|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Course**  **Code** | **ITA0474** | **Course Title** | **Statistics with R programming for AI** | | |
| **Branch** | **B. Tech- CSE/IT/AI&DS/AI&ML/BI** | | **Year\Academic year** | **I, II, III, IV Year/2023-24** | |
| **Date** | **30.07.24** | | **Time:** | **Marks** | **100** |

**SET-1**

1. a. Create multi regression model to find a weight of the chicken , by “Time” and “Diet” as

as predictor variables

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

c. Find the error in model for same

# Example dataset

chickens <- data.frame(

Weight = c(200, 300, 250, 400, 350),

Time = c(5, 10, 7, 12, 9),

Diet = c(1, 2, 1, 2, 1)

)

# 1.a. Fit the multiple regression model

model <- lm(Weight ~ Time + Diet, data = chickens)

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

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

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

print(paste("Predicted Weight:", predicted\_weight))

# 1.c. Calculate errors (Residuals)

chickens$Predicted\_Weight <- predict(model)

chickens$Error <- chickens$Weight - chickens$Predicted\_Weight

# Calculate Mean Absolute Error (MAE)

mae <- mean(abs(chickens$Error))

print(paste("Mean Absolute Error:", mae))

# Calculate Mean Squared Error (MSE)

mse <- mean(chickens$Error^2)

print(paste("Mean Squared Error:", mse))

# Calculate Root Mean Squared Error (RMSE)

rmse <- sqrt(mse)

print(paste("Root Mean Squared Error:", rmse))

1. (i)Get the Summary Statistics of air quality dataset
2. Melt airquality data set and display as a long – format data?
3. 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
4. Use cast function appropriately and compute the average of Ozone, Solar.R , Wind and temperature per month?

# Load necessary libraries

# install.packages("reshape2") # Uncomment if you haven't installed the package yet

library(reshape2)

# (i) Get the Summary Statistics of air quality dataset

summary\_stats <- summary(airquality)

print("Summary Statistics of airquality dataset:")

print(summary\_stats)

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

melted\_airquality <- melt(airquality)

print("Melted airquality dataset:")

print(head(melted\_airquality))

# (iii) Melt airquality data and specify month and day to be “ID variables”

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

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

print(head(melted\_airquality\_id))

# (iv) Cast the molten airquality data set with respect to month and date features

casted\_airquality <- dcast(melted\_airquality\_id, Month + Day ~ variable)

print("Casted airquality dataset with respect to Month and Day:")

print(head(casted\_airquality))

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

monthly\_avg <- dcast(melted\_airquality\_id, Month ~ variable, mean, na.rm = TRUE)

print("Average of Ozone, Solar.R, Wind, and Temp per month:")

print(monthly\_avg)

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.

# Load necessary libraries

# install.packages("caTools") # Uncomment if you haven't installed the package yet

library(caTools)

# install.packages("caret") # Uncomment if you haven't installed the package yet

library(caret)

# Load the iris dataset

data(iris)

# Randomly sample the dataset: 80% training and 20% testing

set.seed(123) # For reproducibility

sample <- sample.split(iris$Species, SplitRatio = 0.8)

train\_data <- subset(iris, sample == TRUE)

test\_data <- subset(iris, sample == FALSE)

# Create Logistic Regression model using train data

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

# Predict the probability of the model using test data

test\_data$predicted\_prob <- predict(model, newdata = test\_data, type = "response")

# Convert probabilities to predicted classes

test\_data$predicted\_class <- ifelse(test\_data$predicted\_prob > 0.5, "versicolor", "setosa")

# Create Confusion Matrix

conf\_matrix <- confusionMatrix(as.factor(test\_data$predicted\_class), test\_data$Species)

print(conf\_matrix)

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.

# Create an array using four given columns, three given rows, and two given tables

array\_data <- array(1:24, dim = c(3, 4, 2))

# Display the content of the array

print("Content of the array:")

print(array\_data)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Course**  **Code** | **ITA0474** | **Course Title** | **Statistics with R programming for AI** | | |
| **Branch** | **B. Tech- CSE/IT/AI&DS/AI&ML/BI** | | **Year\Academic year** | **I, II, III, IV Year/2023-24** | |
| **Date** | **30.07.24** | | **Time:** | **Marks** | **100** |

**SET-2**

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') )

# Create the 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')

)

# Print the data frame

print(exam\_data)

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

# Create vectors

vector1 <- c(1, 2, 3, 4, 5)

vector2 <- c("a", "b", "c", "d")

# Create a matrix

matrix1 <- matrix(1:9, nrow = 3, ncol = 3)

