Q1. Plot the function y = x3 - 2x + 1, for x from -10 to 10.

* The first step is to create a vector named ‘x’ which ranges from -10 to 10 with the difference of value ‘0.1’.
* The ‘R’ code for this sequence is “x = seq(-10, 10, by =0.1)”
* The output of the ‘x’ value is

[1] -10.0 -9.9 -9.8 -9.7 -9.6 -9.5 -9.4 -9.3 -9.2 -9.1 -9.0 -8.9

[13] -8.8 -8.7 -8.6 -8.5 -8.4 -8.3 -8.2 -8.1 -8.0 -7.9 -7.8 -7.7

[25] -7.6 -7.5 -7.4 -7.3 -7.2 -7.1 -7.0 -6.9 -6.8 -6.7 -6.6 -6.5

[37] -6.4 -6.3 -6.2 -6.1 -6.0 -5.9 -5.8 -5.7 -5.6 -5.5 -5.4 -5.3

[49] -5.2 -5.1 -5.0 -4.9 -4.8 -4.7 -4.6 -4.5 -4.4 -4.3 -4.2 -4.1

[61] -4.0 -3.9 -3.8 -3.7 -3.6 -3.5 -3.4 -3.3 -3.2 -3.1 -3.0 -2.9

[73] -2.8 -2.7 -2.6 -2.5 -2.4 -2.3 -2.2 -2.1 -2.0 -1.9 -1.8 -1.7

[85] -1.6 -1.5 -1.4 -1.3 -1.2 -1.1 -1.0 -0.9 -0.8 -0.7 -0.6 -0.5

[97] -0.4 -0.3 -0.2 -0.1 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7

[109] 0.8 0.9 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9

[121] 2.0 2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 3.0 3.1

[133] 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 4.0 4.1 4.2 4.3

[145] 4.4 4.5 4.6 4.7 4.8 4.9 5.0 5.1 5.2 5.3 5.4 5.5

[157] 5.6 5.7 5.8 5.9 6.0 6.1 6.2 6.3 6.4 6.5 6.6 6.7

[169] 6.8 6.9 7.0 7.1 7.2 7.3 7.4 7.5 7.6 7.7 7.8 7.9

[181] 8.0 8.1 8.2 8.3 8.4 8.5 8.6 8.7 8.8 8.9 9.0 9.1

[193] 9.2 9.3 9.4 9.5 9.6 9.7 9.8 9.9 10.0

* Now we need to calculate the values of ‘y’ with each corresponding value of ‘x’.
* The ‘R’ code to calculate ‘y’ value is “y <- x^3 - 2\*x +1”
* The output of the ‘y’ value is

