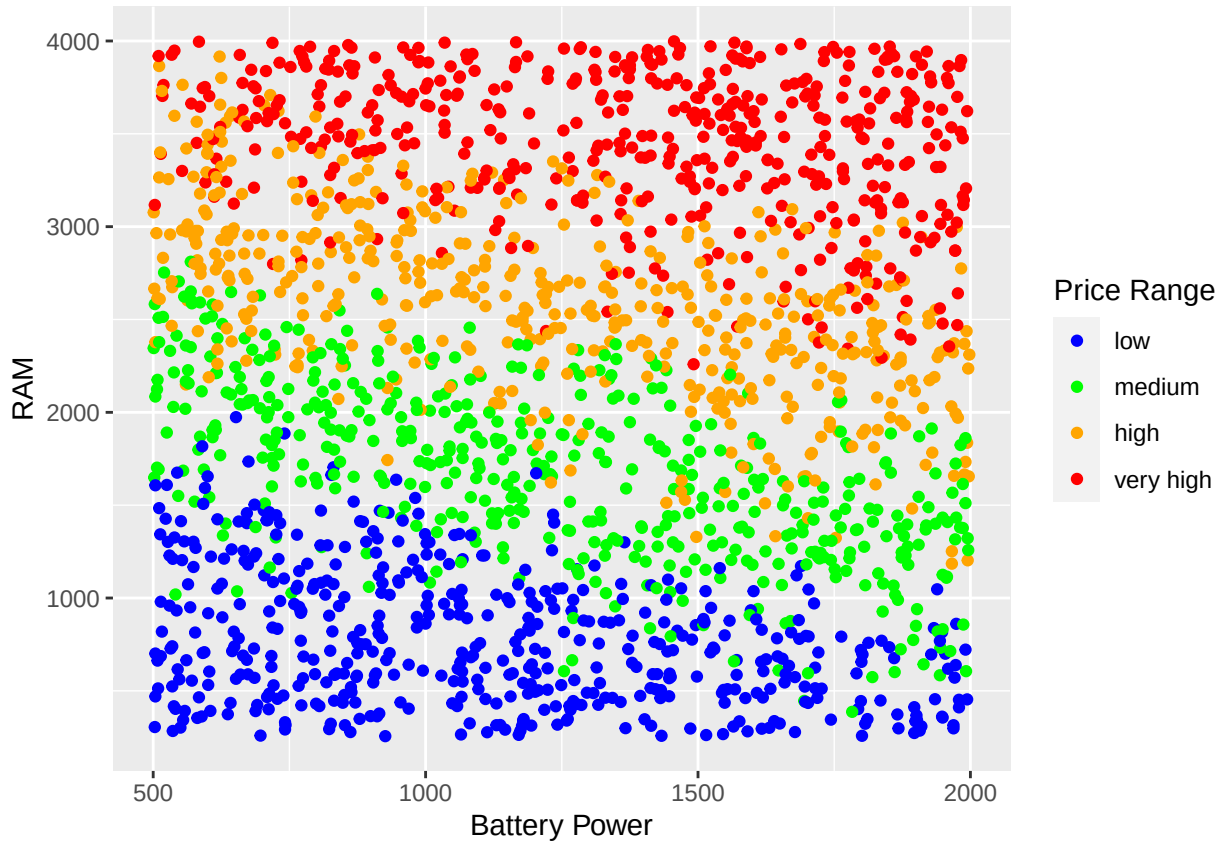
Homework 4

Adyan Rahman

Problem 1 (a)

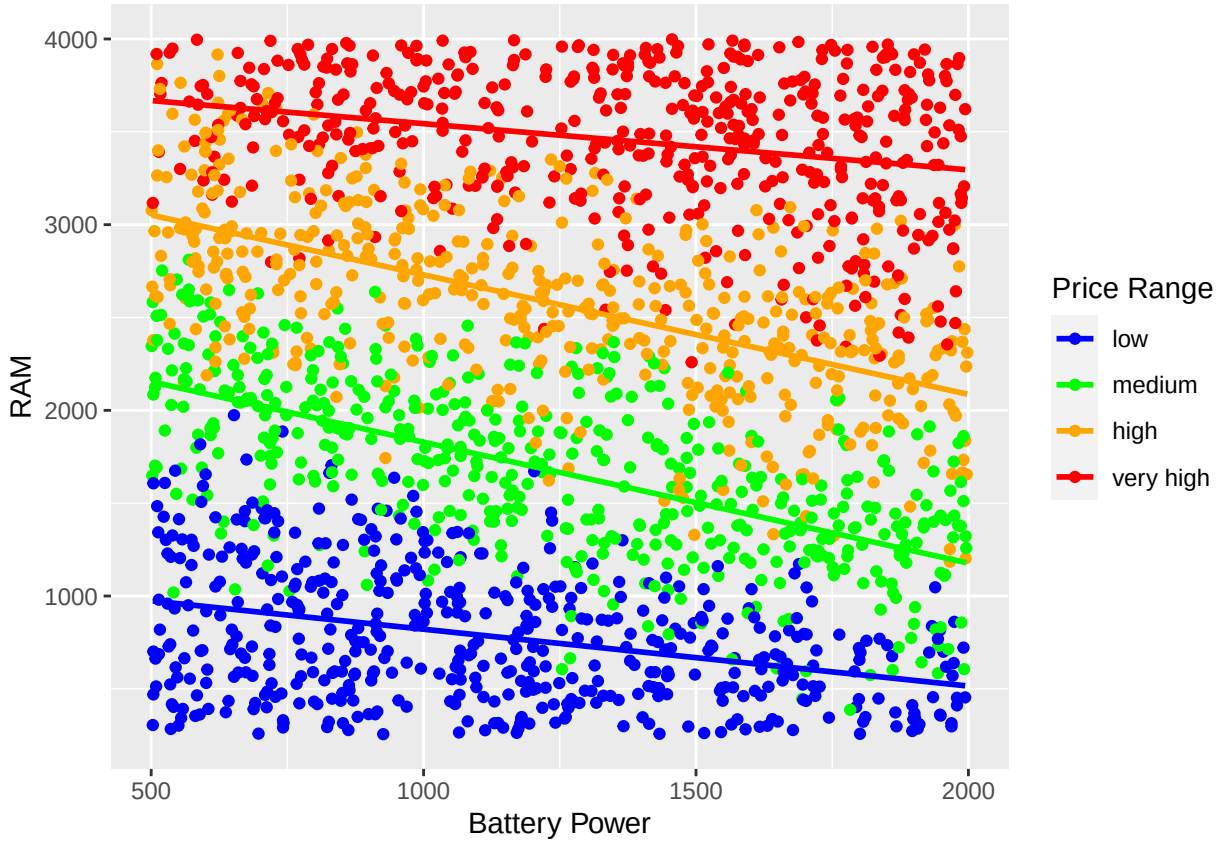
```
# Opening the data set
setwd("C:\\Users\\ktzr1\\OneDrive\\Desktop\\STAT3355 Datasets")

mobile_data <- read.csv("train.csv")

# Convert "price_range" to a factor with specified levels
mobile_data$price_range <- factor(mobile_data$price_range,
                                  levels = c("0", "1", "2", "3"),
                                  labels = c("low", "medium", "high", "very high"))

# Load the ggplot2 package
library(ggplot2)

# Assuming you have a dataframe named 'mobile_data' containing the required variables

# Create the scatter plot
ggplot(mobile_data, aes(x = battery_power, y = ram, color = factor(price_range))) +
  geom_point() +
  labs(x = "Battery Power", y = "RAM", color = "Price Range") +
  scale_color_manual(values = c("low" = "blue", "medium" = "green", "high" = "orange",
                                "very high" = "red"))
```



(b)

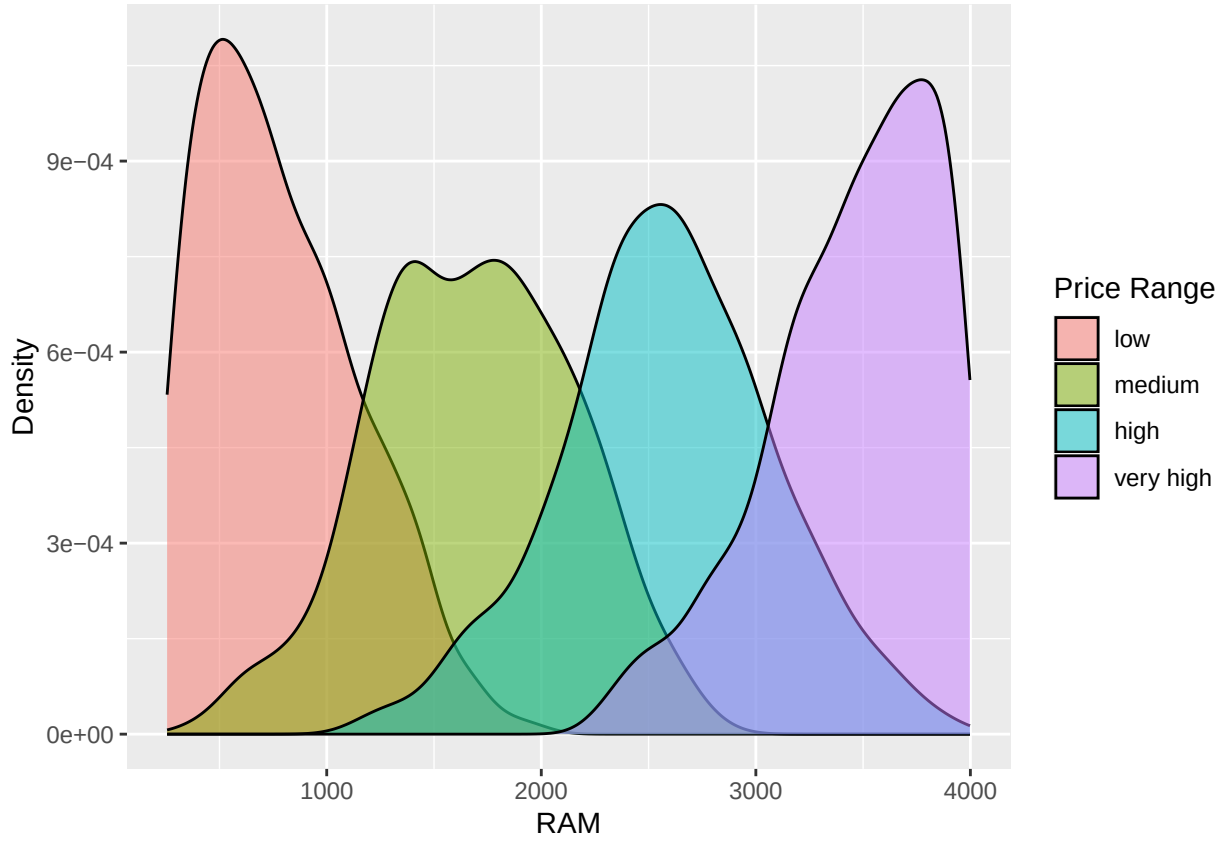
```
#Recreate the plot from part a
ggplot(mobile_data, aes(x = battery_power, y = ram, color = factor(price_range))) +
  geom_point() +
  labs(x = "Battery Power", y = "RAM", color = "Price Range") +
  scale_color_manual(values = c("low" = "blue", "medium" = "green", "high" = "orange",
                                "very high" = "red")) +
# Add trend lines for each price range separately
  geom_smooth(method = "lm", se = FALSE)
```

