Diamond Price Coding Project

Gerald Ayers

11/6/2024

Contents

0.1 Descriptive Analysis of all the variables

```
diamonds_set <- read.csv(file = "Diamonds Prices2022.csv", header = TRUE, sep = ",", str
head(diamonds_set)
```

```
str(diamonds_set)
```

```
## 'data.frame':    53943 obs. of  11 variables:
##  $ X      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ carat  : num  0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
##  $ cut    : Factor w/ 5 levels "Fair","Good",..: 3 4 2 4 2 5 5 5 1 5 ...
##  $ color  : Factor w/ 7 levels "D","E","F","G",..: 2 2 2 6 7 7 6 5 2 5 ...
##  $ clarity: Factor w/ 8 levels "I1","IF","SI1",..: 4 3 5 6 4 8 7 3 6 5 ...
##  $ depth  : num  61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
##  $ table  : num  55 61 65 58 58 57 57 55 61 61 ...
##  $ price  : int  326 326 327 334 335 336 336 337 337 338 ...
##  $ x      : num  3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
##  $ y      : num  3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
##  $ z      : num  2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
```

```
stat.desc(diamonds_set)
```

These lines of code describe the diamonds_set data set by examining the structure of the data frame which describes all the column values (variables) and the statistical description for all the quantitative values in the data frame. The data frame includes, X which is just the number of the observation, carat which is a number representing the weight, cut which is a factor with 5 levels, color which is a factor with 7 levels, clarity which is a factor with 8 levels, depth which a number, table which is a number, price as an integer, and x,y,z which are all numbers of the dimensions of the diamond.

##Random Sample Selection of size 300

```r
set.seed(5)
sample_size = 300
sample_indices = sample(1:nrow(diamonds_set), size = sample_size, replace = FALSE)
diamonds_set = diamonds_set[sample_indices,]
skim(diamonds_set)
```

Table 1: Data summary

Name	diamonds_set
Number of rows	300
Number of columns	11
Column type frequency:	
factor	3
numeric	8
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
cut	0	1	FALSE	5	Ide: 114, Pre: 80, Ver: 72, Goo: 29
color	0	1	FALSE	7	G: 67, F: 55, E: 53, H: 47
clarity	0	1	FALSE	8	VS2: 73, SI1: 66, SI2: 57, VS1: 44

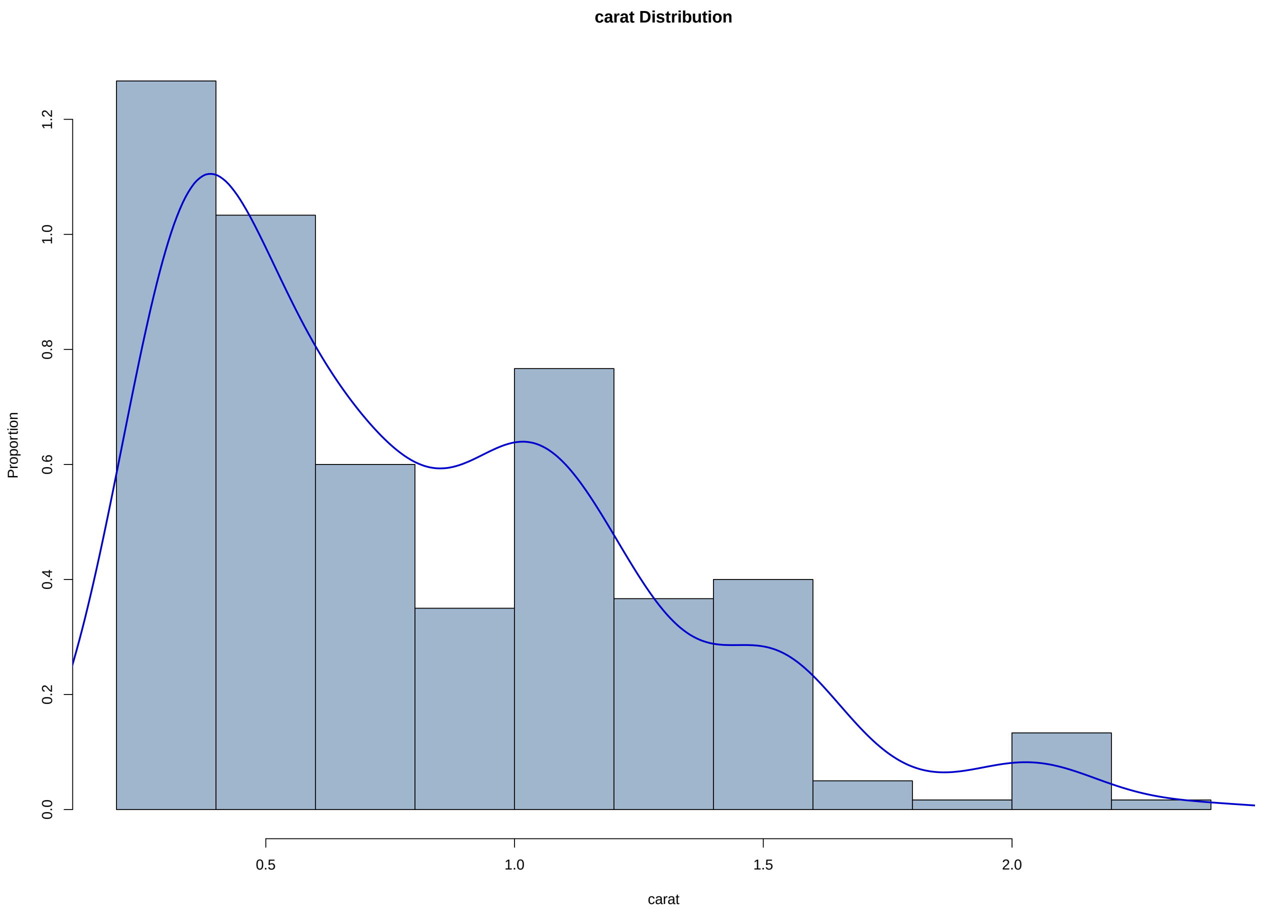
Variable type: numeric

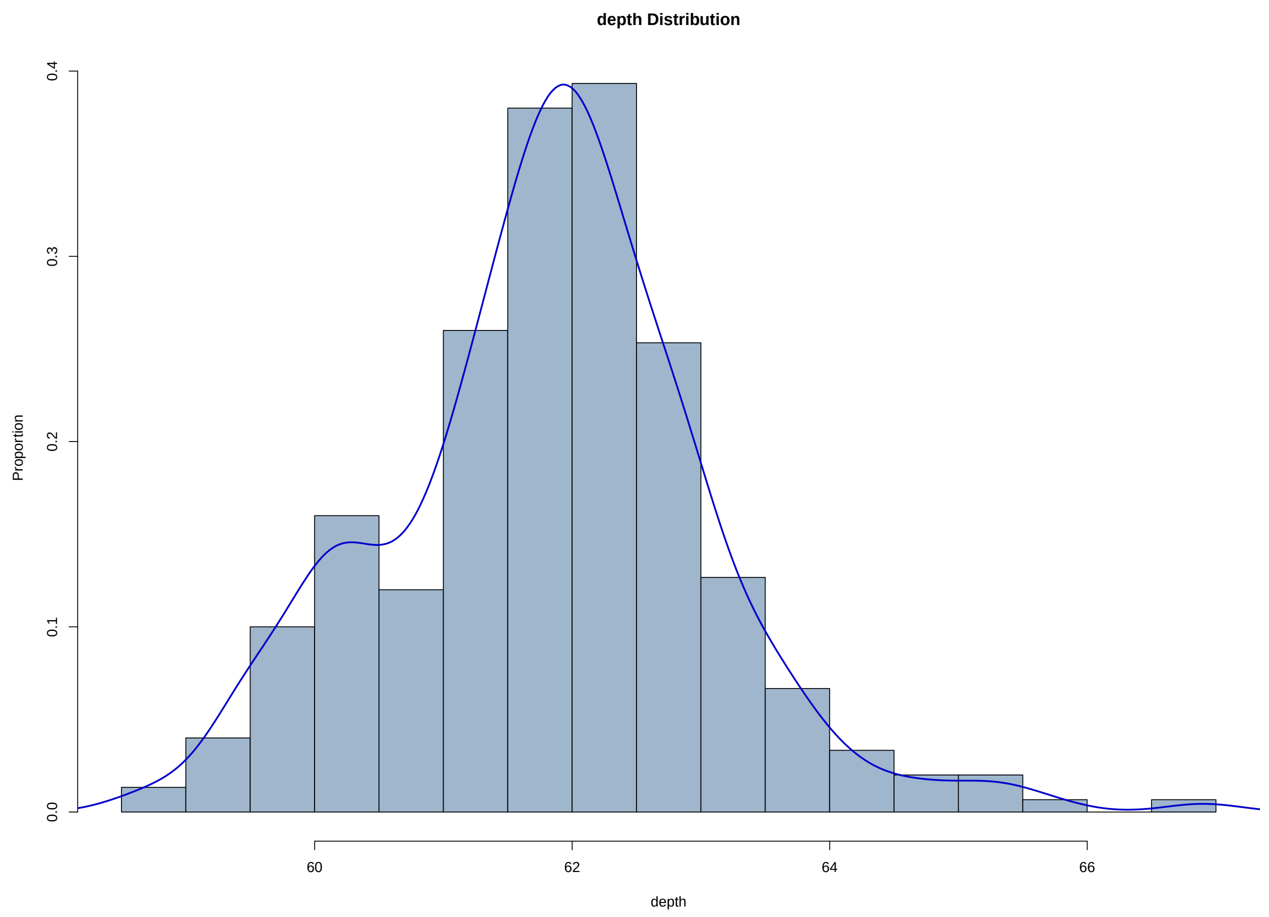
skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
X	0	1	25640.87	15565.61	213.00	12336.50	25631.50	40212.25	53475.00	
carat	0	1	0.79	0.46	0.20	0.39	0.70	1.06	2.37	
depth	0	1	61.81	1.24	58.60	61.10	61.85	62.50	66.90	
table	0	1	57.49	2.17	52.00	56.00	57.00	59.00	65.00	
price	0	1	3839.22	3874.43	357.00	943.25	2352.50	5207.00	18508.00	
x	0	1	5.71	1.11	3.81	4.70	5.67	6.56	8.53	
y	0	1	5.71	1.10	3.77	4.71	5.67	6.56	8.56	
z	0	1	3.53	0.68	2.33	2.89	3.51	4.03	5.29	

I selected a random sample that contains 300 samples from the diamonds data set and ran the skim function to summarize the data of the selected random sample. The skim function shows the mean, standard deviation, and quantiles for each quantitative column of the data frame, and shows the number of unique values of each factor for the categorical variables.

##Distributions of Quantitative Variables

```
for (i in c(2,6,7,8,9,10,11)){
  truehist(diamonds_set[,i], col='slategray3',
         main = paste(colnames(diamonds_set)[i], "Distribution"),
         xlab= colnames(diamonds_set)[i],
         ylab ="Proportion")
lines(density(diamonds_set[,i]),lwd=2, col ="blue3")
}
```




