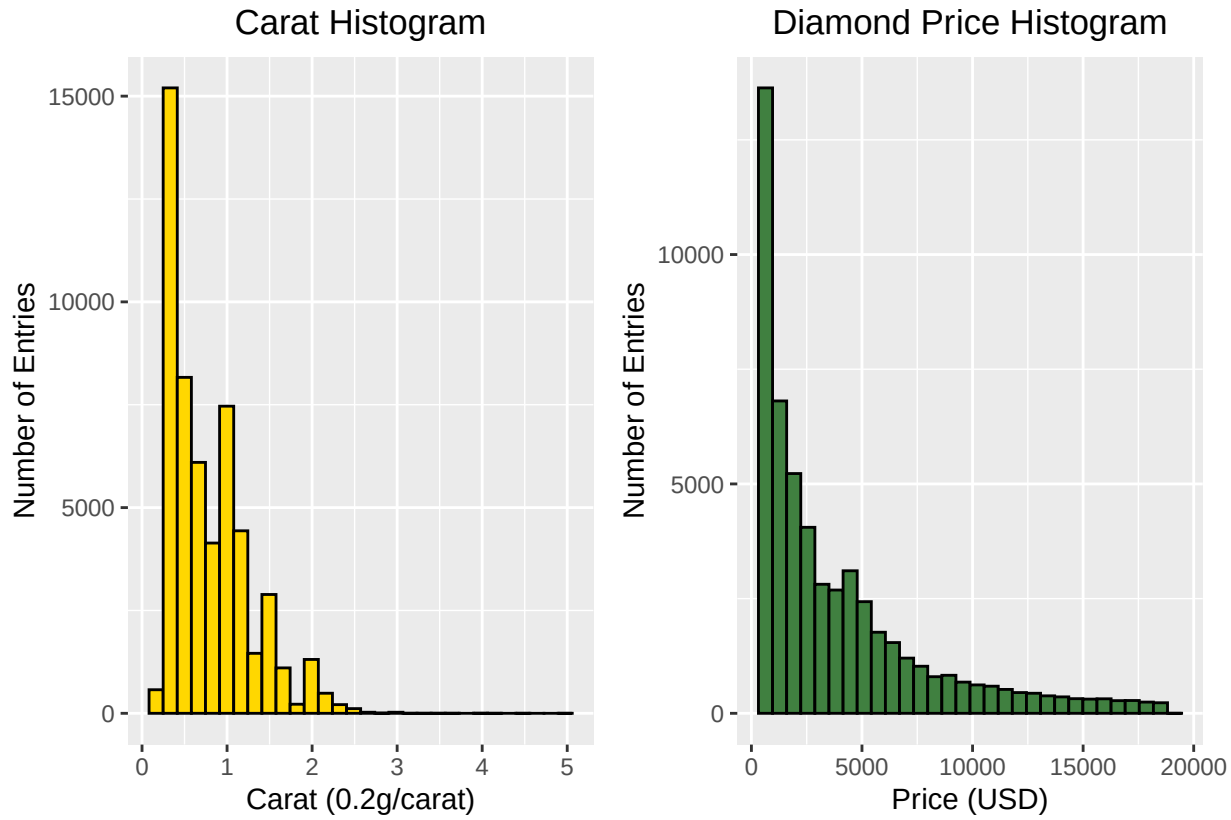
```
## [1] 1.618431
```

```r
#####
sum(diamonds$price > 5000)
```

```
## [1] 14714
```

```r
carat_hist_a <- diamonds %>%
  ggplot(mapping = aes(x = carat)) +
  geom_histogram(color = "black", fill = "gold", stat = "bin") +
  xlab("Carat (0.2g/carat)") +
  ylab("Number of Entries") +
```

```
  ggtitle("Carat Histogram") +
  theme(plot.title = element_text(hjust = 0.5))

price_hist_a <- diamonds %>%
  ggplot(mapping = aes(x = price)) +
  geom_histogram(color = "black", fill = "#408040", stat = "bin") +
  xlab("Price (USD)") +
  ylab("Number of Entries") +
  ggtitle("Diamond Price Histogram") +
  theme(plot.title = element_text(hjust = 0.5))

carat_hist_a + price_hist_a
```



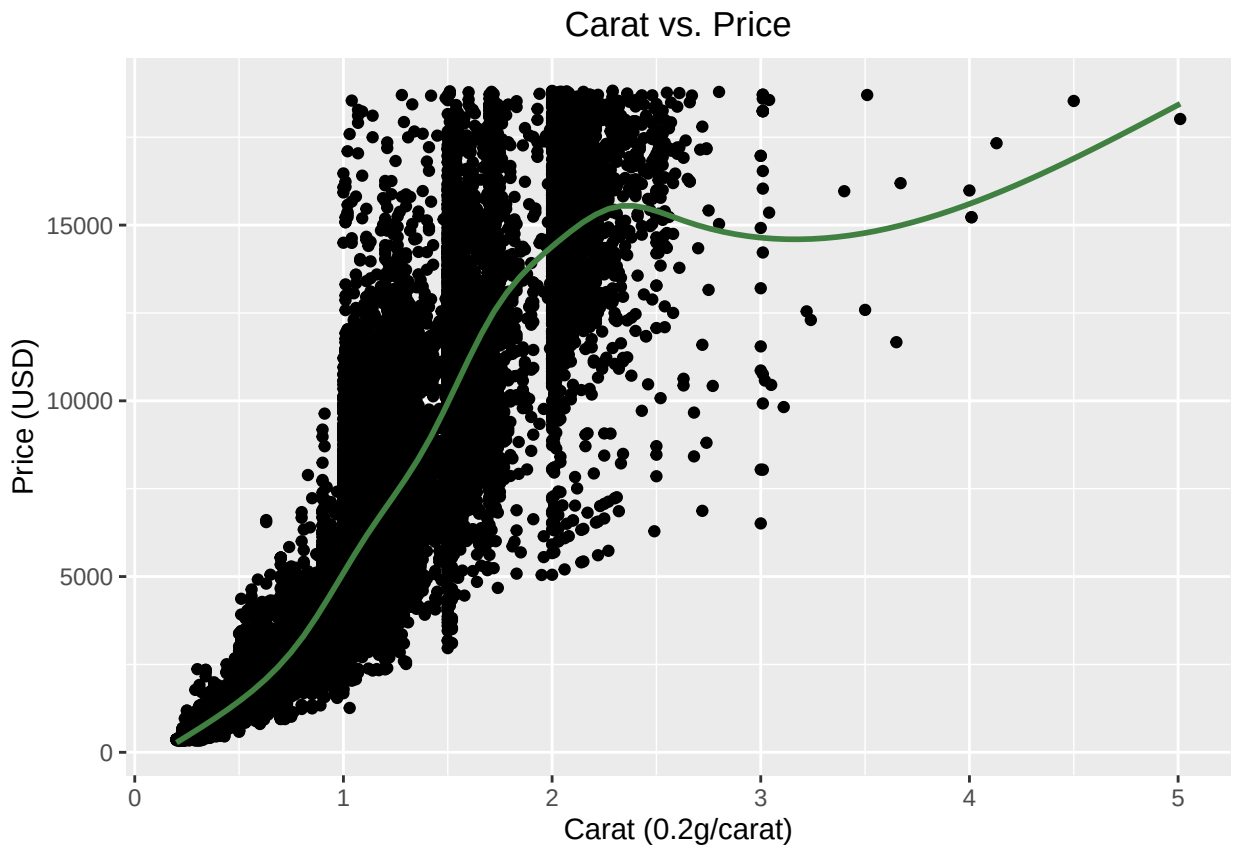
Through visual glance alone, the likeness in skew and in distribution between Carat and Price suggests that variations in price are greatly associated with variations in carat.

Plotting Metric Data Against Price:

Carat vs. Price:

```
carat_price_plot <- diamonds %>%
  ggplot(mapping = aes(x = carat, y = price)) +
  geom_point(color = "black") +
  geom_smooth(color = "#408040",
```

```
              se = FALSE) +
  ggtitle("Carat vs. Price") +
  xlab("Carat (0.2g/carat)") +
  ylab("Price (USD)") +
  theme(plot.title = element_text(hjust = 0.5))


carat_price_plot
```

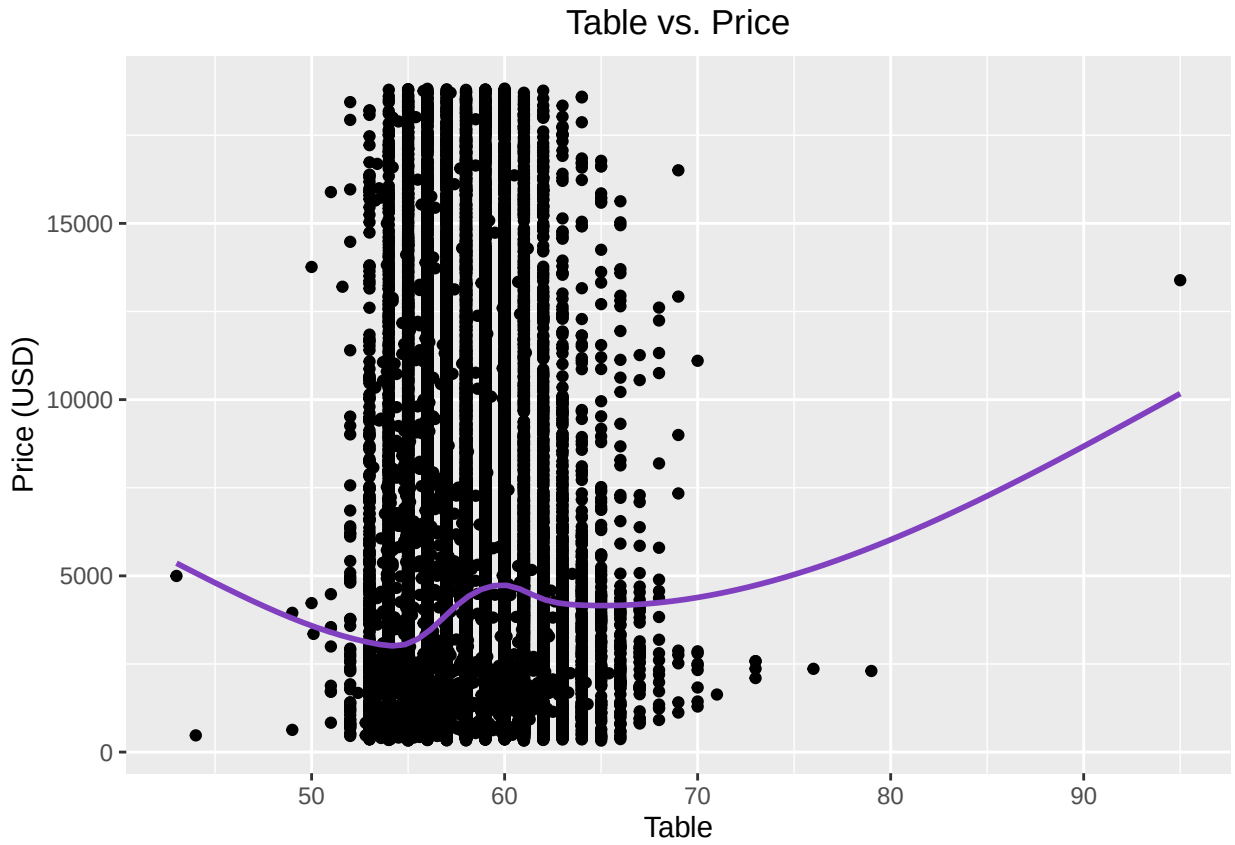


Adding to the comparison between histograms for Carat and Price, we can see here that variations in Carat do align greatly with variations in Price. On a related note, note how the relationship between carat and price mildly resembles an exponential curvature. Once carat reaches approximately 2.3, the smooth plot deviates from this exponential or quadratic curvature. This exponential curvature suggests that we should model a relationship between price and carat that is exponential.

Table vs. Price:

```
table_price_plot <- diamonds %>%
  ggplot(mapping = aes(x = table, y = price)) +
  geom_point(color = "black") +
  geom_smooth(color = "#8040c0",
              se = FALSE) +
  ggtitle("Table vs. Price") +
  xlab("Table") +
```

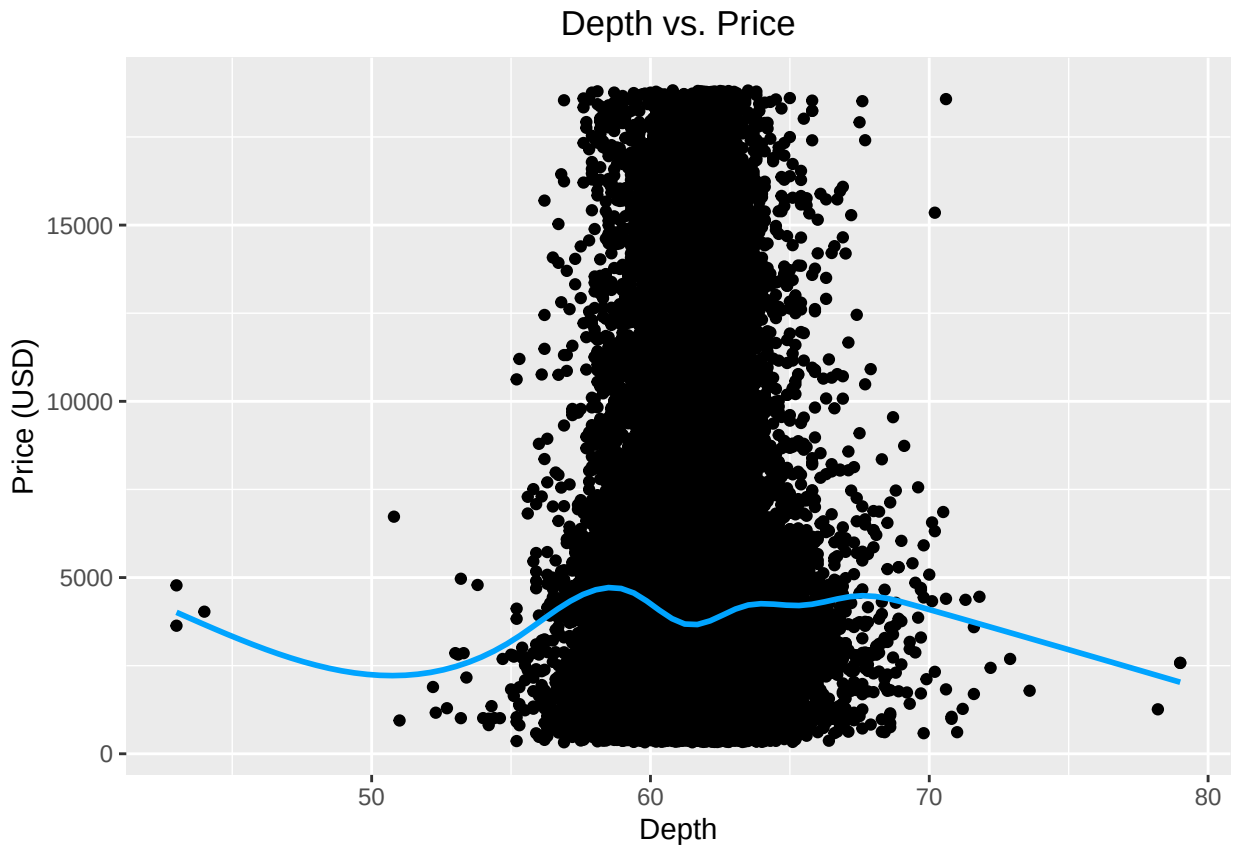
```
  ylab("Price (USD)") +
  theme(plot.title = element_text(hjust = 0.5))


table_price_plot
```



There does not appear to be any relationship between Table and Price. The outliers in table can likewise be visualized in this graph.

Depth vs. Price:

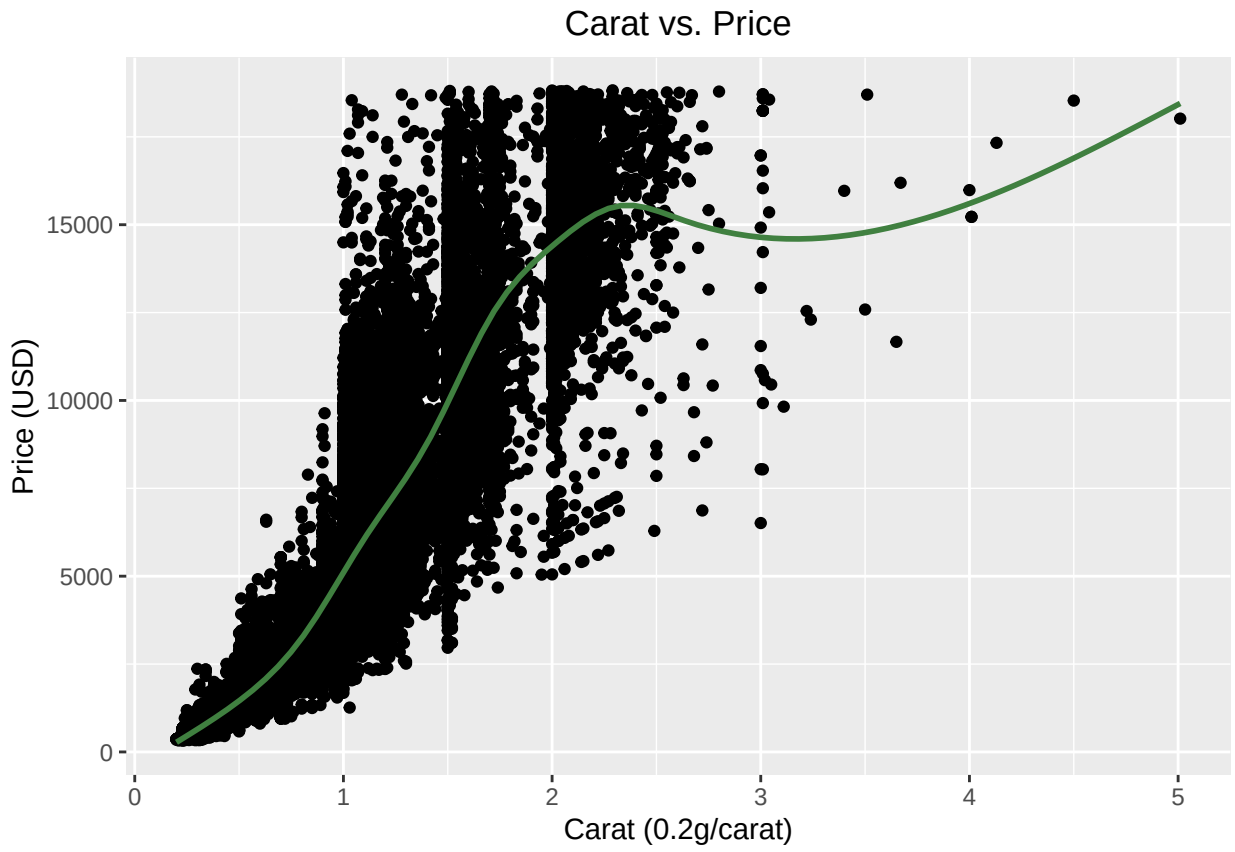
```
depth_price_plot <- diamonds %>%
  ggplot(mapping = aes(x = depth, y = price)) +
  geom_point(color = "black") +
  geom_smooth(color = "#00a4ff",
              se = FALSE) +
  ggtitle("Depth vs. Price") +
  xlab("Depth") +
  ylab("Price (USD)") +
  theme(plot.title = element_text(hjust = 0.5))


depth_price_plot
```



There does not appear to be any relationship between Depth and Price. The outliers in depth can likewise be visualized in this graph.

