**DAY-3 LAB ASSESMENT**

**INDU SEKHAR**

**192124123**

**1)** **1. (i) Write a function in R programming to print generate Fibonacci sequence using**

**Recursion in R**

**CODE:**

recurse\_fibonacci <- function(n) {

if(n <= 1) {

return(n)

} else {

return(recurse\_fibonacci(n-1) + recurse\_fibonacci(n-2))

}

}

# take input from the user

nterms = as.integer(readline(prompt="How many terms? "))

# check if the number of terms is valid

if(nterms <= 0) {

print("Plese enter a positive integer")

} else {

print("Fibonacci sequence:")

for(i in 0:(nterms-1)) {

print(recurse\_fibonacci(i))

}

}

**2)** **Find sum of natural numbers up-to 10, without formula using loop statement.**

**Code:** num = as.integer(readline(prompt = "Enter a number: "))

if(num < 0) {

print("Enter a positive number")

} else {

sum = 0

# use while loop to iterate until zero

while(num > 0) {

sum = sum + num

num = num - 1

}

print(paste("The sum is", sum))

}

**(iii)create a vector 1:10 and Find a square of each number and store that in a**

**separate list.**

**Code:**

vector <- 1:10

squared\_vector <- vector^2

cat("Vector:", vector, "\n")

cat("Squared Vector:", squared\_vector, "\n")

**2.    (motor trend car road test) comprises fuel consumption, performance and  10 aspects**

**of automobile**

**design for 32 automobiles. It comes pre-installed  with  package in R.**

**(i)Find the dimension of the dataset**

**(ii)Give the statistical summary of the features.**

**(iii)Print the categorical features in Dataset**

**(iv)Find the average weight(wt) grouped by Engine shape(vs)**

**(v)Find the largest and smallest value of the variable weight with respect to Engine shape**

**Code:**

**(i)**

mtcars<-mtcars

dim(motor\_trend)

**(ii)**

Summary(mtcars)

(iii)

categorical\_vars <- sapply(mtcars, is.factor)

mtcars[, categorical\_vars]

(iv)

aggregate(wt ~ vs, data=mtcars, mean)

(v)

aggregate(wt ~ vs, data=mtcars, min)

aggregate(wt ~ vs, data=mtcars, max)

**3.Use ggplot package to plot below EDA questions label the plot accordingly**

**(i)Create weight(wt) vs displacement(disp) scatter plot factor by  Engine Shape(vs)**

**(ii) Create horsepower(hp) vs mileage (mgp) scatter plot factor by  Engine Shape(vs)**

**(iv)In above plot , Separate columns according to cylinders(cyl) size**

**(v) Create histogram plot for horsepower (hp) with bin-width size of 5**

Code:

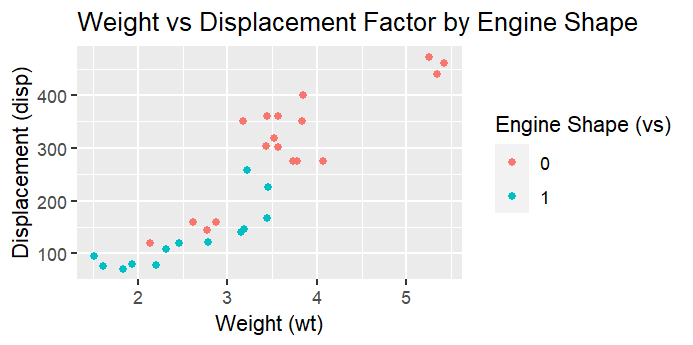
(i)ggplot(mtcars, aes(x = wt, y = disp, color = factor(vs))) +

geom\_point() +

labs(x = "Weight (wt)", y = "Displacement (disp)", color = "Engine Shape (vs)") +

ggtitle("Weight vs Displacement Factor by Engine Shape")

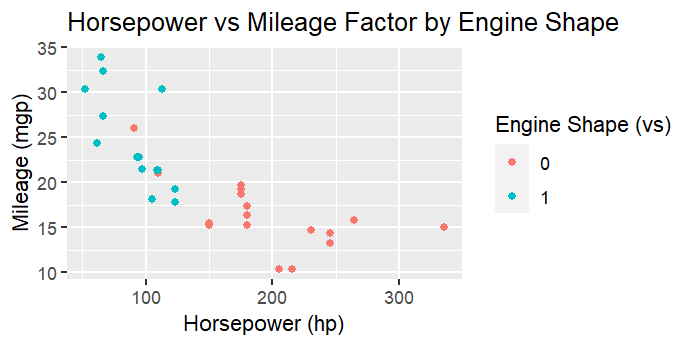
o/p:



(ii) ggplot(mtcars, aes(x = hp, y = mpg, color = factor(vs))) +

geom\_point() +

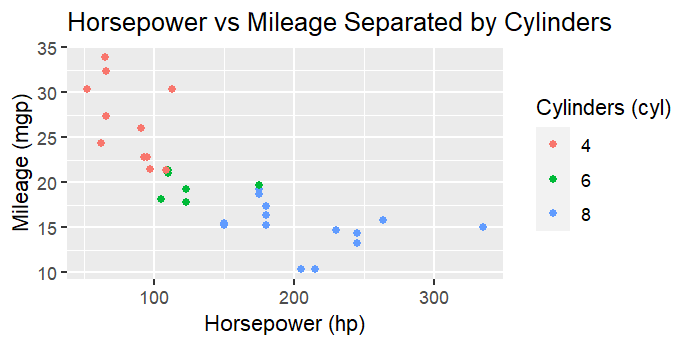
labs(x = "Horsepower (hp)", y = "Mileage (mgp)", color = "Engine Shape (vs)") +

 ggtitle("Horsepower vs Mileage Factor by Engine Shape")

