**PRACTICAL QUESTIONS**

**DAY-01**

**1.** Write a R program to take input from the user (name and age) and display the values. Also print the version of R installation.

name <- readline(prompt = "Enter your name: ")

age <- as.numeric(readline(prompt = "Enter your age: "))

cat("Name: ", name, "\n")

cat("Age: ", age, "\n")

cat("R Installation Version: ", R.version$version.string, "\n")

2.Write a R program to get the details of the objects in memory

my\_vector <- c(1, 2, 3, 4, 5)

my\_data\_frame <- data.frame(Name = c("John", "Jane", "Bob"), Age = c(25, 30, 22))

my\_matrix <- matrix(1:9, nrow = 3, ncol = 3)

cat("Objects in Memory:\n")

print(ls())

cat("\nDetails of Objects:\n")

for (object\_name in ls()) {

cat("Object Name: ", object\_name, "\n")

cat(" Class: ", class(get(object\_name)), "\n")

cat(" Size: ", object.size(get(object\_name)), "bytes\n\n")

}

3.Write a R program to create a sequence of numbers from 20 to 50 and find the mean of numbers from 20 to 60 and sum of numbers from 51 to 91

sequence\_20\_to\_50 <- seq(20, 50)

mean\_20\_to\_60 <- mean(seq(20, 60))

sum\_51\_to\_91 <- sum(seq(51, 91))

cat("Sequence from 20 to 50: ", sequence\_20\_to\_50, "\n")

cat("Mean of numbers from 20 to 60: ", mean\_20\_to\_60, "\n")

cat("Sum of numbers from 51 to 91: ", sum\_51\_to\_91, "\n")

4.Write a R program to create a vector which contains 10 random integer values between -50 and +50.

set.seed(123)

random\_vector <- sample(seq(-50, 50), 10, replace = TRUE)

print(random\_vector)

5.Write a R program to get the first 10 Fibonacci numbers.

fibonacci <- function(n) {

fib\_numbers <- numeric(n)

fib\_numbers[1:2] <- c(0, 1)

for (i in 3:n) {

fib\_numbers[i] <- fib\_numbers[i - 1] + fib\_numbers[i - 2]

}

return(fib\_numbers)

}

first\_10\_fibonacci <- fibonacci(10)

print(first\_10\_fibonacci)

6.Write a R program to get all prime numbers up to a given number (based on the sieve of Eratosthenes)

is\_prime <- function(n) {

if (n <= 1) {

return(FALSE)

}

for (i in 2:(n-1)) {

if (n %% i == 0) {

return(FALSE)

}

}

return(TRUE)

}

get\_primes\_up\_to\_limit <- function(limit) {

primes <- c()

for (num in 2:limit) {

if (is\_prime(num)) {

primes <- c(primes, num)

}

}

return(primes)

}

limit <- 50

prime\_numbers <- get\_primes\_up\_to\_limit(limit)

print(prime\_numbers)

7.Write a R program to print the numbers from 1 to 100 and print "Fizz" for multiples of 3, print "Buzz" for multiples of 5, and print "FizzBuzz" for multiples of both.

for (i in 1:100) {

if (i %% 3 == 0 && i %% 5 == 0) {

print("FizzBuzz")

} else if (i %% 3 == 0) {

print("Fizz")

} else if (i %% 5 == 0) {

print("Buzz")

} else {

print(i)

}

}

8.Write a R program to extract first 10 English letters in lower case and last 10 letters in upper case and extract letters between 22nd to 24th letters in upper case.

alphabet <- letters

first\_10\_lower <- alphabet[1:10]

last\_10\_upper <- toupper(tail(alphabet, 10))

letters\_between\_22\_24\_upper <- toupper(alphabet[22:24])

cat("First 10 English letters in lower case:", first\_10\_lower, "\n")

cat("Last 10 English letters in upper case:", last\_10\_upper, "\n")

cat("Letters between 22nd to 24th in upper case:", letters\_between\_22\_24\_upper, "\n")

