**Course Code: ITA04 Course Title: STATISTICS WITH R PROGRAMMING**

# List of Experiments

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 = "name")

age = as.numeric(readline(prompt = "age"))

print(name)

print(age)

print(R.Version()$version.string)

a = user()

a

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

l = ls()

for (i in l){

siz = object.size(get(i))

cat("size : ",format(siz, unit = "auto"), "\n")

cat("obj name : ", i, "\n")

cat("type : ", class(i),"\n")

}

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.

a = seq(20 , 50, by = 1)

s = seq(20 : 60)

b = sum(20 : 60)/length(s)

c = sum(51 : 91)

cat(a,"\n", b,"\n", c)

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

+50.

a = sample(-50 : 50, 10, replace = TRUE)

a

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

a = 0

b = 1

i = 2

n = 10

while(i<n){

c = a + b

print(c)

a = b

b = c

i = i + 1

}

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

a = 10

for(i in 2 : a){

s = 1

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

if(i %% j ==0){

s = 0

}

}

if(s){

print(i)

}

}

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.

for(i in seq(1, 100)){

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

print("fb")

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

print("f")

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

print("b")

}

print(i)

}

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.

a = "shivaisaprogrammer"

tolower(substr(a,2,3))

toupper(substr(a,2,3))

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

a = as.numeric(readline(prompt = "enter"))

for(i in 1:(a-1)){

if(a%%i==0){

print(i)

}

}

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

a = c(1,2,3,4,5)

max = a[1]

min = a[1]

for(i in a){

if(i>max){

max = i

}else if(i<min){

min = i

}

}

min

max

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

a = c(1,1,2,2,3,4,5,4)

l = 1

b = numeric(length(a))

for(i in a){

p = 0

for(j in 1 :(l+1)){

if(i == b[j]){

p = 1

break

}

}

if(p == 0){

b[l] = i

l = l + 1

}

}

b[b!=0]

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.

a = c(1,2,3)

b = c(4,5,6)

c = c(7,8,9)

m = matrix(c(a,b,c), byrow = FALSE, nrow = 3, ncol = 3)

m

**Course Code: ITA04 Course Title: STATISTICS WITH R PROGRAMMING**

# List of Experiments

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

a = rnorm(10)

a = round(a)

a

table(a)

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

a =read.csv("C:/Users/ashiv/Desktop/New Microsoft Excel Worksheet.csv")

a

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.

a = c(1,2,34)

b = c("a", "b", "c")

c = c(TRUE, FALSE

)

print(class(c))

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 5x4 matrix

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

print(matrix\_5x4)

# Create a 3x3 matrix with labels and fill 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(matrix\_3x3)

# Create a 2x2 matrix with labels and fill by columns

matrix\_2x2 <- matrix(1:4, nrow = 2, ncol = 2, bycol = TRUE)

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

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

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.

a = array(c(1:8), dim = c(2,2,2))

dimnames(a) = list(

Row = c("r1", "r2"),

Column = c("c1", "c2"),

layer = c("l1", "l2")

)

a

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.

a = c(1,2,3)

b = c(4,5,6)

c = array(c(a,b), dim = c(3,3,2))

c

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

fun = function(x){

print(x)

}

a = list(c(1,2,3), array(c(2,2), dim = c(2,2)), fun(10))

a

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

ggplot() +

xlim(0, 10) +

ylim(0, 10) +

labs(title = "Empty Plot with Specified Axes Limits",

x = "X-axis",

y = "Y-axis") +

theme\_minimal()

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.

a = array(c(1 : 9), dim = c(3,3,2))

a[,2,2]

a[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.

a = array(1:9,dim = c(3,3))

b = array(8 :18,dim = c(3,3))

d = array(19 : 27,dim = c(3,3))

c = rbind(a[1,],b[1,],d[1,])

c

**Course Code: ITA04 Course Title: STATISTICS WITH R PROGRAMMING**

# List of Experiments

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 a vector with the values to fill the array

values <- c(1:24)

# Define the dimensions of the array

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

# Create the array using the defined values and dimensions

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

# Optionally, provide names for each dimension

dimnames(my\_array) <- list(

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

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

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

)

# Print the array

print(my\_array)

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

a = seq(50,by = 2, length.out = 15)

b = array(a,dim = c(5,3))

b

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'))

* 1. Write a R program to sort a given data frame by name and score
  2. Write a R program to save the information of a data frame in a file and display the information of the file.

a <- 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[c(3,5),c(1,3)]

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

a

b <- data.frame(

name = c('Shiva', 'Dhanu'),

score = c(12.5, 9),

attempts = c(2, 3),

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

Country = c("in", "usa")

)

c = rbind(a,b)

c

s = a[order(a$name,a$score),]

s

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

r = read.csv("sorted.csv")

r

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.

data("airquality")

c = is.data.frame(airquality)

c

orderdata = airquality[order(airquality[,1],airquality[,2]),]

orderdata

rem = subset(orderdata,select = -c(Solar.R,Wind))

rem

**Course Code: ITA04 Course Title: STATISTICS WITH R PROGRAMMING**

# List of Experiments

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.

data("women")

factor(women$height)

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.)

ran = sample(LETTERS,10,replace = TRUE)

ran

fac = factor(ran)

fac

l = levels(fac)

l

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 .
2. Find dimension, Structure, Summary statistics, Standard Deviation of all features.
3. Find mean and standard deviation of features groped by three species of Iris flowers (Iris setosa, Iris virginica and Iris versicolor)
4. 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.

data(iris)

dim(iris)

summary(iris)

sapply(iris[,1:4], sd)

mean\_by\_species = aggregate(.~iris$Species,data = iris, mean)

sd\_by\_species = aggregate(.~iris$Species, data= iris, sd)

# Quantile values of Sepal.Width

cat("Quantile values of Sepal.Width:\n")

quantile(iris$Sepal.Width)

# Quantile values of Sepal.Length

cat("Quantile values of Sepal.Length:\n")

quantile(iris$Sepal.Length)

iris$Sepal.Length.cate = cut(iris$Sepal.Length, breaks = quantile(iris$Sepal.Length))

iris1 = iris

iv-end

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("caret")

install.packages("caTools")

library(caret)

library(caTools)

# Load the iris dataset

data(iris)

# Step 1: Randomly sample the dataset

set.seed(123) # For reproducibility

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

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

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

# Step 2: Create Logistic Regression Model using caret

train\_data$Species <- factor(ifelse(train\_data$Species == "setosa", "setosa", "non-setosa"))

test\_data$Species <- factor(ifelse(test\_data$Species == "setosa", "setosa", "non-setosa"))

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

# Step 3: Predict the Probability on Test Data

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

# Convert probabilities to predicted classes

# Convert probabilities to predicted classes

predicted\_classes <- ifelse(predicted\_probabilities[, "non-setosa"] > 0.5, "non-setosa", "setosa")

# Create and display the confusion matrix

conf\_matrix <- confusionMatrix(factor(predicted\_classes, levels = c("setosa", "non-setosa")), test\_data$Species)

print(conf\_matrix)

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.

# Given values

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

# Compute mean

mean\_value <- mean(values)

cat("Mean:", mean\_value, "\n")

# Compute median

median\_value <- median(values)

cat("Median:", median\_value, "\n")

# Compute mode

compute\_mode <- function(x) {

unique\_x <- unique(x)

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

}

mode\_value <- compute\_mode(values)

cat("Mode:", mode\_value, "\n")

# Sort the values

sorted\_values <- sort(values)

# Find 2nd highest value

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

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

# Find 3rd lowest value

third\_lowest <- sorted\_values[3]

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

**Course Code: ITA04 Course Title: STATISTICS WITH R PROGRAMMING**

# List of Experiments

1. Exploíe the aiíquality dataset. It contains daily aií quality measuíements fíom New Yoík duíing a peíiod of five months:

* Ozone: mean ozone concentíation (ppb), • Solaí.R: solaí íadiation (Langley),
* Wind: aveíage wind speed (mph), • ľemp: maximum daily tempeíatuíe in degíees Fahíenheit,
* Month: numeíic month (May=5, June=6, and so on),• Day: numeíic day of the month (1 - 31).

1. Compute the mean tempeíatuíe(don’t use build in function)
2. Extíact the fiíst five íows fíom aiíquality.
3. Extíact all columns fíom aiíquality except ľemp and Wind iv.Which was the coldest day duíing the peíiod?

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

# Load the airquality dataset

data("airquality")

# Step (i): Compute the Mean Temperature (Without Using Built-In Function)

sum\_temp <- sum(airquality$Temp, na.rm = TRUE)

num\_entries <- sum(!is.na(airquality$Temp))

mean\_temp <- sum\_temp / num\_entries

cat("Mean Temperature:", mean\_temp, "\n")

# Step (ii): Extract the First Five Rows from airquality

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

cat("\nFirst Five Rows:\n")

print(first\_five\_rows)

# Step (iii): Extract All Columns from airquality Except Temp and Wind

subset\_airquality <- subset(airquality, select = -c(Temp, Wind))

cat("\nSubset of airquality (excluding Temp and Wind):\n")

print(subset\_airquality)

# Step (iv): Identify the Coldest Day During the Period

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

cat("\nColdest Day:\n")

print(coldest\_day\_row)

# Step (v): Count the Number of Days with Wind Speed Greater Than 17 mph

num\_days\_high\_wind <- sum(airquality$Wind > 17, na.rm = TRUE)

cat("\nNumber of days with wind speed greater than 17 mph:", num\_days\_high\_wind, "\n")

1. (i)Get the Summaíy Statistics of aií quality dataset
2. Melt aiíquality data set and display as a long – foímat data?
3. Melt aiíquality data and specify month and day to be “ID vaíiables”?
4. Cast the molten aiíquality data set with íespect to month and date featuíes
5. Use cast function appíopíiately and compute the aveíage of Ozone, Solaí.R , Wind and tempeíatuíe peí month?

# Load necessary libraries

install.packages("reshape2")

library(reshape2)

# Load the airquality dataset

data("airquality")

# Step (i): Get the Summary Statistics of the airquality dataset

summary\_stats <- summary(airquality)

cat("Summary Statistics:\n")

print(summary\_stats)

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

molten\_airquality <- melt(airquality)

cat("\nMolten airquality dataset (long format):\n")

print(head(molten\_airquality))

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

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

cat("\nMolten airquality dataset with Month and Day as ID variables:\n")

print(head(molten\_airquality\_id))

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

cast\_airquality <- dcast(molten\_airquality\_id, Month + Day ~ variable)

cat("\nCasted airquality dataset with respect to Month and Day:\n")

print(head(cast\_airquality))

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

average\_per\_month <- dcast(molten\_airquality\_id, Month ~ variable, 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 featuíes and díop the missing values if its less than 10% else íeplace that with mean of that featuíe.

1. Apply a lineaí íegíession algoíithm using Least Squaíes Method on “Ozone” and “Solaí.R”
2. Plot Scatteí plot between Ozone and Solaí and add íegíession line cíeated by above model

# Load necessary libraries

install.packages("ggplot2")

library(ggplot2)

# Load the airquality dataset

data("airquality")

### Step (i): Handle Missing Values

# Calculate the number of rows with missing values for each feature

missing\_values <- sapply(airquality, function(x) sum(is.na(x)))

# Calculate the total number of entries

total\_entries <- nrow(airquality)

# Drop or replace missing values based on the threshold (10%)

for (feature in names(missing\_values)) {

if (missing\_values[feature] > 0) {

missing\_percentage <- (missing\_values[feature] / total\_entries) \* 100

if (missing\_percentage < 10) {

airquality <- airquality[!is.na(airquality[[feature]]), ]

} else {

airquality[[feature]][is.na(airquality[[feature]])] <- mean(airquality[[feature]], na.rm = TRUE)

}

}

}

### Step (ii): Apply Linear Regression on "Ozone" and "Solar.R"

# Create a linear regression model

linear\_model <- lm(Ozone ~ Solar.R, data = airquality)

# Display the summary of the linear model

summary(linear\_model)

### Step (iii): Plot Scatter Plot and Add Regression Line

# Create scatter plot

ggplot(airquality, aes(x = Solar.R, y = Ozone)) +

geom\_point() + # Add scatter plot points

geom\_smooth(method = "lm", col = "blue") + # Add regression line

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

x = "Solar Radiation (Langley)",

y = "Ozone (ppb)")

1. Load dataset named ChickWeight,

( i).Oídeí the data fíame, in ascending oídeí by featuíe name “weight” gíouped by featuíe

“diet” and Extíact the last 6 íecoíds fíom oídeí data fíame.

(ii).a Peífoím melting function based on “Chick", "ľime", "Diet" featuíes as ID vaíiables

* 1. Peífoím cast function to display the mean value of weight gíouped by Diet
  2. Peífoím cast function to display the mode of weight gíouped by Diet

# Load the ChickWeight dataset

data("ChickWeight")

### Step (i): Order the data frame by weight grouped by Diet and extract the last 6 records

# Order the data frame by weight within each Diet group

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

# Extract the last 6 records from the ordered data frame

last\_six\_records <- tail(ordered\_data, 6)

cat("Last 6 records from the ordered data frame:\n")

print(last\_six\_records)

### Step (ii): Perform Melting and Casting Functions

# Load the reshape2 package

install.packages("reshape2")

library(reshape2)

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

# Perform melting function

molten\_data <- melt(ChickWeight, id.vars = c("Chick", "Time", "Diet"))

cat("\nMolten data with Chick, Time, and Diet as ID variables:\n")

print(head(molten\_data))

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

# Cast function to display the mean value of weight grouped by Diet

mean\_weight\_by\_diet <- dcast(molten\_data, Diet ~ variable, mean)

cat("\nMean value of weight grouped by Diet:\n")

print(mean\_weight\_by\_diet)

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

# Define a function to calculate the mode

compute\_mode <- function(x) {

unique\_x <- unique(x)

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

}

# Apply the function to the dataset

mode\_weight\_by\_diet <- dcast(molten\_data, Diet ~ variable, fun.aggregate = compute\_mode)

cat("\nMode value of weight grouped by Diet:\n")

print(mode\_weight\_by\_diet)

1. a. Cíeate Box plot foí “weight” gíouped by “Diet”
   1. Cíeate a Histogíam foí “weight” featuíes belong to Diet- 1 categoíy
   2. Cíeate Scatteí plot foí “ weight” vs “ľime” gíouped by Diet

# Load necessary libraries

install.packages("ggplot2")

install.packages("reshape2")

library(ggplot2)

library(reshape2)

# Load the ChickWeight dataset

data("ChickWeight")

### Step (i): Order the data frame by weight grouped by Diet and extract the last 6 records

# Order the data frame by weight within each Diet group

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

# Extract the last 6 records from the ordered data frame

last\_six\_records <- tail(ordered\_data, 6)

cat("Last 6 records from the ordered data frame:\n")

print(last\_six\_records)

### Step (ii): Perform Melting and Casting Functions

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

# Perform melting function

molten\_data <- melt(ChickWeight, id.vars = c("Chick", "Time", "Diet"))

cat("\nMolten data with Chick, Time, and Diet as ID variables:\n")

print(head(molten\_data))

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

# Cast function to display the mean value of weight grouped by Diet

mean\_weight\_by\_diet <- dcast(molten\_data, Diet ~ variable, mean)

cat("\nMean value of weight grouped by Diet:\n")

print(mean\_weight\_by\_diet)

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

# Define a function to calculate the mode

compute\_mode <- function(x) {

unique\_x <- unique(x)

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

}

# Apply the function to the dataset

mode\_weight\_by\_diet <- dcast(molten\_data, Diet ~ variable, fun.aggregate = compute\_mode)

cat("\nMode value of weight grouped by Diet:\n")

print(mode\_weight\_by\_diet)

### Step (iii): Create Plots Using ggplot2

#### (a) Create Box Plot for "weight" Grouped by "Diet"

ggplot(ChickWeight, aes(x = as.factor(Diet), y = weight)) +

geom\_boxplot() +

labs(title = "Box Plot of Weight by Diet",

x = "Diet",

y = "Weight") +

theme\_minimal()

#### (b) Create a Histogram for "weight" Features Belonging to Diet-1 Category

ggplot(subset(ChickWeight, Diet == 1), aes(x = weight)) +

geom\_histogram(binwidth = 10, fill = "blue", color = "black") +

labs(title = "Histogram of Weight for Diet-1",

x = "Weight",

y = "Frequency") +

theme\_minimal()

#### (c) Create Scatter Plot for "weight" vs "Time" Grouped by Diet

ggplot(ChickWeight, aes(x = Time, y = weight, color = as.factor(Diet))) +

geom\_point() +

geom\_smooth(method = "lm", se = FALSE) +

labs(title = "Scatter Plot of Weight vs Time by Diet",

x = "Time",

y = "Weight",

color = "Diet") +

theme\_minimal()

**Course Code: ITA04 Course Title: STATISTICS WITH R PROGRAMMING**

# List of Experiments

1. a. Cíeate multi íegíession model to find a weight of the chicken , by “ľime” and “Diet” as as píedictoí vaíiables
   1. Píedict weight foí ľime=10 and Diet=1
   2. Find the eííoí in model foí same

# Load necessary libraries

install.packages("ggplot2")

library(ggplot2)

# Load the ChickWeight dataset

data("ChickWeight")

### Step (a): Create Multiple Regression Model

# Create the multiple regression model

multi\_reg\_model <- lm(weight ~ Time + Diet, data = ChickWeight)

# Display the summary of the model

summary(multi\_reg\_model)

### Step (b): Predict Weight for Time = 10 and Diet = 1

# Create a new data frame for the prediction

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

# Predict the weight

predicted\_weight <- predict(multi\_reg\_model, newdata = new\_data)

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

### Step (c): Find the Error in the Model

# Calculate predictions for the entire dataset

predictions <- predict(multi\_reg\_model, newdata = ChickWeight)

# Calculate residuals (errors)

residuals <- ChickWeight$weight - predictions

# Calculate Mean Absolute Error (MAE)

mae <- mean(abs(residuals))

cat("Mean Absolute Error (MAE):", mae, "\n")

# Calculate Root Mean Squared Error (RMSE)

rmse <- sqrt(mean(residuals^2))

cat("Root Mean Squared Error (RMSE):", rmse, "\n")

1. .Foí this exeícise, use the (built-in) dataset ľitanic.
2. Díaw a Baí chaít to show details of “Suívived” on the ľitanic based on passengeí Class
3. Modify the above plot based on gendeí of people who suívived
4. Díaw histogíam plot to show distíibution of featuíe “Age”

# Install and load necessary packages

install.packages("titanic")

install.packages("ggplot2")

library(titanic)

library(ggplot2)

# Load the Titanic dataset

data("titanic\_train")

titanic\_data <- titanic\_train

### Section (a): Draw a Bar Chart to Show Details of "Survived" Based on Passenger Class

# Bar chart for "Survived" based on "Pclass"

ggplot(titanic\_data, aes(x = as.factor(Pclass), fill = as.factor(Survived))) +

geom\_bar(position = "dodge") +

labs(title = "Survival by Passenger Class",

x = "Passenger Class",

y = "Count",

fill = "Survived") +

theme\_minimal()

### Section (b): Modify the Above Plot Based on Gender of People Who Survived

# Bar chart for "Survived" based on "Pclass" and "Sex"

ggplot(titanic\_data, aes(x = as.factor(Pclass), fill = as.factor(Survived))) +

geom\_bar(position = "dodge") +

facet\_wrap(~ Sex) +

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

x = "Passenger Class",

y = "Count",

fill = "Survived") +

theme\_minimal()

### Section (c): Draw a Histogram to Show Distribution of Feature "Age"

# Histogram for the distribution of "Age"

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

geom\_histogram(binwidth = 5, fill = "blue", color = "black") +

labs(title = "Age Distribution of Titanic Passengers",

x = "Age",

y = "Frequency") +

theme\_minimal()

1. Exploíe the USAííests dataset, contains the numbeí of aííests foí muídeí, assault, and íape foí each of the 50 states in 1973. It also contains the peícentage of people in the state who live in an uíban aíea.

(i) a. Exploíe the summaíy of Data set, like numbeí of Featuíes and its type. Find the numbeí of íecoíds foí each featuíe. Píint the statistical featuíe of data

b. Píint the state which saw the laígest total numbeí of íape

c. Píint the states with the max & min cíime íates foí muídeí (ii).a. Find the coííelation among the featuíes

b. Píint the states which have assault aííests moíe than median of the countíy

c. Píint the states aíe in the bottom 25% of muídeí

(iii). a. Cíeate a histogíam and density plot of muídeí aííests by US stat

b. Cíeate the plot that shows the íelationship between muídeí aííest íate and píopoítion of the population that is uíbanised by state. ľhen eníich the chaít by adding assault aííest íates (by colouíing the points fíom blue (low) to íed (high)).

c. Díaw a baí gíaph to show the muídeí íate foí each of the 50 states .

# Load the USArrests dataset

data("USArrests")

### (i) Exploration and Summary

#### (i.a) Summary of the Dataset

# Summary of the dataset

summary(USArrests)

# Number of features and their types

cat("Number of features:", ncol(USArrests), "\n")

cat("Feature types:\n")

sapply(USArrests, class)

# Number of records for each feature

cat("Number of records for each feature:\n")

sapply(USArrests, length)

#### (i.b) State with the Largest Total Number of Rapes

# State with the largest total number of rapes

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

cat("State with the largest total number of rapes:", max\_rape\_state, "\n")

#### (i.c) States with the Max & Min Crime Rates for Murder

# State with maximum murder rate

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

cat("State with the maximum murder rate:", max\_murder\_state, "\n")

# State with minimum murder rate

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

cat("State with the minimum murder rate:", min\_murder\_state, "\n")

### (ii) Further Analysis

#### (ii.a) Correlation Among Features

# Correlation matrix

correlation\_matrix <- cor(USArrests)

cat("Correlation matrix:\n")

print(correlation\_matrix)

#### (ii.b) States with Assault Arrests More than Median of the Country

# Median of assault arrests

median\_assault <- median(USArrests$Assault)

# States with assault arrests more than median

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

cat("States with assault arrests more than the median:\n")

print(high\_assault\_states)

#### (ii.c) States in the Bottom 25% of Murder Rates

# 25th percentile (quantile) of murder rates

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

# States in the bottom 25% of murder rates

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

cat("States in the bottom 25% of murder rates:\n")

print(low\_murder\_states)

### (iii) Visualization

#### (iii.a) Histogram and Density Plot of Murder Arrests

# Load ggplot2 package

install.packages("ggplot2")

library(ggplot2)

# Histogram of murder arrests

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

geom\_histogram(binwidth = 1, fill = "blue", color = "black", alpha = 0.7) +

labs(title = "Histogram of Murder Arrests by US State",

x = "Murder Arrests",

y = "Frequency") +

theme\_minimal()

# Density plot of murder arrests

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

geom\_density(fill = "blue", alpha = 0.7) +

labs(title = "Density Plot of Murder Arrests by US State",

x = "Murder Arrests",

y = "Density") +

theme\_minimal()

#### (iii.b) Plot Showing Relationship Between Murder Arrest Rate and Proportion of Urban Population, Colored by Assault Arrest Rates

# Scatter plot with murder arrest rate and urban population

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

geom\_point(size = 4, alpha = 0.7) +

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

labs(title = "Murder Arrest Rate vs. Urban Population",

x = "Urban Population (%)",

y = "Murder Arrest Rate",

color = "Assault Arrest Rate") +

theme\_minimal()

#### (iii.c) Bar Graph to Show the Murder Rate for Each of the 50 States

# Bar graph for murder rate by state

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

geom\_bar(stat = "identity", fill = "blue", alpha = 0.7) +

coord\_flip() +

labs(title = "Murder Rate by State",

x = "State",

y = "Murder Arrest Rate") +

theme\_minimal()

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

# Create the data frame

data <- data.frame(

Month = 1:12,

Spends = c(1000, 4000, 5000, 4500, 3000, 4000, 9000, 11000, 15000, 12000, 7000, 3000),

Sales = c(9914, 40487, 54324, 50044, 34719, 42551, 94871, 118914, 158484, 131348, 78504, 36284)

)

# Display the data frame

print(data)

# Create the regression model

regression\_model <- lm(Sales ~ Spends, data = data)

# Display the summary of the regression model

summary(regression\_model)

# Create a new data frame for the prediction

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

# Predict the sales

predicted\_sales <- predict(regression\_model, newdata = new\_data)

cat("Predicted Sales for Spend = 13500:", predicted\_sales, "\n")