(iii) ggplot(mtcars, aes(x = hp, y = mpg, color = factor(cyl))) +

geom\_point() +

labs(x = "Horsepower (hp)", y = "Mileage (mgp)", color = "Cylinders (cyl)") +

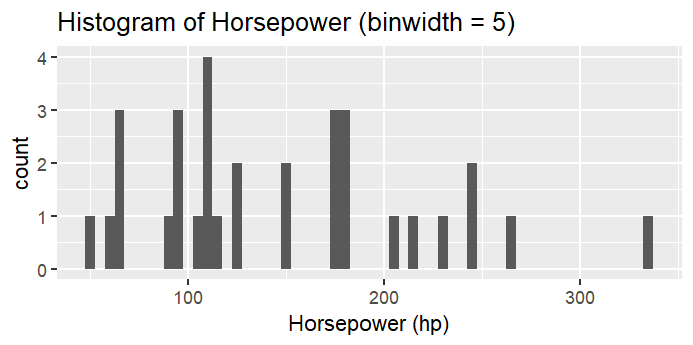
 ggtitle("Horsepower vs Mileage Separated by Cylinders")

(v) ggplot(mtcars, aes(x = hp)) +

geom\_histogram(binwidth = 5) +

labs(x = "Horsepower (hp)") +

ggtitle("Histogram of Horsepower (binwidth = 5)")



**4. Performing Logistic regression on dataset to predict the cars Engine shape(vs) .**

**(i)Do the EDA analysis and find the features which is impact the Engine shape and use**

**this for model.**

**(ii) Split the data set randomly with 80:20 ration to create train and test dataset and create**

**logistic model**

**(iii)Create the Confusion matrix among prediction and test data.**

**Code:**

library(ggplot2)

library(dplyr)

data(mtcars)

summary(mtcars)

ggplot(mtcars, aes(x=mpg)) + geom\_histogram(binwidth = 2)

ggplot(mtcars, aes(x=disp, y=mpg)) + geom\_point()

cor\_matrix <- cor(mtcars[,c("mpg", "disp", "hp", "drat")])

(ii)

set.seed(123)

split <- sample(1:nrow(mtcars), size = 0.8 \* nrow(mtcars), replace = TRUE)

train\_data <- mtcars[split, ]

test\_data <- mtcars[-split, ]

(iii)

model <- glm(vs ~ mpg + disp + hp + drat, data = train\_data, family = binomial())

predictions <- predict(model, test\_data, type = "response")

predictions <- ifelse(predictions > 0.5, 1, 0)

table(predictions, test\_data$vs)

o/p:

predictions 0 1

0 8 0

1 2 4

5. (I) Write R Program to create 15 x15 matrix filled with random numbers between -10 to

10, numbers can repeat. set random seed value to 328

    (ii)Write R Program to display Lower Diagonal and upper Diagonal matrix

   (iii)Write R Program to count 0&#39;s in the matrix and check the matrix is sparse matrix or

not

   (iv) Write R code to remove outliers. Here the outliers are negative numbers. replace the

negative values with positive values

   (v) Find the mean median and mode of the values corresponding to column

   (vi)Find the mean median and mode of the values corresponding to row

Code:

(i

set.seed(328)

matrix <- matrix(runif(225, min=-10, max=10), nrow=15, ncol=15)

(ii)

lower\_diag\_matrix <- diag(matrix)

lower\_diag\_matrix

upper\_diag\_matrix <- t(diag(t(matrix)))

upper\_diag\_matrix

o/p:

lower\_diag\_matrix

[1] 4.6501925 6.4087972 6.4367553 1.4152955 4.4253976 -9.8778837

[7] 4.6515618 4.6226931 -6.8288456 6.5414716 8.8216364 -0.5569335

[13] -0.2875930 0.4255384 7.6948237

> upper\_diag\_matrix

[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]

[1,] 4.650193 6.408797 6.436755 1.415295 4.425398 -9.877884 4.651562 4.622693

[,9] [,10] [,11] [,12] [,13] [,14] [,15]

[1,] -6.828846 6.541472 8.821636 -0.5569335 -0.287593 0.4255384 7.694824

(iii)

zero\_count <- sum(matrix == 0)

zero\_count

if (zero\_count / (nrow(matrix) \* ncol(matrix)) > 0.5) {

print("The matrix is a sparse matrix.")

} else {

print("The matrix is not a sparse matrix.")

}

o/p:

"The matrix is not a sparse matrix."

(vi)

matrix[matrix < 0] <- abs(matrix[matrix < 0])

(v)

find\_statistics <- function(vector) {

mean\_value <- mean(vector)

median\_value <- median(vector)

mode\_value <- names(sort(table(vector), decreasing=TRUE))[1]

return(c(mean=mean\_value, median=median\_value, mode=mode\_value))

}

col\_statistics <- apply(matrix, 2, find\_statistics)

(vi)

row\_statistics <- apply(matrix, 1, find\_statistics)