depth Distribution
Proportion
0.0
0.1
0.2
0.3
0.4
60
62
64
66
depth

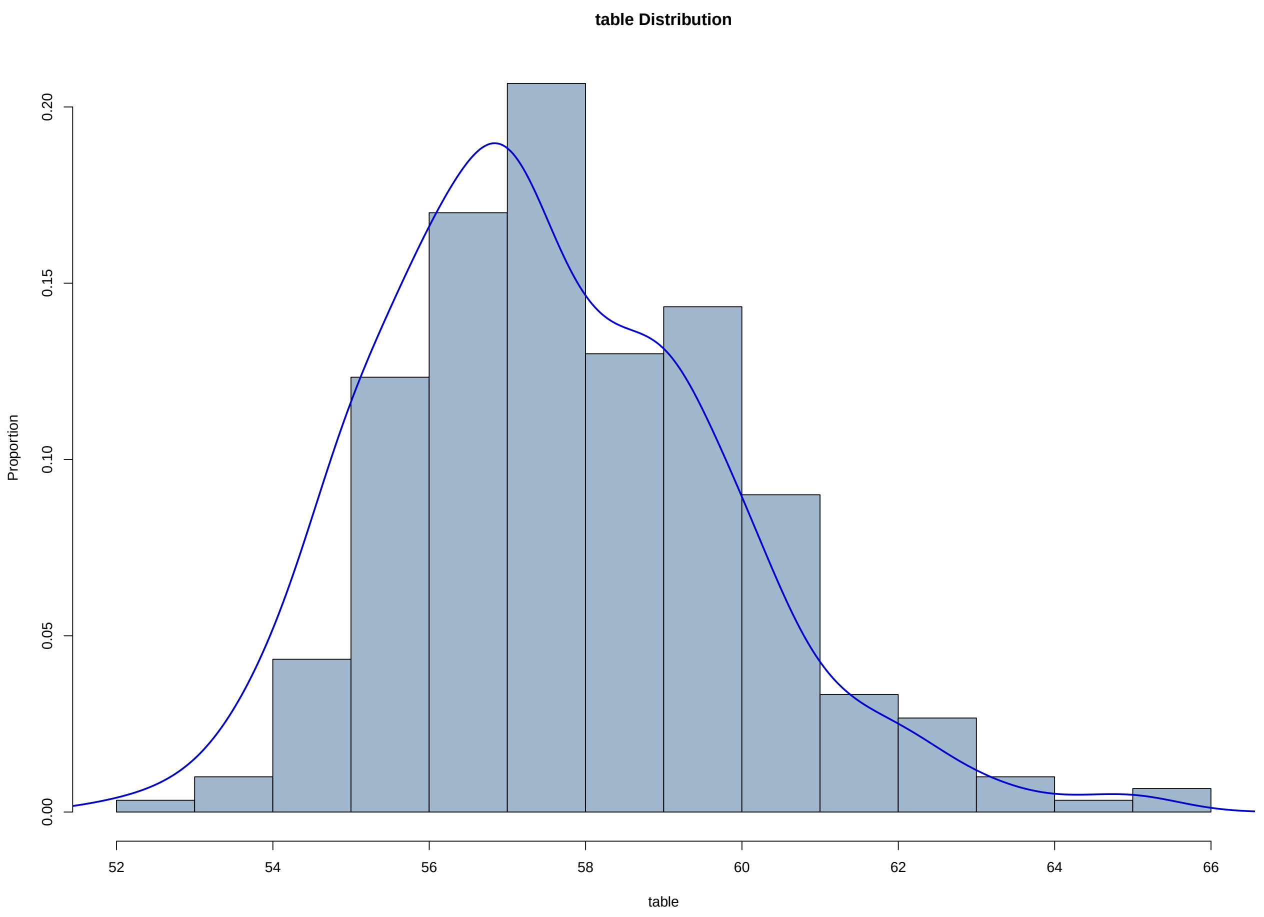
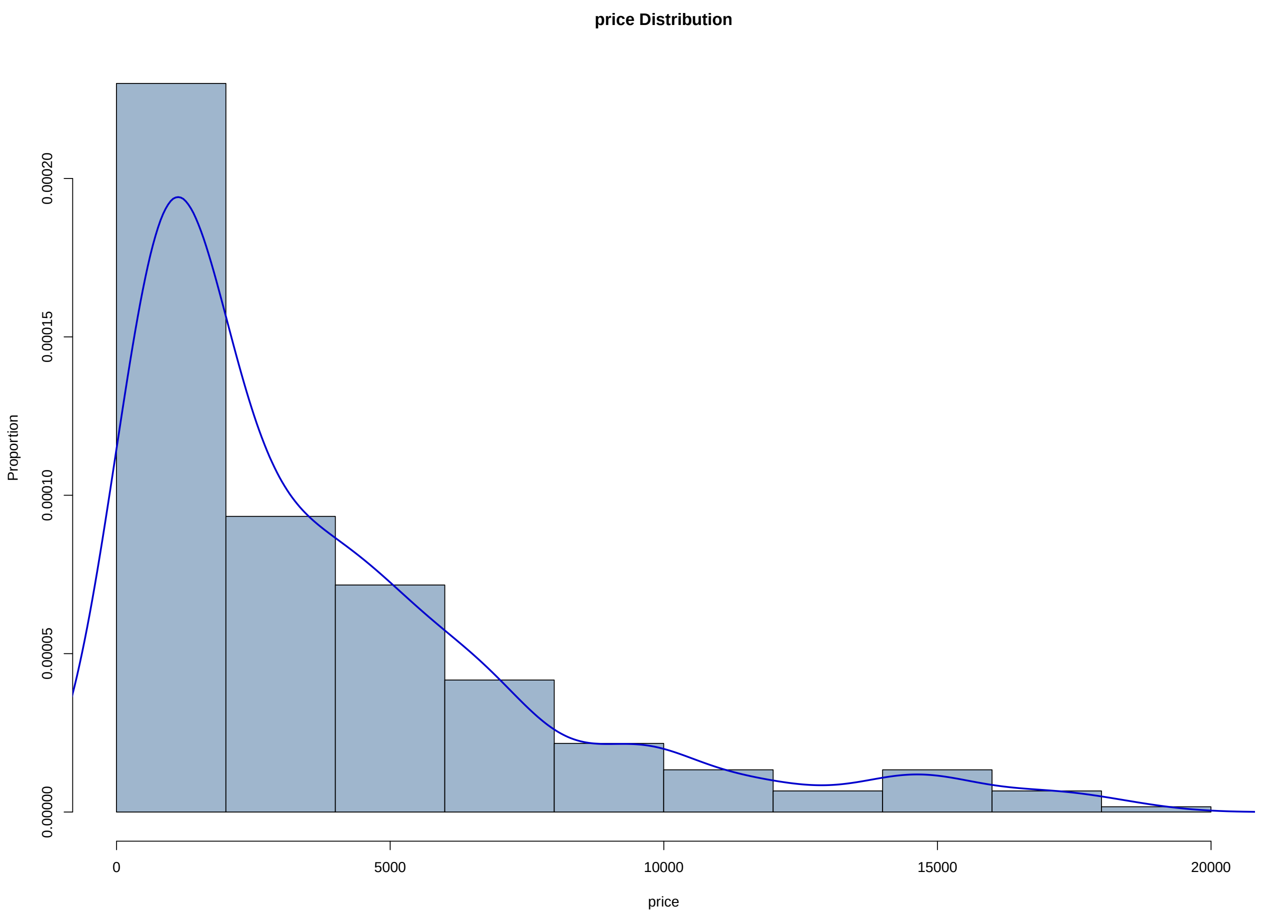
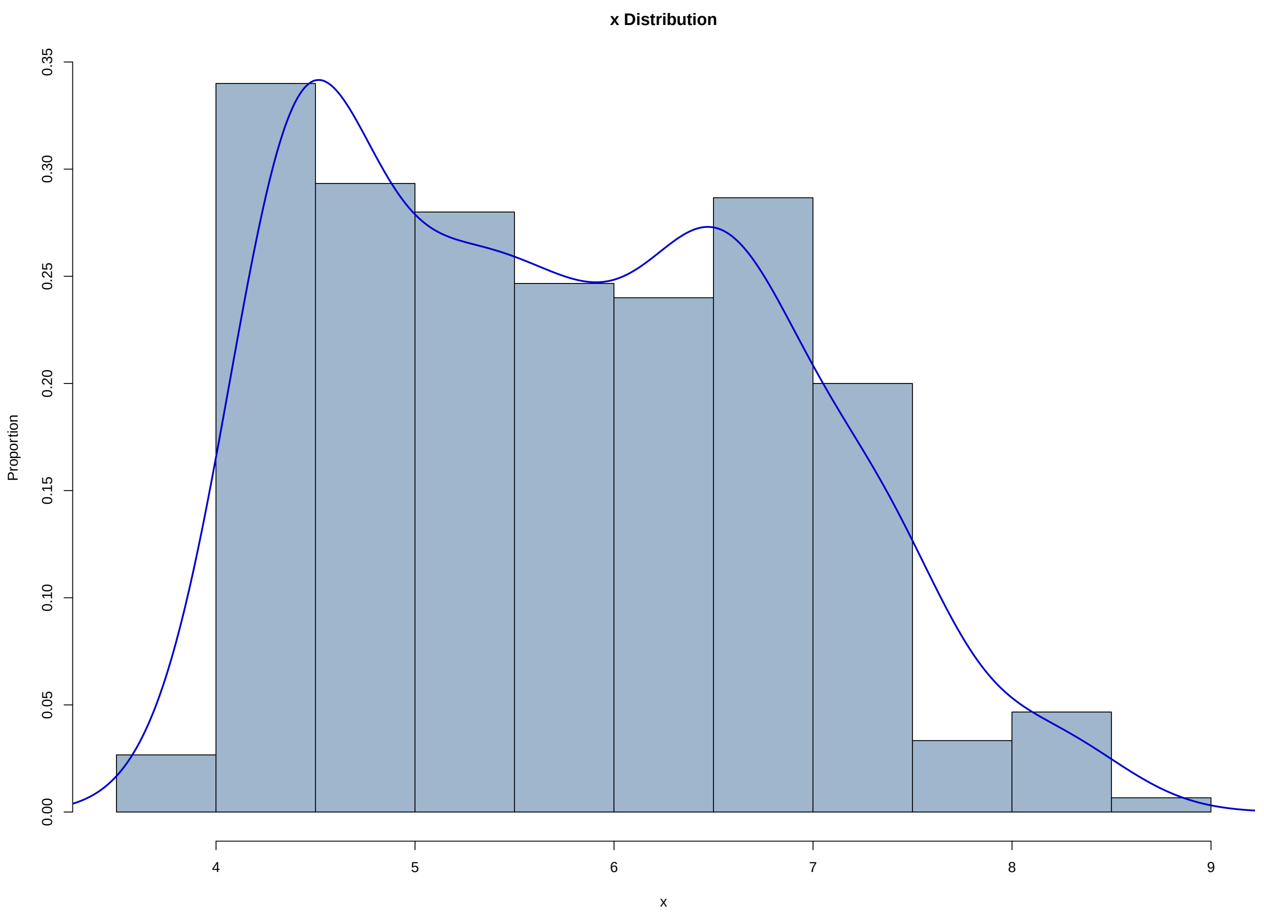
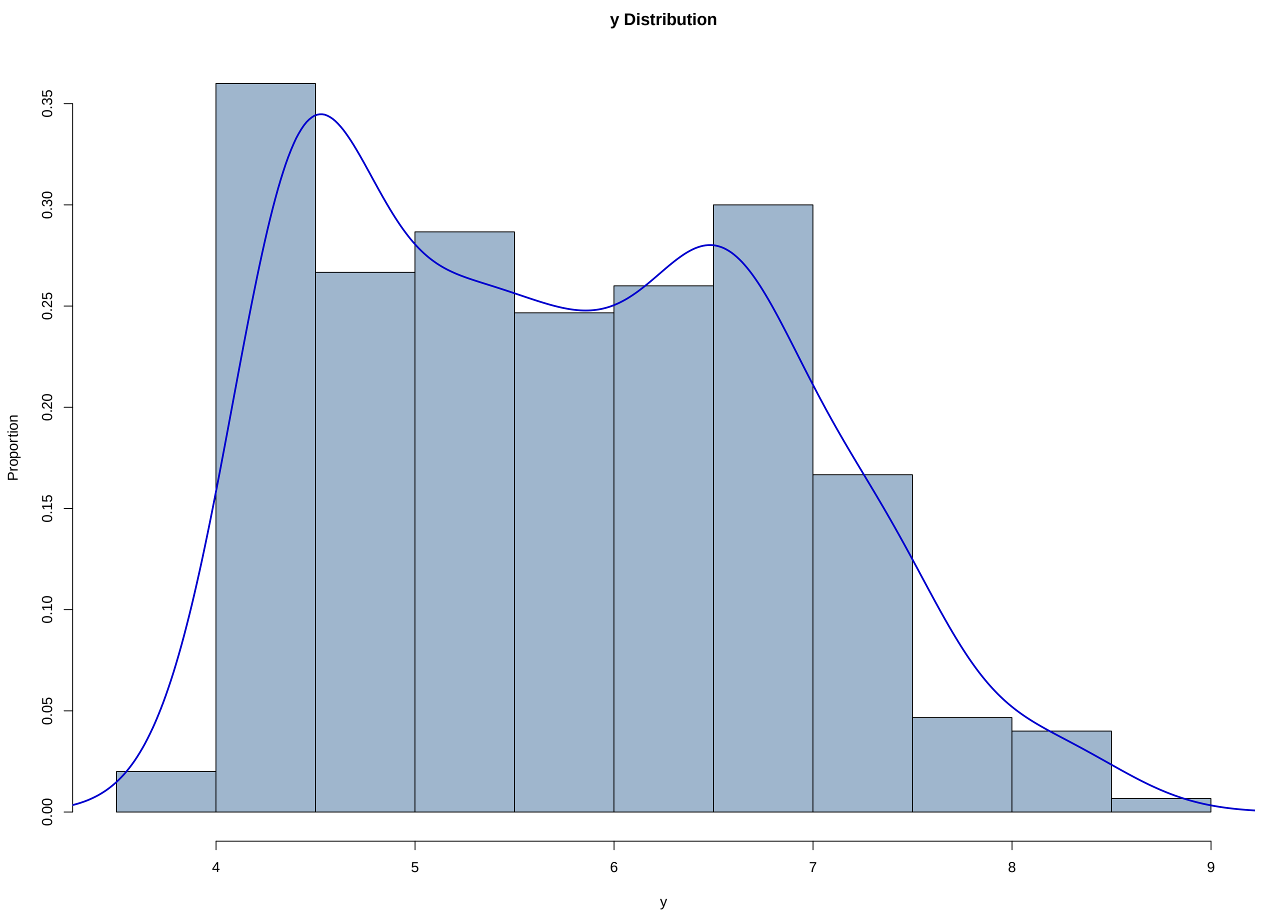
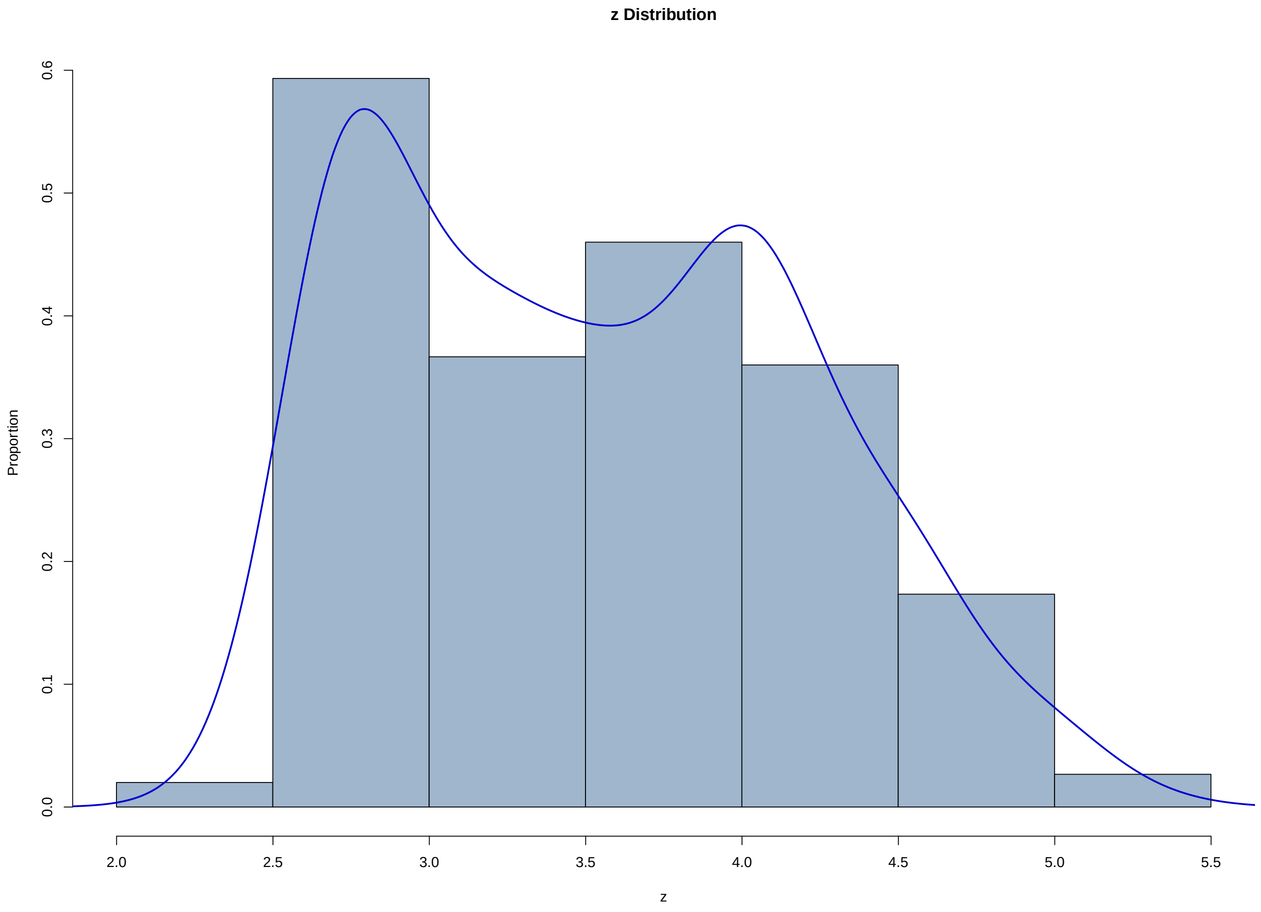


