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.

# Taking input from the user

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

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

# Converting age to numeric (since readline takes input as character)

age <- as.numeric(age)

# Displaying the values

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

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

# Printing the R version

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

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

objects\_in\_memory <- ls()

cat("Objects in memory:\n")

print(objects\_in\_memory)

1. 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.

# Create a sequence of numbers from 20 to 50

sequence <- 20:50

cat("Sequence from 20 to 50:\n")

print(sequence)

# Find the mean of numbers from 20 to 60

mean\_value <- mean(20:60)

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

# Find the sum of numbers from 51 to 91

sum\_value <- sum(51:91)

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

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

# Set the seed for reproducibility (optional)

set.seed(123)

# Create a vector of 10 random integers between -50 and +50

random\_integers <- sample(-50:50, 10, replace = TRUE)

# Display the vector

cat("Vector with 10 random integers between -50 and +50:\n")

print(random\_integers)

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

# Function to generate the first n Fibonacci numbers

fibonacci <- function(n) {

fib\_sequence <- numeric(n) # Initialize a vector to store the sequence

fib\_sequence[1] <- 0

if (n > 1) fib\_sequence[2] <- 1

for (i in 3:n) {

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

}

return(fib\_sequence)

}

# Get the first 10 Fibonacci numbers

first\_10\_fib <- fibonacci(10)

# Display the Fibonacci numbers

cat("The first 10 Fibonacci numbers:\n")

print(first\_10\_fib)

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

# Function to get all prime numbers up to a given number using Sieve of Eratosthenes

sieve\_of\_eratosthenes <- function(n) {

# Create a logical vector of TRUE values representing numbers 2 to n

primes <- rep(TRUE, n)

primes[1] <- FALSE # 1 is not a prime number

for (i in 2:sqrt(n)) {

if (primes[i]) {

# Mark all multiples of i as FALSE (not prime)

primes[seq(i^2, n, by = i)] <- FALSE

}

}

# Return all numbers that are still TRUE (prime numbers)

return(which(primes))

}

# Example: Get all prime numbers up to 50

n <- 50

prime\_numbers <- sieve\_of\_eratosthenes(n)

print(prime\_numbers)

1. 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.

# Loop through numbers from 1 to 100

for (i in 1:100) {

if (i %% 15 == 0) {

cat("FizzBuzz\n")

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

cat("Fizz\n")

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

cat("Buzz\n")

} else {

cat(i, "\n")

}

}

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

# Get the English alphabet

letters\_lower <- letters # Lowercase letters: a-z

letters\_upper <- LETTERS # Uppercase letters: A-Z

# Extract the first 10 lowercase letters

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

# Extract the last 10 uppercase letters

last\_10\_upper <- letters\_upper[17:26] # Since last 10 letters are from 17th to 26th

# Extract the letters between 22nd to 24th in uppercase

letters\_22\_to\_24\_upper <- letters\_upper[22:24]

# Print the results

print("First 10 lowercase letters:")

print(first\_10\_lower)

print("Last 10 uppercase letters:")

print(last\_10\_upper)

print("22nd to 24th uppercase letters:")

print(letters\_22\_to\_24\_upper)

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

# Function to find the factors of a given number

find\_factors <- function(num) {

factors <- numeric() # Initialize an empty vector to store factors

for (i in 1:num) {

if (num %% i == 0) { # Check if i is a factor of num

factors <- c(factors, i) # Append the factor to the vector

}

}

return(factors)

}

number <- 36

factors\_of\_number <- find\_factors(number)

print(paste("Factors of", number, ":"))

print(factors\_of\_number)

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

# Example vector

vector <- c(10, 25, 3, 47, 88, 15, 0, -5, 73)

# Find the maximum value

max\_value <- max(vector)

# Find the minimum value

min\_value <- min(vector)

# Print the results

print("Maximum value:")

print(max\_value)

print("Minimum value:")

print(min\_value)

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

# Example string

string <- "hello world"

# Example vector

vector <- c(10, 25, 3, 10, 47, 25, 15, 0, 73, 47)

# Get unique characters from the string

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

# Get unique numbers from the vector

unique\_numbers <- unique(vector)

# Print the unique characters

print("Unique characters in the string:")

print(unique\_chars)

# Print the unique numbers

print("Unique numbers in the vector:")

print(unique\_numbers)

1. 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.

