

**.Department of Computer Science & Engineering.**

Laboratory Manual

***Subject: Data Analysis using R Programming Subject Code: CS1602-1***

|  |  |
| --- | --- |
| *Course Title:* ***Data Analysis using R Programming*** | *Course Code****: CS1602-1*** |
| *Total Teaching Hours :****0+0+2*** | *Course Type:* ***PCC Lab*** |
| *CIE + SEE Marks* : ***50+50*** | *Credits :* ***1*** |
| *Lab Manual Author****:*** | *Duration of SEE:* ***3 Hours*** |

**Marks Distribution**

**CIE Marks Distribution**

|  |  |
| --- | --- |
| **Lab CIE** | **Distribution of Marks** |
| Record | 10 |
| Continuous Evaluation | 20 |
| Lab MSE | 20 |
| Total | 50 |

**SEE Marks Distribution**

|  |  |
| --- | --- |
|  | **MARKS DISTRIBUTION** |
| Write Up | 15 |
| Execution | 25 |
| Viva | 10 |
| Total | 50 |

Data Analysis using R Programming

PART- A

1. **Perform the following using R**
2. **Create a vector of multiples of 5 using the sequence function** **starting from 5 to 60 and display its values. Create filtered vector whose values are greater than 15 and less than 45.**
3. **Create a list containing a vector, a matrix, and another list.** **Display the class of each element in the list.** **Count the number of objects in a given list and find the length of the first vector of a given list.**
4. **Create matrices in R and perform Addition, Subtraction of two matrices and product of matrix and its transpose.**
5. **Calculate the Column sum, Mean across rows, Total Sum of a matrix and Sort the matrix elements across columns in ascending order.**

**Solution:**

* 1. **Create a vector of multiples of 5 using the sequence function** **starting from 5 to 60 and display its values. Create filtered vector whose values are greater than 15 and less than 45.**

# Creating a vector with multiples of 5 using seq

vector <- seq(5, 60, by = 5)

# Displaying the values vector

cat("Value of vector:", vector, "\n")

# Logical indexing to find elements greater than 15 and less than 45

filtered\_vector <- vector[vector > 15 & vector < 45]

# Displaying the filtered vector

cat("Elements greater than 15 and less than 45:", filtered\_vector, "\n")

1. **Create a list containing a vector, a matrix, and another list.** **Display the class of each element in the list.** **Count the number of objects in a given list and find the length of the first vector of a given list.**

# Creating a vector

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

b. # Creating a matrix

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

# Creating a nested list

my\_nested\_list <- list(

sub\_vector = c("a", "b", "c"),

sub\_number = 42

)

# Creating the main list and naming the elements

my\_list <- list(

vector = my\_vector,

matrix = my\_matrix,

nested\_list = my\_nested\_list

)

# Displaying the class of each element in the list

for (name in names(my\_list)) {

element\_class <- class(my\_list[[name]])

cat("Class of", name, ":", element\_class, "\n")

}

print("Number of objects in the said list:")

length(my\_list)

print("Length of the vector ‘vector' of the said list")

print(length(my\_list$vector))

**c. Create matrices in R and perform Addition, Subtraction of two matrices and product**

**of matrix and its transpose.**

**#**Create matrices

matrix1 <- matrix(c(2,1,1,1,1,-1,1,1,2), nrow=3,ncol=3)

print(matrix1)

matrix2 <- matrix(c(5, 2, 0, 9, 3, 4,2,3,2), nrow=3,ncol=3)

print(matrix2)

# Add the matrices.

result <- matrix1 + matrix2

cat("Result of addition","\n")

print(result)

# Subtract the matrices

result <- matrix1 - matrix2

cat("Result of subtraction","\n")

print(result)

# Product of matrix and its transpose.

result <- matrix1 %\*% t(matrix1)

cat("Result of product of matrix and its transpose","\n")

print(result)

