DS311 - R Lab Assignment

Christina Castillo

2024-11-09

R Assignment 1

- In this assignment, we are going to apply some of the build in data set in R for descriptive statistics analysis.
- To earn full grade in this assignment, students need to complete the coding tasks for each question to get the result.
- After finished all the questions, knit the document into HTML format for submission.

Question 1

Using the **mtcars** data set in R, please answer the following questions.

```r
# Loading the data
data(mtcars)

# Head of the data set
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

a. Report the number of variables and observations in the data set.

The mtcars dataset has 11 variables and 32 obersvations.

```r
# Enter your code here!
num_vars <- ncol(mtcars)
num_obs <- nrow(mtcars)

# Answer:
print(paste("There are total of", num_vars, "variables and", num_obs, "observations in this data set."))
```

```
## [1] "There are total of 11 variables and 32 observations in this data set."
```

b. Print the summary statistics of the data set and report how many discrete and continuous variables are in the data set.

```r
# Enter your code here!
summary(mtcars)
```

```
##       mpg             cyl             disp             hp       
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0  
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
##       drat             wt             qsec             vs        
##  Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
##  1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
##  Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
##  Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
##  3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
##  Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000  
##        am              gear            carb      
##  Min.   :0.0000   Min.   :3.000   Min.   :1.000  
##  1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000  
##  Median :0.0000   Median :4.000   Median :2.000  
##  Mean   :0.4062   Mean   :3.688   Mean   :2.812  
##  3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000  
##  Max.   :1.0000   Max.   :5.000   Max.   :8.000
```

```r
# Identify discrete variables (those with only whole numbers/few unique values)
#Discrete variables are those that:
#Take on specific, countable values (usually whole numbers)
#Have clear, separate categories or steps
#Cannot take values between the steps
#Usually have a limited number of possible values

#In the mtcars dataset:
#'cyl' (cylinders): Only comes in 4, 6, or 8 - you can't have 5.5 cylinders
#'vs' (engine shape): Only 0 or 1 (V-shape or straight)
#'am' (transmission): Only 0 or 1 (automatic or manual)
#'gear': Only 3, 4, or 5 - you can't have 3.5 gears
#'carb' (carburetors): Whole numbers 1, 2, 3, 4, 6, 8

discrete_vars <- c('cyl', 'vs', 'am', 'gear', 'carb')
discrete_count <- length(discrete_vars)

# Identify continuous variables (remaining variables)
#mpg - Miles Per Gallon
#disp - Displacement (engine size in cubic inches)
#hp - Horsepower
#drat - Differential Rear Axle raTio
#wt - Weight (in 1000 lbs)
#qsec - Quarter mile time in Seconds (time to travel 1/4 mile from standing start)
#In contrast, continuous variables:
```

```
#Can take any value within a range
#Can have decimal points
#Have theoretically infinite possible values
#Like:

#'mpg': Can be 21.4, 21.5, 21.523, etc.
#'wt': Can be 2.62, 2.875, etc.
#'qsec': Can be 16.46, 16.47, etc.

#This distinction is important for:

#Choosing appropriate statistical tests
#Selecting visualization methods
#Understanding the nature of the data
continuous_vars <- c('mpg', 'disp', 'hp', 'drat', 'wt', 'qsec')
continuous_count <- length(continuous_vars)

# Print which variables fall into each category
cat("\nDiscrete variables:", paste(discrete_vars, collapse=", "))
```

```
##
## Discrete variables: cyl, vs, am, gear, carb
```

```
cat("\nContinuous variables:", paste(continuous_vars, collapse=", "))
```

```
##
## Continuous variables: mpg, disp, hp, drat, wt, qsec
```

```
# Answer:
print(paste("There are", discrete_count, "discrete variables and", continuous_count, "continuous variabl
```

```
## [1] "There are 5 discrete variables and 6 continuous variables in this data set."
```

c. Calculate the mean, variance, and standard deviation for the variable **mpg** and assign them into variable names m, v, and s. Report the results in the print statement.

```
# Enter your code here!
m <- round(mean(mtcars$mpg),2)
v <- round(var(mtcars$mpg),2)
s <- round(sd(mtcars$mpg),2)

#Answer:
# Print results with rounded numbers for cleaner output
print(paste("The average of Mile Per Gallon from this data set is ", m, " with variance ", v, " and sta
```

```
## [1] "The average of Mile Per Gallon from this data set is  20.09  with variance  36.32  and standard
```

d. Create two tables to summarize 1) average mpg for each cylinder class and 2) the standard deviation of mpg for each gear class.

```r
# Enter your code here!
#create a table:
#aggregate(what_to_calculate ~ what_to_group_by, data = dataset, FUN = function_to_apply)

#Table 1: Average MPG for each cylinder class
#Breaking down the components:
#aggregate(): Function used to compute summary statistics for subgroups
#mpg ~ cyl: Formula indicating we want to calculate statistics for mpg (left) grouped by cyl (right)
#data = mtcars: Specifies which dataset to use
#FUN = mean: Specifies we want to calculate the mean/average
avg_mpg_by_cyl <- aggregate(mpg ~ cyl, data = mtcars, FUN = mean)

#Table 2: Standard deviation of MPG for each gear class
#Breaking down the components:
#mpg ~ gear: Now grouping by gear instead of cylinders
#FUN = sd: Specifies we want to calculate the standard deviation
sd_mpg_by_gear <- aggregate(mpg ~ gear, data = mtcars, FUN = sd)

#Grouping:  splitting the data into groups (by cylinder or gear)
#Summarizing: calculating a summary statistic (mean or sd) for each group
#Variable selection: focusing on mpg as our variable of interest


# Print results in full sentences
cat("Average MPG by cylinder class:\n")
```

```
## Average MPG by cylinder class:
```

```r
cat("\tCars with", avg_mpg_by_cyl$cyl[1], "cylinders have an average of", round(avg_mpg_by_cyl$mpg[1], 2
```

```
##  Cars with 4 cylinders have an average of 26.66 mpg
```

```r
cat("\tCars with", avg_mpg_by_cyl$cyl[2], "cylinders have an average of", round(avg_mpg_by_cyl$mpg[2], 2
```

```
##  Cars with 6 cylinders have an average of 19.74 mpg
```

```r
cat("\tCars with", avg_mpg_by_cyl$cyl[3], "cylinders have an average of", round(avg_mpg_by_cyl$mpg[3], 2
```

```
##  Cars with 8 cylinders have an average of 15.1 mpg
```

```r
cat("Standard deviation of MPG by gear class:\n")
```

```
## Standard deviation of MPG by gear class:
```

```r
cat("\tCars with", sd_mpg_by_gear$gear[1], "gears have a standard deviation of", round(sd_mpg_by_gear$mp
```

```
##  Cars with 3 gears have a standard deviation of 3.37 mpg
```

```r
cat("\tCars with", sd_mpg_by_gear$gear[2], "gears have a standard deviation of", round(sd_mpg_by_gear$mp
```

```
##  Cars with 4 gears have a standard deviation of 5.28 mpg
```

```r
cat("\tCars with", sd_mpg_by_gear$gear[3], "gears have a standard deviation of", round(sd_mpg_by_gear$mp
```

```
##  Cars with 5 gears have a standard deviation of 6.66 mpg
```

e. Create a crosstab that shows the number of observations belong to each cylinder and gear class combinations. The table should show how many observations given the car has 4 cylinders with 3 gears, 4 cylinders with 4 gears, etc. Report which combination is recorded in this data set and how many observations for this type of car.

```r
# Enter your code here!

crosstab_df <- as.data.frame.matrix(table(mtcars$cyl, mtcars$gear)) #method 2
names(crosstab_df) <- paste(names(crosstab_df), "Gears")
rownames(crosstab_df) <- paste(rownames(crosstab_df), "Cylinders")
print(crosstab_df)
```

```
##             3 Gears 4 Gears 5 Gears
## 4 Cylinders       1       8       2
## 6 Cylinders       2       4       1
## 8 Cylinders      12       0       2
```

```r
# Find the maximum count and location
#The which(arr.ind = TRUE) function returns the row and column indices
#where the maximum value occurs in the table.
max_count <- max(crosstab_df)
max_location <- which(crosstab_df == max_count, arr.ind = TRUE)
#print("Location of maximum value:")
#print(max_location)
#print(paste("Maximum value", max_count, "occurs at:"))
#print(paste("Row:", rownames(crosstab_df)[max_location[1]]))
#print(paste("Column:", names(crosstab_df)[max_location[2]]))

# Finds the indices of the maximum value
#
max_indices <- which(crosstab_df == max_count, arr.ind = TRUE)
#print("Maximum indices:")
#print(max_indices)
#print(paste("Row:", rownames(crosstab_df)[max_indices[1]]))
#print(paste("Column:", names(crosstab_df)[max_indices[2]]))
# How: if crosstab_df == max_count: = 12
#hen, which(... , arr.ind = TRUE):
#Finds where the TRUE value(s) are in the matrix
#arr.ind = TRUE tells R to return both row and column indices
#Returns a matrix with row and column numbers where TRUE was found



# Get the corresponding cylinder and gear values
```

```r
#max_cyl <- as.numeric(rownames(crosstab_df)[max_indices[1]]) # use only if number and no text
#max_gear <- as.numeric(colnames(crosstab_df)[max_indices[2]]) # use only if number and no text
#max_cyl <- as.numeric(gsub(" Cylinders", "", rownames(crosstab_df)[max_indices[1]]))
#max_gear <- as.numeric(gsub(" Gears", "", names(crosstab_df)[max_indices[2]]))
max_cyl <- as.numeric(sub(" .*", "", rownames(crosstab_df)[max_indices[1]]))
max_gear <- as.numeric(sub(" .*", "", names(crosstab_df)[max_indices[2]]))

# Print the result
print(paste("The most common car type in this data set is car with", max_cyl, "cylinders and", max_gear
```