# Create three vectors with 3 integers each

a <- c(1, 4, 7)

b <- c(2, 5, 8)

c <- c(3, 6, 9)

# Combine the vectors into a 3×3 matrix with each vector as a column

matrix <- cbind(a, b, c)

# Print the content of the matrix

print("3x3 Matrix:")

print(matrix)

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

# Set seed for reproducibility

set.seed(123)

# Create a list of random numbers from a normal distribution

random\_numbers <- rnorm(100) # Generate 100 random numbers

# Round numbers to make them easier to count (optional)

rounded\_numbers <- round(random\_numbers, 2)

# Count occurrences of each value

value\_counts <- table(rounded\_numbers)

# Print the counts of each value

print("Occurrences of each value:")

print(value\_counts)

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

# Specify the path to the .csv file

file\_path <- "path/to/your/file.csv"

# Read the .csv file into a data frame

data <- read.csv(file\_path)

# Display the content of the data frame

print("Content of the .csv file:")

print(data)

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

# Create three vectors

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

character\_vector <- c("apple", "banana", "cherry") # Character data

logical\_vector <- c(TRUE, FALSE, TRUE) # Logical data

# Display the content of each vector

print("Numeric vector:")

print(numeric\_vector)

print(paste("Type:", typeof(numeric\_vector)))

print("Character vector:")

print(character\_vector)

print(paste("Type:", typeof(character\_vector)))

print("Logical vector:")

print(logical\_vector)

print(paste("Type:", typeof(logical\_vector)))

1. 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.

# Create a 5 x 4 matrix and fill it by rows

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

print("5x4 Matrix filled by rows:")

print(matrix\_5x4)

# Create a 3 x 3 matrix with row and column labels

matrix\_3x3 <- matrix(1:9, nrow = 3, ncol = 3)

rownames(matrix\_3x3) <- c("Row1", "Row2", "Row3")

colnames(matrix\_3x3) <- c("Col1", "Col2", "Col3")

print("3x3 Matrix with labels:")

print(matrix\_3x3)

# Create a 2 x 2 matrix with row and column labels and fill it by columns

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

rownames(matrix\_2x2) <- c("Row1", "Row2")

colnames(matrix\_2x2) <- c("Col1", "Col2")

print("2x2 Matrix filled by columns with labels:")

print(matrix\_2x2)

1. 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.

# Define a vector of values

values <- 1:24

# Define dimensions for the array

dimensions <- c(2, 3, 4) # 2 layers, 3 rows, 4 columns

# Create the array

my\_array <- array(values, dim = dimensions)

# Provide names for each dimension

dimnames(my\_array) <- list(

Layer = c("Layer1", "Layer2"),

Row = c("Row1", "Row2", "Row3"),

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

)

# Print the array

print("Array with dimensions and names:")

print(my\_array)

1. 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.

# Define two vectors with 9 elements each

vector1 <- 1:9

vector2 <- 10:18

# Combine the vectors into a single array

# Each vector will be a separate table in the array

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

# Provide names for the dimensions (optional)

dimnames(my\_array) <- list(

Row = c("Row1", "Row2", "Row3"),

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

Table = c("Table1", "Table2")

)

# Print the array

print("Array with 3 columns, 3 rows, and 2 tables:")

print(my\_array)

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

# Create a vector

vector\_example <- c(1, 2, 3, 4, 5)

# Create a matrix

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

# Create a simple function

simple\_function <- function(x) {

return(x^2)

}

# Create a list containing the vector, matrix, and function

my\_list <- list(

Vector = vector\_example,

Matrix = matrix\_example,

Function = simple\_function

)

# Print the content of the list

print("Content of the list:")

print(my\_list)

# To demonstrate using the function within the list

print("Applying the function from the list to number 4:")

print(my\_list$Function(4))

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

# Draw an empty plot with default settings

plot(1, type = "n", xlab = "X-axis", ylab = "Y-axis", main = "Empty Plot with Default Axes")

# Draw an empty plot with specified axes limits

plot(1, type = "n", xlab = "X-axis", ylab = "Y-axis",

xlim = c(-10, 10), ylim = c(-20, 20),

main = "Empty Plot with Specified Axes Limits")

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.

