In [20]:

```
# Install necessary packages (if you haven't already)
# install.packages(c("tidyverse", "ggplot2", "dplyr"))

# Load libraries
library(tidyverse)
library(ggplot2)
```

In [ ]:

```
# Load the dataset
diamonds <- read_csv("diamonds4.csv")
```

In [3]:

```
# View first few rows
head(diamonds)
```

A tibble: 6 × 5

carat	clarity	color	cut	price
<dbl>	<chr>	<chr>	<chr>	<dbl>
0.51	SI2	I	Very Good	774
0.93	IF	H	Ideal	6246
0.50	VVS2	D	Very Good	1146
0.30	VS1	F	Ideal	538
0.31	SI1	F	Ideal	502
1.00	VS1	F	Ideal	7046

In [4]:

```
# Check the structure of the data
str(diamonds)
```

```
spc_tbl_ [1,214 × 5] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ carat  : num [1:1214] 0.51 0.93 0.5 0.3 0.31 1 1.03 1.31 0.5 0.24 ...
 $ clarity: chr [1:1214] "SI2" "IF" "VVS2" "VS1" ...
 $ color  : chr [1:1214] "I" "H" "D" "F" ...
 $ cut    : chr [1:1214] "Very Good" "Ideal" "Very Good" "Ideal" ...
 $ price  : num [1:1214] 774 6246 1146 538 502 ...
 - attr(*, "spec")=
  .. cols(
  ..   carat = col_double(),
  ..   clarity = col_character(),
  ..   color = col_character(),
  ..   cut = col_character(),
  ..   price = col_double()
  .. )
 - attr(*, "problems")=<externalptr>
```

In [5]:

```
# Summary statistics
summary(diamonds)
```

```
    carat          clarity             color               cut           
 Min.   :0.2300   Length:1214        Length:1214        Length:1214       
 1st Qu.:0.4000   Class :character   Class :character   Class :character  
 Median :0.5200   Mode  :character   Mode  :character   Mode  :character  
 Mean   :0.8134                                                           
 3rd Qu.:1.0000                                                           
 Max.   :7.0900                                                           
    price         
 Min.   :   322.0  
 1st Qu.:   723.5  
 Median :  1463.5  
 Mean   :  7056.7  
 3rd Qu.:  4640.8  
 Max.   :355403.0  
```

In [6]:

```
# Check for missing values
colSums(is.na(diamonds))
```

carat: 0 **clarity:** 0 **color:** 0 **cut:** 0 **price:** 0

In [7]:

```
# Convert categorical variables to factors
diamonds$clarity <- as.factor(diamonds$clarity)
diamonds$color <- as.factor(diamonds$color)
diamonds$cut <- as.factor(diamonds$cut)

#check the structure again.
str(diamonds)
```

```
spc_tbl_ [1,214 × 5] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ carat  : num [1:1214] 0.51 0.93 0.5 0.3 0.31 1 1.03 1.31 0.5 0.24 ...
 $ clarity: Factor w/ 8 levels "FL","IF","SI1",..: 4 2 8 5 3 5 5 8 3 3 ...
 $ color  : Factor w/ 7 levels "D","E","F","G",..: 6 5 1 3 3 3 4 4 1 4 ...
 $ cut    : Factor w/ 4 levels "Astor Ideal",..: 4 3 4 3 3 3 3 3 3 3 ...
 $ price  : num [1:1214] 774 6246 1146 538 502 ...
 - attr(*, "spec")=
  .. cols(
  ..   carat = col_double(),
  ..   clarity = col_character(),
  ..   color = col_character(),
  ..   cut = col_character(),
  ..   price = col_double()
  .. )
 - attr(*, "problems")=<externalptr>
```

```
    Price vs. Carat (Scatter Plot): This is crucial for your regression
    analysis.

    Price vs. Clarity (Box Plot): See how price varies across different
    clarity levels.

    Price vs. Color (Box Plot): Examine price variations based on
    color.

    Price vs. Cut (Box Plot): Investigate price differences across cut
```