```
## [1] "The most common car type in this data set is car with 8 cylinders and 3 gears. There are total o
```

```r
#Note: this code would need to be adjusted if there were many max counts with the same number:
# max_count <- max(crosstab_df)
# max_indices <- which(crosstab_df == max_count, arr.ind = TRUE)
#
# # Handle multiple maximum values
# if(nrow(max_indices) > 1) {
#     print("There are multiple combinations with the maximum count:")
#     for(i in 1:nrow(max_indices)) {
#         cyl <- as.numeric(gsub(" Cylinders", "", rownames(crosstab_df)[max_indices[i, "row"]]))
#         gear <- as.numeric(gsub(" Gears", "", names(crosstab_df)[max_indices[i, "col"]]))
#         print(paste("Combination", i, ":", cyl, "cylinders and", gear, "gears"))
#     }
#     print(paste("Each combination has", max_count, "cars in the dataset."))
# } else {
#     # Original single maximum code
#     max_cyl <- as.numeric(gsub(" Cylinders", "", rownames(crosstab_df)[max_indices[1]]))
#     max_gear <- as.numeric(gsub(" Gears", "", names(crosstab_df)[max_indices[2]]))
#     print(paste("The most common car type in this data set is car with", max_cyl,
#                 "cylinders and", max_gear, "gears."))
#     print(paste("There are total of", max_count, "cars belong to this specification in the data set.")
# }
```

Question 2

Use different visualization tools to summarize the data sets in this question.

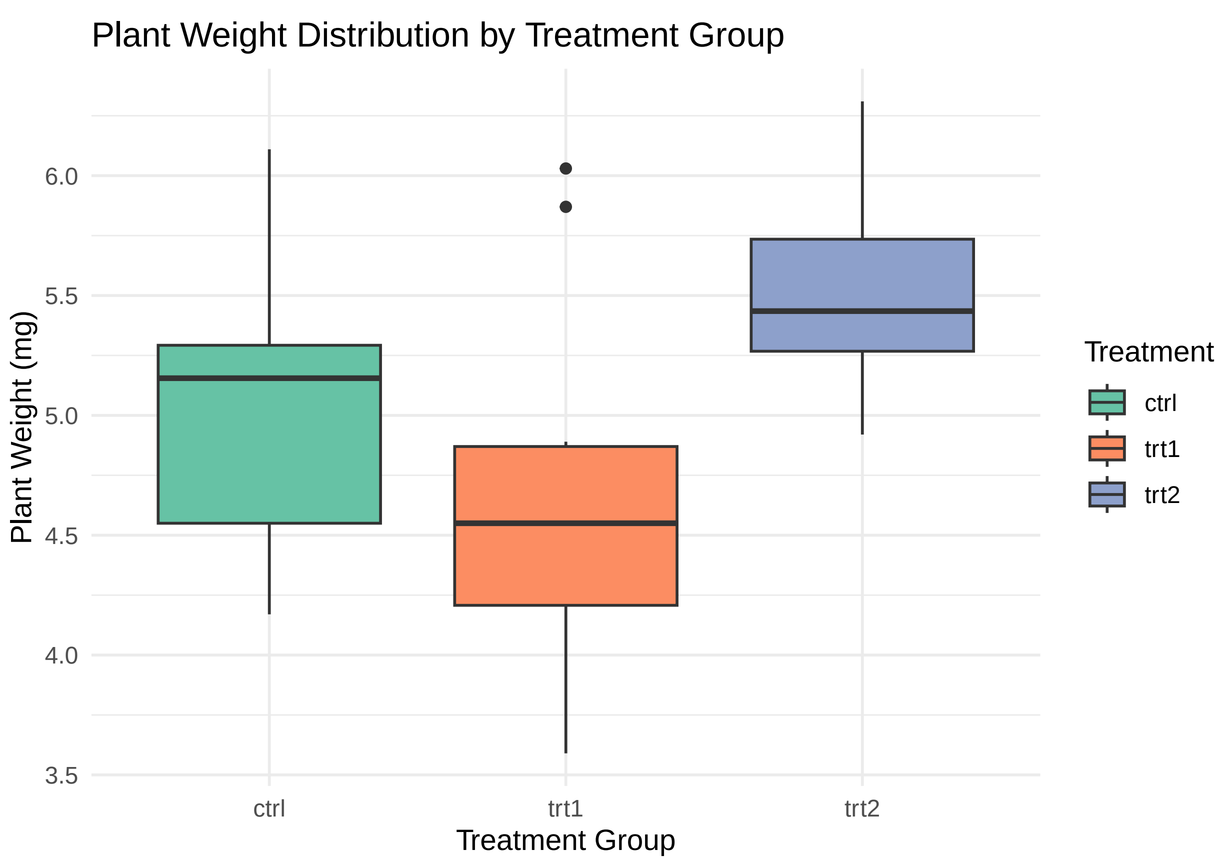
a. Using the **PlantGrowth** data set, visualize and compare the weight of the plant in the three separated group. Give labels to the title, x-axis, and y-axis on the graph. Write a paragraph to summarize your findings.

```r
# Load the data set
data("PlantGrowth")

# Head of the data set
head(PlantGrowth)
```

```
##   weight group
```

```
## 1   4.17  ctrl
## 2   5.58  ctrl
## 3   5.18  ctrl
## 4   6.11  ctrl
## 5   4.50  ctrl
## 6   4.61  ctrl
```

```r
# Enter your code here!
# Load required libraries
# Load required libraries
library(ggplot2)

# Create boxplot
ggplot(PlantGrowth, aes(x=group, y=weight, fill=group)) +
  geom_boxplot() +
  labs(title="Plant Weight Distribution by Treatment Group",
       x="Treatment Group",
       y="Plant Weight (mg)",
       fill="Treatment") +
  theme_minimal() +
  scale_fill_brewer(palette="Set2")
```




```r
# Create violin plot with individual points
ggplot(PlantGrowth, aes(x=group, y=weight, fill=group)) +
  geom_violin(alpha=0.7) +
```

```r
  geom_point(position=position_jitter(width=0.1), alpha=0.5) +
  stat_summary(fun=mean, geom="point", shape=23, size=3, fill="white") +
  labs(title="Plant Weight Distribution and Individual Observations by Treatment Group",
       x="Treatment Group",
       y="Plant Weight (mg)",
       fill="Treatment") +
  theme_minimal() +
  scale_fill_brewer(palette="Set2")
```


```r
# Calculate summary statistics
summary_stats <- tapply(PlantGrowth$weight, PlantGrowth$group,
                        function(x) c(mean=mean(x), sd=sd(x)))
print("Summary Statistics:")
```

```
## [1] "Summary Statistics:"
```

```r
print(summary_stats)
```

```
## $ctrl
##      mean        sd
## 5.0320000 0.5830914
##
## $trt1
##      mean        sd
```

```
## 4.6610000 0.7936757
##
## $trt2
##      mean        sd
## 5.5260000 0.4425733
```