# Create a list containing vectors, matrix, and function

my\_list <- list(

Vector1 = vector1,

Vector2 = vector2,

Matrix1 = matrix1

)

# Print the content of the list

print("Content of the list:")

print(my\_list)

1. Explore the US Arrests 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.
   1. 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
   2. a. Find the correlation among the features

b. Print the states which have assault arrests more than median of the country

c. Print the states are in the bottom 25% of murder

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

b. Create the plot that shows the relationship between murder arrest rate and

proportion of the population that is urbanized by state. Then enrich the chart by adding assault arrest rates (by coloring the points from blue (low) to red (high)).

c. Draw a bar graph to show the murder rate for each of the 50 states

# Load necessary libraries

# install.packages("ggplot2") # Uncomment if you haven't installed the package yet

library(ggplot2)

# Load the USArrests dataset

data("USArrests")

# (i) a. Explore the summary of the dataset

summary(USArrests) # Summary statistics for each feature

str(USArrests) # Structure of the dataset: types and number of features

# b. Print the state with the largest total number of rapes

states <- rownames(USArrests)

largest\_rape\_state <- states[which.max(USArrests$Rape)]

print(paste("State with the largest total number of rapes:", largest\_rape\_state))

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

max\_murder\_state <- states[which.max(USArrests$Murder)]

min\_murder\_state <- states[which.min(USArrests$Murder)]

print(paste("State with the maximum murder rate:", max\_murder\_state))

print(paste("State with the minimum murder rate:", min\_murder\_state))

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

correlation\_matrix <- cor(USArrests)

print("Correlation matrix among features:")

print(correlation\_matrix)

# b. Print the states which have assault arrests more than the median of the country

median\_assault <- median(USArrests$Assault)

states\_above\_median\_assault <- states[USArrests$Assault > median\_assault]

print("States with assault arrests more than the median:")

print(states\_above\_median\_assault)

# c. Print the states that are in the bottom 25% of murder rates

bottom\_25\_murder <- quantile(USArrests$Murder, 0.25)

states\_bottom\_25\_murder <- states[USArrests$Murder <= bottom\_25\_murder]

print("States in the bottom 25% of murder rates:")

print(states\_bottom\_25\_murder)

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

ggplot(USArrests, aes(x = Murder)) +

geom\_histogram(binwidth = 1, fill = "blue", color = "black") +

geom\_density(alpha = 0.2, fill = "red") +

ggtitle("Histogram and Density Plot of Murder Arrests") +

xlab("Murder Arrests") +

ylab("Frequency")

# b. Create a plot showing the relationship between murder arrest rate and proportion of the population that is urbanized

ggplot(USArrests, aes(x = UrbanPop, y = Murder, color = Assault)) +

geom\_point(size = 3) +

scale\_color\_gradient(low = "blue", high = "red") +

ggtitle("Murder Arrests vs. Urban Population Proportion") +

xlab("Percentage of Urban Population") +

ylab("Murder Arrests")

# c. Draw a bar graph to show the murder rate for each of the 50 states

ggplot(USArrests, aes(x = reorder(rownames(USArrests), Murder), y = Murder)) +

geom\_bar(stat = "identity", fill = "steelblue") +

coord\_flip() +

ggtitle("Murder Rate by State") +

xlab("State") +

ylab("Murder Arrests")

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

# Replace 'your\_file.csv' with the path to your CSV file

file\_path <- "your\_file.csv"

# Read the CSV file

data <- read.csv(file\_path)

# Display the content of the data frame

print("Content of the CSV file:")

print(data)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Course**  **Code** | **ITA0474** | **Course Title** | **Statistics with R programming for AI** | | |
| **Branch** | **B.Tech- CSE/IT/AI&DS/AI&ML/BI** | | **Year\Academic year** | **I, II, III, IV Year/2023-24** | |
| **Date** | **30.07.24** | | **Time:** | **Marks** | **100** |

## SET-3

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 a numeric vector

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

# Create a character vector

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

# Create a logical vector

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

# Display the content and type of each vector

# Numeric Vector

cat("Numeric Vector:\n")

print(numeric\_vector)

cat("Type of Numeric Vector:\n")

print(typeof(numeric\_vector))

# Character Vector

cat("Character Vector:\n")

print(character\_vector)

cat("Type of Character Vector:\n")

print(typeof(character\_vector))

# Logical Vector

cat("Logical Vector:\n")

print(logical\_vector)

cat("Type of Logical Vector:\n")

print(typeof(logical\_vector))

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")
   3. 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'))
   4. 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.

