

# **R Programming Lab Practical File**

## **Practical File**

**Submitted for the partial fulfillment of the degree of**

## **Bachelor of Technology**

**In**

## **Mathematics & Computing**

**Submitted By**

**Shivam Singh**

**0901MC211059**

**UNDER THE SUPERVISION AND GUIDANCE OF**

**Ms. Manali Singh**

**Assistant Professor**

**Department of Engineering Mathematics & Computing**



**MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR (M.P.), INDIA**  
**माधव प्रौद्योगिकी एवं विज्ञान संस्थान, ग्वालियर (म.प्र.), भारत**

**A GOVT. AIDED UGC AUTONOMOUS INSTITUTE, AFFILIATED TO R.G.P.V. BHOPAL (M.P.), INDIA**

**NAAC ACCREDITED WITH A++ GRADE**

**DEEMED TO BE UNIVERSITY**

**July – Dec 2024**

## ABSTRACT

This lab file comprehensively explores the realm of **R programming**, a powerful and versatile tool renowned for its capabilities in statistical computing, data analysis, and graphical representation. As one of the most widely used languages in the fields of data science and analytics, R empowers users with an extensive suite of tools for processing and interpreting data, making it indispensable for statisticians, data scientists, and researchers.

The document begins with an in-depth theoretical overview, outlining the foundational aspects of R programming. This includes its history, unique features, advantages, disadvantages, and a detailed comparison with other programming languages like Python and MATLAB. Special emphasis is placed on the installation process and the practical benefits of adopting R for both academic and industrial applications. Real-world scenarios such as predictive modeling, healthcare analytics, and financial forecasting are highlighted to illustrate the language's relevance and practicality.

Following the theoretical section, this lab file presents **eight carefully designed R programs**, each tailored to demonstrate core functionalities and practical use cases of the language. These programs range from basic operations, such as sorting and sampling, to more advanced statistical tasks like generating random numbers from standard distributions. Each program is accompanied by clear and concise comments, step-by-step explanations of the logic and syntax, and corresponding outputs to ensure comprehensive understanding.

This lab file not only serves as a technical resource for understanding the basics of R programming but also aims to bridge the gap between theoretical concepts and real-world applications. It is designed to provide students and practitioners with the skills needed to effectively utilize R in diverse domains, including **data science, bioinformatics, academia, and business analytics**.

By combining theoretical depth with hands-on practice, this document aspires to foster a deeper appreciation for R programming. It demonstrates how R can be leveraged to tackle complex computational problems, analyze large datasets, and create high-quality visualizations. This blend of theory and practice makes the lab file an invaluable resource for both learning and applying R programming in practical scenarios.

## ACKNOWLEDGEMENT

I would like to extend my heartfelt gratitude to **Assistant Prof. Ms. Manali Singh** of the **Computer Science and Business Department** for her unwavering support, guidance, and invaluable insights throughout the preparation of this lab file. Her exceptional mentorship and expertise in R programming and statistical computing have been instrumental in shaping the quality and technical accuracy of this document. Her constructive feedback and timely suggestions helped refine the theoretical and practical components, ensuring a comprehensive understanding of the subject.

I am also profoundly grateful to my peers and collaborators for their cooperation and encouragement during this process. Their collective inputs and shared knowledge contributed significantly to the successful completion of this lab file.

Furthermore, this work reflects the academic rigor and collaborative environment fostered by the **Computer Science and Business Department at Madhav Institute of Technology and Science (MITS)**. The resources and support provided by the department, including access to computing facilities, technical documentation, and research materials, have been pivotal in the seamless execution of this project.

Lastly, I dedicate this effort to my family for their unwavering support and motivation, which have been my constant source of inspiration. This acknowledgment also extends to everyone who has indirectly contributed to this endeavour by providing a conducive learning environment.

---

**Shivam Singh**  
**0901MC211059**

# Table of Contents

## **1. Theoretical Background of R Programming**

1.1 What is R Programming?

1.2 Features of R Programming

1.3 Advantages and Disadvantages of R Programming

1.4 Real-Life Applications of R Programming

1.5 Why and How is R Different from Other Programming Languages?

## **2. Practical Implementation of R Programs**

2.1 R Program to Sample from a Population

2.2 R Program to Find Minimum and Maximum

2.3 R Program to Sort a Vector

2.4 R "Hello World" Program

2.5 R Program to Add Two Vectors

2.6 R Program to Calculate Sum, Mean, and Product of a Vector

2.7 R Program to Take Input from the User

2.8 R Program to Generate Random Numbers from Standard Distributions

# R Programming

## 1.1 What is R Programming?

R is a versatile and powerful programming language specifically designed for statistical computing, data analysis, and graphical representation. Originating from the S language developed at Bell Laboratories, R is an open-source language maintained by the R Foundation for Statistical Computing. Its popularity stems from its simplicity, flexibility, and the extensive range of packages available for data manipulation, modeling, and visualization.



