Q1:

The `mean()` function operates on a numeric vector, providing the average value of the numbers within that vector. Meanwhile, the `sd()` function also works with numeric vectors and calculates the standard deviation, which serves as a measure of how the values in the vector are distributed or spread out. In our specific case, we employ these functions to analyze the "mpg" and "weight" columns of our dataset.

```
#HW2
data <- read.csv("C:/Users/faiya/OneDrive - Texas Tech University/Texas tech course/fall 23/software analytics//HW1/auto.csv")

#the mean and standard deviation of 'mpg'

mean_mpg <- mean(data$mpg)
std_dev_mpg <- sd(data$mpg)

#the mean and standard deviation of 'weight'
mean_weight <- mean(data$weight)
std_dev_weight <- sd(data$weight)

# Print the results
cat("Mean mpg:", mean_mpg, "\n")
cat("Standard Deviation mpg:", std_dev_mpg, "\n")
cat("Mean weight:", mean_weight, "\n")
cat("Standard Deviation weight:", std_dev_weight, "\n")
```

result

```
> # Print the results
> cat("Mean mpg:", mean_mpg, "\n")
Mean mpg: 23.44592
> cat("Standard Deviation mpg:", std_dev_mpg, "\n")
Standard Deviation mpg: 7.805007
> cat("Mean weight:", mean_weight, "\n")
Mean weight: 2977.584
> cat("Standard Deviation weight:", std_dev_weight, "\n")
Standard Deviation weight: 849.4026
```

Q2:

A custom function, `get_outlier`, was defined to detect outliers in a dataset. It accepts three arguments: the data column to analyze, its mean, and its standard deviation. Within the function, outliers are identified by comparing data points to the mean, marking them as TRUE if the absolute difference is ≥ 3 times the standard deviation. The function returns a logical vector, marking TRUE for outliers.

This function was applied individually to the "mpg," "weight," and "horsepower" columns, producing logical vectors (mpg_outliers, weight_outliers, horsepower_outliers) highlighting outlier data points. Car names associated with these outliers were extracted from the data and printed. If no outliers were found for a variable, "No factors" was printed.

```r
get_outlier <- function(column, mean_value, std_dev_value) {
  outliers <- abs(column - mean_value) >= 3 * std_dev_value
  return(outliers)
}

# Identify outliers in 'mpg' and 'weight' and horsepower
mpg_outliers <- get_outlier(data$mpg, mean_mpg, std_dev_mpg)
weight_outliers <- get_outlier(data$weight, mean_weight, std_dev_weight)
horsepower_outliers <- get_outlier(data$horsepower, mean_horsepower, std_dev_horsepower)


# names of the cars that are outliers in each category
outliers_weight_cars <- data$name[weight_outliers]
outliers_mpg_cars <- data$name[mpg_outliers]
outliers_horsepower_cars <- data$name[horsepower_outliers]



# Print the names of the cars that are outliers in each category
cat("Outliers on Weight:")
if (length(outliers_weight_cars) == 0) {
  cat(" No factors\n")
} else {
  cat(outliers_weight_cars, "\n")
}

cat("Outliers on MPG:")
if (length(outliers_mpg_cars) == 0) {
  cat(" No factors\n")
} else {
  cat(outliers_mpg_cars, "\n")
}
cat("Outliers on horsepower:")
if (length(outliers_horsepower_cars) == 0) {
  cat(" No factors\n")
} else {
  cat(outliers_horsepower_cars, "\n")
}

cat("Outliers on Horsepower:", outliers_horsepower_cars, "\n")
```

result:

```
> # Print the names of the cars that are outliers in each category
> cat("Outliers on Weight:")
Outliers on Weight:
> if (length(outliers_weight_cars) == 0) {
+   cat(" No factors\n")
+ } else {
+   cat(outliers_weight_cars, "\n")
+ }
 No factors
> cat("Outliers on MPG:")
Outliers on MPG:
> if (length(outliers_mpg_cars) == 0) {
+   cat(" No factors\n")
+ } else {
+   cat(outliers_mpg_cars, "\n")
+ }
 No factors
> cat("Outliers on horsepower:")
Outliers on horsepower:
> if (length(outliers_horsepower_cars) == 0) {
+   cat(" No factors\n")
+ } else {
+   cat(outliers_horsepower_cars, "\n")
+ }
29 30 54 240 243
```

Q3.

"Origin" and "cylinders" should be considered discrete random variables because they take on a finite and distinct set of values. In the context of the dataset, "origin" represents the country of origin for cars, typically categorized as 1 for the United States, 2 for Europe, and 3 for Asia, among other possibilities. These values are finite and predefined categories, making "origin" a discrete random variable. Similarly, "cylinders" denotes the number of cylinders in a car's engine, such as 4, 6, or 8, with a limited and countable range of possible values. Discrete random variables are characterized by distinct categories or

countable outcomes, and both "origin" and "cylinders" align with this definition, as they have specific, non-continuous values associated with them in the dataset.