# Define two vectors

vector1 <- 1:9

vector2 <- 10:18

# Create two 3x3 matrices from the vectors

matrix1 <- matrix(vector1, nrow = 3, ncol = 3, byrow = TRUE)

matrix2 <- matrix(vector2, nrow = 3, ncol = 3, byrow = TRUE)

# Create an array of the two matrices

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

# Print the second row of the second matrix in the array

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

print(array\_matrices[2, , 2])

# Print the element in the 3rd row and 3rd column of the first matrix

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

print(array\_matrices[3, 3, 1])

1. 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.

# Create three arrays (each with 2 rows, 3 columns, and 2 matrices)

array1 <- array(1:12, dim = c(2, 3, 2))

array2 <- array(13:24, dim = c(2, 3, 2))

array3 <- array(25:36, dim = c(2, 3, 2))

# Combine the first rows of all three arrays

combined\_rows <- rbind(array1[1, , ], array2[1, , ], array3[1, , ])

# Print the combined result

cat("Combined rows from the three arrays:\n")

print(combined\_rows)

1. 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.

# Define a vector with 24 elements (4 columns \* 3 rows \* 2 tables = 24 elements)

elements <- 1:24

# Create an array with 3 rows, 4 columns, and 2 tables

array\_example <- array(elements, dim = c(3, 4, 2))

# Display the content of the array

cat("Array content:\n")

print(array\_example)

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

# Create a sequence of even integers greater than 50

even\_numbers <- seq(from = 52, by = 2, length.out = 15) # 15 even numbers

# Create a 5x3 array

array\_5x3 <- array(even\_numbers, dim = c(5, 3))

# Display the content of the array

cat("5x3 Array of even integers greater than 50:\n")

print(array\_5x3)

1. 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')

)

1. Write a R program to extract 3rd and 5th rows with 1st and 3rd columns from a given data frame
2. 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")

1. 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'))

1. Write a R program to sort a given data frame by name and score

1. Write a R program to save the information of a data frame in a file and display the information of the file.

# Create the initial 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. Extract 3rd and 5th rows with 1st and 3rd columns

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

print("Extracted Data (3rd and 5th rows with 1st and 3rd columns):")

print(extracted\_data)

# b. Add a new column named 'country'

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

exam\_data$country <- Country

print("Data Frame with Country Column Added:")

print(exam\_data)

# c. Add new rows to the data frame

new\_exam\_data <- data.frame(

name = c('Robert', 'Sophia'),

score = c(10.5, 9),

attempts = c(1, 3),

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

)

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

print("Data Frame with New Rows Added:")

print(exam\_data)

# d. Sort the data frame by 'name' and 'score'

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

print("Sorted Data Frame (by name and score):")

print(sorted\_exam\_data)

# e. Save the data frame to a CSV file and display the file content

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

# Read the CSV file and display its content

saved\_data <- read.csv("exam\_data.csv")

print("Content of the Saved CSV File:")

print(saved\_data)

1. 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.

# Create the initial 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. Extract 3rd and 5th rows with 1st and 3rd columns

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

cat("Extracted 3rd and 5th rows with 1st and 3rd columns:\n")

print(extracted\_data)

# b. Add a new column 'country' to the data frame

exam\_data$country <- c("USA", "USA", "USA", "USA", "UK", "USA", "USA", "India", "USA", "USA")

cat("\nData frame with new column 'country':\n")

print(exam\_data)

# c. Create new rows and add them to the 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')

)

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

cat("\nData frame after adding new rows:\n")

print(exam\_data)

# d. Sort the data frame by 'name' and 'score'

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

cat("\nSorted data frame by 'name' and 'score':\n")

print(sorted\_data)

# e. Save the data frame to a file and display the information

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

cat("\nInformation from the saved file:\n")

loaded\_data <- read.csv("exam\_data.csv")

print(loaded\_data)

1. 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.

