Load the dataset auto.csv (given in the shared Google folder) into R. This dataset has some columns:

**name**: name of the car model (e.g., bmw 2002)

**origin**: where the car is produced. 1 = US, 2 = EU, 3 = Asia.

**mpg**: miles per gallon (the higher the better).

**weight**: weight in lbs

**model\_year**: the model year

**horsepower**: the engine power (measured in horsepower). The higher the stronger.

**cylinders**: the number of cylinders in the car engine. The higher the stronger.

To answer the below questions, we need to set the directory where the “auto.csv” file is located.

In my local machine, the code to set the working directory is

setwd(“C:/Users/cscha/OneDrive/Desktop/Software analytics”)

Now read the “auto.csv” file as typing the following code

> cars <- read.csv(“auto.csv”)

Here, “cars” work as a variable to perform operations on the “auto.csv” file

Q1. What is the mean and standard deviation of mpg and weight?

The ‘R’ code to calculate the mean and standard deviation of mpg is as follows

> mean\_of\_mpg <- mean(cars$mpg)

> sd\_of\_mpg <- sd(cars$mpg)

The above code mean\_of\_mpg is the variable that stores the mpg values' mean. The mean() calculates the mean of the objects assigned to it. The above code mean(cars$mpg) calculates the mean of mpg values presented in the auto.csv file. Similarly, the sd() calculates the standard deviation of the objects assigned to it. The above code sd(cars$mpg) calculates the standard deviation of mpg values presented in the auto.csv file and stored in the ”sd\_of\_mpg” variable.

print(mean\_of\_mpg), it prints the mean of mpg values of the auto.csv file

Output is: [1] 23.44592

Print(sd\_of\_mpg), prints the standard deviation of mpg values of the auto.csv file.

Output is: [1] 7.805007

The ‘R’; code to calculate the mean and standard deviation of weight is as follows

> mean\_of\_weight <- mean(cars$weight)

> sd\_of\_weight <- sd(cars$weight)

The above code mean\_of\_weight is the variable that stores the weight values' mean. The mean() calculates the mean of the objects assigned to it. The above code mean(cars$weight) calculates the mean of weight values presented in the auto.csv file. Similarly, the sd() calculates the standard deviation of the objects assigned to it. The above code sd(cars$weight) calculates the standard deviation of weight values presented in the auto.csv file and stored in the ”sd\_of\_weight” variable.

print(mean\_of\_weight), it prints the mean of weight values of the auto.csv file

Output is: [1] 2977.584

Print(sd\_of\_mpg), prints the standard deviation of mpg values of the auto.csv file.

Output is: [1] 849.4026

We consider outliers as values extended 3 standard deviations or more from the mean (e.g., |*x* - μ| ≥ 3σ).

Q2. What cars are outliers on weight, mpg, and horsepower?

To determine the outliers on weight, mpg, and horsepower, first, we consider the given criteria which is

(|*x* - μ| ≥ 3σ).

The above function helps to identify the outliers by comparing each data point's absolute deviation from the mean to three times the standard deviation.

From the above function, we can determine that ‘x’, represents the object such as the mpg column or weight column fromthe “auto.csv” file, ‘μ’ represents the mean of the respective object, and ‘3σ´represents the standard deviation of the respective object which is multiplied by three.

The ‘R’ code to find the outliers of mpg is

> outliers\_of\_mpg <- cars[abs(cars$mpg - mean\_of\_mpg) >= 3\*sd\_of\_mpg,]

Here, from the above question we consider the mean of mpg by referencing mean\_of\_mpg <- mean(cars$mpg) where this line computes the mean (average) of the "mpg" column in the dataset “auto.csv” file.