table Distribution
Proportion
0.00
0.05
0.10
0.15
0.20
52
54
56
58
60
62
64
66
table


price Distribution
Proportion
0.00000
0.00005
0.00010
0.00015
0.00020
0
5000
10000
15000
20000
price


x Distribution
Proportion
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
4
5
6
7
8
9
x

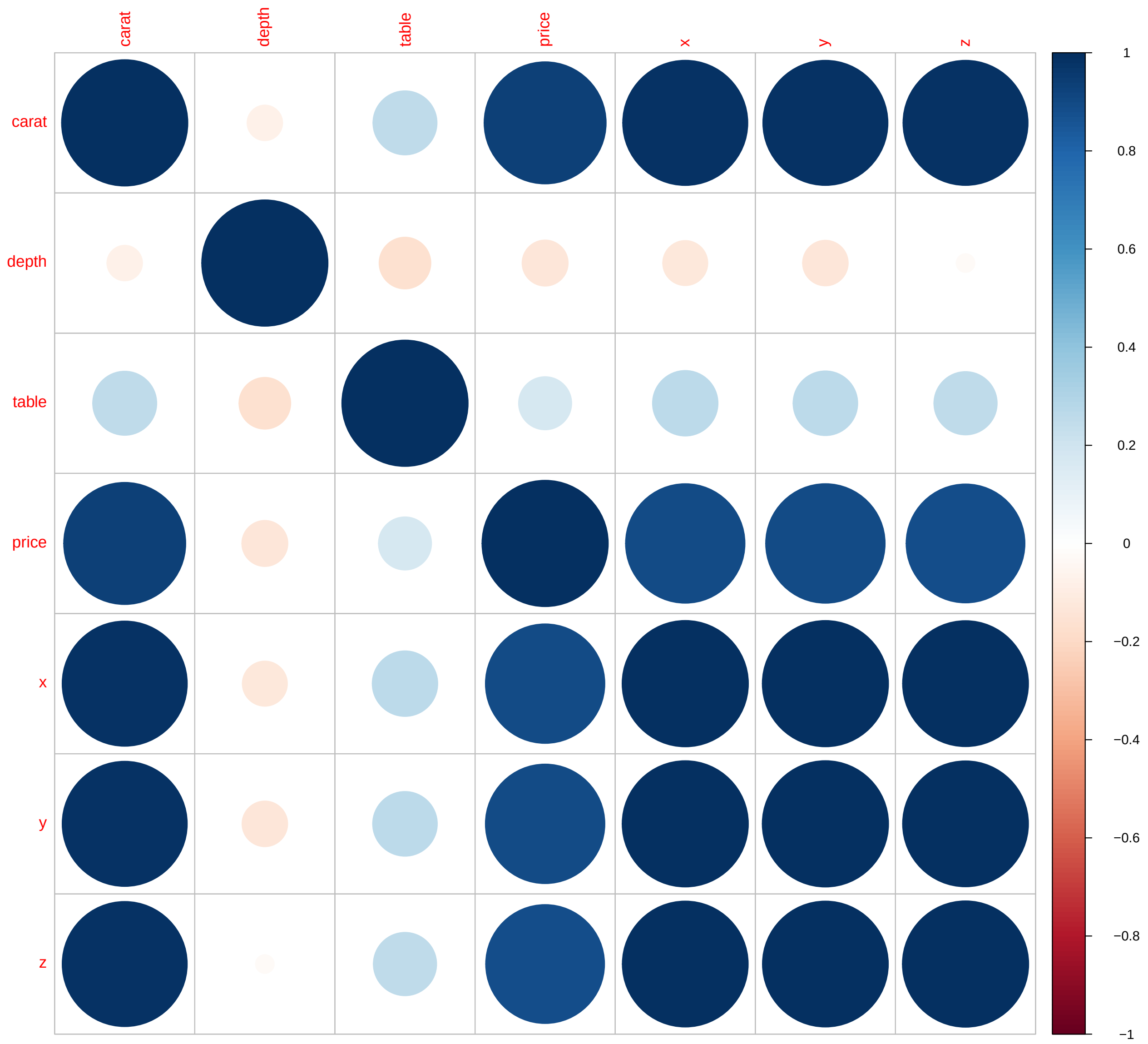

y Distribution
Proportion
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
4
5
6
7
8
9
y



The above histograms show the distributions for each quantitative variable across the 300 samples from the random sample I selected. The carat, price, and table distributions looks right-skewed, the depth distribution looks relatively symmetric, and x,y,z seem to not follow a specific distribution trend.

##Correlation Matrix to help choose independent variables

```
diamonds_num = diamonds_set[,-c(1,3,4,5)]
cor_matrix= cor(diamonds_num,use = "pairwise.complete.obs")
corrplot(cor_matrix)
```



```
#checking correlation between all quantitative variables

cor_matrix[c("carat", "depth", "table"),
           c("carat", "depth", "table")]
```

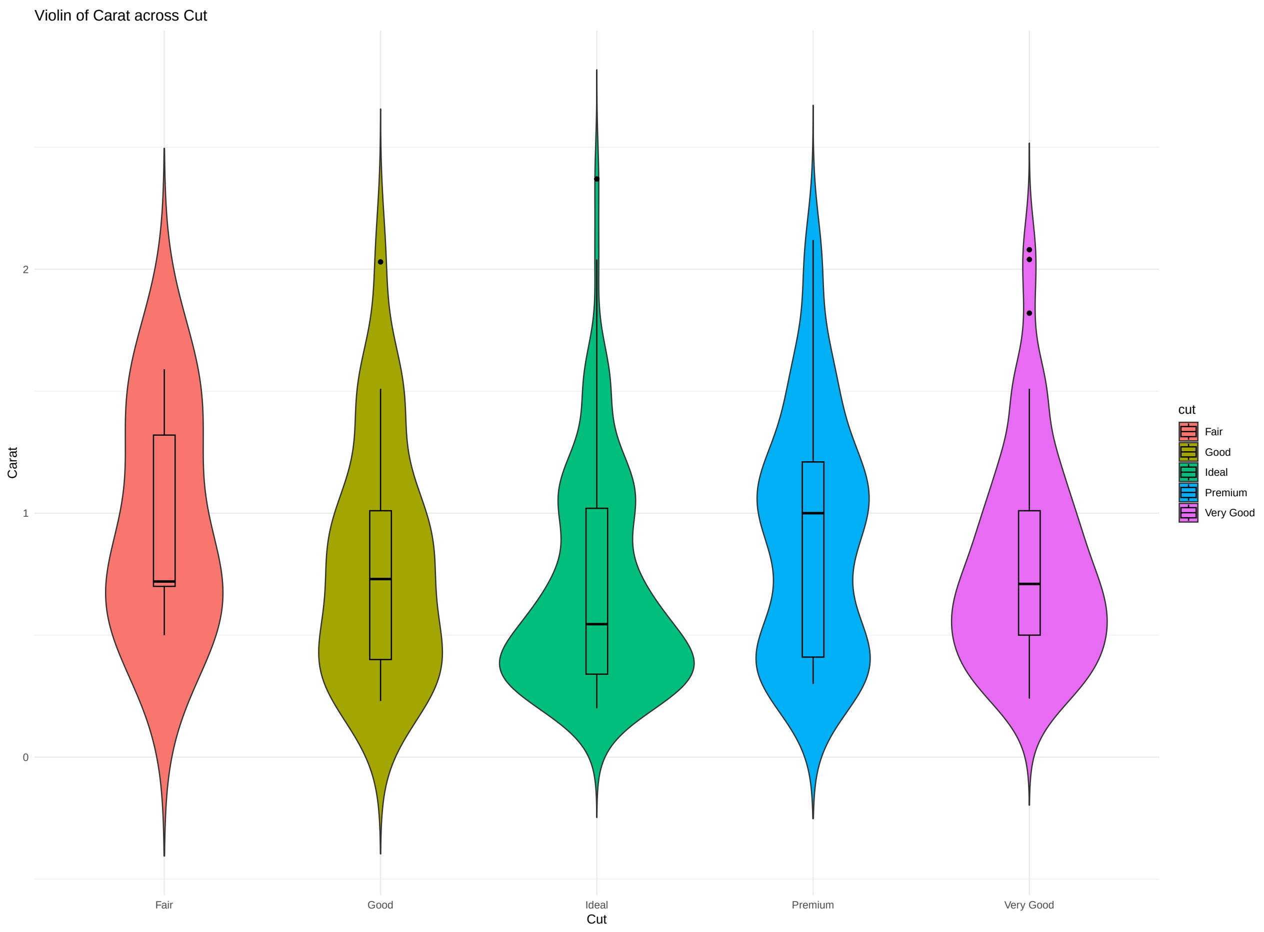
```
##              carat       depth      table
## carat  1.00000000 -0.07898915  0.2568989
## depth -0.07898915  1.00000000 -0.1673810
## table  0.25689887 -0.16738100  1.0000000
```

```
#checking the correlation between my 3 chosen independents is low.
```

The above code creates the correlation matrix for the quantitative values of the data set. Based on the relatively low correlations between carat, depth, and table, these are the three quantitative independent variables I'm choosing from the diamonds data set, and the two categorical variables I'm choosing are cut and color. Based on the correlation matrix, the best pick for the dependent variable is price.