```r
# library(ggplot2)
# library(dplyr)
#
# # Create a boxplot
# ggplot(PlantGrowth, aes(x=group, y=weight, fill=group)) +
#   geom_boxplot() +
#   labs(title="Plant Weight Distribution by Treatment Group",
#        x="Treatment Group",
#        y="Plant Weight (mg)",
#        fill="Treatment") +
#   theme_minimal() +
#   scale_fill_brewer(palette="Set2")
#
# # Create a violin plot with points
# ggplot(PlantGrowth, aes(x=group, y=weight, fill=group)) +
#   geom_violin(alpha=0.7) +
#   geom_jitter(width=0.1, alpha=0.5) +
#   stat_summary(fun = "mean", geom = "point", shape = 23, size = 3, fill = "white") +
#   labs(title="Plant Weight Distribution and Individual Observations by Treatment Group",
#        x="Treatment Group",
#        y="Plant Weight (mg)",
#        fill="Treatment") +
#   theme_minimal() +
#   scale_fill_brewer(palette="Set2")
#
# # Calculate summary statistics
# summary_stats <- PlantGrowth %>%
#   group_by(group) %>%
#   summarise(
#     mean_weight = mean(weight),
#     sd_weight = sd(weight),
#     min_weight = min(weight),
#     max_weight = max(weight)
#   )
#
# print(summary_stats)
```

Result: ==> Report a paragraph to summarize your finding from the plot: Paragraph:Based on the provided boxplot, violin plot, and data sample, here's a comprehensive summary of the findings: The plant growth experiment compared three conditions: a control group (ctrl) and two treatments (trt1 and trt2). The boxplot and violin plot reveal distinct patterns across these groups. Treatment 2 (trt2) showed the most promising results, with consistently higher plant weights and less variability in the distribution, as evidenced by the more compact box in the boxplot and the shape of the violin plot. The control group (ctrl) demonstrated moderate performance. Treatment 1 (trt1) appears to be the least effective, showing lower overall weights and greater extremes of variability than both the control and Treatment 2. The violin plot's shape for trt1 is more spread out, indicating higher variability in plant response to this treatment. The individual points overlaid on the violin plot also help visualize how the measurements are distributed within each group, showing that Treatment 2 consistently produced heavier plants while Treatment 1 had more scattered results.

Not part of the paragraph: Based on the boxplot, violin plot, and summary statistics for the PlantGrowth dataset, here's a comprehensive summary of the findings:

The analysis reveals distinct patterns across the three treatment groups:

Control Group (ctrl): - Mean weight of 5.032 mg with a standard deviation of 0.583 mg - Looking at the green box (control group) in the boxplot:

–Most plants weighed around 5.0-5.2 mg –When plants weighed less than this, their weights bounced around more –When plants weighed more than this, their weights stayed pretty close together

–In other words: In the control group (where plants got no special treatment), the heavier plants were more similar in weight to each other, while the lighter –plants had more different weights. –This information is identify by the uneven spacing in the green box - the line (median) being closer to the top of the box means this pattern exists. - Median around 5.15 mg - Serves as the baseline for comparison Treatment 1 (trt1): –These plants were generally lighter (weighed less) than both the regular plants (control) and Treatment 2 plants —The lowest average weight —The most variation in weights (highest standard deviation) - meaning Treatment 1 had the most variable (least consistent) results –The weights were evenly balanced around the middle value (shown by the black line being in the middle of the orange box-in boxplot) –A couple plants surprisingly grew heavier than expected (shown by the dots above - these are outliers) –Some plants grew very poorly, weighing much less than others (shown by the long line extending down) –Information from the violin plot shows that plants in Treatment 1 tended to fall into two distinct weight groups rather than being spread evenly across all weights. –Overall, this treatment doesn't seem to help plants grow better - if anything, it might be making them lighter than if we did nothing (control group)

Treatment 2 (trt2): Overall, Treatment 2 appears to be the most effective in promoting plant growth, showing both higher weights and more consistent results. In contrast, Treatment 1 appears to have a slightly negative impact on plant growth compared to the control group and shows more variable results.

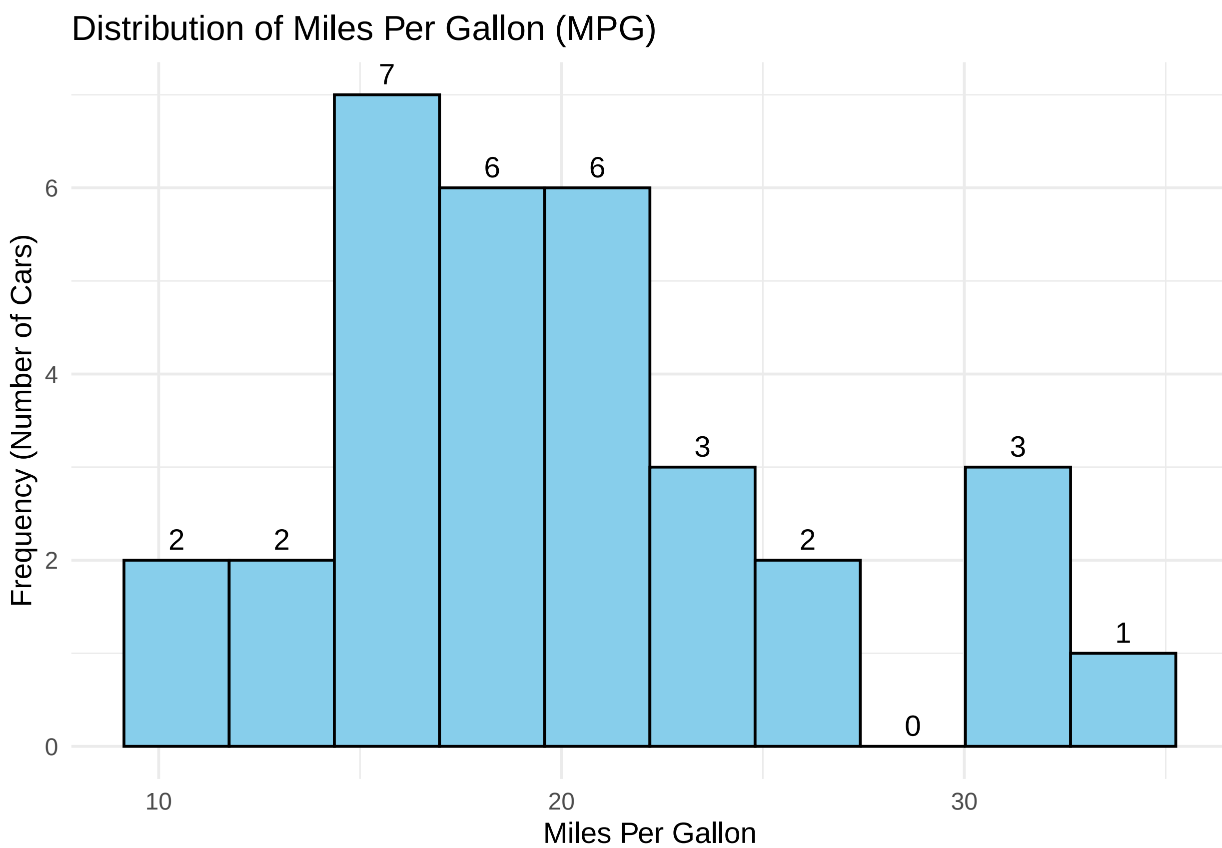
–Boxplot shows: —Highest position (heaviest weights) —Compact box (consistent results) —Median line fairly centered

–Violin plot adds important details about individual plants: —We can see actual dots showing each plant's weight —About 10 individual plants in this group —Plants clustered mostly between 5.0-6.0 mg —The "violin" shape is wider in the middle, showing where most plants' weights clustered —The white diamond shows the mean —We can see there's a slight gap in weights (where there are no dots) in a few places. —The wider blue shape in the higher weight range shows most plants clustered at heavier weights — Individual dots are grouped closer together, again showing more consistent results

-Bottom line: Treatment 2 seems to work the best because: -Plants grew heavier than other cntrol or treatment 1 -Results were more predictable (plants responded more consistently to treatment2)

b. Using the **mtcars** data set, plot the histogram for the column **mpg** with 10 breaks. Give labels to the title, x-axis, and y-axis on the graph. Report the most observed mpg class from the data set.

```r
# Load required libraries
library(ggplot2)

# Create histogram with data labels
ggplot(mtcars, aes(x=mpg)) +
  geom_histogram(bins=10, fill="skyblue", color="black") +
  labs(title="Distribution of Miles Per Gallon (MPG)",
       x="Miles Per Gallon",
       y="Frequency (Number of Cars)") +
  theme_minimal() +
  # Add counts on top of each bar using after_stat(count)
  stat_bin(bins=10, geom="text", aes(label=after_stat(count)),
           vjust=-0.5, color="black")  # Adjust vjust to position labels
```




```r
# Calculate the breaks and frequency
breaks <- seq(min(mtcars$mpg), max(mtcars$mpg), length.out=11)
mpg_groups <- cut(mtcars$mpg, breaks=breaks, include.lowest=TRUE)
freq_table <- table(mpg_groups)
most_common_class <- names(freq_table)[which.max(freq_table)]

# Extract the range for the most common class
class_range <- strsplit(gsub("\\(|\\]", "", most_common_class), ",")[[1]]
class_start <- round(as.numeric(class_range[1]), 1)
class_end <- round(as.numeric(class_range[2]), 1)

# Print the most observed mpg class
print(paste0("Most of the cars in this data set are in the class of ",
             class_start, "-", class_end, " miles per gallon."))
```

```
## [1] "Most of the cars in this data set are in the class of 15.1-17.5 miles per gallon."
```

"Most of the cars in this data set are in the class of 15-17.5 miles per gallon."

This can be determined by looking at the tallest bar in the histogram, which occurs in the range of approximately 15-17.5 MPG and shows a frequency of 7 cars. This represents the mode of the distribution, meaning more cars fall into this MPG range than any other range when the data is divided into 10 breaks.

Most Common Range: The histogram shows that the most frequently observed MPG class is between 15 and #17.5 miles per gallon. This range has the highest bar, with 7 cars in this class, indicating that many #cars in the dataset are relatively less fuel-efficient.

General Distribution: The MPG distribution has a left-skewed shape, with the majority of cars clustered #in the lower MPG ranges (around 10 to 20 MPG). This suggests that many of the cars in the mtcars dataset #are not highly fuel-efficient by modern standards.

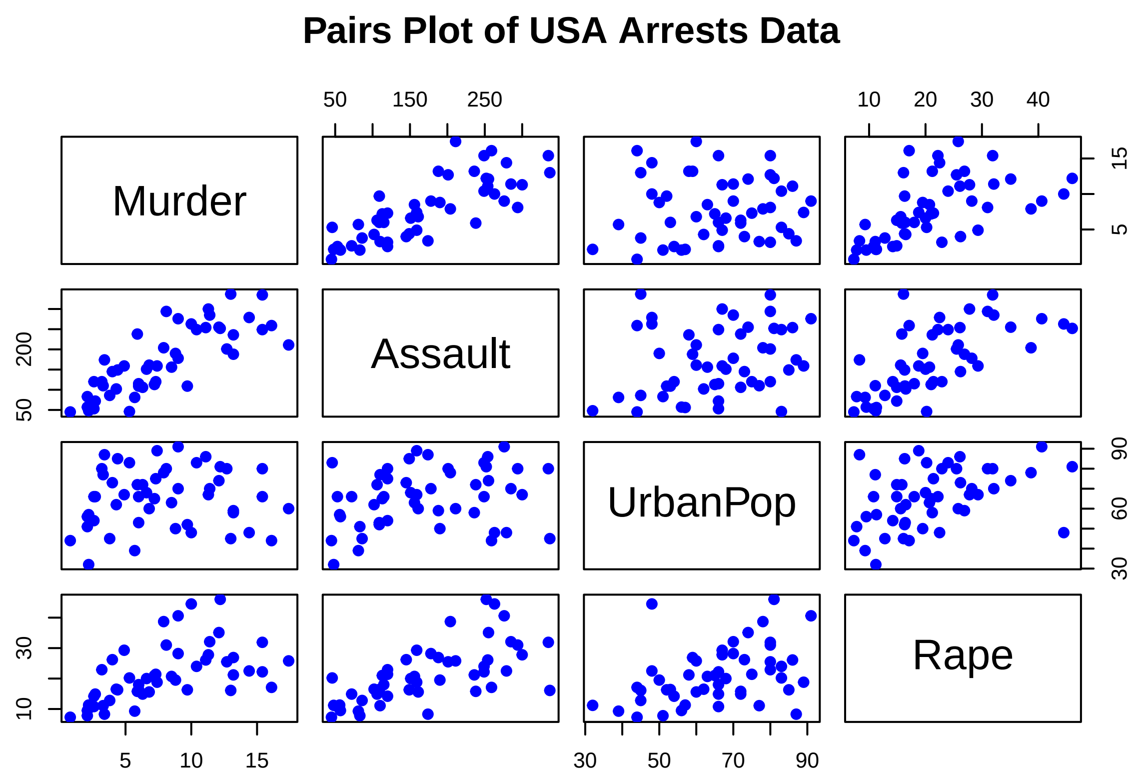
Other Observations: The next most populated bins are between 17.5-20 and 12.5-15 MPG, each containing 6 cars. There are a few cars with higher MPG values, in the ranges of 25-27.5 and 30-32.5, but these are less #common. No cars in this dataset achieve MPG in the range of 27.5-30. Fuel Efficiency Trend: Given that the dataset mostly represents cars with MPG values under 20, it could #suggest that the mtcars dataset includes many older or larger vehicles, which tend to consume more fuel #compared to modern, more efficient vehicles. Overall, the histogram indicates a concentration of cars with lower MPG values, with very few cars reaching above 25 MPG.

c. Using the **USArrests** data set, create a pairs plot to display the correlations between the variables in the data set. Plot the scatter plot with **Murder** and **Assault**. Give labels to the title, x-axis, and y-axis on the graph. Write a paragraph to summarize your results from both plots.