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



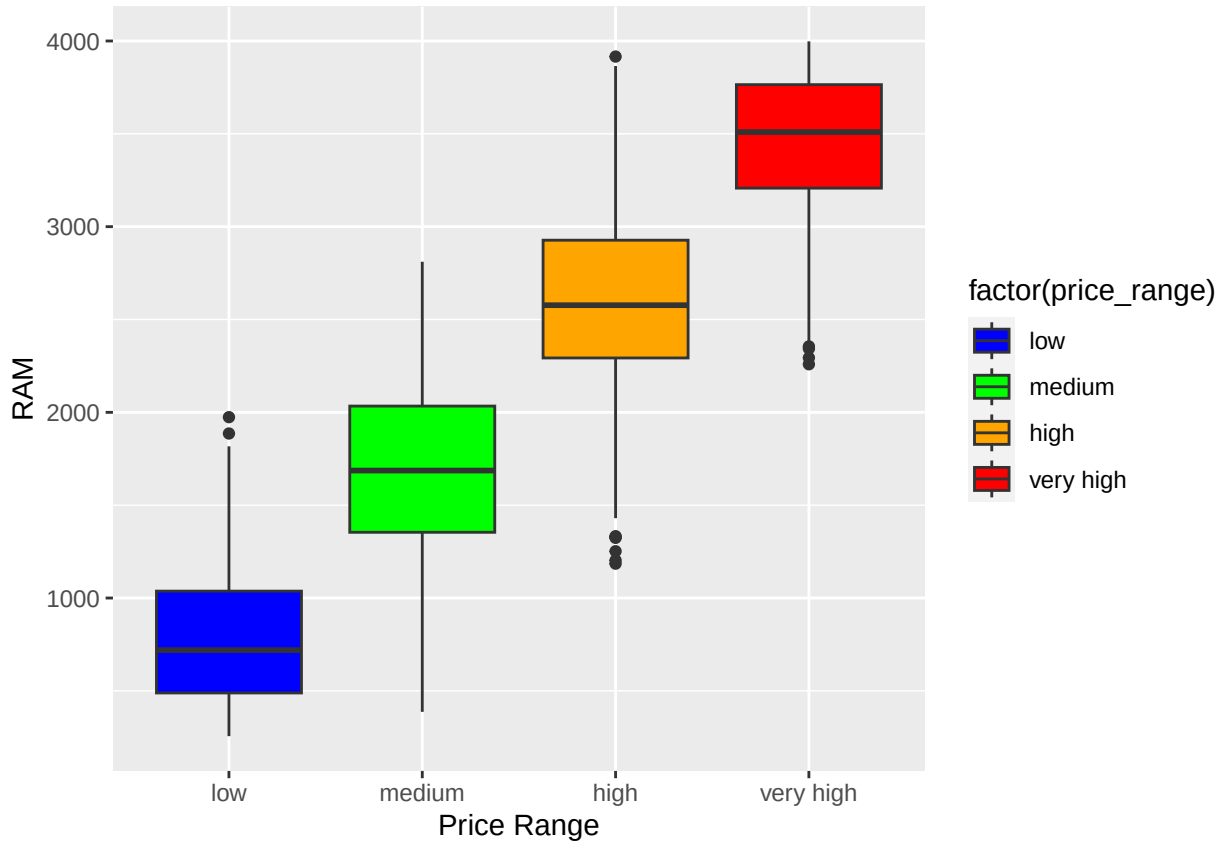
(c)

```r
# Create density curves of RAM for each price range in one plot
ggplot(mobile_data, aes(x = ram, fill = factor(price_range))) +
  geom_density(alpha = 0.5) +
  labs(x = "RAM", y = "Density", fill = "Price Range")
```



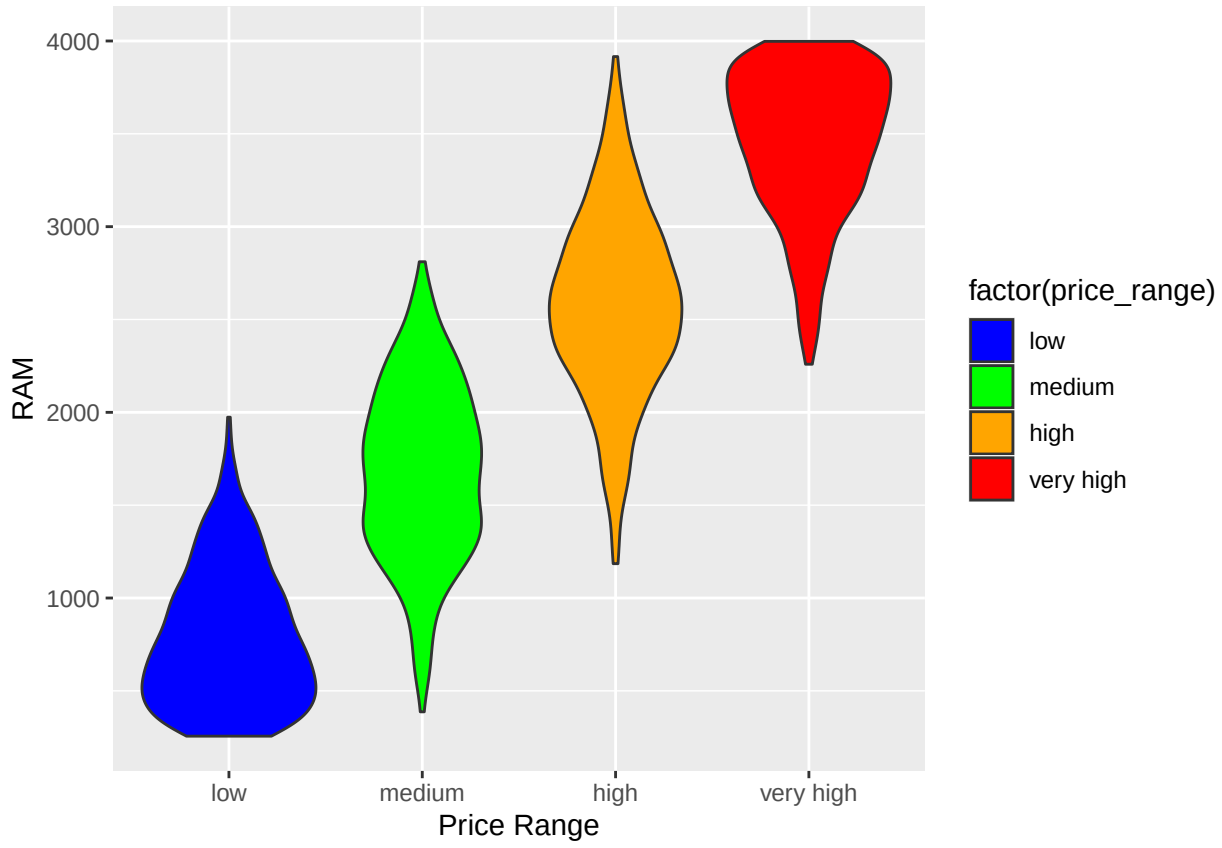
(d)

```r
# Create box plots of RAM for each price range in one plot
ggplot(mobile_data, aes(x = factor(price_range), y = ram, fill = factor(price_range))) +
  geom_boxplot() +
  labs(x = "Price Range", y = "RAM") +
  scale_fill_manual(values = c("blue", "green", "orange", "red"))
```



(e)

```r
# Create a violin plot of RAM for each price range in one plot
ggplot(mobile_data, aes(x = factor(price_range), y = ram, fill = factor(price_range))) +
  geom_violin() +
  labs(x = "Price Range", y = "RAM") +
  scale_fill_manual(values = c("blue", "green", "orange", "red"))
```