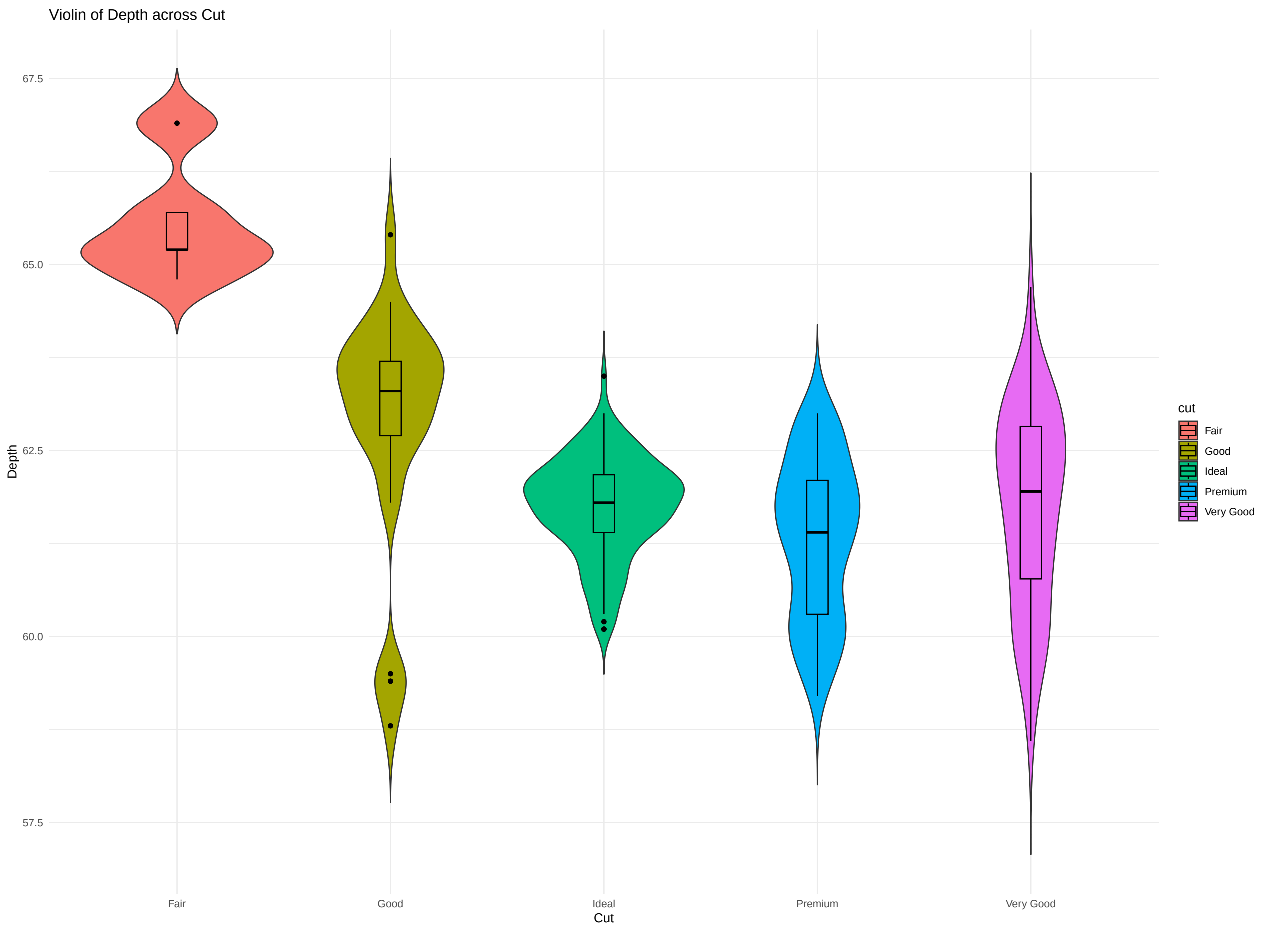
##Checking Correlation between Quantitative and Categorical Variables

```r
# Create a violin plot
diamonds_set %>% ggplot(aes(x = cut, y = carat, fill = cut)) +
  geom_violin(trim = FALSE) +
  geom_boxplot(width = 0.1, color = "black",
               position = position_dodge(0.9))+
  labs(title = "Violin of Carat across Cut",
       x = "Cut", y = "Carat") +
  theme_minimal()
```



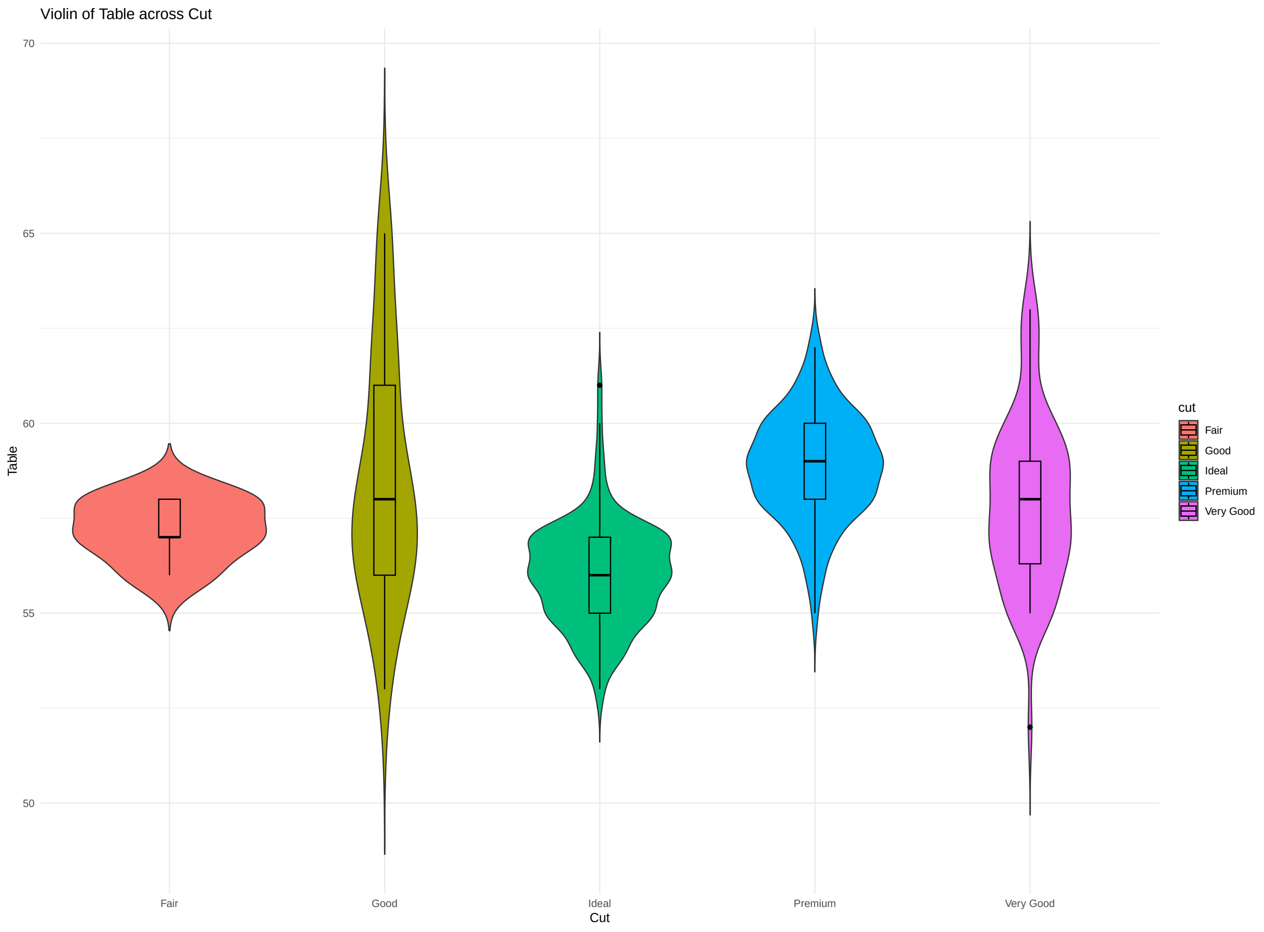
Creating a violin plot of the categorical variable cut against carat to examine if they are correlated, from this plot we see carat and cut aren't very correlated.

```r
diamonds_set %>% ggplot(aes(x = cut, y = depth, fill = cut)) +
  geom_violin(trim = FALSE) +
  geom_boxplot(width = 0.1, color = "black",
               position = position_dodge(0.9))+
  labs(title = "Violin of Depth across Cut",
       x = "Cut", y = "Depth") +
  theme_minimal()
```



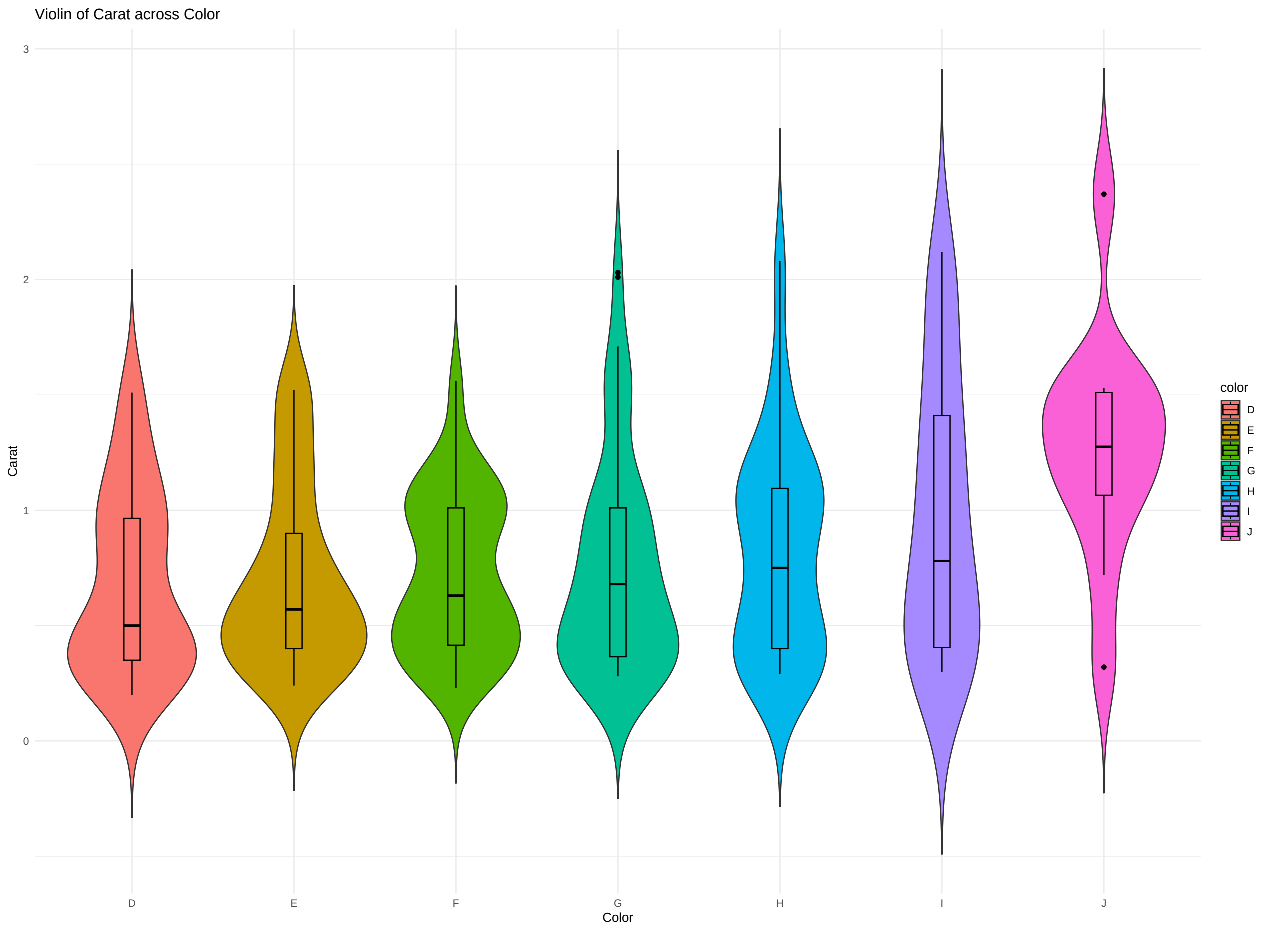
This violin plot is used to determine the correlation between depth and cut which according to this plot appear to be not very correlated.

```
diamonds_set %>% ggplot(aes(x = cut, y = table, fill = cut)) +
  geom_violin(trim = FALSE) +
  geom_boxplot(width = 0.1, color = "black",
               position = position_dodge(0.9))+
  labs(title = "Violin of Table across Cut",
       x = "Cut", y = "Table") +
  theme_minimal()
```