```r
# Load the data set
data("USArrests")

# Head of the data set
head(USArrests)
```

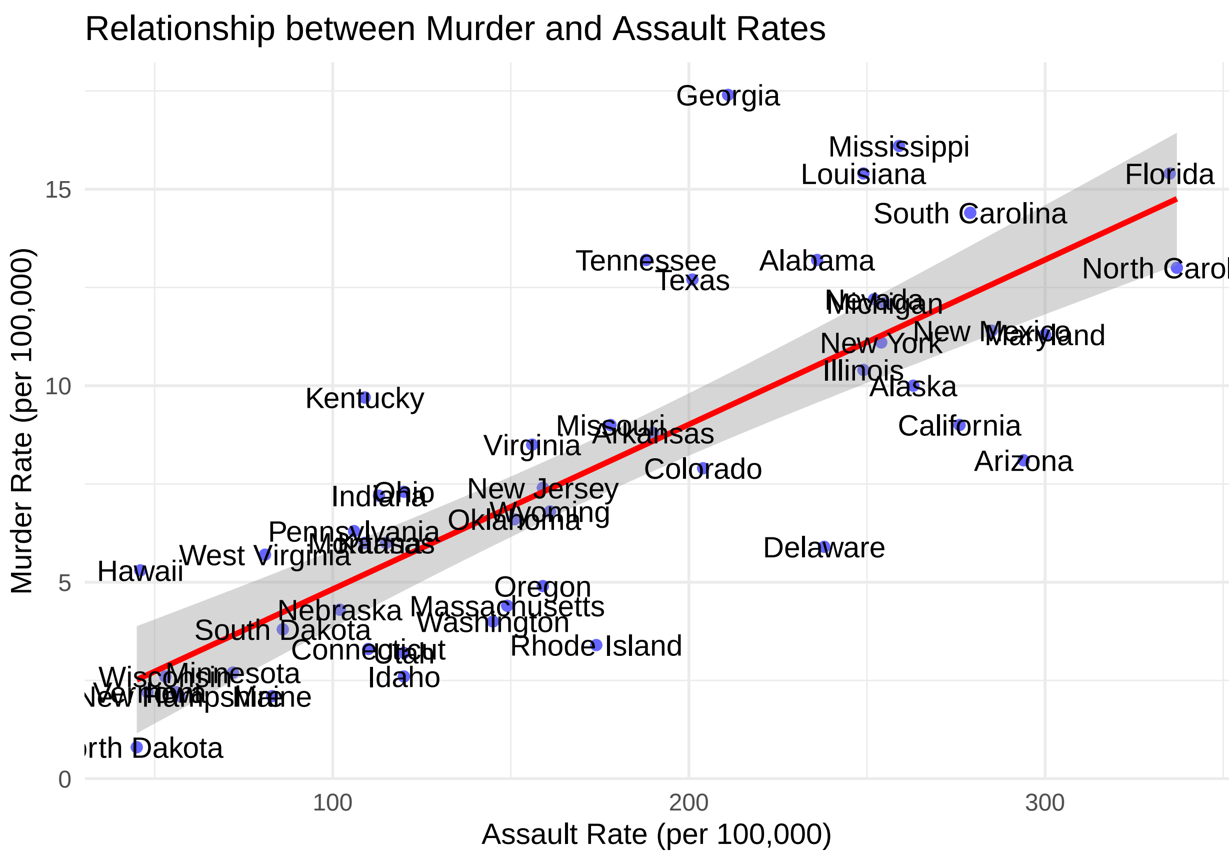
```
##            Murder Assault UrbanPop Rape
## Alabama      13.2     236       58 21.2
## Alaska       10.0     263       48 44.5
## Arizona       8.1     294       80 31.0
## Arkansas      8.8     190       50 19.5
## California    9.0     276       91 40.6
## Colorado      7.9     204       78 38.7
```

```r
# Enter your code here!

# Load required libraries
library(ggplot2)

# Create pairs plot using base R
pairs(USArrests,
      main="Pairs Plot of USA Arrests Data",
      pch=19,
      col="blue")
```




```r
# Create scatter plot for Murder vs. Assault using ggplot2
ggplot(USArrests, aes(x=Assault, y=Murder)) +
  geom_point(color="blue", alpha=0.6) +
  geom_smooth(method=lm, color="red", se=TRUE) +
  geom_text(aes(label=rownames(USArrests))) +
  labs(title="Relationship between Murder and Assault Rates",
       x="Assault Rate (per 100,000)",
       y="Murder Rate (per 100,000)") +
  theme_minimal()
```

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



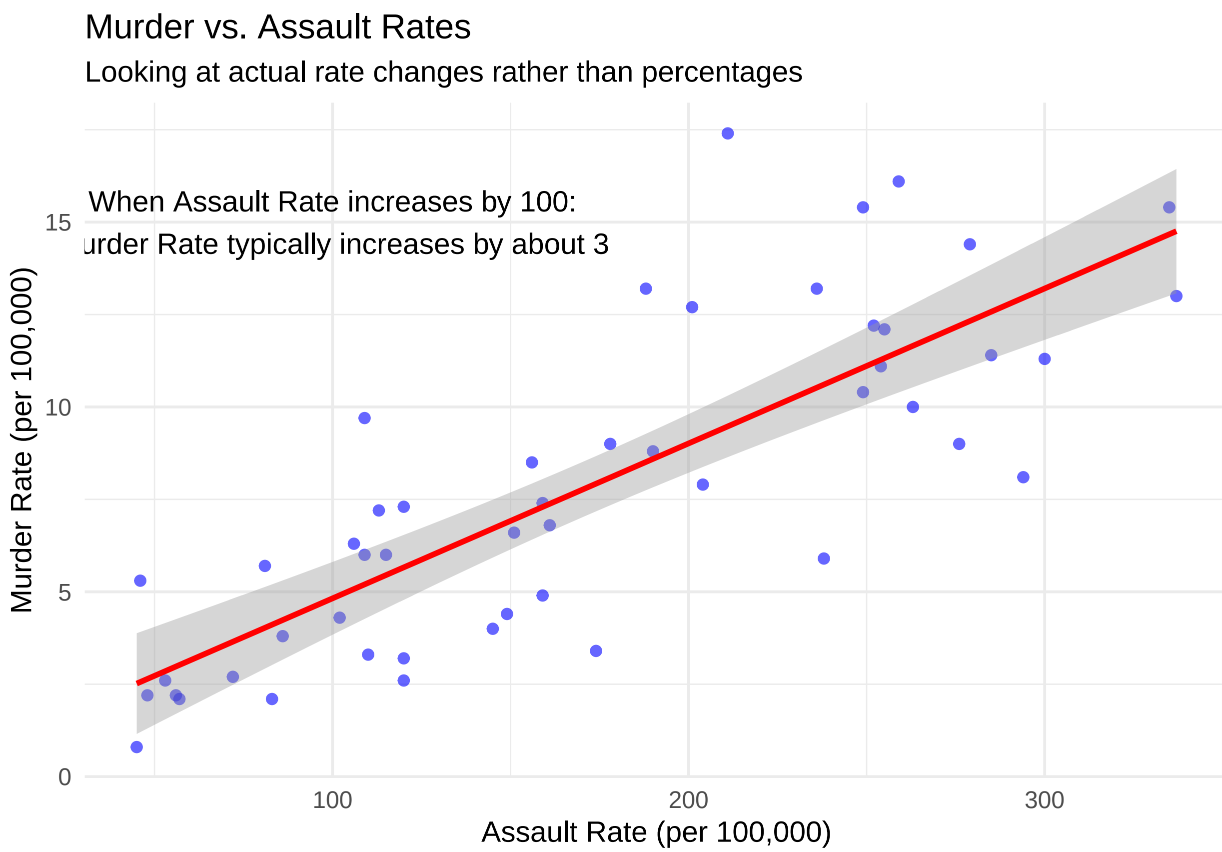

```
# Calculate correlation
correlation <- cor(USArrests$Murder, USArrests$Assault)
print(paste("Correlation between Murder and Assault:", round(correlation, 3)))
```

```
## [1] "Correlation between Murder and Assault: 0.802"
```

Scatter plots used to see and understand trend better