In this process, we calculated the Probability Mass Function (PMF) for two discrete random variables, "cylinders" and "origin," we defined a reusable function called `calculate_pmf`. This function takes a data vector as its input and calculates the PMF based on the frequency of each unique value in the dataset. We then applied this `calculate_pmf` function to two different variables, "cylinders_data" and "origin_data," which represent the number of cylinders in cars and the country of origin for those cars, respectively. By passing these variables as arguments to the function, we obtained their respective PMFs.

By encapsulating the PMF calculation in a function and applying it to various variables or datasets, we can efficiently analyze and visualize the probability distributions of discrete random variables in our data.

```
}
# Calculate the PMF
calculate_pmf <- function(data) {
  pmf <- table(data) / length(data)
  return(pmf)
}

#cylinder
cylinders_data <- data$cylinders
#origin
origin_data <- data$origin
# Calculate the PMF using the function
pmf_cylinders <- calculate_pmf(cylinders_data)
pmf_origin <-calculate_pmf(origin_data)

# Print the PMF
cat("Probability Mass Function (PMF) for 'cylinders':\n")
print(pmf_cylinders)
cat("Probability Mass Function (PMF) for 'origins':\n")
print(pmf_origin)
```

result

```
> # Calculate the PMF
> calculate_pmf <- function(data) {
+   pmf <- table(data) / length(data)
+   return(pmf)
+ }
> #cylinder
> cylinders_data <- data$cylinders
> #origin
> origin_data <- data$origin
> # Calculate the PMF using the function
> pmf_cylinders <- calculate_pmf(cylinders_data)
> pmf_origin <-calculate_pmf(origin_data)
> # Print the PMF
> cat("Probability Mass Function (PMF) for 'cylinders':\n")
Probability Mass Function (PMF) for 'cylinders':
> print(pmf_cylinders)
data
          3           4           5           6           8
0.010204082 0.507653061 0.007653061 0.211734694 0.262755102
> cat("Probability Mass Function (PMF) for 'origins':\n")
Probability Mass Function (PMF) for 'origins':
> print(pmf_origin)
data
        1         2         3
0.6250000 0.1734694 0.2015306
>
```

Q4:

"mpg" and "weight" should be considered continuous random variables because they can take on a wide range of real-number values within a specific interval. In the dataset provided, these variables represent measurements that can vary continuously, with no distinct, countable set of possible values. For example, "mpg" can take on any real number within the range of possible fuel efficiency values, and "weight" can vary continuously across a range of vehicle weights. Unlike discrete variables, which have

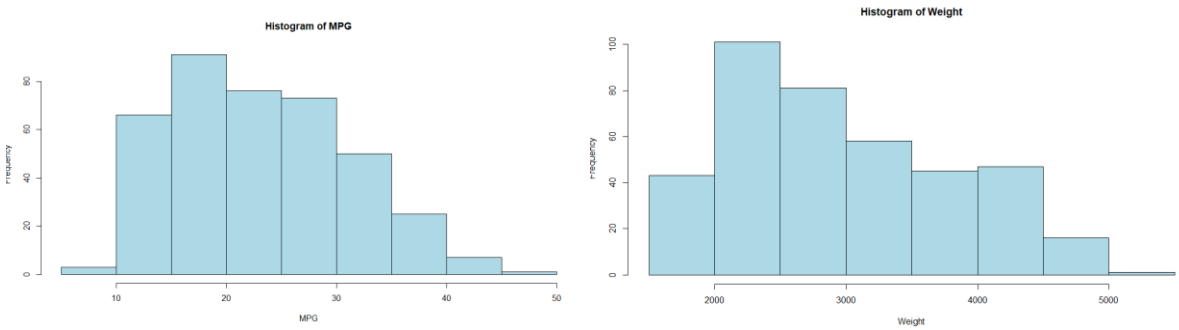
distinct and countable values, these variables can assume an infinite number of potential values within their respective continuous domains.

Q5:

The `hist()` function is a fundamental tool for visualizing data distributions by dividing them into intervals and displaying the frequency of data points within each interval, offering a graphical representation of data distribution. In R, histogram function takes a numeric vector (in this case, `data$mpg`) as input and constructs a histogram. Key arguments are used to customize the plot: `main` sets the title as "Histogram of MPG," `xlab` assigns the x-axis label as "MPG," `ylab` designates the y-axis label as "Frequency" to denote the data point counts in each bin, and `col` selects the color for the histogram bars, which is set to "lightblue" in this instance.

```
# Create a histogram for 'mpg'
hist(data$mpg, main = "Histogram of MPG", xlab = "MPG", ylab = "Frequency", col = "lightblue")

# Create a histogram for 'weight'
hist(data$weight, main = "Histogram of Weight", xlab = "Weight", ylab = "Frequency", col = "lightblue")
```



The mpg histogram is not symmetrical, with the peak of the distribution shifted to the right. This suggests that the data does not follow a normal distribution to any significant extent. The weight histogram is unimodal, which means that there is a single peak. This suggests that the data is not normally distributed, the data right skewed which means that there are more cars with weight on the right side of the distribution than on the left side.

So, No, they dont follow normal distribution.

End----