# Load the built-in 'women' dataset

data(women)

# Display the first few rows of the dataset

print("Original women dataset:")

print(head(women))

# Create a factor for the height column

height\_factor <- factor(women$height)

# Display the factor

print("Factor corresponding to height:")

print(height\_factor)

# Display summary of the factor

print("Summary of the height factor:")

print(summary(height\_factor))

1. 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 for reproducibility

set.seed(123)

# Create a random sample of 20 letters from the LETTERS vector

sample\_letters <- sample(LETTERS, 20, replace = TRUE)

# Create a factor from the random sample

letter\_factor <- factor(sample\_letters)

# Display the factor levels

print("Factor levels:")

print(levels(letter\_factor))

# Extract and display the first 5 levels of the factor

print("First 5 levels of the factor:")

print(levels(letter\_factor)[1:5])

1. **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 varialbes by Species and Sepal.Length.Cate (vii)Create Pivot Table based on Species and Sepal.Length.Cate.

# Load the built-in iris dataset

data(iris)

# (i) Find dimension, structure, summary statistics, and standard deviation of all features

print("Dimension of iris dataset:")

print(dim(iris))

print("Structure of iris dataset:")

print(str(iris))

print("Summary statistics of iris dataset:")

print(summary(iris))

print("Standard deviation of all features:")

print(apply(iris[, 1:4], 2, sd)) # Apply sd function to each column (features)

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

library(dplyr)

print("Mean and Standard Deviation by Species:")

mean\_sd\_by\_species <- iris %>%

group\_by(Species) %>%

summarise(across(everything(), list(mean = mean, sd = sd), .names = "{col}\_{fn}"))

print(mean\_sd\_by\_species)

# (iii) Find quantile values of Sepal.Width and Sepal.Length

print("Quantile values of Sepal.Width:")

print(quantile(iris$Sepal.Width))

print("Quantile values of Sepal.Length:")

print(quantile(iris$Sepal.Length))

# (iv) Create a new data frame iris1 with a new column Sepal.Length.Cate that categorizes Sepal.Length by quantile

iris1 <- iris %>%

mutate(Sepal.Length.Cate = cut(Sepal.Length, breaks = quantile(Sepal.Length, probs = 0:4 / 4), include.lowest = TRUE, labels = FALSE))

print("Iris dataset with Sepal.Length.Cate:")

print(head(iris1))

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

print("Average values of numerical variables by Species and Sepal.Length.Cate:")

avg\_by\_species\_cate <- iris1 %>%

group\_by(Species, Sepal.Length.Cate) %>%

summarise(across(where(is.numeric), mean), .groups = 'drop')

print(avg\_by\_species\_cate)

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

print("Mean values of numerical variables by Species and Sepal.Length.Cate:")

mean\_by\_species\_cate <- iris1 %>%

group\_by(Species, Sepal.Length.Cate) %>%

summarise(across(where(is.numeric), mean, .names = "mean\_{col}"), .groups = 'drop')

print(mean\_by\_species\_cate)

# (vii) Create a pivot table based on Species and Sepal.Length.Cate

library(tidyr)

print("Pivot Table based on Species and Sepal.Length.Cate:")

pivot\_table <- iris1 %>%

count(Species, Sepal.Length.Cate) %>%

pivot\_wider(names\_from = Sepal.Length.Cate, values\_from = n, names\_prefix = "Quantile\_")

print(pivot\_table)

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

1. (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.

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

m<-mean(values)

print(m)

med<-median(values)

print(med)

mod <- as.numeric(names(sort(table(values), decreasing = TRUE)[1]))

print(mod)

# Sort the vector

sorted\_values <- sort(unique(values))

# Find the 2nd highest value

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

# Find the 3rd lowest value

third\_lowest <- sorted\_values[3]

# Display the results

cat("2nd Highest Value:", second\_highest, "\n")

cat("3rd Lowest Value:", third\_lowest, "\n")

1. Explore the airquality 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).

i. Compute the mean temperature(don’t use build in function) ii.Extract the first five rows from airquality.

iii.Extract all columns from airquality except Temp and Wind iv.Which was the coldest day during the period?

v.How many days was the wind speed greater than 17 mph?

data("airquality")

m<-mean(airquality$Temp,na.rm=TRUE)

print(m)

extr<-airquality[1:5, ]

print(extr)

exc<-airquality[,!names(airquality) %in% c("Temp","wind")]

print(exc)

cold<-airquality[which.min(airquality$Temp), ]

print(cold)

days<-sum(airquality$Wind>17,na.rm=TRUE)

print(days)

33. (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?