```
# Create plot with simpler interpretation
ggplot(USArrests, aes(x=Assault, y=Murder)) +
  # Add all states as points
  geom_point(color="blue", alpha=0.6) +
  # Add trend line
  geom_smooth(method=lm, color="red", se=TRUE) +
  # Add labels
  labs(title="Murder vs. Assault Rates",
       subtitle="Looking at actual rate changes rather than percentages",
       x="Assault Rate (per 100,000)",
       y="Murder Rate (per 100,000)") +

  # Add annotation with actual rate changes
  annotate("text", x=100, y=15,
           label="When Assault Rate increases by 100:\nMurder Rate typically increases by about 3") +
  theme_minimal()
```

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




```r
# Calculate average rates for context
print(paste("Average Assault Rate:", round(mean(USArrests$Assault), 1)))
```

```
## [1] "Average Assault Rate: 170.8"
```

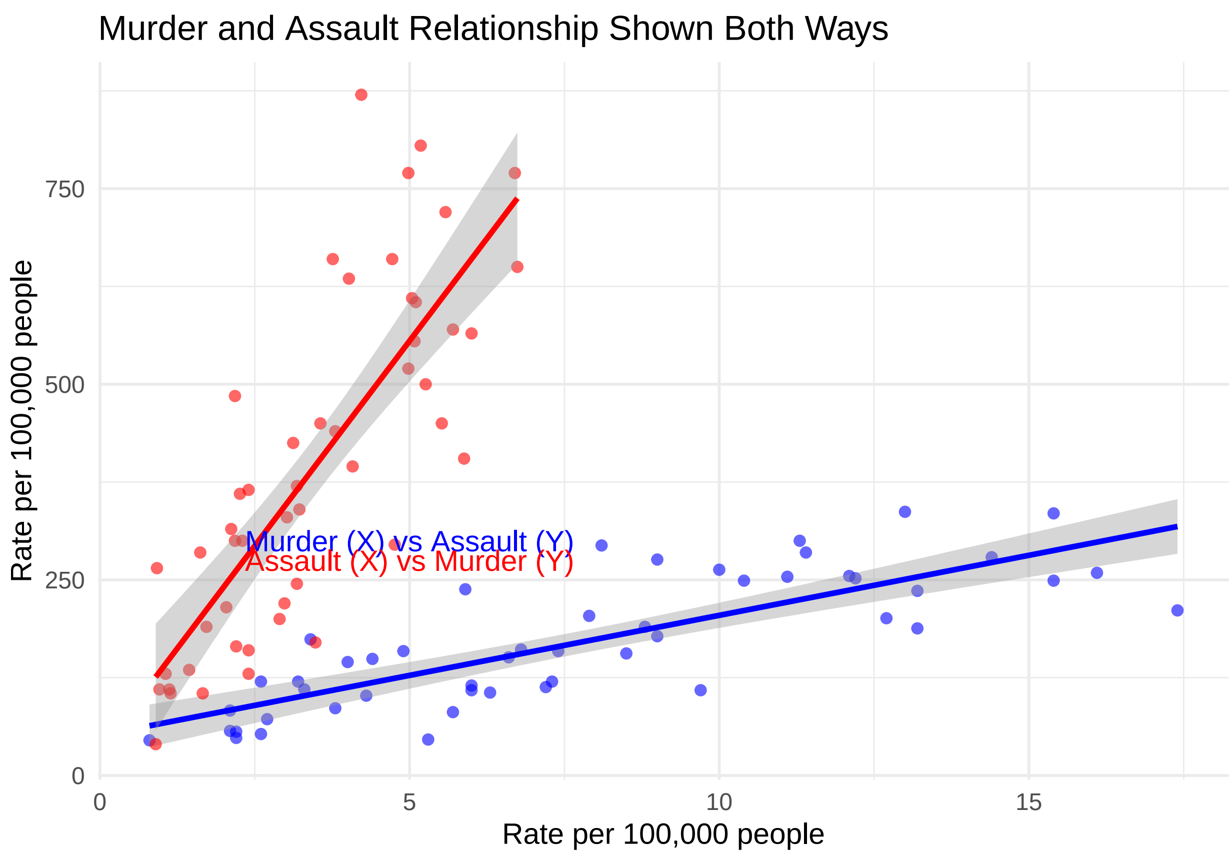
```r
print(paste("Average Murder Rate:", round(mean(USArrests$Murder), 1)))
```

```
## [1] "Average Murder Rate: 7.8"
```

```r
# Create plot comparing both ways of looking at Murder vs Assault
ggplot() +
  # Plot Murder vs Assault (one way)
  geom_point(data=USArrests, aes(x=Murder, y=Assault), color="blue", alpha=0.6) +
  geom_smooth(data=USArrests, aes(x=Murder, y=Assault), method=lm, color="blue", se=TRUE) +

  # Plot Assault vs Murder (other way, with transformed axes to match scale)
  geom_point(data=USArrests, aes(y=Murder*50, x=Assault/50), color="red", alpha=0.6) +
  geom_smooth(data=USArrests, aes(y=Murder*50, x=Assault/50), method=lm, color="red", se=TRUE) +

  # Add labels
  labs(title="Murder and Assault Relationship Shown Both Ways",
       x="Rate per 100,000 people",
```

```
       y="Rate per 100,000 people") +

  # Add legend
  annotate("text", x=5, y=300, color="blue", label="Murder (X) vs Assault (Y)") +
  annotate("text", x=5, y=275, color="red", label="Assault (X) vs Murder (Y)") +

  theme_minimal()
```

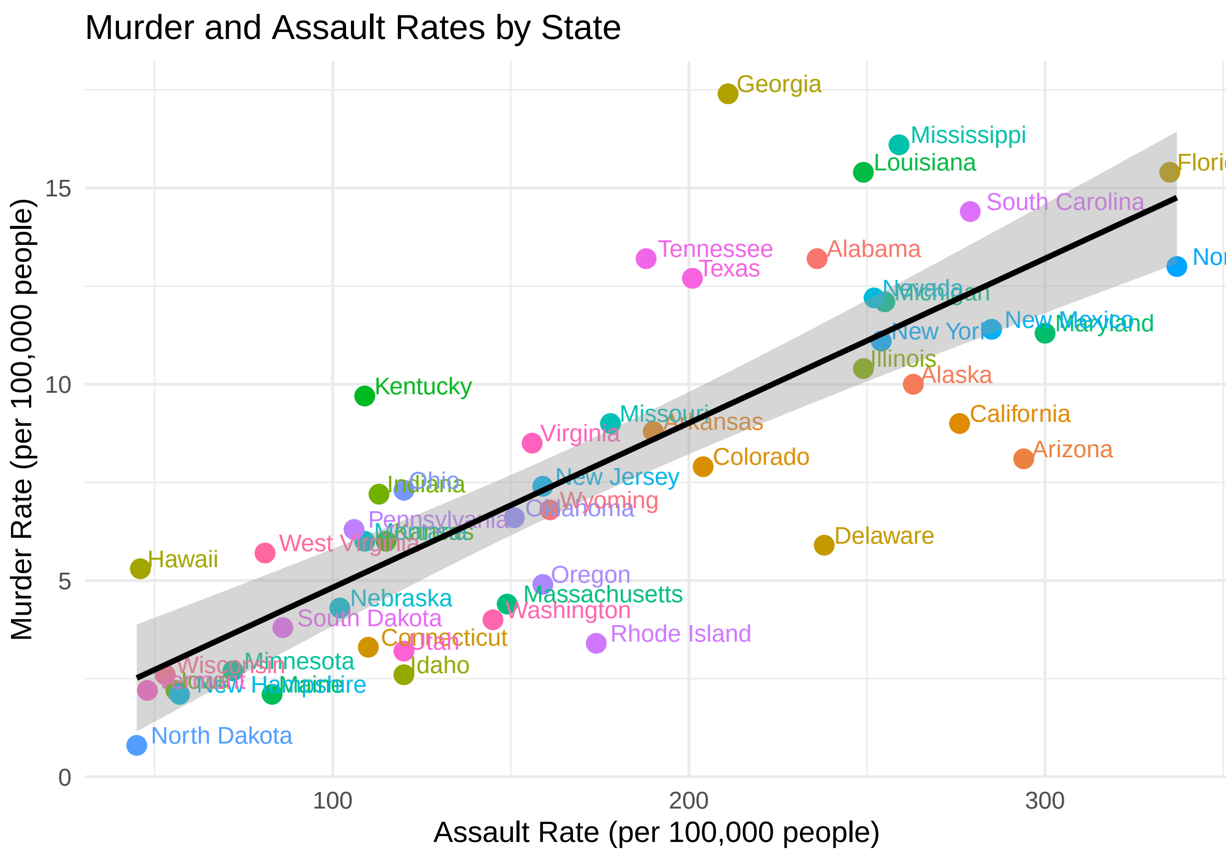
```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```




```
# Create scatter plot with all states colored and labeled
ggplot(USArrests, aes(x=Assault, y=Murder)) +
  # Add points with different colors for each state
  geom_point(aes(color=rownames(USArrests)), size=3) +
  # Add labels for each state
  geom_text(aes(label=rownames(USArrests), color=rownames(USArrests)),
            hjust=-0.1, vjust=-0.1, size=3) +
  # Add trend line
  geom_smooth(method=lm, color="black", se=TRUE) +
  # Add labels and title
  labs(title="Murder and Assault Rates by State",
       x="Assault Rate (per 100,000 people)",
       y="Murder Rate (per 100,000 people)",
```

```r
         color="State") +
  theme_minimal() +
  # Adjust legend position
  theme(legend.position="none")  # Remove legend as state names are on plot
```

