SIDDHARTH INSTITUTE OF ENGINEERING & TECHNOLOGY
(AUTONOMOUS)
(Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu)
(Accredited by NAAC with 'A+' grade)
(Accredited by NBA for CIVIL, EEE, MECH, ECE & CSE Courses)
SIDDHARTH NAGAR, NARAYANAVANAM ROAD, PUTTUR – 517 583
TIRUPATI DIST., A.P., INDIA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & DATA SCIENCE)

LAB MANUAL

R PROGRAMMING LAB

(20CS1108)

III B.TECH -II SEMESTER

SIDDHARTH INSTITUTE OF ENGINEERING & TECHNOLOGY

VISION

To become an eminent academic institute for academic and research that producing global leaders in science and technology to serve the betterment of mankind.

MISSION

M1. To provide broad-based education and contemporary knowledge by adopting modern teaching-learning methods.

M2. To inculcate a spirit of research and innovation in students through industrial interactions.

M3. To develop individual's potential to its fullest extent so that they can emerge as gifted leaders in their fields.

DEPARTMENT OF COMPUETR SCIENCE AND ENGINEERING

VISION

To become a well-known department of Computer Science and Engineering producing competent professionals with research and innovation skills, inculcating moral values and societal concerns.

MISSION OF THE DEPARTMENT

M1. To educate students to become highly qualified computer engineers with full commitments to professional ethics.

M2. To inculcate a mind of innovative research in the field of computer science and related interdisciplinary areas to provide advanced professional service to the society.

M3. To prepare students with industry ready knowledge base as well as entrepreneurial skills by introducing duly required industry oriented educational program.

PROGRAM EDUCATIONAL OBJECTIVES STATEMENTS

PEO1. Graduates with basic and advanced knowledge in science, mathematics, computer science and allied engineering, capable of analyzing, design and development of solutions for real life problems.

PEO2. Graduates who serve the Industry, consulting, government organizations, or who pursue higher education or research.Graduates with qualities of professional leadership, communication skills, team work, ethical values and lifelong learning abilities.

PROGRAMME OUTCOMES (PO'S):

PO1 **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2 **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3 **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4 **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5 **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6 **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7 **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8 **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9 **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10 **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11 **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12 **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

PROGRAMME SPECIFIC OUTCOMES (PSO'S):

PSO1 **Architecture of Computer System**: Ability to visualize and articulate computer hardware and software systems for various complex applications.

PSO2 **Design and develop computer programs**: Ability to design and develop computer-based systems in the areas related to algorithms, networking, web design, cloud computing, IoT, data analytics and mobile applications of varying complexity.

PSO3 **Applications of Computing and Research Ability:** Ability to use knowledge in various domains to identify research gaps and hence to provide solution to new ideas and innovations.

SIDDHARTH INSTITUTE OF ENGINEERING & TECHNOLOGY:: PUTTUR

(AUTONOMOUS)

L	T	P	C
-	-	2	1.5

III B. Tech – II Sem.

(20CS1108) R PROGRAMMING LAB

COURSE OBJECTIVES

The objectives of this course:

1. Understand the fundamentals of 'R' programming
2. Learn how to carry out a range of commonly used statistical methods including analysis of variance and linear regression.
3. Explore data-sets to create testable hypotheses and identify appropriate statistical tests.

COURSE OUTCOMES (COs)

On successful completion of this course, student will be able to:

1. Implement R Analytics to create Business Insights
2. Analyze the data and results using R.
3. Apply analytical methods and produce presentation quality graphics.
4. Explore data-sets to create testable hypotheses.
5. Perform appropriate statistical tests using R.
6. Create and edit visualizations with R.

1. a) R Multiplication Table
   b) Check if a Number is Odd or Even in R Programming

2. a) R Program to Check for Leap Year
   b) R Program to Check if a Number is Positive, Negative or Zero

3. a) R Program to Find the Sum of Natural Numbers
   b) R Program to Print the Fibonacci Sequence

4. a) R Program to check Armstrong Number
   b) R Program to Check Prime Number

5. a) R Program to Find the Factors of a Number
   b) R program to Find the Factorial of a Number Using Recursion

6. a) Convert Decimal into Binary using Recursion in R
   b) Fibonacci Sequence Using Recursion in R

7. a) R Program to Find H.C.F. or G.C.D.
   b) R Program to Find L.C.M.