# Load necessary libraries

library(reshape2)

# Load the dataset

data("airquality")

# (i) Get the Summary Statistics of airquality dataset

cat("Summary Statistics of airquality dataset:\n")

print(summary(airquality))

# (ii) Melt the airquality dataset and display as a long-format data

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

cat("\nMelted airquality dataset:\n")

print(head(melted\_data))

# (iii) Melt airquality data and specify Month and Day to be "ID variables"

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

cat("\nMelted airquality dataset with ID variables:\n")

print(head(melted\_data\_with\_ids))

# (iv) Cast the melted airquality dataset with respect to Month and Day

casted\_data <- dcast(melted\_data\_with\_ids, Month + Day ~ variable, value.var = "value")

cat("\nCasted airquality dataset:\n")

print(head(casted\_data))

# (v) Compute the average of Ozone, Solar.R, Wind, and Temp per Month

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

cat("\nAverage of Ozone, Solar.R, Wind, and Temp per Month:\n")

print(average\_per\_month)

34.(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

# Load necessary libraries

library(ggplot2)

# Load the dataset

data("airquality")

# (i) Handle missing values

# Drop columns with less than 10% missing values

missing\_pct <- colMeans(is.na(airquality)) \* 100

cleaned\_data <- airquality[, missing\_pct < 10]

# Replace missing values with column mean for columns with 10% or more missing values

cleaned\_data[is.na(cleaned\_data)] <- sapply(cleaned\_data, function(x) ifelse(is.na(x), mean(x, na.rm = TRUE), x))

# (ii) Apply linear regression on Ozone and Solar.R

model <- lm(Ozone ~ Solar.R, data = cleaned\_data)

# (iii) Plot scatter plot with regression line

ggplot(cleaned\_data, aes(x = Solar.R, y = Ozone)) +

geom\_point() +

geom\_smooth(method = "lm", col = "blue") +

labs(title = "Scatter Plot of Ozone vs Solar.R with Regression Line",

x = "Solar Radiation",

y = "Ozone Concentration") +

theme\_minimal()

# Print model summary

print(summary(model))

1. 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

1. a. Create Box plot for “weight” grouped by “Diet”
   1. Create a Histogram for “weight” features belong to Diet- 1 category
   2. Create Scatter plot for “ weight” vs “Time” grouped by Diet

1. a. Create multi regression model to find a weight of the chicken , by “Time” and “Diet” as as predictor variables
   1. Predict weight for Time=10 and Diet=1
   2. Find the error in model for same
2. .For this exercise, use the (built-in) dataset Titanic.
   1. Draw a Bar chart to show details of “Survived” on the Titanic based on passenger Class
   2. Modify the above plot based on gender of people who survived
   3. Draw histogram plot to show distribution of feature “Age”
3. Explore the USArrests dataset, contains the number of arrests for murder, assault, and rape for each of the 50 states in 1973. It also contains the percentage of people in the state who live in an urban area.

(i) a. Explore the summary of Data set, like number of Features and its type. Find the number of records for each feature. Print the statistical feature of data b. Print the state which saw the largest total number of rape

c. Print the states with the max & min crime rates for murder

(ii).a. Find the correlation among the features

* 1. Print the states which have assault arrests more than median of the country
  2. Print the states are in the bottom 25% of murder

(iii). a. Create a histogram and density plot of murder arrests by US stat

* 1. Create the plot that shows the relationship between murder arrest rate and proportion of the population that is urbanised by state. Then enrich the chart by adding assault arrest rates (by colouring the points from blue (low) to red (high)).
  2. Draw a bar graph to show the murder rate for each of the 50 states .

1. 4. a. Create a data frame based on below table.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Month | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Spends | 1000 | 4000 | 5000 | 4500 | 3000 | 4000 | 9000 | 11000 | 15000 | 12000 | 7000 | 3000 |
| Sales | 9914 | 40487 | 54324 | 50044 | 34719 | 42551 | 94871 | 118914 | 158484 | 131348 | 78504 | 36284 |

b. Create a regression model for that data frame table to show the amount of sales(Sales) based on the how much the company spends (Spends) in advertising c. Predict the Sales if Spend=13500