Boxplots of Categorical Data:

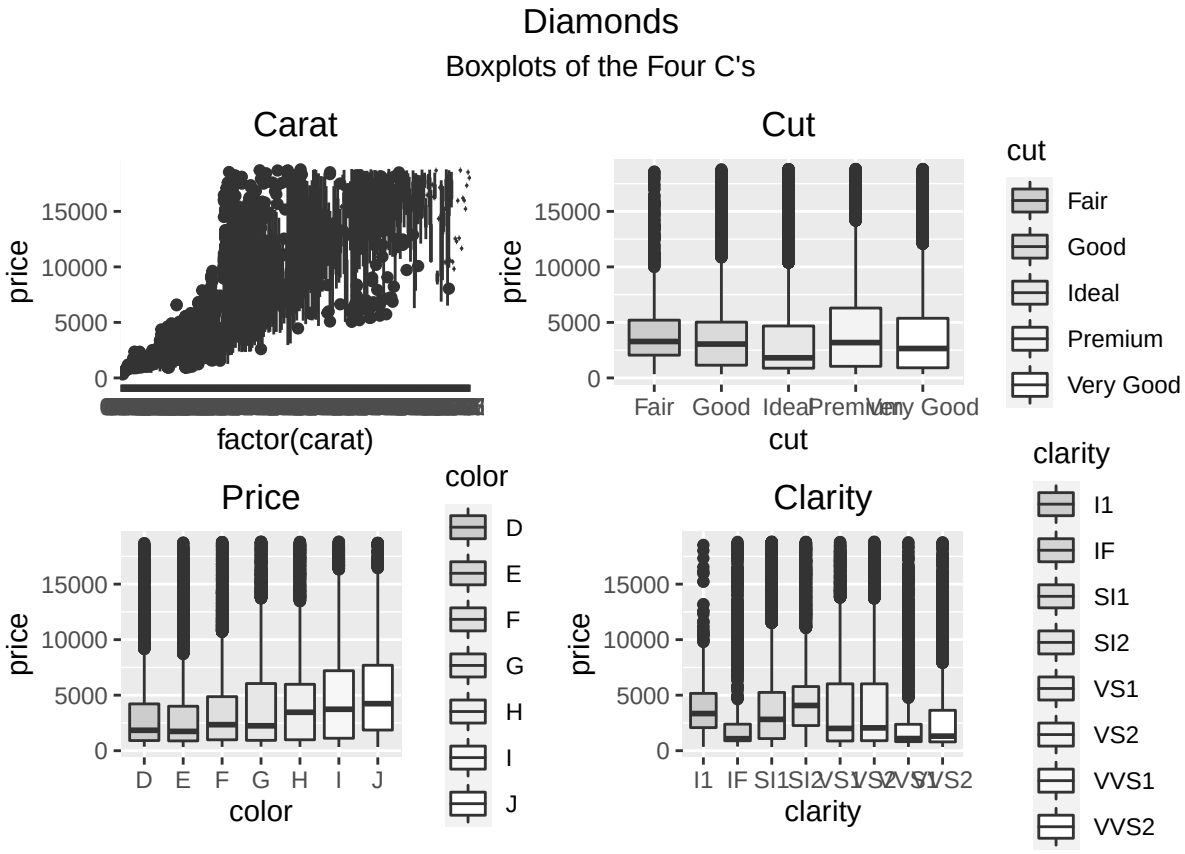
```
carat_price_plot
```



Boxplots for Carat, Cut, Color and Clarity

```
pb1 <- ggplot(diamonds, aes(factor(carat), price)) + geom_boxplot() +
  scale_fill_grey(start = 0.8, end = 1) +
  ggtitle("Carat") +
  theme(plot.title = element_text(hjust = 0.5))

pb2 <- ggplot(diamonds, aes(cut, price, fill = cut)) + geom_boxplot() +
  scale_fill_grey(start = 0.8, end = 1) +
  ggtitle("Cut") +
  theme(plot.title = element_text(hjust = 0.5))

pb3 <- ggplot(diamonds, aes(color, price, fill = color)) + geom_boxplot() +
  scale_fill_grey(start = 0.8, end = 1) +
  ggtitle("Price") +
  theme(plot.title = element_text(hjust = 0.5))

pb4 <- ggplot(diamonds, aes(clarity, price, fill = clarity)) + geom_boxplot() +
  scale_fill_grey(start = 0.8, end = 1) +
  ggtitle("Clarity") +
  theme(plot.title = element_text(hjust = 0.5))

gpb1 <- pb1 + pb2
gpb2 <- pb3 + pb4
```

```
plot <- gpb1 / gpb2

plot + plot_annotation(
  title = "Diamonds",
  subtitle = "Boxplots of the Four C's",
  theme = theme(plot.title = element_text(hjust = 0.5),
                plot.subtitle = element_text(hjust = 0.5)))
```



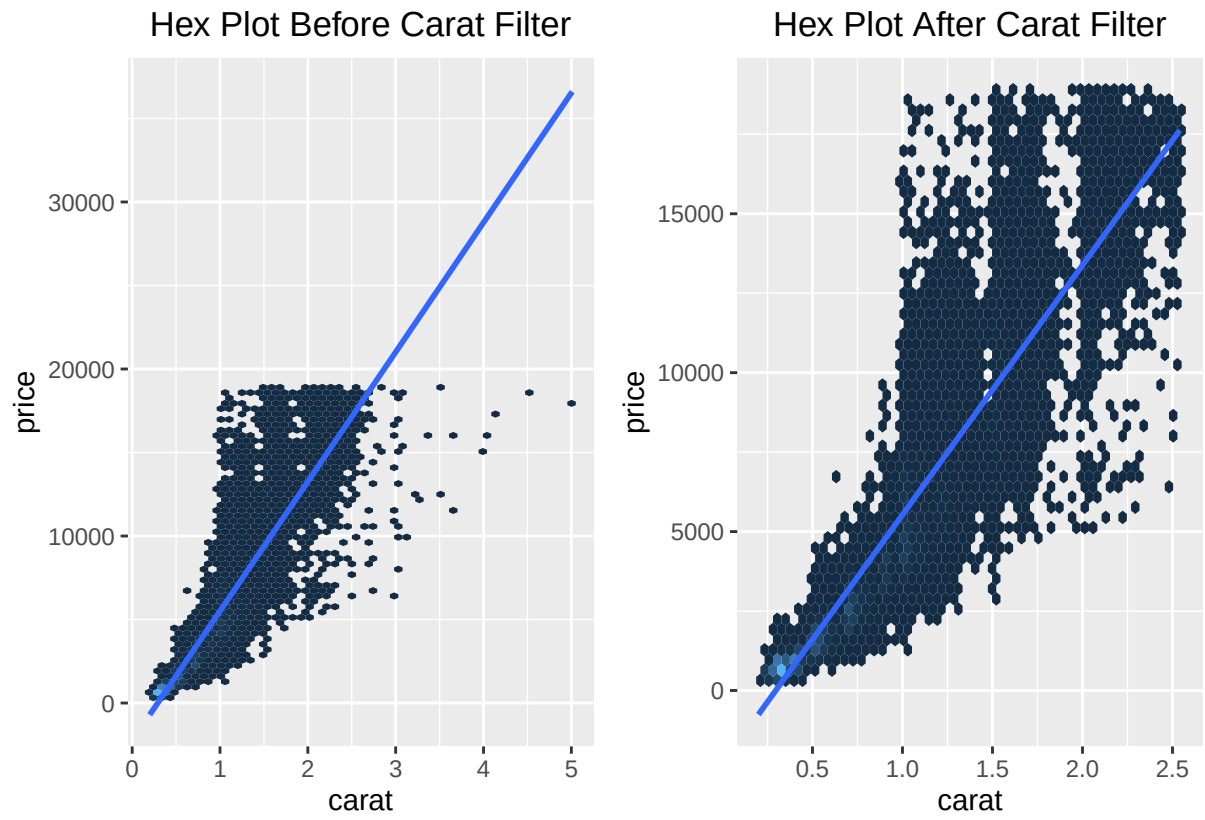
Data Transformation Candidates:

1. Remove Carat Values Above 2.54:

The EDA demonstrates that carat values above 2.54 represent a minuscule minority of the carat sample group, but likewise greatly skew the dataset. Removing these datapoints creates the following contrast to Carat vs. Price:

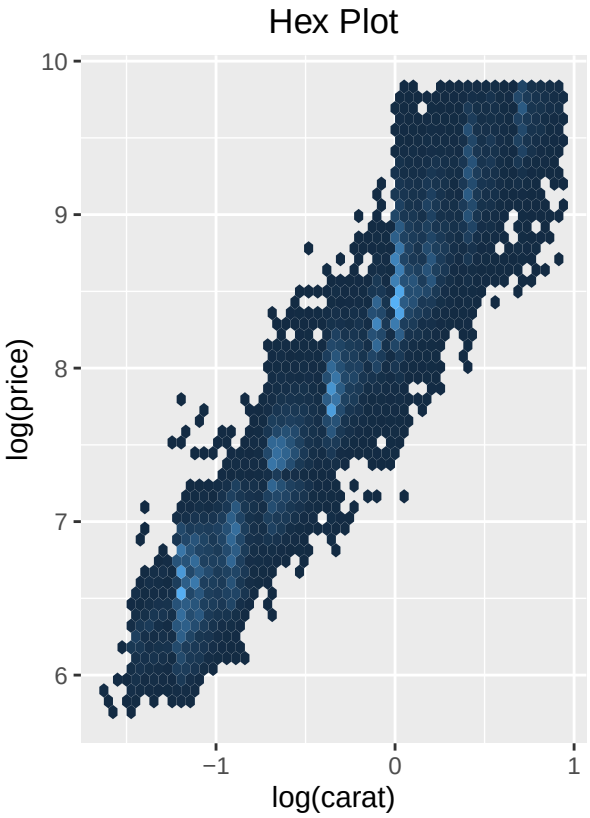
```
pre_filt_carat <- diamonds %>%
  ggplot(mapping = aes(carat, price)) +
  geom_hex(bins=50) +
  ggtitle("Hex Plot Before Carat Filter") +
  geom_smooth(method = "lm") +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position="none")
```

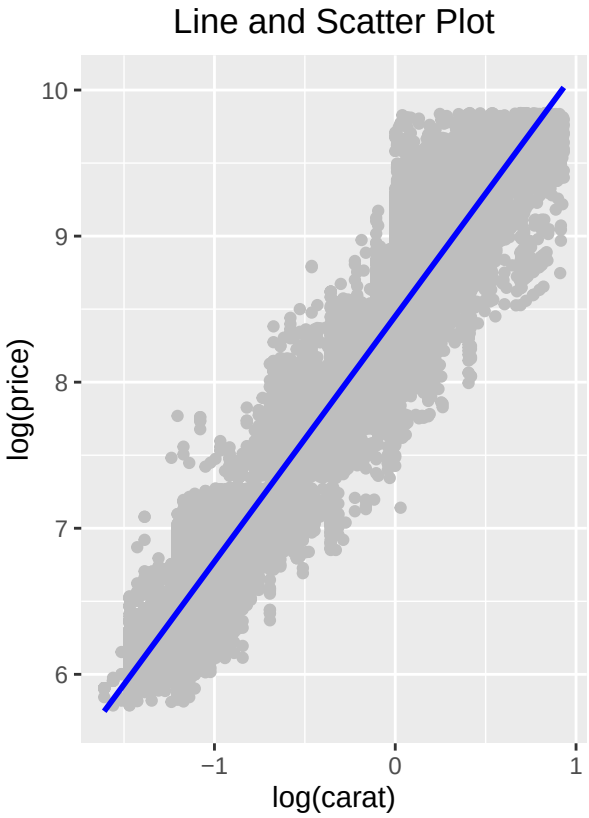
```
post_filt_carat <- diamonds %>%
  filter(carat <= 2.54) %>%
  ggplot(mapping = aes(carat, price)) +
  geom_hex(bins=50) +
  geom_smooth(method = "lm") +
  ggtitle("Hex Plot After Carat Filter") +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position="none")

pre_filt_carat + post_filt_carat
```



Secondly, due to the quadratic nature of the filtered hex plot, plotting the log(price) vs log(carat) reveals a linear relationship between the logs:

```
hex_log <- diamonds %>%
  filter(carat <= 2.54) %>%
  mutate(lprice=log(price), lcarat=log(carat)) %>%
  ggplot(mapping = aes(lcarat, lprice)) + geom_hex(bins=50) +
  xlab("log(carat)") +
  ylab("log(price)") +
  ggtitle("Hex Plot") +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position="none")

line_log <- diamonds %>%
  filter(carat <= 2.54) %>%
  mutate(lprice=log(price), lcarat=log(carat)) %>%
  ggplot(mapping = aes(lcarat, lprice)) +
  geom_point(color = "grey") +
  geom_smooth(color = "blue",
              method = "lm",
              se = TRUE) +
  xlab("log(carat)") +
  ylab("log(price)") +
  ggtitle("Line and Scatter Plot") +
  theme(plot.title = element_text(hjust = 0.5))

hex_log + line_log
```


Hex Plot
log(price)
log(carat)
10
9
8
7
6
−1
0
1


Line and Scatter Plot
log(price)
log(carat)
10
9
8
7
6
−1
0
1

Remove Outliers from Depth and Table:

As mentioned in the EDA, we will be removing depth values < 55 and > 70. We will similarly be removing Table values < 50 and >70. Please see the EDA section for further visual and computational justification.

Apply Proposed Transformations:

```
### Pre-Filter Carat:
pre_carat <- coef(lm(price ~ carat, data = diamonds))

diamonds <- diamonds %>%
  filter(carat <= 2.54)

### Post-Filter Carat:
post_carat <- coef(lm(price ~ carat, data = diamonds))

### Pre-Filter Table and Depth Outliers Proposed in EDA
diamonds <- diamonds %>%
  filter((55 <= depth) & (70 >= depth) & (50 <= table) & (70 >= table))

### note how pre_slope has a smaller slope than post_carat (underestimates the whole curve)
print(pre_carat)
```

```
## (Intercept)       carat
##   -2256.395    7756.436
```

```
print(post_carat)
```

```
## (Intercept)       carat
##   -2325.346    7854.191
```

```
diamonds$logprice <- log(diamonds$price)

diamonds$logcarat <- log(diamonds$carat)
```

Split Data 30/70

```
## Establish 30% of Sample Size:
sample_size <- floor(0.30 * nrow(diamonds))

## Prepare a Seed for the Data and Create a Random Sample using sample():
set.seed(514)
train_indices <- sample(seq_len(nrow(diamonds)), size = sample_size)

diamonds_train_set <- diamonds[train_indices, ]
diamonds_test_set <- diamonds[-train_indices, ] #Selects indices that are not the training indices
```

Creation of Models:

Model Specs:

The bolded models will be the three models we will present on for the PPT. We took measures to perform statistical assumptions on all the remaining variables as well.

non_log_models:

- Model 1: two_c_model: Price as a function of carat and clarity.
- Model 2: three_c_model: Price as a function of carat, clarity and cut.
- Model 3: four_c_model: Price as a function of carat, clarity, cut and color.
- Model 4: all_var_model: Price as a function of carat, clarity, cut, color, depth and table.
  ### log_models: - *Model 5: log_model_1: log(price) as a function of log(carat).*
- Model 6: log_model_2: log(price) as a function of log(carat) and clarity.
  - *Model 7: log_model_3: log(price) as a function of log(carat), clarity, cut and color.*
  - *Model 8: log_model_4: log(price) as a function of log(carat), clarity, cut, color, depth and table.*

```
two_c_model <- lm(price ~ carat + clarity, data = diamonds_train_set)
three_c_model <- lm(price ~ carat + clarity + cut, data = diamonds_train_set)
four_c_model <- lm(price ~ carat + clarity + cut + color, data = diamonds_train_set)
all_var_model <- lm(price ~ carat + clarity + cut + color + depth + table, data = diamonds_train_set)
log_model_1 <- lm(logprice ~ logcarat, data = diamonds_train_set)
log_model_2 <- lm(logprice ~ logcarat + clarity, data = diamonds_train_set)
log_model_3 <- lm(logprice ~ logcarat + clarity + cut + color, data = diamonds_train_set)
log_model_4 <- lm(logprice ~ logcarat + clarity + cut + color + depth + table, data = diamonds_train_se
```

create LaTeX Math Expressions for the models.