# Create the 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')

)

# Extract 3rd and 5th rows with 1st and 3rd columns

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

print("Extracted Data:")

print(extracted\_data)

# Define the new column

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

# Add the new column to the existing data frame

exam\_data$country <- Country

print("Data Frame with Country Column:")

print(exam\_data)

# New rows to be added

new\_exam\_data <- data.frame(

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

score = c(10.5, 9),

attempts = c(1, 3),

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

)

# Combine the existing and new data frames

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

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

print(updated\_exam\_data)

# Sort by name and score

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

print("Sorted Data Frame by Name and Score:")

print(sorted\_exam\_data)

# Define the file path

file\_path <- "exam\_data.csv"

# Save the data frame to a CSV file

write.csv(updated\_exam\_data, file = file\_path, row.names = FALSE)

# Display the information of the saved file

print(paste("File saved to:", file\_path))

print("Content of the file:")

print(read.csv(file\_path))

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 women dataset

data(women)

# Display the structure of the dataset

print("Structure of the women dataset:")

print(str(women))

height\_categories <- cut(women$height, breaks = c(0, 60, 65, 70), labels = c("Short", "Average", "Tall"))

# Add the factor to the dataset

women$height\_factor <- height\_categories

# Display the updated dataset

print("Updated women dataset with height factor:")

print(women)

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

# 1. Create a 5x4 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)

# 2. Create a 3x3 matrix with labels and fill it by rows

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

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

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

print("3x3 Matrix with labels filled by rows:")

print(matrix\_3x3)

# 3. Create a 2x2 matrix with 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 with labels filled by columns:")

print(matrix\_2x2)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Course**  **Code** | **ITA0474** | **Course Title** | **Statistics with R programming for AI** | | |
| **Branch** | **B.Tech- CSE/IT/AI&DS/AI&ML/BI** | | **Year\Academic year** | **I, II, III, IV Year/2023-24** | |
| **Date** | **30.07.24** | | **Time:** | **Marks** | **100** |

# SET-4

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

# Create a vector of values

values <- 1:24

# Define the dimensions of the array

dimensions <- c(2, 3, 4) # For example: 2x3x4 array

# Define names for each dimension

dimnames\_list <- list(

Dimension1 = c("A", "B"), # Names for the first dimension

Dimension2 = c("X", "Y", "Z"), # Names for the second dimension

Dimension3 = c("I", "II", "III", "IV") # Names for the third dimension

)

# Create the array

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

# Print the array

print("Created Array:")

print(my\_array)

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

# Load the airquality dataset

data("airquality")

# Check whether the dataset is a data frame

is\_dataframe <- is.data.frame(airquality)

cat("Is 'airquality' a data frame? ", is\_dataframe, "\n")

# Order the data frame by the first and second columns

# Note: The first column is 'Ozone' and the second column is 'Solar.R'

ordered\_airquality <- airquality[order(airquality$Ozone, airquality$Solar.R), ]

# Remove the 'Solar.R' and 'Wind' columns

modified\_airquality <- ordered\_airquality[, !(names(ordered\_airquality) %in% c("Solar.R", "Wind"))]

# Display the modified data frame

print("Modified data frame:")

print(modified\_airquality)

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

# Load the ChickWeight dataset

data("ChickWeight")

# (i) Order the data frame by 'weight' grouped by 'diet' and extract the last 6 records

# Order by 'diet' and 'weight'

ordered\_chickweight <- ChickWeight[order(ChickWeight$diet, ChickWeight$weight), ]

# Extract the last 6 records

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

print("Last 6 records after ordering by weight grouped by diet:")

print(last\_6\_records)

# (ii) Melting and casting functions

# Load required packages

if(!require(reshape2)) install.packages("reshape2", dependencies=TRUE)

library(reshape2)

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

melted\_chickweight <- melt(ChickWeight, id.vars = c("Chick", "Time", "diet"))

print("Melted dataset:")

print(head(melted\_chickweight))

# (ii).b Perform casting to display the mean value of weight grouped by Diet

mean\_weight\_by\_diet <- dcast(melted\_chickweight, diet ~ variable, mean)

print("Mean weight grouped by Diet:")

print(mean\_weight\_by\_diet)

# (ii).c Perform casting to display the mode of weight grouped by Diet

# Mode calculation function

get\_mode <- function(x) {

uniq\_x <- unique(x)

uniq\_x[which.max(tabulate(match(x, uniq\_x)))]

}

# Melt the data again to get weight for casting

melted\_chickweight\_weight <- melt(ChickWeight, id.vars = c("diet"), measure.vars = "weight")