[1] -979.000 -949.499 -920.592 -892.273 -864.536 -837.375 -810.784 -784.757

[9] -759.288 -734.371 -710.000 -686.169 -662.872 -640.103 -617.856 -596.125

[17] -574.904 -554.187 -533.968 -514.241 -495.000 -476.239 -457.952 -440.133

[25] -422.776 -405.875 -389.424 -373.417 -357.848 -342.711 -328.000 -313.709

[33] -299.832 -286.363 -273.296 -260.625 -248.344 -236.447 -224.928 -213.781

[41] -203.000 -192.579 -182.512 -172.793 -163.416 -154.375 -145.664 -137.277

[49] -129.208 -121.451 -114.000 -106.849 -99.992 -93.423 -87.136 -81.125

[57] -75.384 -69.907 -64.688 -59.721 -55.000 -50.519 -46.272 -42.253

[65] -38.456 -34.875 -31.504 -28.337 -25.368 -22.591 -20.000 -17.589

[73] -15.352 -13.283 -11.376 -9.625 -8.024 -6.567 -5.248 -4.061

[81] -3.000 -2.059 -1.232 -0.513 0.104 0.625 1.056 1.403

[89] 1.672 1.869 2.000 2.071 2.088 2.057 1.984 1.875

[97] 1.736 1.573 1.392 1.199 1.000 0.801 0.608 0.427

[105] 0.264 0.125 0.016 -0.057 -0.088 -0.071 0.000 0.131

[113] 0.328 0.597 0.944 1.375 1.896 2.513 3.232 4.059

[121] 5.000 6.061 7.248 8.567 10.024 11.625 13.376 15.283

[129] 17.352 19.589 22.000 24.591 27.368 30.337 33.504 36.875

[137] 40.456 44.253 48.272 52.519 57.000 61.721 66.688 71.907

[145] 77.384 83.125 89.136 95.423 101.992 108.849 116.000 123.451

[153] 131.208 139.277 147.664 156.375 165.416 174.793 184.512 194.579

[161] 205.000 215.781 226.928 238.447 250.344 262.625 275.296 288.363

[169] 301.832 315.709 330.000 344.711 359.848 375.417 391.424 407.875

[177] 424.776 442.133 459.952 478.239 497.000 516.241 535.968 556.187

[185] 576.904 598.125 619.856 642.103 664.872 688.169 712.000 736.371

[193] 761.288 786.757 812.784 839.375 866.536 894.273 922.592 951.499

[201] 981.000

* Now, we use the “Scatter plot” to plot all these ‘x’ and ‘y’' values.
* The ‘R’ code to plot all the ‘x’ and ‘y’ values is

plot(x,y, type="p", col="blue", xlab="x", ylab="y",

main="The Scatter plot of given equation y = x^3 -2\*x +1")

* Here in the above code, we plot ‘x’ and ‘y’ values by mentioning the following details:
* Type =” p” states, the plot is a “Scatter Plot”,
* “xlab “, and “ylab” describe the plot's x and y axis, respectively containing x values on the x-axis and y values on the y-axis.
* Here the attribute “main” describes the title of the plot and the “col” attribute states the color of the connecting line of the points.

Output of the Plot.

A screen shot of a computer screen

Description automatically generated

Q2. Given two vectors *x* and *y*. Write expressions in R to calculate their Euclidean distance and dot product.

From the given data, we have the function y = y = x3 - 2x + 1, for x from -10 to 10.

To calculate the values of vector ‘x’ we have the ‘R’ code as “x <- seq(-10, 10, by =1)”.

This code gives the vector ‘x’ values ranging from -10 to 10 with a difference of ‘1’.

The value of the ‘x’ vector is

[1] -10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8

[20] 9 10

To calculate the values of ‘y’ vector with all the ‘x’ vector values we have the following ‘R’ code as “y<- x^3-2\*x+1”

The values of the ‘y’ vector is

[1] -979 -710 -495 -328 -203 -114 -55 -20 -3 2 1 0 5 22 57

[16] 116 205 330 497 712 981

The general formula to calculate the Euclidean distance between two vectors x and y is

Here and are the components or elements of the vectors *x* and *y*. In other words, they are the values at the position in each vector.

The ‘R’ code for the present ‘x’ and ‘y’ vector values to calculate Euclidean distance is

euclidean\_distance <- sqrt(sum((x-y)^2))

The output is : [1] 1913.051

The general formula to calculate the Dot product of two vectors ‘x’ and ‘y’ is

The ‘R’ code to calculate the dot product is dot\_product <- sum(x\*y)

In this code section, we calculate the dot product between vectors ‘x’ and ‘y’ using the sum function and element-wise multiplication ( x\*y).

The output is: [1] 49126

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

Q3. Which car has the highest mpg?

The ‘R’ code to answer the above question is

> index <- which.max(cars$mpg)

> cars\_highest <- cars$name[index]

> print(cars\_highest)

Here, the mpg column in the “auto.csv” file is extracted by cars$mpg.

which.max() is a function that provides the index of a data frame's highest value. In this instance, it locates the row

index corresponding to the highest mpg value and assigns that row index to the variable “index”.

The name column in the “auto.csv” file is extracted by cars$name.

Using the row index found in the previous step, cars$name[index] returns the name of the car with the highest mpg. The variable “cars\_highest” holds the outcome.

> print(cars\_highest) prints the output.

Output: [1] "mazda glc"

“mazda glc” is the car which has the highest mpg value.

Q4. What is the average mpg of US cars, EU cars, and Asian cars? Are Asian cars more fuel-efficient?

To calculate the average mpg values of US cars we have following ‘R’ code

> avg\_us\_mpg <- mean (cars$mpg[cars$origin == 1])

Here avg\_us\_mpg variable stores the average mpg value of US cars.