9.Write a R program to find the factors of a given number

find\_factors <- function(number) {

factors <- c()

for (i in 1:number) {

if (number %% i == 0) {

factors <- c(factors, i) # Add 'i' to the factors vector

}

}

return(factors)

}

given\_number <- 36

factors\_of\_given\_number <- find\_factors(given\_number)

cat("Factors of", given\_number, "are:", factors\_of\_given\_number, "\n")

10.Write a R program to find the maximum and the minimum value of a given vector

given\_vector <- c(3, 8, 2, 6, 1, 9, 4, 7, 5)

max\_value <- max(given\_vector)

min\_value <- min(given\_vector)

cat("Maximum value:", max\_value, "\n")

cat("Minimum value:", min\_value, "\n")

DAY-02

1.Write a R program to get the unique elements of a given string and unique numbers of vector

get\_unique\_elements <- function(string) {

unique\_chars <- unique(strsplit(string, "")[[1]])

return(unique\_chars)

}

get\_unique\_numbers <- function(vector) {

unique\_nums <- unique(vector)

return(unique\_nums)

}

input\_string <- "hellol"

input\_vector <- c(1, 2, 3, 1, 2, 4, 5, 3)

unique\_elements\_string <- get\_unique\_elements(input\_string)

unique\_elements\_vector <- get\_unique\_numbers(input\_vector)

print("Unique elements of the string:")

print(unique\_elements\_string)

print("Unique numbers of the vector:")

print(unique\_elements\_vector)

2.Write a R program to create three vectors a,b,c with 3 integers. Combine the three vectors to become a 3×3 matrix where each column represents a vector. Print the content of the matrix.

a <- c(1, 2, 3)

b <- c(4, 5, 6)

c <- c(7, 8, 9)

combined\_matrix <- cbind(a, b, c)

print("Combined Matrix:")

print(combined\_matrix)

3.Write a R program to create a list of random numbers in normal distribution and count occurrences of each value.

set.seed(123)

random\_numbers <- rnorm(100, mean = 0, sd = 1) # Generating 100 random numbers with mean 0 and standard deviation 1

occurrences <- table(random\_numbers)

print("Occurrences of each value:")

print(occurrences)

4.Write a R program to read the .csv file and display the content

data <- read.csv("your\_file.csv")

print("Contents of the CSV file:")

print(data)

5.Write a R program to create three vectors numeric data, character data and logical data. Display the content of the vectors and their type

numeric\_vector <- c(1, 2, 3, 4, 5)

character\_vector <- c("apple", "banana", "orange")

logical\_vector <- c(TRUE, FALSE, TRUE)

print("Numeric vector:")

print(numeric\_vector)

print("Character vector:")

print(character\_vector)

print("Logical vector:")

print(logical\_vector)

print("Data types:")

print(paste("Numeric vector type:", class(numeric\_vector)))

print(paste("Character vector type:", class(character\_vector)))

print(paste("Logical vector type:", class(logical\_vector)))

6.Write a R program to create a 5 x 4 matrix, 3 x 3 matrix with labels and fill the matrix by rows and 2 × 2 matrix with labels and fill the matrix by columns.

matrix\_5x4 <- matrix(1:20, nrow = 5, ncol = 4, byrow = TRUE,

dimnames = list(c("Row1", "Row2", "Row3", "Row4", "Row5"),

c("Col1", "Col2", "Col3", "Col4")))

matrix\_3x3 <- matrix(letters[1:9], nrow = 3, ncol = 3,

dimnames = list(c("Row1", "Row2", "Row3"),

c("Col1", "Col2", "Col3")))

matrix\_2x2 <- matrix(1:4, nrow = 2, ncol = 2, byrow = FALSE,

dimnames = list(NULL, c("Col1", "Col2")))

print("5x4 Matrix with Row Labels:")

print(matrix\_5x4)

print("3x3 Matrix with Row and Column Labels:")

print(matrix\_3x3)

print("2x2 Matrix with Column Labels:")

print(matrix\_2x2)

7.Write a R program to create an array, passing in a vector of values and a vector of dimensions. Also provide names for each dimension

values <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12)

dimensions <- c(3, 2, 2) # Represents a 3x2x2 array

my\_array <- array(values, dim = dimensions, dimnames = list(c("Dim1", "Dim2", "Dim3"),

c("Col1", "Col2"),

c("Depth1", "Depth2")))

print(my\_array)