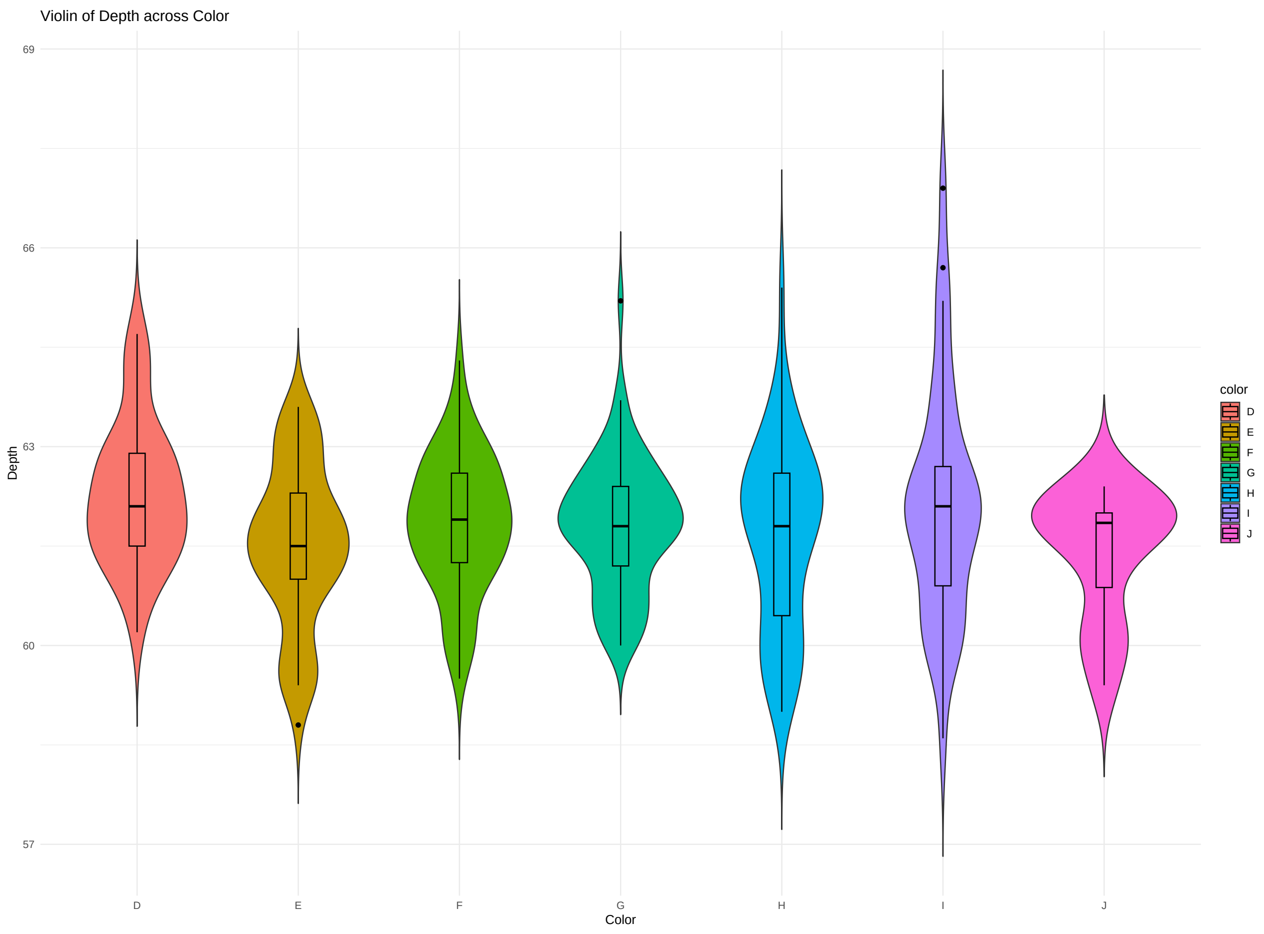
This violin plot shows the table values across the different cuts and shows the two aren't very correlated.

```
diamonds_set %>% ggplot(aes(x = color, y = carat, fill = color)) +
  geom_violin(trim = FALSE) +
  geom_boxplot(width = 0.1, color = "black",
               position = position_dodge(0.9))+
  labs(title = "Violin of Carat across Color",
       x = "Color", y = "Carat") +
  theme_minimal()
```



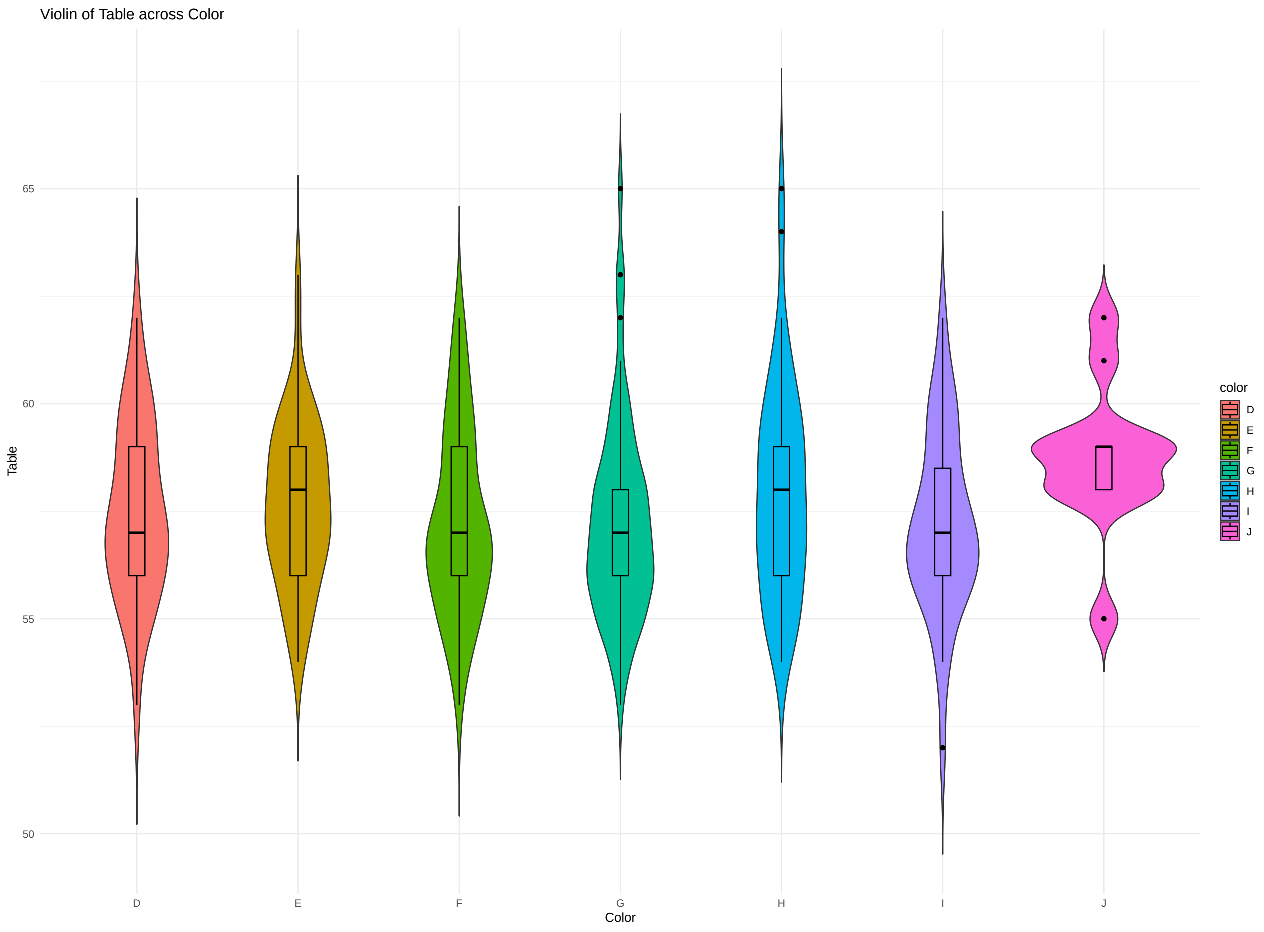
This plots my second chosen categorical variable color with carat and shows the two are not very correlated.

```
diamonds_set %>% ggplot(aes(x = color, y = depth, fill = color)) +
  geom_violin(trim = FALSE) +
  geom_boxplot(width = 0.1, color = "black",
               position = position_dodge(0.9))+
  labs(title = "Violin of Depth across Color",
       x = "Color", y = "Depth") +
  theme_minimal()
```



This plot shows depth being plotted across color and shows the two aren't very correlated.

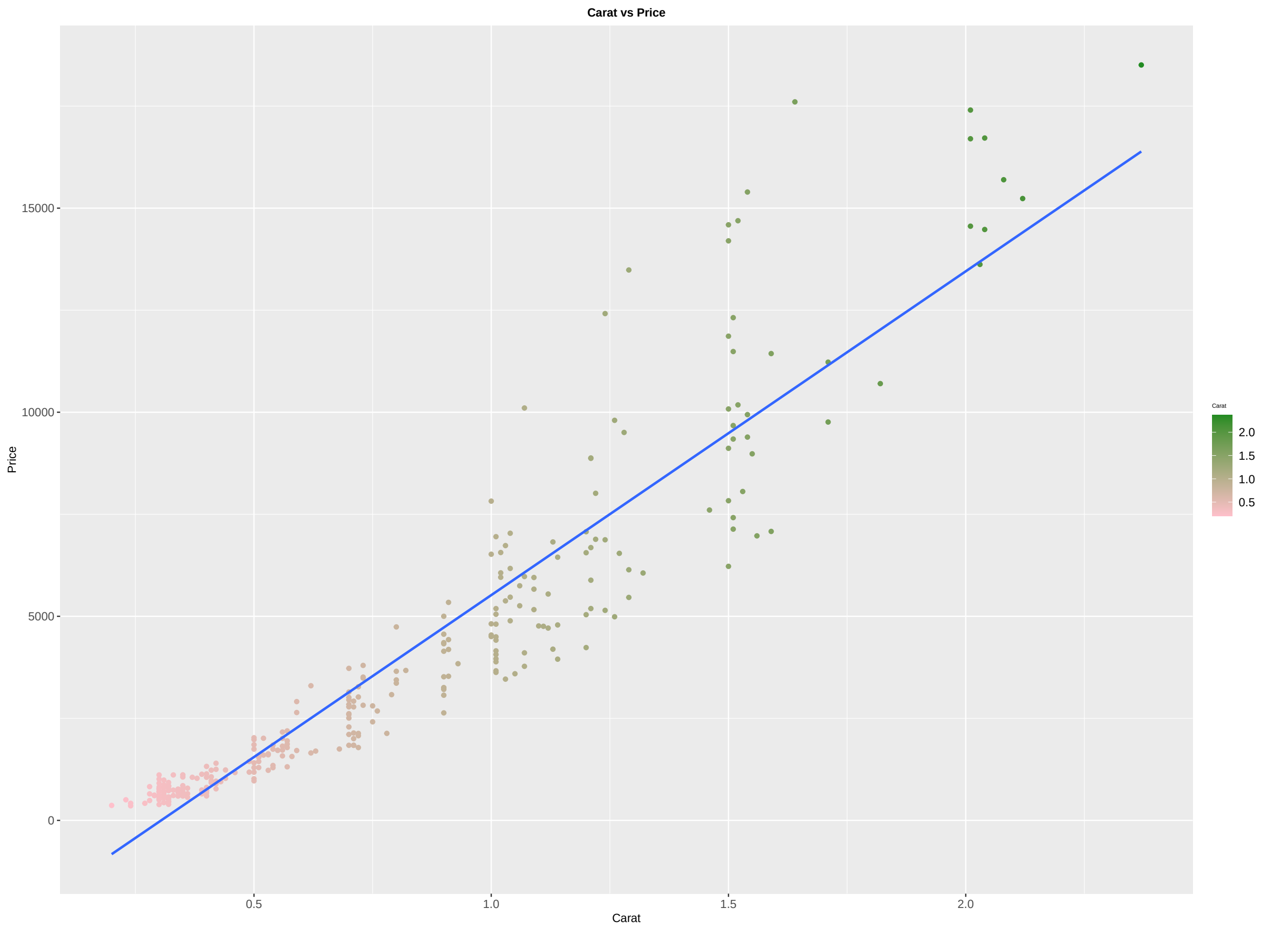
```
diamonds_set %>% ggplot(aes(x = color, y = table, fill = color)) +
  geom_violin(trim = FALSE) +
  geom_boxplot(width = 0.1, color = "black",
               position = position_dodge(0.9))+
  labs(title = "Violin of Table across Color",
       x = "Color", y = "Table") +
  theme_minimal()
```



This plot of table across color shows that the two values aren't very correlated.