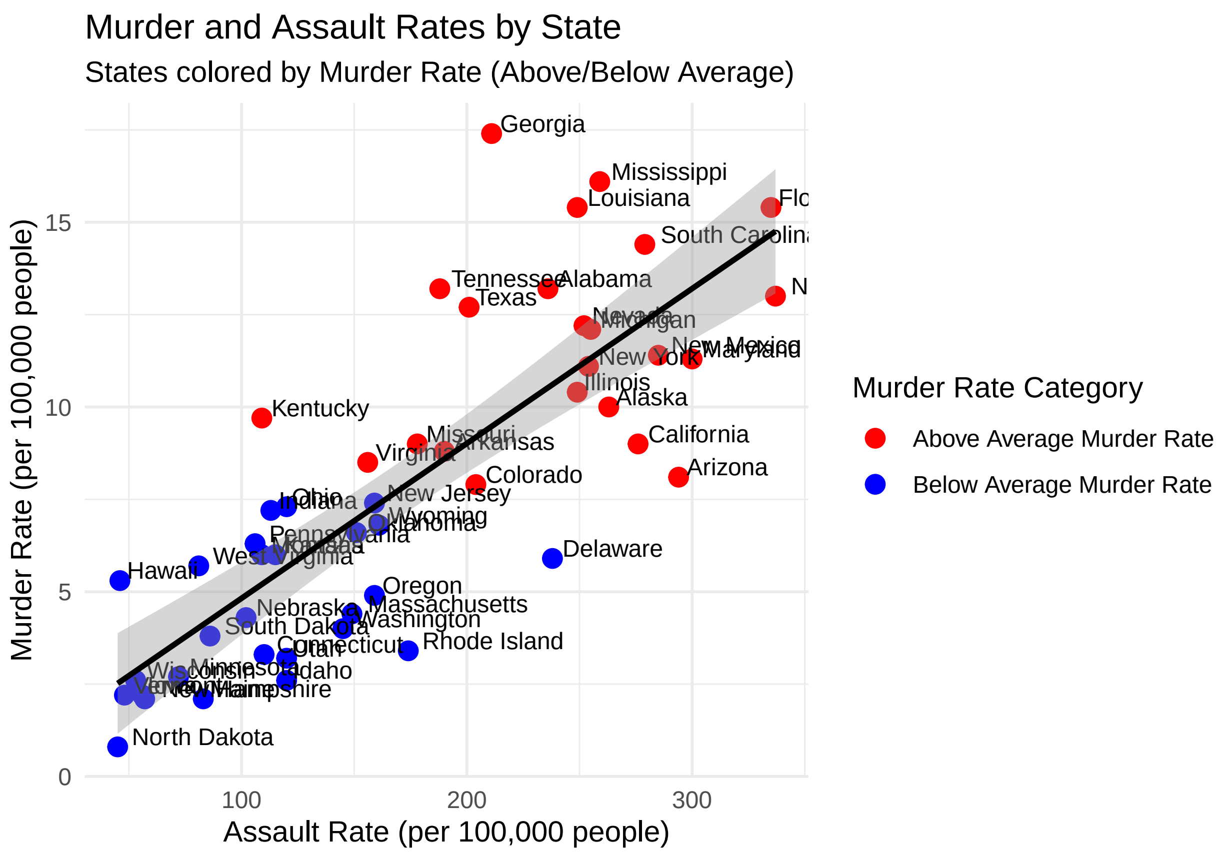
```
## `geom_smooth()` using formula = 'y ~ x'
```




```r
# We could also group states by region for better color coding:
ggplot(USArrests, aes(x=Assault, y=Murder)) +
  # Add points with colors by regions
  geom_point(aes(color=factor(ifelse(Murder > mean(Murder), "Above Average Murder Rate",
                                   "Below Average Murder Rate"))),
             size=3) +
  # Add labels
  geom_text(aes(label=rownames(USArrests)),
            hjust=-0.1, vjust=-0.1, size=3) +
  # Add trend line
  geom_smooth(method=lm, color="black", se=TRUE) +
  # Add labels and title
  labs(title="Murder and Assault Rates by State",
       subtitle="States colored by Murder Rate (Above/Below Average)",
       x="Assault Rate (per 100,000 people)",
       y="Murder Rate (per 100,000 people)",
       color="Murder Rate Category") +
```

```
theme_minimal() +
scale_color_manual(values=c("Above Average Murder Rate"="red",
                            "Below Average Murder Rate"="blue"))
```

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



Result:

=> Report a paragraph to summarize your findings from the plot! Based on the pairs plot, scatter plot, and the provided correlation coefficient, here's a comprehensive summary #of the findings:

The data visualization reveals several important patterns in the USArrests dataset:

The pairs plot shows relationships between all variables (Murder, Assault, UrbanPop, and Rape). Most notably, there is a strong positive correlation between Murder and Assault rates, with a correlation coefficient of 0.802. What is clear for the scatter plot is that when assault rates go up by 100 per 100,000 people the murder rate goes up by 3. This strong relationship is clearly visible in both the pairs plot and the dedicated scatter plot. The scatter plot with its fitted regression line (red) and confidence interval (grey band) demonstrates that as assault rates increase across states, murder rates tend to increase in a fairly predictable linear pattern. The narrow confidence band suggests this relationship is consistent and reliable.

Interestingly, the UrbanPop variable shows much weaker relationships with both Murder and Assault, as evidenced by the more scattered patterns in the pairs plot. This suggests that the level of urbanization in a state is not strongly related to these violent crime rates. The Rape variable shows moderate positive correlations with both Murder and Assault, but these relationships are not as strong as the Murder-Assault correlation.

Question 3

Download the housing data set from www.jaredlander.com and find out what explains the housing prices in New York City. Note: Check your working directory to make sure that you can download the data into the data folder.

a. Create your own descriptive statistics and aggregation tables to summarize the data set and find any meaningful results between different variables in the data set.

```
#Add library
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# Head of the cleaned data set
head(housingData)
```

```
##   Neighborhood Market.Value.per.SqFt      Boro Year.Built
## 1    FINANCIAL                200.00 Manhattan       1920
## 2    FINANCIAL                242.76 Manhattan       1985
## 4    FINANCIAL                271.23 Manhattan       1930
## 5      TRIBECA                247.48 Manhattan       1985
## 6      TRIBECA                191.37 Manhattan       1986
## 7      TRIBECA                211.53 Manhattan       1985
```

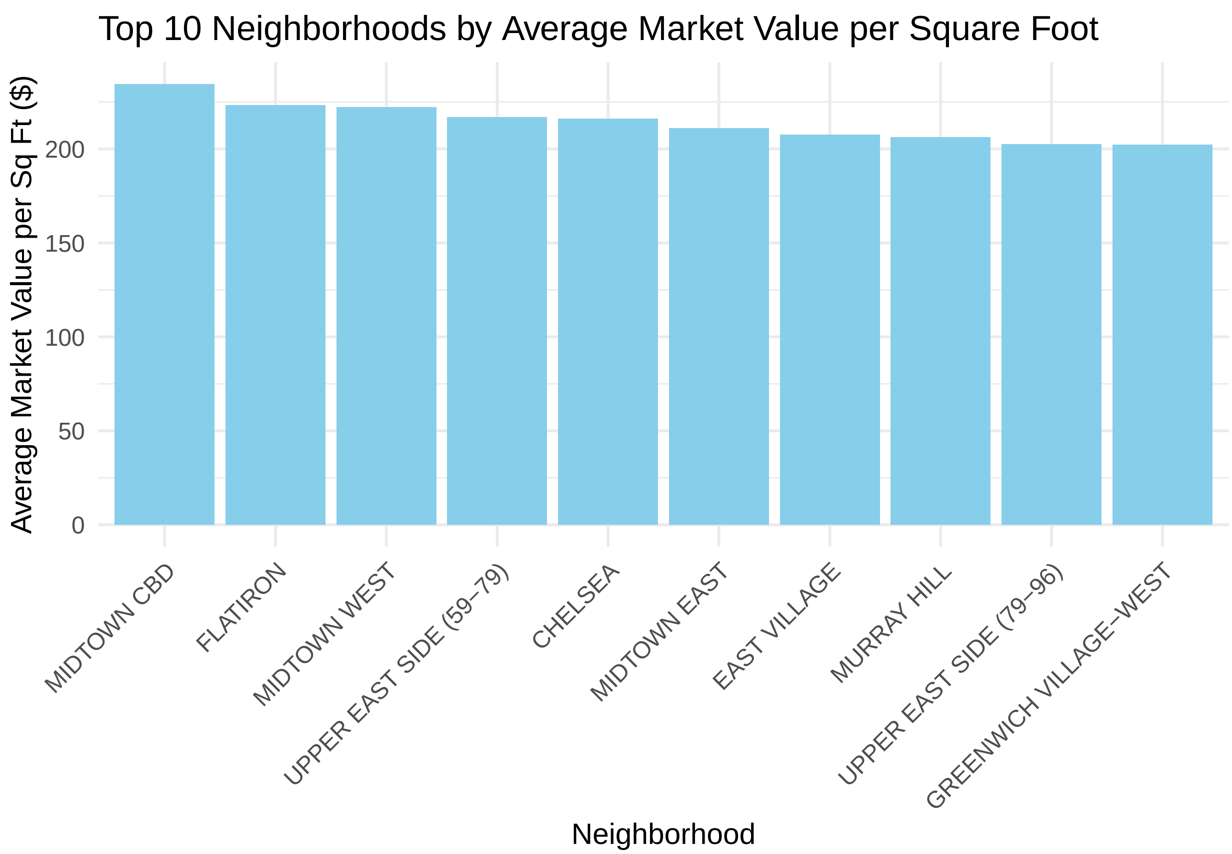
```
# Enter your code here!
# Calculate summary statistics by neighborhood
neighborhood_stats <- housingData %>%
  group_by(Neighborhood) %>%
  summarize(
    Average_Value = mean(Market.Value.per.SqFt),
    Median_Value = median(Market.Value.per.SqFt),
    Min_Value = min(Market.Value.per.SqFt),
    Max_Value = max(Market.Value.per.SqFt),
    Number_of_Properties = n()
  ) %>%
  arrange(desc(Average_Value))  # Sort by highest average value

# Print top 10 neighborhoods by average value
print("Top 10 Neighborhoods by Average Market Value per Square Foot:")
```

```
## [1] "Top 10 Neighborhoods by Average Market Value per Square Foot:"
```

```r
print(head(neighborhood_stats, 10))
```

```
## # A tibble: 10 x 6
##    Neighborhood            Average_Value Median_Value Min_Value Max_Value
##    <chr>                           <dbl>        <dbl>     <dbl>     <dbl>
##  1 MIDTOWN CBD                      234.         227.     180.       328.
##  2 FLATIRON                         223.         230.     171.       295.
##  3 MIDTOWN WEST                     222.         223.     159.       358.
##  4 UPPER EAST SIDE (59-79)          217.         218.     110.       301.
##  5 CHELSEA                          216.         214.      81.1      302.
##  6 MIDTOWN EAST                     211.         220.      98.7      334.
##  7 EAST VILLAGE                     207.         200.     148.       294.
##  8 MURRAY HILL                      206.         209.      24.7      264.
##  9 UPPER EAST SIDE (79-96)          202.         210.      70.0      262.
## 10 GREENWICH VILLAGE-WEST           202.         214.     113.       313.
## # i 1 more variable: Number_of_Properties <int>
```

```r
# Create a visualization of top 10 neighborhoods
ggplot(head(neighborhood_stats, 10),
       aes(x=reorder(Neighborhood, -Average_Value), y=Average_Value)) +
  geom_bar(stat="identity", fill="skyblue") +
  theme_minimal() +
  labs(title="Top 10 Neighborhoods by Average Market Value per Square Foot",
       x="Neighborhood",
       y="Average Market Value per Sq Ft ($)") +
  theme(axis.text.x = element_text(angle=45, hjust=1))
```




```r
# Calculate average value by borough
boro_stats <- aggregate(Market.Value.per.SqFt ~ Boro,
                        data = housingData,
                        FUN = mean)

# Look at trends by Year Built
year_stats <- aggregate(Market.Value.per.SqFt ~ Year.Built,
                        data = housingData,
                        FUN = mean)

# Create visualization
ggplot(housingData, aes(x=Year.Built, y=Market.Value.per.SqFt, color=Neighborhood)) +
  geom_point(size=3) +
  geom_smooth(method="lm", se=FALSE) +
  labs(title="Market Value per Square Foot by Year Built",
       x="Year Built",
       y="Market Value per Square Foot ($)") +
  theme_minimal()
```