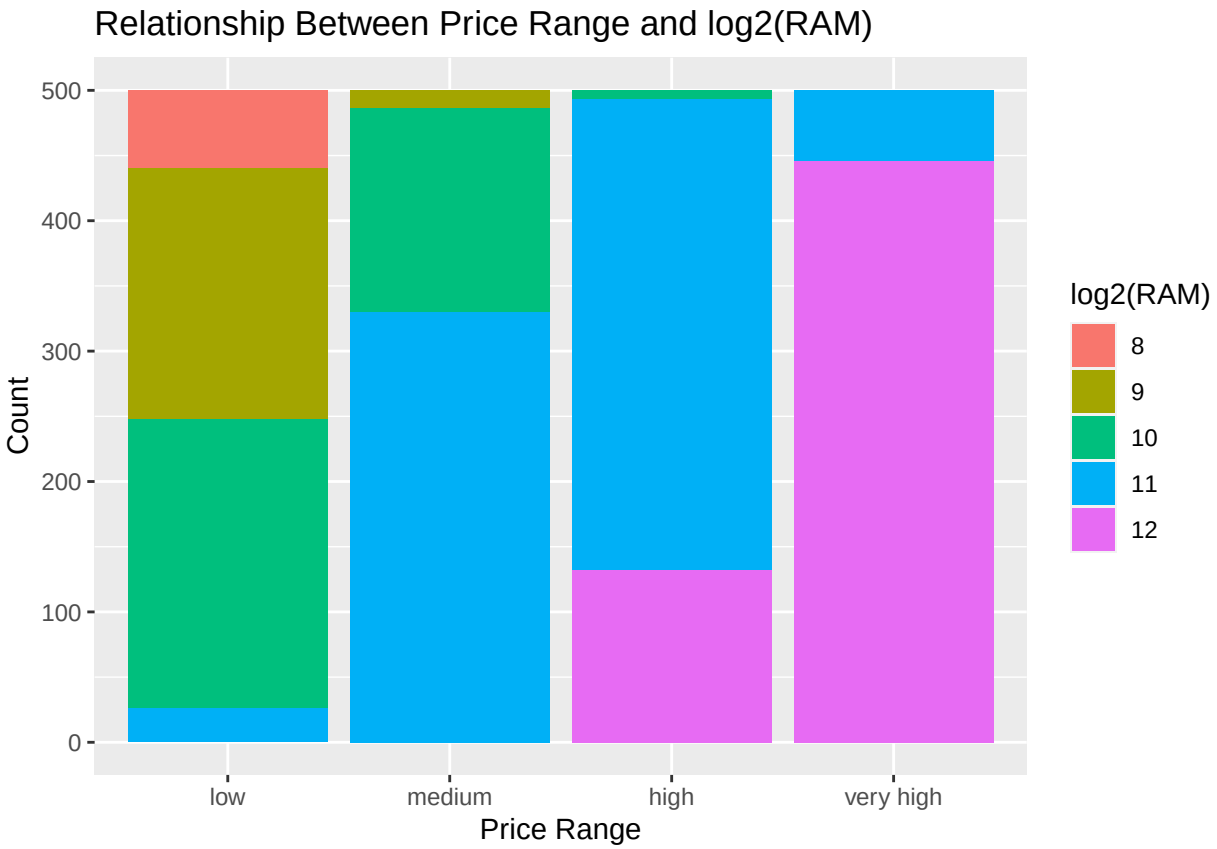


(f)

```r
ram_log <- as.factor(round(log2(mobile_data$ram)))

# Print the unique values of the factor variable
print(unique(ram_log))
```

```
## [1] 11 10 12 9  8
## Levels: 8 9 10 11 12
```

```r
# Create the stacked bar plot
ggplot(mobile_data, aes(x = price_range, fill = ram_log)) +
geom_bar(position = "stack") +
labs(title = "Relationship Between Price Range and log2(RAM)",
x = "Price Range",
y = "Count") +
scale_fill_discrete(name = "log2(RAM)")
```



Problem 2 (a)

```
# Install and load the "UsingR" package
library(UsingR)
```

```
## Warning: package 'UsingR' was built under R version 4.3.3

## Loading required package: MASS

## Loading required package: HistData

## Warning: package 'HistData' was built under R version 4.3.3

## Loading required package: Hmisc

## Warning: package 'Hmisc' was built under R version 4.3.3

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```r
# Access the "UScereal" dataset
data("UScereal")

# Replace the levels of the "mfr" variable with full names
UScereal$mfr <- factor(UScereal$mfr, levels = c("G", "K", "N", "P", "Q", "R"),
                       labels = c("General Mills", "Kelloggs", "Nabisco", "Post", "Quaker Oats", "Ralsto

# View the unique levels of the "mfr" variable to verify the change
unique(UScereal$mfr)
```

```
## [1] Nabisco        Kelloggs       General Mills  Ralston Purina Post
## [6] Quaker Oats
## Levels: General Mills Kelloggs Nabisco Post Quaker Oats Ralston Purina
```

(b)

```r
# Convert the "shelf" variable to a factor variable with specified levels and labels
UScereal$shelf <- factor(UScereal$shelf, levels = c(1, 2, 3),
                         labels = c("low", "middle", "upper"))

# View the unique levels of the "shelf" variable to verify the change
unique(UScereal$shelf)
```

```
## [1] upper  low    middle
## Levels: low middle upper
```

(c)

```r
# Create a new variable named "Product" for the product name
UScereal$Product <- paste(UScereal$name, UScereal$mfr, sep = " - ")

# View the first few rows of the dataset to verify the new variable
head(UScereal)
```

```
##                                    mfr calories   protein      fat   sodium
## 100% Bran                      Nabisco 212.1212 12.121212 3.030303 393.9394
## All-Bran                      Kelloggs 212.1212 12.121212 3.030303 787.8788
## All-Bran with Extra Fiber     Kelloggs 100.0000  8.000000 0.000000 280.0000
## Apple Cinnamon Cheerios   General Mills 146.6667  2.666667 2.666667 240.0000
## Apple Jacks                   Kelloggs 110.0000  2.000000 0.000000 125.0000
## Basic 4                 General Mills 173.3333  4.000000 2.666667 280.0000
##                             fibre    carbo   sugars  shelf potassium vitamins
## 100% Bran                30.303030 15.15152 18.18182  upper 848.48485 enriched
## All-Bran                 27.272727 21.21212 15.15151  upper 969.69697 enriched
## All-Bran with Extra Fiber 28.000000 16.00000  0.00000  upper 660.00000 enriched
## Apple Cinnamon Cheerios    2.000000 14.00000 13.33333    low  93.33333 enriched
## Apple Jacks                1.000000 11.00000 14.00000 middle  30.00000 enriched
## Basic 4                    2.666667 24.00000 10.66667  upper 133.33333 enriched
##                                    Product
## 100% Bran                      - Nabisco
## All-Bran                      - Kelloggs
```

```
## All-Bran with Extra Fiber       - Kelloggs
## Apple Cinnamon Cheerios    - General Mills
## Apple Jacks                     - Kelloggs
## Basic 4                    - General Mills
```

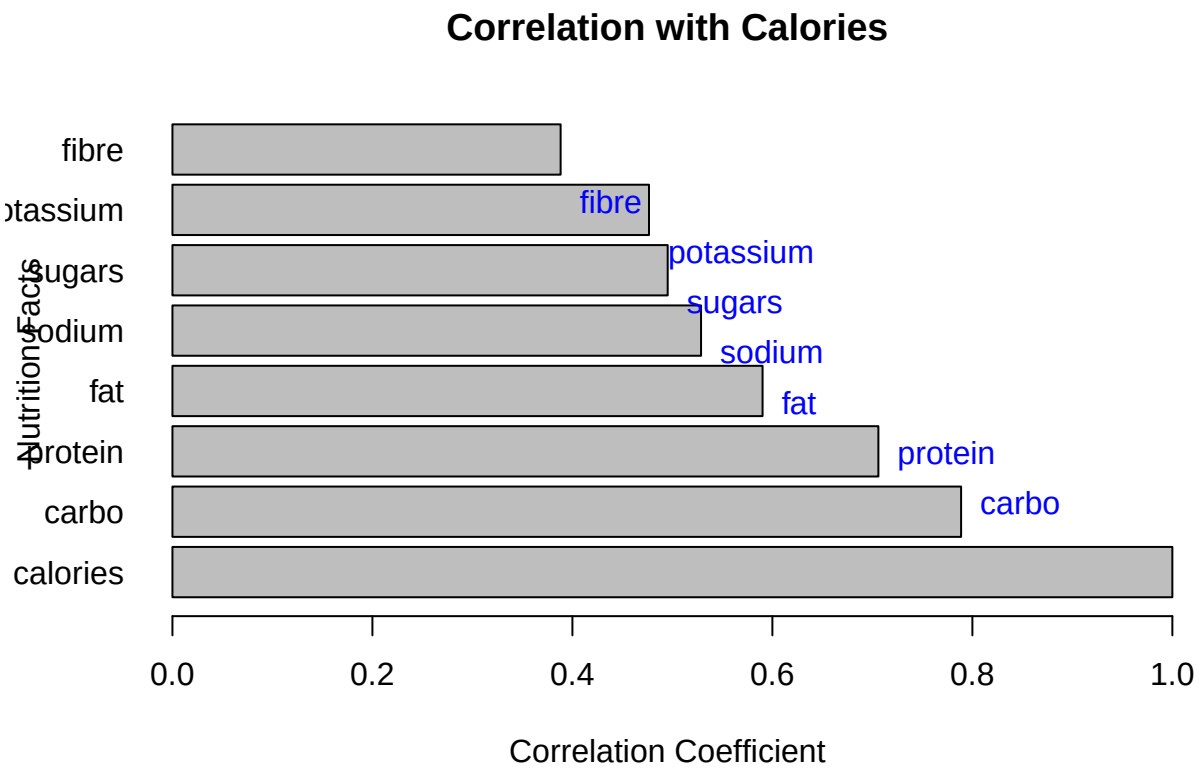
(d)

```r
# Based on the column names, select the appropriate variables for calculation
nutrition_data <- UScereal[, c("calories", "protein", "fat", "sodium", "fibre",
                               "carbo", "sugars", "potassium")]

# Calculate Pearson correlation coefficients
correlation <- cor(nutrition_data)

# Calculate Pearson correlation coefficients between calories and other nutrition facts
correlation_calories <- correlation["calories", ]

# Print the correlations
print(correlation_calories)
```

```
##  calories   protein       fat    sodium     fibre     carbo    sugars potassium
## 1.0000000 0.7060105 0.5901757 0.5286552 0.3882179 0.7887227 0.4952942 0.4765955
```