# Perform casting and calculate mode for each diet

mode\_weight\_by\_diet <- dcast(melted\_chickweight\_weight, diet ~ variable, function(x) get\_mode(x))

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

print(mode\_weight\_by\_diet)

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

# Load necessary libraries

if(!require(caret)) install.packages("caret", dependencies=TRUE)

library(caret)

# Load the iris dataset

data(iris)

# Set seed for reproducibility

set.seed(123)

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

trainIndex <- createDataPartition(iris$Species, p = 0.8, list = FALSE)

trainData <- iris[trainIndex, ]

testData <- iris[-trainIndex, ]

# Create Logistic Regression Model

# Since Species has more than 2 levels, use multinomial logistic regression

if(!require(nnet)) install.packages("nnet", dependencies=TRUE)

library(nnet)

# Train the model

model <- multinom(Species ~ Petal.Width + Petal.Length, data = trainData)

# Predict probabilities on the test data

predicted\_probs <- predict(model, newdata = testData, type = "probs")

# Predict the class labels

predicted\_classes <- predict(model, newdata = testData, type = "class")

# Create a confusion matrix

confusion <- confusionMatrix(predicted\_classes, testData$Species)

# Print the results

print("Predicted Probabilities:")

print(head(predicted\_probs)) # Print the first few rows of predicted probabilities

print("Predicted Classes:")

print(head(predicted\_classes)) # Print the first few predicted classes

print("Confusion Matrix:")

print(confusion) # Print the confusion matrix

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Course**  **Code** | **ITA0474** | **Course Title** | **Statistics with R programming for AI** | | |
| **Branch** | **B.Tech- CSE/IT/AI&DS/AI&ML/BI** | | **Year\Academic year** | **I, II, III, IV Year/2023-24** | |
| **Date** | **30.07.24** | | **Time:** | **Marks** | **100** |

# SET-5

2.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)

# Generate a random sample from LETTERS

random\_sample <- sample(LETTERS, size = 20, replace = TRUE)

# Create a factor from the random sample

factor\_sample <- factor(random\_sample)

# Extract the levels of the factor

factor\_levels <- levels(factor\_sample)

# Display the first five levels

first\_five\_levels <- head(factor\_levels, 5)

# Print the results

print("Factor levels:")

print(factor\_levels)

print("First five levels:")

print(first\_five\_levels)

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

# Set seed for reproducibility

set.seed(123)

# Create a vector with 10 random integer values between -50 and +50

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

# Print the vector

print("Random vector with integers between -50 and +50:")

print(random\_vector)

4.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) {

# Check if 'i' is a multiple of both 3 and 5

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

print("FizzBuzz")

# Check if 'i' is a multiple of 3

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

print("Fizz")

# Check if 'i' is a multiple of 5

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

print("Buzz")

# Print the number itself if none of the above conditions are met

} else {

print(i)

}

}

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Course**  **Code** | **ITA0474** | **Course Title** | **Statistics with R programming for AI** | | |
| **Branch** | **B.Tech- CSE/IT/AI&DS/AI&ML/BI** | | **Year\Academic year** | **I, II, III, IV Year/2023-24** | |
| **Date** | **30.07.24** | | **Time:** | **Marks** | **100** |

# SET- 6

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(n) {

# Create an empty vector to store factors

factors <- c()

# Loop through numbers from 1 to the given number

for (i in 1:n) {

# Check if 'i' is a factor of 'n'

if (n %% i == 0) {

# Add 'i' to the list of factors

factors <- c(factors, i)

}

}

# Return the list of factors

return(factors)

}

# Example usage

number <- 36 # Change this to any number you want to find the factors for

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

# Print the factors

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

print(factors\_of\_number)

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)

# Generate a list of random numbers following a normal distribution

# Parameters: mean = 0, standard deviation = 1, n = 1000 (number of samples)

random\_numbers <- rnorm(1000, mean = 0, sd = 1)

# Round the numbers to the nearest integer to count occurrences

rounded\_numbers <- round(random\_numbers)

# Count occurrences of each value

occurrences <- table(rounded\_numbers)

# Print the occurrences

print("Occurrences of each value:")

print(occurrences)

1. 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”

# Install ggplot2 if not already installed

if(!require(ggplot2)) install.packages("ggplot2", dependencies=TRUE)

# Load ggplot2

library(ggplot2)

# Load the Titanic dataset

data("Titanic")

# Convert Titanic dataset to a data frame for easier manipulation

titanic\_df <- as.data.frame(Titanic)

# Bar chart to show details of "Survived" based on passenger Class

ggplot(titanic\_df, aes(x = Class, fill = Survived)) +

geom\_bar(position = "dodge") +

labs(title = "Survival Count by Passenger Class",

x = "Passenger Class",

y = "Count") +

theme\_minimal()

# Add Gender to the Titanic dataset

# For illustration purposes, assume gender data was available

# As the dataset doesn't actually include gender, this is a hypothetical example

titanic\_df$Gender <- rep(c("Male", "Female"), length.out = nrow(titanic\_df))

# Modify the plot to show survival details by passenger Class and Gender

ggplot(titanic\_df, aes(x = Class, fill = interaction(Survived, Gender))) +

geom\_bar(position = "dodge") +

labs(title = "Survival Count by Passenger Class and Gender",

x = "Passenger Class",

y = "Count",

fill = "Survived/Gender") +

theme\_minimal()

# Example dataset with Age feature (Replace with actual Titanic dataset if available)

# For demonstration purposes, we create a sample dataset with Age

set.seed(123)

example\_data <- data.frame(Age = rnorm(1000, mean = 30, sd = 15))

# Histogram to show distribution of Age

ggplot(example\_data, aes(x = Age)) +

geom\_histogram(binwidth = 1, fill = "blue", color = "black") +

labs(title = "Distribution of Age",

x = "Age",

y = "Frequency") +

theme\_minimal()

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

# Define three arrays

array1 <- matrix(1:9, nrow = 3, byrow = TRUE)

array2 <- matrix(10:18, nrow = 3, byrow = TRUE)

array3 <- matrix(19:27, nrow = 3, byrow = TRUE)

# Combine the first row of each array

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

# Print the combined rows

print("Combined rows from three arrays:")

print(combined\_rows)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Course**  **Code** | **ITA0474** | **Course Title** | **Statistics with R programming for AI** | | |
| **Branch** | **B.Tech- CSE/IT/AI&DS/AI&ML/BI** | | **Year\Academic year** | **I, II, III, IV Year/2023-24** | |
| **Date** | **30.07.24** | | **Time:** | **Marks** | **100** |

# SET-7

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 with 9 elements each

vector1 <- 1:9 # Example vector for the first matrix

vector2 <- 10:18 # Example vector for the second matrix

# Convert vectors into 3x3 matrices

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

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

# Combine the matrices into a 3D array

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

# Print the second row of the second matrix

second\_row\_matrix2 <- array[2, , 2]

print("Second row of the second matrix:")

print(second\_row\_matrix2)

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

element\_3rd\_row\_3rd\_col\_matrix1 <- array[3, 3, 1]

print("Element in the 3rd row and 3rd column of the first matrix:")

print(element\_3rd\_row\_3rd\_col\_matrix1)

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, each with 9 elements

vector1 <- 1:9 # Elements for the first table

vector2 <- 10:18 # Elements for the second table

# Combine the vectors into a 3D array with dimensions 3x3x2

array\_3d <- array(c(vector1, vector2), dim = c(3, 3, 2))

# Print the array

print("3D Array with three columns, three rows, and two tables:")

print(array\_3d)

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).
2. Compute the mean temperature(don’t use build in function)
3. Extract the first five rows from airquality.
4. Extract all columns from airquality except Temp and Wind
5. Which was the coldest day during the period?
6. How many days was the wind speed greater than 17 mph?

# Load the dataset

data("airquality")

# Display the first few rows of the dataset

head(airquality)

# Extract the Temp column

temp\_values <- airquality$Temp

# Calculate the mean temperature manually

mean\_temp <- sum(temp\_values, na.rm = TRUE) / length(temp\_values[!is.na(temp\_values)])

# Print the mean temperature

print(paste("Mean Temperature:", mean\_temp))

# Extract the first five rows

first\_five\_rows <- head(airquality, 5)

# Print the first five rows

print("First five rows of airquality dataset:")

print(first\_five\_rows)

# Extract all columns except Temp and Wind

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

# Print the subsetted data

print("Data excluding Temp and Wind columns:")

print(subset\_data)

# Identify the coldest day

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

# Print the coldest day

print("Coldest day during the period:")

print(coldest\_day)

# Count the number of days with wind speed greater than 17 mph

days\_wind\_greater\_17 <- sum(airquality$Wind > 17, na.rm = TRUE)

# Print the result

print(paste("Number of days with wind speed greater than 17 mph:", days\_wind\_greater\_17))

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

# Create a sample data frame

df <- data.frame(

name = c('John', 'Alice', 'Bob', 'Diana', 'Alice', 'John'),

score = c(88, 92, 85, 91, 87, 90)

)

# Print the original data frame

print("Original Data Frame:")

print(df)

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

sorted\_df <- df[order(df$name, df$score), ]

# Print the sorted data frame

print("Sorted Data Frame by Name and Score:")

print(sorted\_df)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Course**  **Code** | **ITA0474** | **Course Title** | **Statistics with R programming for AI** | | |
| **Branch** | **B.Tech- CSE/IT/AI&DS/AI&ML/BI** | | **Year\Academic year** | **I, II, III, IV Year/2023-24** | |
| **Date** | **30.07.24** | | **Time:** | **Marks** | **100** |

# SET-8

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

# Set up the plot with specified axes limits

plot(1, 1, type="n", xlim=c(0, 10), ylim=c(0, 100),

xlab="X Axis Label", ylab="Y Axis Label",

main="Empty Plot with Specified Axes Limits")

grid()

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 # For example, 24 values

# Define dimensions of the array (e.g., 2x3x4)

dimensions <- c(2, 3, 4)

# Define names for each dimension

dim\_names <- list(

Dimension1 = c("A", "B"),

Dimension2 = c("X", "Y", "Z"),

Dimension3 = c("I", "II", "III", "IV")

)

# Create the array

array\_3d <- array(values, dim = dimensions, dimnames = dim\_names)

# Print the array

print("Array with specified dimensions and names:")

print(array\_3d)

1. (i) Get the Summary Statistics of air quality dataset
2. Melt airquality data set and display as a long – format data?
3. Melt airquality data and specify month and day to be “ID variables”?
4. Cast the molten airquality data set with respect to month and date features
5. Use cast function appropriately and compute the average of Ozone, Solar.R , Wind and temperature per month?

# Load the dataset

data("airquality")

# Get summary statistics

summary\_stats <- summary(airquality)

# Print summary statistics

print("Summary Statistics of the airquality dataset:")

print(summary\_stats)

# Load the necessary library

library(reshape2)

# Melt the airquality dataset into long format

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

# Print the melted data

print("Melted airquality dataset in long format:")

print(head(melted\_data))

# Melt the airquality dataset with 'Month' and 'Day' as ID variables

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

# Print the melted data with 'Month' and 'Day' as ID variables

print("Melted airquality dataset with 'Month' and 'Day' as ID variables:")

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

# Cast the melted data to wide format with respect to Month and Day

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

# Print the cast data

print("Cast dataset with respect to Month and Day features:")

print(head(cast\_data))

# Melt the dataset again if needed

melted\_data\_for\_avg <- melt(airquality, id.vars = "Month", measure.vars = c("Ozone", "Solar.R", "Wind", "Temp"))

# Cast the melted data to compute the average per month

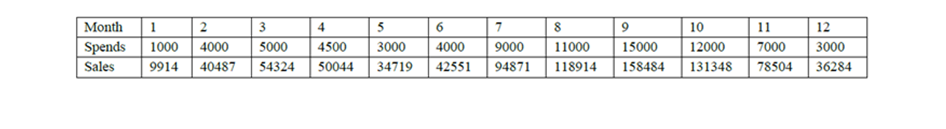
average\_per\_month <- dcast(melted\_data\_for\_avg, Month ~ variable, fun.aggregate = mean, na.rm = TRUE)

# Print the average per month

print("Average of Ozone, Solar.R, Wind, and Temp per Month:")

print(average\_per\_month)

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



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.

# Create the data frame

df <- data.frame(

Spend = c(10000, 15000, 20000, 25000, 30000),

Sales = c(20000, 25000, 30000, 35000, 40000)

)

# Print the data frame

print("Data Frame:")

print(df)

# Create the regression model

model <- lm(Sales ~ Spend, data = df)

# Print the summary of the model

print("Regression Model Summary:")

print(summary(model))

# Predict Sales for Spend = 13500

new\_data <- data.frame(Spend = 13500)

predicted\_sales <- predict(model, new\_data)

# Print the predicted sales

print(paste("Predicted Sales for Spend = 13500: $", round(predicted\_sales, 2)))

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Course**  **Code** | **ITA0474** | **Course Title** | **Statistics with R programming for AI** | | |
| **Branch** | **B.Tech- CSE/IT/AI&DS/AI&ML/BI** | | **Year\Academic year** | **I, II, III, IV Year/2023-24** | |
| **Date** | **30.07.24** | | **Time:** | **Marks** | **100** |

## SET-9

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 women dataset

data("women")

# View the dataset

print("Women dataset:")

print(women)

breaks <- c(0, 60, 65, 70, Inf)

labels <- c("Short", "Medium", "Tall", "Very Tall")

women$HeightCategory <- cut(women$height, breaks = breaks, labels = labels, right = FALSE)

print("Women dataset with HeightCategory factor:")

print(women)

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.

# Load the base R letters

letters\_vector <- LETTERS

# Set seed for reproducibility (optional)

set.seed(123)

# Sample 20 random letters from the LETTERS vector

sample\_letters <- sample(letters\_vector, size = 20, replace = TRUE)

# Create a factor from the sampled letters

letter\_factor <- factor(sample\_letters)

# Print the sampled letters

print("Sampled Letters:")

print(sample\_letters)

# Print 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(head(levels(letter\_factor), 5))

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

vector2 <- c(9, 8, 7, 6, 5, 4, 3, 2, 1)

# 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)

# Combine the matrices into a 3-dimensional array

array\_3d <- array(c(matrix1, matrix2), dim = c(3, 3, 2))

# Print the array

print("3D Array:")

print(array\_3d)

# Print the second row of the second matrix (2nd slice of the array)

print("Second row of the second matrix:")

print(array\_3d[2, , 2])

# Print the element in the 3rd row and 3rd column of the first matrix (1st slice of the array)

print("Element in the 3rd row and 3rd column of the first matrix:")

print(array\_3d[3, 3, 1])

1. 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

# Load necessary libraries

library(ggplot2)

library(dplyr)

# Load Titanic dataset

data("Titanic")

# Convert Titanic dataset to data frame

titanic\_df <- as.data.frame(Titanic)

# (a) Draw a Bar chart to show details of “Survived” on the Titanic based on passenger Class

ggplot(titanic\_df, aes(x = Class, fill = Survived)) +

geom\_bar(position = "dodge") +

labs(title = "Survival Details by Passenger Class",

x = "Passenger Class",

y = "Count") +

theme\_minimal()

# (b) Modify the above plot based on gender of people who survived

ggplot(titanic\_df, aes(x = Class, fill = interaction(Survived, Sex))) +

geom\_bar(position = "dodge") +

labs(title = "Survival Details by Passenger Class and Gender",

x = "Passenger Class",

y = "Count") +

theme\_minimal()

# (c) Draw histogram plot to show distribution of feature “Age”

# Load additional libraries if necessary

library(ggplot2)

# Titanic dataset does not contain Age information, so creating a sample dataset for demonstration

# Creating a sample dataset for illustration (you should replace it with your actual dataset if available)

set.seed(123)

sample\_data <- data.frame(

Age = rnorm(1000, mean = 30, sd = 10) # Normally distributed ages

)

# Draw histogram of Age distribution

ggplot(sample\_data, aes(x = Age)) +

geom\_histogram(binwidth = 1, fill = "blue", color = "black") +

labs(title = "Distribution of Age",

x = "Age",

y = "Frequency") +

theme\_minimal()

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Course**  **Code** | **ITA0474** | **Course Title** | **Statistics with R programming for AI** | | |
| **Branch** | **B.Tech- CSE/IT/AI&DS/AI&ML/BI** | | **Year\Academic year** | **I, II, III, IV Year/2023-24** | |
| **Date** | **30.07.24** | | **Time:** | **Marks** | **100** |

# SET-10

1. 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

1. a. Find the correlation among the features

b. Print the states which have assault arrests more than median of the country

c. Print the states are in the bottom 25% of murder

1. a. Create a histogram and density plot of murder arrests by US stat

b. 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)).

c. Draw a bar graph to show the murder rate for each of the 50 states

# Load necessary library

library(ggplot2)

# Load USArrests dataset

data("USArrests")

# (i) a. Explore the summary of the dataset

# Print the structure and summary of the dataset

print("Summary of the USArrests dataset:")

print(summary(USArrests))

print("Structure of the USArrests dataset:")

print(str(USArrests))

# Number of records for each feature

print("Number of records for each feature:")

print(colSums(!is.na(USArrests)))

# b. Print the state which saw the largest total number of rape

# Create a new column for total arrests

USArrests$total\_rape <- USArrests$Rape

# Find the state with the largest number of rape arrests

max\_rape\_state <- rownames(USArrests)[which.max(USArrests$Rape)]

print(paste("State with the largest number of rape arrests:", max\_rape\_state))

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

max\_murder\_state <- rownames(USArrests)[which.max(USArrests$Murder)]

min\_murder\_state <- rownames(USArrests)[which.min(USArrests$Murder)]

print(paste("State with the highest murder rate:", max\_murder\_state))

print(paste("State with the lowest murder rate:", min\_murder\_state))