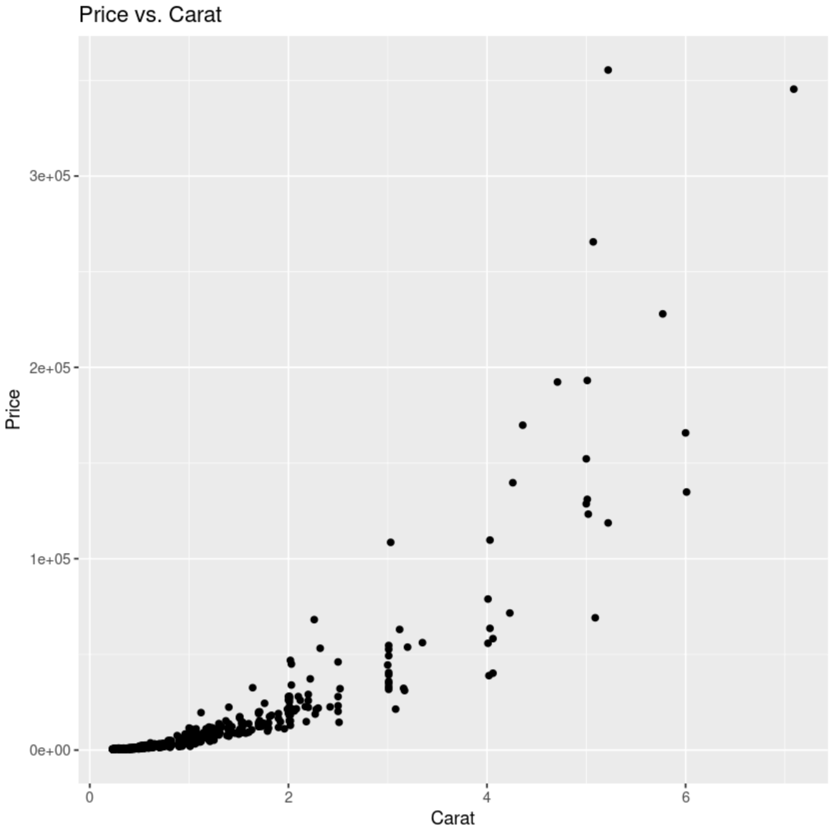
qualities.

Histograms of Price and Carat: See the distribution of these numerical variables.

Carat vs. Color, Carat vs. Clarity, Carat vs. Cut: Investigate these relationships as well.

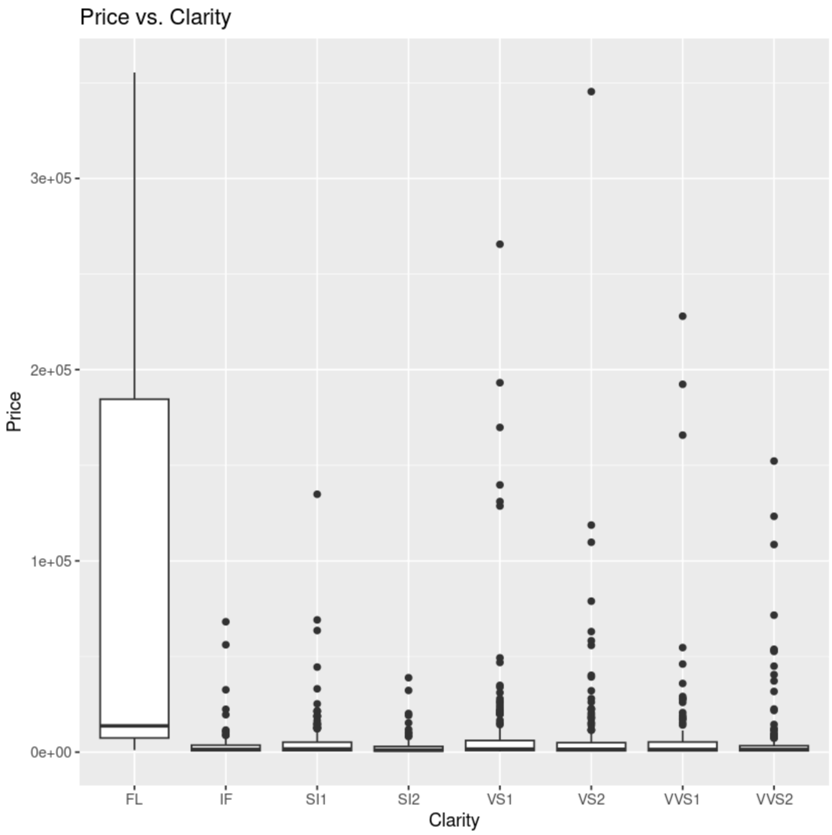
In [8]:

```
# Price vs. Carat (Scatter Plot)
ggplot(diamonds, aes(x = carat, y = price)) +
  geom_point() +
  labs(title = "Price vs. Carat", x = "Carat", y = "Price")
```