**Fig 1. R Programming**

## 1.2 Features of R Programming

1. **Comprehensive Statistical Analysis:** Includes functions for linear and nonlinear modeling, time-series analysis, and more.
2. **Data Visualization:** Offers libraries like ggplot2 for creating high-quality plots.
3. **Open-Source:** Free and accessible to everyone.
4. **Cross-Platform Compatibility:** Runs seamlessly on Windows, macOS, and Linux.
5. **Extensibility:** Users can create custom functions or use pre-built packages from CRAN.
6. **Community Support:** A vast, active community ensures constant innovation and help for newcomers.

## 1.3 Advantages and Disadvantages of R Programming

### 1.3.1 Advantages of R Programming

1. **Extensive Libraries for Data Analysis:**

R offers a vast collection of packages (e.g., dplyr, tidyr, and caret) for data manipulation, statistical analysis, and machine learning, simplifying complex workflows.

2. **Strong Visualization Capabilities:**

Tools like ggplot2, lattice, and Shiny enable the creation of interactive, customizable, and publication-quality visualizations. These are especially useful for conveying insights in data-driven environments.

3. **Cross-Platform Support:**

R is compatible with multiple operating systems, including Windows, macOS, and Linux, ensuring flexibility in diverse work environments.

4. **Active Community and Resources:**

With a vast, active user base, R's community provides extensive support through forums, tutorials, and open-source contributions. This helps beginners and advanced users alike to resolve issues quickly.

5. **Integration with Other Languages:**

R can seamlessly integrate with Python, Java, C++, and databases like MySQL or MongoDB, enabling its use in hybrid environments and large-scale applications.

6. **Customizability:**

R allows users to create custom packages, functions, and scripts, tailoring solutions to specific needs.

7. **Open Source:**

Being free and open-source, R is accessible to individuals, startups, and enterprises without licensing costs.

8. **Wide Range of Statistical Techniques:**

R supports a comprehensive range of methods, such as regression models, time-series analysis, hypothesis testing, and clustering.

### 1.3.2 Disadvantages of R Programming

1. **Steep Learning Curve:**

For beginners without a statistical background, R's syntax and environment can be challenging to learn compared to more intuitive languages like Python.

2. **Memory-Intensive:**

R processes all objects in memory (RAM), which can limit performance when handling large datasets.

### 3. **Slower Execution:**

As an interpreted language, R is slower than compiled languages like C++ or Java when performing intensive computations.

### 4. **Limited Support for GUIs:**

While powerful for scripting and analysis, R lacks robust graphical user interfaces for developing standalone applications.

### 5. **Dependency Management:**

Managing multiple package versions and dependencies can become cumbersome, especially for collaborative projects or production-grade environments.

### 6. **Debugging Challenges:**

Errors in R are sometimes cryptic and harder to debug, requiring users to rely on online forums or extensive documentation.

## 1.4 Real-Life Applications of R Programming

### 1. **Data Science and Machine Learning:**

R is a cornerstone in data science, used for predictive modelling, clustering, and recommendation systems. Popular libraries like caret and mlr simplify machine learning workflows.

*Example:* Retail companies use R to analyse customer purchase behaviours and recommend products.

### 2. **Bioinformatics:**

R is extensively used to analyse genomic and proteomic data, helping researchers identify patterns and make breakthroughs in personalized medicine.

*Example:* Analysing DNA sequences or predicting disease markers.

### 3. **Finance:**

R is a popular tool for financial modelling, portfolio optimization, and risk management.

*Example:* Investment firms use R to simulate stock price behaviours and manage risks in real-time trading environments.

### 4. **Healthcare:**

Hospitals and research institutions use R to study patient data, predict treatment outcomes, and model the spread of diseases.

*Example:* Modelling the spread of infectious diseases like COVID-19 to optimize containment strategies.

### 5. **Academia and Research:**

R is widely adopted in academic settings for statistical analysis, hypothesis testing, and research publication.

*Example:* Writing research papers with visualizations generated directly in R.

6. **Social Media Analytics:**

Companies use R to perform sentiment analysis and study user behavior on platforms like Twitter and Instagram.