8.Write a R program to create an array with three columns, three rows, and two "tables", taking two vectors as input to the array. Print the array.

vector1 <- c(1, 2, 3)

vector2 <- c(4, 5, 6)

vector3 <- c(7, 8, 9)

matrix1 <- cbind(vector1, vector2, vector3)

matrix2 <- matrix(c(10, 11, 12, 13, 14, 15, 16, 17, 18), nrow = 3)

my\_array <- array(c(matrix1, matrix2), dim = c(3, 3, 2),

dimnames = list(c("Row1", "Row2", "Row3"),

c("Col1", "Col2", "Col3"),

c("Table1", "Table2")))

print(my\_array)

9.Write a R program to create a list of elements using vectors, matrices and a function. Print the content of the list

vector1 <- c(1, 2, 3)

vector2 <- c(4, 5, 6)

vector3 <- c(7, 8, 9)

my\_array <- array(c(vector1, vector2, vector3), dim = c(3, 3, 2))

print(my\_array)

10.Write a R program to draw an empty plot and an empty plot specify the axes limits of the graphic

vector <- c(1, 2, 3, 4)

character\_vector <- c("apple", "banana", "orange")

logical\_vector <- c(TRUE, FALSE, TRUE)

matrix1 <- matrix(1:4, nrow = 2)

matrix2 <- matrix(c("a", "b", "c", "d"), nrow = 2)

my\_function <- function(x) {

return(x^2)

}

my\_list <- list(vector = vector,

character\_vector = character\_vector,

logical\_vector = logical\_vector,

matrix1 = matrix1,

matrix2 = matrix2,

my\_function = my\_function)

print(my\_list)

DAY-03

1.Write a R program to create an array of two 3x3 matrices each with 3 rows and 3 columns from two given two vectors. Print the second row of the second matrix of the array and the element in the 3rd row and 3rd column of the 1st matrix

vector1 <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)

vector2 <- c(10, 11, 12, 13, 14, 15, 16, 17, 18)

your\_array <- array(c(vector1, vector2), dim = c(3, 3, 2), dimnames = list(c("Row1", "Row2", "Row3"), c("Col1", "Col2", "Col3"), c("Matrix1", "Matrix2")))

print(your\_array)

cat("\nSecond row of the second matrix:\n")

print(your\_array["Row2", , "Matrix2"])

cat("\nElement in the 3rd row and 3rd column of the 1st matrix:\n")

print(your\_array["Row3", "Col3", "Matrix1"])

2.Write a R program to combine three arrays so that the first row of the first array is followed by the first row of the second array and then first row of the third array

array1 <- matrix(1:3, nrow = 1, ncol = 3)

array2 <- matrix(4:6, nrow = 1, ncol = 3)

array3 <- matrix(7:9, nrow = 1, ncol = 3)

combined\_array <- rbind(array1, array2, array3)

print(combined\_array)

3.Write a R program to create an array using four given columns, three given rows, and two given tables and display the content of the array

data\_table1 <- matrix(1:12, nrow = 3)

data\_table2 <- matrix(13:24, nrow = 3)

your\_array <- array(c(data\_table1, data\_table2), dim = c(3, 4, 2), dimnames = list(c("Row1", "Row2", "Row3"), c("Col1", "Col2", "Col3", "Col4"), c("Table1", "Table2")))

print(your\_array)

4.Write a R program to create a two-dimensional 5x3 array of sequence of even integers greater than 50

your\_array <- matrix(seq(from = 52, by = 2, length.out = 5 \* 3), nrow = 5, ncol = 3)

print(your\_array)

5.Create below data frame exam\_data = data. Frame( name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'), score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19), attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1), qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes') ) a. Write a R program to extract 3rd and 5th rows with 1st and 3rd columns from a given data frame b. Write a R program to add a new column named country in a given data frame Country<-c("USA","USA","USA","USA","UK","USA","USA","India","USA","USA")

exam\_data <- data.frame(

name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),

score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),

attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),

qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')