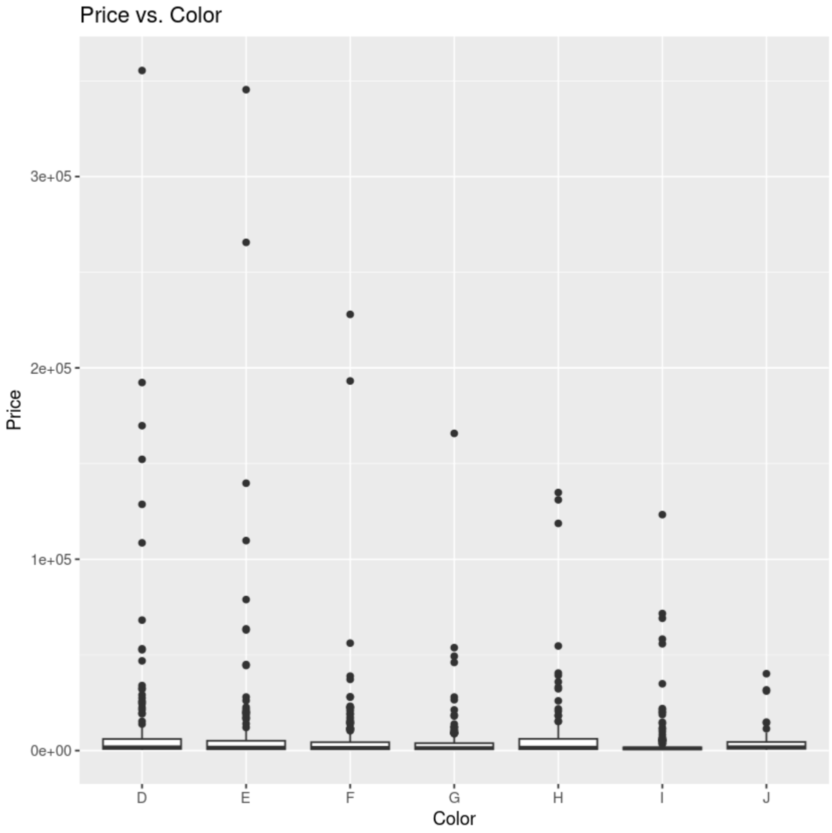


In [9]:

```
# Price vs. Clarity (Box Plot)
ggplot(diamonds, aes(x = clarity, y = price)) +
  geom_boxplot() +
  labs(title = "Price vs. Clarity", x = "Clarity", y = "Price")
```