$$Model_1 : price = \beta_0 + \beta_1 \cdot carat + \beta_2 \cdot clarity$$

$$Model_2 : price = \beta_0 + \beta_1 \cdot carat + \beta_2 \cdot clarity + \beta_3 \cdot cut$$

$$Model_3 : price = \beta_0 + \beta_1 \cdot carat + \beta_2 \cdot clarity + \beta_3 \cdot cut + \beta_4 \cdot color$$

$$Model_4 : price = \beta_0 + \beta_1 \cdot carat + \beta_2 \cdot clarity + \beta_3 \cdot cut + \beta_4 \cdot color + \beta_5 \cdot table + \beta_6 \cdot depth$$

$$Model_5 : ln(price) = \beta_0 + \beta_1 \cdot ln(carat)$$

$$Model_6 : ln(price) = \beta_0 + \beta_1 \cdot ln(carat) + \beta_2 \cdot clarity$$

$$Model_7 : ln(price) = \beta_0 + \beta_1 \cdot ln(carat) + \beta_2 \cdot clarity + \beta_3 \cdot cut + \beta_4 \cdot color$$

$$Model_8 : price = \beta_0 + \beta_1 \cdot carat + \beta_2 \cdot clarity + \beta_3 \cdot cut + \beta_4 \cdot color + \beta_5 \cdot table + \beta_6 \cdot depth$$

Stargazer Tables of Models:

To be completed (take from Alejandro)

```
summary(two_c_model)
```

```
##
## Call:
## lm(formula = price ~ carat + clarity, data = diamonds_train_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6906.9  -637.3   -95.5   497.7 10522.1
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6662.69      95.25  -69.95   <2e-16 ***
## carat        8569.46      23.17  369.80   <2e-16 ***
## clarityIF    5250.26     107.58   48.80   <2e-16 ***
## claritySI1   3374.56      93.80   35.98   <2e-16 ***
## claritySI2   2504.24      94.40   26.53   <2e-16 ***
## clarityVS1   4292.07      95.33   45.02   <2e-16 ***
## clarityVS2   4027.64      94.11   42.80   <2e-16 ***
## clarityVVS1  4864.00     100.06   48.61   <2e-16 ***
## clarityVVS2  4822.93      97.91   49.26   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1274 on 16132 degrees of freedom
## Multiple R-squared:  0.8974, Adjusted R-squared:  0.8974
## F-statistic: 1.764e+04 on 8 and 16132 DF,  p-value: < 2.2e-16
```

```
summary(three_c_model)
```

```
##
## Call:
## lm(formula = price ~ carat + clarity + cut, data = diamonds_train_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6275.0  -637.3  -102.1   496.0 10604.9
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -7372.05     104.29  -70.69   <2e-16 ***
## carat         8600.40      23.07  372.86   <2e-16 ***
## clarityIF     4920.97     107.97   45.58   <2e-16 ***
## claritySI1    3132.36      94.09   33.29   <2e-16 ***
## claritySI2    2277.05      94.52   24.09   <2e-16 ***
## clarityVS1    4015.93      95.72   41.96   <2e-16 ***
## clarityVS2    3758.82      94.48   39.79   <2e-16 ***
## clarityVVS1   4559.32     100.52   45.36   <2e-16 ***
```

```
## clarityVVS2   4533.89      98.32   46.11   <2e-16 ***
## cutGood        744.96      68.68   10.85   <2e-16 ***
## cutIdeal      1085.08      62.60   17.33   <2e-16 ***
## cutPremium     922.74      63.08   14.63   <2e-16 ***
## cutVery Good   914.31      63.89   14.31   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1260 on 16128 degrees of freedom
## Multiple R-squared:  0.8996, Adjusted R-squared:  0.8996
## F-statistic: 1.205e+04 on 12 and 16128 DF,  p-value: < 2.2e-16
```

```
summary(four_c_model)
```

```
##
## Call:
## lm(formula = price ~ carat + clarity + cut + color, data = diamonds_train_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7102.1  -683.2  -194.1   480.2  9711.6
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -7112.42      96.22 -73.920  < 2e-16 ***
## carat         9032.57      21.86 413.205  < 2e-16 ***
## clarityIF     5154.39      96.91  53.188  < 2e-16 ***
## claritySI1    3212.47      84.31  38.104  < 2e-16 ***
## claritySI2    2260.60      84.68  26.697  < 2e-16 ***
## clarityVS1    4203.89      85.81  48.993  < 2e-16 ***
## clarityVS2    3849.85      84.64  45.483  < 2e-16 ***
## clarityVVS1   4754.56      90.13  52.751  < 2e-16 ***
## clarityVVS2   4631.84      88.12  52.563  < 2e-16 ***
## cutGood        674.77      61.53  10.967  < 2e-16 ***
## cutIdeal      1004.90      56.09  17.917  < 2e-16 ***
## cutPremium     860.56      56.51  15.229  < 2e-16 ***
## cutVery Good   829.57      57.24  14.492  < 2e-16 ***
## colorE        -221.74      32.63  -6.796 1.11e-11 ***
## colorF        -314.19      33.19  -9.468  < 2e-16 ***
## colorG        -512.93      32.39 -15.835  < 2e-16 ***
## colorH        -999.06      34.50 -28.960  < 2e-16 ***
## colorI       -1490.45      38.83 -38.387  < 2e-16 ***
## colorJ       -2378.17      47.94 -49.605  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1129 on 16122 degrees of freedom
## Multiple R-squared:  0.9195, Adjusted R-squared:  0.9194
## F-statistic: 1.023e+04 on 18 and 16122 DF,  p-value: < 2.2e-16
```

```
summary(all_var_model)
```

```
##
```

```
## Call:
## lm(formula = price ~ carat + clarity + cut + color + depth +
##     table, data = diamonds_train_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7183.7  -678.7  -192.5   477.0  9687.7
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -3727.540    693.538  -5.375 7.78e-08 ***
## carat         9043.675     21.919 412.588  < 2e-16 ***
## clarityIF     5125.886     96.992  52.849  < 2e-16 ***
## claritySI1    3195.796     84.282  37.918  < 2e-16 ***
## claritySI2    2243.598     84.669  26.498  < 2e-16 ***
## clarityVS1    4184.361     85.819  48.758  < 2e-16 ***
## clarityVS2    3832.088     84.635  45.278  < 2e-16 ***
## clarityVVS1   4732.476     90.157  52.491  < 2e-16 ***
## clarityVVS2   4610.530     88.161  52.297  < 2e-16 ***
## cutGood        627.575     62.821   9.990  < 2e-16 ***
## cutIdeal       857.538     62.539  13.712  < 2e-16 ***
## cutPremium     789.884     60.297  13.100  < 2e-16 ***
## cutVery Good   747.081     60.308  12.388  < 2e-16 ***
## colorE        -220.572     32.594  -6.767 1.36e-11 ***
## colorF        -315.573     33.151  -9.519  < 2e-16 ***
## colorG        -516.100     32.381 -15.938  < 2e-16 ***
## colorH        -997.593     34.486 -28.927  < 2e-16 ***
## colorI       -1489.730     38.802 -38.393  < 2e-16 ***
## colorJ       -2374.393     47.905 -49.564  < 2e-16 ***
## depth          -22.798      7.591  -3.003  0.00267 **
## table          -32.504      5.439  -5.977 2.33e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1127 on 16120 degrees of freedom
## Multiple R-squared:  0.9197, Adjusted R-squared:  0.9196
## F-statistic:  9232 on 20 and 16120 DF,  p-value: < 2.2e-16
```

```
summary(log_model_1)
```

```
##
## Call:
## lm(formula = logprice ~ logcarat, data = diamonds_train_set)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.14118 -0.16997 -0.00582  0.16526  1.25849
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 8.455079   0.002463  3432.7   <2e-16 ***
## logcarat    1.679751   0.003519   477.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.2598 on 16139 degrees of freedom
## Multiple R-squared:  0.9339, Adjusted R-squared:  0.9339
## F-statistic: 2.279e+05 on 1 and 16139 DF,  p-value: < 2.2e-16
```

```
summary(log_model_2)
```

```
##
## Call:
## lm(formula = logprice ~ logcarat + clarity, data = diamonds_train_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.7249 -0.1208  0.0109  0.1238  0.8537
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 7.784586   0.013314  584.68   <2e-16 ***
## logcarat    1.810727   0.002743  660.02   <2e-16 ***
## clarityIF   1.109462   0.015732   70.52   <2e-16 ***
## claritySI1  0.611671   0.013684   44.70   <2e-16 ***
## claritySI2  0.465130   0.013775   33.77   <2e-16 ***
## clarityVS1  0.811507   0.013913   58.33   <2e-16 ***
## clarityVS2  0.763006   0.013734   55.56   <2e-16 ***
## clarityVVS1 1.019197   0.014633   69.65   <2e-16 ***
## clarityVVS2 0.963008   0.014303   67.33   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1859 on 16132 degrees of freedom
## Multiple R-squared:  0.9662, Adjusted R-squared:  0.9661
## F-statistic: 5.758e+04 on 8 and 16132 DF,  p-value: < 2.2e-16
```

```
summary(log_model_3)
```

```
##
## Call:
## lm(formula = logprice ~ logcarat + clarity + cut + color, data = diamonds_train_set)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.54473 -0.08523 -0.00064  0.08143  0.94156
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.867727   0.010877  723.35   <2e-16 ***
## logcarat      1.888449   0.002042  924.82   <2e-16 ***
## clarityIF     1.115268   0.011297   98.73   <2e-16 ***
## claritySI1    0.585370   0.009807   59.69   <2e-16 ***
## claritySI2    0.422586   0.009853   42.89   <2e-16 ***
## clarityVS1    0.807984   0.009983   80.94   <2e-16 ***
## clarityVS2    0.735066   0.009848   74.64   <2e-16 ***
## clarityVVS1   1.018201   0.010507   96.91   <2e-16 ***
```

```
## clarityVVS2   0.939428   0.010262   91.54   <2e-16 ***
## cutGood       0.080178   0.007160   11.20   <2e-16 ***
## cutIdeal      0.158075   0.006531   24.20   <2e-16 ***
## cutPremium    0.133491   0.006578   20.30   <2e-16 ***
## cutVery Good  0.115092   0.006663   17.27   <2e-16 ***
## colorE       -0.053633   0.003797  -14.13   <2e-16 ***
## colorF       -0.095078   0.003863  -24.61   <2e-16 ***
## colorG       -0.157832   0.003770  -41.87   <2e-16 ***
## colorH       -0.248673   0.004006  -62.08   <2e-16 ***
## colorI       -0.372645   0.004496  -82.88   <2e-16 ***
## colorJ       -0.504493   0.005550  -90.91   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1313 on 16122 degrees of freedom
## Multiple R-squared:  0.9831, Adjusted R-squared:  0.9831
## F-statistic: 5.216e+04 on 18 and 16122 DF,  p-value: < 2.2e-16
```

```
summary(log_model_4)
```

```
##
## Call:
## lm(formula = logprice ~ logcarat + clarity + cut + color + depth +
##     table, data = diamonds_train_set)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.54396 -0.08482 -0.00076  0.08121  0.94377
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   8.0324414  0.0809237  99.259   <2e-16 ***
## logcarat      1.8887198  0.0020522 920.335   <2e-16 ***
## clarityIF     1.1138717  0.0113165  98.429   <2e-16 ***
## claritySI1    0.5845861  0.0098134  59.570   <2e-16 ***
## claritySI2    0.4217002  0.0098622  42.759   <2e-16 ***
## clarityVS1    0.8069900  0.0099942  80.745   <2e-16 ***
## clarityVS2    0.7341975  0.0098569  74.485   <2e-16 ***
## clarityVVS1   1.0171241  0.0105192  96.692   <2e-16 ***
## clarityVVS2   0.9383014  0.0102766  91.304   <2e-16 ***
## cutGood       0.0772431  0.0073181  10.555   <2e-16 ***
## cutIdeal      0.1514710  0.0072856  20.790   <2e-16 ***
## cutPremium    0.1288181  0.0070241  18.339   <2e-16 ***
## cutVery Good  0.1105916  0.0070255  15.742   <2e-16 ***
## colorE       -0.0535791  0.0037965 -14.113   <2e-16 ***
## colorF       -0.0951227  0.0038631 -24.623   <2e-16 ***
## colorG       -0.1577962  0.0037723 -41.830   <2e-16 ***
## colorH       -0.2484551  0.0040083 -61.985   <2e-16 ***
## colorI       -0.3724699  0.0044981 -82.806   <2e-16 ***
## colorJ       -0.5042370  0.0055508 -90.840   <2e-16 ***
## depth        -0.0016297  0.0008842  -1.843   0.0653 .
## table        -0.0010099  0.0006342  -1.592   0.1113
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.1313 on 16120 degrees of freedom
## Multiple R-squared:  0.9831, Adjusted R-squared:  0.9831
## F-statistic: 4.696e+04 on 20 and 16120 DF,  p-value: < 2.2e-16
```

These summaries demonstrate that log_model_1 has an RSq that is superior to the all_vars model, and also explains price increases in the most conservative and straightforward way. As complexity is added to the log model (log_model_2 -> log_model_4), RSq marginally increases and the residual SE marginally decreases.

Statistical Assumptions Tests:

List of Statistical Assumptions:

The *five* CLM Assumptions are:
1. IID (i.e. Random) Sampling
2. No Perfect Collinearity
3. Linear Conditional Expectation
4. Homoskedastic Conditional Variance (Constant Conditional Variance)
5. Normally Distributed Errors

1. IID

Consistent regression estimates require an assumption of independent and identically distributed (iid) observations. We conducted an exploratory study using a dataset from a single online retailer. The insights may not necessarily generalize to other contexts. For example, it is likely that the consumer experience of shopping for diamonds offline would likely be very different. Anecdotal evidence suggests, offline diamond business is relatively opaque. Our study does not yield any information on the consumer motivation for purchasing diamonds. For example, consumers could be purchasing diamonds as an investment and such purchases may be affected by a different set of considerations beyond the diamond attributes that we considered in our study. Further research would be required to assess the generalization of our observations. Moreover, an evenly distributed data across all features would likely create a better model. Our dataset primarily contains Clarity Groups SI1, SI2, VS1 and VS2, whereas I1 and IF only make up 1.4% and 3.3% of the data, respectively. However, this population perhaps resembles what can be realistically found on the market and hence, we decided to to stick with our data and work with this limitation. With these limitations in mind, we argue that the dataset is not ideally IID, yet we are choosing to utilize this data due to 1) the possibility that the data resembles the conditions observed in the diamonds market, and 2) that models made from this data can likewise be useful in said market.

2. No Perfect Collinearity:

```
library(car)
vif(two_c_model)
```

```
##             GVIF Df GVIF^(1/(2*Df))
## carat   1.157423  1        1.075836
## clarity 1.157423  7        1.010497
```

```
vif(three_c_model)
```

```
##             GVIF Df GVIF^(1/(2*Df))
## carat   1.171625  1        1.082416
## clarity 1.231270  7        1.014971
## cut     1.098678  4        1.011833
```

```
vif(four_c_model)
```

```
##             GVIF Df GVIF^(1/(2*Df))
## carat   1.311970  1        1.145412
## clarity 1.300463  7        1.018943
## cut     1.102918  4        1.012320
## color   1.174825  6        1.013517
```

```
vif(all_var_model)
```

```
##             GVIF Df GVIF^(1/(2*Df))
## carat   1.321920  1        1.149748
## clarity 1.309846  7        1.019466
## cut     1.957131  4        1.087558
## color   1.180442  6        1.013920
## depth   1.396282  1        1.181644
## table   1.824659  1        1.350799
```

```
vif(log_model_2)
```

```
##              GVIF Df GVIF^(1/(2*Df))
## logcarat 1.187667  1        1.089801
## clarity  1.187667  7        1.012361
```

```
vif(log_model_3)
```

```
##              GVIF Df GVIF^(1/(2*Df))
## logcarat 1.318047  1        1.148062
## clarity  1.331086  7        1.020638
## cut      1.103099  4        1.012341
## color    1.150248  6        1.011733
```

```
vif(log_model_4)
```

```
##              GVIF Df GVIF^(1/(2*Df))
## logcarat 1.331499  1        1.153906
## clarity  1.340183  7        1.021135
## cut      1.955121  4        1.087418
## color    1.156180  6        1.012167
## depth    1.396193  1        1.181606
## table    1.828985  1        1.352400
```

All Models Pass The Colinearity Assumption
* two_c_model demonstrates minimal colinearity, with both values nearing 1.
* three_c_model also demonstrates minimal colinearity with values nearing 1.
* four_c_model has somewhat larger variance inflation factors for carat and clarity, but still, they are close to 1.
* all_vars_model has minor variance inflation factors for cut and table. It appears as if the cut and the table may be somewhat associated. This observation makes sense given how you *cut* the diamond will change the *table* of that diamond. However, they still pass the VIF test.
* log_model_1 automatically passes since it only contains 1 input parameter. * log_model_2 and log_model_3 do not demonstrate significant inflation, although log_model_4 demonstrates minor inflation of the cut and table variables for reasons we suspect in the all_vars_model.