)

selected\_data <- exam\_data[c(3, 5), c(1, 3)]

print(selected\_data)

exam\_data <- data.frame(

name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),

score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),

attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),

qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')

)

country <- c("USA", "USA", "USA", "USA", "UK", "USA", "USA", "India", "USA", "USA")

exam\_data$country <- country

print(exam\_data)

6.Write a R program to add new row(s) to an existing data frame new\_exam\_data = data.frame(name = c('Robert', 'Sophia'),score = c(10.5, 9), attempts = c(1, 3),qualify = c('yes', 'no')) d. Write a R program to sort a given data frame by name and score e. Write a R program to save the information of a data frame in a file and display the information of the file.

exam\_data <- data.frame(

name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),

score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),

attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),

qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')

)

new\_exam\_data <- data.frame(name = c('Robert', 'Sophia'), score = c(10.5, 9), attempts = c(1, 3), qualify = c('yes', 'no'))

combined\_data <- rbind(exam\_data, new\_exam\_data)

print(combined\_data)

sorted\_data <- combined\_data[order(combined\_data$name, combined\_data$score), ]

print(sorted\_data)

write.csv(sorted\_data, "sorted\_data.csv", row.names = FALSE)

file\_content <- read.csv("sorted\_data.csv")

print(file\_content)

7.Write a R program to call the (built-in) dataset airquality. Check whether it is a data frame or not? Order the entire data frame by the first and second column. remove the variables 'Solar.R' and 'Wind' and display the data frame

data <- airquality

if (is.data.frame(data)) {

cat("The 'airquality' dataset is a data frame.\n")

} else {

cat("The 'airquality' dataset is not a data frame.\n")

}

ordered\_data <- data[order(data$Ozone, data$Solar.R), ]

cat("\nOrdered Data Frame:\n")

print(ordered\_data)

data\_without\_solar\_wind <- data[, !(names(data) %in% c("Solar.R", "Wind"))]

cat("\nData Frame without 'Solar.R' and 'Wind':\n")

print(data\_without\_solar\_wind)

8.Write a R program to create a factor corresponding to height of women data set, which inbuild in R, contains height and weights for a sample of women.

data(women)

height\_factor <- cut(women$height, breaks = c(55, 60, 65, 70, 75), labels = c("Short", "Average", "Tall", "Very Tall"))

cat("Original Women Dataset:\n")

print(women)

cat("\nFactor Corresponding to Height:\n")

print(height\_factor)

9.Write a R program to extract the five of the levels of factor created from a random sample from the LETTERS (Part of the base R distribution.)

set.seed(123)

random\_factor <- factor(sample(LETTERS, 20, replace = TRUE))

cat("Original Factor:\n")

print(random\_factor)

selected\_levels <- levels(random\_factor)[1:5]

cat("\nSelected Levels:\n")

print(selected\_levels)

10..Iris dataset is a very famous dataset in almost all data mining, machine learning courses, and it has been an R build-in dataset. The dataset consists of 50 samples from each of three species of Iris flowers (Iris setosa, Iris virginica and Iris versicolor). Four features(variables) were measured from each sample, they are the length and the width of sepal and petal, in centimetres. Perform the following EDA steps.

(i)Find dimension, Structure, Summary statistics, Standard Deviation of all features.

(ii)Find mean and standard deviation of features groped by three species of Iris flowers (Iris setosa, Iris virginica and Iris versicolor)

(iii)Find quantile value of sepal width and length

(iv)create new data frame named iris1 which have a new column name Sepal.Length.Cate that categorizes “Sepal.Length” by quantile

(v) Average value of numerical varialbes by two categorical variables: Species and Sepal.Length.Cate:

(vi) Average mean value of numerical variables by Species and Sepal.Length.Cate

(vii)Create Pivot Table based on Species and Sepal.Length.Cate.