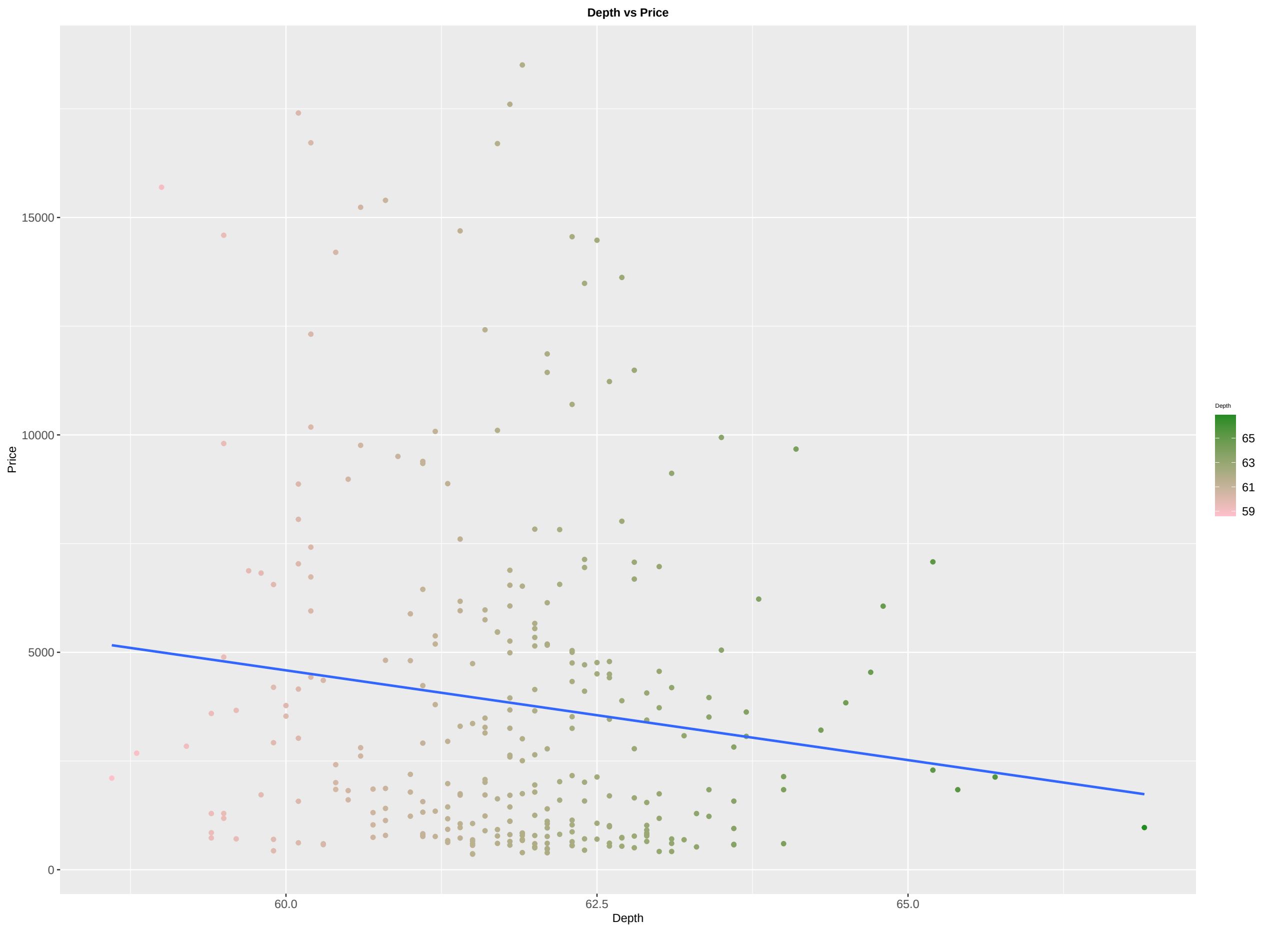
##Scatterplot of Independent variables vs Price

```
diamonds_set %>% ggplot() +
  geom_point(aes(x=carat,y= price, color=carat)) +
  labs(x="Carat",
       y = "Price",
       color = "Carat",
       title = "Carat vs Price")+
  scale_color_gradient(low = "pink", high = "forestgreen") +
  theme(plot.title = element_text(size = 10, face = "bold", hjust = 0.5),
    axis.title.x = element_text(size = 10),
    axis.title.y = element_text(size = 10),
    axis.text.x = element_text(size = 10),
    axis.text.y = element_text(size = 10),
    legend.title = element_text(size = 5),
    legend.text = element_text(size = 10)) +
  geom_smooth(method="lm", se=FALSE, aes(x=carat,y=price))
```



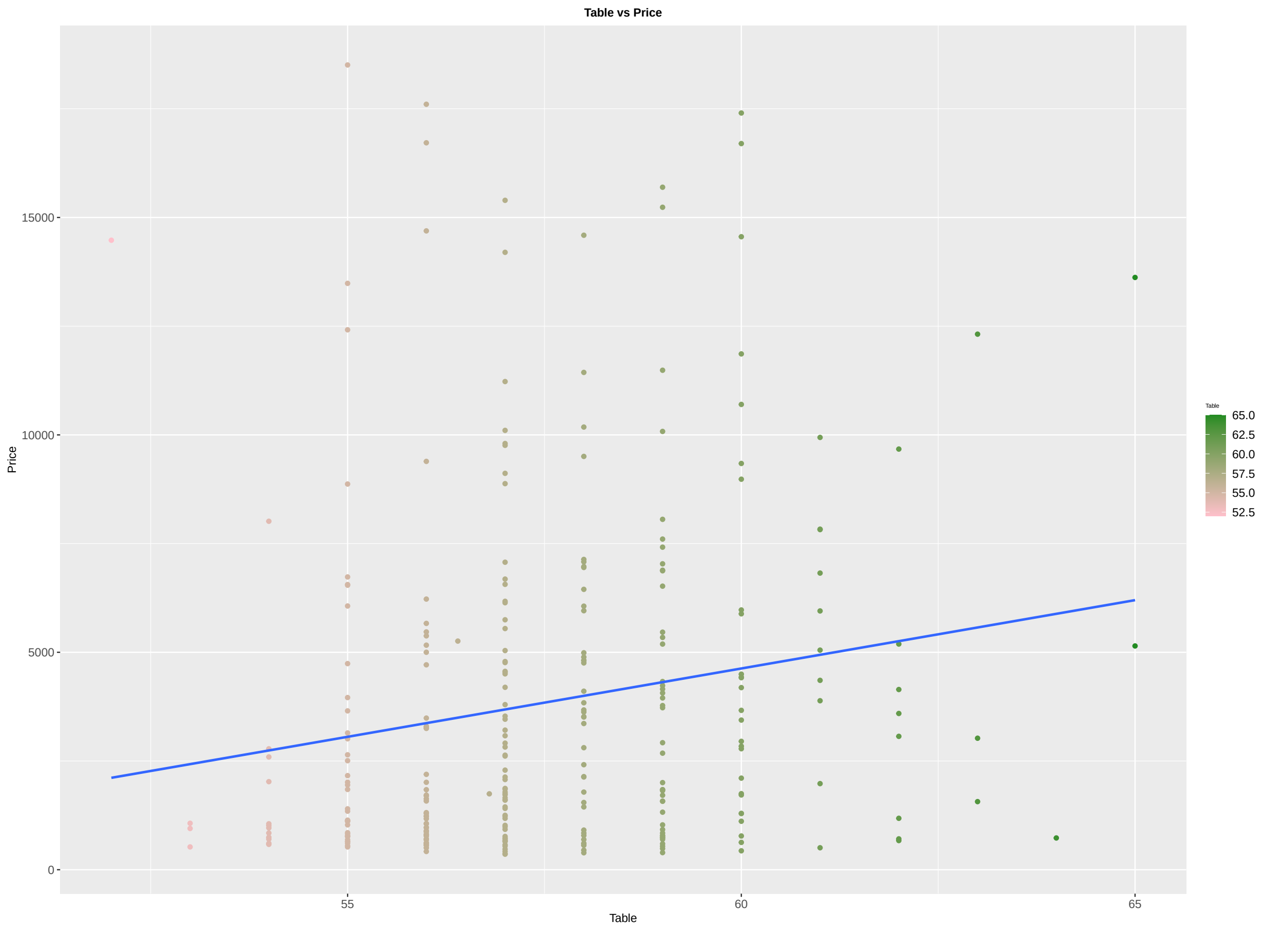
This plot shows strong correlation of my chosen dependent variable price and carat.

```
diamonds_set %>% ggplot() +
  geom_point(aes(x=depth,y=price, color=depth)) +
  labs(x="Depth",
       y = "Price",
       color = "Depth",
       title = "Depth vs Price")+
  scale_color_gradient(low = "pink", high = "forestgreen") +
  theme(plot.title = element_text(size = 10, face = "bold", hjust = 0.5),
    axis.title.x = element_text(size = 10),
    axis.title.y = element_text(size = 10),
    axis.text.x = element_text(size = 10),
    axis.text.y = element_text(size = 10),
    legend.title = element_text(size = 5),
    legend.text = element_text(size = 10)) +
  geom_smooth(method="lm", se=FALSE, aes(x=depth,y=price))
```



This plot shows the slight negative correlation between depth and price as my dependent variable.

```
diamonds_set %>% ggplot() +
  geom_point(aes(x=table,y=price, color=table)) +
  labs(x="Table",
       y = "Price",
       color = "Table",
       title = "Table vs Price")+
  scale_color_gradient(low = "pink", high = "forestgreen") +
  theme(plot.title = element_text(size = 10, face = "bold", hjust = 0.5),
    axis.title.x = element_text(size = 10),
    axis.title.y = element_text(size = 10),
    axis.text.x = element_text(size = 10),
    axis.text.y = element_text(size = 10),
    legend.title = element_text(size = 5),
    legend.text = element_text(size = 10)) +
  geom_smooth(method="lm", se=FALSE, aes(x=table,y=price))
```



This plot shows the relationship between my chosen independent and dependent variable of table and price.

##Multiple Linear Regression Model

```
linear_model <- lm(price ~ carat + depth + table + cut + color, data = diamonds_set)

summary(linear_model)
```

```
##
## Call:
## lm(formula = price ~ carat + depth + table + cut + color, data = diamonds_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3142.3  -660.9   -70.1   596.1  5893.7
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  11311.91    5677.10   1.993  0.04726 *
## carat         8359.53     172.51  48.459  < 2e-16 ***
## depth          -158.26      70.77  -2.236  0.02611 *
## table           -78.19      41.90  -1.866  0.06301 .
## cutGood         318.53     662.87   0.481  0.63121
## cutIdeal       1014.01     664.23   1.527  0.12797
## cutPremium      644.25     676.44   0.952  0.34169
## cutVery Good    526.01     659.41   0.798  0.42571
## colorE         -210.20     291.54  -0.721  0.47151
## colorF         -535.09     285.63  -1.873  0.06204 .
## colorG         -166.18     276.65  -0.601  0.54853
## colorH         -783.45     294.52  -2.660  0.00825 **
## colorI         -999.74     326.55  -3.062  0.00241 **
## colorJ        -2198.39     443.31  -4.959 1.22e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1256 on 286 degrees of freedom
## Multiple R-squared:  0.8994, Adjusted R-squared:  0.8948
## F-statistic: 196.7 on 13 and 286 DF,  p-value: < 2.2e-16
```

The above code runs a multiple linear regression model with the dependent variable price being dependent on the five independent variables, carat, depth, table, cut and color, then observes the summary statistics for all across the 300 samples.