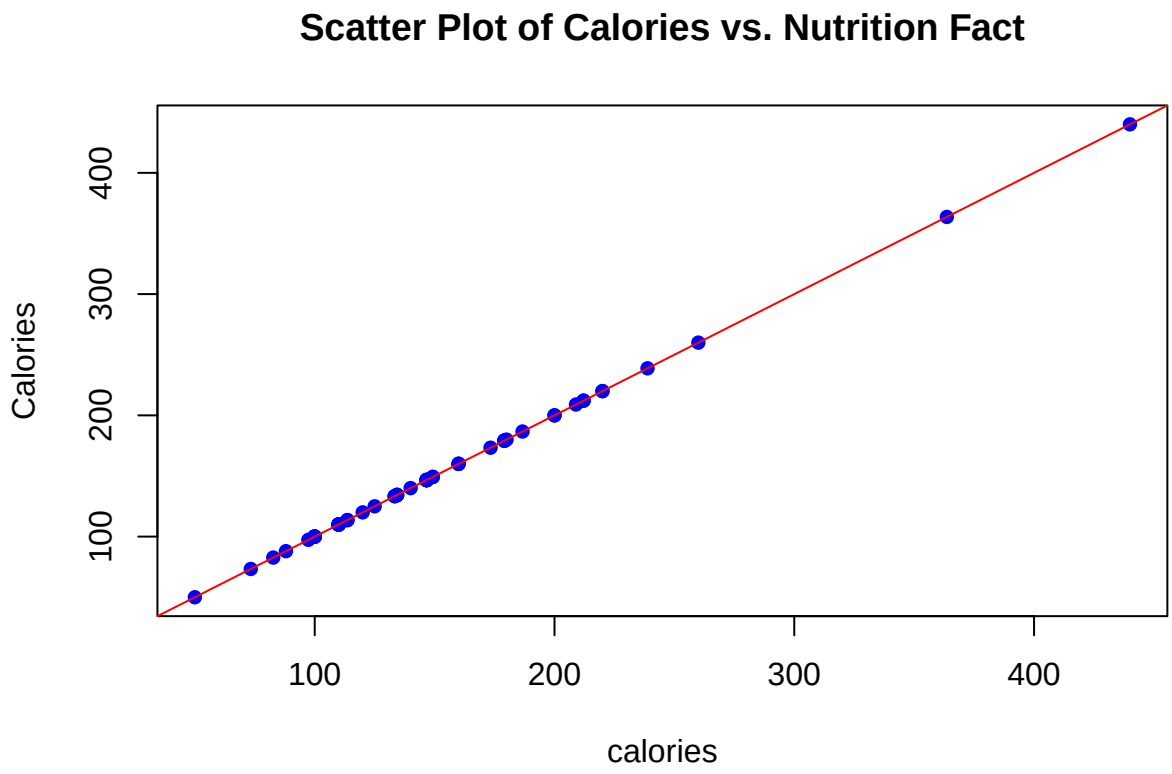
(e)

```r
# Arrange nutrition facts in decreasing order based on their correlation with calories
correlation_sorted <- sort(correlation_calories, decreasing = TRUE)

# Create a bar plot
barplot(correlation_sorted, horiz = TRUE, las = 1, main = "Correlation with Calories",
        xlab = "Correlation Coefficient", ylab = "Nutrition Facts")

# Add labels for the bars
text(correlation_sorted, 1:length(correlation_sorted), labels = names(correlation_sorted), pos = 4, col
```



(f)

```r
# Find the nutrition fact with the largest correlation with calories
max_cor_nutrition <- names(correlation_sorted)[1]

# Create the scatter plot
plot(UScereal[, max_cor_nutrition], UScereal$calories,
     xlab = max_cor_nutrition, ylab = "Calories",
     main = "Scatter Plot of Calories vs. Nutrition Fact",
     pch = 16, col = "blue")

# Add a trend line
abline(lm(UScereal$calories ~ UScereal[, max_cor_nutrition]), col = "red")
```

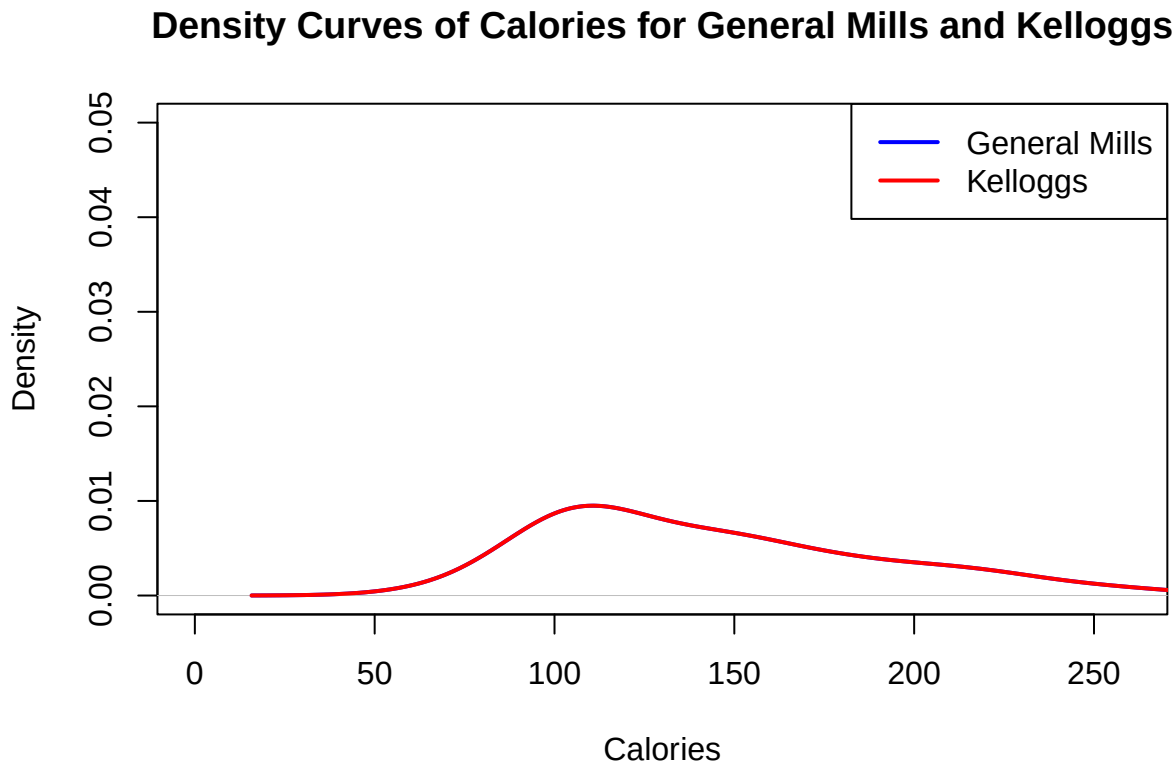



```
# Interpret the intercept and slope
summary(lm(UScereal$calories ~ UScereal[, max_cor_nutrition]))
```

```
## Warning in summary.lm(lm(UScereal$calories ~ UScereal[, max_cor_nutrition])):
## essentially perfect fit: summary may be unreliable

##
## Call:
## lm(formula = UScereal$calories ~ UScereal[, max_cor_nutrition])
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -3.651e-14 -2.110e-15 -1.300e-17  2.356e-15  3.787e-14
##
## Coefficients:
##                                 Estimate Std. Error   t value Pr(>|t|)
## (Intercept)                    2.820e-14  2.404e-15 1.173e+01   <2e-16 ***
## UScereal[, max_cor_nutrition] 1.000e+00  1.487e-17 6.727e+16   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.423e-15 on 63 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 4.525e+33 on 1 and 63 DF,  p-value: < 2.2e-16
```

(g)

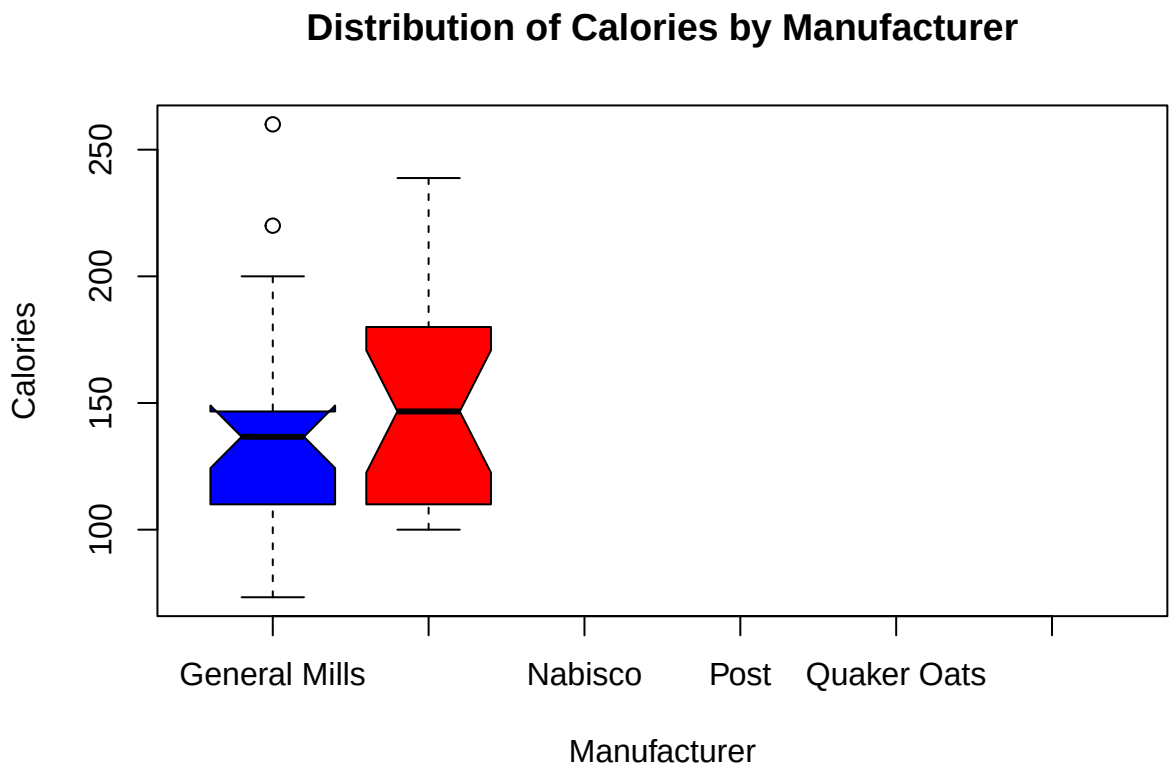
```r
# Filter the dataset for samples from General Mills and Kelloggs
gm_kelloggs <- subset(UScereal, mfr %in% c("General Mills", "Kelloggs"))

# Create density curves for calories for General Mills and Kelloggs
plot(density(gm_kelloggs$calories), main = "Density Curves of Calories for General Mills and Kelloggs",
     xlab = "Calories", xlim = c(0, max(gm_kelloggs$calories, na.rm = TRUE)),
     col = "blue", lwd = 2, ylim = c(0, 0.05))
lines(density(gm_kelloggs$calories), col = "red", lwd = 2)
legend("topright", legend = c("General Mills", "Kelloggs"), col = c("blue", "red"), lwd = 2)
```