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


```r
# Print summary statistics
print("Summary by Neighborhood:")
```

```
## [1] "Summary by Neighborhood:"
```

```r
print(neighborhood_stats)
```

```
## # A tibble: 148 x 6
##    Neighborhood             Average_Value Median_Value Min_Value Max_Value
##    <chr>                            <dbl>        <dbl>     <dbl>     <dbl>
##  1 MIDTOWN CBD                       234.         227.     180.       328.
##  2 FLATIRON                          223.         230.     171.       295.
##  3 MIDTOWN WEST                      222.         223.     159.       358.
##  4 UPPER EAST SIDE (59-79)           217.         218.     110.       301.
##  5 CHELSEA                           216.         214.      81.1      302.
##  6 MIDTOWN EAST                      211.         220.      98.7      334.
##  7 EAST VILLAGE                      207.         200.     148.       294.
##  8 MURRAY HILL                       206.         209.      24.7      264.
##  9 UPPER EAST SIDE (79-96)           202.         210.      70.0      262.
## 10 GREENWICH VILLAGE-WEST            202.         214.     113.       313.
## # i 138 more rows
## # i 1 more variable: Number_of_Properties <int>
```

```r
print("\nSummary by Borough:")
```

```
## [1] "\nSummary by Borough:"
```

```r
print(boro_stats)
```

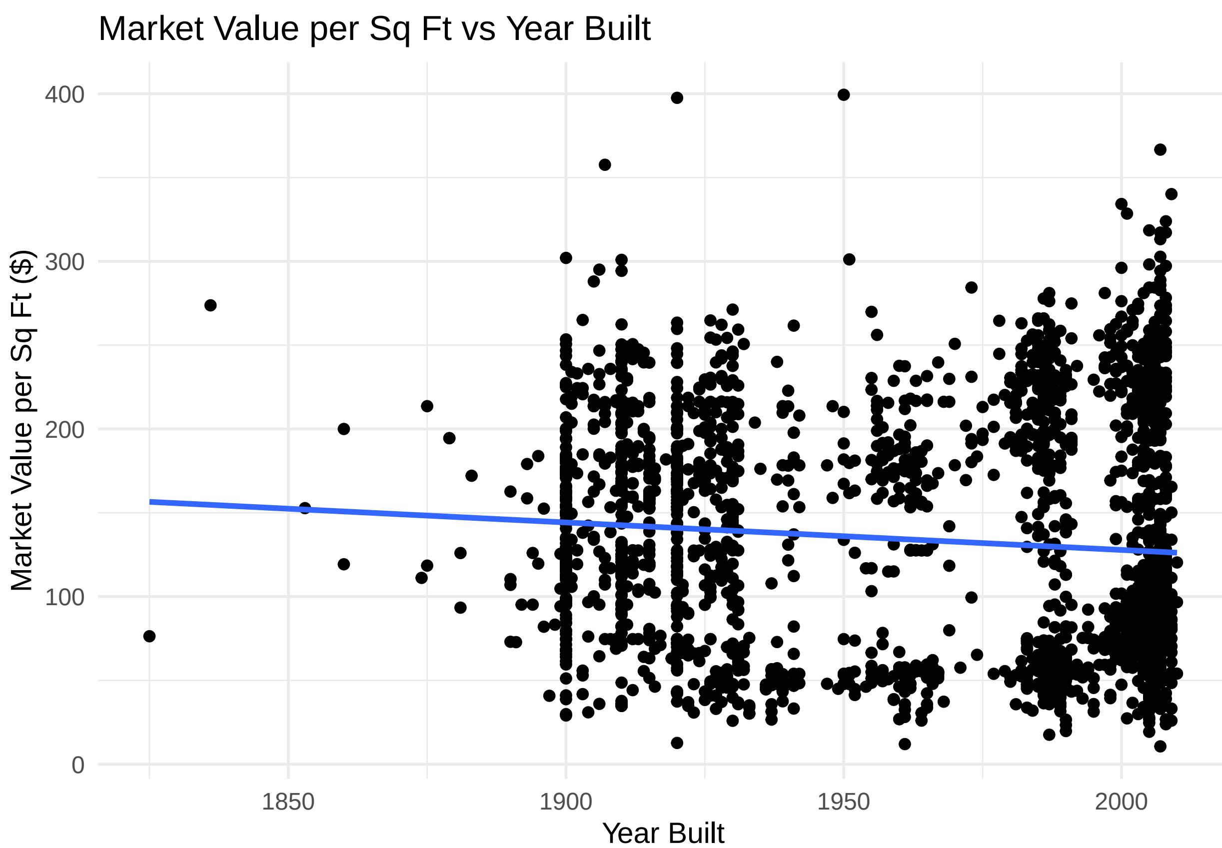
```
##            Boro Market.Value.per.SqFt
## 1         Bronx              47.93232
## 2      Brooklyn              80.13439
## 3     Manhattan             180.59265
## 4        Queens              77.38137
## 5 Staten Island              41.26958
```

Results: The neighborhood with the highest average market value per square foot is MIDTOWN CBD at $234.36. (Taken from top 10 neighborhoods..) The next highest are FLATIRON at $223.30, MIDTOWN WEST at $222.06, and UPPER EAST SIDE (59-79) at $216.84. (Taken from top 10 neighborhoods..) The neighborhoods with the lowest average values in the top 10 are GREENWICH VILLAGE-WEST at $202.14 and MURRAY HILL at $206.27. (Taken from top 10 neighborhoods..) There is a significant range in average values, with the top neighborhood MIDTOWN CBD over $30 higher per square foot than the bottom ranked GREENWICH VILLAGE-WEST. (Taken from top 10 neighborhoods..) The median values are generally close to the averages, indicating a relatively symmetric distribution of property values within each neighborhood. The minimum and maximum values show there is still a wide range of individual property values within each neighborhood, with the highest max being $399.39 in LOWER EAST SIDE, 19th down the list.