# Load the Iris dataset

data(iris)

# (i) Find dimension, Structure, Summary statistics, Standard Deviation of all features

cat("Dimension of the dataset:\n")

print(dim(iris))

cat("\nStructure of the dataset:\n")

str(iris)

cat("\nSummary statistics of all features:\n")

print(summary(iris))

cat("\nStandard Deviation of all features:\n")

print(sapply(iris, sd))

# (ii) Find mean and standard deviation of features grouped by three species of Iris flowers

mean\_by\_species <- tapply(iris$Sepal.Length, iris$Species, mean)

sd\_by\_species <- tapply(iris$Sepal.Length, iris$Species, sd)

cat("\nMean of Sepal.Length grouped by Species:\n")

print(mean\_by\_species)

cat("\nStandard Deviation of Sepal.Length grouped by Species:\n")

print(sd\_by\_species)

# (iii) Find quantile value of sepal width and length

cat("\nQuantile values of Sepal.Width and Sepal.Length:\n")

print(quantile(iris$Sepal.Width))

print(quantile(iris$Sepal.Length))

# (iv) Create new data frame named iris1 with a new column named Sepal.Length.Cate

iris1 <- iris

iris1$Sepal.Length.Cate <- cut(iris1$Sepal.Length, breaks = quantile(iris1$Sepal.Length), labels = c("Q1", "Q2", "Q3", "Q4"))

# (v) Average value of numerical variables by two categorical variables: Species and Sepal.Length.Cate

cat("\nAverage value of numerical variables by Species and Sepal.Length.Cate:\n")

print(tapply(iris1[, c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width")], list(iris1$Species, iris1$Sepal.Length.Cate), mean))

# (vi) Average mean value of numerical variables by Species and Sepal.Length.Cate

cat("\nAverage mean value of numerical variables by Species and Sepal.Length.Cate:\n")

print(aggregate(cbind(Sepal.Length, Sepal.Width, Petal.Length, Petal.Width) ~ Species + Sepal.Length.Cate, data = iris1, mean))

# (vii) Create Pivot Table based on Species and Sepal.Length.Cate

pivot\_table <- table(iris1$Species, iris1$Sepal.Length.Cate)

cat("\nPivot Table based on Species and Sepal.Length.Cate:\n")

print(pivot\_table)

DAY-04

1.Randomly Sample the iris dataset such as 80% data for training and 20% for test and create Logistics regression with train data, use species as target and petals width and length as feature variables, Predict the probability of the model using test data, Create Confusion matrix for above test model

(i)Write suitable R code to compute the mean, median, mode of the following values c (90, 50, 70, 80, 70, 60, 20, 30, 80, 90, 20)

(ii) Write R code to find 2nd highest and 3rd Lowest value of above problem

# Load required library

library(datasets)

# Load iris dataset

data(iris)

# Set seed for reproducibility

set.seed(123)

# Randomly sample 80% data for training and 20% for testing

train\_index <- sample(1:nrow(iris), 0.8 \* nrow(iris))

train\_data <- iris[train\_index, ]

test\_data <- iris[-train\_index, ]

# Create Logistic Regression model

model <- glm(Species ~ Petal.Width + Petal.Length, data = train\_data, family = binomial)

# Predict probabilities using test data

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

# Convert probabilities to predicted class

predicted\_class <- ifelse(probabilities > 0.5, "versicolor", "not versicolor")

# Create Confusion matrix

conf\_matrix <- table(test\_data$Species, predicted\_class)

c\_values <- c(90, 50, 70, 80, 70, 60, 20, 30, 80, 90, 20)

mean\_c <- mean(c\_values)

median\_c <- median(c\_values)

mode\_c <- names(sort(table(c\_values), decreasing = TRUE))[1]

d

sorted\_unique\_c <- sort(unique(c\_values))

second\_highest <- sorted\_unique\_c[length(sorted\_unique\_c) - 1]

third\_lowest <- sorted\_unique\_c[3]

print("Mean of c:")

print(mean\_c)

print("Median of c:")

print(median\_c)

print("Mode of c:")

print(mode\_c)

print("Second highest value of c:")

print(second\_highest)

print("Third lowest value of c:")

print(third\_lowest)

print("Confusion Matrix:")

print(conf\_matrix)

2.Explore the air quality dataset. It contains daily air quality measurements from New York during a period of five months:

• Ozone: mean ozone concentration (ppb),

• Solar.R: solar radiation (Langley),

• Wind: average wind speed (mph),

• Temp: maximum daily temperature in degrees Fahrenheit,

• Month: numeric month (May=5, June=6, and so on),

• Day: numeric day of the month (1- 31).