(h)

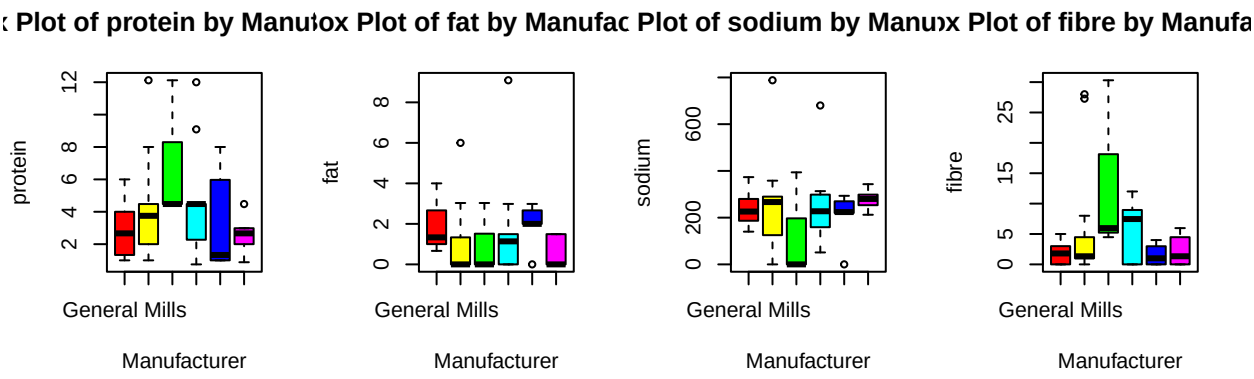
```r
# Create a box plot of calories for General Mills and Kelloggs
boxplot(calories ~ mfr, data = gm_kelloggs,
        main = "Distribution of Calories by Manufacturer",
        xlab = "Manufacturer", ylab = "Calories",
        col = c("blue", "red"), border = "black", notch = TRUE)
```

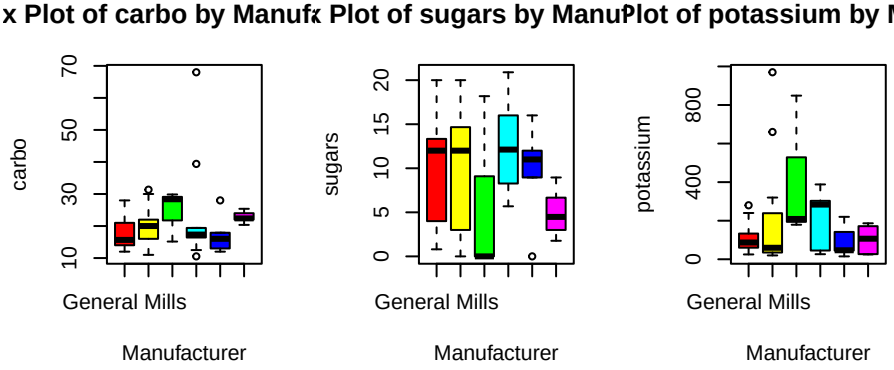
```
## Warning in (function (z, notch = FALSE, width = NULL, varwidth = FALSE, : some
## notches went outside hinges ('box'): maybe set notch=FALSE
```



(i)

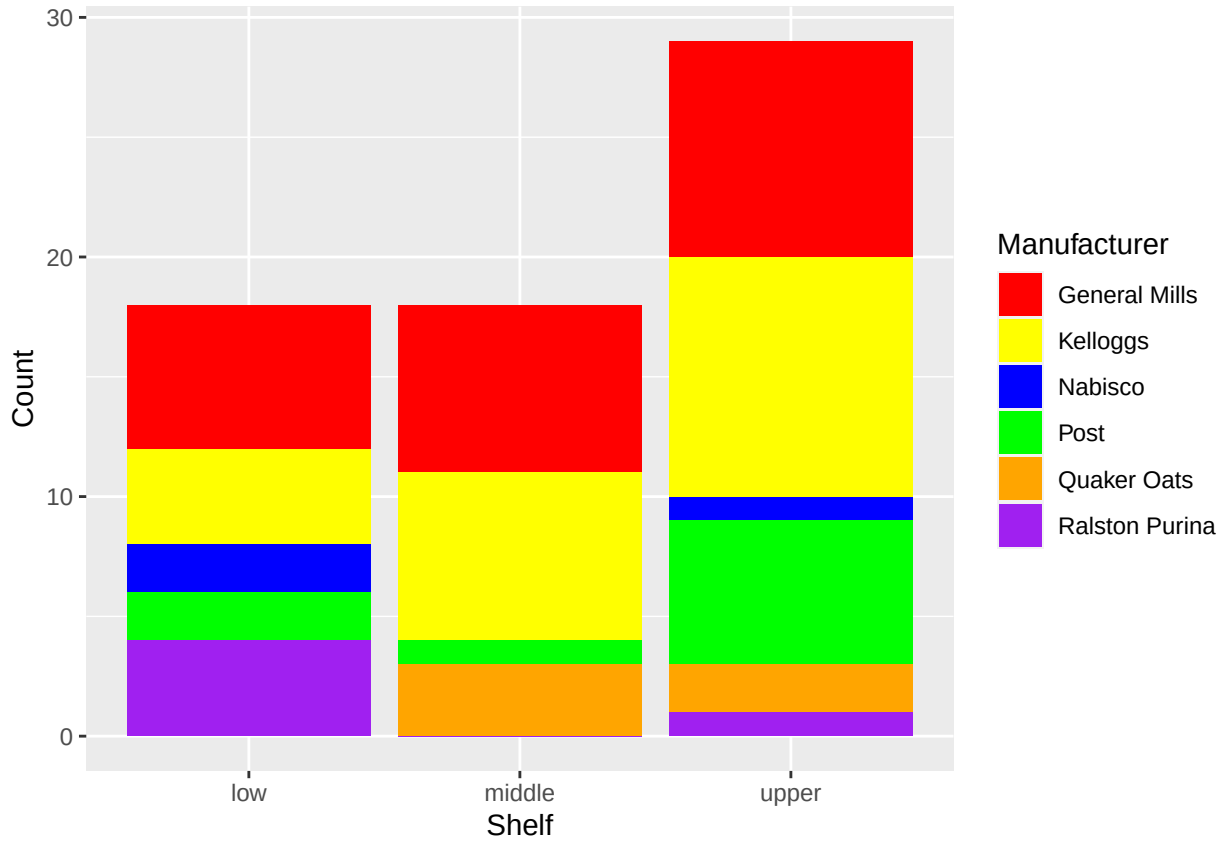
```
# Filter the dataset for samples from the six manufacturers
manufacturers <- c("General Mills", "Kelloggs", "Nabisco", "Post", "Quaker Oats",
                   "Ralston Purina")
manufacturers_data <- subset(UScereal, mfr %in% manufacturers)

# Create side-by-side box plots for each nutrition fact
par(mfrow = c(2, 4))  # Arrange plots in a 2x4 grid
for (fact in c("protein", "fat", "sodium", "fibre", "carbo", "sugars", "potassium")) {
  boxplot(get(fact) ~ mfr, data = manufacturers_data,
          main = paste("Box Plot of", fact, "by Manufacturer"),
          xlab = "Manufacturer", ylab = fact,
          col = rainbow(length(manufacturers)))
}
```