> sd\_of\_mpg <- sd(cars$mpg), This line computes the "mpg" column's standard deviation for the “auto.csv” dataset and stores the result in the variable sd\_of\_mpg. The column of data points' spread, or dispersion is measured by the standard deviation.

abs(cars$mpg – mean\_of\_mpg) >= 3 \* sd\_of\_mpg. The absolute difference between each "mpg" value and the mean (mean\_of\_mpg) is determined by this section of the code. The next step is to determine if this absolute difference exceeds or equals three times the standard deviation (sd\_of\_mpg). A data point is an outlier in the "mpg" column if this criterion is met for that particular data point.

From the below line of code

outliers\_of\_mpg <- cars[abs(cars$mpg – mean\_of\_mpg) >= 3 \* sd\_of\_mpg, ]

Here a new data frame “outliers\_of\_mpg”, is established that only contains the rows for which the criteria for identifying outliers are satisfied. The rows in the “auto.csv” file that have "mpg" values that are three or more standard deviations outside of the mean are included in this statement.

The rows of the dataset where the "mpg" values are deemed outliers based on the criteria of being 3 or more standard deviations from the mean will therefore be contained in “outliers\_of\_mpg”.

The outliers\_of\_mpg result is:

[1] mpg cylinders displacement horsepower weight

[6] acceleration model\_year origin name

The ‘R’ code to find the outliers of weight is

> outliers\_of\_weight <- cars[abs(cars$weight - mean\_of\_weight) >= 3\*sd\_of\_weight,]

Here, from the above question we consider the mean of weight by referencing mean\_of\_weight <- mean(cars$weight) where this line computes the mean (average) of the "weight" column in the dataset “auto.csv” file.

> sd\_of\_weight <- sd(cars$weight), This line computes the "weight" column's standard deviation for the “auto.csv” dataset and stores the result in the variable sd\_of\_weight. The column of data points' spread, or dispersion is measured by the standard deviation.

abs(cars$weight – mean\_of\_weight) >= 3 \* sd\_of\_weight. The absolute difference between each "weight" value and the mean (mean\_of\_weight) is determined by this section of the code. The next step is to determine if this absolute difference exceeds or equals three times the standard deviation (sd\_of\_weight). A data point is an outlier in the "weight" column if this criterion is met for that particular data point.

From the below line of code

outliers\_of\_weight <- cars[abs(cars$weight – mean\_of\_weight) >= 3 \* sd\_of\_weight, ]

Here a new data frame “outliers\_of\_weight”, is established that only contains the rows for which the criteria for identifying outliers are satisfied. The rows in the “auto.csv” file that have "weight" values that are three or more standard deviations outside of the mean are included in this statement.

The rows of the dataset where the "weight" values are deemed outliers based on the criteria of being 3 or more standard deviations from the mean will therefore be contained in “outliers\_of\_weight”.

The “outliers\_of\_weight” result is:

[1] mpg cylinders displacement horsepower weight

[6] acceleration model\_year origin name

To identify the outliers of horsepower we first need to calculate the mean and standard deviation of horsepower

The ‘R’ code to calculate the mean and standard deviation of horsepower is

> mean\_of\_horsepower <- mean(cars$horsepower)

> sd\_of\_horsepower <- sd(cars$horsepower)

>print(mean\_of\_horsepower)

The above line prints the mean of horsepower. The result is

[1] 104.4694

>print(sd\_of\_horsepower)

The above line prints the standard deviation of horsepower. The result is

[1] 38.49116

The ‘R’ code to find the outliers of horsepower is

> outliers\_of\_horsepower <- cars[abs(cars$horsepower - mean\_of\_horsepower) >= 3\*sd\_of\_horsepower,]

Here, from the above question we consider the mean of horsepower by referencing mean\_of\_horsepower <- mean(cars$horsepower) where this line computes the mean (average) of the "horsepower" column in the dataset “auto.csv” file.