1. **Calculate the Column sum, Mean across rows, Total Sum of a matrix and Sort the matrix elements across columns in ascending order.**

print("Column sum:")

colSums(matrix1)

print("Mean across rows:")

apply( matrix1, 1, mean)

print("Total sum:")

sum(matrix1)

print("Matrix elements in sorted order column wise:")

apply(matrix1, 2, sort)

1. **Write a R program for the following**
   1. **Check the number is even or odd**
   2. **Print squares of numbers in sequence.**
   3. **Create vector of integers and sort them in ascending/descending order.**

**Solution:**

1. **Check the number is even or odd**

even\_odd <-function(a ) #Function Definition

{

#of lines to be scanned

if(a %% 2 == 0)

{

print("The number is Even Number")

}

else

{

print("The number is Odd Number")

}

}

print("Enter the number to be checked")

a <-scan(nlines=1) #Scan function allows to read and nlines allows number

even\_odd(a) #Function Call

1. **Print squares of numbers in sequence.**

sqr <- function (n)

{

print ("The Square of Numbers is:")

for(i in 0:n)

print(i^2)

}

print("Enter the Range:")

n <- scan(nlines=1)

sqr(n)

1. **Create vector of integers and sort them in ascending/descending order.**

srt<-function(a){

v<-sort(a, decreasing = TRUE)

print("DESCENDING ORDER")

print(v)

x<-sort(a, decreasing = FALSE)

print("ASCENDING ORDER")

print(x)

}

a <-scan(nlines=6) #Scan function allows to read and nlines allows number

srt(a) #Function Call

1. **Create R program and perform the following operations on them.**
   1. **Take user name as input string, display the number characters present in the string, convert the string into uppercase and display the middle character of the string.**
   2. **Create function called is\_palindrome() that determines whether or not a given string is a palindrome. The function should take a single parameter.**

Solution:

1. **Take user name as input string, display the number characters present in the string, convert the string into uppercase and display the middle character of the string.**

midf<-function(str)

{

print(nchar(str)) print(toupper(str)) n1<-nchar(str)+1

mc<-substring(str,n1%/%2,(n1+1)%/%2) print(mc)

}

name<- readline("Enter your name:") midf(name)

1. **Create function called is\_palindrome() that determines whether or not a given string is a palindrome. The function should take a single parameter.**

is\_palindrome <- function(x)

{

a <- substring(x,seq(1,nchar(x)) , seq(1,nchar(x))) paste(rev(a),sep="",collapse="") = = paste(a,sep="",collapse="")

}

str<- readline("Enter string:") print(is\_palindrome(str))

1. **Program for Data Visualization:**

Use packages like ggplot2 or plot to create various types of charts, such as bar charts, line plots, scatter plots, or heatmaps.

* 1. **Read a dataset from a CSV file or other data sources.**
  2. **Customize the charts by adding labels, titles, legends, and adjusting the axis scales.**
  3. **Create interactive visualizations with tooltips, zooming, or filtering options.**
  4. **Export the visualizations to different file formats**

**Solution:**

* 1. **Read a dataset from a CSV file or other data sources.**

# Load necessary packages library(ggplot2) library(dplyr) library(plotly)

# Read Titanic dataset

titanic <- read.csv("C:/Users/ANKITHA/Dropbox/1\_NITTE/Jan 2024/R/Rexam/Daata set/train.csv")

# Convert factor levels

titanic$Survived <- factor(titanic$Survived, levels = c("0", "1"))

* 1. **Customize the charts by adding labels, titles, legends, and adjusting the axis scales.**

# Customise the charts

# Bar chart showing count of survivors by passenger class

bar\_chart <- ggplot(titanic, aes(x = factor(Pclass), fill = factor(Survived))) + geom\_bar(position = "dodge") +

labs(title = "Survivors by Passenger Class", x = "Passenger Class",

y = "Count",

fill = "Survived") +

scale\_fill\_manual(values = c("0" = "red", "1" = "blue")) # Ensure consistent levels with the factor

# Line plot showing age distribution of passengers line\_plot <- ggplot(titanic, aes(x = Age, y = ..density..)) + geom\_density(fill = "blue", alpha = 0.5) +

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

y = "Density")

# Scatter plot showing fare vs age with color indicating survival status scatter\_plot <- ggplot(titanic, aes(x = Age, y = Fare, color = factor(Survived))) + geom\_point() +

labs(title = "Fare vs Age", x = "Age",

y = "Fare",

color = "Survived") +

scale\_color\_manual(values = c("0" = "red", "1" = "blue")) # Ensure consistent levels with the factor

#heatmap

heatmap <- ggplot(titanic, aes(x = Pclass, y = Survived)) + stat\_bin\_2d(bins = 10, aes(fill = ..count..)) +

labs(title = "Titanic Survival Heatmap", x = "Pclass", y = "Survived", fill = "Frequency") + scale\_fill\_continuous(name = "Frequency", low = "white", high = "blue") +

theme\_minimal()

* 1. **Create interactive visualizations with tooltips, zooming, or filtering options.**

plotly # Interactive bar chart

interactive\_bar\_chart <- ggplotly(bar\_chart)

# Interactive line plot

interactive\_line\_plot <- ggplotly(line\_plot)

# Interactive scatter plot

interactive\_scatter\_plot <- ggplotly(scatter\_plot) #interactive heat map

interactive\_heatr\_plot <- ggplotly(heatmap)

* 1. **Export the visualizations to different file formats**

visualizations # Export visualizations to png files

ggsave("bar\_plot.png", plot = bar\_plot, width = 8, height = 6) ggsave("line\_plot.png", plot = line\_plot, width = 8, height = 6) ggsave("scatter\_plot.png", plot = scatter\_plot, width = 8, height = 6) ggsave("titanic\_heatmap\_ggplot.png", heatmap, width = 8, height = 6)

1. **Program for Data Analysis:**
2. **Read a dataset from a CSV file and perform exploratory data analysis, including summary statistics, and identifying missing values.**
3. **Data cleaning by removing duplicates, handling missing values, and transforming variables if necessary.**
4. **Perform data manipulation operations such as filtering and sorting based on certain criteria.**
5. **Generate reports or visualizations to present the analysis results.**

**Solution :**

* 1. **Read a dataset from a CSV file and perform exploratory data analysis, including summary statistics, and identifying missing values.**

library(dplyr)

library(ggplot2)

library(tidyr)

#Read dataset from CSV file

file\_path <- "train.csv"

data <- read.csv(file\_path)

**#**Exploratory Data Analysis (EDA)

# Summary statistics

summary\_stats <- summary(data)

print(summary\_stats)

# Data visualization

# For example, let's create a histogram for age

print(ggplot(data, aes(x = Age)) +

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

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

x = "Age",

y = "Frequency"))

# Identifying missing values

missing\_values <- colSums(is.na(data))

print(missing\_values)

* 1. **Data cleaning by removing duplicates, handling missing values, and transforming variables if necessary.**

# Remove duplicates

data <- distinct(data)

#Removing missing values

data$Age[is.na(data$Age)] <- mean(data$Age, na.rm = TRUE)

missing\_values <- colSums(is.na(data))

print(missing\_values)

* 1. **Perform data manipulation operations such as filtering and sorting based on certain criteria.**

# Filtering: Select passengers with age greater than 18

adult\_passengers <- filter(data, Age > 18)

#print(adult\_passengers)

# Sorting: Sort data by Fare in descending order

sorted\_titanic <- arrange(data, desc(Fare))

#print(sorted\_titanic)

# Merging datasets (if applicable)

# Example: If you have another dataset named "additional\_data"

file\_path <- "Adata.csv"

A <- read.csv(file\_path)

# Check column names in both datasets

print(colnames(data))

print(colnames(A))

# Merge based on 'PassengerId'

merged\_data <- merge(data, A, by.x = "PassengerId", by.y = "PassengerId")

print(merged\_data )

* 1. **Generate reports or visualizations to present the analysis results.**

# Example: Hypothesis testing (t-test)

# Check the assumptions (visualize the distribution of ages for each group)

print(boxplot(Age ~ Survived, data = data, col = c("red", "blue"), main = "Boxplot of Age by Survived"))

# Conduct t-test

t\_test\_result <- t.test(Age ~ Survived, data = data)

# Print the t-test result

print(t\_test\_result)

# Calculate the correlation coefficient between 'Age' and 'Fare'

correlation\_coefficient <- cor(data$Age, data$Fare)

# Print the result

print(correlation\_coefficient)

print(ggplot(data, aes(x = factor(Survived), fill = factor(Survived))) +

geom\_bar() +

labs(title = "Number of Survivors on Titanic",

x = "Survived",

y = "Count") +

scale\_fill\_manual(values = c("red", "green")))

**6. Program for Data Manipulation:**

1. **Read multiple datasets from different files or sources.**
2. **Merge or join the datasets based on common variables or keys.**
3. **Perform aggregation operations, such as calculating sums, means, or counts, by groups or categories.**
4. **Filter the data based on specific conditions or criteria.**
5. **Create new variables or transform existing variables using functions or mathematical operations.**
6. **Read multiple datasets from different files or sources.**

library(dplyr)

# Read datasets from different files or sources

dataset1 <- read.csv("sales.csv")

dataset2 <- read.csv("salary.csv")

1. **Merge or join datasets based on common variables or keys**

merged\_data <- merge(dataset1, dataset2, by = "ID")

1. **Perform aggregation operations, such as calculating sums, means, or counts, by groups or categories.**

aggregated\_data <- merged\_data %>%

group\_by(Gender) %>%

summarise(

total\_salary = sum(Salary),

average\_age = mean(Age),

count = n()

)

1. **Filter the data based on specific conditions or criteria**

filtered\_data <- merged\_data %>%

filter(Age > 25)

1. **Create new variables or transform existing variables using functions or mathematical operations**

transformed\_data <- merged\_data %>%

mutate(

doubled\_salary = Salary \* 2,

seniority = ifelse(Age > 28, "Senior", "Junior")

)

# Print the results

print("Merged Data:")

print(merged\_data)

print("Aggregated Data:")

print(aggregated\_data)

print("Filtered Data:")

print(filtered\_data)

print("Transformed Data:")

print(transformed\_data)

**7. Program for Web Scraping and Data Extraction:**

Use R packages like rvest or httr to scrape data from a specific website or API.

* 1. **Define the target website or API endpoints and specify the data to be extracted.**
  2. **Retrieve the HTML content or JSON response from the website or API.**
  3. **Parse and extract the desired data using CSS selectors, XPath, or JSON parsing techniques.**
  4. **Save the extracted data to a file or perform further analysis on it.**

Solution:

1. **Define the target website or API endpoints and specify the data to be extracted.**

# Load necessary libraries library(rvest)

# Specify the URL of the website to scrape url <- "<http://books.toscrape.com/>"

1. **Retrieve the HTML content or JSON response from the website or API.**

# Download the HTML content html\_content <- read\_html(url)

# Define XPath selectors to extract data title\_xpath <- '//\*[@class="product\_pod"]/h3/a'

price\_xpath <- '//\*[@class="product\_pod"]/div[2]/p[1]'

1. **Parse and extract the desired data using CSS selectors, XPath, or JSON parsing techniques.**

# Extract data using XPath selectors titles <- html\_content %>% html\_nodes(xpath = title\_xpath) %>% html\_text() %>%

trimws() # Remove leading/trailing whitespaces

prices <- html\_content %>% html\_nodes(xpath = price\_xpath) %>% html\_text() %>%

trimws() # Remove leading/trailing whitespaces

1. **Save the extracted data to a file or perform further analysis on it.**

# Combine the extracted data into a data frame book\_data <- data.frame(Title = titles, Price = prices)

# Print the extracted data print(book\_data)

# Save the extracted data to a CSV file

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