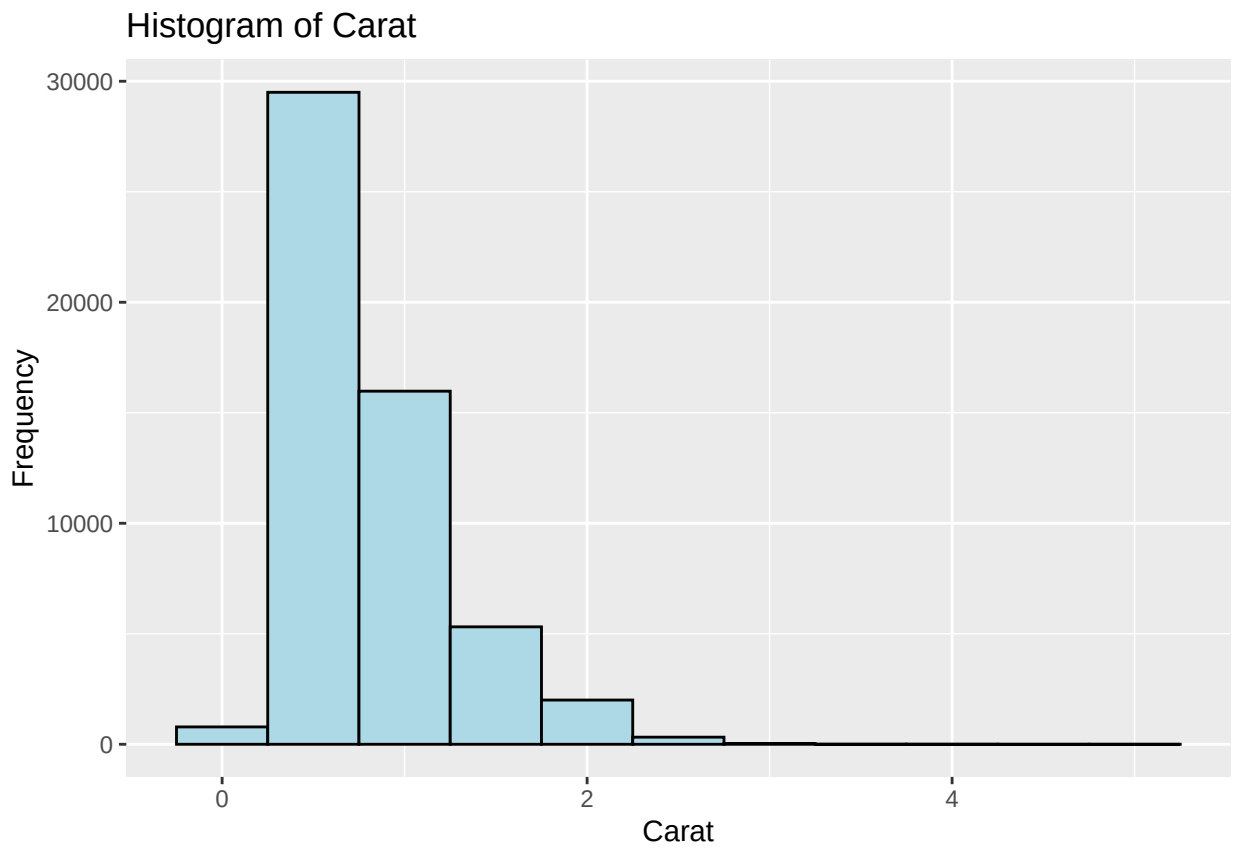
(j)

```
# Aggregate the data to count occurrences for each combination of manufacturer and shelf
manufacture_shelf_count <- table(UScereal$mfr, UScereal$shelf)

# Convert the aggregated table to a data frame
manufacture_shelf_df <- as.data.frame(manufacture_shelf_count)

# Rename the columns for clarity
colnames(manufacture_shelf_df) <- c("Manufacturer", "Shelf", "Count")

# Convert Manufacturer and Shelf to factors to preserve ordering in plot
manufacture_shelf_df$Manufacturer <- factor(manufacture_shelf_df$Manufacturer, levels = unique(manufactu
manufacture_shelf_df$Shelf <- factor(manufacture_shelf_df$Shelf, levels = unique(manufacture_shelf_df$Sh

# Create the stacked bar plot
ggplot(manufacture_shelf_df, aes(x = Shelf, y = Count, fill = Manufacturer)) +
  geom_bar(stat = "identity") +
  labs(x = "Shelf", y = "Count", fill = "Manufacturer") +
  scale_fill_manual(values = c("General Mills" = "red", "Kelloggs" = "yellow",
                               "Nabisco" = "blue", "Post" = "green",
                               "Quaker Oats" = "orange", "Ralston Purina" = "purple"))
```



Problem 3 (a)

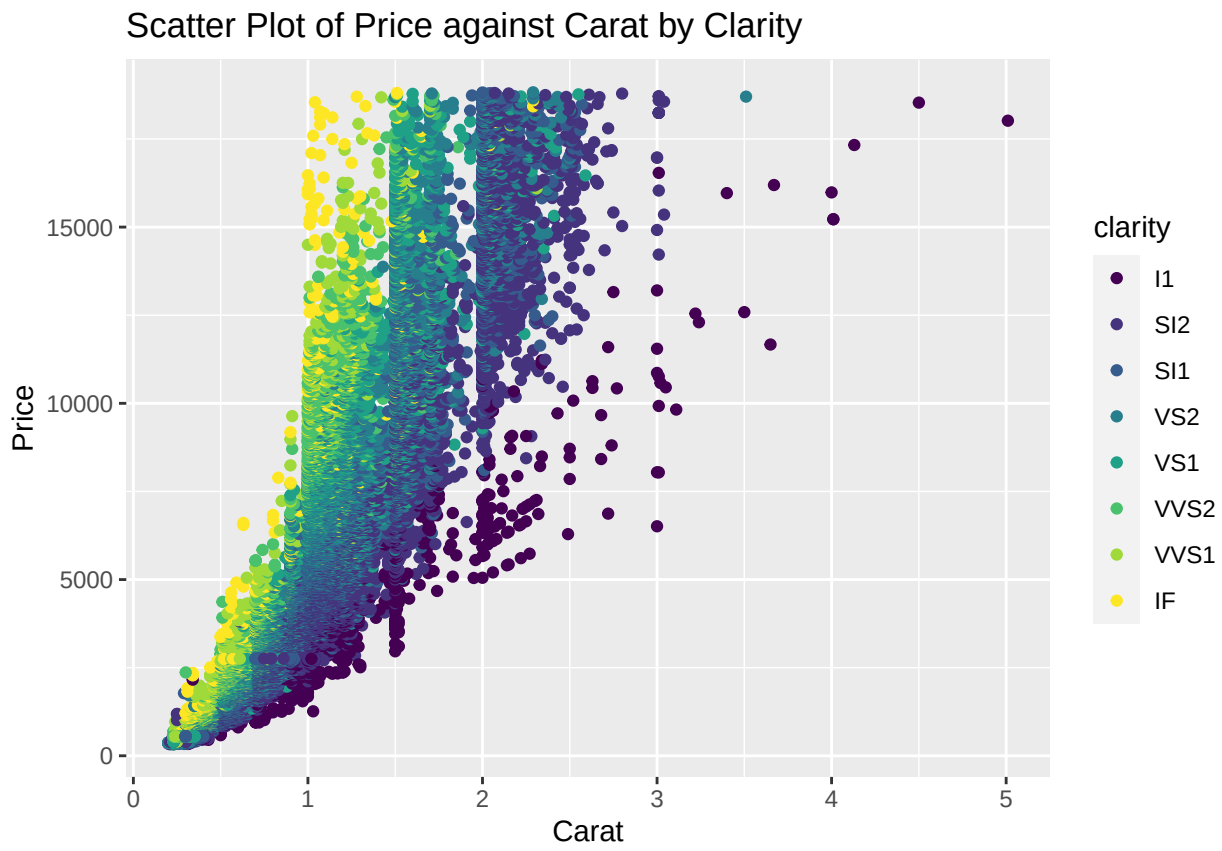
```
# Access the diamonds dataset
data("diamonds")

# Create a histogram of the "carat" variable
ggplot(diamonds, aes(x = carat)) +
  geom_histogram(binwidth = 0.5, fill = "lightblue", color = "black") +
  labs(title = "Histogram of Carat", x = "Carat", y = "Frequency")
```




```
#The histogram of carat sizes in the diamonds dataset shows a
#right-skewed distribution, indicating that the majority of diamonds have
#smaller carat sizes, with a gradual decrease in frequency as carat size increases.
```

(b)

```
# Create the scatter plot
ggplot(diamonds, aes(x = carat, y = price, color = clarity)) +
  geom_point() +
  labs(title = "Scatter Plot of Price against Carat by Clarity",
       x = "Carat",
       y = "Price")
```

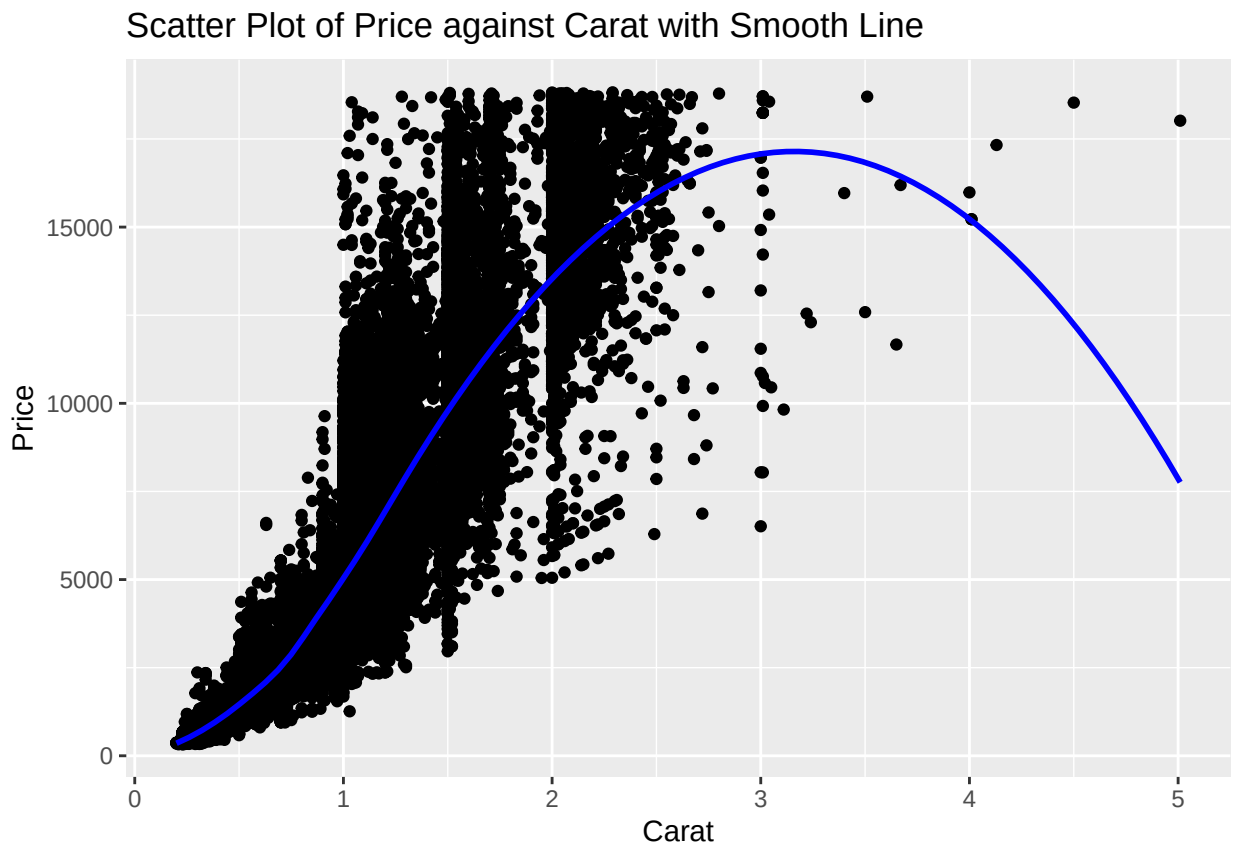



```
#The scatter plot of price against carat, colored by clarity, reveals a positive
#correlation between price and carat size, indicating that larger diamonds tend
#to have higher prices. Additionally, the various colors representing clarity
#levels show how clarity impacts price within different carat ranges.
```