*Example:* Analyzing customer sentiment about a brand during a marketing campaign.

7. **Business Analytics:**

Organizations leverage R for sales forecasting, customer segmentation, and operational efficiency improvement.

*Example:* Predicting inventory requirements for e-commerce platforms using time-series analysis.

8. **Environmental Science:**

R helps researchers model climate change patterns, pollution data, and ecological behaviors.

*Example:* Modeling carbon emissions and their effect on global warming.

## 1.5 Why and How is R Different from Other Programming Languages?

R stands out for its focus on statistical analysis and data visualization. Unlike Python, which is general-purpose, R is specifically designed for statistical tasks. While Python offers faster execution, R's extensive statistical packages make it an unparalleled choice for statisticians.

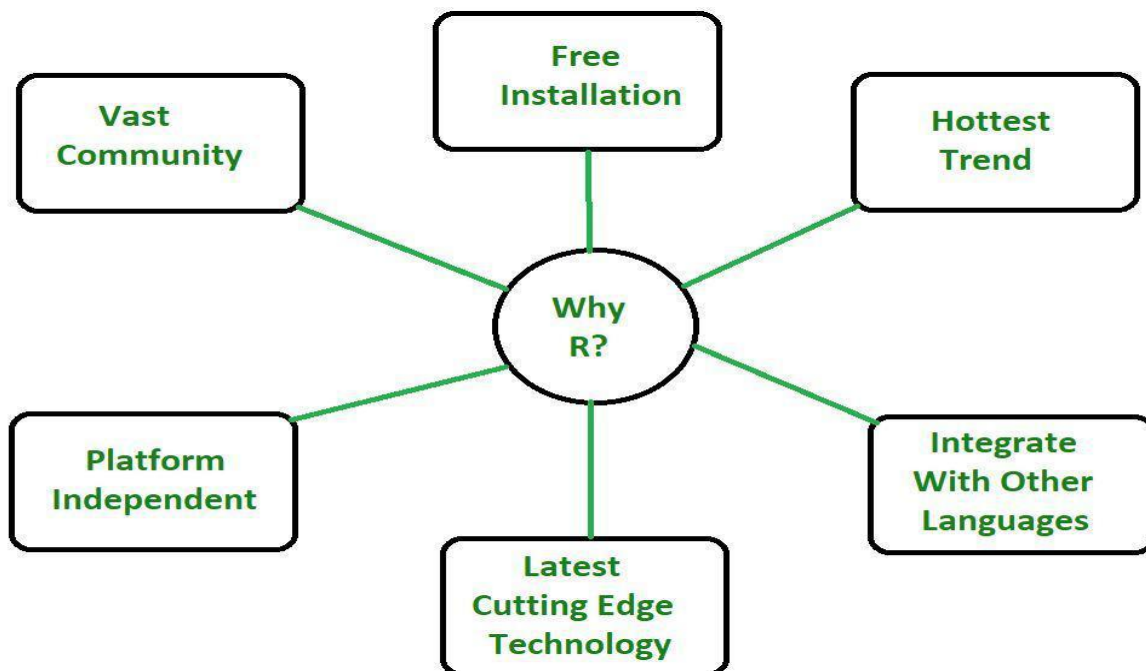


Fig 2. Why R?



## Practical Implementation of R Programs

### 2.1 R Program to Sample from a Population

```
main.r  [Icons] [Share] [Run]

1 # Program to generate a random sample
2 # Generate a random sample of 10 elements from a population of 1
   to 100
3 sample_data <- sample(1:100, 10)
4 print(sample_data)
5
```

**Explanation:** This program uses the `sample()` function to randomly select 10 elements from the specified range

```
Output  [Clear]

[1] 58 98 61 35 80  1 83 73 63 32

=== Code Execution Successful ===
```

### 2.2 R Program to Find Minimum and Maximum

```
main.r  [Icons] [Share] [Run]

1 # Program to find the minimum and maximum of a vector
2 vector <- c(23, 45, 12, 67, 34)
3 min_val <- min(vector)
4 max_val <- max(vector)
5 print(paste("Minimum:", min_val))
6 print(paste("Maximum:", max_val))
7
```

**Explanation:** This program uses `min()` and `max()` functions to identify the smallest and largest values in the vector.