Refer to the following graphs for diagnostic plots and modifiers: https://stat.ethz.ch/R-manual/R-devel/library/stats/html/plot.lm.html

3. Linear Conditional Expectation: Residuals vs. Fitted

Prepare non-log plots:

```
LCE1gg<-ggplot(two_c_model, aes(.fitted, .resid)) +
    geom_point() +
    stat_smooth(method="loess") +
    geom_hline(yintercept=0, col="red", linetype="dashed") +
    xlab("Fitted Values") +
    ylab("Residuals") +
    ggtitle("two_c_model") +
    theme(plot.title = element_text(hjust = 0.5))
LCE2gg<-ggplot(three_c_model, aes(.fitted, .resid)) +
    geom_point() +
    stat_smooth(method="loess") +
    geom_hline(yintercept=0, col="red", linetype="dashed") +
    xlab("Fitted Values") +
    ylab("Residuals") +
    ggtitle("three_c_model") +
    theme(plot.title = element_text(hjust = 0.5))
LCE3gg<-ggplot(four_c_model, aes(.fitted, .resid)) +
    geom_point() +
    stat_smooth(method="loess") +
    geom_hline(yintercept=0, col="red", linetype="dashed") +
    xlab("Fitted Values") +
    ylab("Residuals") +
    ggtitle("four_c_model") +
    theme(plot.title = element_text(hjust = 0.5))
LCE4gg<-ggplot(all_var_model, aes(.fitted, .resid)) +
    geom_point() +
    stat_smooth(method="loess") +
    geom_hline(yintercept=0, col="red", linetype="dashed") +
    xlab("Fitted Values") +
    ylab("Residuals") +
    ggtitle("all_var_model") +
    theme(plot.title = element_text(hjust = 0.5))
```

Prepare log plots:

```
LCE5gg<-ggplot(log_model_1, aes(.fitted, .resid)) +
    geom_point() +
    stat_smooth(method="loess") +
    geom_hline(yintercept=0, col="red", linetype="dashed") +
    xlab("Fitted Values") +
    ylab("Residuals") +
    ggtitle("log_model_1") +
    theme(plot.title = element_text(hjust = 0.5))
LCE6gg<-ggplot(log_model_2, aes(.fitted, .resid)) +
    geom_point() +
    stat_smooth(method="loess") +
    geom_hline(yintercept=0, col="red", linetype="dashed") +
```

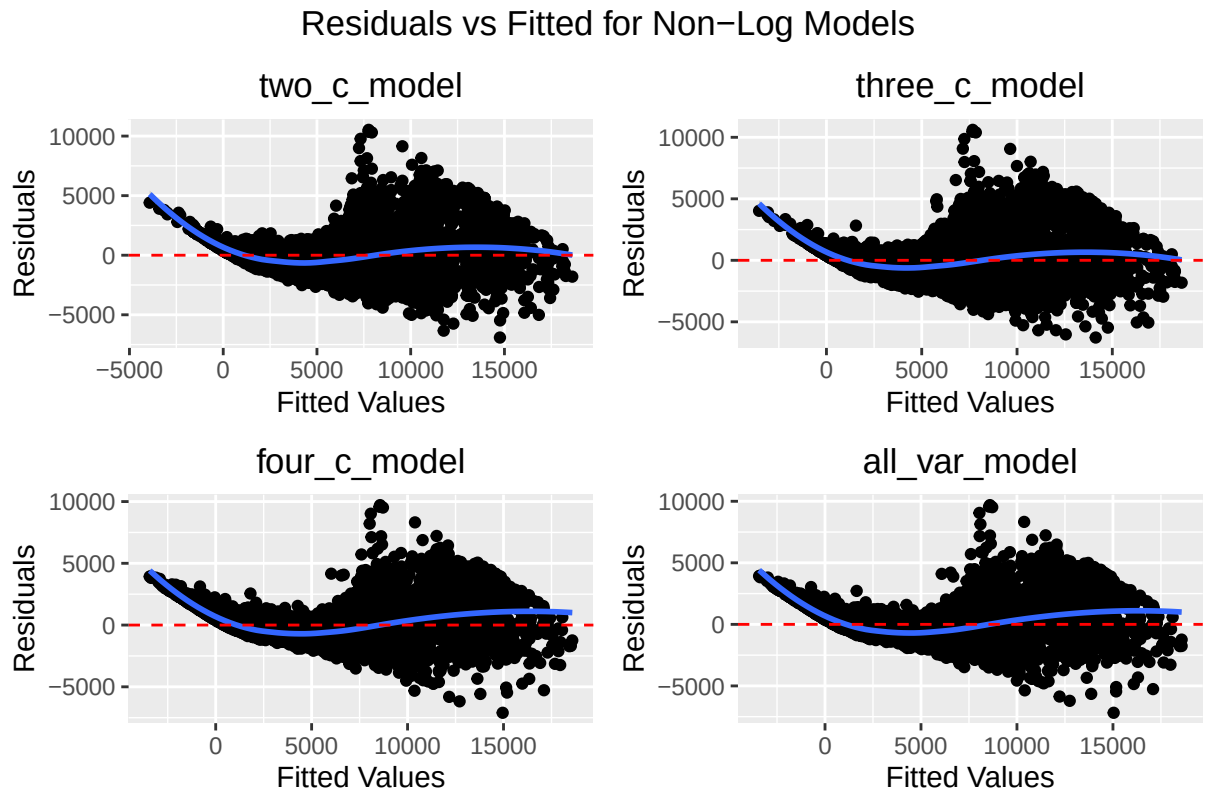
```
    xlab("Fitted Values") +
    ylab("Residuals") +
    ggtitle("log_model_2") +
    theme(plot.title = element_text(hjust = 0.5))
LCE7gg<-ggplot(log_model_3, aes(.fitted, .resid)) +
    geom_point() +
    stat_smooth(method="loess") +
    geom_hline(yintercept=0, col="red", linetype="dashed") +
    xlab("Fitted Values") +
    ylab("Residuals") +
    ggtitle("log_model_3") +
    theme(plot.title = element_text(hjust = 0.5))
LCE8gg<-ggplot(log_model_4, aes(.fitted, .resid)) +
    geom_point() +
    stat_smooth(method="loess") +
    geom_hline(yintercept=0, col="red", linetype="dashed") +
    xlab("Fitted Values") +
    ylab("Residuals") +
    ggtitle("log_model_4") +
    theme(plot.title = element_text(hjust = 0.5))
```

Prepare Subplots:

```
LCEgg_non_log <- (LCE1gg + LCE2gg)/(LCE3gg + LCE4gg) + plot_annotation(title = "Residuals vs Fitted for
                                                    caption = "Assumption 3: Linear Conditional Expectation",
                                                    theme = theme(plot.title = element_text(hjust = 0.5)))
LCEgg_log <- (LCE5gg + LCE6gg)/(LCE7gg + LCE8gg) + plot_annotation(title = "Residuals vs Fitted for Log
                                                    caption = "Assumption 3: Linear Conditional Expectation",
                                                    theme = theme(plot.title = element_text(hjust = 0.5)))
```

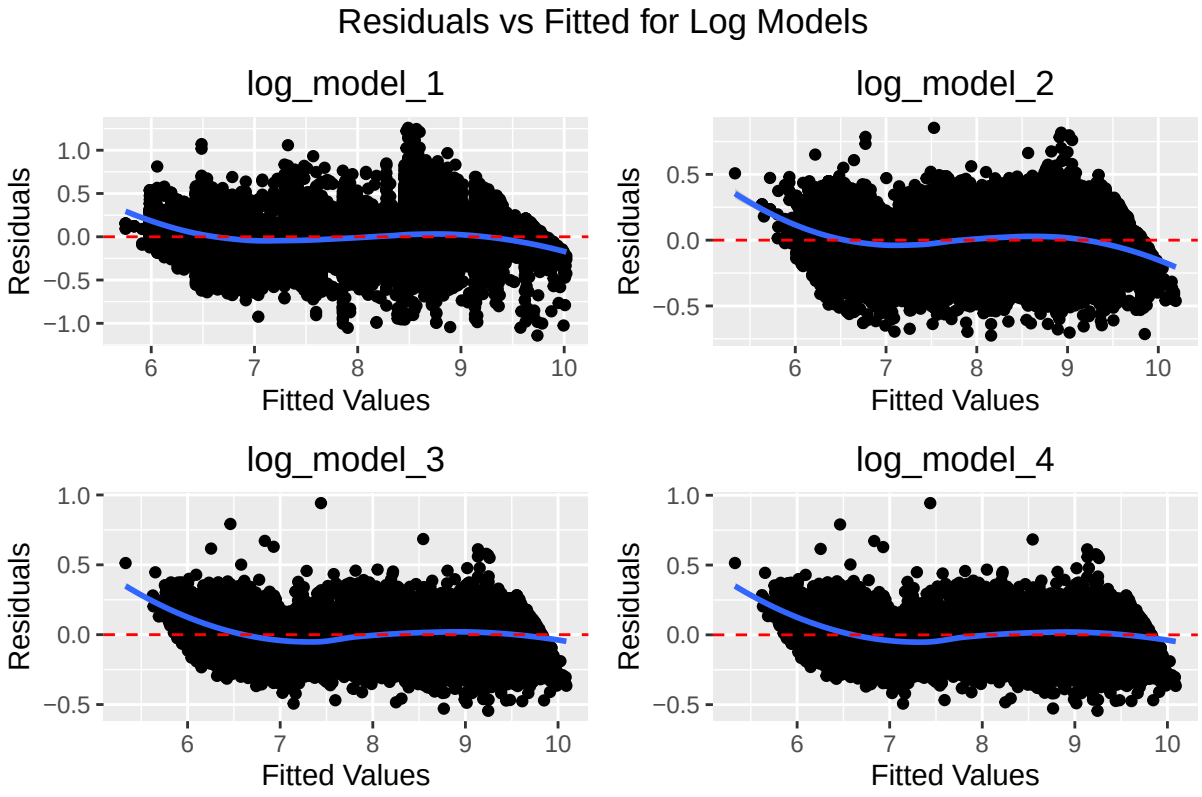
Graph Subplots:

```
LCEgg_non_log
```



Assumption 3: Linear Conditional Expectation

```
LCEgg_log
```



Assumption 3: Linear Conditional Expectation

- It appears as if two_c_model, three_c_model, four_c_model and all_vars_model all possess linear conditional expectation curves that are not constant and centered around residuals = 0. *These models all fail to pass the Linear Conditional Expectation assumption.*

- In contrast, log_model_1 demonstrates that the curve representing expectation value of the residuals throughout the entire span of the linear model is essentially constant and near 0. *log_model passes the Linear Conditional Expectation assumption.*

- Finally, log_model_2, log_model_3 and log_model_4 have Residuals vs Fitted plots that mirror log_model_1, yet all include some minor deviation from the residual = 0 line. It should be noted that these deviations are minimal relative to the entire plot. Hence, we believe that *log_model_2 -> log_model_4 pass the Linear Conditional Expectation Assumption.*

4. Homoskedastic Conditional Variance (Constant Conditional Variance): Scale Location

Prepare non-log plots:

```
HCV1 <- ggplot(two_c_model, aes(.fitted, sqrt(abs(.stdresid)))) +
    geom_point(na.rm=TRUE,
               size = 1,
               shape = 1) +
    stat_smooth(method="loess", na.rm = TRUE) +
    xlab("Fitted Values") +
    ylab(expression(sqrt("|Standardized residuals|"))) +
    ggtitle("two_c_model") +
    theme(plot.title = element_text(hjust = 0.5))
HCV2 <- ggplot(three_c_model, aes(.fitted, sqrt(abs(.stdresid)))) +
    geom_point(na.rm=TRUE,
               size = 1,
               shape = 1) +
    stat_smooth(method="loess", na.rm = TRUE) +
    xlab("Fitted Values") +
    ylab(expression(sqrt("|Standardized residuals|"))) +
    ggtitle("three_c_model") +
    theme(plot.title = element_text(hjust = 0.5))
HCV3 <- ggplot(four_c_model, aes(.fitted, sqrt(abs(.stdresid)))) +
    geom_point(na.rm=TRUE,
               size = 1,
               shape = 1) +
    stat_smooth(method="loess", na.rm = TRUE) +
    xlab("Fitted Values") +
    ylab(expression(sqrt("|Standardized residuals|"))) +
    ggtitle("four_c_model") +
    theme(plot.title = element_text(hjust = 0.5))
HCV4 <- ggplot(all_var_model, aes(.fitted, sqrt(abs(.stdresid)))) +
    geom_point(na.rm=TRUE,
               size = 1,
               shape = 1) +
    stat_smooth(method="loess", na.rm = TRUE) +
    xlab("Fitted Values") +
    ylab(expression(sqrt("|Standardized residuals|"))) +
    ggtitle("all_var_model") +
    theme(plot.title = element_text(hjust = 0.5))
```

Prepare log plots:

```
HCV5 <- ggplot(log_model_1, aes(.fitted, sqrt(abs(.stdresid)))) +
    geom_point(na.rm=TRUE,
               size = 1,
               shape = 1) +
    stat_smooth(method="loess", na.rm = TRUE) +
    xlab("Fitted Values") +
    ylab(expression(sqrt("|Standardized residuals|"))) +
```

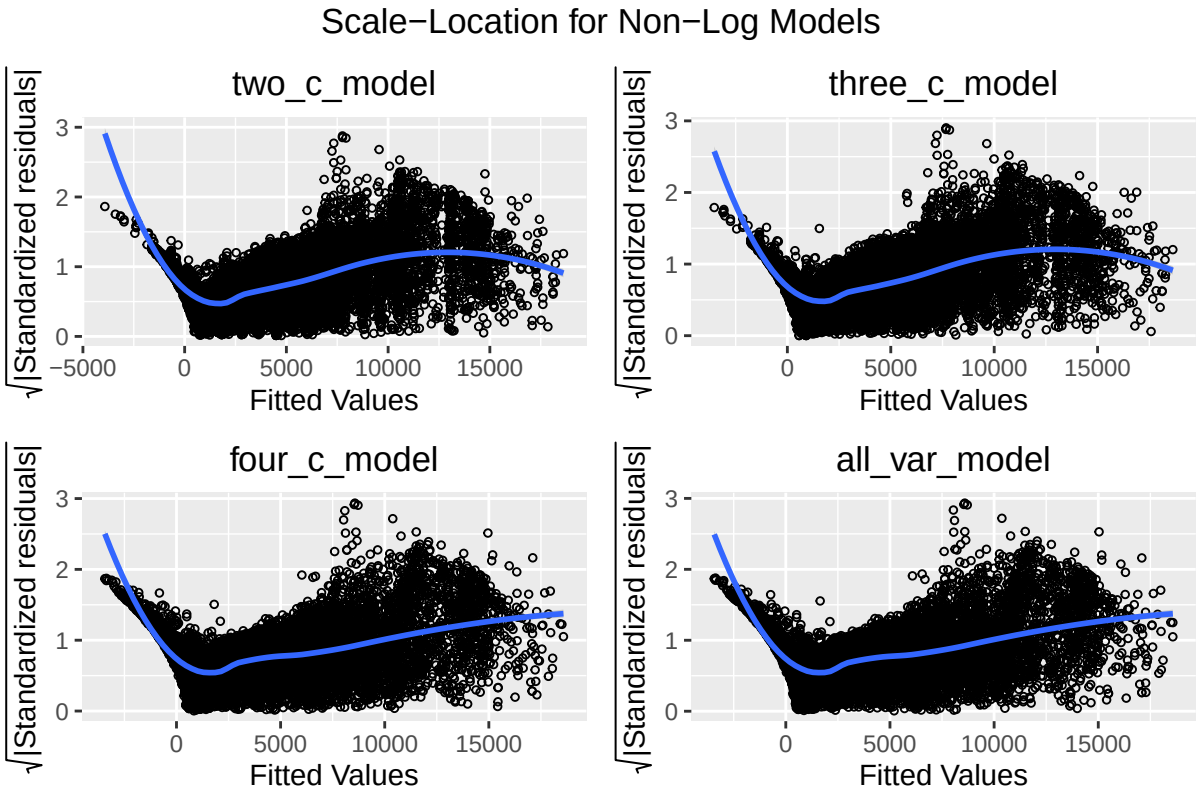
```
    ggtitle("log_model_1") +
    theme(plot.title = element_text(hjust = 0.5))
HCV6 <- ggplot(log_model_2, aes(.fitted, sqrt(abs(.stdresid)))) +
    geom_point(na.rm=TRUE,
               size = 1,
               shape = 1) +
    stat_smooth(method="loess", na.rm = TRUE) +
    xlab("Fitted Values") +
    ylab(expression(sqrt("|Standardized residuals|"))) +
    ggtitle("log_model_2") +
    theme(plot.title = element_text(hjust = 0.5))
HCV7 <- ggplot(log_model_3, aes(.fitted, sqrt(abs(.stdresid)))) +
    geom_point(na.rm=TRUE,
               size = 1,
               shape = 1) +
    stat_smooth(method="loess", na.rm = TRUE) +
    xlab("Fitted Values") +
    ylab(expression(sqrt("|Standardized residuals|"))) +
    ggtitle("log_model_3") +
    theme(plot.title = element_text(hjust = 0.5))
HCV8 <- ggplot(log_model_4, aes(.fitted, sqrt(abs(.stdresid)))) +
    geom_point(na.rm=TRUE,
               size = 1,
               shape = 1) +
    stat_smooth(method="loess", na.rm = TRUE) +
    xlab("Fitted Values") +
    ylab(expression(sqrt("|Standardized residuals|"))) +
    ggtitle("log_model_4") +
    theme(plot.title = element_text(hjust = 0.5))
```

Prepare Subplots:

```
HCVgg_non_log <- (HCV1 + HCV2)/(HCV3 + HCV4) + plot_annotation(title = "Scale-Location for Non-Log Model
                                                caption = "Assumption 4: Homoskedastic Conditional Varian
                                                theme = theme(plot.title = element_text(hjust = 0.5)))
HCVgg_log <- (HCV5 + HCV6)/(HCV7 + HCV8) + plot_annotation(title = "Scale-Location for Non-Log Models",
                                                caption = "Assumption 4: Homoskedastic Conditional Varian
                                                theme = theme(plot.title = element_text(hjust = 0.5)))
```