(c)

```
# Create the scatter plot with a smooth line
ggplot(diamonds, aes(x = carat, y = price)) +
  geom_point() +
  geom_smooth(method = "loess", se = FALSE, color = "blue") +
  labs(title = "Scatter Plot of Price against Carat with Smooth Line",
       x = "Carat",
       y = "Price")
```

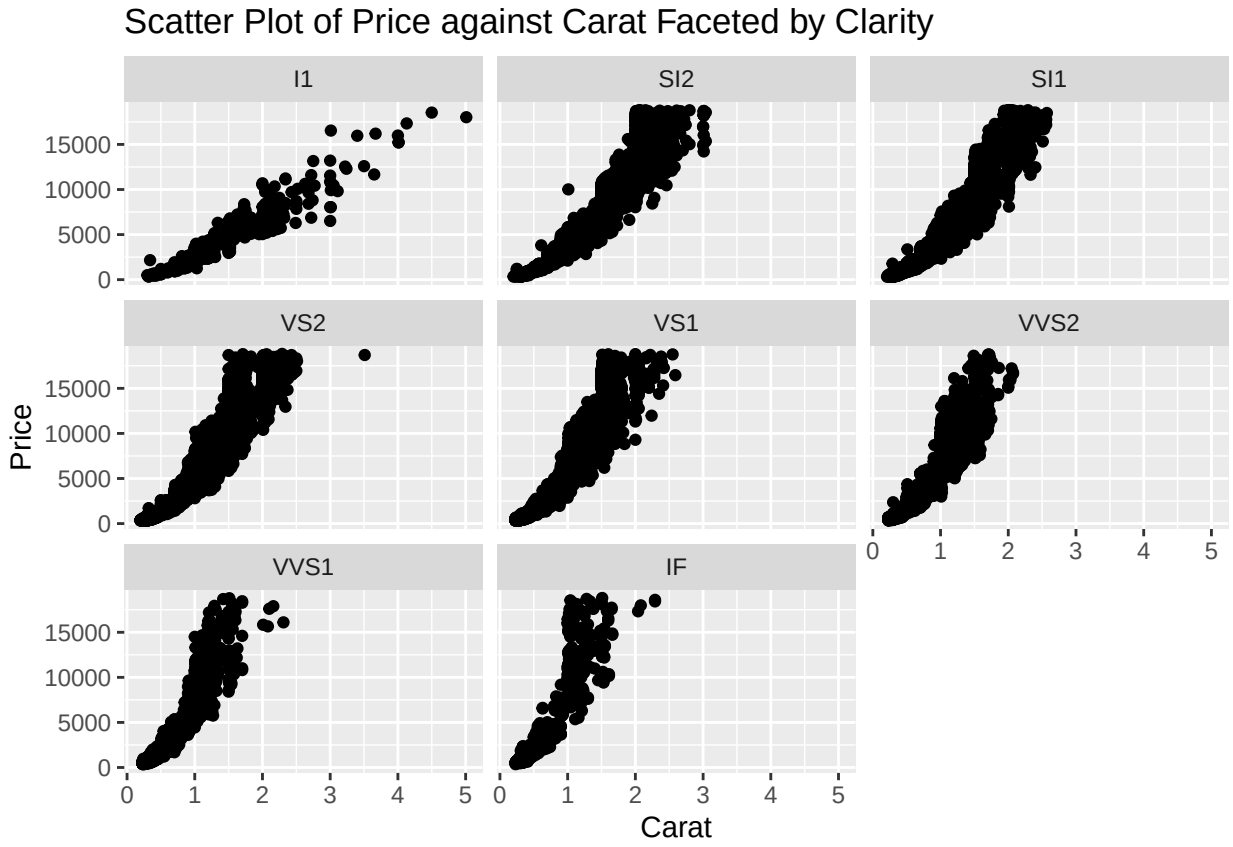
```
## `geom_smooth()` using formula = 'y ~ x'
```




```
#The scatter plot of price against carat with a smooth line suggests a strong
#positive correlation between diamond price and carat size, indicating that
#larger diamonds tend to have higher prices. The smooth line helps visualize
#the overall trend in the relationship between price and carat, showing how
#prices generally increase as carat size increases.
```

(d)

```
# Create the scatter plot with facets by clarity
ggplot(diamonds, aes(x = carat, y = price)) +
  geom_point() +
  facet_wrap(~ clarity) +
  labs(title = "Scatter Plot of Price against Carat Faceted by Clarity",
       x = "Carat",
       y = "Price")
```

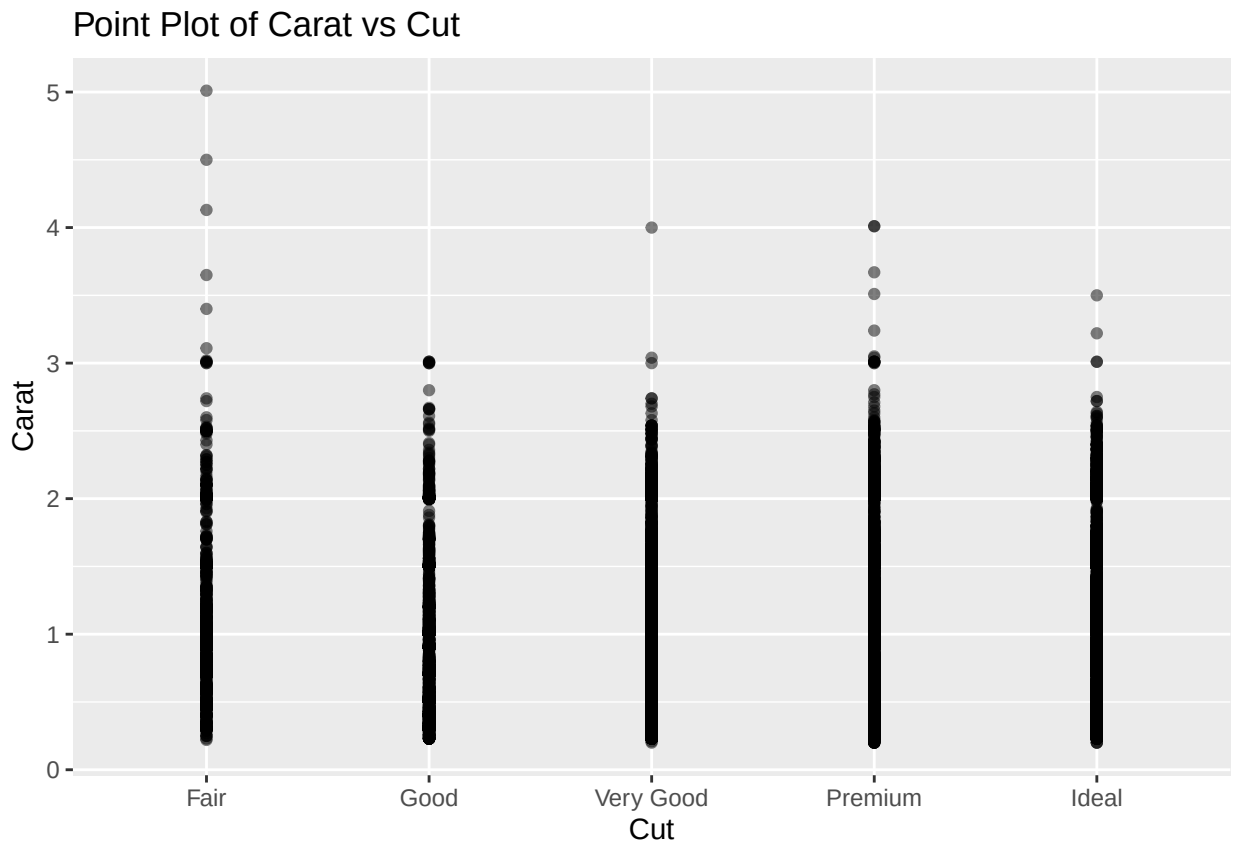



```
#The scatter plot of price against carat, faceted by clarity, reveals distinct
#patterns in the relationship between these variables for different clarity levels.
#Higher clarity diamonds tend to exhibit stronger positive correlations between
#price and carat size, while lower clarity diamonds may show more variability in
#this relationship.
```