##Expectation/Interesting Aspects of Data The data was mostly what I expected except the lack of correlation between cut and table of the diamonds. Upon researching the definitions of those two characteristics of diamonds, I expected them to be highly correlated, however the correlation matrix showed a relatively low correlation between these two values. The rest of the aspects of the data aligned with how I expected them. I believe my selected sample was representative of the population since I selected a high sample size of 300 observations and the data behaved as expected of the population.

##Identify Best-Fitting Model for Dataset

```
diamonds_set = diamonds_set[, c("price", "carat", "cut", "table", "depth", "color")]
diamonds_set
```

```
linear_model = lm(price~., data = diamonds_set) ##full regression model from above
lm_base = lm(price~1, data = diamonds_set) #base model for forward selection

fwd = stats::step(lm_base, direction = 'forward', scope=list(upper=formula(linear_model)
                      trace = FALSE)

bwd = stats::step(linear_model, direction = 'backward', trace = FALSE)

stw = stats::step(linear_model, direction = 'both', trace = FALSE)

fwd$call #best model identified from forward selection
```

```
## lm(formula = price ~ carat + color + cut + depth + table, data = diamonds_set)
```

```
bwd$call #best model identified from backward selection
```

```
## lm(formula = price ~ carat + cut + table + depth + color, data = diamonds_set)
```

```
stw$call #best model identified from step-wise selection
```

```
## lm(formula = price ~ carat + cut + table + depth + color, data = diamonds_set)
```

```
bwd_reg = lm(price ~ carat + color + cut + depth + table, data = diamonds_set)
```

The forward, backward, and step-wise selection all return the same linear model as being the best which also happens to be the full model for my chosen dependent and independent variables, therefore the best-fitting model is the full model which is price ~ carat + depth + table + cut + color.

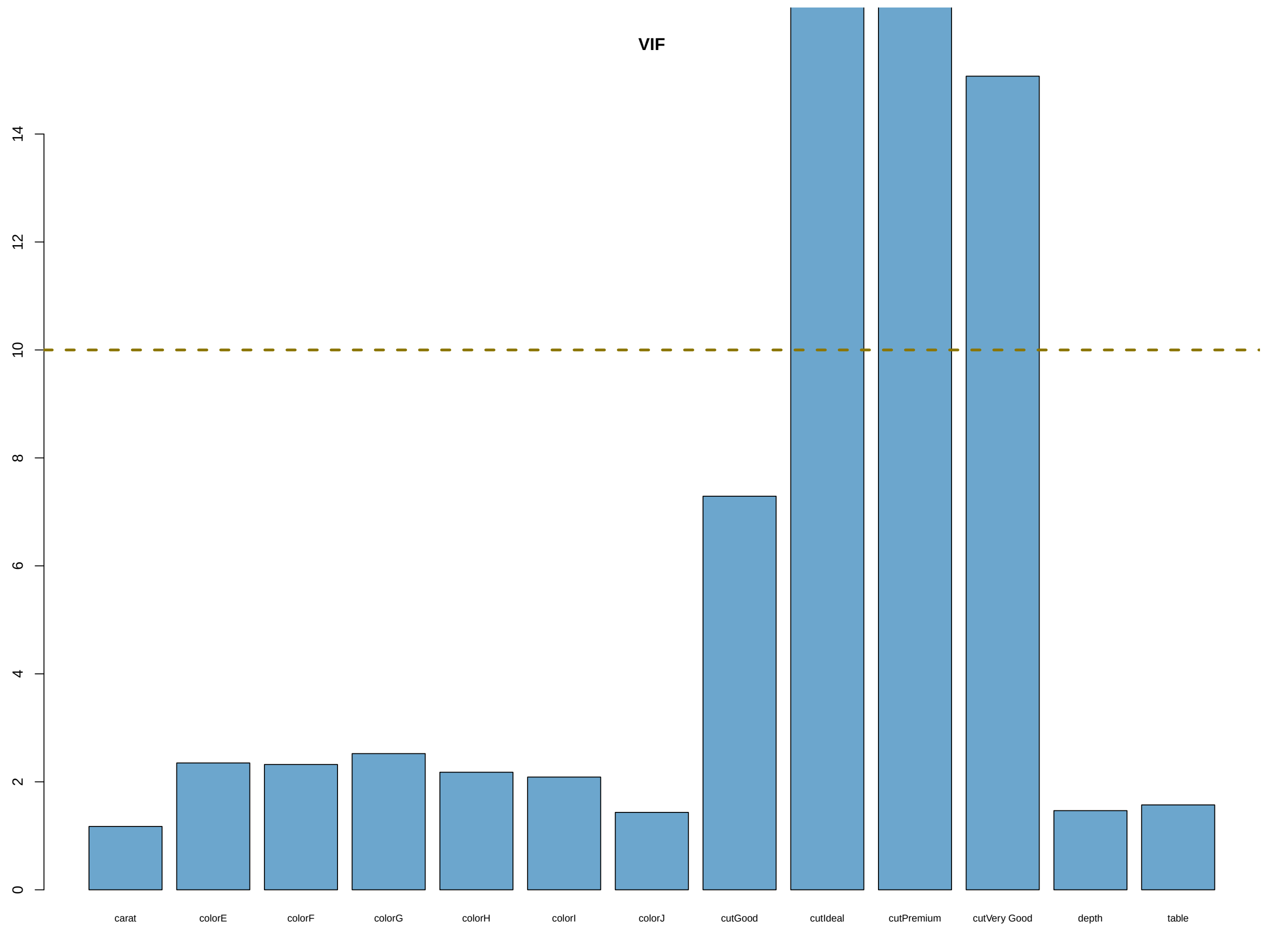
##Detecting Multicollinearity

```
library(car)
knitr::kable(sort(vif(bwd_reg)))
```

	x
carat	1.173093
colorJ	1.433988
depth	1.466323
table	1.572228
colorI	2.088141
colorH	2.177688
colorF	2.321096
colorE	2.349275
colorG	2.522658
cutGood	7.290856
cutVery Good	15.070848
cutPremium	17.003124
cutIdeal	19.751915

```
vif_vals_bwd = vif(bwd_reg)
barplot(
  vif_vals_bwd,
  main = "VIF",
  col = 'skyblue3',
  ylim = c(0, 15),
  cex.names = 0.7,
  width= 2
)
#vif_threshold1 = 4.0
```

```
vif_threshold2 = 10
#abline(h = vif_threshold1, lwd = 3, lty = 2, col = 'red4')
abline(h = vif_threshold2, lwd = 3, lty = 2, col = 'gold4')
```



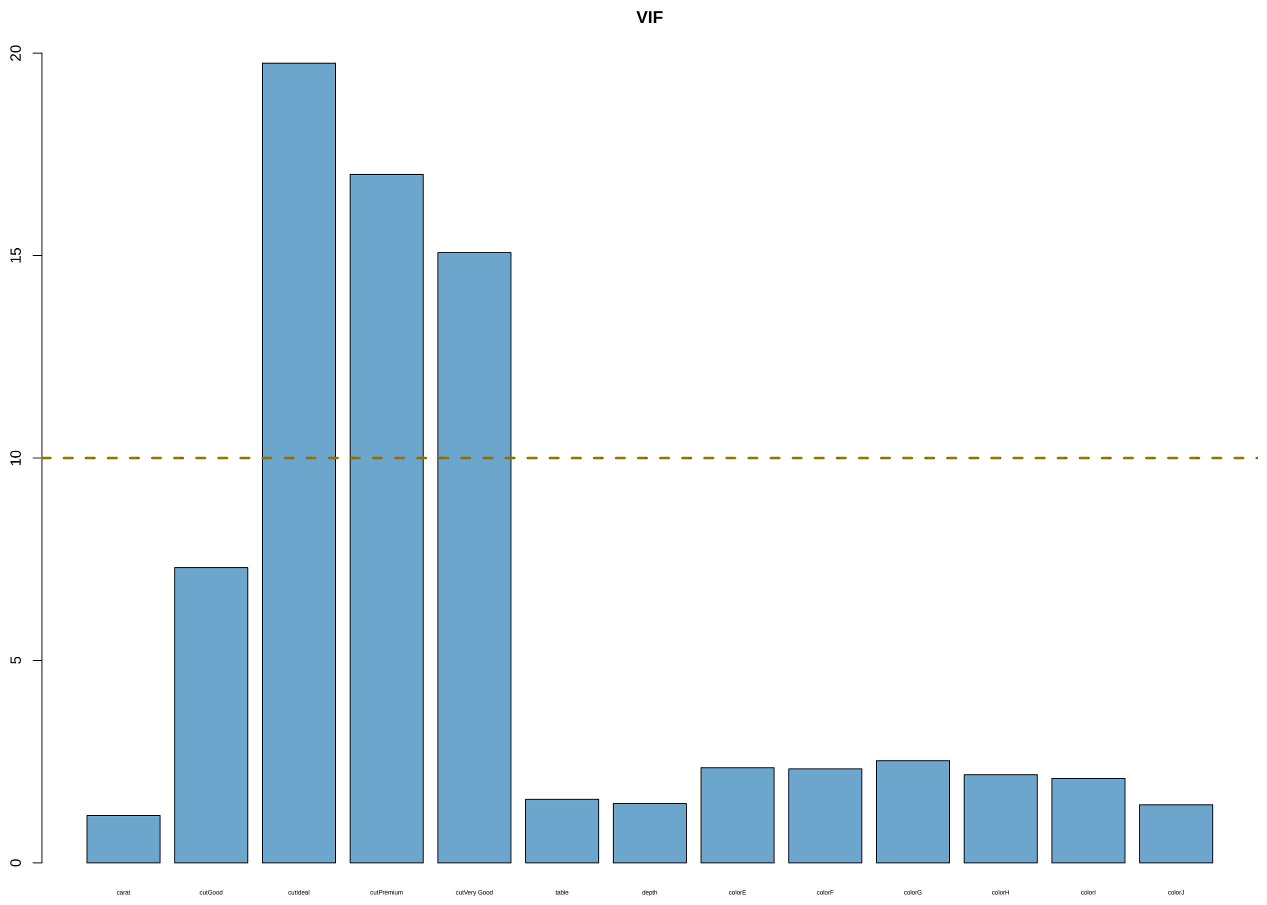
```
mreg.mod = lm(price ~., data=diamonds_set)
summary(mreg.mod)
```

```
##
## Call:
## lm(formula = price ~ ., data = diamonds_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3142.3  -660.9   -70.1   596.1  5893.7
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  11311.91    5677.10   1.993  0.04726 *
## carat         8359.53     172.51  48.459  < 2e-16 ***
## cutGood         318.53     662.87   0.481  0.63121
## cutIdeal       1014.01     664.23   1.527  0.12797
## cutPremium      644.25     676.44   0.952  0.34169
## cutVery Good    526.01     659.41   0.798  0.42571
## table           -78.19      41.90  -1.866  0.06301 .
## depth          -158.26      70.77  -2.236  0.02611 *
## colorE         -210.20     291.54  -0.721  0.47151
## colorF         -535.09     285.63  -1.873  0.06204 .
## colorG         -166.18     276.65  -0.601  0.54853
## colorH         -783.45     294.52  -2.660  0.00825 **
## colorI         -999.74     326.55  -3.062  0.00241 **
## colorJ        -2198.39     443.31  -4.959 1.22e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1256 on 286 degrees of freedom
## Multiple R-squared:  0.8994, Adjusted R-squared:  0.8948
## F-statistic: 196.7 on 13 and 286 DF,  p-value: < 2.2e-16
```

```r
knitr::kable(sort(vif(mreg.mod)))
```

	x
carat	1.173093
colorJ	1.433988
depth	1.466323
table	1.572228
colorI	2.088141
colorH	2.177688
colorF	2.321096
colorE	2.349275
colorG	2.522658
cutGood	7.290856
cutVery Good	15.070848
cutPremium	17.003124
cutIdeal	19.751915

```r
# We can also plot the results
vif_vals = vif(mreg.mod)
barplot(
  vif_vals,
  main = "VIF",
  col = 'skyblue3',
  ylim = c(0, 20),
  cex.names = 0.4,
  width= 2
)
#vif_threshold1 = 4.0
vif_threshold2 = 10
#abline(h = vif_threshold1, lwd = 3, lty = 2, col = 'red4')
abline(h = vif_threshold2, lwd = 3, lty = 2, col = 'gold4')
```