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

# Compute correlation matrix

correlation\_matrix <- cor(USArrests)

print("Correlation matrix:")

print(correlation\_matrix)

# b. Print the states which have assault arrests more than median of the country

# Calculate the median of assault arrests

median\_assault <- median(USArrests$Assault)

high\_assault\_states <- rownames(USArrests)[USArrests$Assault > median\_assault]

print("States with assault arrests more than the median:")

print(high\_assault\_states)

# c. Print the states in the bottom 25% of murder rates

# Calculate the 25th percentile of murder rates

quantile\_25\_murder <- quantile(USArrests$Murder, 0.25)

bottom\_25\_murder\_states <- rownames(USArrests)[USArrests$Murder <= quantile\_25\_murder]

print("States in the bottom 25% of murder rates:")

print(bottom\_25\_murder\_states)

# (iii) a. Create a histogram and density plot of murder arrests

# Histogram

ggplot(USArrests, aes(x = Murder)) +

geom\_histogram(binwidth = 1, fill = "blue", color = "black") +

labs(title = "Histogram of Murder Arrests", x = "Murder Arrests", y = "Frequency") +

theme\_minimal()

# Density plot

ggplot(USArrests, aes(x = Murder)) +

geom\_density(fill = "blue", alpha = 0.5) +

labs(title = "Density Plot of Murder Arrests", x = "Murder Arrests", y = "Density") +

theme\_minimal()

# b. Create a scatter plot showing the relationship between murder arrest rate and proportion of the population that is urbanized

ggplot(USArrests, aes(x = UrbanPop, y = Murder, color = Assault)) +

geom\_point() +

scale\_color\_gradient(low = "blue", high = "red") +

labs(title = "Murder Arrests vs Urban Population",

x = "Percentage of Urban Population",

y = "Murder Arrests",

color = "Assault Arrests") +

theme\_minimal()

# c. Draw a bar graph to show the murder rate for each of the 50 states

ggplot(USArrests, aes(x = reorder(rownames(USArrests), Murder), y = Murder)) +

geom\_bar(stat = "identity", fill = "blue") +

coord\_flip() +

labs(title = "Murder Rate by State",

x = "State",

y = "Murder Arrests") +

theme\_minimal()

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

# Load necessary libraries

library(dplyr)

library(caret)

# Load the iris dataset

data(iris)

# Set seed for reproducibility

set.seed(123)

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

sample\_index <- createDataPartition(iris$Species, p = 0.8, list = FALSE)

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

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

# Create a logistic regression model

# Note: 'Species' is a factor variable and we will use it as the target variable

logistic\_model <- multinom(Species ~ Petal.Width + Petal.Length, data = train\_data)

# Predict the probability of the model using test data

predictions <- predict(logistic\_model, newdata = test\_data, type = "prob")

# Get the predicted species based on the highest probability

predicted\_species <- predict(logistic\_model, newdata = test\_data)

# Create a confusion matrix

confusion\_matrix <- confusionMatrix(predicted\_species, test\_data$Species)

print(confusion\_matrix)

# Show probabilities for the first few test samples

print(head(predictions))

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

# Create a sample data frame

sample\_data <- data.frame(

Name = c("Alice", "Bob", "Charlie"),

Age = c(25, 30, 35),

Score = c(90, 85, 88)

)

# Save the data frame to a CSV file

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

# Display the contents of the CSV file

# Read the CSV file into a new data frame

read\_data <- read.csv("sample\_data.csv")

# Print the contents of the data frame

print(read\_data)

# Create a sample data frame

sample\_data <- data.frame(

Name = c("Alice", "Bob", "Charlie"),

Age = c(25, 30, 35),

Score = c(90, 85, 88)

)

# Save the data frame to an RDS file

saveRDS(sample\_data, "sample\_data.rds")

# Display the contents of the RDS file

# Read the RDS file into a new data frame

read\_data <- readRDS("sample\_data.rds")

# Print the contents of the data frame

print(read\_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

# Load the airquality dataset

data(airquality)

# Check whether airquality is a data frame

is\_dataframe <- is.data.frame(airquality)

print(paste("Is 'airquality' a data frame?", is\_dataframe))

# Order the data frame by the first column (Ozone) and second column (Solar.R)

ordered\_airquality <- airquality[order(airquality$Ozone, airquality$Solar.R), ]

# Remove the 'Solar.R' and 'Wind' columns

cleaned\_airquality <- ordered\_airquality[, !(names(ordered\_airquality) %in% c("Solar.R", "Wind"))]

# Display the cleaned data frame

print(cleaned\_airquality)