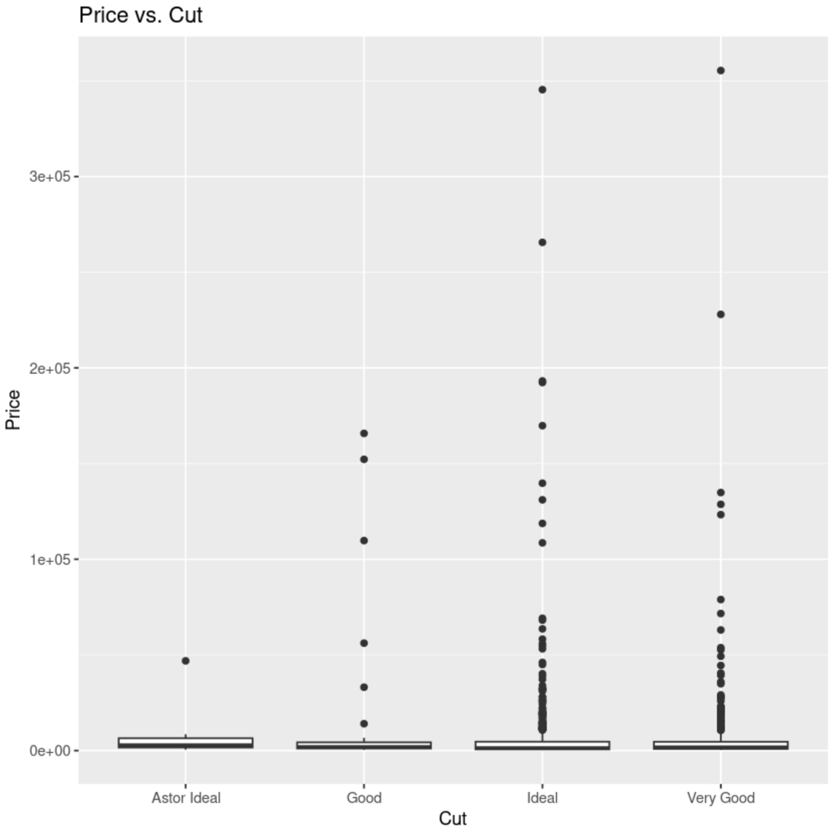


In [10]:
```r
# Price vs. Color (Box Plot)
ggplot(diamonds, aes(x = color, y = price)) +
  geom_boxplot() +
  labs(title = "Price vs. Color", x = "Color", y = "Price")
```



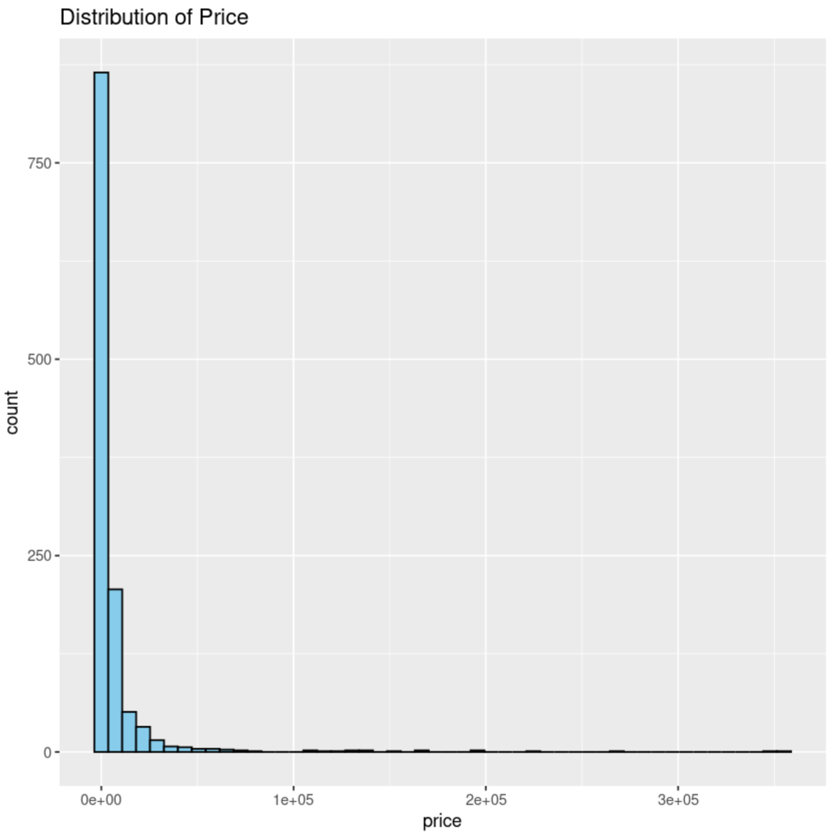
In [11]:

```
# Price vs. Cut (Box Plot)
ggplot(diamonds, aes(x = cut, y = price)) +
  geom_boxplot() +
  labs(title = "Price vs. Cut", x = "Cut", y = "Price")
```