> sd\_of\_horsepower <- sd(cars$horsepower), This line computes the "horsepower" column's standard deviation for the “auto.csv” dataset and stores the result in the variable sd\_of\_horsepower. The column of data points' spread, or dispersion is measured by the standard deviation.

abs(cars$horsepower – mean\_of\_horsepower) >= 3 \* sd\_of\_horsepower. The absolute difference between each "horsepower" value and the mean (mean\_of\_horsepower) is determined by this section of the code. The next step is to determine if this absolute difference exceeds or equals three times the standard deviation (sd\_of\_horsepower). A data point is an outlier in the "horsepower" column if this criterion is met for that particular data point.

From the below line of code

outliers\_of\_horsepower <- cars[abs(cars$horsepower – mean\_of\_horsepower) >= 3 \* sd\_of\_horsepower, ]

Here a new data frame “outliers\_of\_horsepower”, is established that only contains the rows for which the criteria for identifying outliers are satisfied. The rows in the “auto.csv” file that have "horsepower" values that are three or more standard deviations outside of the mean are included in this statement.

The rows of the dataset where the "horsepower" values are deemed outliers based on the criteria of being 3 or more standard deviations from the mean will therefore be contained in “outliers\_of\_horsepower”.

The “outliers\_of\_horsepower” result is:

mpg cylinders displacement horsepower weight acceleration model\_year origin

43 12 8 455 225 4951 11.0 73 1

44 14 8 455 225 3086 10.0 70 1

79 14 8 454 220 4354 9.0 70 1

312 14 8 455 225 4425 10.0 70 1

317 16 8 400 230 4278 9.5 73 1

name

43 buick electra 225 custom

44 buick estate wagon (sw)

79 chevrolet impala

312 pontiac catalina

317 pontiac grand prix

The outlier cars are “buick electra 225 Custom”, “buick estate wagon (sw)”, “chevrolet impala”,

“pontiac catalina”, and “pontiac grand prix”.

The method which is used to calculate the outliers for mpg, weight, and horsepower using |x - μ| ≥ 3σ , where

x is a data point, μ is the mean, and σ is the standard deviation is a widespread technique for finding the outliers in the given dataset.

Some of the reasons to choose this method are:

The above method is simple and robust to implement and more than three standard deviations away from the mean data points are regarded as exceptional and are most likely to be outliers.

In many statistical applications, the 3-sigma rule (3 standard deviations from the mean) is a commonly used cutoff point for identifying outliers. It is a common data analysis technique that is based on the characteristics of the normal distribution.

This method strikes a balance between sensitivity (finding potential outliers) and specificity (reducing false positives) by employing 3 standard deviations as a cutoff.

Q3. Explain why origin and cylindersshould be considered as discrete random variables. What are their probability mass functions?

From the concepts of statistical analysis, we can clearly mention that origin and cylinders are considered to be discrete random variables because they can count on only a limited number of unique values and have no relevant concept of “in-between” values. In other terms, they can be considered discrete random variables by considering the statement as, ‘they have a countable number of outcomes rather than being continuous variables that can take on any value with in a range’.

Considering the Origin:

The main concept of Origin represents the production location of a car.

For example, if the origin value is ‘1’, it is noted that the car has originated in the U.S.,

Likely origin with value ‘2’ represents the car has originated in EU and the origin value ‘3’ represents the car has originated in Asia.

From the above values we can note that origin can only take these 3 values and there are no more values in between them.

As a result, by considering the above statements we can state that ‘Origin’ is a discrete random variable with a probability mass function (PMF) which is used to assign probabilities to each of these three values.

Considering the Cylinders:

Cylinders here mainly represent the number of cylinders in a car’s engine.

The number of cylinders is a discrete random variable, just like the origin, because it can only take on integer values (like 4, 6, or 8), and it cannot have continuous or fractional values.

A car must have a specified number of cylinders; for example, it cannot have 5.5 cylinders.