**Output**



Clear


```
[1] "Minimum: 12"
[1] "Maximum: 67"

=== Code Execution Successful ===
```

### 2.3 R Program to Sort a Vector

main.r



 Share

Run

```
1 # Program to sort a vector in ascending order
2 unsorted_vector <- c(34, 12, 56, 78, 23)
3 sorted_vector <- sort(unsorted_vector)
4 print(sorted_vector)
5 |
```

**Explanation:** The `sort()` function arranges the vector in ascending order.

**Output**

Clear

```
[1] 12 23 34 56 78

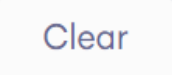
=== Code Execution Successful ===
```

## 2.4 R "Hello World" Program

main.r    Share 

```
1 # Program to display Hello World
2 print("Hello World")
3
```





**Explanation:** A simple program to display the text "Hello World" on the console.

Output 

```
[1] "Hello World"
```

=== Code Execution Successful ===

## 2.5 R Program to Add Two Vectors

main.r    Share 

```
1 # Program to add two vectors
2 vector1 <- c(1, 2, 3)
3 vector2 <- c(4, 5, 6)
4 sum_vector <- vector1 + vector2
5 print(sum_vector)
6
```

**Explanation:** Adds corresponding elements of two vectors.

Output

Clear

```
[1] 5 7 9
```

=== Code Execution Successful ===

## 2.6 R Program to Calculate Sum, Mean, and Product of a Vector

main.r



Share

Run

```
1 # Program to compute sum, mean, and product
2 numbers <- c(10, 20, 30)
3 sum_result <- sum(numbers)
4 mean_result <- mean(numbers)
5 product_result <- prod(numbers)
6 print(paste("Sum:", sum_result))
7 print(paste("Mean:", mean_result))
8 print(paste("Product:", product_result))
9
```

**Explanation:** Demonstrates arithmetic operations on a vector.





Output

Clear

```
[1] "Sum: 60"
[1] "Mean: 20"
[1] "Product: 6000"
```


=== Code Execution Successful ===

## 2.7 R Program to Take Input from the User

```
main.r    Share 
```

```
1 # Program to take user input and display it
2 name <- readline(prompt = "Enter your name: ")
3 print(paste("Hello", name))
4
```





**Explanation:** Uses `readline()` to take input from the user and `paste()` to concatenate strings.

```
Output 
```

```
Enter your name: Shivam
[1] "Hello Shivam"

=== Code Execution Successful ===
```

## 2.8 R Program to Generate Random Numbers

```
main.r    Share 
```

```
1 # Program to generate random numbers from a normal distribution
2 random_numbers <- rnorm(10, mean = 0, sd = 1)
3 print(random_numbers)
4
```

**Explanation:** Generates 10 random numbers from a normal distribution with mean 0 and standard deviation 1.

## Output

[Clear](#)

```
[1]  1.90197543  0.60432840 -0.43052983  0.20737232 -0.69428424 -0
     .02600149
```

```
[7] -0.18020607  0.09644251  0.77221445  0.83279944
```

=== Code Execution Successful ===

## Conclusion

In this lab file, we have explored the fundamentals and practical applications of R programming. The document began with a detailed theoretical overview, highlighting R's strengths in statistical computing and data visualization, along with its limitations and real-world applications. The practical section provided eight diverse programs demonstrating R's versatility and efficiency in handling data and performing computations.

Through this lab work, the significance of R as a tool for data science, analytics, and research became evident. Its extensive libraries, strong community support, and adaptability to different domains make it a preferred choice for statisticians and data scientists. By combining theoretical concepts with hands-on programming, this lab file serves as a valuable resource for beginners and practitioners aiming to utilize R effectively in their academic and professional pursuits.

This experience underscores the importance of learning programming languages tailored for specific domains, such as R for statistical and data-driven analysis. It is hoped that this document will act as a guide for leveraging R to solve real-world problems and inspire further exploration into the field of statistical programming.

# References

## 1. R Documentation and Official Manuals

- R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. <https://cran.r-project.org/manuals.html>

## 2. Books and Guides

- Wickham, H., & Golemund, G. (2017). *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. O'Reilly Media.
- Kabacoff, R. (2015). *R in Action: Data Analysis and Graphics with R*. Manning Publications.

## 3. Online Learning Resources

- CRAN (Comprehensive R Archive Network): <https://cran.r-project.org/>
- TutorialsPoint R Programming Guide: <https://www.tutorialspoint.com/r/index.htm>

## 4. Research Papers

- Peng, R. D. (2008). *A Practical Introduction to the R Programming Language*. *The American Statistician*, 62(2), 138-142.

## 5. Communities and Forums

- Stack Overflow R Community: <https://stackoverflow.com/questions/tagged/r>
- R-bloggers: <https://www.r-bloggers.com/>