In the above code, cars$mpg reads the mpg column in “auto.csv”.

cars$origin == 1, point to the cars which originated in the US.

cars$mpg[cars$origin == 1] gives us the mpg values of cars which are originated in the US and to calculate the

average of these mpg values we mention,” mean (cars$mpg[cars$origin == 1])” and store these values in

“avg\_us\_mpg”, variable.

> print(avg\_us\_mpg)

Output is: [1] 20.03347

To calculate the average mpg values of EU cars we have the following ‘R’ code

> avg\_europe\_mpg <- mean (cars$mpg[cars$origin == 2])

Here avg\_europe\_mpg variable stores the average mpg value of EU cars.

In the above code, cars$mpg reads the mpg column in “auto.csv”.

cars$origin == 2, points to the cars which originated in the EU.

cars$mpg[cars$origin ==2] gives us the mpg values of cars which are originated in the EU and to calculate the

average of these mpg values we mention,” mean (cars$mpg[cars$origin == 2])” and store these values in

“avg\_europe\_mpg”, variable.

> print(avg\_europe\_mpg)

Output is: [1] 27.60294

To calculate the average mpg values of Asian cars we have following ‘R’ code

> avg\_asia\_mpg <- mean (cars$mpg[cars$origin == 3])

Here avg\_asia\_mpg variable stores the average mpg value of Asian cars.

In the above code, cars$mpg reads the mpg column in “auto.csv”.

cars$origin == 3, point to the cars which originated in Asia.

cars$mpg[cars$origin == 3] gives us the mpg values of cars which are originated in Asia and to calculate the

average of these mpg values we mention,” mean (cars$mpg[cars$origin == 3])” and store these values in

“avg\_asia\_mpg”, variable.

> print(avg\_asia\_mpg)

Output is : [1] 30.45063

By comparing the average mpg values of US, EU, and Asian cars we can state which country cars are more

fuel-efficient. From considering the above outputs we can state that Asian cars are more fuel-efficient.

The ‘R’ code to support the above statement is

> if( avg\_asia\_mpg > avg\_us\_mpg && avg\_asia\_mpg > avg\_europe\_mpg){

+ print("Asian cars are more fuel-efficient")} else{

+ print("Asian cars are not more fiel-efficient")}

Here in if statement avg\_asia\_mpg is compared with avg\_us\_mpg and avg\_europe\_mpg values.

If, avg\_asia\_mpg is greater than avg\_us\_mpg and avg\_europe\_mpg we print Asian cars more fuel-efficient.

Output of above ‘R’ code is: [1] "Asian cars are more fuel-efficient"

Q5. Which car is strongest (highest horsepower

The ‘R’ code to answer the above question is

> index <- which.max(cars$horsepower)

> cars\_hp <- cars$name[index]

> print(cars\_hp)

Here, the horsepower column in the “auto.csv” file is extracted by cars$horsepower.

which.max() is a function that provides the index of a data frame's highest value. In this instance, it locates the row

index corresponding to the highest horsepower value and assigns that row index to the variable “index”

The name column in the “auto.csv” file is extracted by cars$name.

Using the row index found in the previous step, cars$name[index] returns the name of the car with the highest

horsepower value. The variable “cars\_hp” holds the outcome.

> print(cars\_hp) prints the output.

Output is [1] "pontiac grand prix"

The Pontiac grand prix car has the highest horsepower.

Q6. How many car models are in each year? Which year has the most models?

The ‘R’ code to get the car models in each year is

> models\_by\_year <- table(cars$model\_year)

> highest\_models <- names(models\_by\_year[which.max(models\_by\_year)])

The "model\_year" column, which contains the model years of the cars, is extracted from the “auto.csv” file by the

code,” cars$model\_year”.

A frequency table that counts the occurrences of each model\_year in the dataset is made using the table () method.

The “models\_by\_year” variable holds the outcome.

which.max(models\_by\_year) displays the year with the highest number of car models by locating the index of the

maximum count in the “models\_by\_year table”.

The “highest\_models”, variable is stored with the information obtained by the names() method from the table, which is returned as a string.

> print(modesl\_by\_year)

This prints the table with values of car models corresponding to each year.

Output is:

70 71 72 73 74 75 76 77 78 79 80 81 82

29 27 28 40 26 30 34 28 36 29 27 28 30

From above table we can say that in the year 73 we have 29 car models and the output continues the data with year

and models accordingly,

> print(highest\_models)

Prints the year with most models.

Output is: [1] "73"