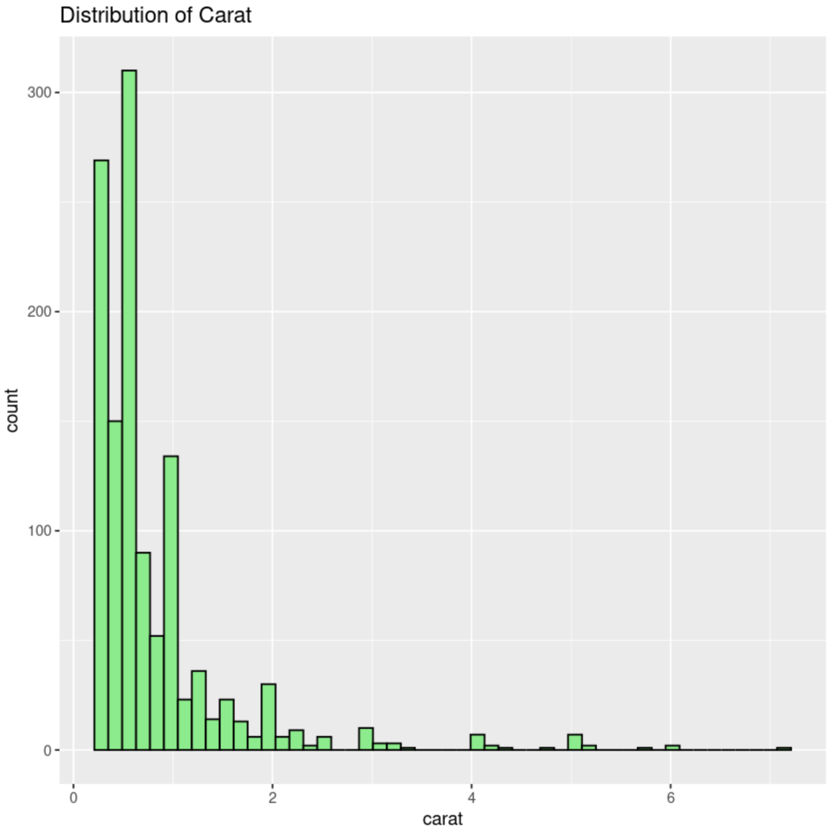
In [12]:

```
# Histogram of Price
ggplot(diamonds, aes(x = price)) +
  geom_histogram(bins = 50, fill = "skyblue", color = "black") +
  labs(title = "Distribution of Price")
```



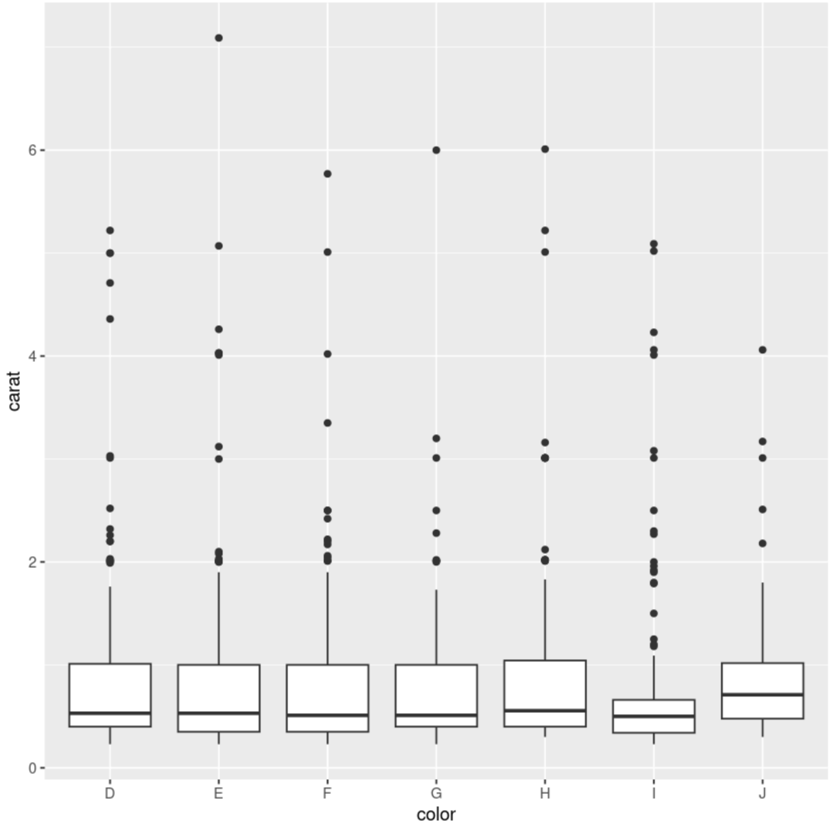
In [13]:

```r
# Histogram of Carat
ggplot(diamonds, aes(x = carat)) +
  geom_histogram(bins = 50, fill = "lightgreen", color = "black") +
  labs(title = "Distribution of Carat")
```