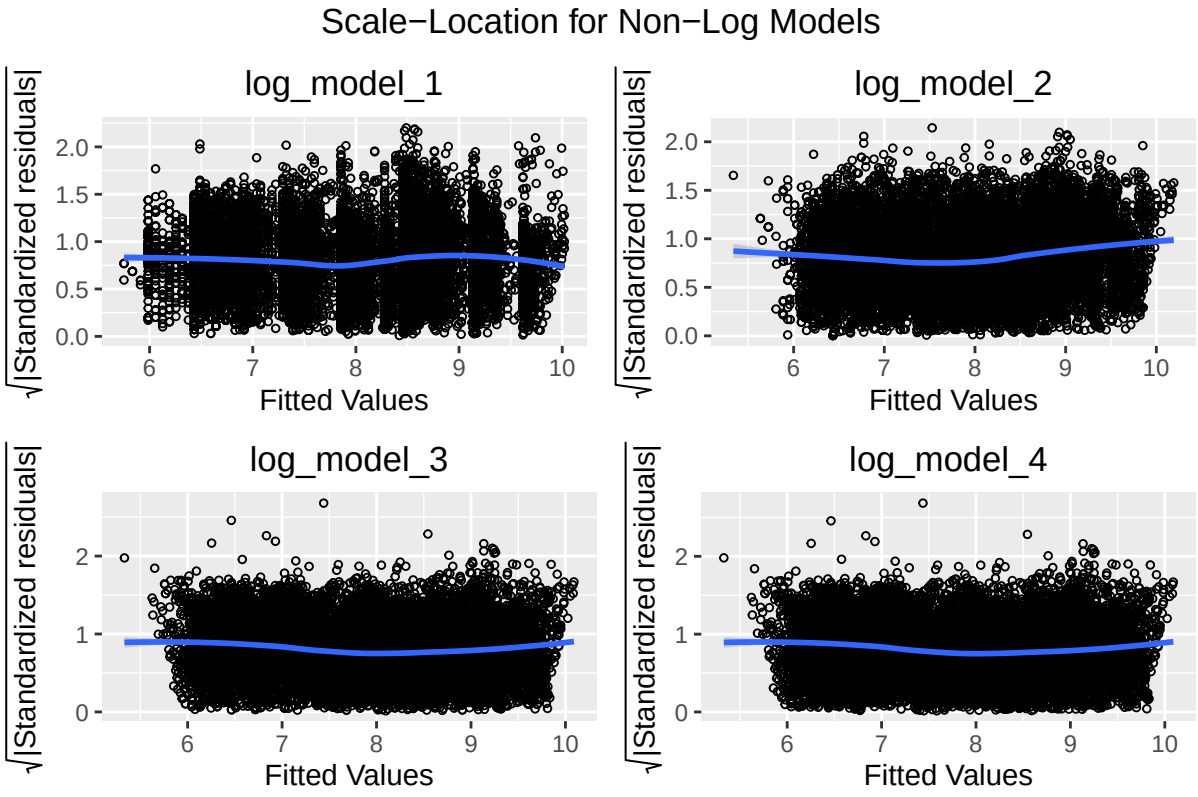
Graph Subplots:

```
HCVgg_non_log
```



Assumption 4: Homoskedastic Conditional Variance

```
HCVgg_log
```



Assumption 4: Homoskedastic Conditional Variance

- It appears that two_c_model, three_c_model, four_c_model and all_vars_model all possess variations in standard error values across their respective spans of fitted values. In other words, these models are possess heteroskedastic conditional variance *Hence, none of the non_log_models pass the Homoskedastic Conditional Variance assumption.*

- In contrast, all of the log_models demonstrate that the square root of the magnitude of standardized residuals is constant throughout the entire span of fitted values. It should be noted that log_model_2 -> log_model_4 demonstrate higher deviance of some trace values, but compared to the entire population this deviation is minimal. *All of the log_models pass the Homoskedastic Conditional Variance assumption.*

5. Normally Distributed Errors

a. QQ Plots:

Prepare non-log plots:

```
QQ1 <- ggplot(data = two_c_model,
              aes(x = qqnorm(rstandard(two_c_model), plot = FALSE)[[1]],
                  y = qqnorm(rstandard(two_c_model), plot = FALSE)[[2]])
              ) +
    geom_point(na.rm = TRUE) +
    stat_qq_line(mapping = aes(sample = qqnorm(rstandard(two_c_model), plot = FALSE)[[2]]),
                 col="red",
                 linetype="dashed") +
    xlab("Theoretical Quantiles") +
    ylab("Standardized Residuals") +
    ggtitle("two_c_model") +
    theme(plot.title = element_text(hjust = 0.5))

QQ2 <- ggplot(data = three_c_model,
              aes(x = qqnorm(rstandard(three_c_model), plot = FALSE)[[1]],
                  y = qqnorm(rstandard(three_c_model), plot = FALSE)[[2]])
              ) +
    geom_point(na.rm = TRUE) +
    stat_qq_line(mapping = aes(sample = qqnorm(rstandard(three_c_model), plot = FALSE)[[2]]),
                 col="red",
                 linetype="dashed") +
    xlab("Theoretical Quantiles") +
    ylab("Standardized Residuals") +
    ggtitle("three_c_model") +
    theme(plot.title = element_text(hjust = 0.5))

QQ3 <- ggplot(data = four_c_model,
              aes(x = qqnorm(rstandard(four_c_model), plot = FALSE)[[1]],
                  y = qqnorm(rstandard(four_c_model), plot = FALSE)[[2]])
              ) +
    geom_point(na.rm = TRUE) +
    stat_qq_line(mapping = aes(sample = qqnorm(rstandard(four_c_model), plot = FALSE)[[2]]),
                 col="red",
                 linetype="dashed") +
    xlab("Theoretical Quantiles") +
    ylab("Standardized Residuals") +
    ggtitle("four_c_model") +
    theme(plot.title = element_text(hjust = 0.5))

QQ4 <- ggplot(data = all_var_model,
              aes(x = qqnorm(rstandard(all_var_model), plot = FALSE)[[1]],
                  y = qqnorm(rstandard(all_var_model), plot = FALSE)[[2]])
              ) +
    geom_point(na.rm = TRUE) +
    stat_qq_line(mapping = aes(sample = qqnorm(rstandard(all_var_model), plot = FALSE)[[2]]),
                 col="red",
                 linetype="dashed") +
```

```
    xlab("Theoretical Quantiles") +
    ylab("Standardized Residuals") +
    ggtitle("all_var_model") +
    theme(plot.title = element_text(hjust = 0.5))
```

```
QQ5 <- ggplot(data = log_model_1,
              aes(x = qqnorm(rstandard(log_model_1), plot = FALSE)[[1]],
                  y = qqnorm(rstandard(log_model_1), plot = FALSE)[[2]])
              ) +
    geom_point(na.rm = TRUE) +
    stat_qq_line(mapping = aes(sample = qqnorm(rstandard(log_model_1), plot = FALSE)[[2]]),
                 col="red",
                 linetype="dashed") +
    xlab("Theoretical Quantiles") +
    ylab("Standardized Residuals") +
    ggtitle("log_model_1") +
    theme(plot.title = element_text(hjust = 0.5))

QQ6 <- ggplot(data = log_model_2,
              aes(x = qqnorm(rstandard(log_model_2), plot = FALSE)[[1]],
                  y = qqnorm(rstandard(log_model_2), plot = FALSE)[[2]])
              ) +
    geom_point(na.rm = TRUE) +
    stat_qq_line(mapping = aes(sample = qqnorm(rstandard(log_model_2), plot = FALSE)[[2]]),
                 col="red",
                 linetype="dashed") +
    xlab("Theoretical Quantiles") +
    ylab("Standardized Residuals") +
    ggtitle("log_model_2") +
    theme(plot.title = element_text(hjust = 0.5))

QQ7 <- ggplot(data = log_model_3,
              aes(x = qqnorm(rstandard(log_model_3), plot = FALSE)[[1]],
                  y = qqnorm(rstandard(log_model_3), plot = FALSE)[[2]])
              ) +
    geom_point(na.rm = TRUE) +
    stat_qq_line(mapping = aes(sample = qqnorm(rstandard(log_model_3), plot = FALSE)[[2]]),
                 col="red",
                 linetype="dashed") +
    xlab("Theoretical Quantiles") +
    ylab("Standardized Residuals") +
    ggtitle("log_model_3") +
    theme(plot.title = element_text(hjust = 0.5))

QQ8 <- ggplot(data = log_model_4,
              aes(x = qqnorm(rstandard(log_model_4), plot = FALSE)[[1]],
                  y = qqnorm(rstandard(log_model_4), plot = FALSE)[[2]])
              ) +
    geom_point(na.rm = TRUE) +
    stat_qq_line(mapping = aes(sample = qqnorm(rstandard(log_model_4), plot = FALSE)[[2]]),
                 col="red",
                 linetype="dashed") +
    xlab("Theoretical Quantiles") +
```

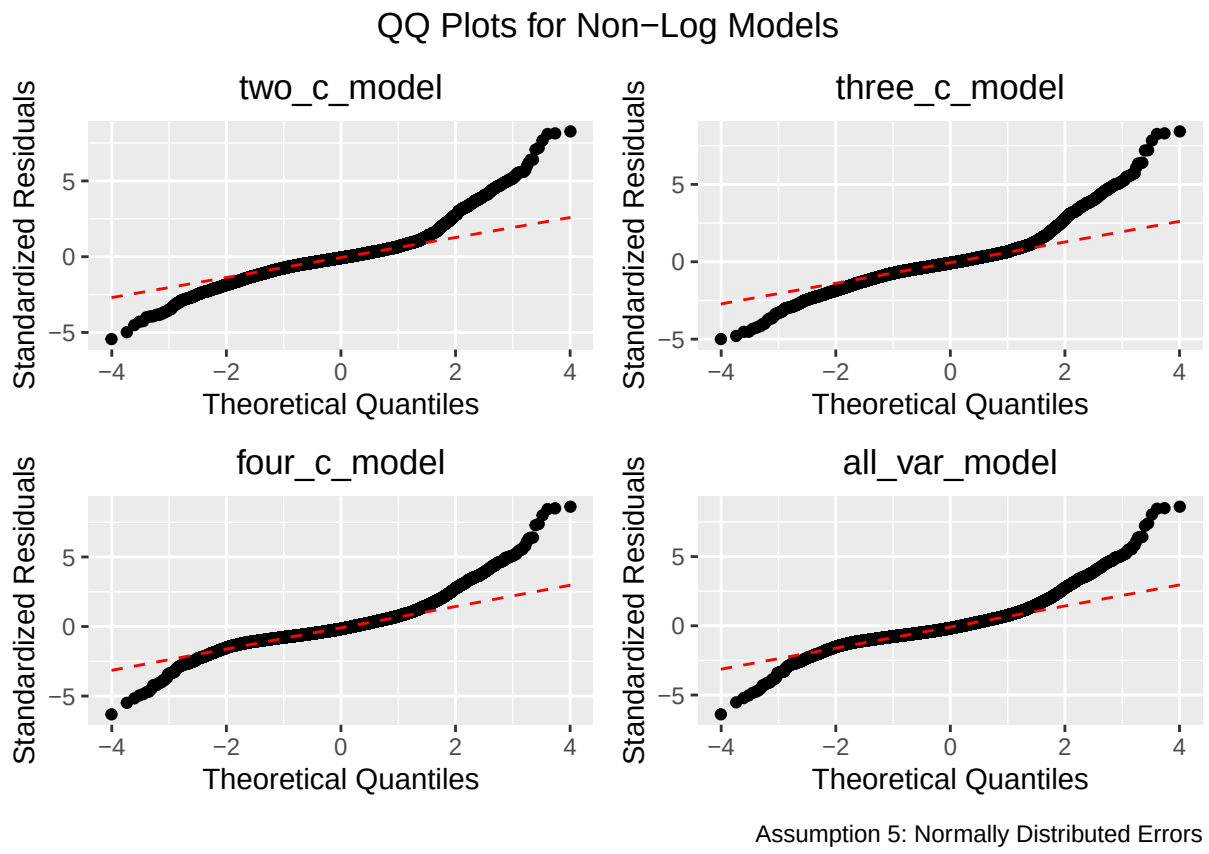
```
    ylab("Standardized Residuals") +
    ggtitle("log_model_4") +
    theme(plot.title = element_text(hjust = 0.5))
```

Prepare QQ Subplots:

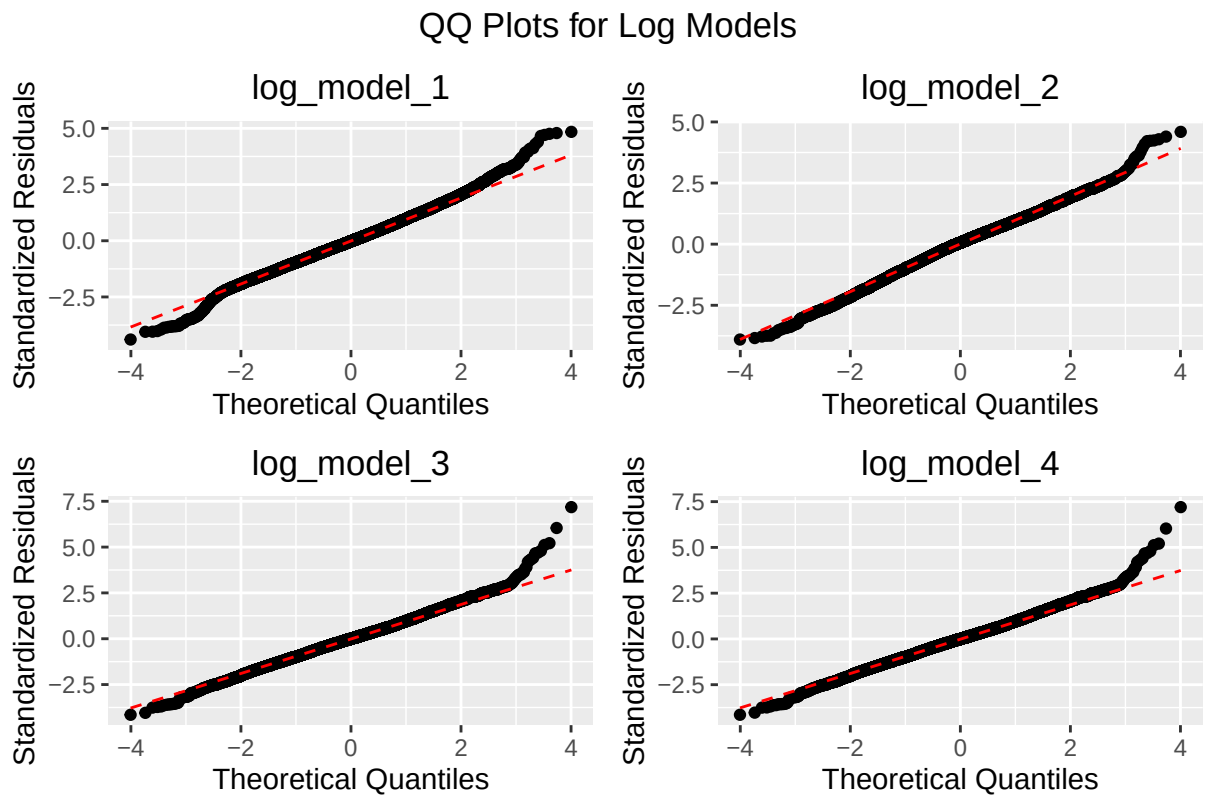
```
QQ_non_log <- (QQ1 + QQ2)/(QQ3 + QQ4) + plot_annotation(title = "QQ Plots for Non-Log Models",
                                                        caption = "Assumption 5: Normally Distributed Errors",
                                                        theme = theme(plot.title = element_text(hjust = 0.5)))
QQ_log <- (QQ5 + QQ6)/(QQ7 + QQ8) + plot_annotation(title = "QQ Plots for Log Models",
                                                    caption = "Assumption 5: Normally Distributed Errors",
                                                    theme = theme(plot.title = element_text(hjust = 0.5)))
```

Graph QQ Subplots:

```
QQ_non_log
```




```
QQ_log
```



Assumption 5: Normally Distributed Errors

b. Histgram of Residuals:

Prepare non-log plots:

```
HR1 <- ggplot(data = two_c_model, mapping = aes(x = .resid)) +
  geom_histogram(stat = "bin",
                           color = "black",
                           fill = "blue",
                           alpha = 0.8) +
  xlab("Residual Values") +
  ylab("Count") +
  ggtitle("two_c_model") +
  theme(plot.title = element_text(hjust = 0.5))
HR2 <- ggplot(data = three_c_model, mapping = aes(x = .resid)) +
  geom_histogram(stat = "bin",
                           color = "black",
                           fill = "blue",
                           alpha = 0.8) +
  xlab("Residual Values") +
  ylab("Count") +
  ggtitle("three_c_model") +
  theme(plot.title = element_text(hjust = 0.5))
HR3 <- ggplot(data = four_c_model, mapping = aes(x = .resid)) +
  geom_histogram(stat = "bin",
```

```
                          color = "black",
                          fill = "blue",
                          alpha = 0.8) +
  xlab("Residual Values") +
  ylab("Count") +
  ggtitle("four_c_model") +
  theme(plot.title = element_text(hjust = 0.5))
HR4 <- ggplot(data = all_var_model, mapping = aes(x = .resid)) +
  geom_histogram(stat = "bin",
                          color = "black",
                          fill = "blue",
                          alpha = 0.8) +
  xlab("Residual Values") +
  ylab("Count") +
  ggtitle("all_var_model") +
  theme(plot.title = element_text(hjust = 0.5))
```

Prepare log plots:

```
HR5 <- ggplot(data = log_model_1, mapping = aes(x = .resid)) +
  geom_histogram(stat = "bin",
                          color = "black",
                          fill = "blue",
                          alpha = 0.8) +
  xlab("Residual Values") +
  ylab("Count") +
  ggtitle("log_model_1") +
  theme(plot.title = element_text(hjust = 0.5))
HR6 <- ggplot(data = log_model_2, mapping = aes(x = .resid)) +
  geom_histogram(stat = "bin",
                          color = "black",
                          fill = "blue",
                          alpha = 0.8) +
  xlab("Residual Values") +
  ylab("Count") +
  ggtitle("log_model_2") +
  theme(plot.title = element_text(hjust = 0.5))
HR7 <- ggplot(data = log_model_3, mapping = aes(x = .resid)) +
  geom_histogram(stat = "bin",
                          color = "black",
                          fill = "blue",
                          alpha = 0.8) +
  xlab("Residual Values") +
  ylab("Count") +
  ggtitle("log_model_3") +
  theme(plot.title = element_text(hjust = 0.5))
HR8 <- ggplot(data = log_model_4, mapping = aes(x = .resid)) +
  geom_histogram(stat = "bin",
                          color = "black",
                          fill = "blue",
                          alpha = 0.8) +
  xlab("Residual Values") +
```

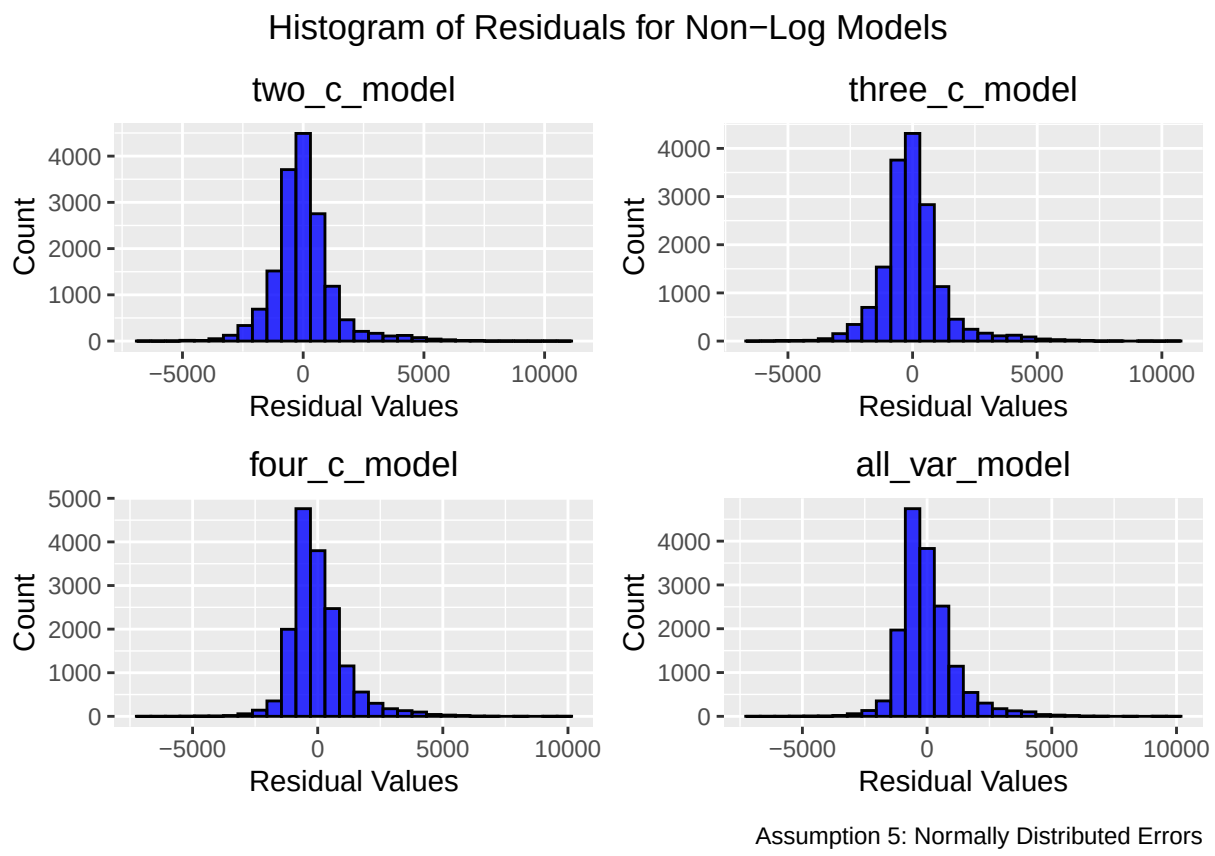
```
  ylab("Count") +
  ggtitle("log_model_4") +
  theme(plot.title = element_text(hjust = 0.5))
```

Prepare Subplots:

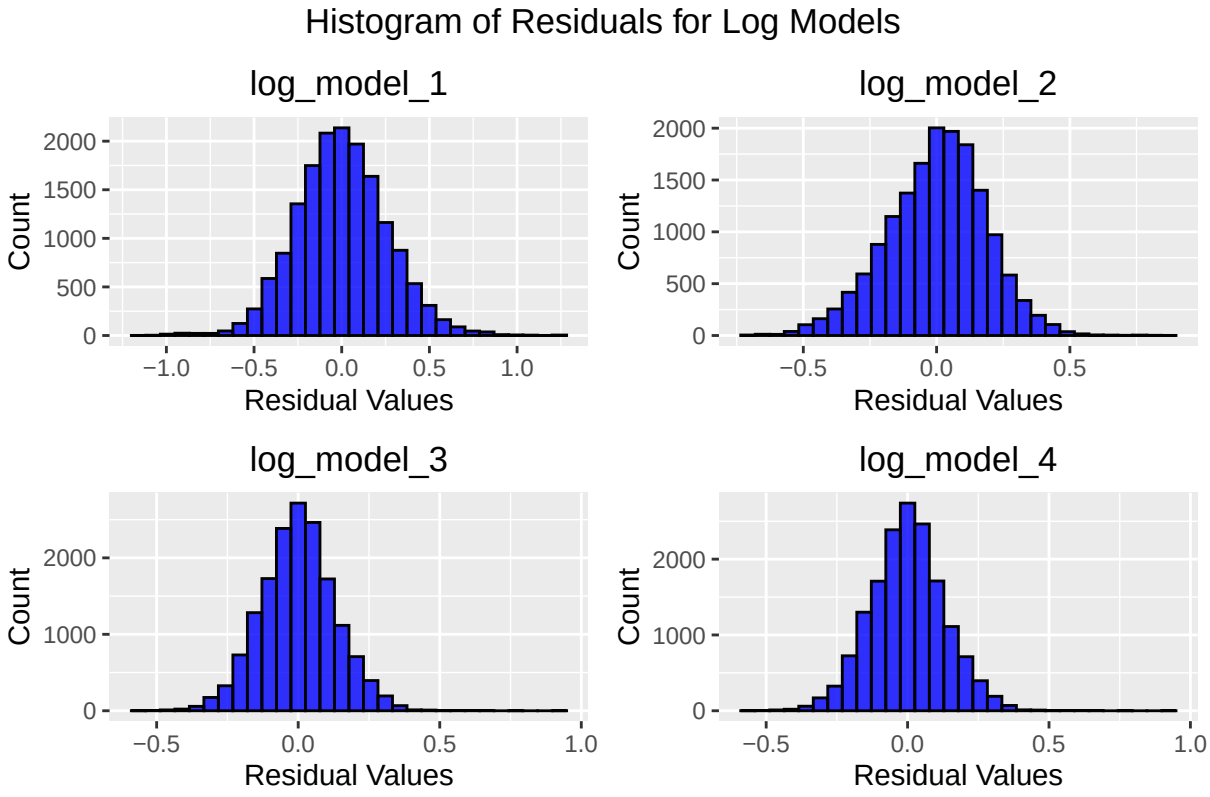
```
HR_non_log <- (HR1 + HR2)/(HR3 + HR4) + plot_annotation(title = "Histogram of Residuals for Non-Log Mode
                                                        caption = "Assumption 5: Normally Distributed Errors",
                                                        theme = theme(plot.title = element_text(hjust = 0.5)))
HR_log <- (HR5 + HR6)/(HR7 + HR8) + plot_annotation(title = "Histogram of Residuals for Log Models",
                                                        caption = "Assumption 5: Normally Distributed Errors",
                                                        theme = theme(plot.title = element_text(hjust = 0.5)))
```

Graph Residual Histogram Plots:

```
HR_non_log
```




```
HR_log
```



Assumption 5: Normally Distributed Errors

- two_c_model, three_c_model, four_c_model and all_vars_model possess distributions of error terms that do not reflect a normal distribution outside of theoretical quantiles (-1,1). Likewise, histograms of the residuals for each model are leptokurtic *These models all fail to pass the Normally Distributed Errors assumption.*

- all of the log_models possess residual distributions that resemble a normal parametric class within theoretical quantiles (-2, 2). The histograms of each of the log_models residuals are visually mesokurtic, although log_model_3 and log_model_4 have a miniscule positive skew. *All of the log_models pass the Normally Distributed Errors assumption.*

Testing Trained Models Against Test Data

Prediction:

```
diamonds_test_set$predict_1 <- predict(two_c_model, diamonds_test_set)
diamonds_test_set$predict_2 <- predict(three_c_model, diamonds_test_set)
diamonds_test_set$predict_3 <- predict(four_c_model, diamonds_test_set)
diamonds_test_set$predict_4 <- predict(all_var_model, diamonds_test_set)
diamonds_test_set$predict_5 <- predict(log_model_1, diamonds_test_set)
diamonds_test_set$predict_6 <- predict(log_model_2, diamonds_test_set)
diamonds_test_set$predict_7 <- predict(log_model_3, diamonds_test_set)
diamonds_test_set$predict_8 <- predict(log_model_4, diamonds_test_set)
```

Head Tables Comparing Predictions vs. Actuals:

```
head(diamonds_test_set[,c("predict_1", "price")])
```

```
## # A tibble: 6 x 2
##   predict_1 price
##       <dbl> <dbl>
## 1    -2187.   326
## 2    -1489.   326
## 3     -400.   327
## 4     -150.   334
## 5    -1502.   335
## 6      217.   336
```

```
head(diamonds_test_set[,c("predict_2", "price")])
```

```
## # A tibble: 6 x 2
##   predict_2 price
##       <dbl> <dbl>
## 1    -2032.   326
## 2    -1511.   326
## 3     -633.   327
## 4     -196.   334
## 5    -1684.   335
## 6      140.   336
```

```
head(diamonds_test_set[,c("predict_3", "price")])
```

```
## # A tibble: 6 x 2
##   predict_3 price
##       <dbl> <dbl>
## 1    -1991.   326
## 2    -1364.   326
## 3     -378.   327
## 4    -1273.   334
## 5    -3755.   335
## 6    -1861.   336
```

```
head(diamonds_test_set[,c("predict_4", "price")])
```

```
## # A tibble: 6 x 2
##   predict_4 price
##       <dbl> <dbl>
## 1    -1957.   326
## 2    -1409.   326
## 3     -466.   327
## 4    -1281.   334
## 5    -3756.   335
## 6    -1858.   336
```

```
head(diamonds_test_set[,c("predict_5", "logprice")])
```

```
## # A tibble: 6 x 2
##   predict_5 logprice
##       <dbl>    <dbl>
## 1      5.99     5.79
## 2      5.83     5.79
## 3      5.99     5.79
## 4      6.38     5.81
## 5      6.49     5.81
## 6      6.06     5.82
```

```
head(diamonds_test_set[,c("predict_6", "logprice")])
```

```
## # A tibble: 6 x 2
##   predict_6 logprice
##       <dbl>    <dbl>
## 1      5.59     5.79
## 2      5.57     5.79
## 3      5.93     5.79
## 4      6.31     5.81
## 5      6.13     5.81
## 6      6.16     5.82
```

```
head(diamonds_test_set[,c("predict_7", "logprice")])
```

```
## # A tibble: 6 x 2
##   predict_7 logprice
##       <dbl>    <dbl>
## 1      5.62     5.79
## 2      5.59     5.79
## 3      5.93     5.79
## 4      6.03     5.81
## 5      5.65     5.81
## 6      5.72     5.82
```

```
head(diamonds_test_set[,c("predict_8", "logprice")])
```

```
## # A tibble: 6 x 2
##   predict_8 logprice
##       <dbl>    <dbl>
## 1      5.62     5.79
## 2      5.59     5.79
## 3      5.93     5.79
## 4      6.02     5.81
## 5      5.65     5.81
## 6      5.72     5.82
```

Assess RSq of Each Model to Test Data:

```
rss1 <- sum(((diamonds_test_set$predict_1) - (diamonds_test_set$price))^2)
tss1 <- sum((diamonds_test_set$price - mean(diamonds_test_set$price))^2)
rsq1 <-1 - rss1/tss1

rss2 <- sum(((diamonds_test_set$predict_2) - (diamonds_test_set$price))^2)
tss2 <- sum((diamonds_test_set$price - mean(diamonds_test_set$price))^2)
rsq2 <-1 - rss2/tss2

rss3 <- sum(((diamonds_test_set$predict_3) - (diamonds_test_set$price))^2)
tss3 <- sum((diamonds_test_set$price - mean(diamonds_test_set$price))^2)
rsq3 <-1 - rss3/tss3

rss4 <- sum(((diamonds_test_set$predict_4) - (diamonds_test_set$price))^2)
tss4 <- sum((diamonds_test_set$price - mean(diamonds_test_set$price))^2)
rsq4 <-1 - rss4/tss4

rss5 <- sum(((diamonds_test_set$predict_5) - (diamonds_test_set$logprice))^2)
tss5 <- sum((diamonds_test_set$logprice - mean(diamonds_test_set$logprice))^2)
rsq5 <-1 - rss5/tss5

rss6 <- sum(((diamonds_test_set$predict_6) - (diamonds_test_set$logprice))^2)
tss6 <- sum((diamonds_test_set$logprice - mean(diamonds_test_set$logprice))^2)
rsq6 <-1 - rss6/tss6

rss7 <- sum(((diamonds_test_set$predict_7) - (diamonds_test_set$logprice))^2)
tss7 <- sum((diamonds_test_set$logprice - mean(diamonds_test_set$logprice))^2)
rsq7 <-1 - rss7/tss7

rss8 <- sum(((diamonds_test_set$predict_8) - (diamonds_test_set$logprice))^2)
tss8 <- sum((diamonds_test_set$logprice - mean(diamonds_test_set$logprice))^2)
rsq8 <-1 - rss8/tss8

print(c(rsq1, rsq2, rsq3, rsq4, rsq5, rsq6, rsq7, rsq8))
```

```
## [1] 0.8976301 0.8995929 0.9188462 0.9189759 0.9336559 0.9653696 0.9827421
## [8] 0.9827407
```

- It can be seen that all of the models trained by the diamonds training set generate RSq values that are approximately equal to 0.9, or are above 0.9. This means that these models are all able to predict at least 90% of the test set data.

Predicted vs Actual Price Plots:

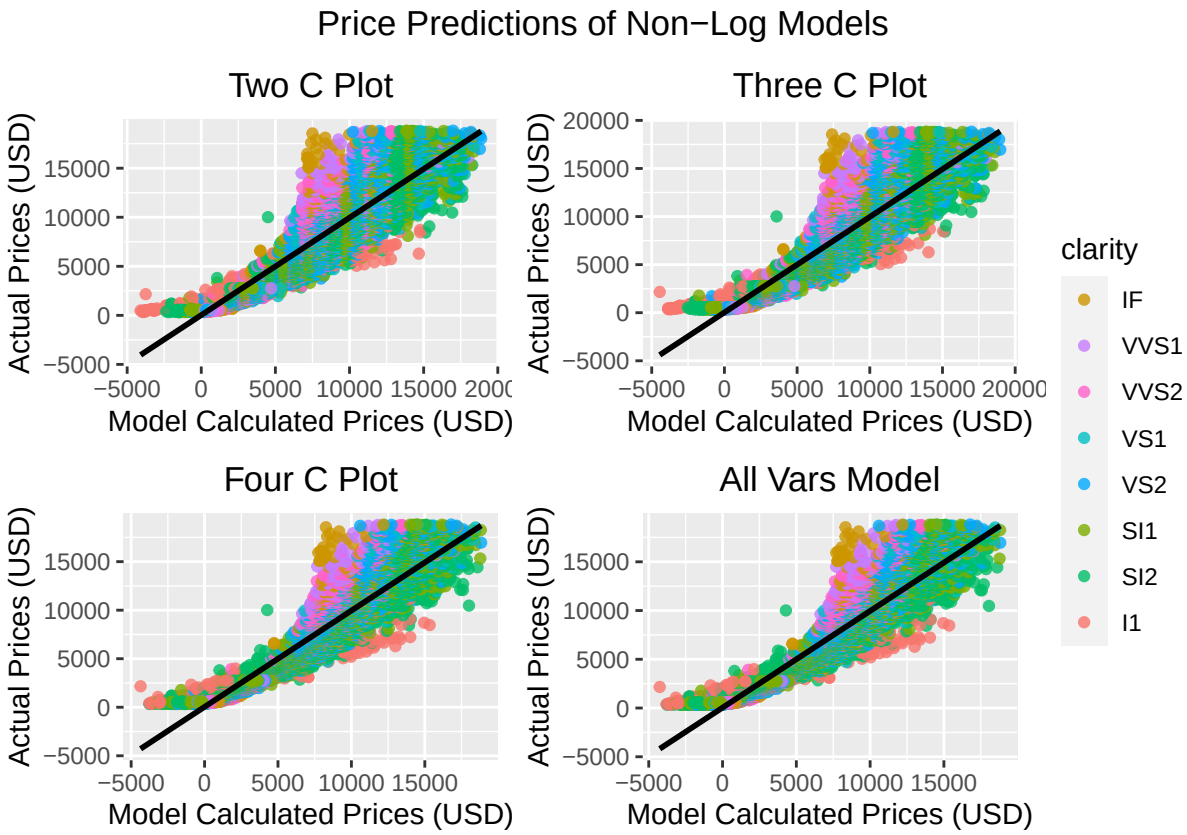
```
pred_act_1 <- diamonds_test_set %>%
  ggplot(mapping = aes(x = predict_1, y = price, color = clarity)) +
  geom_point(alpha = 0.8) +
  geom_smooth(method = "lm",
              color = "black")  +
  xlab("Model Calculated Prices (USD)") +
  ylab("Actual Prices (USD)") +
  ggtitle("Two C Plot") +
  theme(plot.title = element_text(hjust = 0.5))  +
  scale_color_discrete(breaks=c('IF', 'VVS1', 'VVS2', "VS1", "VS2", "SI1", "SI2", "I1"))

pred_act_2 <- diamonds_test_set %>%
  ggplot(mapping = aes(x = predict_2, y = price, color = clarity)) +
  geom_point(alpha = 0.8) +
  geom_smooth(method = "lm",
              color = "black")  +
  xlab("Model Calculated Prices (USD)") +
  ylab("Actual Prices (USD)") +
  ggtitle("Three C Plot") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_color_discrete(breaks=c('IF', 'VVS1', 'VVS2', "VS1", "VS2", "SI1", "SI2", "I1"))

pred_act_3 <- diamonds_test_set %>%
  ggplot(mapping = aes(x = predict_3, y = price, color = clarity)) +
  geom_point(alpha = 0.8) +
  geom_smooth(method = "lm",
              color = "black")  +
  xlab("Model Calculated Prices (USD)") +
  ylab("Actual Prices (USD)") +
  ggtitle("Four C Plot") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_color_discrete(breaks=c('IF', 'VVS1', 'VVS2', "VS1", "VS2", "SI1", "SI2", "I1"))

pred_act_4 <-diamonds_test_set %>%
  ggplot(mapping = aes(x = predict_4, y = price, color = clarity)) +
  geom_point(alpha = 0.8) +
  geom_smooth(method = "lm",
              color = "black")  +
  xlab("Model Calculated Prices (USD)") +
  ylab("Actual Prices (USD)") +
  ggtitle("All Vars Model") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_color_discrete(breaks=c('IF', 'VVS1', 'VVS2', "VS1", "VS2", "SI1", "SI2", "I1"))

pred_act_5 <-diamonds_test_set %>%
  ggplot(mapping = aes(x = predict_5, y = logprice, color = clarity)) +
  geom_point(alpha = 0.8) +
  geom_smooth(method = "lm",
              color = "black")  +
  xlab("Model Calculated Prices (USD)") +
  ylab("Actual Prices (USD)") +
```

```
  ggtitle("Log-C Plot") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_color_discrete(breaks=c('IF', 'VVS1', 'VVS2', "VS1", "VS2", "SI1", "SI2", "I1"))

pred_act_6 <-diamonds_test_set %>%
  ggplot(mapping = aes(x = predict_6, y = logprice, color = clarity)) +
  geom_point(alpha = 0.8) +
  geom_smooth(method = "lm",
              color = "black")  +
  xlab("Model Calculated Prices (USD)") +
  ylab("Actual Prices (USD)") +
  ggtitle("Log 2C Plot") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_color_discrete(breaks=c('IF', 'VVS1', 'VVS2', "VS1", "VS2", "SI1", "SI2", "I1"))

pred_act_7 <-diamonds_test_set %>%
  ggplot(mapping = aes(x = predict_7, y = logprice, color = clarity)) +
  geom_point(alpha = 0.8) +
  geom_smooth(method = "lm",
              color = "black")  +
  xlab("Model Calculated Prices (USD)") +
  ylab("Actual Prices (USD)") +
  ggtitle("Log 4C Plot") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_color_discrete(breaks=c('IF', 'VVS1', 'VVS2', "VS1", "VS2", "SI1", "SI2", "I1"))

pred_act_8 <-diamonds_test_set %>%
  ggplot(mapping = aes(x = predict_8, y = logprice, color = clarity)) +
  geom_point(alpha = 0.8) +
  geom_smooth(method = "lm",
              color = "black")  +
  xlab("Model Calculated Prices (USD)") +
  ylab("Actual Prices (USD)") +
  ggtitle("Log All Vars Plot") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_color_discrete(breaks=c('IF', 'VVS1', 'VVS2', "VS1", "VS2", "SI1", "SI2", "I1"))

no_log_mods <- (pred_act_1 | pred_act_2) / (pred_act_3 | pred_act_4) +
  plot_layout(guides = "collect") +
  plot_annotation(title = "Price Predictions of Non-Log Models",
                  theme = theme(plot.title = element_text(hjust = 0.5)))
log_mods <- (pred_act_5 | pred_act_6) / (pred_act_7 | pred_act_8) +
  plot_layout(guides = "collect")
  plot_annotation(title = "Price Predictions of Non-Log Models",
                  theme = theme(plot.title = element_text(hjust = 0.5)))
```

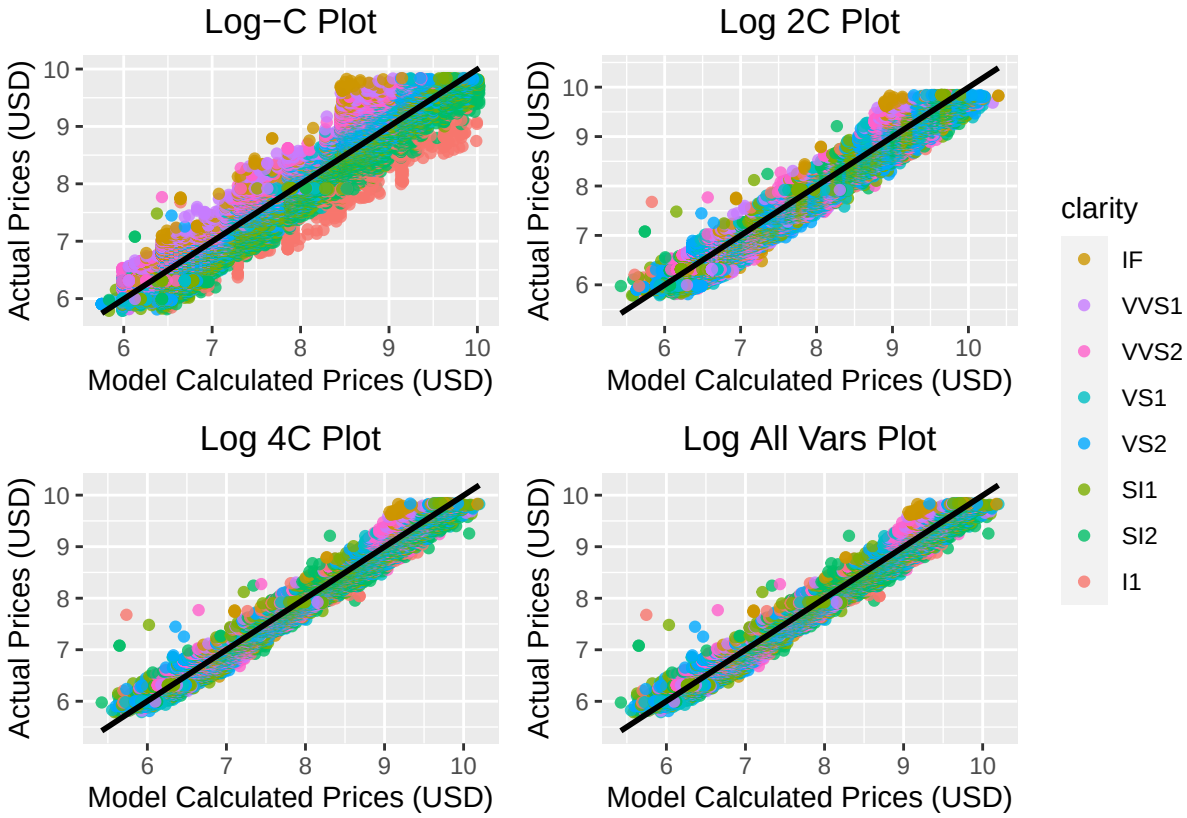
```
## $title
## [1] "Price Predictions of Non-Log Models"
##
## $subtitle
## NULL
##
## $caption
```

```
## NULL
##
## $tag_levels
## NULL
##
## $tag_prefix
## NULL
##
## $tag_suffix
## NULL
##
## $tag_sep
## NULL
##
## $theme
## List of 1
##  $ plot.title:List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0.5
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi FALSE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  - attr(*, "class")= chr [1:2] "theme" "gg"
##  - attr(*, "complete")= logi FALSE
##  - attr(*, "validate")= logi TRUE
##
## attr(,"class")
## [1] "plot_annotation"
```

```
no_log_mods
```


Price Predictions of Non−Log Models
Two C Plot
Three C Plot
Four C Plot
All Vars Model
Actual Prices (USD)
Model Calculated Prices (USD)
clarity
IF
VVS1
VVS2
VS1
VS2
SI1
SI2
I1

```
log_mods
```



Observe how the non-log models globally underestimate prices at the lower end of diamond prices. Likewise, the non-log Four-C plot is not effective enough at predicting gradations in clarity. We will select the log_1_model, log_3_model and log_model_4 to demonstrate how 1) the log_1_model is effective at generalizing price increases but fails to capture gradataions in cut, 2) log_model_3 is both a strong predictor of diamond price and is likewise very effective at explaining gradations in diamond price relative to cut, and 3) log_model_4 is a highly complex model that predicts the data effectively but is poor at explaining diamond price given the additional covariates of table and depth.

Preparation of Graphs and Tables for Presentation:

Stargazer Table:

a. Define SE Values for Log Models:

```
se_log_model_1 <- log_model_1 %>% vcovHC(type = "HC1") %>% diag() %>% sqrt()
se_log_model_3 <- log_model_3 %>% vcovHC(type = "HC1") %>% diag() %>% sqrt()
se_log_model_4 <- log_model_4 %>% vcovHC(type = "HC1") %>% diag() %>% sqrt()
```

b. Plot Table in Text Format

Estimated Regressions

```
                                  Output Variable: Log (Price of Diamond)
```

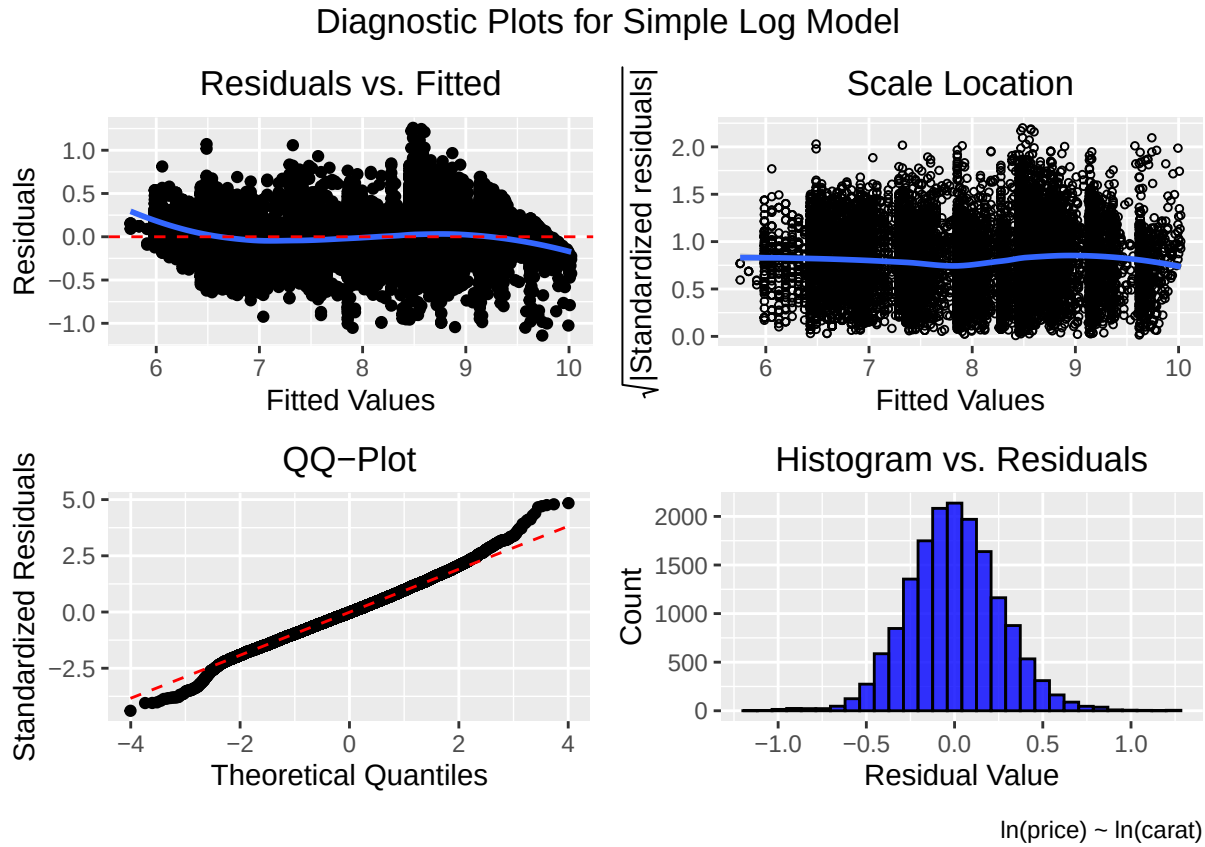
	(1)	(2)	(3)
Log Weight(Carat)	1.68*** (0.004)	1.89*** (0.002)	1.89*** (0.002)
Clarity IF	1.12*** (0.02)	1.11*** (0.02)	
Clarity SI1	0.59*** (0.01)	0.58*** (0.01)	
Clarity SI2	0.42*** (0.01)	0.42*** (0.01)	
Clarity VS1	0.81*** (0.01)	0.81*** (0.01)	
Clarity VS2	0.74*** (0.01)	0.73*** (0.01)	
Clarity VVS1	1.02*** (0.02)	1.02*** (0.02)	
Clarity VVS2	0.94*** (0.01)	0.94*** (0.01)	
Cut Good	0.08*** (0.01)	0.08*** (0.01)	
Cut Ideal	0.16*** (0.01)	0.15*** (0.01)	
Cut Premium	0.13*** (0.01)	0.13*** (0.01)	
Cut Very Good	0.12*** (0.01)	0.11*** (0.01)	
Color E	-0.05*** (0.004)	-0.05*** (0.004)	
Color F	-0.10*** (0.004)	-0.10*** (0.004)	
Color G	-0.16*** (0.004)	-0.16*** (0.004)	
Color H	-0.25*** (0.004)	-0.25*** (0.004)	
Color I	-0.37*** (0.005)	-0.37*** (0.005)	
Color J	-0.50*** (0.01)	-0.50*** (0.01)	
Depth	-0.002 (0.001)		
Table	-0.001 (0.001)		
Constant	8.46*** (0.003)	7.87*** (0.02)	8.03*** (0.08)
Weight (Carat)			
Clarity/Cut/Color			
Additional Features			
Observations	16,141	16,141	16,141
R2	0.93	0.98	0.98
Residual Std. Error	0.26 (df = 16139)	0.13 (df = 16122)	0.13 (df = 16120)

Note: *p<0.05;* ***p<0.01;*** p<0.001
HCrobust standard errors in parentheses.features are Depth and Table.
Note that converting to LaTeX will add additional features, such as check mark guide of present variables (Above Observations Section)

Model 5 Stat Plots

```
model_5_HCV_pres <- ggplot(log_model_1, aes(.fitted, sqrt(abs(.stdresid)))) +
    geom_point(na.rm=TRUE,
               size = 1,
               shape = 1) +
    stat_smooth(method="loess", na.rm = TRUE) +
    xlab("Fitted Values") +
    ylab(expression(sqrt("|Standardized residuals|"))) +
    ggtitle("Scale Location")+
    theme(plot.title = element_text(hjust = 0.5))
model_5_LCE_pres <- ggplot(log_model_1, aes(.fitted, .resid)) +
```