This VIF plot shows that certain categories in the factor cut can be removed from the model, such as cutIdeal, cutPremium, and cutVery Good since their VIF is greater than 10, so we have to delete these categories from the data frame and leave only cutGood which is under the threshold.

```
#add cut good to data frame
cut_good = ifelse(diamonds_set$cut == "Good", 1, 0)
diamonds_set$cutGood = cut_good
#drop original cut column from data frame
diamonds_set = dplyr::select(diamonds_set, -cut)
diamonds_set
```

The code above creates a new column and adds it to the data frame for when cut = good since this was the only value of cut that had a VIF under 10, and deletes the original cut column that contained the values of cut with VIF values greater than 10, which corrects the multicollinearity between the different cut values.

```r
new_lm = lm(price ~., data = diamonds_set)
summary(new_lm)
```

```
##
## Call:
## lm(formula = price ~ ., data = diamonds_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3205.5  -715.5   -98.1   601.6  6073.5
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 17677.01    4826.19   3.663 0.000297 ***
## carat        8344.76     173.41  48.123  < 2e-16 ***
## table        -126.65      36.35  -3.484 0.000570 ***
## depth        -203.28      63.89  -3.181 0.001625 **
## colorE       -289.08     290.05  -0.997 0.319770
## colorF       -548.29     286.73  -1.912 0.056836 .
## colorG       -177.33     277.22  -0.640 0.522886
## colorH       -800.10     296.03  -2.703 0.007285 **
## colorI      -1110.71     317.10  -3.503 0.000533 ***
## colorJ      -2287.98     444.53  -5.147  4.9e-07 ***
## cutGood      -344.13     264.72  -1.300 0.194649
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1265 on 289 degrees of freedom
## Multiple R-squared:  0.8969, Adjusted R-squared:  0.8933
## F-statistic: 251.4 on 10 and 289 DF,  p-value: < 2.2e-16
```

```r
knitr::kable(sort(vif(new_lm)))
```

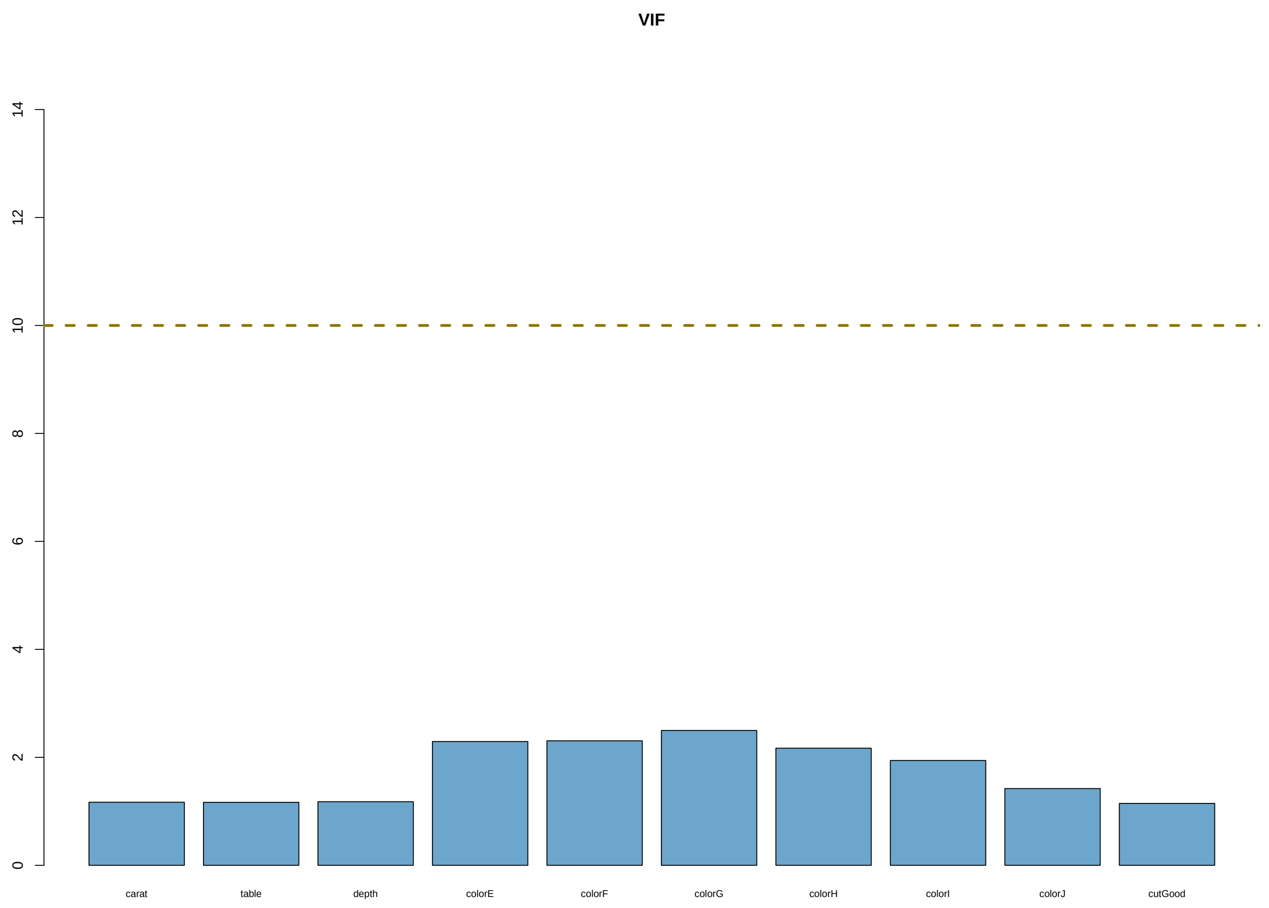
	x
cutGood	1.146565
table	1.166824
carat	1.168765
depth	1.178463
colorJ	1.421725
colorI	1.941536
colorH	2.169450
colorE	2.292793

	x
colorF	2.306353
colorG	2.497631

```r
vif_vals_bwd = vif(new_lm)
print(vif_vals_bwd)
```

```
##    carat    table    depth   colorE   colorF   colorG   colorH   colorI
## 1.168765 1.166824 1.178463 2.292793 2.306353 2.497631 2.169450 1.941535
##   colorJ  cutGood
## 1.421725 1.146565
```

```r
barplot(
  vif_vals_bwd,
  main = "VIF",
  col = 'skyblue3',
  ylim = c(0, 15),
  cex.names = 0.7,
  width= 2
)
#vif_threshold1 = 4.0
vif_threshold2 = 10
#abline(h = vif_threshold1, lwd = 3, lty = 2, col = 'red4')
abline(h = vif_threshold2, lwd = 3, lty = 2, col = 'gold4')
```



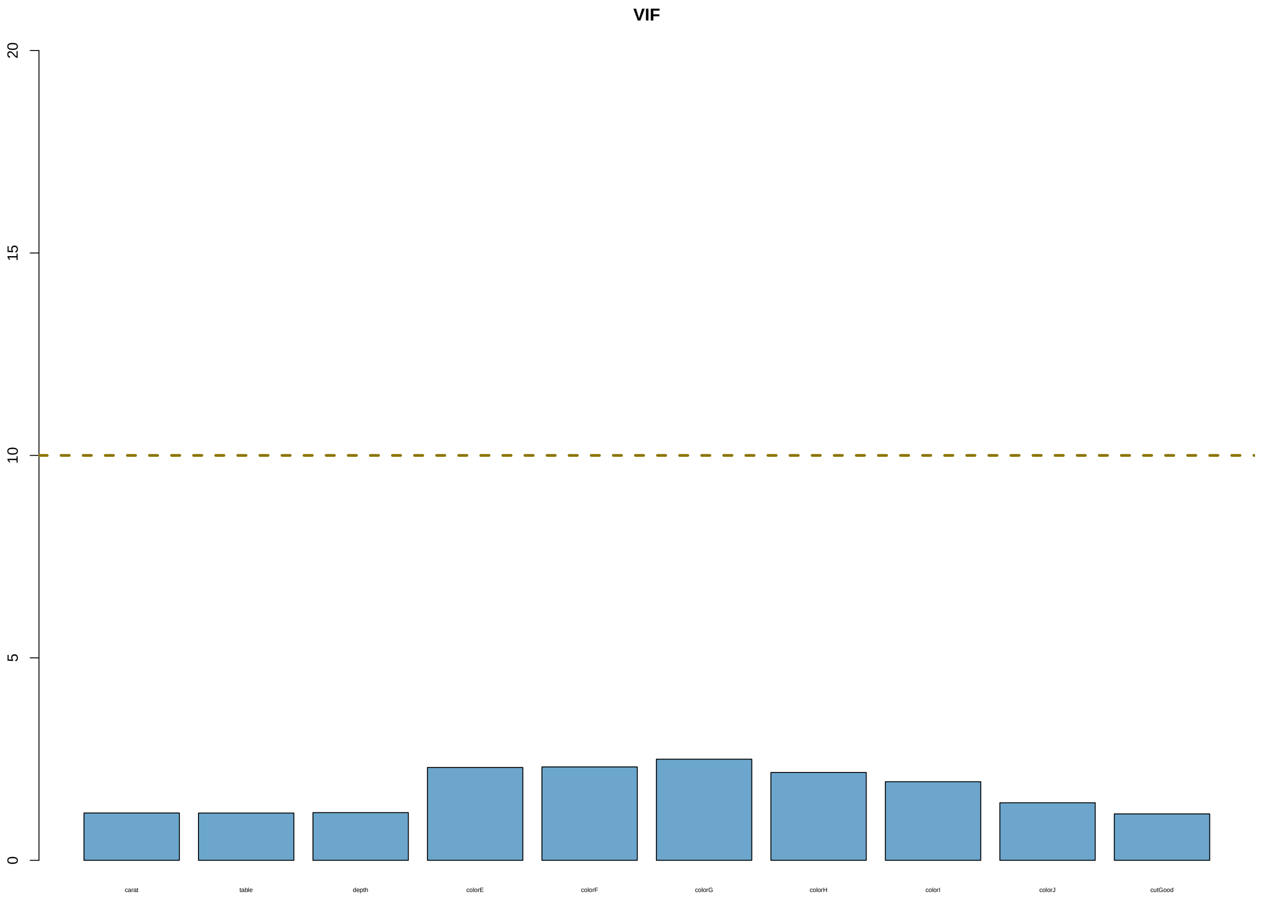
```r
mreg.mod = lm(price ~., data=diamonds_set)
summary(mreg.mod)
```

```
##
## Call:
## lm(formula = price ~ ., data = diamonds_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3205.5  -715.5   -98.1   601.6  6073.5
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 17677.01    4826.19   3.663 0.000297 ***
## carat        8344.76     173.41  48.123  < 2e-16 ***
## table        -126.65      36.35  -3.484 0.000570 ***
## depth        -203.28      63.89  -3.181 0.001625 **
## colorE       -289.08     290.05  -0.997 0.319770
## colorF       -548.29     286.73  -1.912 0.056836 .
## colorG       -177.33     277.22  -0.640 0.522886
## colorH       -800.10     296.03  -2.703 0.007285 **
## colorI      -1110.71     317.10  -3.503 0.000533 ***
## colorJ      -2287.98     444.53  -5.147  4.9e-07 ***
## cutGood      -344.13     264.72  -1.300 0.194649
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1265 on 289 degrees of freedom
## Multiple R-squared:  0.8969, Adjusted R-squared:  0.8933
## F-statistic: 251.4 on 10 and 289 DF,  p-value: < 2.2e-16
```

```
knitr::kable(sort(vif(mreg.mod)))
```

	x
cutGood	1.146565
table	1.166824
carat	1.168765
depth	1.178463
colorJ	1.421725
colorI	1.941536
colorH	2.169450
colorE	2.292793
colorF	2.306353
colorG	2.497631

```
# We can also plot the results
vif_vals = vif(mreg.mod)
barplot(
  vif_vals,
  main = "VIF",
  col = 'skyblue3',
  ylim = c(0, 20),
  cex.names = 0.4,
  width= 2
)
#vif_threshold1 = 4.0
vif_threshold2 = 10
#abline(h = vif_threshold1, lwd = 3, lty = 2, col = 'red4')
abline(h = vif_threshold2, lwd = 3, lty = 2, col = 'gold4')
```



The code above takes the new cutGood column and creates a regression model and checks the VIF for each predictor, and all the values are under 10, showing that deleting the original cut column and replacing it with cutGood corrects the issue of multicollinearity.