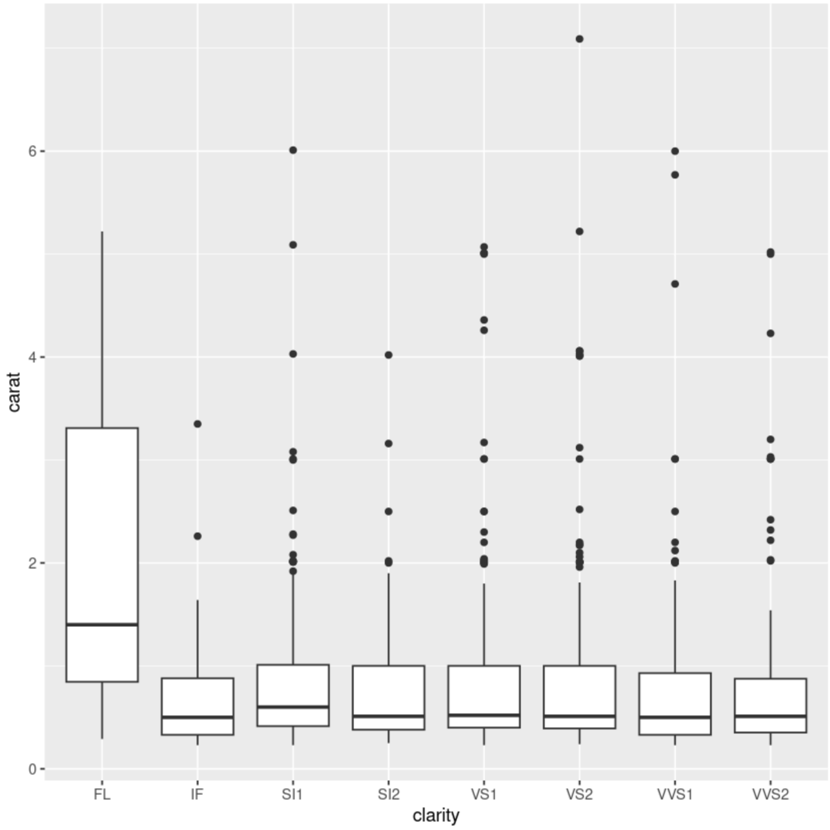
In [14]:

```
# Carat vs Color boxplot
ggplot(diamonds, aes(x=color, y=carat))+ geom_boxplot()
```



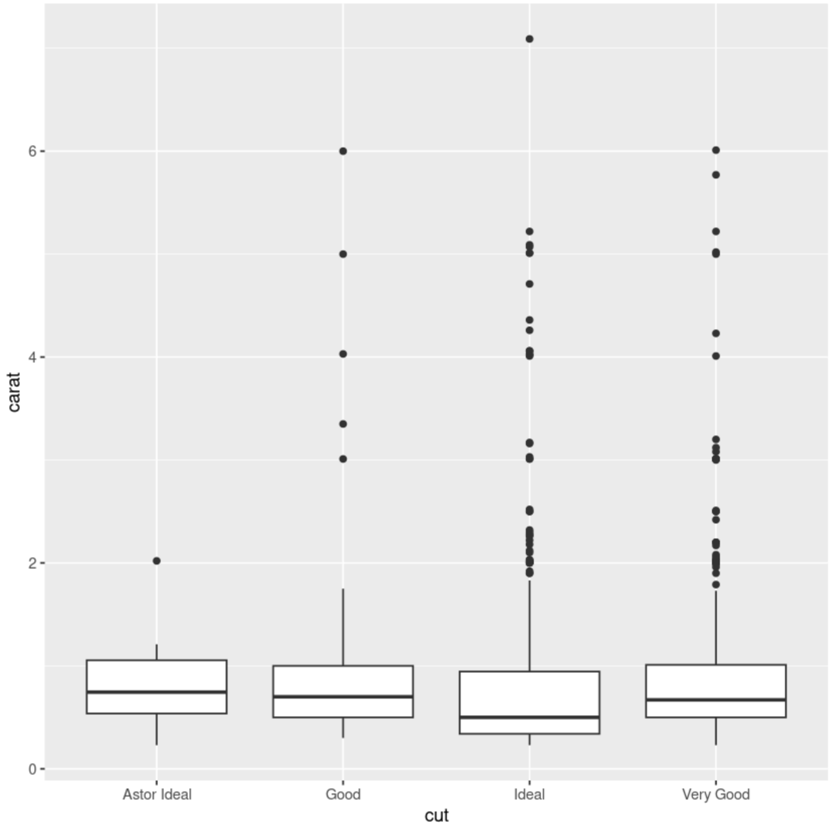
In [15]:

```
#Carat vs clarity boxplot
ggplot(diamonds, aes(x=clarity, y=carat))+ geom_boxplot()
```