As a result, cylinders are a discrete random variable with a Probability Mass Function that describes the probabilities associated with each conceivable integer value

The ‘R’ code to calculate the PMF for ‘Origin’ is as follows

> required\_origin<- cars$origin

> unique\_origin <- unique(required\_origin)

> number\_of\_cars <- nrow(cars)

> origin\_pmf <- table(required\_origin) / number\_of\_cars

> print(origin\_pmf)

From the above code, we can state that

> required\_origin<- cars$origin, This line extracts the origin column from the “auto.csv” file and store those value in the ‘required\_origin’ variable.

> unique\_origin <- unique(required\_origin), This line creates a variable called ‘unique\_origin’ and it has the values which are distinct with respect to the origin column of the “auto.csv” file

> number\_of\_cars <- nrow(cars), This line creates, and a variable called ‘number\_of\_cars’ where it has the calculated total number of cars from the “auto.csv” file.

> origin\_pmf <- table(required\_origin) / number\_of\_cars, By computing this line of code we get the PMF of Origin

Here it uses the table() function to count the frequency of each distinct value of ‘required\_origin’ variable and to determine the likelihood of each unique origin, it divides each count by ‘number\_of\_cars’ in the “auto.csv” file.

The results are stored in ‘origin\_pmf’ variable.

> print(origin\_pmf)

It prints the result and

The output is:

required\_origin

1 2 3

0.6250000 0.1734694 0.2015306

Here 1, 2, 3 represents the origin values where 1 is represented as U.S , 2 is represented as EU and 3 is represented as Asia. The respective outcomes are PMF for corresponding origin values. If we add all the values of origin\_pmf it is equal to ‘1’ which holds the property of PMF.

The ‘R’ code to calculate the PMF for ‘Cylinders’ is as follows

> required\_cylinders <- cars$cylinders

> unique\_cylinders <- unique(required\_cylinders)

> number\_of\_cars <- nrow(cars)

> cylinders\_pmf <- table(required\_cylinders) / number\_of\_cars

> print(cylinders\_pmf)

From the above code, we can state that

> required\_cylinders<- cars$cylinders, This line extracts the cylinders column from the “auto.csv” file and store those value in the ‘required\_cylinders’ variable.

> unique\_cylinders <- unique(required\_cylinders), This line creates a variable called ‘unique\_cylinders’ and it has the values which are distinct with respect to the cylinders column of the “auto.csv” file

> number\_of\_cars <- nrow(cars), This line creates, and a variable called ‘number\_of\_cars’ where it has the calculated total number of cars from the “auto.csv” file.

> cylinders\_pmf <- table(required\_cylinders) / number\_of\_cars, By computing this line of code we get the PMF of Cylinders

Here it uses the table() function to count the frequency of each distinct value of ‘required\_cylinders’ variable and to determine the likelihood of each unique cylinder, it divides each count by ‘number\_of\_cars’ in the “auto.csv” file.

The results are stored in the ‘cylinders\_pmf’ variable.

> print(cylinders\_pmf)

It prints the result and

The output is:

required\_cylinders

3 4 5 6 8

0.010204082 0.507653061 0.007653061 0.211734694 0.262755102

Here the value 3,4,5,6,8 are the distinct values of cylinders, and the below numeric value sin table represents the frequency of each cylinder value respectively. If we add all the values of cylinders\_pmf it is equal to ‘1’ which holds the property of PMF.

Q4. Explain why mpg and weightshould be considered as continuous random variables.

Here mpg refers to Miles per gallon and it states the number of miles a car can travel on one gallon of fuel.

Its values can range widely, including decimal numbers. The fuel economy of cars, for instance, could be 20.5 mpg, 30.2 mpg, or any other non-integer figure. Unlike discrete variables, which have distinct, separate categories, "mpg" does not. It is continuous instead, forming a range of conceivable values.

These values are continuous because there is no natural limitation on the number of decimal places they can have.

In practical terms, cars can have very similar but different mpg ratings, therefore this level of accuracy is required to accurately reflect the range in fuel efficiency.