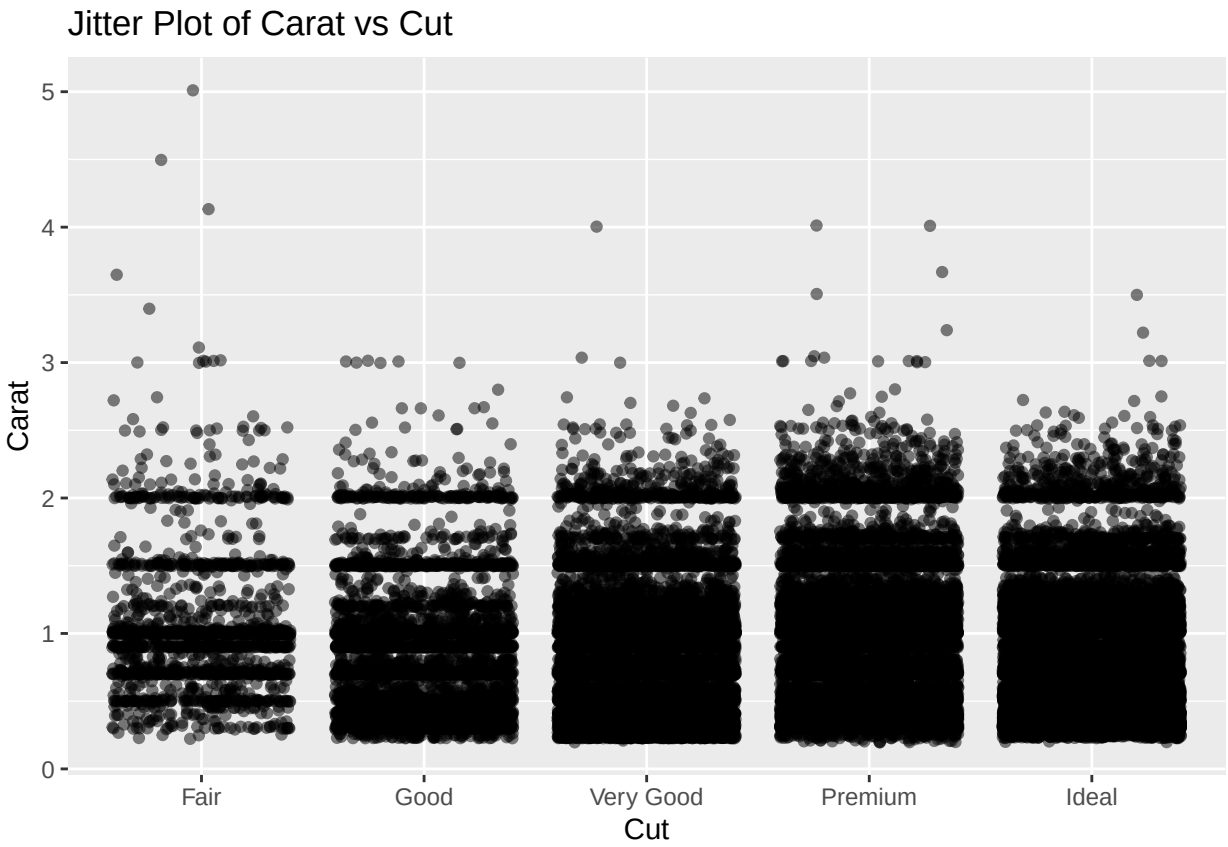
(e)

```
# Create point plot of carat vs cut
point_plot <- ggplot(diamonds, aes(x = cut, y = carat)) +
  geom_point(alpha = 0.5) +
  labs(title = "Point Plot of Carat vs Cut",
       x = "Cut",
       y = "Carat")

# Create jitter plot of carat vs cut
jitter_plot <- ggplot(diamonds, aes(x = cut, y = carat)) +
  geom_jitter(alpha = 0.5) +
  labs(title = "Jitter Plot of Carat vs Cut",
       x = "Cut",
       y = "Carat")

# Create box plot of carat vs cut
box_plot <- ggplot(diamonds, aes(x = cut, y = carat)) +
  geom_boxplot() +
```

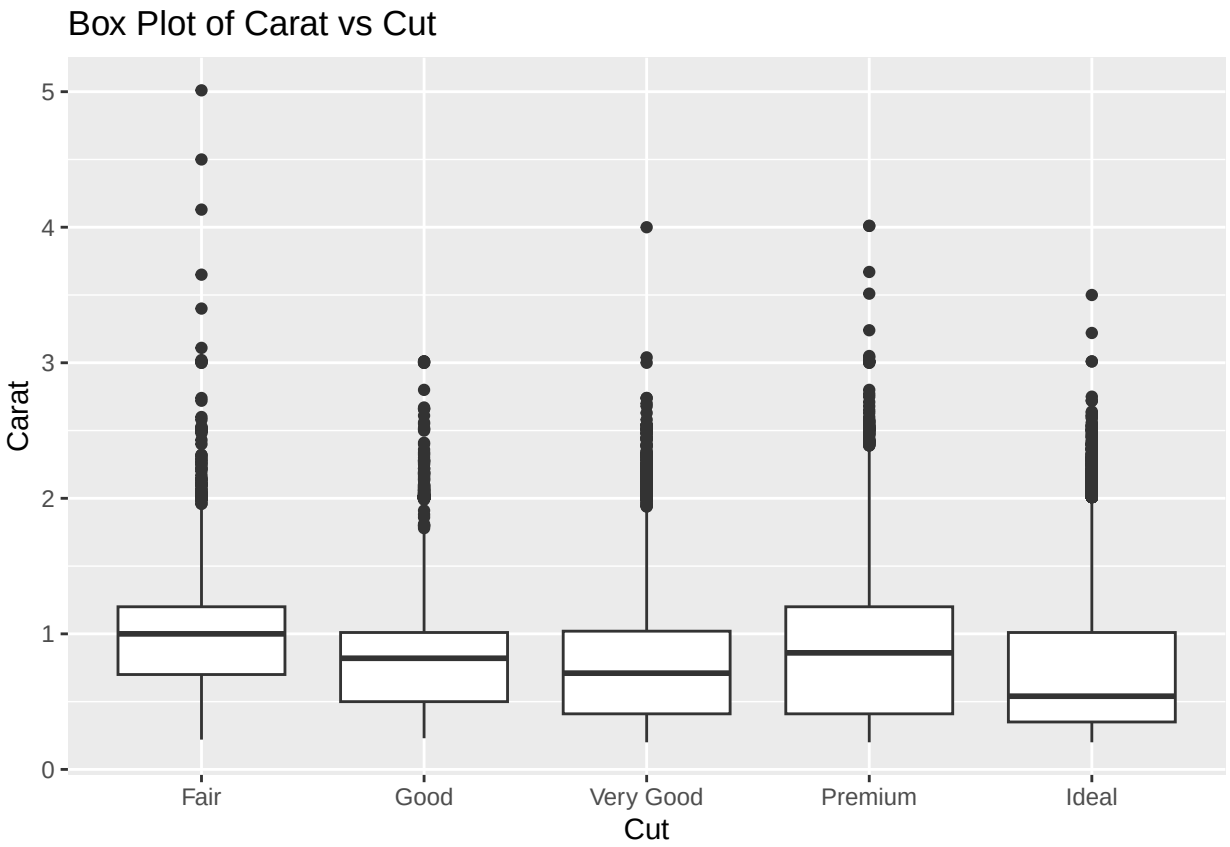
```
  labs(title = "Box Plot of Carat vs Cut",
       x = "Cut",
       y = "Carat")

# Create violin plot of carat vs cut
violin_plot <- ggplot(diamonds, aes(x = cut, y = carat)) +
  geom_violin() +
  labs(title = "Violin Plot of Carat vs Cut",
       x = "Cut",
       y = "Carat")

# Output all plots
point_plot
```



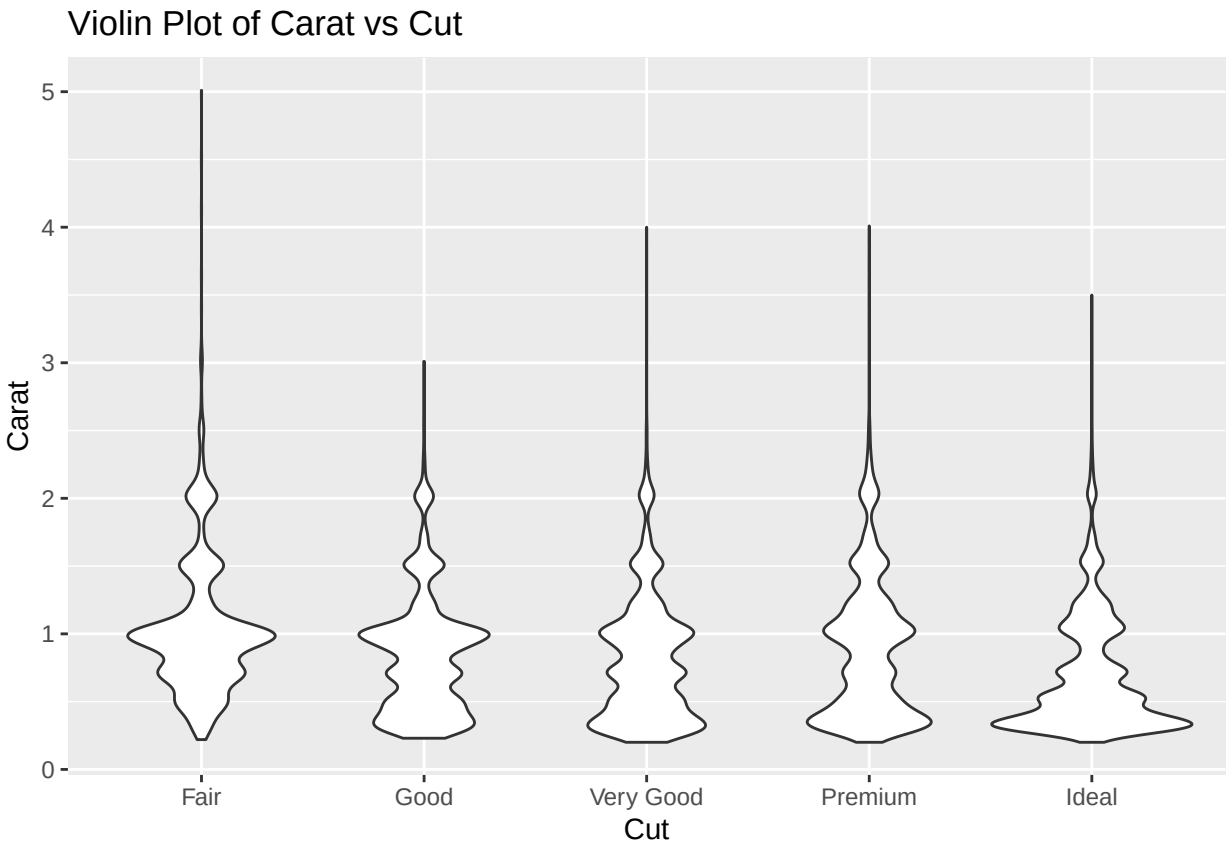

```
jitter_plot
```


Jitter Plot of Carat vs Cut
Carat
5
4
3
2
1
0
Fair
Good
Very Good
Premium
Ideal
Cut

```
box_plot
```


Box Plot of Carat vs Cut
Carat
Cut
0
1
2
3
4
5
Fair
Good
Very Good
Premium
Ideal

```
violin_plot
```




```
#the violin plot may be the most effective as it combines the advantages of the
#box plot (summary statistics) with additional density information, allowing for
#a more comprehensive understanding of the distribution of carat values across
#different cut categories.
```