8. R Program to Make a Simple Calculator

9. a) Creating Vectors and sequences numbers
   b) Manipulating data frames and lists.
10. a) Writing functions in R. using If/else statements.
    b) Write a Program on For/while loops
11. a) Write a Program on Using apply() to iterate over data.
    b) Write a Program on Using with() to specify environment.
12. a) Multivariate statistical summaries using plyr
    b) Program using ggplot2 graphics
13. a) Write a Program on QQ plots
    b) Write a Program on Interpreting categorical variables in regression

TEXT BOOK

1. Dr. Mark Gardener, Beginning R the statistical programming language, Wiley Publications, 2015.

REFERENCES

1. Grolemund & Garrett, Hands-On Programming with R Paperback, SPD,20142. Michael J. Crawley, The R Book WILEY, 2012.

SIDDHARTH INSTITUTE OF ENGINEERING & TECHNOLOGY : PUTTUR
(AUTONOMOUS)

DEPARTMENT OF COMPUETR SCIENCE AND ENGINEERING

Do's

- Follow the right procedures for starting the computer and shutting down the computer to avoid loss of data and damage of computer components
- You should only be on the specific program/site your lab faculty has assigned for you.
- Report any problems/damages immediately to the lab In-charge.
- Always use a light touch on the keyboard.
- When working on documents always save several times while working on it.
- Remember to log out whenever you are done using any lab computer.
- Shut down computer properly before you leave the lab.
- Know the location of the fire extinguisher and the first aid box and how to use them in case of an emergency.
- Read and understand how to carry out an experiment thoroughly before coming to the laboratory.

Dont's

- Computers and peripherals are not to be moved or reconfigured without approval of Lab In-charge.
- Students may not install software on lab computers. If you have a question regarding specific software that you need to use, contact the Lab In-charge.
- Never bang on the keys.
- Do not open up the metallic covers of computers or peripherals devices a particularly when the power is on.
- Never open attachments from unknown sources.
- Keep all liquids away from computers and equipment, liquid may spill and cause an electrical shock or the computer not to operate properly.
- Do not insert metal objects such as clips, pins and needles into the computer casings. They may cause fire.
- Do not touch, connect or disconnect any plug or cable without Lab in-charge's permission.

1.a	**R Multiplication Table**

AIM: To write a R program to implement Multiplication Table.

Program:

```
#  For loop is used and take input from the user
number <- as.integer(readline(prompt = "Please Enter a Number for Table"))
# For iterating from one to ten, for loop is used
for( t in 1:10)
{
   print ( paste ( number, '*', t, '=', number * t))
}
```

Output:

Result :

1.b	Check if a Number is Odd or Even in R Programming

AIM: To write to check if a Number is Odd or Even in R Programming

Program:

```
num = as.integer(readline(prompt="Enter a number: "))
if((num %% 2) == 0) {
print(paste(num,"is Even number"))
} else {
print(paste(num,"is Odd number"))
}
```

Output:

Result :

2.a	R Program to Check for Leap Year

AIM: To write a R program to Check for Leap Year

Program:

```
year = as.integer(readline(prompt="Enter a year: "))
if((year %% 4) == 0) {
   if((year %% 100) == 0) {
      if((year %% 400) == 0) {
         print(paste(year,"is a leap year"))
      } else {
         print(paste(year,"is not a leap year"))
      }
   } else {
      print(paste(year,"is a leap year"))
   }
} else {
   print(paste(year,"is not a leap year"))
}
```

Output :

Result:

2.b	R Program to Check if a Number is Positive, Negative or Zero

AIM: To write a R program to Check if a Number is Positive, Negative or Zero.

Program:

```
number = as.double(readline(prompt="Enter a number: "))
if(number > 0)
{
  print("It is a Positive number")
} else
{
  if(number == 0)
  {
     print("number is Zero")
  }
  else
  {
    print("It is a Negative number")
  }
}
```

Output:

Result:

3.a	R Program to Find the Sum of Natural Numbers

AIM: To write a R program to Find the Sum of Natural Numbers.

Program:

```
n = as.integer(readline(prompt = "Enter a number: "))
if(n< 0) {
print("Enter a positive number")
} else {
sum = 0
# use while loop to iterate until zero
while(n > 0) {
sum = sum + n
n = n - 1
}
print(paste("The sum of numbers up to the given limit is", sum))
}
```

Output :

Result :

3.b	R Program to Print the Fibonacci Sequence

AIM: To write a R program to Print the Fibonacci Sequence

Program:

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

num1 = 0 # first number
num2 = 1 # second number
count = 2
# check if the total_terms  is valid
if (total_terms  <= 0) {
   print("Please enter a positive integer")
} else {
   if (total_terms == 1) {
      print("Fibonacci sequence:")
      print(num1)
   } else {
      print("Fibonacci sequence:")
      print(num1)
      print(num2)
      while (count < total_terms  ) {
         nxt = num1 + num2
         print(nxt)
         # update values
         num1 = num2
         num2 = nxt
         count = count + 1
      }
   }
}
```

Output :

Result :

4.a	R Program to check Armstrong Number

AIM: To write a R program to check Armstrong Number

Program:

```
# take input from the user
num = as.integer(readline(prompt="Enter a number: "))
# initialize sum
sum = 0
# find the sum of the cube of each digit
temp = num
while(temp > 0)
{
digit = temp %% 10
sum = sum + (digit ^ 3)
temp = floor(temp / 10)
}
# display the result
if(num == sum) {
print(paste(num, "is an Armstrong number"))
} else {
print(paste(num, "is not an Armstrong number"))
}
```

Output :

Result :

4.b	R Program to Check Prime Number

AIM: To write a R program to Check Prime Number

Program:

```
# Program to check if the input number is prime or not

# take input from the user
num = as.integer(readline(prompt="Enter a number: "))

flag = 0
# prime numbers are greater than 1
if(num > 1) {
   # check for factors
   flag = 1
   for(i in 2:(num-1)) {
      if ((num %% i) == 0) {
         flag = 0
         break
      }
   }
}

if(num == 2)
         flag = 1
if(flag == 1) {
   print(paste(num,"is a prime number"))
} else {
   print(paste(num,"is not a prime number"))
}
```

Output:

Result :

5.a	R Program to Find the Factors of a Number

AIM: To write to Find the Factors of a Number in R programming.

Program:

```
print_factors <- function(k) {
print(paste("The factors of given number",k,"are:"))
for(i in 1:k) {
if((k %% i) == 0) {
print(i)
}
}
}
print_factors(7)
```

Output:

Result:

5.b	R program to Find the Factorial of a Number Using Recursion

AIM: To write to calculate Factorial of a Number Using Recursion.

Program:

```
recur_fact <- function(num) {
if(num <= 1) {
return(1)
} else {
return(num * recur_fact(num-1))
}
}

print(paste("The factorial of 5 is",recur_fact (5)))
```

Output :

Result:

6.a	Convert Decimal into Binary using Recursion in R

AIM: To write to Convert Decimal into Binary using Recursion in R.

Program:

```
convert_to_binary <- function(decnum) {
if(decnum > 1) {
convert_to_binary(as.integer(decnum/2))
}
cat(decnum %% 2)
}
convert_to_binary(10)
```

output :

Result :

6.b	Fibonacci Sequence Using Recursion in R

AIM: To write to implement Fibonacci Sequence Using Recursion in R.

Program:

```
fibonacci <- function(n) {
   if(n <= 1) {
      return(n)
   } else {
      return(fibonacci(n-1) + fibonacci(n-2))
   }
}

# take input from the user
total_terms = as.integer(readline(prompt="How many terms? "))

# check if the total_terms is valid or not
if(total_terms<= 0) {
   print("Plese enter a positive integer")
} else {
   print("Fibonacci sequence:")
   for(i in 0:(total_terms-1)) {
      print(fibonacci(i))
   }
}
```

Output :

Result :

7.a	R Program to Find H.C.F. or G.C.D.

AIM: To write to R Program to Find H.C.F. or G.C.D..

Program:

```
hcf <- function(x, y) {
# choose the smaller number
if(x > y) {
smaller = y
} else {
smaller = x
}
for(i in 1:smaller) {
if((x %% i == 0) && (y %% i == 0)) {
hcf = i
}
}
return(hcf)
}
# take input from the user
n1 = as.integer(readline(prompt = "Enter first number: "))
n2 = as.integer(readline(prompt = "Enter second number: "))
print(paste("The H.C.F. of", n1,"and", n2,"is", hcf(n1, n2)))
```

Output :

Result :

7.b	R Program to Find L.C.M.

AIM: To write to R Program to Find L.C.M

Program:

```
lcm <- function(x, y) {
# choose the greater number
if(x > y) {
greater = x
} else {
greater = y
}
while(TRUE) {
if((greater %% x == 0) && (greater %% y == 0)) {
lcm = greater
break
}
greater = greater + 1
}
return(lcm)
}
# take input from the user
n1 = as.integer(readline(prompt = "Enter first number: "))
n2 = as.integer(readline(prompt = "Enter second number: "))
print(paste("The L.C.M. of", n1,"and", n2,"is", lcm(n1, n2)))
```

Output :

Result :

8	R Program to Make a Simple Calculator

AIM: To write to R Program to Make a Simple Calculator.

Program:

```
add <- function(x, y) {
return(x + y)
}
subtract <- function(x, y) {
return(x - y)
}
multiply <- function(x, y) {
return(x * y)
}
divide <- function(x, y) {
return(x / y)
}
# take input from the user
print("Select operation.")
print("1.Add")
print("2.Subtract")
print("3.Multiply")
print("4.Divide")
choice = as.integer(readline(prompt="Enter choice[1/2/3/4]: "))
n1 = as.integer(readline(prompt="Enter first number: "))
n2 = as.integer(readline(prompt="Enter second number: "))
operator <- switch(choice,"+","-","*","/")
result <- switch(choice, add(n1, n2), subtract(n1, n2), multiply(n1, n2), divide(n1, n2))
print(paste(n1, operator, n2, "=", result))
```

Output:

Result:

9.a	Creating Vectors and sequences numbers

AIM: To write to Creating Vectors and sequences numbers in R programming.

Program:

```
numeric_vector <- c(1, 2, 3, 4, 5)
cat("Numeric Vector:", numeric_vector, "\n")

character_vector <- c("apple", "banana", "orange", "grape")
cat("Character Vector:", character_vector, "\n")

sequence_vector <- 1:5
cat("Sequence Vector (using colon operator):", sequence_vector, "\n")

seq_vector <- seq(from = 1, to = 10, by = 2)
cat("Sequence Vector (using seq() function):", seq_vector, "\n")

len_vector <- seq_len(5)
cat("Sequence Vector (using seq_len() function):", len_vector, "\n")
```

Output :

Result:

9.b	Manipulating data frames and lists.

AIM: To write a R program Manipulating data frames and lists.

Program:

i) Manipulating data frames :

```
#create 3 vectors Roll_No, Student_Name, Grade using c()
Roll_No = c(1,2,3,4,5,6) #numeric type
Student_Name = c("ram","krishna","sai","ganga","balaji","siva") #character type
Grade = c("B","A","O","A","C","A") #character type
#Created a data frame and assigned to Variable D
D = data.frame(Roll_No,Student_Name,Grade )

print(D)
```

ii) Manipulating lists :

```
Student_Name = c("sai","ram","krishna") #vectors are created of character, numeric data types
Student_Rollno = c(501,151,519)
class = c("cse")
section=c("A","B","C")
item = c("Singing","Playing","Dance")
student_list = list(Student_Name,Student_Rollno,class,section,item)    #a list is created using list()
print(student_list)    #prints the result
```

Output:

Result :

10.a	Writing functions in R. using If/else statements.

AIM: To Writing functions in R. using If/else statements.

Program:

```
# Function to check if a number is positive, negative, or zero
check_number <- function(x)
{
  if (x > 0)
  {
    result <- "Positive"
  } else if (x < 0)
  {
    result <- "Negative"
  } else
  {
    result <- "Zero"
  }

  return(result)
}

number <- as.numeric(readline("Enter a number: "))

if (is.na(number))
{
  cat("Invalid input. Please enter a valid number.\n")
} else
{
  result <- check_number(number)
  cat("The entered number is:", result, "\n")
}
```

Output :

Result :

10.b	Write a Program on For/while loops

AIM: To Write a R Program on For/while loops

Program:

```
# Function to calculate the sum of the first N natural numbers using a for loop
sum_of_natural_numbers <- function(N)
{
  total_sum <- 0

  for (i in 1:N)
  {
    total_sum <- total_sum + i
  }

  return(total_sum)
}
N <- as.integer(readline("Enter a positive integer N: "))

if (is.na(N) || N <= 0)
{
  cat("Invalid input. Please enter a positive integer.\n")
} else
{
  result <- sum_of_natural_numbers(N)
  cat("The sum of the first", N, "natural numbers is:", result, "\n")
}
```

Output :

Result :

11.a	Write a Program on Using apply() to iterate over data.

AIM: To Write a Program on Using apply() to iterate over data.

Program:

```
# Create a sample matrix
sample_matrix <- matrix(1:12, nrow = 3, byrow = TRUE)
# Function to calculate the sum of each row using apply()
sum_of_rows <- apply(sample_matrix, 1, sum)
# Display the original matrix and the sum of each row
cat("Original Matrix:\n")
print(sample_matrix)
cat("\nSum of Each Row:\n")
print(sum_of_rows)
```

Output :

Result :

11.b	Write a Program on Using with() to specify environment.

AIM: To Write a Program on Using with() to specify environment.

Program:

```
# Creating a custom environment
custom_environment <- new.env()

# Assigning variables in the custom environment
custom_environment$x <- 5
custom_environment$y <- 10

# Function to perform calculations using with()
calculate_sum_product <- function(env)
{
  with(env,
  {
    sum_result <- x + y
    product_result <- x * y

    cat("Sum:", sum_result, "\n")
    cat("Product:", product_result, "\n")
  })
}
```

Output :

Result :

12.a	Multivariate statistical summaries using plyr

AIM: To implement Multivariate statistical summaries using plyr in R programming.

Program:

```
# Install and load the plyr package
#install.packages("plyr")
#library(plyr)

# Load the mtcars dataset
data(mtcars)

# Function to calculate mean and standard deviation
summary_stats <- function(x) {
  mean_val <- mean(x)
  sd_val <- sd(x)
  return(c(mean = mean_val, sd = sd_val))
}

# Multivariate statistical summaries using plyr
result <- ddply(mtcars, .(gear, cyl), summarise,
                mpg_mean = summary_stats(mpg)['mean'],
                mpg_sd = summary_stats(mpg)['sd'],
                hp_mean = summary_stats(hp)['mean'],
                hp_sd = summary_stats(hp)['sd'])

# Display the result
print(result)
```

Output :

Result :

12.b	Program using ggplot2 graphics

AIM: To implement using ggplot2 graphics in R programming.

Program:

```
# Install and load the ggplot2 package
install.packages("ggplot2")
library(ggplot2)

# Load the iris dataset
data(iris)

# Create a scatter plot using ggplot2
scatter_plot <- ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +
  geom_point() +
  labs(title = "Scatter Plot of Sepal Length vs Sepal Width",
       x = "Sepal Length",
       y = "Sepal Width",
       color = "Species")

# Display the scatter plot
print(scatter_plot)
```

Output :

Result :

```
# Load the ggplot2 package
library(ggplot2)
# Load the mpg dataset
data(mpg)
# Create a bar chart using ggplot2
bar_chart <- ggplot(mpg, aes(x = class, y = hwy, fill = class)) +
  geom_bar(stat = "summary", fun = "mean", position = "dodge") +
  labs(title = "Average Highway MPG by Vehicle Class",
       x = "Vehicle Class",
       y = "Average Highway MPG",
       fill = "Vehicle Class")

# Display the bar chart
print(bar_chart)
```

Output :

Result :

13.a	Write a Program on QQ plots

AIM: To implement using QQ plots in R programming.

Implementation of Basic QQplot Interpretation using qqline() Function in R:

```
# Set seed for reproducibility
set.seed(500)
# Create random normally distributed values
x <- rnorm(1200)
# QQplot of normally distributed values
qqnorm(x)
# Add qqline to plot
qqline(x, col = "darkgreen")
```

Implementation of QQplot of Logistically Distributed Values:

```
# Set seed for reproducibility
 # Random values according to logistic distribution
# QQplot of logistic distribution
y <- rlogis(800)
 # QQplot of normally distributed values
qqnorm(y)
 # Add qqline to plot
qqline(y, col = "darkgreen")
```

Output :

Result :

13.a	Write a Program on Interpreting categorical variables in regression

AIM: To generate Program on Interpreting categorical variables in regression.

Program 1 :

```
df <- data.frame(points=c(7, 7, 9, 10, 13, 14, 12, 10, 16, 19, 22, 18),
                 hours=c(1, 2, 2, 3, 2, 6, 4, 3, 4, 5, 8, 6),
                 program=c(1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3))
#convert 'program' to factor
df$program <- as.factor(df$program)
#fit linear regression model
fit <- lm(points ~ hours + program, data = df)
#view model summary
summary(fit)
Call:
lm(formula = points ~ hours + program, data = df)
```

Program 2:

```
set.seed(123)
data <- data.frame(
  Age = c(25, 30, 35, 40, 45, 50),
  Gender = c("Male", "Female", "Male", "Female", "Male", "Female"),
  Income = c(50000, 60000, 75000, 80000, 90000, 100000)
)
# Convert Gender to a factor with two levels (Male and Female)
data$Gender <- as.factor(data$Gender)
# Create dummy variables for Gender
dummy_gender <- model.matrix(~ Gender - 1, data = data)
# Combine the original data with dummy variables
data <- cbind(data, dummy_gender)
# Linear regression model
model <- lm(Income ~ Age + Male + Female, data = data)
# Summary of the model
summary(model)
```

Output :

Result :