Here weight is the term which defines the mass of the cars. The weight in general is measured in pounds or in kilograms. Wight can also possess the range of real-number values like in the case of ‘mpg’ and also it can have decimal values. When cars can weigh up to 2500 pounds or 2010,7 pounds then there is no limitation for these values which makes weight to bean a continuous random variable. From the present data set which is “auto.csv” when we look into weight, we have weights ranging from 2130 and it has also a weight value of 4190, which tells us there are values in between these ranges as we also have the weight more than 4190. This data shows that weight can be considered as a continuous random variable with no limit in the values which is restricted to a certain range.

At last, we can conclude that, as there are no perfect boundary limitations separating the values of mpg and weights, they are regarded as continuous random variables. Since from examining the values of weight and mpg from the “auto.csv” file we can state that mpg and weights can take on a wide range of real-number values in a specified period.

This continuous nature enables a more precise description of the fundamental properties of cars in terms of weight and fuel efficiency.

Q5. Draw a histogram of mpg and weight. Is this reasonable to assume that they follow normal distribution?

The ‘R’ code to draw a histogram for mpg is as follows:

hist(cars$mpg,

+ breaks= 20,

+ col='white',

+ border='black',

+ xlab='MPG',

+ ylab='Frequency',

+ main='MPG Distribution')

In the above code, hist is a function where the values such as cars$mpg are pointing to the mpg column of the “auto.csv” file. The number of bins or intervals into which the data range, in this case, the "mpg" values is divided in the histogram and is referred to as the "breaks." Every bin corresponds to a certain range of "mpg" values. In the above code, breaks = 20 which causes the data range of "mpg" values to be separated into bins of 20 equal widths.

Here xlab= ‘MPG’ represents the MPG values and ylab=’Frequency’ represents the frequency of each xlab individual value, main='MPG Distribution' states the title of the respective histogram.

The ‘R’ code to draw a histogram for weight is as follows:

> hist(cars$weight,

+ breaks=20,

+ col='blue',

+ border='black',

+ xlab='Weight',

+ ylab='Frequency',

+ main='Weight Distribution')

In the above code, hist is a function where the values such as cars$weight are pointing to the weight column of the “auto.csv” file. The number of bins or intervals into which the data range, in this case, the "weight" values is divided in the histogram and is referred to as the "breaks." Every bin corresponds to a certain range of "weight" values. In the above code, breaks = 20 which causes the data range of "weight" values to be separated into bins of 20 equal widths.

Here xlab= ‘Weight’ represents the Weight values and ylab=’Frequency’ represents the frequency of each xlab individual value, main=’Weight Distribution' states the title of the respective histogram.

The output of ‘MPG’ histogram is: The output of ‘Weight’ histogram is:

A graph of a number of columns

Description automatically generated with medium confidence A screenshot of a computer

Description automatically generated

By observing the above mpg and weight histograms through visual inspection we can note that the ‘mpg’ histogram outputs the rightly skewed with a rightward tail while the ‘weight’ histogram is notably long-tail skewed. This visual information presents us with some characteristics of normal distribution.

While a visual inspection is instructive, statistical tests like the Shapiro-Wilk test or Q-Q plots should be used for a more in-depth assessment to quantitatively determine normal distribution.

In ’R’ we have qqnorm() and qqline() functions.

By implementing them as below

such as

qqnorm(cars$mpg)

qqline(cars$mpg)

qqnorm(cars$weight)

qqline(cars$weight).

If the outcome appears to closely follow a straight line then the objects passed in the qqnorm() and qqline() are said to follow the normal distribution. In this case, the mpg and weight are the objects passed in qqnorm() and qqline() functions to check for normal distribution.

In general, histograms provide a general idea of how the data is distributed and the relationship between x-axis and y-axis. It is important to combine the visual analysis with statistical tests and Q-Q plots to decide whether ‘mpg’ and ‘weight’ follow a normal distribution.