1. Compute the mean temperature (don’t use build in function)
2. Extract the first five rows from airquality.
3. Extract all columns from airquality except Temp and Wind
4. Which was the coldest day during the period?

(i)Get the Summary Statistics of air quality dataset

(ii)Melt airquality data set and display as a long – format data?

(iii)Melt airquality data and specify month and day to be “ID variables”? (iv)Cast the molten airquality data set with respect to month and date features

(v) Use cast function appropriately and compute the average of Ozone, Solar.R , Wind and temperature per month?

data(airquality)

mean\_temp <- sum(airquality$Temp) / length(airquality$Temp)

first\_five\_rows <- airquality[1:5, ]

selected\_columns <- airquality[, !(names(airquality) %in% c("Temp", "Wind"))]

coldest\_day <- airquality$Day[which.min(airquality$Temp)]

summary\_stats <- summary(airquality)

library(reshape2)

melted\_data <- melt(airquality)

melted\_data\_id <- melt(airquality, id.vars = c("Month", "Day"))

cast\_data <- dcast(melted\_data\_id, Month + Day ~ variable)

average\_per\_month <- aggregate(cbind(Ozone, Solar.R, Wind, Temp) ~ Month, data = airquality, mean)

print("Mean temperature:")

print(mean\_temp)

print("First five rows:")

print(first\_five\_rows)

print("Selected columns:")

print(selected\_columns)

print("Coldest day:")

print(coldest\_day)

print("Summary statistics:")

print(summary\_stats)

print("Melted data:")

print(head(melted\_data))

print("Melted data with Month and Day as ID variables:")

print(head(melted\_data\_id))

print("Casted data:")

print(head(cast\_data))

print("Average per month:")

print(average\_per\_month)

3.(i) Find any missing values(na) in features and drop the missing values if its less than 10% else replace that with mean of that feature.

(ii) Apply a linear regression algorithm using Least Squares Method on “Ozone” and “Solar.R”

(iii)Plot Scatter plot between Ozone and Solar and add regression line created by above model

data(airquality)

missing\_values <- colMeans(is.na(airquality))

threshold <- 0.1

for (col in names(airquality)) {

if (missing\_values[col] < threshold) {

airquality <- na.omit(airquality)

} else {

mean\_value <- mean(airquality[[col]], na.rm = TRUE)

airquality[[col]][is.na(airquality[[col]])] <- mean\_value

}

}

model <- lm(Ozone ~ Solar.R, data = airquality)

plot(airquality$Solar.R, airquality$Ozone, xlab = "Solar Radiation", ylab = "Ozone", main = "Scatter Plot with Regression Line")

abline(model, col = "red")

4.Load dataset named ChickWeight,

( i).Order the data frame, in ascending order by feature name “weight” grouped by feature “diet” and Extract the last 6 records from order data frame.

(ii)a. Perform melting function based on “Chick", "Time", "Diet" features as ID variables

b. Perform cast function to display the mean value of weight grouped by Diet

c. Perform cast function to display the mode of weight grouped by Diet

data(ChickWeight)

ordered\_df <- ChickWeight[order(ChickWeight$weight, ChickWeight$Diet), ]

last\_6\_records <- tail(ordered\_df, 6)

melted\_df <- data.frame(

Chick = rep(ChickWeight$Chick, each = 4),

Time = rep(ChickWeight$Time, each = 4),

Diet = rep(ChickWeight$Diet, each = 4),

variable = rep(c("weight", "Time", "Chick", "Diet"), times = nrow(ChickWeight)),

value = c(ChickWeight$weight, ChickWeight$Time, ChickWeight$Chick, ChickWeight$Diet)