In [16]:

```
#Carat vs cut boxplot
ggplot(diamonds, aes(x=cut, y=carat))+ geom_boxplot()
```



Simple Linear Regression (Price vs. Carat)

Fit the Model: Use the lm() function to fit a linear regression model.

Check Assumptions: Use diagnostic plots to assess the linearity, normality of residuals, equal variance, and independence assumptions.

Interpret the Results: Explain the meaning of the coefficients and the R-squared value.

In [17]:

```r
# Fit the linear regression model
model <- lm(price ~ carat, data = diamonds)
```

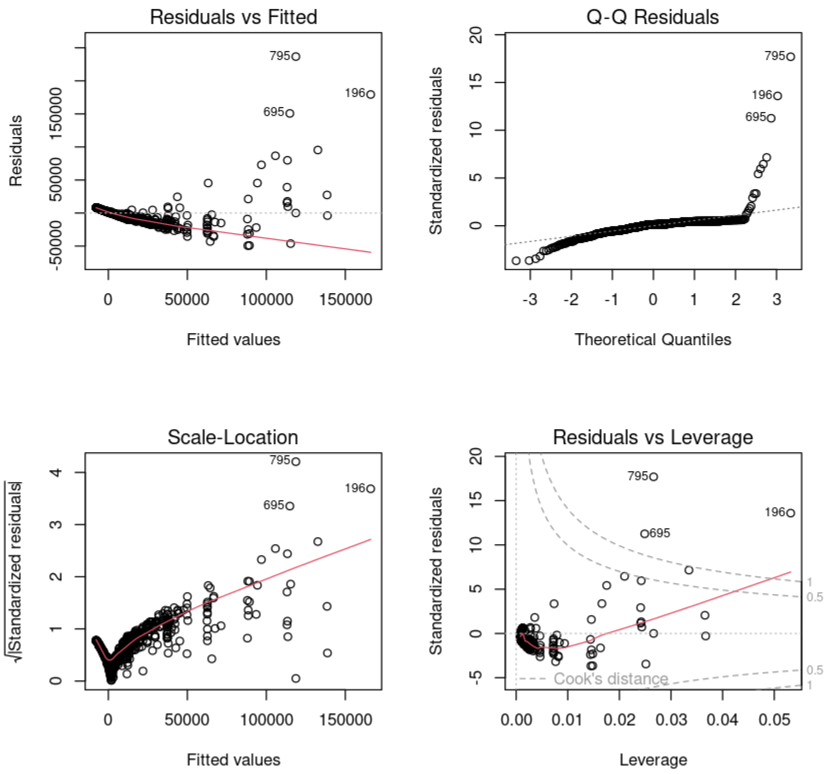
In [18]:

```r
# Summary of the model
summary(model)
```

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

Residuals:
   Min      1Q Median      3Q     Max 
-49375  -5048   1867   4965 236711 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -13550.9      559.7  -24.21   <2e-16 ***
carat        25333.9      494.4   51.24   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 13560 on 1212 degrees of freedom
Multiple R-squared:  0.6842,	Adjusted R-squared:  0.6839 
F-statistic:  2625 on 1 and 1212 DF,  p-value: < 2.2e-16
```

In [19]:

```r
# Diagnostic plots
par(mfrow = c(2, 2)) # Arrange plots in a 2x2 grid
plot(model)
par(mfrow=c(1,1)) #set back to one plot.
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: