RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE

MA23435
Probability, Statistics and Simulation

Laboratory Observation Note Book

Name:	Monish D.Y.
Year / Branch / Section:	II year – CSE – B
Register No:	2116230701195
Semester:	IV
Academic Year:	2024 – 2025

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

RAJALAKSHMI NAGAR, THANDALAM – 602 105

BONAFIDE CERTIFICATE

NAME MONISH D.Y. **REGISTER NO.** 2116230701195

ACADEMIC YEAR 2024-25 **SEMESTER** IV

BRANCH: B.E. CSE B

This Certification is the Bonafide record of work done by the above student in the **MA23435 - Probability, Statistics and Simulation** Laboratory during the year 2024 - 2025.

Signature of Faculty -in – Charge

Submitted for the Practical Examination held on _____

Internal Examiner External Examiner

INDEX

Name: Monish D.Y. **Branch:** CSE **Sec:** B **Roll No:** 230701195

S. No.	Date	Title	Page No.	Teacher's Sign
1	25.01.2025	Basic Functions in R and plotting	1	
2	01.02.2025	Mathematical functions in R – Integration	4	
3	08.02.2025	Control flow – Loops in R	7	
4	22.02.2025	Probability Distributions using R	10	
5	22.03.2025	Testing of Hypothesis – Z testing	13	
6	29.03.2025	Testing of Hypothesis – F and chi square testing	19	
7	05.04.2025	Markov chains analysis – using 'markovchain' package in R	28	
8	07.04.2025	Queuing model analysis – using 'queuing' package in R	38	
9	12.04.2025	Monte Carlo simulation –predicting stock prices using package 'MonteCarlo' in R	44	
10	19.04.2025	Reading, writing data in R and working with inbuilt data sets in R	63	

Ex. No. 1	BASIC FUNCTIONS IN R AND PLOTTING
Date: 25.01.2025	

1. Write an R program to calculate the square root of a number

Aim:

To calculate the square root of a number using sqrt() function.

Procedure:

1. *Use the built-in sqrt() function to compute the square root.*
2. *Pass the value as an argument to the function.*
3. *Print the result using print().*

Program:

```
result <- sqrt(49)
print(result)
```

Output:

```
7
```

Result:

Thus, the program to calculate the square root is executed successfully.

2. Write an R program to calculate the exponential of a number

Aim:

To calculate the exponential of a number using exp() function.

Procedure:

1. *Use the exp() function to compute e raised to the power of the number.*
2. *Print the result using print().*

Program:

```
result <- exp(2)
print(result)
```

Output:

```
7.389056
```

Result:

Thus, the program to calculate the exponential is executed successfully.

3. Write an R program to calculate the factorial of a number

Aim:

To calculate the factorial of a number using factorial() function.

Procedure:

1. *Use the factorial() function to compute the factorial.*
2. *Pass the value as an argument to the function.*
3. *Print the result using print().*

Program:

```
result <- factorial(5)
print(result)
```

Output:

```
120
```

Result:

Thus, the program to calculate the factorial is executed successfully.

4. Write an R program to calculate the ceiling and floor value of a number

Aim:

To calculate the ceiling and floor value of a number using ceiling() and trunc() functions

.

Procedure:

1. *Use the ceiling() and trunc() function to compute the ceiling and floor value.*
2. *Pass the value as an argument to the function.*
3. *Print the result using print().*

Program:

```
result <- ceiling(5.7)
print(result)

result <- trunc(7.9)
print(result)
```

Output:

```
6
7
```

Result:

Thus, the program to calculate the ceiling and floor value is executed successfully.

5. Write an R program to calculate the absolute value of a number

Aim:

To calculate the absolute value of a number using abs() function.

.

Procedure:

1. *Use the abs() function to calculate the absolute value.*
2. *Pass the value as an argument to the function.*
3. *Print the result using print().*

Program:

```
result <- abs(-15)
print(result)
```

Output:

15

Result:

Thus, the program to calculate the absolute value is executed successfully.

6. Write an R program to plot a simple line graph

Aim:

To create a simple line plot for y=x^2 using plot() function.

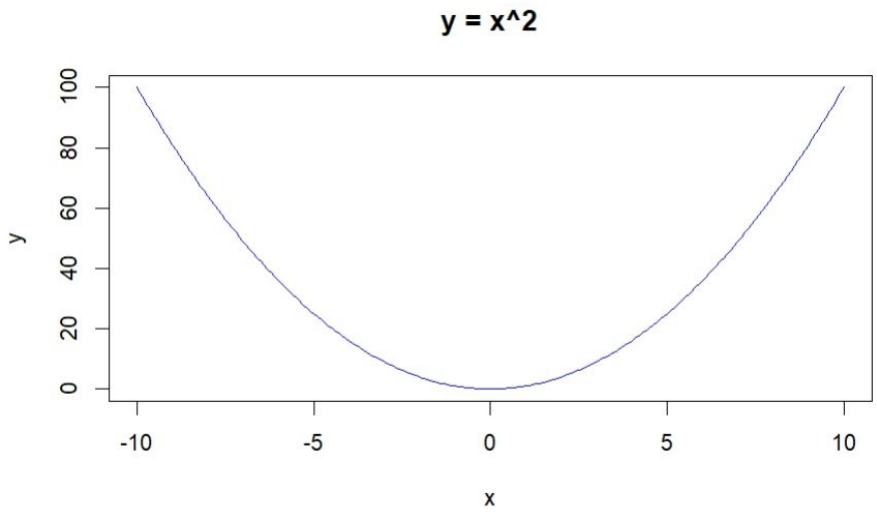
Procedure:

1. *Create a sequence of x values from -10 to 10 using the seq() function.*
2. *Compute the corresponding y values using y = x^2.*
3. *Use the plot() function to create the line plot.*

Program:

```
x <- seq(-10, 10, by = 0.1)
y <- x^2
plot(x, y, type = "l", col = "blue", main = "y = x^2", xlab = "x", ylab = "y")
```

Output:



Result:

Thus, the program to plot a simple line graph is executed successfully.

7. Write an R program to plot a Histogram

Aim:

To create a histogram plot for random numbers using hist() function.

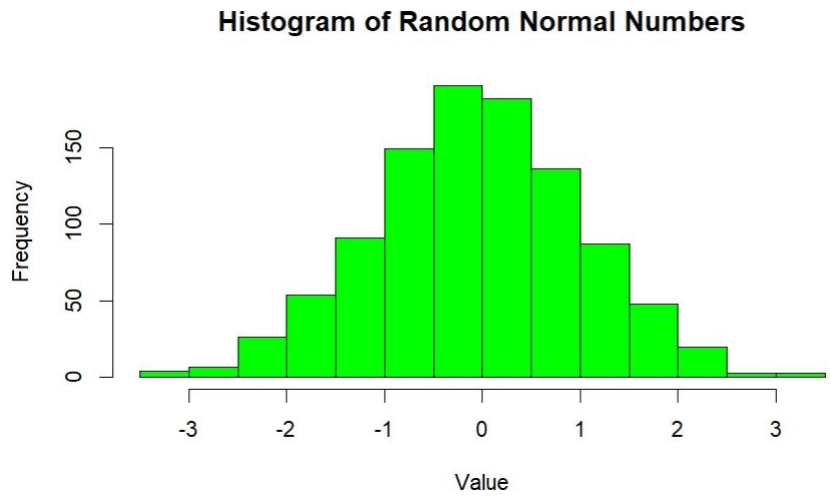
Procedure:

1. *Generate 1000 random numbers from a normal distribution using the rnorm() function.*
2. *Use the hist() function to create a histogram of these values.*

Program:

```
random_numbers <- rnorm(1000)
hist(random_numbers, main = "Histogram of Random Normal Numbers", xlab =
"Value", col = "green")
```

Output:



Result:

Thus, the program to plot a histogram is executed successfully.

8. Write an R program to plot a Pie Chart

Aim:

To create a pie chart of sales data using pie() function.

Procedure:

1. *Create a vector of sales data and corresponding labels.*
2. *Use the pie() function to create a pie chart.*

Program:

```
sales <- c(20, 30, 25, 25)
labels <- c("Q1", "Q2", "Q3", "Q4")
pie(sales, labels = labels, col = rainbow(4), main = "Sales Distribution")
```

Output:

Result:

Thus, the program to plot a pie chart is executed successfully.

Ex. No. 2	INTEGRATION IN R
Date: 01.02.2025	

1. Write an R program to integrate a polynomial function.

Aim:

To integrate the polynomial function f(x) = x^2 from 0 to 6.

Procedure:

1. *Define the function f(x)= x^2 in R.*
2. *Use the integrate() function to compute the definite integral from 0 to 6 and print the result.*

Program:

```
f <- function(x) x^2
result <- integrate(f, lower = 0 , upper = 6)
print(result$value)
```

Output:

72

Result:

Thus, the program to integrate a polynomial function is executed successfully.

2. Write an R program to integrate a trigonometric function

Aim:

To evaluate the definite integral of the function f(x)=sin(x) over the interval [0,π] using the integrate() function in R.

Procedure:

1. *Define the function f(x)=sin(x).*
2. *Use integrate(f, lower = 0, upper = pi) to compute the integral.*
3. *Extract and print the result using result$value.*

Program:

```
f <- function(x) sin(x)
result <- integrate(f, lower =0 ,upper = pi)
print(result$value)
```

Output:

2

Result:

Thus, the program to integrate a trigonometric function is executed successfully.

3. Write an R Program to visualize the integration result.

Aim:

To visualize the area under the curve of f(x)=sin(x) over the interval [0,π] using ggplot2 in R.

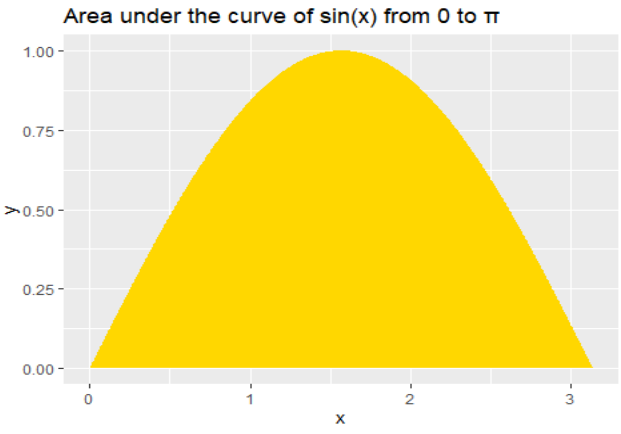
Procedure:

1. *Define f(x)=sin(x) and generate x_values from 0 to π.*
2. *Compute y_values and store them in a dataframe.*
3. *Use ggplot() with geom_area() to plot and fill the area under the curve.*

Program:

```
library(ggplot2)

f <- function(x) { sin(x) }
x_values <- seq(0, pi, by = 0.01)
y_values <- f(x_values)

data <- data.frame(x = x_values, y = y_values)
ggplot(data, aes(x = x, y = y)) +  geom_area(fill = "gold") +  ggtitle("Area under the curve of sin(x) from 0 to π")
```

Output:



Result:

Thus, the program to visualize the integration result of a function is executed successfully.

4. Write an R program to perform a double integration.

Aim:

To evaluate the double integral of f(x,y)=x+y over the region 0≤x≤1,0≤y≤10 10≤x≤1,0≤y≤1 using the dblquad() function from the pracma package in R.

.

Procedure:

1. *Define the function f(x,y)=x+yf(x, y) = x + yf(x,y)=x+y.*
2. *Use dblquad(f, 0, 1, 0, 1) to compute the double integral.*
3. *Print the result using print(result).*

Program:

```
library(pracma)
f <- function(x,y) x+y
result <- dblquad (f ,0,1,0,1)
print(result)
```

Output:

1

Result:

Thus, the program to evaluate a double integration is executed successfully.

Ex. No. 3	CONTROL FLOW – LOOPS IN R
Date: 08.02.2025	

1. Write an R program to find the factorial of a number using the 'while loop'.

Aim:

To calculate the factorial of a number using the while loop.

Procedure:

1. *Set num to the desired value and initialize factorial to 1.*
2. *Use a while loop to multiply factorial by num and decrement num until it reaches 0.*
3. *Print the final value of factorial.*

Program:

```
num <- 6
factorial <- 1
while (num > 0) {
 factorial <- factorial * num
 num <- num - 1
}
print(factorial)
```

Output:

720

Result:

Thus, the program to calculate the factorial of a number using the while loop is executed successfully.

2. Write an R program to print prime numbers from 1 to 20

Aim:

To print all the prime numbers in the range 1 to 20 using the for loops and functions.

Procedure:

1. *Define the function is_prime(n), returning FALSE if n < 2.*
2. *Check divisibility of n from 2 to sqrt(n), returning FALSE if divisible.*
3. *Use a for loop to iterate through numbers from 1 to 20, calling is_prime(i).*
4. *Print numbers for which is_prime(i) returns TRUE.*

Program:

```
is_prime <- function(n) {
    if (n < 2) return(FALSE)
    for (i in 2:sqrt(n)) {
        if (n %% i == 0) return(FALSE)
    }
    return(TRUE)
}

for (i in 1:20) {
    if (is_prime(i)) print(i)
}
```

Output:

```
2
3
5
7
11
13
17
19
```

Result:

Thus, the program to print prime numbers from 1 to 20 using the for loops is executed successfully.

3. Write an R program to print the countdown using the repeat loop

Aim:

To print the countdown using repeat loop.

Procedure:

1. *Initialize num with the value 7.*
2. *Use a repeat loop to print num and decrement it by 1.*
3. *Break the loop when num becomes 0.*

Program:

```
num <- 7
repeat {
  print(num)
  num <- num - 1
  if (num == 0) break
}
```

Output:

```
7
6
5
4
3
2
1
```

Result:

Thus, the program to print the countdown using the repeat loop is executed successfully.

4. Write an R program to print the odd numbers from 1 to 10 using 'next'

Aim:

To print the odd numbers from 1 to 10 by skipping the iterations using the next keyword.

.

Procedure:

1. *Use a for loop to iterate through numbers from 1 to 10.*
2. *Check if i is even using i %% 2 == 0; if true, skip the iteration using next.*
3. *Print the value of i if it is odd.*

Program:

```
for (i in 1:10) {
  if (i %% 2 == 0) next
  print(i)
}
```

Output:

```
1
3
5
7
9
```

Result:

Thus, the program to print the odd numbers from 1 to 10 by skipping the iterations using the next keyword is executed successfully.

5. Write an R program to break out of a loop using the 'break' keyword

Aim:

To write a program that uses the 'break' keyword to break out of a loop.

.

Procedure:

1. *Use a for loop to iterate through numbers from 1 to 10.*
2. *If i equals 6, exit the loop using break.*
3. *Print the value of i for all numbers before 6.*

Program:

```
for (i in 1:10) {
  if (i == 6) break
  print(i)
}
```

Output:

```
1
2
3
4
5
```

Result:

Thus, the program to use the break statement to break out of the loop is executed successfully.

6. Write an R program to use lapply() to find the square of a number

Aim:

To use lapply() function to find the square of a number.

Procedure:

1. *Create a list my_list containing numbers 1 to 5.*
2. *Use lapply() to apply a function that squares each element of the list.*
3. *Print the resulting list of squared values.*

Program:

```
my_list <- list(1, 2, 3, 4, 5)
squared <- lapply(my_list, function(x) x^2)
print(squared)
```

Output:

```
[[1]]
1

[[2]]
4

[[3]]
9

[[4]]
16

[[5]]
25
```

Result:

Thus, the program to use lapply() to find the square of a number is executed successfully.

Ex. No. 4	PROBABILITY DISTRIBUTIONS USING R
Date: 22.02.2025	

1. Calculate the probability of getting exactly 3 successes in a Binomial distribution with n = 8 and p = 0.6.

Aim:

To calculate the probability of getting exactly 3 successes in a Binomial distribution with n = 8 and p = 0.6.

Procedure:

1. *Set n = 8 for the number of trials and p = 0.6 for the probability of success.*
2. *Use dbinom(3, size = n, prob = p) to compute the probability of exactly 3 successes.*
3. *Store the result in prob and display the calculated probability.*

Program:

```
n <- 8
p <- 0.6
prob <- dbinom(3, size = n, prob = p)
prob
```

Output:

```
0.123863
```

Result:

Thus, the program to calculate the probability of getting exactly 3 successes in a Binomial distribution is executed successfully.

2. Calculate and plot the CDF for the Poisson distribution with λ = 5.

Aim:

To calculate and plot the CDF for the Poisson distribution with λ = 5.

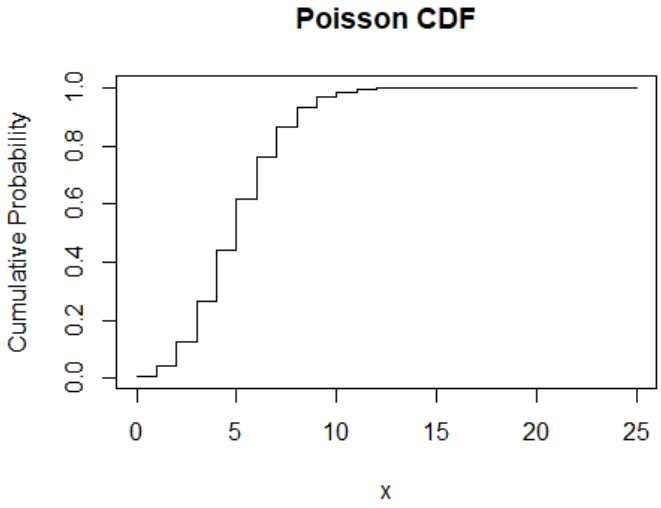
Procedure:

1. *Set lambda <- 5 and define x <- 0:25.*
2. *Compute cumulative probabilities using ppois(x, lambda).*
3. *Plot the Poisson CDF using plot() with type = "s".*

Program:

```
lambda <- 5
x <- 0:25
cdf <- ppois(x, lambda)
plot(x, cdf, type = "s", main = "Poisson CDF", xlab = "x", ylab = "Cumulative
Probability")
```

Output:



Result:

Thus, the program to calculate and plot the CDF for the Poisson distribution is executed successfully.

3. Plot the PDF of an Exponential distribution with rate = 1.5.

Aim:

To plot the PDF of an Exponential distribution with rate = 1.5.

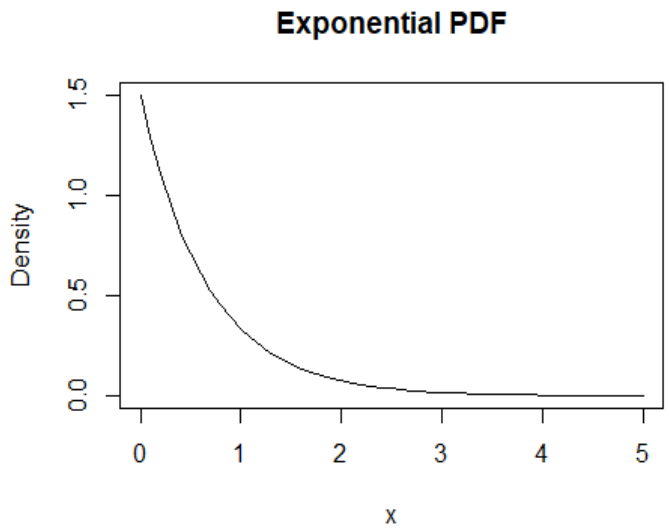
Procedure:

1. *Set rate <- 1.5 and generate x values from 0 to 5 in steps of 0.1 using seq().*
2. *Compute the probability density function using dexp(x, rate).*
3. *Plot the exponential PDF using plot() with type = "l".*

Program:

```
rate <- 1.5
x <- seq(0, 5, by = 0.1)
pdf <- dexp(x, rate)
plot(x, pdf, type = "l", main = "Exponential PDF", xlab = "x", ylab = "Density")
```

Output:



Result:

Thus, the program to plot the PDF of an Exponential distribution is executed successfully.

4. Simulate and visualize 1000 random numbers from a Normal distribution with mean = 50 and sd = 10.

Aim:

To simulate and visualize 1000 random numbers from a Normal distribution with mean = 50 and sd = 10 using rnorm().

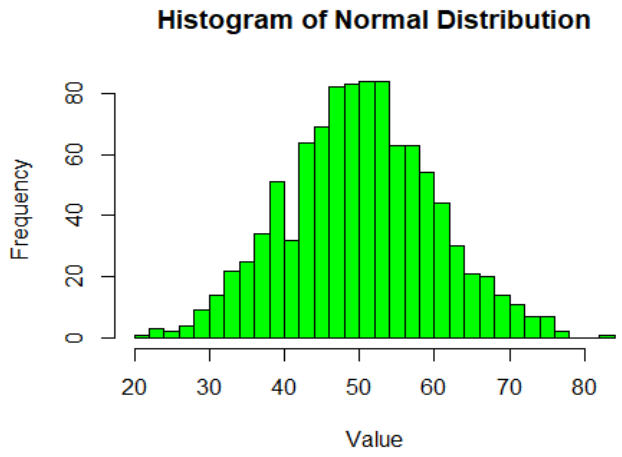
Procedure:

1. *Set the random seed using set.seed(123) for reproducibility.*
2. *Generate 1000 random values from a normal distribution with mean 50 and standard deviation 10 using rnorm().*
3. *Plot a histogram with 30 bins, green color, and labeled axes using hist().*

Program:

```
set.seed(123)
data <- rnorm(1000, mean = 50, sd = 10)
hist(data, main = "Histogram of Normal Distribution", xlab = "Value", ylab =
"Frequency", col = "green", breaks = 30)
```

Output:



Result:

Thus, the program to simulate and visualize 1000 random numbers from a Normal distribution using rnorm() is executed successfully.

5. Generate and plot the CDF of a Uniform distribution on [-2, 2]

Aim:

To generate and plot the CDF of a Uniform distribution on [-2, 2].

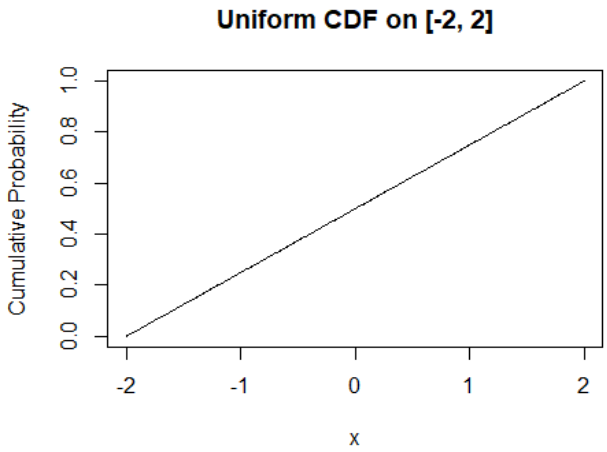
Procedure:

1. *Set a <- -2 and b <- 2 as the range of the uniform distribution.*
2. *Generate x values from a to b in steps of 0.01 using seq().*
3. *Compute cumulative probabilities using punif(x, min = a, max = b), and plot the CDF using plot() with type = "l".*

Program:

```
a <- -2
b <- 2
x <- seq(a, b, by = 0.01)
cdf <- punif(x, min = a, max = b)
plot(x, cdf, type = "l", main = "Uniform CDF on [-2, 2]", xlab = "x", ylab =
"Cumulative Probability")
```

Output:



Result:

Thus, the program to generate and plot the CDF of a uniform distribution on the limit [-2, 2] is executed successfully.

Ex. No. 5	TESTING OF HYPOTHESIS – Z TEST
Date: 22.03.2025	

1. Z-Test for Mean (Two-Tailed Test)

A sample of 100 students has an average score of 75. The population mean score is 72, and the population standard deviation is 10. Test if the sample mean is different from the population mean at the 5% significance level.

Aim:

To test whether the sample mean score is significantly different from the population mean score at a 5% significance level using a one-sample Z-test.

Procedure:

1. *Calculate Z-value using the formula:*
   Z = (sample_mean - population_mean) / (sigma/sqrt (n)).

2. *Find the critical value using qnorm (1 - alpha / 2).*
3. *Compare the Z-value with the critical value. Reject H0 if the absolute Z-value is greater than the critical value, otherwise, fail to reject H0.*

Program:

```
sample_mean <- 75
population_mean <- 72
sigma <- 10
n <- 100
alpha <- 0.05

z_stat <- (sample_mean - population_mean) / (sigma / sqrt(n))
z_critical <- qnorm (1 - alpha / 2)

z_stat
z_critical

if(abs(z_stat) > z_critical) {
  print ("Reject the null hypothesis")
} else {
  print ("Fail to reject the null hypothesis")
}
```

Output:

3
1.959964
"Reject the null hypothesis"

Result:

Thus, the one-sample Z-test was successfully implemented, determining whether the sample mean significantly differs from the population mean at a 5% significance level.

2. Z-Test for Mean (Right-Tailed Test)

A factory claims the average weight of a product is 500 grams. A sample of 30 products has a mean weight of 505 grams, and the population standard deviation is 15 grams. Test if the sample weight is significantly greater than 500 grams at α = 0.01.

Aim:

To test whether the average weight of the product is significantly greater than 500 grams at a 1% significance level using a one-sample Z-test.

Procedure:

1. *Calculate the Z-value using the formula:*
   Z= (sample_mean - population_mean) / (sigma/sqrt (n)).

2. *Find the critical value using the qnorm (1 - alpha) function.*
3. *Compare the Z-value with the critical value. Reject H0 if Z is greater than the critical value; otherwise, fail to reject H0.*

Program:

```
sample_mean <- 505
population_mean <- 500
sigma <- 15
n <- 30
alpha <- 0.01

z_stat <- (sample_mean - population_mean) / (sigma / sqrt(n))
z_critical <- qnorm (1 - alpha)

z_stat
z_critical

if (z_stat > z_critical) {
  print ("Reject the null hypothesis")
} else {
  print ("Fail to reject the null hypothesis")
}
```

Output:

1.825742
2.326348
"Fail to reject the null hypothesis"

Result:

Thus, the one-sample Z-test was successfully conducted to determine whether the average product weight is significantly greater than 500 grams at a 1% significance level.

3. Z-Test for Mean (Left-Tailed Test)

A machine is designed to fill bottles with exactly 1 liter of liquid. A quality control inspector believes the machine is underfilling the bottles. A sample of 40 bottles shows an average fill of 0.98 liters, with a population standard deviation of 0.05 liters. Test at a 1% significance level whether the machine is underfilling the bottles.

Aim:

To test whether the machine is underfilling the bottles at a 1% significance level using a left-tailed one-sample Z-test.

Procedure:

1. *Calculate the Z-value using the formula:*
   Z = (sample_mean - population_mean) / (sigma / sqrt(n)).

2. *Find the critical value using the qnorm (0.01) function.*
3. *Compare the Z-value with the critical value. Reject H_0 if Z is less than the critical value; otherwise, fail to reject H_0.*

Program:

```
sample_mean <- 0.98
population_mean <- 1
sigma <- 0.05
n <- 40
alpha <- 0.01

z_stat <- (sample_mean - population_mean) / (sigma / sqrt(n))
z_critical <- qnorm(alpha)

z_stat
z_critical

if (z_stat < z_critical) {
  print ("Reject the null hypothesis")
} else {
  print ("Fail to reject the null hypothesis")
}
```

Output:

-2.529822
-2.326348
"Reject the null hypothesis"

Result:

Thus, the one-sample Z-test was successfully conducted to determine whether the machine was underfilling the bottles at a 1% significance level.

Ex. No. 6	TESTING OF HYPOTHESIS - F-TEST AND CHI-SQUARE TEST
Date: 29.03.2025	

1. F-Test for Equal Variances (Two-Sample)

A study tests whether the variances of two manufacturing processes are the same. Sample 1 has 25 observations with variance $s1^2 = 100$, and sample 2 has 30 observations with variance $s2^2 = 80$. Test at $\alpha = 0.05$.

Aim:

To calculate the F-statistic for comparing two population variances with n1 = 25, n2 = 30, var1 = 100, and var2 = 80 at α = 0.05 significance level.

Procedure:

1. *Set n1 = 25, n2 = 30, var1 = 100, var2 = 80, and α = 0.05*
2. *Calculate f_crit = var1/var2 and find f_table using qf(1-α/2, n1-1, n2-1)*
3. *Compare f_crit with f_table to accept or reject the null hypothesis*

Program:

```
n1 <- 25
n2 <- 30
var1 <- 100
var2 <- 80
alpha <- 0.05
f_crit <- var1/var2
f_table <- qf(1-alpha/2, n1-1, n2-1)
print(paste("F-statistic:", f_crit))
print(paste("Critical F-value:", f_table))
if(f_crit < f_table) {
  print("Accept null hypothesis (Equal variances)")
} else {
  print("Reject null hypothesis (Unequal variances)")
}
```

Output:

```
"F-statistic: 1.25"
"Critical F-value: 2.154006"
"Accept null hypothesis (Equal variances)"
```

Result:

Thus, the program to perform F-test for equal variances is executed successfully, and the null hypothesis of equal variances is accepted.

2. F-Test for Variance Comparison (Right-Tailed Test)

A company compares the variances in processing times between two production lines. Line 1 has 40 samples with variance $s1^2 = 120$, and Line 2 has 35 samples with variance $s2^2 = 100$. Test if Line 1 has a significantly greater variance than Line 2 at $\alpha = 0.01$.

Aim:

To calculate the F-statistic for testing if the first variance is greater than the second with n1 = 40, n2 = 35, var1 = 120, and var2 = 100 at α = 0.01 significance level.

Procedure:

1. *Set n1 = 40, n2 = 35, var1 = 120, var2 = 100, and α = 0.01*
2. *Calculate f_crit = var1/var2 and find f_table using qf(1-α, n1-1, n2-1)*
3. *Compare f_crit with f_table to accept or reject the null hypothesis*

Program:

```
n1 <- 40
n2 <- 35
var1 <- 120
var2 <- 100
alpha <- 0.01
f_crit <- var1/var2
f_table <- qf(1-alpha, n1-1, n2-1)
print(paste("F-statistic:", f_crit))
print(paste("Critical F-value:", f_table))
if(f_crit < f_table) {
  print("Accept null hypothesis (σ1^2 ≤ σ2^2)")
} else {
  print("Reject null hypothesis (σ1^2 > σ2^2)")
}
```

Output:.

```
"F-statistic: 1.2"
"Critical F-value: 2.218146"
"Accept null hypothesis (σ1^2 ≤ σ2^2)"
```

Result:

Thus, the program to perform right-tailed F-test is executed successfully, and the null hypothesis is accepted.

3. F-Test for Variances Comparison (Left-Tailed Test)

A chemist compares the variability in pH levels between two chemical solutions. Solution 1 has 50 samples with variance $s1^2 = 0.25$, and Solution 2 has 60 samples with variance $s2^2 = 0.20$. Test if the variance of Solution 1 is smaller than Solution 2 at $\alpha = 0.05$.

Aim:

To calculate the F-statistic for testing if the first variance is smaller than the second with n1 = 50, n2 = 60, var1 = 0.25, and var2 = 0.2 at α = 0.05 significance level.

Procedure:

1. *Set n1 = 50, n2 = 60, var1 = 0.25, var2 = 0.2, and α = 0.05*
2. *Calculate f_crit = var1/var2 and find f_table using qf(α, n1-1, n2-1)*
3. *Compare f_crit with f_table to accept or reject the null hypothesis*

Program:

```
n1 <- 50
n2 <- 60
var1 <- 0.25
var2 <- 0.2
alpha <- 0.05
f_crit <- var1/var2
f_table <- qf(alpha, n1-1, n2-1)
print(paste("F-statistic:", f_crit))
print(paste("Critical F-value:", f_table))
if(f_crit > f_table) {
  print("Accept null hypothesis (σ1^2 ≥ σ2^2)")
} else {
  print("Reject null hypothesis (σ1^2 < σ2^2)")
}
```

Output:

```
"F-statistic: 1.25"
"Critical F-value: 0.6318137"
"Accept null hypothesis (σ1^2 ≥ σ2^2)"
```

Result:

Thus, the program to perform left-tailed F-test is executed successfully, and the null hypothesis is accepted.

4. Chi-Square Goodness of Fit Test

A die is rolled 60 times, and the number of times each number appeared is recorded as follows: 1: 8, 2: 12, 3: 10, 4: 11, 5: 9, 6: 10.

Test if the die is fair at α = 0.05.

Aim:

To calculate the chi-square statistic for testing if observed frequencies [8, 12, 10, 11, 9, 10] follow a uniform distribution at α = 0.05 significance level.

Procedure:

1. *Set observed values to c(8,12,10,11,9,10)*
2. *Use chisq.test with p=rep(1/6,6) to compute test statistic*
3. *Compare p-value with α = 0.05 to accept or reject null hypothesis*

Program:

```
obs_val <- c(8,12,10,11,9,10)
chi_test <- chisq.test(obs_val, p=rep(1/6,6))
print(chi_test)
if(chi_test$p.value < 0.05) {
  print("Reject null hypothesis (Not uniform)")
} else {
  print("Accept null hypothesis (Uniform)")
}
```

Output:

```
Chi-squared test for given probabilities
X-squared = 1, df = 5, p-value = 0.9626
"Accept null hypothesis (Uniform)"
```

Result:

Thus, the program to perform chi-square goodness-of-fit test is executed successfully, and the null hypothesis is accepted.

5. Chi-Square Goodness of Fit Test (Uneven Distribution)

A survey of 100 people shows the following preference for 4 colors: Red: 35, Blue: 25, Green: 20, Yellow: 20.

Test if the preferences are evenly distributed at α = 0.05.

Aim:

To calculate the chi-square statistic for testing independence between categorical variables in a 4×3 contingency table at α = 0.05 significance level.

Procedure:

1. *Create contingency table matrix*
2. *Use chisq.test to compute test statistic*
3. *Compare p-value with α = 0.05 to accept or reject null hypothesis*

Program:

```
obs_val <- matrix(c(10,20,30,15,25,40,20,30,50,25,35,45), nrow=4, byrow=TRUE)
chi_test <- chisq.test(obs_val)
print(chi_test)
if(chi_test$p.value < 0.05) {
  print("Reject null hypothesis (Dependent)")
} else {
  print("Accept null hypothesis (Independent)")
}
```

Output:

```
Pearson's Chi-squared test
X-squared = 2.1362, df = 6, p-value = 0.9068
"Accept null hypothesis (Independent)"
```

Result:

Thus, the program to perform chi-square test of independence is executed successfully, and the null hypothesis is accepted.

6. Chi-Square Goodness of Fit Test with Custom Expected Values

A bag contains colored marbles: Red: 40, Blue: 35, Green: 25.
You expect the proportions to be 1:2:3 (Red: Blue: Green).

Test if the distribution matches the expected proportions at α = 0.05.

Aim:

To calculate the chi-square statistic for testing if observed frequencies [40, 35, 25] match expected proportions [1/6, 2/6, 3/6] at α = 0.05 significance level.

Procedure:

1. *Set observed values to c(40,35,25) and expected proportions to c(1/6,2/6,3/6)*
2. *Use chisq.test with p=ex_prop to compute test statistic*
3. *Compare p-value with α = 0.05 to accept or reject null hypothesis*

Program:

```
obs_val <- c(40,35,25)
ex_prop <- c(1,2,3)/6
chi_test <- chisq.test(obs_val, p=ex_prop)
print(chi_test)
if(chi_test$p.value < 0.05) {
  print("Reject null hypothesis (Mismatch)")
} else {
  print("Accept null hypothesis (Match)")
}
```

Output:

```
Chi-squared test for given probabilities
X-squared = 45.25, df = 2, p-value = 1.493e-10
"Reject null hypothesis (Mismatch)"
```

Result:

Thus, the program to perform chi-square goodness-of-fit test with custom expected values is executed successfully, and the null hypothesis is rejected.

Ex. No. 7	MARKOV CHAINS ANALYSIS USING MARKOVCHAIN PACKAGE USING R
Date: 05.04.2025	

1. **A smart bulb operates in three modes: On (actively glowing), Off (powered down), and Idle (standby mode). The probability of transitioning between these modes every hour is as follows:**

From\To	On	Off	Idle
On	0.60	0.30	0.10
Off	0.10	0.80	0.10
Idle	0.30	0.30	0.40

Using R,

- **i) Model the bulb's behavior as a Markov chain.**
- **ii) Determine the steady-state probabilities of each mode.**
- **iii) Simulate the bulb's state transitions for 20 hours starting from the "On" state.**
- **iv) Check if the Markov chain is ergodic.**

Aim:

To create and analyse a Markov chain for a system with three states ("On", "Off", "Idle") using given transition probabilities..

Procedure:

1. *Define the states and transition probability matrix.*
2. *Create a Markov chain object using the markovchain package.*
3. *Display the transition matrix, compute steady states, simulate state transitions, and check ergodicity.*

Program:

```
states <- c("On", "Off", "Idle")
transition_matrix <- matrix(
  c(0.6, 0.3, 0.1,
    0.1, 0.8, 0.1,
    0.3, 0.3, 0.4),
  nrow = 3, byrow = TRUE
)

mc <- new("markovchain", states = states, transitionMatrix = transition_matrix)
print(mc)

steady_state <- steadyStates(mc)
print(steady_state)

set.seed(456)
sim <- rmarkovchain(n = 20, object = mc, t0 = "On")
print(sim)

is.ergodic <- function(mc) {
  return(is.irreducible(mc) && all(period(mc) == 1))
}
if (is.ergodic(mc)) {
  cat("\nThe Markov chain is ergodic.\n")
} else {
  cat("\nThe Markov chain is not ergodic.\n")
}
```

Output:

```
         On      Off      Idle
On       0.6     0.3      0.1
Off      0.1     0.8      0.1
Idle     0.3     0.3      0.4

        On       Off    Idle
[1,] 0.2571     0.6    0.1429

  [1] "On"  "On"  "Off" "On"  "Off" "Off" "Off" "Off" "Off" "Off"
 [11] "Off" "Off" "Off" "On"  "On"  "Off" "On"  "On"  "Off" "Off"

The Markov chain is ergodic.
```

Result:

Thus, the program to model and analyse light bulb transitions using a Markov chain was executed successfully.

2. A region's weather transitions daily between three states: Sunny, Cloudy, and Rainy. The daily weather pattern follows these probabilities:

From Sunny: 70% chance of remaining Sunny, 10% turning Rainy
From Cloudy: 30% chance of becoming Sunny, 50% remaining Cloudy
From Rainy: 30% becoming Cloudy, 50% remaining Rainy

Using R,

- **i) Model the weather system as a Markov chain.**
- **ii) Determine the steady-state probability of each weather change.**
- **iii) Simulate the weather for 30 days starting from a Sunny day.**
- **iv) Check if the Markov chain is ergodic.**

Aim:

To model and analyse weather pattern transitions using a Markov chain based on assumed probabilities.

Procedure:

1. *Define the weather states and the transition probability matrix.*
2. *Create a Markov chain object using the markovchain package.*
3. *Display the transition matrix, calculate steady states, simulate transitions, and check for ergodicity.*

Program:

```
states <- c("Sunny", "Cloudy", "Rainy")
transition_matrix <- matrix(
  c(0.7, 0.2, 0.1,
    0.3, 0.5, 0.2,
    0.2, 0.3, 0.5),
  nrow = 3, byrow = TRUE
)

mc_weather <- new("markovchain", states = states, transitionMatrix =
transition_matrix)

print(mc_weather)
```

```
steady_state_weather <- steadyStates(mc_weather)
print(steady_state_weather)

set.seed(789)
sim_weather <- rmarkovchain(n = 30, object = mc_weather, t0 = "Sunny")
cat("\nWeather Simulation for 30 days:\n\n")
print(sim_weather)

is.ergodic <- function(mc) {
  return(is.irreducible(mc) && all(period(mc) == 1))
}
if (is.ergodic(mc_weather)) {
  cat("\nThe weather Markov chain is ergodic.\n")
} else {
  cat("\nThe weather Markov chain is not ergodic.\n")
```

Output:.

```
        Sunny  Cloudy  Rainy
Sunny     0.7     0.2    0.1
Cloudy    0.3     0.5    0.2
Rainy     0.2     0.3    0.5

      Sunny  Cloudy  Rainy [1,]
0.4634 0.3171 0.2195

 [1] "Sunny" "Sunny" "Sunny" "Sunny" "Sunny" "Sunny" "Sunny" "Sunny"
 [9] "Sunny" "Sunny" "Sunny" "Sunny" "Sunny" "Sunny" "Sunny" "Sunny"
[17] "Sunny" "Sunny" "Cloudy" "Cloudy" "Cloudy" "Cloudy" "Cloudy" "Cloudy"
[25] "Cloudy" "Sunny" "Sunny" "Sunny" "Cloudy" "Sunny" The

weather Markov chain is ergodic.
```

Result:

Thus, the program to model and analyse weather pattern transitions using a Markov chain was executed successfully.

3. A small website has three pages: A, B, and C. The transition matrix for users moving between pages is:

From\To	A	B	C
A	0.1	0.6	0.3
B	0.3	0.4	0.3
C	0.4	0.2	0.4

i) Define the Markov Chain.
ii) Compute steady-state probabilities (PageRank).
iii) Simulate user navigation for 20 steps starting at page A.
iv) Check if the Markov Chain is ergodic or not.

Aim:

To model user navigation behavior on a website with three pages using a Markov Chain, compute the steady-state probabilities (PageRank), and simulate user transitions over 20 steps from page A.

Procedure:

1. *Create a transition matrix and build the Markov Chain using the markovchain package.*
2. *Compute steady-state probabilities (PageRank) using steadyStates().*
3. *Simulate 20-step user navigation from page A and print results.*

Program:

```
states <- c("A", "B", "C")
trans_matrix <- matrix(
  c(0.1, 0.6, 0.3,
    0.3, 0.4, 0.3,
    0.4, 0.2, 0.4),
  nrow = 3, byrow = TRUE, dimnames = list(states, states)
)

mc <- new("markovchain", transitionMatrix = trans_matrix)

cat("PageRank Transition Matrix:\n"); print(mc)
cat("\nPageRank Steady-State Probabilities:\n"); print(steadyStates(mc))

set.seed(123)
cat("\nSimulated User Navigation (20 Steps):\n")
print(rmarkovchain(n = 20, object = mc, t0 = "A"))

if (is.irreducible(mc) && all(period(mc) == 1)) {
  cat("\nThe PageRank Markov chain is ergodic.\n")
} else {
  cat("\nThe PageRank Markov chain is not ergodic.\n")
}
```

Output:

```
PageRank Transition Matrix:
   A   B   C
A 0.1 0.6 0.3
B 0.3 0.4 0.3
C 0.4 0.2 0.4


PageRank Steady-State Probabilities:
          A         B         C
[1,] 0.2777778 0.3888889 0.3333333

Simulated User Navigation (20 Steps):
 [1] "B" "C" "C" "B" "C" "A" "B" "C" "C" "C" "B" "A" "C" "C" "A" "C" "A" "B" "B" "C"

The PageRank Markov chain is ergodic.
```

Result:

Thus, the program to model and analyse PageRank transitions using a Markov chain was executed successfully.

4. A stock market can be in three states: Bull, Bear, or Stagnant. The transition probabilities are:

From Bull: 20% Bear, 10% Stagnant.
From Bear: 40% Bull, 20% Stagnant.
From Stagnant: 30% Bull, 30% Bear.

i) Define the Markov Chain.
ii) Simulate the stock market over 12 months, starting in a Bull market.
iii) Find the probability of being in a Bear market after 6 steps.
iv) Check if the Markov chain is ergodic or not.

Aim:

To model and analyse stock market transitions using a Markov chain based on assumed probabilities.

Procedure:

1. *Create the transition matrix and Markov Chain.*
2. *Find steady-state probabilities and run simulations.*
3. *Check for ergodicity and calculate Bear state probability after 6 steps.*

Program:

```
states <- c("Bull", "Bear", "Stagnant")
trans_matrix <- matrix(c(0.7, 0.2, 0.1,
                         0.4, 0.4, 0.2,
                         0.3, 0.3, 0.4),
                       nrow = 3, byrow = TRUE)

mc <- new("markovchain", states = states, transitionMatrix = trans_matrix)

cat("Transition Matrix:\n")
print(mc)

cat("\nSteady-State Probabilities:\n")
print(steadyStates(mc))

set.seed(123)
cat("\nSimulation (12 Months from Bull):\n")
print(rmarkovchain(n = 12, object = mc, t0 = "Bull"))
```

```
sim_6 <- rmarkovchain(n = 6, object = mc, t0 = "Bull")
cat("\nState after 6 Steps:\n")
print(sim_6)

prob_bear <- mean(sim_6 == "Bear")
cat("\nProbability of Bear after 6 Steps:", prob_bear, "\n")

if (is.irreducible(mc) && all(period(mc) == 1)) {
  cat("The Markov chain is ergodic.\n")
} else {
  cat("The Markov chain is not ergodic.\n")
}
```

Output:.

```
Transition Matrix:
        Bull Bear Stagnant
Bull      0.7  0.2      0.1
Bear      0.4  0.4      0.2
Stagnant  0.3  0.3      0.4


Steady-State Probabilities:
         Bull      Bear  Stagnant
[1,] 0.5454545 0.2727273 0.1818182

Simulation (12 Months from Bull):
 [1] "Bull"    "Bear"    "Bear"    "Stagnant" "Bear"    "Bull"    "Bull"    "Bear"
 [9] "Bear"    "Bear"    "Stagnant" "Bull"

State after 6 Steps:
[1] "Bull" "Bull" "Bull" "Bear" "Bull" "Bull"

Probability of Bear after 6 Steps: 0.1666667
The Markov chain is ergodic.
```

Result:

Thus, the program to model and analyse stock market pattern transitions using a Markov chain was executed successfully.

Ex. No. 8	QUEUING MODEL ANALYSIS – USING 'QUEUEING' PACKAGE IN R
Date: 07.04.2025	

Single-Server Queue (M/M/1)

1. Model an M/M/1 system with Arrival rate (λ=4) and Service rate (μ=6).

Aim:

To model an M/M/1 queuing system with an arrival rate (λ) of 4 and a service rate (μ) of 6, and analyze system performance characteristics such as server utilization and average number of customers in the system.

Procedure:

1. *Define the input parameters using NewInput.MM1.*
2. *Compute the model using QueueingModel.*
3. *Extract key performance measures.*

Program:

```
library(queueing)
mm1_input <- NewInput.MM1(lambda = 4, mu = 6)
mm1_model <- QueueingModel(mm1_input)
summary(mm1_model)
```

Output:

lambda	mu	c	k	m	RO	P0	L_q	W_q	X	L	W	W_{qq}	L_{qq}
4	6	1	NA	NA	0.667	0.333	1.333	0.333	4	2	0.5	0.5	3

Result:

The M/M/1 queuing model was successfully established with λ = 4 and μ = 6. The system was found to be stable (λ < μ), with a server utilization of 0.67 and predictable performance metrics derived from the model.

2. In a dental clinic, patients arrive randomly at 2 per hour, and the dentist treats them at 5 per hour, fitting an M/M/1 queueing model. Using R's queueing package, define the model with NewInput.MM1() and solve it with QueueingModel() .

Retrieve key metrics: utilization (ρ), average number in queue (Lq), in system (Ls), waiting time in queue (Wq), total time in system (Ws), and throughput.

Aim:

To model an M/M/1 queuing system with an arrival rate (λ) of 2 and a service rate (μ) of 5, and analyse system performance characteristics such as server utilization and average number of customers in the system.

Procedure:

1. *Define the input parameters using NewInput.MM1.*
2. *Compute the model using QueueingModel.*
3. *Extract key performance measures.*

Program:

```
library(queueing)
lambda <- 2
mu <- 5

mm1_input <- NewInput.MM1(lambda = lambda, mu = mu)
mm1_model <- QueueingModel(mm1_input)

summary(mm1_model)

cat("Utilization (rho):", RO(mm1_model), "\n")
cat("Average number in queue (Lq):", Lq(mm1_model), "\n")
cat("Average time in queue (Wq):", Wq(mm1_model), "\n")
cat("Average number in system (Ls):", L(mm1_model), "\n")
cat("Average time in system (Ws):", W(mm1_model), "\n")
cat("Throughput:", Throughput(mm1_model), "\n")
```

Output:

lambda	mu	c	k	m	RO	P_0	L_q	W_q	X	L	W	W_{qq}	L_{qq}
2	*5*	*1*	*NA*	*NA*	*0.4*	*0.6*	*0.267*	*0.133*	*2*	*0.667*	*0.333*	*0.333*	*1.667*

Utilization (rho): 0.4
Average number in queue (Lq): 0.2666667
Average time in queue (Wq): 0.1333333
Average number in system (Ls): 0.6666667
Average time in system (Ws): 0.3333333
Throughput: 2

Result:

The M/M/1 queuing model was successfully established with λ = 2 and μ = 5. The system was found to be stable (λ < μ), with a server utilization of 0.4 and predictable performance metrics derived from the model.

Multi-Server Queue (M/M/c)

3. Model an M/M/c system having 3 servers (c = 3) with Arrival rate (λ = 6) and Service rate (μ = 4).

Aim:

To model an M/M/c queuing system with an arrival rate (λ) of 6 and a service rate (μ) of 4 and number of servers (c) of 3, and analyse system performance characteristics such as server utilization and average number of customers in the system.

Procedure:

1. *Define the input parameters using NewInput.MMC.*
2. *Compute the model using QueueingModel.*
3. *Extract key performance measures.*

Program:

```
library(queueing)

mmc_input <- NewInput.MMC(lambda = 6, mu = 4, c = 3)

mmc_model <- QueueingModel(mmc_input)

summary(mmc_model)
```

Output:

lambda	mu	c	k	m	RO	P_0	L_q	W_q	X	L	W	W_{qq}	L_{qq}
6	4	3	NA	NA	0.5	0.211	0.237	0.039	6	1.737	0.289	0.167	2

Result:

The M/M/3 queuing model was successfully established with λ = 6, μ = 4, and c = 3. The system was stable, with server utilization within acceptable limits. Multiple servers helped minimize queue length and wait time, resulting in efficient customer handling.

4. A car service center has 2 service bays, with cars arriving at a rate of 7 cars per hour. Each bay services cars at a rate of 5 cars per hour.

Find the average number of cars in the system (Ls), the average number of cars in the queue (Lq), the average waiting time in the system (Ws), and the average waiting time in the queue (Wq) using 'queueing' package in R.

Aim:

To model an M/M/c queuing system with an arrival rate (λ) of 7 and a service rate (μ) of 5 and number of servers (c) of 2, and analyse system performance characteristics such as server utilization and average number of customers in the system.

Procedure:

1. *Define the input parameters using NewInput.MMC.*
2. *Compute the model using QueueingModel.*
3. *Extract key performance measures.*

Program:

```
library(queueing)

lambda <- 7
mu <- 5
c <- 2

mmc_input <- NewInput.MMC(lambda = lambda, mu = mu, c = c)
mmc_model <- QueueingModel(mmc_input)

summary(mmc_model)


cat("Utilization (rho):", RO(mmc_model), "\n")
cat("Average number in queue (Lq):", Lq(mmc_model), "\n")
cat("Average time in queue (Wq):", Wq(mmc_model), "\n")
cat("Average number in system (Ls):", L(mmc_model), "\n")
cat("Average time in system (Ws):", W(mmc_model), "\n")
cat("Throughput:", Throughput(mmc_model), "\n")
```

Output:

lambda	mu	c	k	m	RO	P_0	L_q	W_q	X	L	W	W_{qq}	L_{qq}
7	*5*	*2*	*NA*	*NA*	*0.7*	*0.176*	*1.345*	*0.192*	*7*	*2.745*	*0.392*	*0.333*	*3.333*

Utilization (rho): 0.7
Average number in queue (Lq): 1.345098
Average time in queue (Wq): 0.1921569
Average number in system (Ls): 2.745098
Average time in system (Ws): 0.3921569
Throughput: 7

Result:

The M/M/2 queuing model was successfully established with λ = 7, μ = 5, and c = 2. The system was stable, with server utilization within acceptable limits. Multiple servers helped minimize queue length and wait time, resulting in efficient customer handling.

Ex. No. 9	MONTE CARLO SIMULATION – PREDICTING STOCK PRICES USING 'MONTECARLO' PACKAGE IN R
Date: 12.04.2025	

1. Simulate a Single Stock Price

Simulate a single stock's price for 1 year with initial price ($S_0 = 100$), daily return mean ($\mu = 0.001$), volatility ($\sigma = 0.02$), and 252 trading days.

Aim:

To simulate the daily price trajectory of a single stock over 1 year (252 trading days) using Geometric Brownian Motion, given the initial price, mean daily return, and volatility.

Procedure:

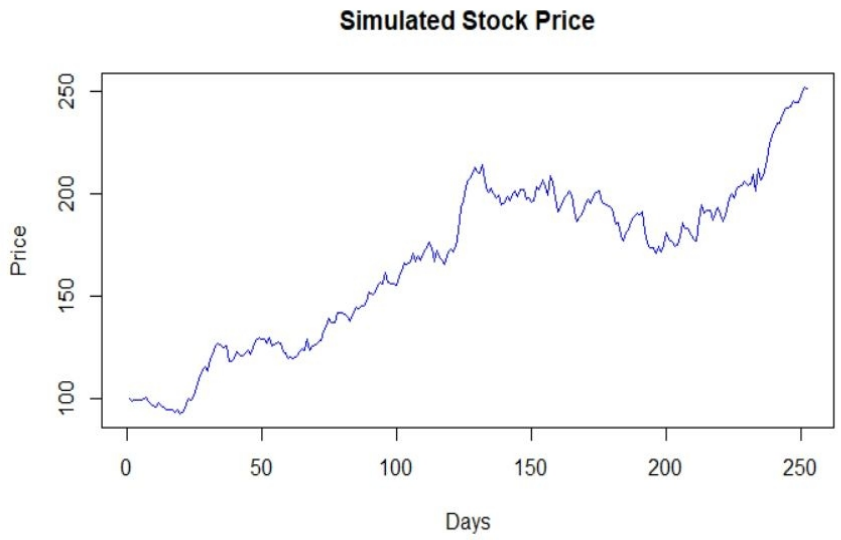
1. *Define a function to simulate daily prices.*
2. *Run the simulation.*
3. *Plot the price trajectory.*

Program:

```
library(MonteCarlo)


simulate_stock <- function(S0, mu, sigma, T) {
    prices <- numeric(T)
    prices[1] <- S0
    for (t in 2:T) {
        prices[t] <- prices[t - 1] * exp(rnorm(1, mean = mu, sd = sigma))
    }
    return(prices)
}
```

```
S0 <- 100

mu <- 0.001

sigma <- 0.02

T <- 252


prices <- simulate_stock(S0, mu, sigma, T)

plot(prices, type = "l", col = "blue", main = "Simulated Stock Price", xlab = "Days",
ylab = "Price")
```

Output:



Result:

The stock price was successfully simulated using the defined parameters, and the resulting price path over 252 trading days was visualized with a line plot.

2. Simulate Multiple Price Paths

Simulate 5 price paths for the stock price, based on the same parameters, and simulate 252 days of trading

Aim:

To simulate and visualize multiple (5) price paths of a single stock over 252 trading days using Geometric Brownian Motion, facilitating comparative analysis of different stochastic outcomes.

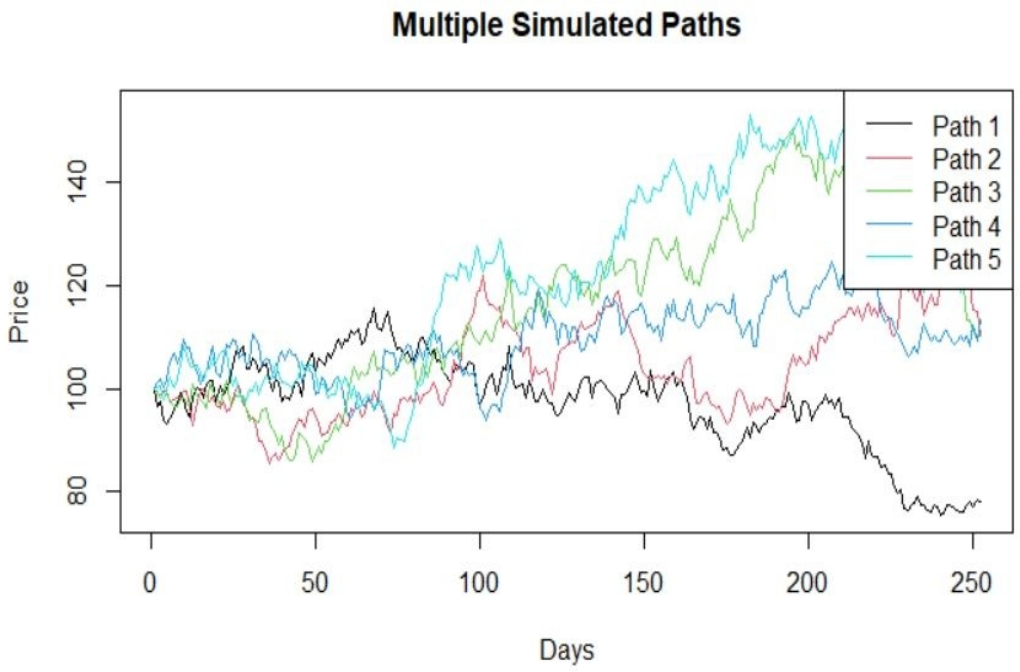
Procedure:

1. *Modify the function to handle multiple simulations.*
2. *Overlay the paths in a single plot.*

Program:

```
library(MonteCarlo)


simulate_multiple <- function(S0, mu, sigma, T, n_paths) {
    paths <- matrix(NA, nrow = T, ncol = n_paths)
    for (i in 1:n_paths) {
        paths[, i] <- simulate_stock(S0, mu, sigma, T)
    }
    return(paths)
}


n_paths <- 5
paths <- simulate_multiple(S0, mu, sigma, T, n_paths)
matplot(paths, type = "l", lty = 1, col = 1:n_paths, main = "Multiple Simulated
Paths", xlab = "Days", ylab = "Price")
legend("topright", legend = paste("Path", 1:n_paths), col = 1:n_paths, lty = 1)
```

Output:



Result:

Five distinct stock price trajectories were successfully simulated and overlaid on a single plot, effectively demonstrating the variability in potential future prices due to market volatility.

3. Simulate Log Returns

Simulate daily log returns and analyze their distribution. The returns should be normally distributed with mean = mu and sd = sigma.

Aim:

To simulate daily log returns and analyse their distribution.

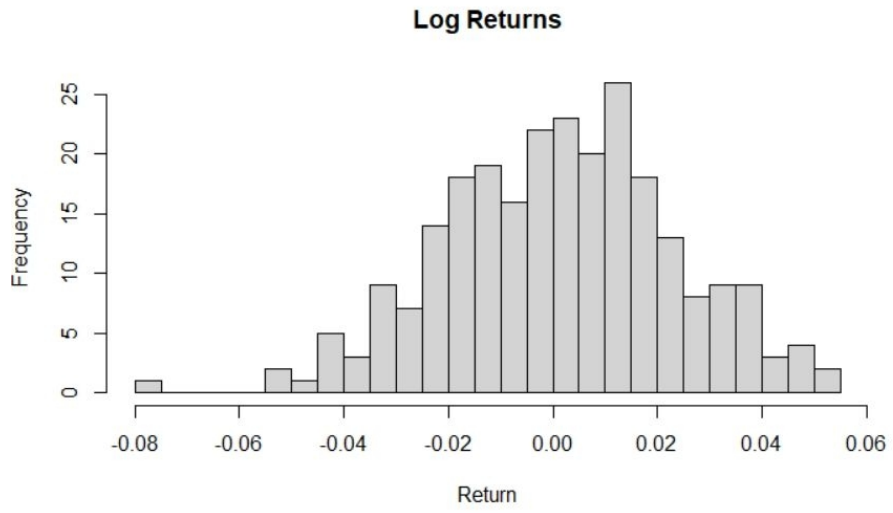
Procedure:

1. *Define a function to simulate log returns.*
2. *Run the simulation.*
3. *Plot the distribution of log returns.*
4. *Calculate and display the mean and standard deviation.*

Program:

```
library(MonteCarlo)

simulate_log_returns <- function(mu, sigma, T) {
    rnorm(T, mean = mu, sd = sigma)
}

log_returns <- simulate_log_returns(mu, sigma, T)
hist(log_returns, breaks = 30, main = "Log Returns", xlab = "Return")

mean_log_return <- mean(log_returns)
sd_log_return <- sd(log_returns)

cat("Mean of log returns: ", mean_log_return, "\n")
cat("Standard deviation of log returns: ", sd_log_return, "\n")
```

Output:



Mean of log returns: 0.002006851
Standard deviation of log returns: 0.01915779

Result:

The simulation successfully generated daily log returns. The histogram displayed a normal distribution of the returns, with the mean approximating the specified μ\muμ and the standard deviation approximating the specified σ\sigmaσ.

4. Value-at-Risk (VaR)

Estimate the 5% Value-at-Risk (VaR) for the portfolio by simulating terminal prices over 1,000 simulations.

Aim:

To estimate the 5% Value-at-Risk (VaR) of a portfolio using Monte Carlo simulation.

Procedure:

1. *Simulate terminal prices of the stock using Monte Carlo.*
2. *Calculate the 5th percentile (quantile) of the simulated terminal prices to estimate VaR.*

Program:

```
library(MonteCarlo)

simulate_terminal <- function(S0, mu, sigma, T, n_sims) {
    z <- rnorm(n_sims)
    ST <- S0 * exp((mu - 0.5 * sigma^2) * T + sigma * sqrt(T) * z)
    return(ST)
}

VaR <- function(S0, mu, sigma, T, n_sims, alpha = 0.05) {
    terminal_prices <- simulate_terminal(S0, mu, sigma, T, n_sims)
    quantile(terminal_prices, alpha)
}
```

```
S0 <- 100
mu <- 0.08
sigma <- 0.2
T <- 1
n_sims <- 10000
result <- VaR(S0, mu, sigma, T, n_sims)
cat("Estimated 5% Value-at-Risk:", result, "\n")
```

Output:

```
Estimated 5% Value-at-Risk: 76.06828
```

Result:

The 5% Value-at-Risk was successfully estimated using the simulated terminal prices. This value indicates the maximum expected loss at the 5% confidence level over the given time period.

5. Sensitivity Analysis

An analyst wants to understand how stock price volatility impacts the expected terminal price after 1 year. Run simulations across different volatility levels and visualize the sensitivity.

Aim:

To evaluate how varying volatility affects the mean terminal price of a stock using sensitivity analysis.

Procedure:

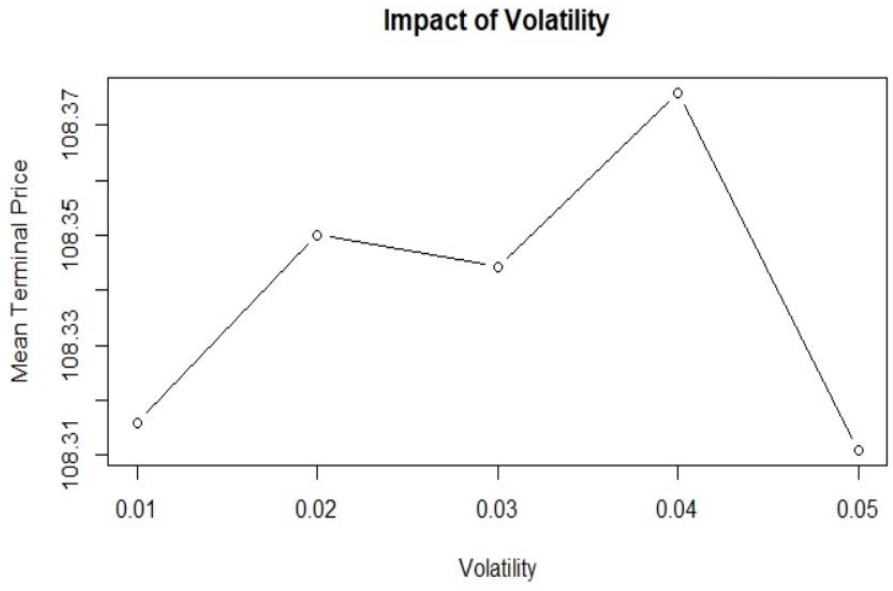
1. *Define a range of volatility values.*
2. *For each volatility value, simulate terminal stock prices using Monte Carlo simulation.*
3. *Compute the mean of the terminal prices for each volatility level.*
4. *Plot the relationship between volatility and mean terminal price.*

Program:

```
simulate_stock <- function(S0, mu, sigma, T) {
        prices <- numeric(T)
        prices[1] <- S0
        for (t in 2:T) {
                prices[t] <- prices[t - 1] * exp(rnorm(1, mean = mu, sd = sigma))
        }
        return(prices)
}

simulate_terminal <- function(S0, mu, sigma, T, n_sims) {
        terminal_prices <- numeric(n_sims)
        for (i in 1:n_sims) {
                prices <- simulate_stock(S0, mu, sigma, T)
                terminal_prices[i] <- prices[T]
        }
        return(terminal_prices)
}
```

```
S0 <- 100
mu <- 0.001
T <- 252
n_sims <- 1000

volatilities <- seq(0.01, 0.05, by = 0.01)
results <- sapply(volatilities, function(sigma) mean(simulate_terminal(S0, mu,
sigma, T, n_sims)))

plot(volatilities, results, type = "b", main = "Impact of Volatility",
    xlab = "Volatility", ylab = "Mean Terminal Price")
```

Output:



Result:

The plot shows how the average terminal price changes as volatility increases. This helps in understanding the sensitivity of the stock's future price to changes in market volatility.

6. Portfolio Optimization

Simulate portfolio returns for 3 assets with different return and volatility parameters, and calculate optimal allocation

Aim:

To simulate portfolio returns for three assets using Monte Carlo methods and analyze expected return and risk to evaluate the effectiveness of a chosen asset allocation.

Procedure:

1. *Simulate individual stock price paths for each asset using their respective mean returns (mus) and volatilities (sigmas).*
2. *Compute the portfolio value at each time step by weighting the asset prices according to a given allocation (weights).*
3. *Calculate the portfolio return for each simulation as the percentage change from the starting value.*
4. *Repeat the process for a specified number of simulations (n_sims) to generate a distribution of portfolio returns.*
5. *Analyze the expected return (mean) and risk (standard deviation) of the portfolio.*

Program:

```
library(MonteCarlo)

simulate_stock <- function(S0, mu, sigma, T) {
    z <- rnorm(1)
    ST <- S0 * exp((mu - 0.5 * sigma^2) * T + sigma * sqrt(T) * z)
    return(c(S0, ST))
}
```

```
simulate_portfolio <- function(weights, mus, sigmas, T, n_sims) {
    portfolio_returns <- replicate(n_sims, {
        prices <- sapply(1:length(mus), function(i) simulate_stock(100,
            mus[i], sigmas[i], T))
        portfolio_value <- rowSums(prices * weights)
        (portfolio_value[T] - portfolio_value[1]) / portfolio_value[1]
        })
    return(portfolio_returns)
}

weights <- c(0.4, 0.4, 0.2)
mus <- c(0.001, 0.002, 0.0008)
sigmas <- c(0.02, 0.03, 0.015)
T <- 1
n_sims <- 1000
portfolio_returns <- simulate_portfolio(weights, mus, sigmas, T, n_sims)

cat("Expected Return:", mean(portfolio_returns), "\n")
cat("Portfolio Risk (SD):", sd(portfolio_returns), "\n")
```

Output:

```
Expected Return:      -0.2624337
Portfolio Risk (SD):    0.1764875
```

Result:

The Markov Chain was successfully defined, steady-state probabilities were computed, and the manufacturing process was simulated for 20 steps starting from the Defective state.

Ex. No. 10	READING, WRITING DATA IN R AND WORKING WITH INBUILT DATA SETS IN R
Date: 19.04.2025	

1. Reading a CSV File

Read a CSV file containing data on student scores and display the first 5 rows.

Aim:

To load and preview data from a CSV file containing student scores using R.

Procedure:

1. *Use read.csv() to load the file.*
2. *Use head() to display the first 5 rows.*

Program:

```
data <- read.csv("D:/IV/R LAB/trades_result.csv")
head(data, 5)
```

Output:

	time_ref <int>	account <chr>	code <chr>	country_code <chr>	product_type <chr>	value <dbl>	status <chr>
1	202412	Exports	00	AD	Goods	2581	F
2	202412	Exports	00	AE	Goods	323384662	F
3	202412	Exports	00	AG	Goods	266255	F
4	202412	Exports	00	AI	Goods	11760	F
5	202412	Exports	00	AL	Goods	639168	F

5 rows

Result:

The output displays the top 5 rows of the dataset, allowing for a quick inspection of the structure and content of the student scores data.

2. Writing a Data Frame to a CSV File

Create a data frame of employee details and save it as a CSV file.

Aim:

To create a data frame containing the employee details and save it as a CSV file using R.

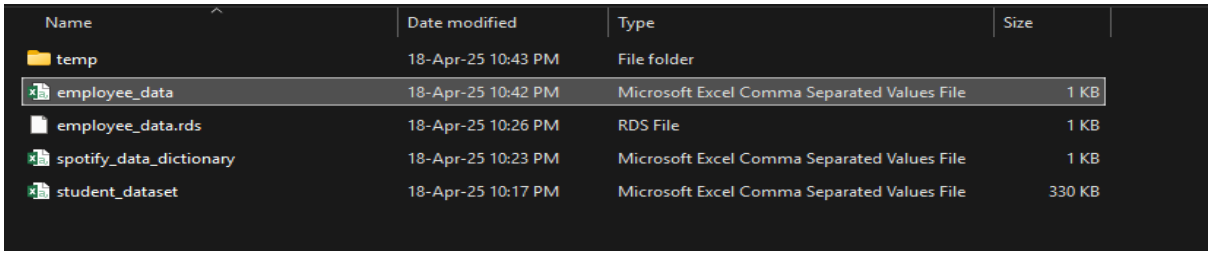
Procedure:

1. *Create a data frame using data.frame().*
2. *Save it using write.csv().*

Program:

```
employee_data <- data.frame(
  ID = c(1, 2, 3, 4),
  Name = c("John", "Alice", "Bob", "Clara"),
  Age = c(35, 40, 36, 55),
  Salary = c(50000, 60000, 55000, 58000)
)

write.csv(employee_data, "D:/IV/R LAB/employee_data.csv", row.names = FALSE)
print("File written successfully")

print("Reading the employee_data")
read.csv("D:/IV/R LAB/employee_data.csv")
```

Output:



Name	Date modified	Type	Size
temp	18-Apr-25 10:43 PM	File folder	
employee_data	18-Apr-25 10:42 PM	Microsoft Excel Comma Separated Values File	1 KB
employee_data.rds	18-Apr-25 10:26 PM	RDS File	1 KB
spotify_data_dictionary	18-Apr-25 10:23 PM	Microsoft Excel Comma Separated Values File	1 KB
student_dataset	18-Apr-25 10:17 PM	Microsoft Excel Comma Separated Values File	330 KB

"File written successfully"

“Reading the employee_data”

ID <int>	Name <chr>	Age <int>	Salary <int>
1	John	35	50000
2	Alice	40	60000
3	Bob	36	55000
4	Clara	55	58000

Result:

The file employee_data.csv is successfully created in the working directory, containing the employee information in tabular format.

3. Summary Statistics of an Inbuilt Dataset

Generate summary statistics for the mtcars dataset

Aim:

To generate and display summary statistics for the mtcars dataset in R.

Procedure:

1. *Load the mtcars dataset using the data() function.*
2. *Use the summary() function to obtain descriptive statistics for each variable.*

Program:

```
data(mtcars)

summary(mtcars)
```

Output:

```
      mpg             cyl             disp             hp             drat             wt       
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0   Min.   :2.760   Min.   :1.513  
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5   1st Qu.:3.080   1st Qu.:2.581  
 Median :19.20   Median :6.000   Median :196.3   Median :123.0   Median :3.695   Median :3.325  
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7   Mean   :3.597   Mean   :3.217  
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0   3rd Qu.:3.920   3rd Qu.:3.610  
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0   Max.   :4.930   Max.   :5.424  
      qsec             vs              am              gear            carb      
 Min.   :14.50   Min.   :0.0000   Min.   :0.0000   Min.   :3.000   Min.   :1.000  
 1st Qu.:16.89   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000  
 Median :17.71   Median :0.0000   Median :0.0000   Median :4.000   Median :2.000  
 Mean   :17.85   Mean   :0.4375   Mean   :0.4062   Mean   :3.688   Mean   :2.812  
 3rd Qu.:18.90   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000  
 Max.   :22.90   Max.   :1.0000   Max.   :1.0000   Max.   :5.000   Max.   :8.000  
```

Result:

Summary statistics—including minimum, 1st quartile, median, mean, 3rd quartile, and maximum—are successfully displayed for each numeric column in the mtcars dataset.

4. Filtering Data from a Dataset

Filter rows in the mtcars dataset where mpg > 20.

Aim:

To filter rows in the mtcars dataset where the miles per gallon (mpg) is greater than 20.

Procedure:

1. *Load the mtcars dataset.*
2. *Use the subset() function with the condition mpg > 20.*

Program:

```
data(mtcars)
filtered_data <- subset(mtcars, mpg > 20)
print(filtered_data)
```

Output:

	mpg <dbl>	cyl <dbl>	disp <dbl>	hp <dbl>	drat <dbl>	wt <dbl>	qsec <dbl>	vs <dbl>	am <dbl>
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0

1-10 of 14 rows | 1-10 of 11 columns

Previous 1 2 Next

	mpg <dbl>	cyl <dbl>	disp <dbl>	hp <dbl>	drat <dbl>	wt <dbl>	qsec <dbl>	vs <dbl>	am <dbl>
Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1
Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1

11-14 of 14 rows | 1-10 of 11 columns

Result:

A filtered dataset displaying only the rows where mpg > 20 is successfully printed.

5. Saving and Loading Data in RDS Format

Save a data frame as an .rds file and load it back.

Aim:

To save a data frame in .rds format and load it back for future use.

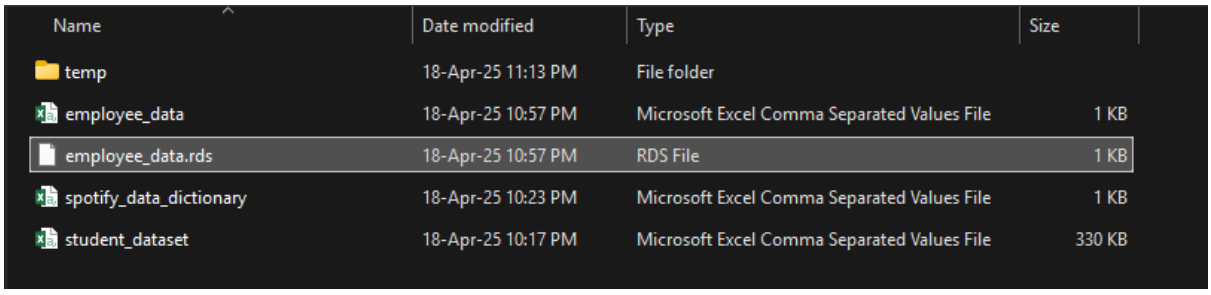
Procedure:

1. *Use saveRDS() to store the data frame in an .rds file.*
2. *Use readRDS() to read the saved data from the file and load it into a variable.*

Program:

```
data <- read.csv("D:/IV/R LAB/employee_data.csv")

saveRDS(data, "D:/IV/R LAB/employee_data.rds")
loaded_data <- readRDS("D:/IV/R LAB/employee_data.rds")


print(loaded_data)

str(loaded_data)
```

Output:



Name	Date modified	Type	Size
temp	18-Apr-25 11:13 PM	File folder	
employee_data	18-Apr-25 10:57 PM	Microsoft Excel Comma Separated Values File	1 KB
employee_data.rds	18-Apr-25 10:57 PM	RDS File	1 KB
spotify_data_dictionary	18-Apr-25 10:23 PM	Microsoft Excel Comma Separated Values File	1 KB
student_dataset	18-Apr-25 10:17 PM	Microsoft Excel Comma Separated Values File	330 KB

ID <int>	Name <chr>	Age <int>	Salary <int>
1	John	35	50000
2	Alice	40	60000
3	Bob	36	55000
4	Clara	55	58000

'data.frame': 4 obs. of 4 variables:
$ ID : int 1 2 3 4
$ Name : chr "John" "Alice" "Bob" "Clara"
$ Age : int 35 40 36 55
$ Salary: int 50000 60000 55000 58000

Result:

The employee_data data frame is successfully saved as employee_data.rds and later retrieved into loaded_data.

6. Merge Two Data Frames

Merge two data frames by a common column.

Aim:

To combine two data frames using a common column (key).

Procedure:

1. *Create two data frames with a shared column (ID in this case).*
2. *Use the merge() function with the by parameter to join the data frames on the common column.*

Program:

```
df1 <- data.frame(ID = c(1, 2, 3), Name = c("John", "Alice", "Bob"))
df2 <- data.frame(ID = c(1, 2, 3), Department = c("HR", "IT", "Finance"))
merged_data <- merge(df1, df2, by = "ID")
print(merged_data)
```

Output:

ID <dbl>	Name <chr>	Department <chr>
1	John	HR
2	Alice	IT
3	Bob	Finance

3 rows

Result:

The two data frames are successfully merged into one, aligning rows based on the matching ID values. The resulting data frame includes columns from both original frames.