)

mean\_weight <- tapply(ChickWeight$weight, ChickWeight$Diet, mean)

mode\_weight <- tapply(ChickWeight$weight, ChickWeight$Diet, function(x) {

ux <- unique(x)

ux[which.max(tabulate(match(x, ux)))]

})

print("Last 6 records:")

print(last\_6\_records)

print("Mean value of weight grouped by Diet:")

print(mean\_weight)

print("Mode of weight grouped by Diet:")

print(mode\_weight)

5.a. Create Box plot for “weight” grouped by “Diet”

b. Create a Histogram for “weight” features belong to Diet- 1 category

c. Create Scatter plot for “ weight” vs “Time” grouped by Diet

data(ChickWeight)

boxplot(ChickWeight$weight ~ ChickWeight$Diet,

xlab = "Diet",

ylab = "Weight",

main = "Box plot of Weight grouped by Diet")

hist(ChickWeight$weight[ChickWeight$Diet == 1],

xlab = "Weight",

ylab = "Frequency",

main = "Histogram of Weight for Diet-1",

col = "skyblue")

par(mfrow = c(2, 2))

for (diet in unique(ChickWeight$Diet)) {

subset\_data <- subset(ChickWeight, Diet == diet)

plot(subset\_data$Time, subset\_data$weight,

xlab = "Time",

ylab = "Weight",

main = paste("Scatter plot of Weight vs Time for Diet", diet),

col = diet)

}

Day-05

1. a. Create Box plot for "weight" grouped by "Diet" b. Create a Histogram for "weight" features belong to Diet- 1 category c. Create Scatter plot for "weight" vs "Time" grouped by Diet.

data(ChickWeight)

boxplot(ChickWeight$weight ~ ChickWeight$Diet, main="Box plot of Weight grouped by Diet",

xlab="Diet", ylab="Weight", col=c("blue","green","red","orange"))

hist(subset(ChickWeight, Diet == 1)$weight, main="Histogram of Weight for Diet-1",

xlab="Weight", ylab="Frequency", col="lightblue")

plot(ChickWeight$Time, ChickWeight$weight, col=ChickWeight$Diet,

main="Scatter plot of Weight vs Time grouped by Diet",

xlab="Time", ylab="Weight")

legend("topright", legend=unique(ChickWeight$Diet), col=1:length(unique(ChickWeight$Diet)), pch=1, cex=0.8)

2.a. Create multi regression model to find a weight of the chicken, by

“Time” and “Diet” as predictor variables

b. Predict weight for Time=10 and Diet=1

c. Find the error in model for same.

data(ChickWeight)

ChickWeight$Diet <- as.factor(ChickWeight$Diet)

model <- lm(weight ~ Time + Diet, data = ChickWeight)

new\_data <- data.frame(Time = 10, Diet = factor(1, levels = levels(ChickWeight$Diet)))

predicted\_weight <- predict(model, newdata = new\_data)

cat("Predicted weight for Time = 10 and Diet = 1:", predicted\_weight, "\n")

actual\_weight <- subset(ChickWeight, Time == 10 & Diet == "1")$weight

error <- actual\_weight - predicted\_weight

cat("Error in the model:", error, "\n")

3. For this exercise, use the (built-in) dataset Titanic.

a. Draw a Bar chart to show details of “Survived” on the Titanic based

on passenger Class

b. Modify the above plot based on gender of people who survived

c. Draw histogram plot to show distribution of feature “Age”

data(Titanic)

barplot(rowSums(Titanic[,,,"Yes"]), beside=TRUE, col=rainbow(2), legend=TRUE,

main="Survival Count by Passenger Class", xlab="Passenger Class", ylab="Count")

legend("topright", legend=c("Survived", "Did not survive"), fill=rainbow(2))

barplot(as.matrix(Titanic[,,c("Male","Female"),"Yes"]), beside=TRUE, col=c("lightblue", "pink"), legend=TRUE,

main="Survival Count by Passenger Class and Gender", xlab="Passenger Class", ylab="Count")

legend("topright", legend=c("Male", "Female"), fill=c("lightblue", "pink"))

hist(Titanic$Age, breaks=20, col="skyblue", main="Age Distribution", xlab="Age", ylab="Frequency")