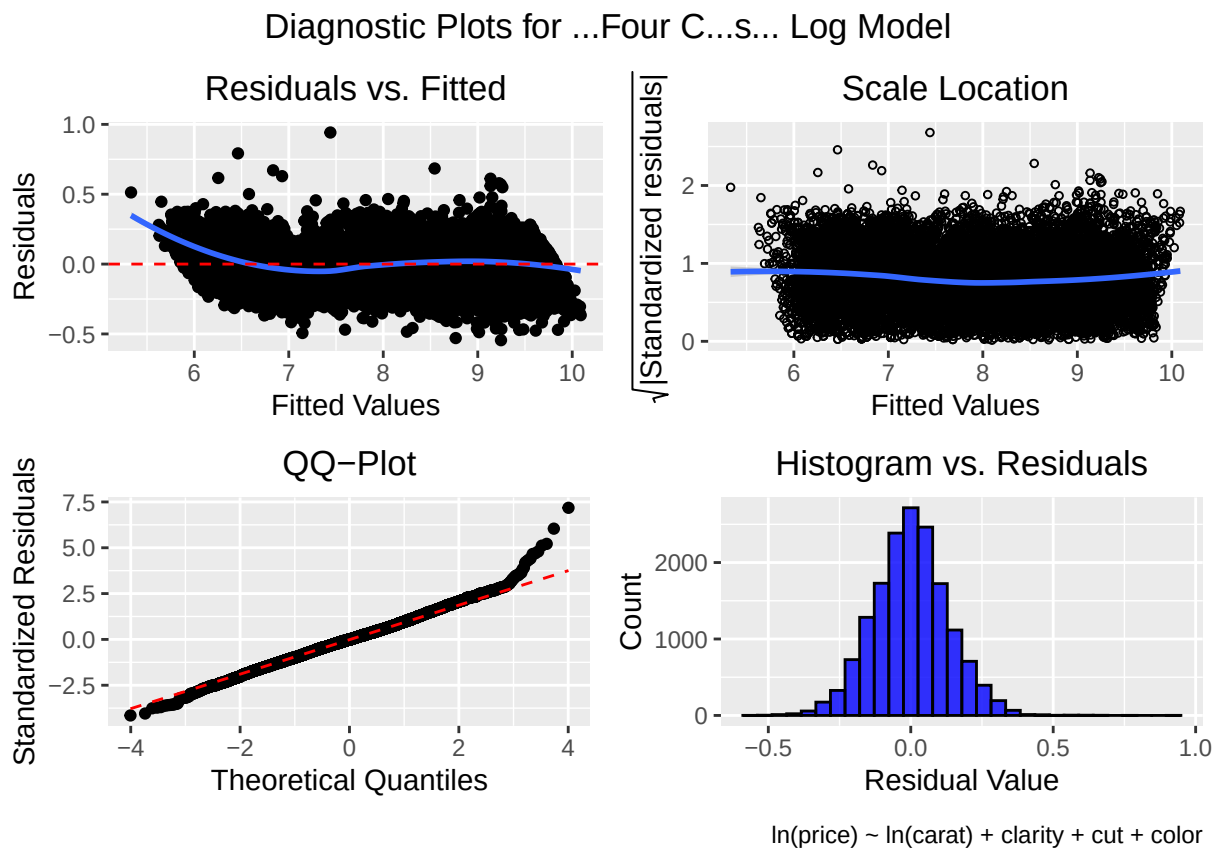
```
    geom_point() +
    stat_smooth(method="loess") +
    geom_hline(yintercept=0, col="red", linetype="dashed") +
    xlab("Fitted Values") +
    ylab("Residuals") +
    ggtitle("Residuals vs. Fitted") +
    theme(plot.title = element_text(hjust = 0.5))
model_5_QQ_pres <- ggplot(data = log_model_1,
              aes(x = qqnorm(rstandard(log_model_1), plot = FALSE)[[1]],
                  y = qqnorm(rstandard(log_model_1), plot = FALSE)[[2]])
              ) +
    geom_point(na.rm = TRUE) +
    stat_qq_line(mapping = aes(sample = qqnorm(rstandard(log_model_1), plot = FALSE)[[2]]),
                 col="red",
                 linetype="dashed") +
    xlab("Theoretical Quantiles") +
    ylab("Standardized Residuals") +
    ggtitle("QQ-Plot") +
    theme(plot.title = element_text(hjust = 0.5))
model_5_hr_pres <- ggplot(data = log_model_1, mapping = aes(x = .resid)) +
  geom_histogram(stat = "bin",
                         color = "black",
                         fill = "blue",
                         alpha = 0.8) +
  xlab("Residual Value") +
  ylab("Count") +
  ggtitle("Histogram vs. Residuals") +
  theme(plot.title = element_text(hjust = 0.5))

model_5_sub_stat <- (model_5_LCE_pres + model_5_HCV_pres) / (model_5_QQ_pres + model_5_hr_pres) +
  plot_annotation(title = "Diagnostic Plots for Simple Log Model",
                  theme = theme(plot.title = element_text(hjust = 0.5)),
                  caption = "ln(price) ~ ln(carat)")

model_5_sub_stat
```



Model 7 Stat Plots

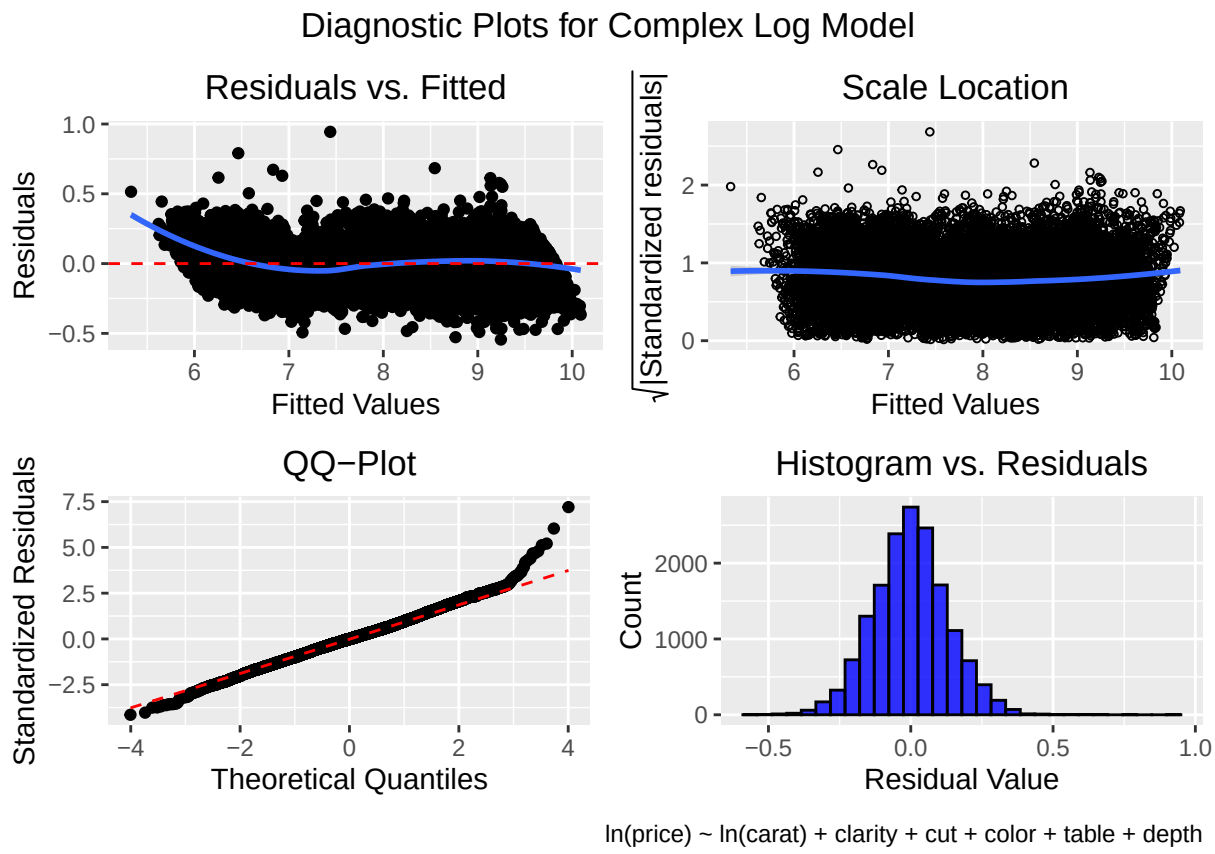
```
model_7_HCV_pres <- ggplot(log_model_3, aes(.fitted, sqrt(abs(.stdresid)))) +
    geom_point(na.rm=TRUE,
               size = 1,
               shape = 1) +
    stat_smooth(method="loess", na.rm = TRUE) +
    xlab("Fitted Values") +
    ylab(expression(sqrt("|Standardized residuals|"))) +
    ggtitle("Scale Location")+
    theme(plot.title = element_text(hjust = 0.5))
model_7_LCE_pres <- ggplot(log_model_3, aes(.fitted, .resid)) +
    geom_point() +
    stat_smooth(method="loess") +
    geom_hline(yintercept=0, col="red", linetype="dashed") +
    xlab("Fitted Values") +
    ylab("Residuals") +
    ggtitle("Residuals vs. Fitted") +
    theme(plot.title = element_text(hjust = 0.5))
model_7_QQ_pres <- ggplot(data = log_model_3,
              aes(x = qqnorm(rstandard(log_model_3), plot = FALSE)[[1]],
                  y = qqnorm(rstandard(log_model_3), plot = FALSE)[[2]])
              ) +
    geom_point(na.rm = TRUE) +
```

```
    stat_qq_line(mapping = aes(sample = qqnorm(rstandard(log_model_3), plot = FALSE)[[2]]),
                 col="red",
                 linetype="dashed") +
    xlab("Theoretical Quantiles") +
    ylab("Standardized Residuals") +
    ggtitle("QQ-Plot") +
    theme(plot.title = element_text(hjust = 0.5))
model_7_hr_pres <- ggplot(data = log_model_3, mapping = aes(x = .resid)) +
  geom_histogram(stat = "bin",
                           color = "black",
                           fill = "blue",
                           alpha = 0.8) +
  xlab("Residual Value") +
  ylab("Count") +
  ggtitle("Histogram vs. Residuals") +
  theme(plot.title = element_text(hjust = 0.5))

model_7_sub_stat <- (model_7_LCE_pres + model_7_HCV_pres) / (model_7_QQ_pres + model_7_hr_pres) +
  plot_annotation(title = "Diagnostic Plots for "Four C's" Log Model",
                  theme = theme(plot.title = element_text(hjust = 0.5)),
                  caption = "ln(price) ~ ln(carat) + clarity + cut + color")

model_7_sub_stat
```



Model 8 Stat Plots

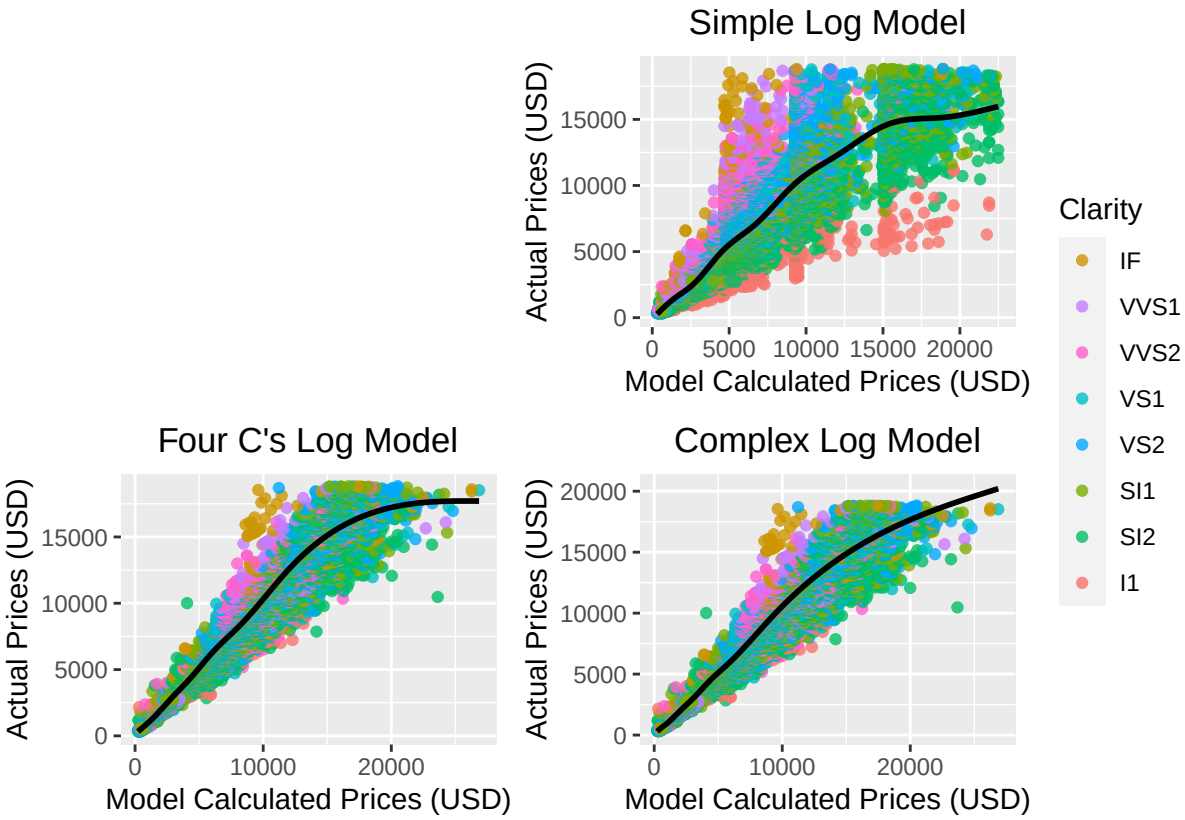
```
model_8_HCV_pres <- ggplot(log_model_4, aes(.fitted, sqrt(abs(.stdresid)))) +
    geom_point(na.rm=TRUE,
               size = 1,
               shape = 1) +
    stat_smooth(method="loess", na.rm = TRUE) +
    xlab("Fitted Values") +
    ylab(expression(sqrt("|Standardized residuals|"))) +
    ggtitle("Scale Location") +
    theme(plot.title = element_text(hjust = 0.5))
model_8_LCE_pres <- ggplot(log_model_4, aes(.fitted, .resid)) +
    geom_point() +
    stat_smooth(method="loess") +
    geom_hline(yintercept=0, col="red", linetype="dashed") +
    xlab("Fitted Values") +
    ylab("Residuals") +
    ggtitle("Residuals vs. Fitted") +
    theme(plot.title = element_text(hjust = 0.5))
model_8_QQ_pres <- ggplot(data = log_model_4,
              aes(x = qqnorm(rstandard(log_model_4), plot = FALSE)[[1]],
                  y = qqnorm(rstandard(log_model_4), plot = FALSE)[[2]])
              ) +
    geom_point(na.rm = TRUE) +
    stat_qq_line(mapping = aes(sample = qqnorm(rstandard(log_model_4), plot = FALSE)[[2]]),
                 col="red",
                 linetype="dashed") +
    xlab("Theoretical Quantiles") +
    ylab("Standardized Residuals")+
    ggtitle("QQ-Plot") +
    theme(plot.title = element_text(hjust = 0.5))
model_8_hr_pres <- ggplot(data = log_model_4, mapping = aes(x = .resid)) +
  geom_histogram(stat = "bin",
                         color = "black",
                         fill = "blue",
                         alpha = 0.8) +
  xlab("Residual Value") +
  ylab("Count") +
  ggtitle("Histogram vs. Residuals") +
  theme(plot.title = element_text(hjust = 0.5))

model_8_sub_stat <- (model_8_LCE_pres + model_8_HCV_pres) / (model_8_QQ_pres + model_8_hr_pres) +
  plot_annotation(title = "Diagnostic Plots for Complex Log Model",
                  theme = theme(plot.title = element_text(hjust = 0.5)),
                  caption = "ln(price) ~ ln(carat) + clarity + cut + color + table + depth") +
  ggtitle("Histogram vs. Residuals") +
  theme(plot.title = element_text(hjust = 0.5))

model_8_sub_stat
```



Side-By-Side Modeled Price Predictions vs. Actual Prices, Smooth Plots:

```
pred_act_ppt_1 <- diamonds_test_set %>%
  ggplot(mapping = aes(x = exp(predict_5), y = price, color = clarity)) +
  geom_point(alpha = 0.8) +
  geom_smooth(color = "black")  +
  xlab("Model Calculated Prices (USD)") +
  ylab("Actual Prices (USD)") +
  ggtitle("Simple Log Model") +
  theme(plot.title = element_text(hjust = 0.5)) +
  guides(col=guide_legend("Clarity")) +
  scale_color_discrete(breaks=c('IF', 'VVS1', 'VVS2', "VS1", "VS2", "SI1", "SI2", "I1"))

pred_act_ppt_2 <- diamonds_test_set %>%
  ggplot(mapping = aes(x = exp(predict_7), y = price, color = clarity)) +
  geom_point(alpha = 0.8) +
  geom_smooth(color = "black")  +
  xlab("Model Calculated Prices (USD)") +
  ylab("Actual Prices (USD)") +
  ggtitle("Four C's Log Model") +
  theme(plot.title = element_text(hjust = 0.5)) +
  guides(col=guide_legend("Clarity")) +
  scale_color_discrete(breaks=c('IF', 'VVS1', 'VVS2', "VS1", "VS2", "SI1", "SI2", "I1"))
```

```
pred_act_ppt_3 <-diamonds_test_set %>%
  ggplot(mapping = aes(x = exp(predict_8), y = price, color = clarity)) +
  geom_point(alpha = 0.8) +
  geom_smooth(color = "black")  +
  xlab("Model Calculated Prices (USD)") +
  ylab("Actual Prices (USD)") +
  ggtitle("Complex Log Model") +
  theme(plot.title = element_text(hjust = 0.5)) +
  guides(col=guide_legend("Clarity")) +
  scale_color_discrete(breaks=c('IF', 'VVS1', 'VVS2', "VS1", "VS2", "SI1", "SI2", "I1"))

plot_spacer() + pred_act_ppt_1 + pred_act_ppt_2 + pred_act_ppt_3 +
  plot_layout(guides = "collect")
```



The above smooth plots demonstrate the prediction curves generated after exponentiating the model calculated prices for our three target models. It can be seen how the Simple Log Model (log_model_5) can predict diamond prices accurately, but precision is lost due to its inability to capture diamond clarity. The Complex Log Model does not demonstrate superior predictive power versus the Four C's model, and likewise includes covariates that do not pass ### ln transformation plots (using non-filtered data):

```
diamonds_unfilt <- read_csv("/Users/moinzade/Desktop/UCB_MIDS/W203/lab_2/Diamonds Prices2022.csv")
diamonds_unfilt <- as_tibble(diamonds_unfilt)

non_ln_plot <- ggplot(data = diamonds_unfilt, mapping = aes(x = carat, y = price, color = clarity)) +
  geom_point() +
  ggtitle("Figure 1: Before Transformation") +
```

```
  xlab("Carats (0.2g/c)") +
  ylab("Price (USD)") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5)) +
  guides(col=guide_legend("Clarity")) +
  scale_color_discrete(breaks=c('IF', 'VVS1', 'VVS2', "VS1", "VS2", "SI1", "SI2", "I1"))

ln_plot <- ggplot(data = diamonds_unfilt, mapping = aes(x = log(carat), y = log(price), color = clarity)
  geom_point() +
  ggtitle("Figure 2. After Transformation") +
  xlab("ln(Carats)") +
  ylab("ln(Price)") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))+
  guides(col=guide_legend("Clarity")) +
  scale_color_discrete(breaks=c('IF', 'VVS1', 'VVS2', "VS1", "VS2", "SI1", "SI2", "I1"))

non_ln_plot + ln_plot +
  plot_layout(guides = "collect") +
  plot_annotation(title = "Logarithmic Transformation of Carat and Price",
                  theme = theme(plot.title = element_text(hjust = 0.5)))
```