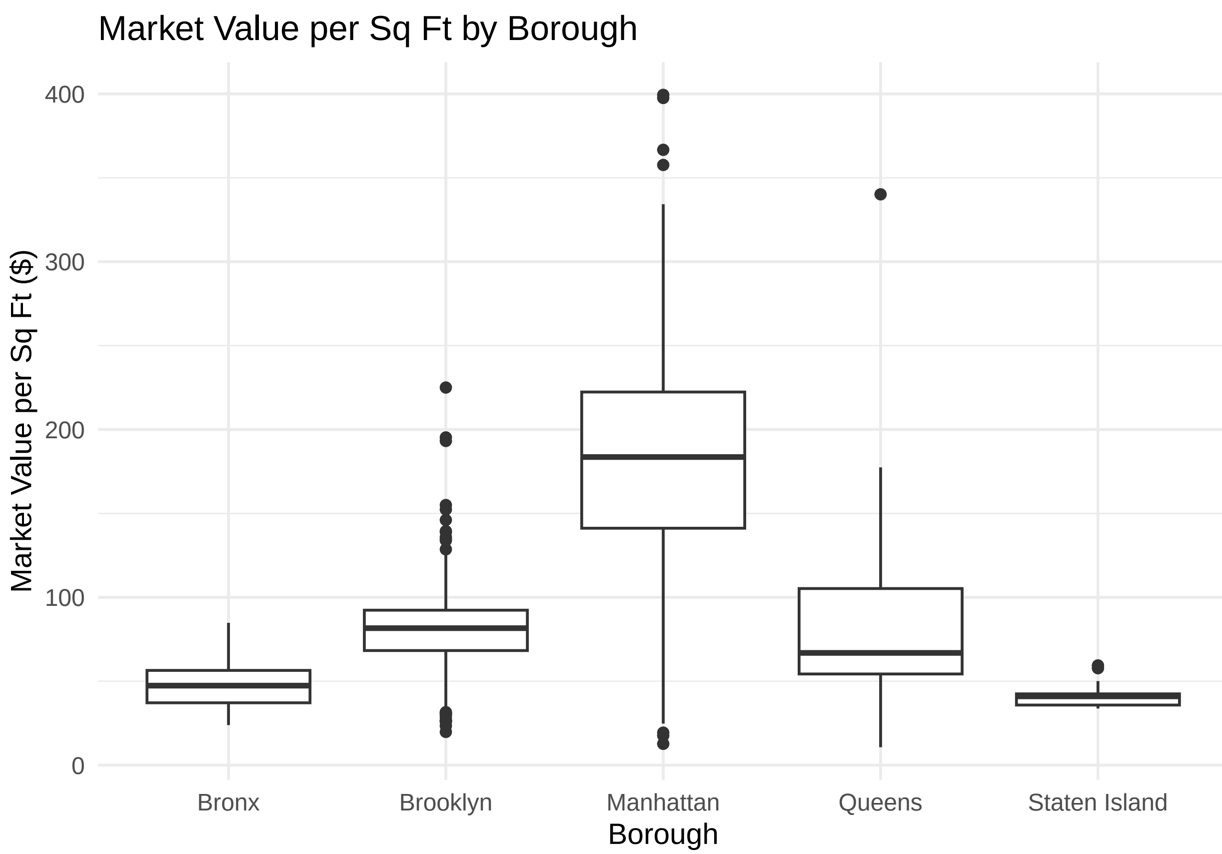
b. Create multiple plots to demonstrates the correlations between different variables. Remember to label all axes and give title to each graph.

```r
# Enter your code here!
# Load required packages
library(ggplot2)

# Scatterplot of Market Value per Sq Ft vs Year Built
ggplot(housingData, aes(x = Year.Built, y = Market.Value.per.SqFt)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(
    title = "Market Value per Sq Ft vs Year Built",
    x = "Year Built",
    y = "Market Value per Sq Ft ($)"
  ) +
  theme_minimal()
```

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

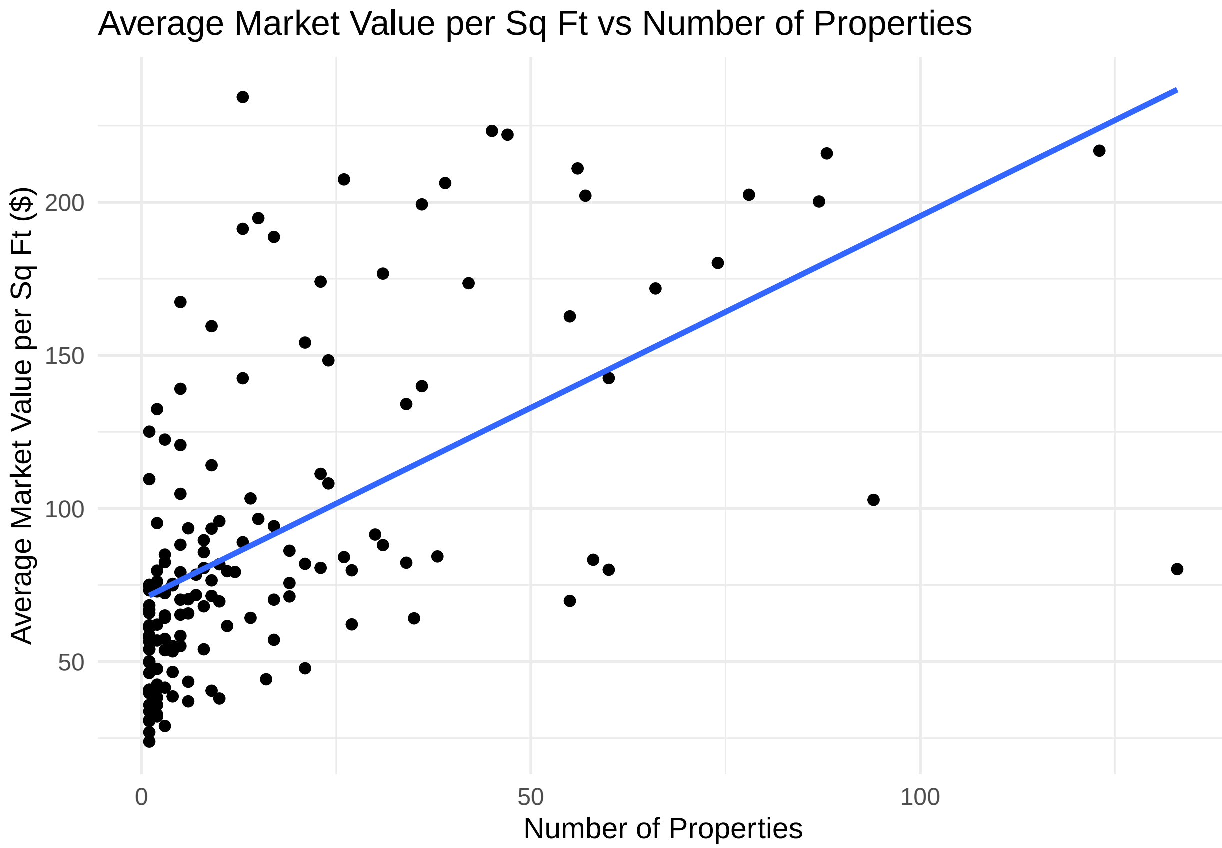



```r
# Boxplot of Market Value per Sq Ft by Boro
ggplot(housingData, aes(x = Boro, y = Market.Value.per.SqFt)) +
  geom_boxplot() +
  labs(
    title = "Market Value per Sq Ft by Borough",
    x = "Borough",
    y = "Market Value per Sq Ft ($)"
  ) +
  theme_minimal()
```




```
# Scatterplot of Market Value per Sq Ft vs Number of Properties
ggplot(neighborhood_stats, aes(x = Number_of_Properties, y = Average_Value)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(
    title = "Average Market Value per Sq Ft vs Number of Properties",
    x = "Number of Properties",
    y = "Average Market Value per Sq Ft ($)"
  ) +
  theme_minimal()
```

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



c. Write a summary about your findings from this exercise.

=> Enter your answer here! In summary, these visualizations help us understand the key relationships in the housing data, such as how property values are influenced by the year built, the differences in values across boroughs, and the connection between the number of properties in a neighborhood and the average market value per square foot. Manhattan and Brooklyn can be very expensive places to live, but new construction, although considered better does not hold well in value compared to older constructed buildings. By looking at box plot we can see there are outliers in both Brooklyn and Manhattan, as expected and well within range as low as under 50 the square foot - definately fixer uppers.

Not part of the summary:

Image 1: Market Value per Sq Ft vs Year Built This scatterplot shows the relationship between the market value per square foot and the year the property was built. The points represent individual properties, with the x-axis showing the year the property was built and the y-axis showing the market value per square foot. The scatterplot of Market Value per Sq Ft vs Year Built actually shows a downward trending line. This indicates that older properties, those built earlier - like brownstones, tend to have higher market values per square foot compared to newer properties. The downward sloping trend line suggests that there is an inverse relationship between the year a property was built and its market value per square foot. Older properties, built further in the past, appear to have higher per square foot values than more recently constructed properties. The data is clearly showing that newer construction does not necessarily equate to higher property values on a per square foot basis.

Image 2: Market Value per Sq Ft by Borough This is a box plot that compares the market value per square foot across different boroughs or neighborhoods. The horizontal line in the middle of each box represents the median value for that borough. The top and bottom of the boxes represent the 25th and 75th percentiles, showing the range of values within each borough. The outliers are shown as individual points above and below the boxes. For Bronx, Brooklyn, and Manhattan:

The median line being centered within the boxes indicates that the median market value per square foot in these boroughs is representative of the overall distribution. This suggests a relatively symmetric spread of property values, with half the values falling above the median and half below. The centered median implies these boroughs likely have a mix of both higher and lower-priced properties, without an extreme skew in either direction.

For Queens and Staten Island: In Queens, the median line is positioned towards the lower end of the box plot. This means the median market value per square foot in Queens is on the lower side of the overall range of values in that borough. The data is skewed, with more properties falling below the median than above it. For Staten Island, the median line is positioned towards the higher end of the box plot. This indicates the median market value in Staten Island is on the higher side of the value range in that borough. The data is skewed, with more properties valued above the median than below it.

The outliers in Brooklyn and Manhattan: The presence of numerous outlier points above and below the main box plots for these boroughs suggests a wider range of property values. These outliers represent properties with market values per square foot that are significantly higher or lower than the bulk of the data in those areas. The outliers imply Brooklyn and Manhattan have both very high-end and very low-end properties compared to the typical range, contributing to a greater overall spread of values.

In summary, the median positioning and outliers provide insights into the distribution and variability of property values within each borough. Queens and Staten Island exhibit more skewed distributions, while Bronx, Brooklyn, and Manhattan have more symmetric spreads, albeit with some extreme high and low outliers.

Image 3: Average Market Value per Sq Ft vs Number of Properties This scatterplot shows the relationship between the average market value per square foot and the number of properties in each neighborhood. Each point represents a different neighborhood, with the x-axis showing the number of properties and the y-axis showing the average market value per square foot for that neighborhood. The blue line represents a trend line, indicating a positive correlation - neighborhoods with more properties tend to have higher average market values per square foot. This suggests that neighborhoods with a larger number of properties may have more desirable characteristics that drive up the average property values.